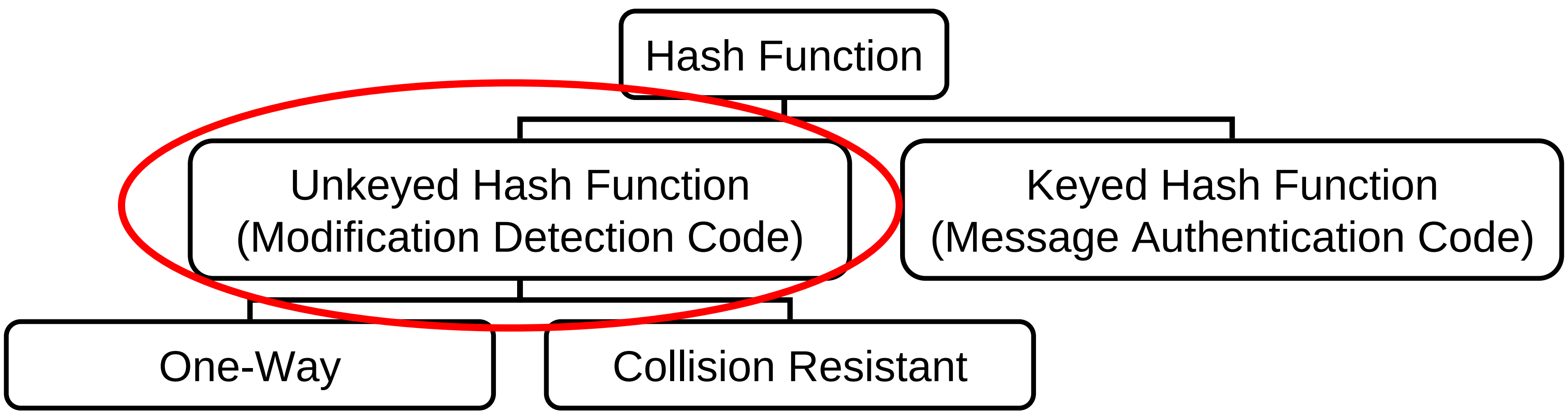
Cryptographic Hash Functions

William R. Speirs

What is a hash function?

- <u>Compression</u>: A function that maps arbitrarily long binary strings to fixed length binary strings
- <u>Ease of Computation</u>: Given a hash function and an input it should be easy to calculate the output
- Often used to create a hash table where the hash function computes the index into the table

Types of Cryptographic Hash Functions



Math Background

- Domain & Range: $y = f(x)$, $y \in Y$, $x \in X$
  - Y is called the range of f
  - X is called the domain of f
  - y is the image of x, and x is the preimage of y
- There are $|Y|^{|X|}$ functions from X to Y
- If $|X| > |Y|$ then $\exists\, x_1, x_2 \in X$ such that $f(x_1) = f(x_2)$

Properties of a Hash Function

- Preimage Resistance (One Way): For essentially all pre-specified outputs, it is computationally infeasible to find any input which hashes to that output.
- Second Preimage Resistance (Weak Col. Res.): It is computationally infeasible to find any second input which has the same output as any specified input.
- Collision Resistance (Strong Col. Res.): It is computationally infeasible to find any two distinct inputs which hash to the same output.

Game Definitions

- An attacker is provided information and must find other information that meets certain criteria
- Preimage Resistance
  - Given: $H(M)$
  - Find: M
- Second Preimage Resistance
  - Given: $H(M)$ and M
  - Find: M' such that $H(M) = H(M')$ and $M \neq M'$
- Collision Resistance
  - Given: *nothing*
  - Find: M and M' such that $H(M) = H(M')$ and $M \neq M'$

Applications for Hash Functions

- Data Integrity
  - Downloading of large files often include the MD5 digest to verify the integrity of the file
- Proof of Ownership
  - Publish the digest of a document describing a patent idea in the New York Times (“All the news that is fit to print”)
- Digital Signatures
  - Digitally sign the digest of a message instead of the message to save computational time

Building Secure Hash Functions

- Merkle-Damgård Construction
  - Defines a method for padding a message
  - Creates a hash function from a fixed input size compression function
    - A compression function *g* takes a fixed size binary string as input and creates a smaller fixed size binary string
  - If the compression function is preimage resistant and collision resistant the hash function is preimage resistant and collision resistant
- Other Constructions
  - HAIFA, EMD, RMX, Dynamic Construction

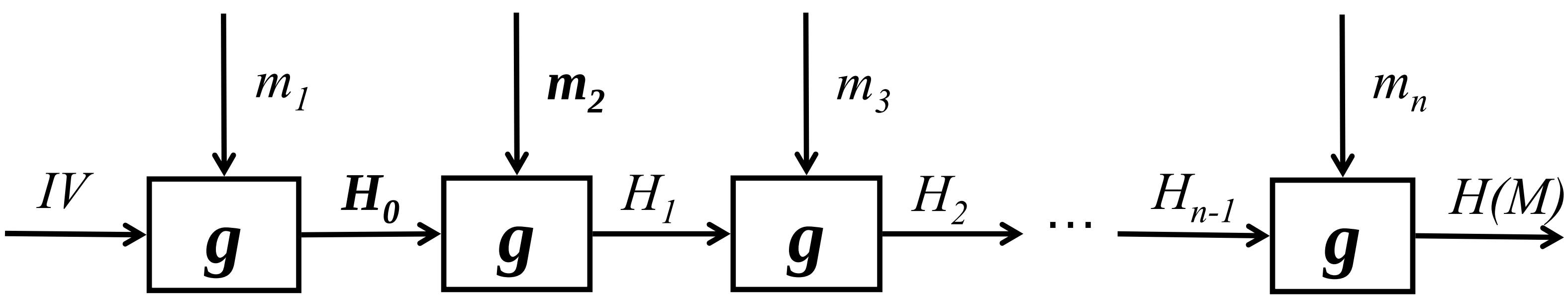
The Merkle-Damgård Padding

- Append a single 1 bit to the end of the message
- Append as many 0 bits to the end of the message as is required to make the message a multiple of the input size of the compression function, minus 64 bits
- Append the bit length of the message as a 64-bit integer to the end of the message

Message	1	00…00	Size of Message (64 bits)

The Merkle-Damgård Construction

- Construction, where $M = m_1 || m_2 || \ldots || m_n$
- $H(M) :=$
  - $H_0 = g(IV, m_1)$
  - $H_i = g(H_{i-1}, m_i)$ for $i = 2, 3, \ldots n$
  - $H(M) = H_n$



Notation

- For both MD5 and SHA-1 all variables are 32-bit quantities and all operations are bitwise

$\neg$ – Logical NOT

$\wedge$ – Logical AND

$\vee$ – Logical OR

$\oplus$ – Logical Exclusive OR

$\boxplus$ – Addition mod 32-bit

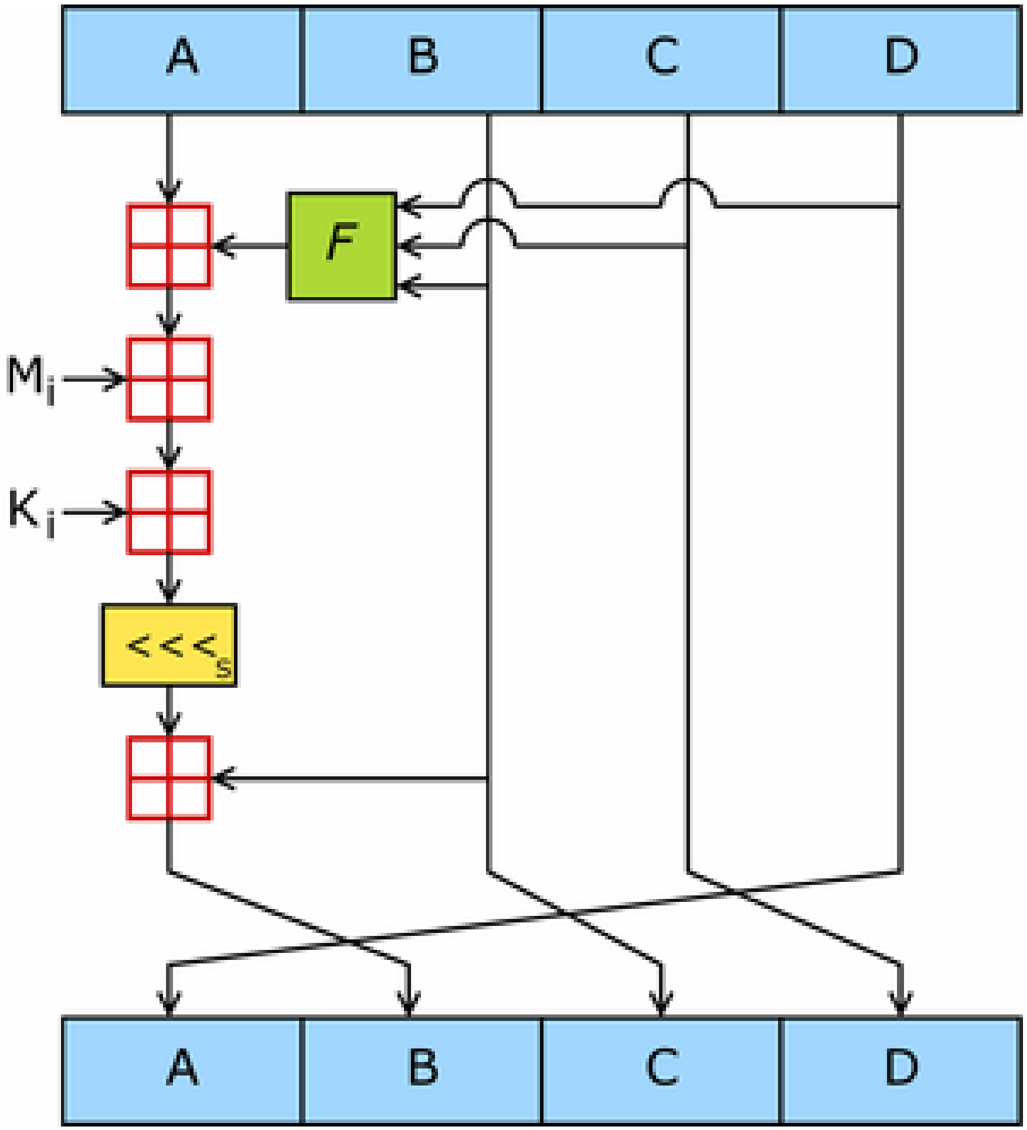
The MD5 Function

- Specifies the compression function to use
- Processes 512-bit message blocks
- Produces a 128-bit digest
- Is derived from MD4 to address security concerns
- Created by Ronald Rivest and described in RFC 1321

The MD5 Compression Function

- Each iteration consists of 4 rounds of 16 steps each, each step processing 32-bits of the message
- Each round processes all 512 of the input bits
- 4 round functions, 1 per round: F(X, Y, Z)
- 4 internal 32-bit registers store the current state of the algorithm
- The internal registers are initialized to fixed constants

An MD5 Step



- A – D are the 32-bit internal registers
- F is 1 of 4 functions

$$
\begin{aligned}
F_1 &= (X \wedge Y) \vee (\neg X \wedge Z) \\
F_2 &= (X \wedge Z) \vee (Y \wedge \neg Z) \\
F_3 &= X \oplus Y \oplus Z \\
F_4 &= Y \oplus (X \vee \neg Z)
\end{aligned}
$$

- M_i is the i^{th} message block
- K_i is the i^{th} round constant
- <<<$_s$ is an s-bit rotation to the left

The SHA-1 Function

- Processes 512-bit message blocks
- Produces a 160-bit digest
- Is also derived from MD4
- A slight modification of SHA-0 to fix a "technical error"
- Created by NIST with the aid of the NSA
- Defined in FIPS PUB 180-2

The SHA-1 Compression Function

- Message expansion round expanding 16 32-bit message words to 80 32-bit message words
- Consists of 4 rounds of 20 steps each, each step processing 32-bits of the expanded message
- Each round processes 20 of the 80 expanded words
- 3 round functions, 1 used twice
- 5 internal 32-bit registers store the current state of the algorithm
- The internal registers are initialized to fixed constants

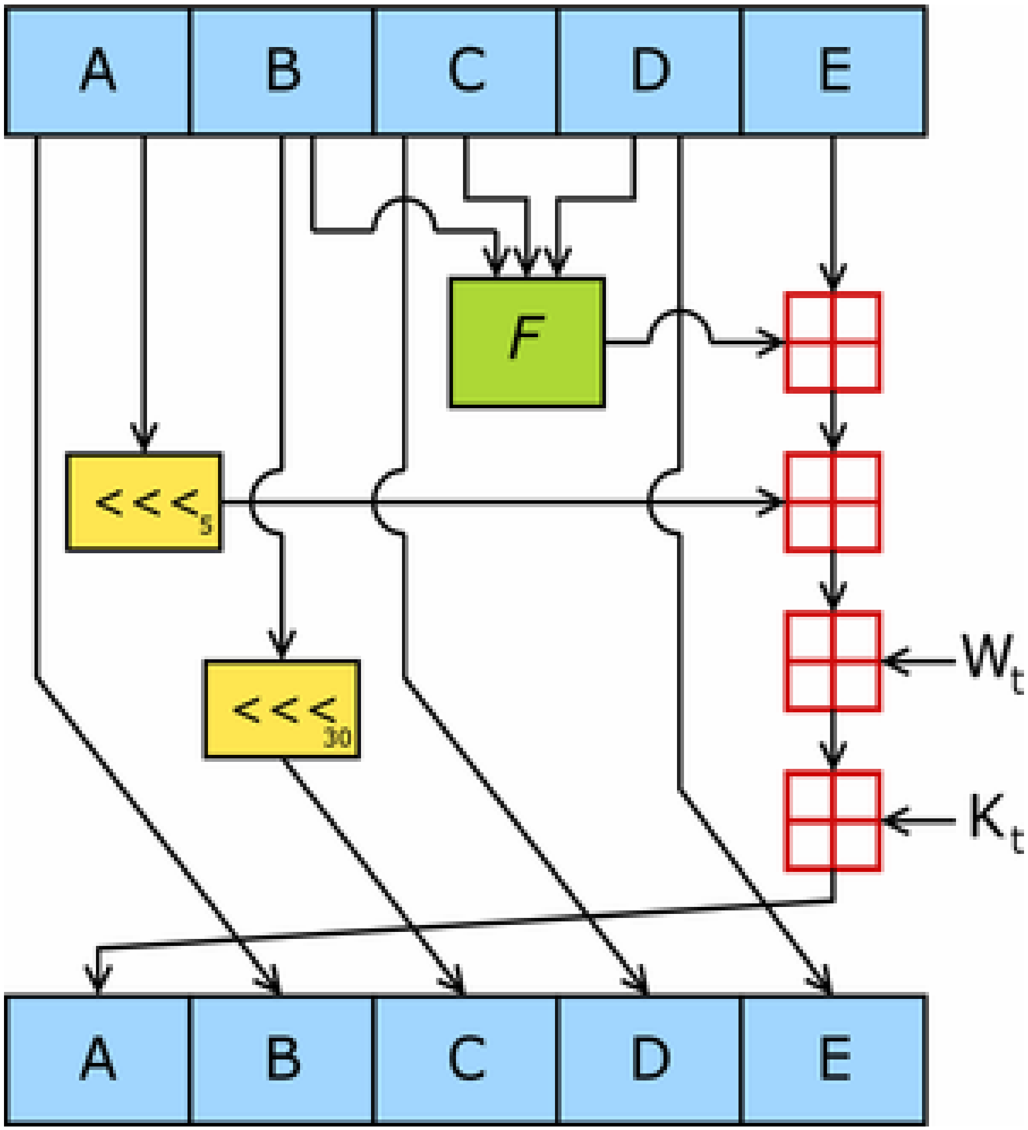
SHA-1 Expansion Round

- Expands 512-bit (16 32-bit words) to 2,560 bits (80 32-bit words)
- Defined by the recurrence:

$$W_t = \begin{cases} M_t & t = 0 \ldots 15 \\ (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) \lll 1 & t = 16 \ldots 79 \end{cases}$$

- Differs from SHA-0 by the 1-bit rotation to the left

An SHA-1 Step



- A – E are the 32-bit internal registers
- F is one of the following

$$
\begin{aligned}
F_1 &= (X \wedge Y) \vee (\neg X \wedge Z) \\
F_2 &= X \oplus Y \oplus Z \\
F_3 &= (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z) \\
F_4 &= X \oplus Y \oplus Z
\end{aligned}
$$

- W_t is the t^{th} expanded message word
- K_t is the t^{th} round constant
- <<< is a rotation to the left

Real World Examples

- Let M = 0000 0000 and M’ = 0000 00001
- MD5(M) = `0x93b885adfe0da089cdf634904fd59f71`
- MD5(M’) = `0x55a54008ad1ba589aa210d2629c1df41`
- SHA-1(M) = `0x5ba93c9db0cff93f52b521d7420e43f6eda2784f`
- SHA-1(M’) = `0xbf8b4530d8d246dd74ac53a13471bba17941dff7`
- On average changing 1 input bit will change 50% of the output bits
  - Avalanche condition

Current State of Hash Functions

- Researchers try to break hash functions in 1 of 2 ways
  1. Finding two messages (usually single message blocks) hash to the same digest
  2. Find the message that creates the digest of all zeros or all ones (essentially finding a preimage)
- Collisions and preimages can be found for MD4
- Collisions can be constructed for MD5 and SHA-0
- Theoretic collisions discovered for SHA-1
- SHA-256 & SHA-512 has no known attacks

References

- "Handbook of Applied Cryptography", Menezes, van Oorshot, Vanstone
- FIPS PUB 180-2, NIST
- RFC 1321
- Pictures from
  - http://en.wikipedia.org/wiki/Image:MD5.png
  - http://en.wikipedia.org/wiki/Image:SHA-1.png