CNT4406/5412 Network Security

Cryptographic Hash Functions

Zhi Wang

Florida State University

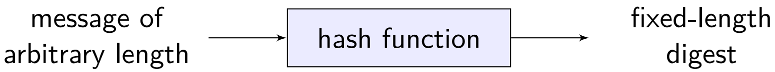
Fall 2014

Introduction

- A cryptographic hash (a.k.a. message digest) is a one-way function

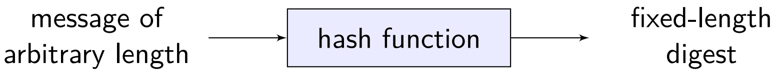
Introduction

- A cryptographic hash (a.k.a. message digest) is a one-way function
  - ➠ One-way: no reverse function for a hash (unlike encryption)
  - ➠ It takes arbitrary-length input and generate fixed-length output
  - ➠ It requires at least 128-bit output



Introduction

- A cryptographic hash (a.k.a. message digest) is a one-way function
  - One-way: no reverse function for a hash (unlike encryption)
  - It takes arbitrary-length input and generate fixed-length output
  - It requires at least 128-bit output (birthday problem)



Introduction

- A cryptographic hash (a.k.a. message digest) is a one-way function
  - One-way: no reverse function for a hash (unlike encryption)
  - It takes arbitrary-length input and generate fixed-length output
  - It requires at least 128-bit output (birthday problem)
- It should be fast to compute and have strong cryptographic strengths

Hash Function Properties

One-way property (pre-image resistance):

Given h, it's computationally infeasible to find m with $h = H(m)$

Hash Function Properties

One-way property (pre-image resistance):

Given h, it's computationally infeasible to find m with $h = H(m)$

Weak collision resistance (second-preimage resistance):

Given m_1, it's computationally infeasible to find m_2 with $H(m_1) = H(m_2)$

Hash Function Properties

One-way property (pre-image resistance):

Given h, it's computationally infeasible to find m with $h = H(m)$

Weak collision resistance (second-preimage resistance):

Given m_1, it's computationally infeasible to find m_2 with $H(m_1) = H(m_2)$

Strong collision resistance:

It's computationally infeasible to find m_1 and m_2 with $H(m_1) = H(m_2)$

Length of Hash Output

- What is the “right” size?
  - ➠ unnecessary overhead if too long
  - ➠ loss of strong collision resistance if too short (birthday problem)

Length of Hash Output

- What is the "right" size?
  - ➠ unnecessary overhead if too long
  - ➠ loss of strong collision resistance if too short (birthday problem)
- A hash normally has 128 or 160 bits of output

Birthday Problem

Birthday Problem:

what's the smallest number of people (n) in a room such that the probability of at least two of them having the same birthday is greater than 50%?

➠ assuming 365 days/year (k), and equal distribution of birthdays

Birthday Problem

Birthday Problem:

what's the smallest number of people (n) in a room such that the probability of at least two of them having the same birthday is greater than 50%?

➠ assuming 365 days/year (k), and equal distribution of birthdays

- The answer is 23, or, more generally, about $k^{\frac{1}{2}}$

Birthday Problem

Birthday Problem:

what's the smallest number of people (n) in a room such that the probability of at least two of them having the same birthday is greater than 50%?

➠ assuming 365 days/year (k), and equal distribution of birthdays

- The answer is 23, or, more generally, about $k^{\frac{1}{2}}$
- A brute-force attack needs to try $2^{\frac{len(h)}{2}}$ messages before finding two messages with the same hash at 50% chance ($k = 2^{len(h)}$)

Birthday Problem

Birthday Problem:

what's the smallest number of people (n) in a room such that the probability of at least two of them having the same birthday is greater than 50%?

➠ assuming 365 days/year (k), and equal distribution of birthdays

- The answer is 23, or, more generally, about $k^{\frac{1}{2}}$
- A brute-force attack needs to try $2^{\frac{len(h)}{2}}$ messages before finding two messages with the same hash at 50% chance ($k = 2^{len(h)}$)
  ➠ 64-bit hash only has 32 bits of strong collision resistance

Birthday Problem (by Wrong Math)

- Birthday problem
  - ➠ n people can form $\frac{n(n-1)}{2}$ different groups
  - ➠ each group has a chance of $\frac{1}{k}$ to have the same birthday
  - ➠ adding them together: $\frac{n(n-1)}{2k}$

Birthday Problem (by Wrong Math)

- Birthday problem
  - ➠ n people can form $\frac{n(n-1)}{2}$ different groups
  - ➠ each group has a chance of $\frac{1}{k}$ to have the same birthday
  - ➠ adding them together: $\frac{n(n-1)}{2k}$
- But, groups are not independent of each other.
  - ➠ cannot simply add them!
  - ➠ e.g., if 30 people $\rightarrow$ 435 groups $\rightarrow$ a probability of 119

Birthday Problem

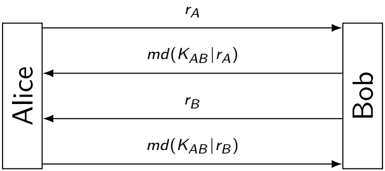
- Probability of n people have different birthdays is:
  $P = \frac{(365)_n}{365^n} = \frac{365 \times 364 \times \ldots \times (365-n+1)}{365^n}$

Birthday Problem

- Probability of n people have different birthdays is:
  $P = \frac{(365)_n}{365^n} = \frac{365 \times 364 \times ... \times (365-n+1)}{365^n}$
- Probability of at least two people have the same birthday is: $1 - P$
  $1 - P \geq 0.5 \rightarrow P < 0.5 \rightarrow \frac{(365)_n}{365^n} < 0.5 \rightarrow n = 23$

Authentication

- Alice and Bob shares a secret key K_{AB} in advance
- Alice challenges Bob by sending a random number r_A
- Bob returns the hash of $K_{AB}|r_A$
  ➠ in authentication with SKC, Bob returns $K_{AB}\{r_A\}$
- Alice also computes its hash, and compares it to Bob's
  ➠ in authentication with SKC, Alice decrypts Bob's message



MAC (Message Integrity)

- Sending hash with plaintext does not work
  - ➠ hash can detect benign errors (e.g., storage or network failure)
  - ➠ attackers can modify the message and regenerate the hash

MAC (Message Integrity)

- Sending hash with plaintext does not work
  - ➠ hash can detect benign errors (e.g., storage or network failure)
  - ➠ attackers can modify the message and regenerate the hash
- MAC requires keyed hash: Alice and Bob shares a secret K_{AB}

MAC (Message Integrity)

- Sending hash with plaintext does not work
  ➠ hash can detect benign errors (e.g., storage or network failure)
  ➠ attackers can modify the message and regenerate the hash
- MAC requires keyed hash: Alice and Bob shares a secret K_{AB}
  ➠ **extension attack**: given m_1 and $md(K_{AB}|m_1)$, we can compute $md(K_{AB}|m_1|m_2)$ by using $md(K_{AB}|m_1)$ as the IV, how?

MAC (Message Integrity)

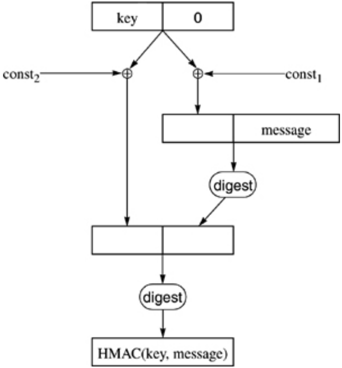
- Sending hash with plaintext does not work
  ➠ hash can detect benign errors (e.g., storage or network failure)
  ➠ attackers can modify the message and regenerate the hash
- MAC requires keyed hash: Alice and Bob shares a secret K_{AB}
  ➠ **extension attack**: given m_1 and $md(K_{AB}|m_1)$, we can compute $md(K_{AB}|m_1|m_2)$ by using $md(K_{AB}|m_1)$ as the IV, how?
  ➠ **solutions:** $md(m|K_{AB})$, $md(K_{AB}|m|K_{AB})$, and sending-half-of-the-hash

MAC (Message Integrity)

- Sending hash with plaintext does not work
  ➠ hash can detect benign errors (e.g., storage or network failure)
  ➠ attackers can modify the message and regenerate the hash
- MAC requires keyed hash: Alice and Bob shares a secret K_{AB}
  ➠ **extension attack**: given m_1 and $md(K_{AB}|m_1)$, we can compute $md(K_{AB}|m_1|m_2)$ by using $md(K_{AB}|m_1)$ as the IV, how?
  ➠ **solutions:** $md(m|K_{AB})$, $md(K_{AB}|m|K_{AB})$, and sending-half-of-the-hash
  ➠ HMAC is the de facto standard

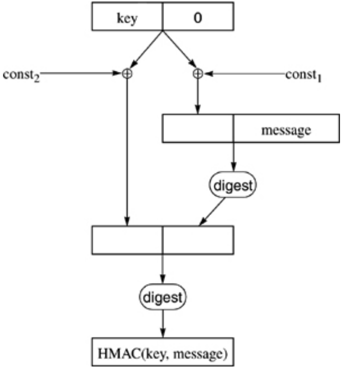
HMAC (Hash-based MAC)

- $HMAC(K, m) = MD((K \oplus c_2)|MD((K \oplus c_1)|m))$



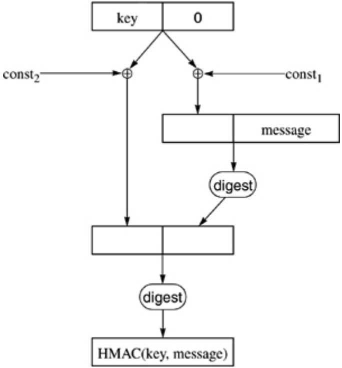
HMAC (Hash-based MAC)

- $HMAC(K, m) = MD((K \oplus c_2)|MD((K \oplus c_1)|m))$
- Nested digest with secrets prevents the extension attack
  - ➠ the inner digest is not revealed to the attacker



HMAC (Hash-based MAC)

- $HMAC(K, m) = MD((K \oplus c_2)|MD((K \oplus c_1)|m))$
- Nested digest with secrets prevents the extension attack
  ➠ the inner digest is not revealed to the attacker
- HMAC is proved to be secure if underlying message digest is secure



Encryption with Message Digest

- Generate a one-time pad to be $\oplus$'ed to the plaintext
  - from IV and a key (like OFB):
    $b_1 = MD(K_{AB}|IV), b_2 = MD(K_{AB}|b_1), \ldots$

Encryption with Message Digest

- Generate a one-time pad to be $\oplus$'ed to the plaintext
  - ➠ from IV and a key (like OFB):
    $b_1 = MD(K_{AB}|IV), b_2 = MD(K_{AB}|b_1), \ldots$

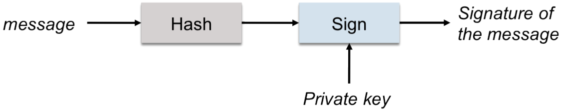
  or

  - ➠ mixing in the ciphertext (like CFB):

$$b_1 = MD(K_{AB}|IV) \quad c_1 = p_1 \oplus b_1$$
$$b_2 = MD(K_{AB}|c_1) \quad c_2 = p_2 \oplus b_2$$
$$\ldots$$

Digital Signature

- Public key cryptography is too slow to sign large messages
  - ➠ generate and sign the cryptographic hash of the message
  - ➠ rely on the security of the hash function



Commitment Protocol

- Commitment protocol: making a verifiable commitment without revealing it
  - ➠ Alice and Bob play the game of “odd or even” online:

Commitment Protocol

- Commitment protocol: making a verifiable commitment without revealing it
  - ➠ Alice and Bob play the game of "odd or even" online:
  - ➠ Alice and Bob both pick a number

Commitment Protocol

- Commitment protocol: making a verifiable commitment without revealing it
  - ➠ Alice and Bob play the game of "odd or even" online:
  - ➠ Alice and Bob both pick a number
  - ➠ they exchange the number at the "exactly" same time

Commitment Protocol

- Commitment protocol: making a verifiable commitment without revealing it
  - ➠ Alice and Bob play the game of “odd or even” online:
  - ➠ Alice and Bob both pick a number
  - ➠ they exchange the number at the “exactly” same time
  - ➠ Alice wins if the sum of the numbers are odd, otherwise Bob wins

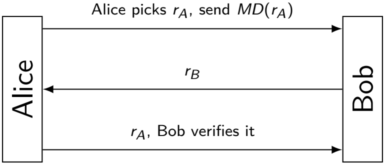
Commitment Protocol

- Commitment protocol: making a verifiable commitment without revealing it
  - ➠ Alice and Bob play the game of "odd or even" online:
  - ➠ Alice and Bob both pick a number
  - ➠ they exchange the number at the "exactly" same time
  - ➠ Alice wins if the sum of the numbers are odd, otherwise Bob wins

  ➠ but, it is difficult to get the "exactly" same time, one who delays until having received the other's number can easily cheat!

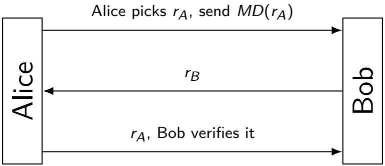
Commitment Protocol

- **Solution:** Alice makes a verifiable commitment before Bob sends his number, explain in details?



Commitment Protocol

- **Solution:** Alice makes a verifiable commitment before Bob sends his number, explain in details?
- Will this protocol work for the paper-scissors-rock game? why?



Popular Hash Functions

- MD5
  - ➠ designed by Ron Rivest in 1992 after MD2 and MD4
  - ➠ operate on 512-bit blocks and produce 128-bit message digest

Popular Hash Functions

- MD5
  - ➠ designed by Ron Rivest in 1992 after MD2 and MD4
  - ➠ operate on 512-bit blocks and produce 128-bit message digest
  - ➠ found to be broken wrt. collision resistance (2004-2007)
  - ➠ MD5 "should be considered cryptographically broken and unsuitable for further use."

  http://www.win.tue.nl/hashclash/SoftIntCodeSign/

Popular Hash Functions

- MD5
  - ➠ designed by Ron Rivest in 1992 after MD2 and MD4
  - ➠ operate on 512-bit blocks and produce 128-bit message digest
  - ➠ found to be broken wrt. collision resistance (2004-2007)
  - ➠ MD5 "should be considered cryptographically broken and unsuitable for further use."
    http://www.win.tue.nl/hashclash/SoftIntCodeSign/
  - ➠ SHA-2 is recommended as SHA-1 is also flawed

Popular Hash Functions

- SHA-1 (Secure Hash Algorithm)
  - ➠ designed by NSA and published by NIST
  - ➠ operate on 512-bit blocks and produce 160-bit output

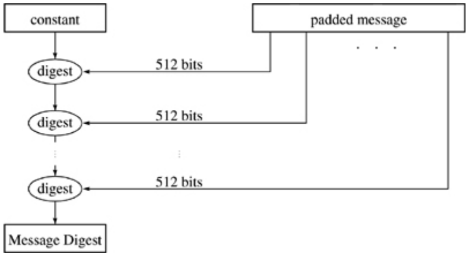
Popular Hash Functions

- SHA-1 (Secure Hash Algorithm)
  ➠ designed by NSA and published by NIST
  ➠ operate on 512-bit blocks and produce 160-bit output

  ➠ collision can be found in 2^{69} calculations, 2000 times faster than brute-force (2^{80}) (2005)
  ➠ "that is just on the far edge of feasibility with current technology." (http://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html)

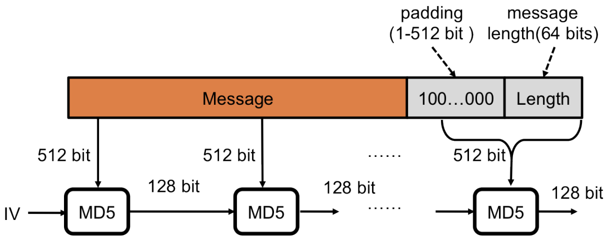
Common Structure

- Initialize message digest to a fixed constant
- Update the current digest with the next block of message
  - ➠ also called the compression function (512 bits → digest length)
  - ➠ block by block (extension attack)
- Output the final result as the digest for the entire message



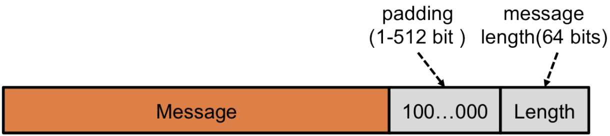
MD5: Overview

- Pad message to a multiple of 512 bits
- Digest message block by block (also called stages)



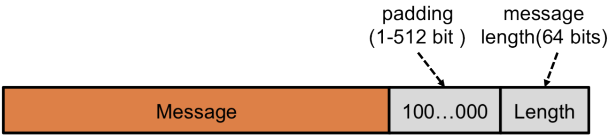
MD5: Message Padding

- Start padding with a 1, followed by just enough 0 bits to make the message of $512 \times n - 64$ bits



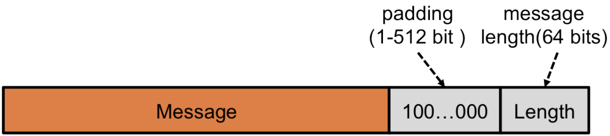
MD5: Message Padding

- Start padding with a 1, followed by just enough 0 bits to make the message of $512 \times n - 64$ bits
  ➠ what if the original message has $512 \times n$ bits?



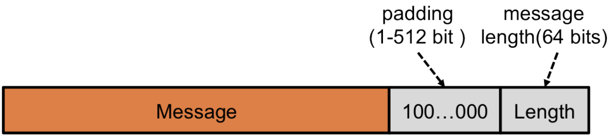
MD5: Message Padding

- Start padding with a 1, followed by just enough 0 bits to make the message of $512 \times n - 64$ bits
  - ➠ what if the original message has $512 \times n$ bits?
  - ➠ $512 \times n - 63$? $512 \times n - 64$? $512 \times n - 65$?



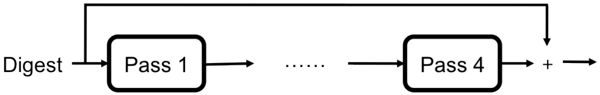
MD5: Message Padding

- Start padding with a 1, followed by just enough 0 bits to make the message of $512 \times n - 64$ bits
  - ➠ what if the original message has $512 \times n$ bits?
  - ➠ $512 \times n - 63$? $512 \times n - 64$? $512 \times n - 65$?
- Append 64 bit of message length



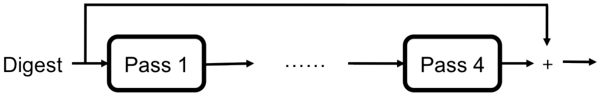
MD5: A Stage

- Each stage takes a block of message and intermediate digest
  - ➠ 512-bit message block: 16 32-bit words named $m_0, m_1, \ldots, m_{15}$
  - ➠ 128-bit intermediate digest: 4 32-bit words named d_0, d_1, d_2, d_3



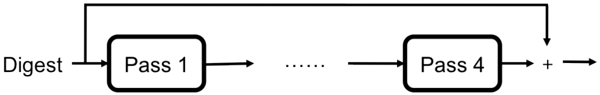
MD5: A Stage

- Each stage takes a block of message and intermediate digest
  - 512-bit message block: 16 32-bit words named $m_0, m_1, ..., m_{15}$
  - 128-bit intermediate digest: 4 32-bit words named d_0, d_1, d_2, d_3
- Each stage makes 4 passes over the block to update the digest



MD5: A Stage

- Each stage takes a block of message and intermediate digest
  - ➠ 512-bit message block: 16 32-bit words named $m_0, m_1, ..., m_{15}$
  - ➠ 128-bit intermediate digest: 4 32-bit words named d_0, d_1, d_2, d_3
- Each stage makes 4 passes over the block to update the digest
- The output is the final modified digest + pre-stage digest



MD5: Notation

- $\sim x$: bit-wise complement of x
- $x \vee y$, $x \wedge y$, $x \oplus y$: bit-wise OR, AND, XOR of x and y
- $x \Lsh n$: left-rotate x by n bits
- T: a table of 64 constants

MD5: Pass 1

- Select function: $F(x, y, z) = (x \wedge y) \vee (\sim x \wedge z)$
  ➠ Select n-th bit of y if n-th bit of x is 1, otherwise n-th bit of z
- For $i = 0$ to 15:
  $d_{-i\wedge 3} = d_{(1-i)\wedge 3} + (d_{-i\wedge 3} + F(d_{(1-i)\wedge 3}, d_{(2-i)\wedge 3}, d_{(3-i)\wedge 3}) + m_i + T_{i+1}) \looparrowleft S_1(i \wedge 3)$
- $d_0 = d_1 + (d_0 + F(d_1, d_2, d_3) + m_0 + T_1) \looparrowleft 7$
  $d_3 = d_0 + (d_3 + F(d_0, d_1, d_2) + m_1 + T_2) \looparrowleft 12$
  $d_2 = d_3 + (d_2 + F(d_3, d_0, d_1) + m_2 + T_3) \looparrowleft 17$
  $d_1 = d_2 + (d_1 + F(d_2, d_3, d_0) + m_3 + T_4) \looparrowleft 22$
  ...

MD5: Pass 2

- Select function: $G(x, y, z) = (x \wedge z) \vee (y \wedge \sim z)$
- For $i = 0$ to 15:

  $d_{-i \wedge 3} = d_{(1-i)\wedge 3} + (d_{-i\wedge 3} + G(d_{(1-i)\wedge 3}, d_{(2-i)\wedge 3}, d_{(3-i)\wedge 3}) + m_{(5i+1)\wedge 15} + T_{i+17}) \looparrowleft S_2(i \wedge 3)$
- $d_0 = d_1 + (d_0 + G(d_1, d_2, d_3) + m_1 + T_{17}) \looparrowleft 5$

  $d_3 = d_0 + (d_3 + G(d_0, d_1, d_2) + m_6 + T_{18}) \looparrowleft 9$

  $d_2 = d_3 + (d_2 + G(d_3, d_0, d_1) + m_{11} + T_{19}) \looparrowleft 14$

  $d_1 = d_2 + (d_1 + G(d_2, d_3, d_0) + m_0 + T_{20}) \looparrowleft 20$

  ...

MD5: Pass 3

- Select function: $H(x, y, z) = x \oplus y \oplus z$
- For $i = 0$ to 15:

  $d_{-i \wedge 3} = d_{(1-i) \wedge 3} + (d_{-i \wedge 3} + H(d_{(1-i) \wedge 3}, d_{(2-i) \wedge 3}, d_{(3-i) \wedge 3}) + m_{(3i+5) \wedge 15} + T_{i+33}) \looparrowleft S_3(i \wedge 3)$
- $d_0 = d_1 + (d_0 + H(d_1, d_2, d_3) + m_5 + T_{33}) \looparrowleft 4$

  $d_3 = d_0 + (d_3 + H(d_0, d_1, d_2) + m_8 + T_{34}) \looparrowleft 11$

  $d_2 = d_3 + (d_2 + H(d_3, d_0, d_1) + m_{11} + T_{35}) \looparrowleft 16$

  $d_1 = d_2 + (d_1 + H(d_2, d_3, d_0) + m_{14} + T_{36}) \looparrowleft 23$

  ...

MD5: Pass 4

- Select function: $I(x, y, z) = y \oplus (x \vee \sim z)$
- For $i = 0$ to 15:

  $d_{-i \wedge 3} = d_{(1-i) \wedge 3} + (d_{-i \wedge 3} + I(d_{(1-i) \wedge 3}, d_{(2-i) \wedge 3}, d_{(3-i) \wedge 3}) + m_{(7i) \wedge 15} + T_{i+49}) \looparrowleft S_4(i \wedge 3)$
- $d_0 = d_1 + (d_0 + I(d_1, d_2, d_3) + m_0 + T_{49}) \looparrowleft 6$

  $d_3 = d_0 + (d_3 + I(d_0, d_1, d_2) + m_7 + T_{50}) \looparrowleft 10$

  $d_2 = d_3 + (d_2 + I(d_3, d_0, d_1) + m_{14} + T_{51}) \looparrowleft 15$

  $d_1 = d_2 + (d_1 + I(d_2, d_3, d_0) + m_5 + T_{52}) \looparrowleft 21$

  ...

SHA-1

- Structurally similar to MD4 and MD5
  ➠ work in stages and use the same message padding

SHA-1

- Structurally similar to MD4 and MD5
  - ➠ work in stages and use the same message padding
- Process messages in 512-bit blocks, produce 160-bit digest
  - ➠ message block: 16 32-bit words named as $w_0, w_1, ..., w_{15}$
  - ➠ digest: 5 words named as A, B, C, D, E

SHA-1

- Structurally similar to MD4 and MD5
  ➠ work in stages and use the same message padding
- Process messages in 512-bit blocks, produce 160-bit digest
  ➠ message block: 16 32-bit words named as $w_0, w_1, \ldots, w_{15}$
  ➠ digest: 5 words named as A, B, C, D, E
- Each SHA-1 stage has 5 passes
  ➠ a pre-process pass to extend the block to 80 words (32 bits), grouped into 4 sets of 20 words
  ➠ four digest passes each updates the digest with its set of words

SHA1: Preprocess Pass

- Extend the message block into 80 words
  - ➠ words 0 to 15 are just copied over
  - ➠ for $i = 16$ to 79: $w_i = (w_{i-16} \oplus w_{i-14} \oplus w_{i-8} \oplus w_{i-3}) \looparrowleft 1$

SHA1: Preprocess Pass

- Extend the message block into 80 words
  - ➠ words 0 to 15 are just copied over
  - ➠ for $i = 16$ to 79: $w_i = (w_{i-16} \oplus w_{i-14} \oplus w_{i-8} \oplus w_{i-3}) \looparrowleft 1$
- Group these 80 words into 4 sets, each containing 20 words
  $w_0 - w_{19}, w_{20} - w_{39}, w_{40} - w_{59}, w_{60} - w_{79}$

SHA1: Digest Pass

- Each pass updates the digest using 20 words, w_0 to w_{19}
- Assume p is the pass number $(0, 1, 2, 3)$, then
  for $i = 0$ to 19:
  $A' = E + (A \Lsh 5) + w_i + K_p + f_p(B, C, D)$
  $B' = A$
  $C' = B \Lsh 30$
  $D' = C$
  $E' = D$
  K_p is a constant, f_p is a function (both vary wrt. p)

SHA1: Digest Pass

- Why MD5 and SHA1 is much faster than DES?

Digest Pass	SHA-1	MD5
1	$(B \wedge C) \vee (\sim B \wedge D)$	$(x \wedge y) \vee (\sim x \wedge z)$
2	$B \oplus C \oplus D$	$(x \wedge z) \vee (y \wedge \sim z)$
3	$(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$	$x \oplus y \oplus z$
4	$B \oplus C \oplus D$	$y \oplus (x \vee \sim z)$

Conclusion

Bruce Schneier:

"Hash functions are the least-well-understood cryptographic primitive, and hashing techniques are much less developed than encryption techniques."