Cryptographic Primitives and the SHA-1 Hashing Algorithm

Wesley Romberger (USC)

Faculty Advisors: Dr. Manki Min

SPACT Center for Security Printing & Anti-Counterfeiting Technology

SDSU South Dakota State University

USC

The Secure Hashing Algorithm 1 (SHA1)

- SHA-1 is a cryptographic hash function designed by the United States National Security Agency and is a U.S. Federal Information Processing Standard.
- Published in 1995, SHA-1 is very similar to SHA-0, but alters the original SHA hash specification to correct alleged weaknesses.
- SHA-1 is the most widely used of the existing SHA hash functions, and is employed in several widely used applications and protocols.
- SHA-1 produces a 160-bit (20-byte) hash value. A SHA-1 hash value is typically rendered as a hexadecimal number, 40 digits long.

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

Output Digest: *2FD4E1C67A2D28FCED849EE1BB76E7391B93EB12*

Uses

- Verifying the integrity of files or messages

  Used to determine if a file of messages has been tampered with.

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

  Output Digest: *2FD4E1C67A2D28FCED849EE1BB76E7391B93EB12*

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

  Output Digest: *DE9F2C7FD25E1B3AFAD3E85A0BD17D9B100DB4B3*

- Password Security

  Password is hashed (and usually salted) then stored in its hashed version to prevent theft and fraud.

  SHA1("Password")

  Output Digest: *8BE3C943B1609FFFBFC51AAD666D0A04ADF83C9D*

- Pseudorandom generation and key derivation

  Hash functions can also be used in the generation of pseudorandom bits, or to derive new keys or passwords from a single, secure key or password.

- Digital Signatures

Problems

- Collisions

  It should be difficult to find two different messages m_1 and m_2 such that $hash(m_1) = hash(m_2)$. Such a pair is called a cryptographic hash collision.

Hypothetical Hash("I, Wesley Romberger, agree to receive $5000 for the privilege of working for the SPACT REU")

Output Digest: *2FD4E1C67A2D28FCED849EE1BB76E7391B93EB12*

Hypothetical Hash("I, Wesley Romberger, agree to pay $5000 for the privilege of working for the SPACT REU")

Output Digest: *2FD4E1C67A2D28FCED849EE1BB76E7391B93EB12*

Modeling

- A working version of the SHA-1 was created in the CodeBlocks IDE using the C++ language.

Initialize variables:

h0 = 0x67452301
h1 = 0xEFCDAB89
h2 = 0x98BADCFE
h3 = 0x10325476
h4 = 0xC3D2E1F0

Pre-processing:
append the bit '1' to the message i.e. by adding 0x80 if characters are 8 bits.
append 0 ≤ k < 512 bits '0', thus the resulting message length (in bits)
is congruent to 448 (mod 512)
append ml, in a 64-bit big-endian integer. So now the message length is a multiple of 512 bits.

Process the message in successive 512-bit chunks:
break message into 512-bit chunks
for each chunk
break chunk into sixteen 32-bit big-endian words w[i], 0 ≤ i ≤ 15

Extend the sixteen 32-bit words into eighty 32-bit words:
for i from 16 to 79
w[i] = (w[i-3] xor w[i-8] xor w[i-14] xor w[i-16]) leftrotate 1

Initialize hash value for this chunk:
a = h0
b = h1
c = h2
d = h3
e = h4

Main loop:
for i from 0 to 79
if 0 ≤ i ≤ 19 then
f = (b and c) or ((not b) and d)
k = 0x5A827999
else if 20 ≤ i ≤ 39
f = b xor c xor d
k = 0x6ED9EBA1
else if 40 ≤ i ≤ 59
f = (b and c) or (b and d) or (c and d)
k = 0x8F1BBCDC
else if 60 ≤ i ≤ 79
f = b xor c xor d
k = 0xCA62C1D6

temp = (a leftrotate 5) + f + e + k + w[i]
e = d
d = c
c = b leftrotate 30
b = a
a = temp

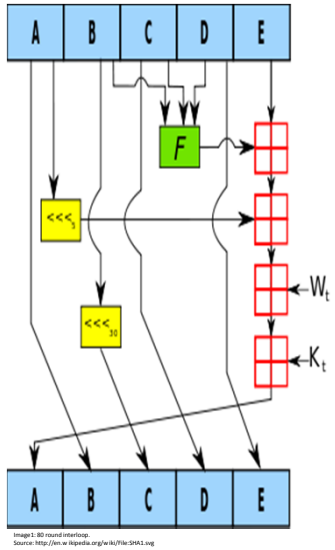


Image 1: 80 round interloop.
Source: http://en.wikipedia.org/wiki/File:SHA1.svg

Add this chunk's hash to result so far:
h0 = h0 + a
h1 = h1 + b
h2 = h2 + c
h3 = h3 + d
h4 = h4 + e

Output the concatenation (h0,h1,h2,h3,h4) which is the message digest.

Benchmarks

- Benchmarks used for measuring the efficiency of the program are memory consumption and runtime
- The will be collected using the win32 API for C++, and the Stanford C++ Library

Benchmark Tests include:

1,000,000 strings

Every word from an online dictionary

The novels: Alice in Wonderland, Tom Sawyer, and Hamlet

Alterations

- Two types of changes will be made to the original algorithm in an attempt to increase the security of the algorithm, while minimizing the impact on the memory consumption and time-to-hash of the program

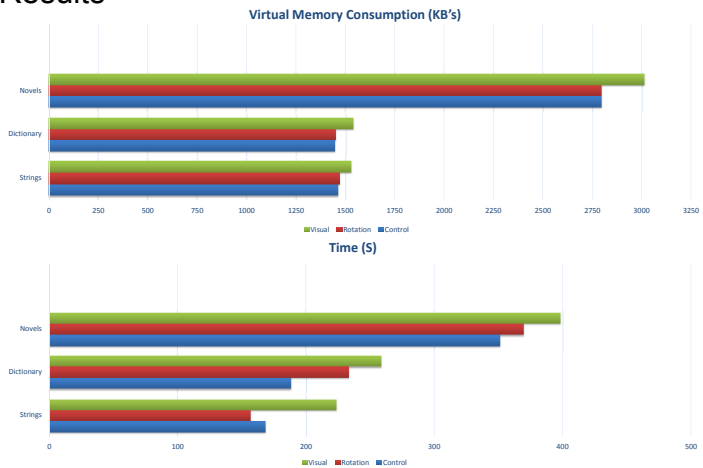
AES(Advanced Encryption Standard) influenced bit-rotation

Using a written bit rotator sub routine a rotation will be implemented on the length 80 array as it is constructed

Extra Salt and Text to Image

Introduce an additional salt for each round and use this to increase the digest. Afterwards use the output digest to create a visual of the hash in the form of a picture.

Results



Acknowledgements:
This work was made possible by the National Science Foundation REU Security Printing and Anti-Counterfeiting Site EEC-1263343

NSF