More Hashing

CAC 210
Spring 2018
Amber Wagner

Cryptography

- A cryptographic hash function is a special class of hash function
  - Mathematical algorithm that maps data of arbitrary size to a bit string of a fixed size (hash)
  - One-way function
- The input is the message
- The output (hash) is the message digest

Cryptographic Hash Functions

- Five main properties:

    1. Deterministic - same message always results in the same hash

    2. Quick to compute

    3. Infeasible to generate a message from the hash value except by brute force

    4. Small change to the message should result in a large change in the hash value

    5. Infeasible to find two different messages with the same hash

Cryptographic Hash Functions

- Five main properties:

    1. Deterministic - same message always results in the same hash

    2. Quick to compute

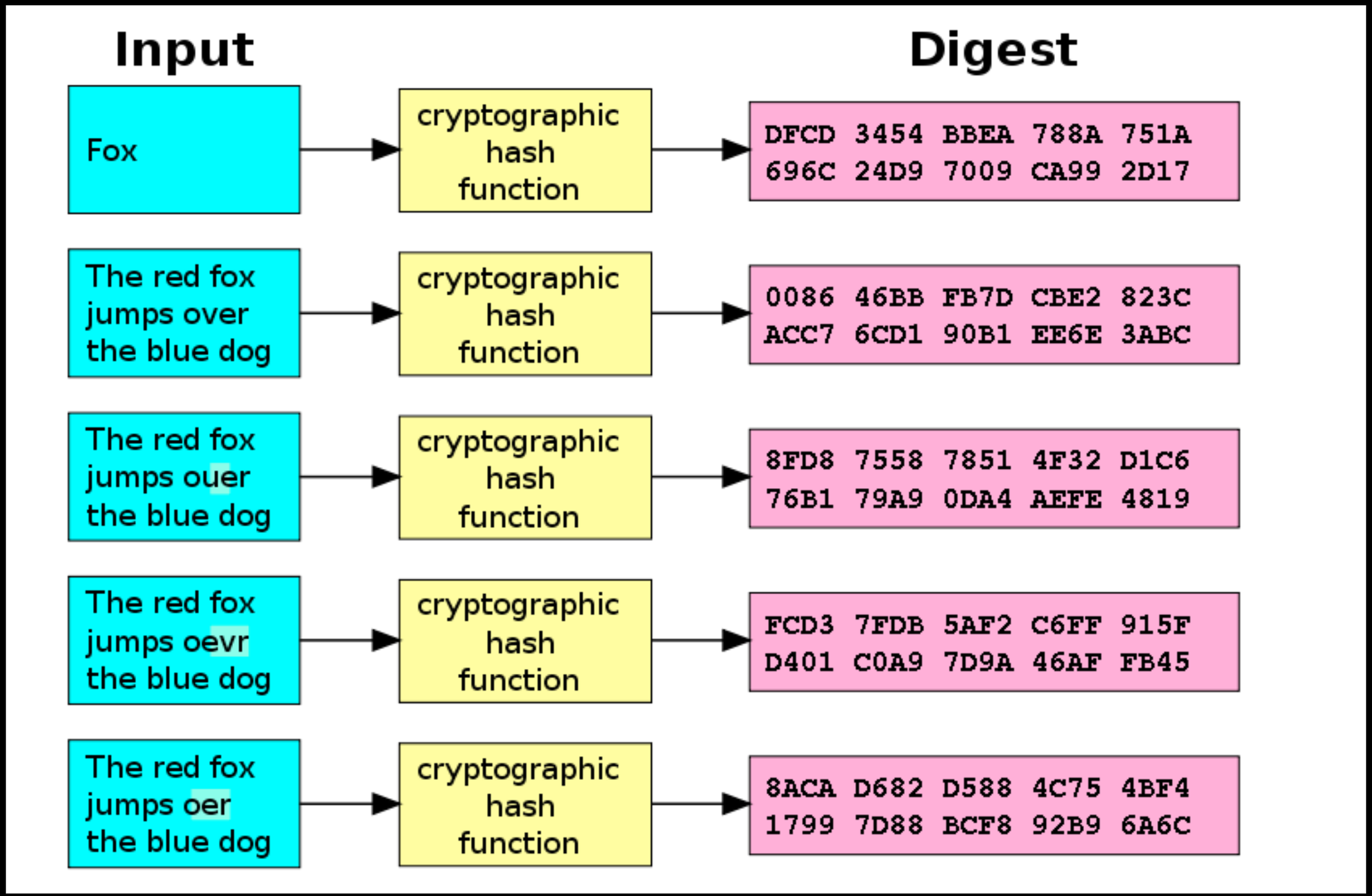
    3. Infeasible to generate a message from the hash value except by brute force

    4. Small change to the message should result in a large change in the hash value

    5. Infeasible to find two different messages with the same hash

Collision

Cryptographic Hash Functions



Cryptographic Hash Functions

- Designed to take a string of any length as input and produce a fixed-length hash value

- Must be able to withstand any cryptanalytic attack

- Preimage resistance

- Second preimage resistance

- Collision resistance

Preimage Resistance

Trying to find a message that has a specific hash

- Given a hash value h, it should be difficult to find any message m such that h = hash(m)
- This concept is related to that of a one-way function

Second-Preimage Resistance

- Given an input m, it should be difficult to find a different input m’ such that hash(m) = hash(m’).

- Similar to collision resistance

- Collision resistance implies second-preimage resistance, but does not guarantee preimage resistance

- A second-preimage attack implies a collision attack and a preimage attack

Collision Resistance

- It should be difficult to find two different messages m and m' such that hash(m) = hash(m')
- Such a pair is called a cryptographic hash collision
- Requires a hash value at least twice as long as that required for preimage resistance
- A birthday attack is a way to find collisions: https://en.wikipedia.org/wiki/Birthday_attack

Applications

- Verifying the integrity of files or messages
- Password verification
- Proof-of-work
- File or data identifier
- Pseudorandom generation and key derivation

2013 Password Hashing Competition
2015 > Argon2
June 2017, NIST issued revised guidelines:
“Verifies SHALL store memorized secrets in a form that is resistant to offline attacks. Memorized secrets SHALL be salted and hashed using a suitable one-way key derivation function.”

Collisions

- Collisions occur when two different objects result in the same hash

- In thinking about a hash table (dictionary), how could you put two items into the same slot?

- Two solutions:

  - Separate Chaining: use a linked list in each slot adding each collided object to the list

  - Linear Probing: if the slot we need (k) is filled, check k +1, if it's filled, check k+2 and so on

Separate Chaining

- What kind of problems are there with this approach?
  - The beauty of hash tables is the constant runtime
  - If we start using lists, then we are back to linear runtime
  - We can keep this constant though, if we were to use something called a load factor
  - The goal would be to keep the average length of the lists under this load factor

Linear Probing

- What kind of problems exist here?
  - Will you necessarily find an empty slot?
  - What happens when you reach the end of the hash table?
  - How do you find the element again?

Are collisions that big of a deal?

Collision Attack

- A collision attack on a cryptographic hash tries to find two inputs producing the same hash value
- Find two different messages m and m’ such that hash(m) = hash(m’)
- Attacker has no control over the content of either message
- Every cryptographic hash function is inherently vulnerable to collisions using a birthday attack: a hash of n bits can be broken in $2^{n/2}$ time
- When a collision attack is discovered that is found to be faster than a birthday attack, a hash function is often denounced as “broken”

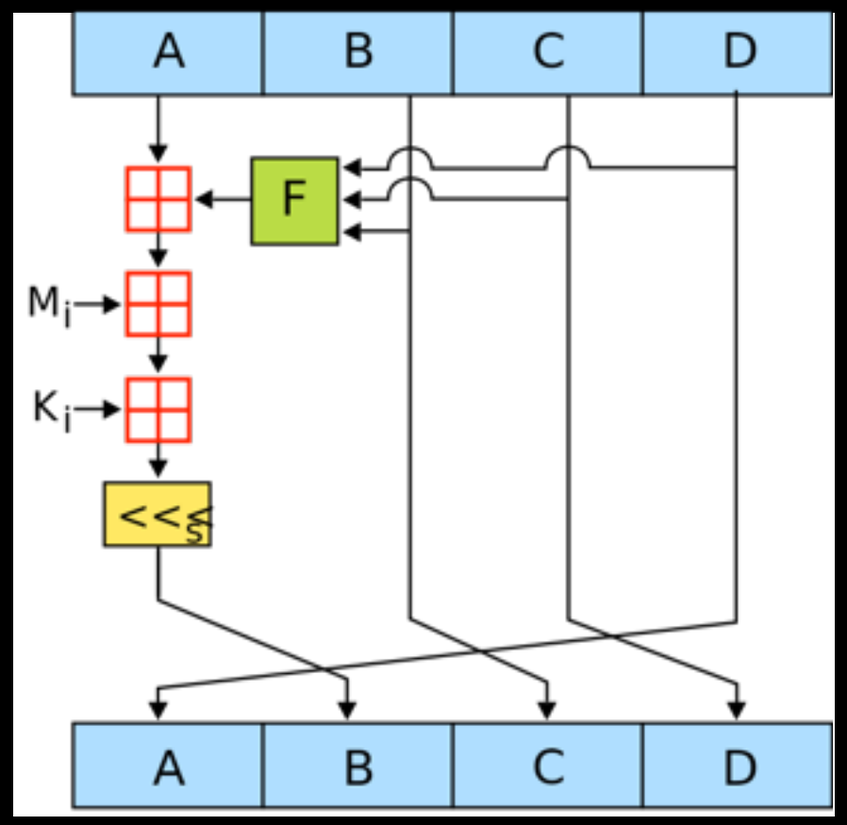
Collision Attack

- So, what does this mean?
- Do you know what a security certificate is?
- SSL (https) implies that there is a signed (validated) certificate approved by a valid certificate authority indicating that your content is secure
- An attacker can generate two certs with matching hashes, a legitimate one to send to the certificate authority and an invalid/malicious one
- The legitimate one would get signed, and that signature could then be transferred to the invalid one
- My computer would think the content is secure when it wasn’t
- https://www.cnet.com/news/web-browser-flaw-could-put-e-commerce-security-at-risk/

Classic Cryptographic Hash Functions

MD4

- Created by Ronald Rivest (MIT) in 1990
- Digest length is 128 bits
- Collision attack published in 1995
- As of 2007, an attack can generate collisions in less than 2 MD4 hash operations



One MD4 operation : MD4 consists of 48 of these operations, grouped in three rounds of 16 operations. F is a nonlinear function; one function is used in each round. Mi denotes a 32-bit block of the message input, and Ki denotes a 32-bit constant, different for each operation.

https://en.wikipedia.org/wiki/MD4

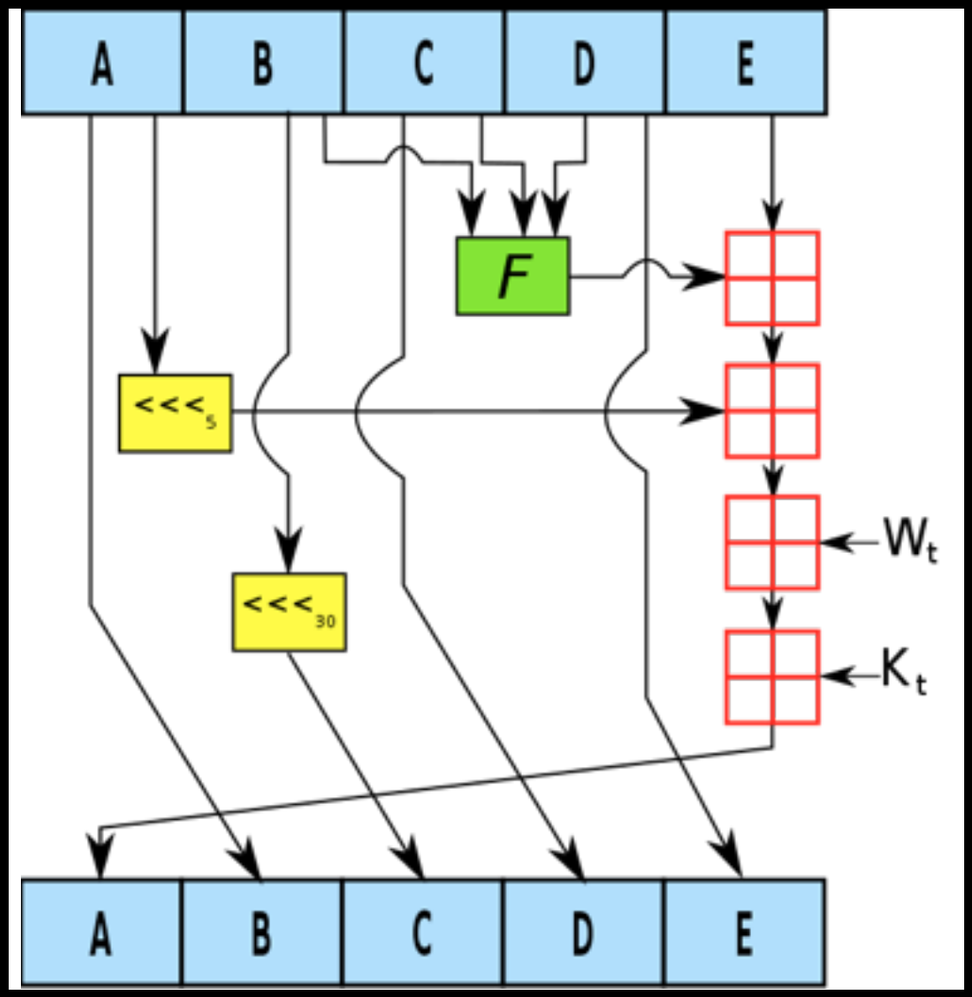
MD5

- Designed by Ronald Rivest in 1991 to replace MD4
- Produces 128-bit hash value
- Extensive vulnerabilities
- Can be cracked by brute-force attack
- A collision attack exists that can find collisions within seconds on a computer with a 2.6GHz Pentium processor
- Despite this, as of 2015, it is still widely used

https://en.wikipedia.org/wiki/MD5

SHA-1

- Based on principles Rivest used
- Published in 1993 by NIST
- NSA withdrew the original (SHA-0) and replaced it with SHA-1 in 1995



One iteration within the SHA-1 compression function:
A, B, C, D and E are 32-bit words of the state;
F is a nonlinear function that varies;
left shiftn denotes a left bit rotation by n places;
n varies for each operation;
Wt is the expanded message word of round t;
Kt is the round constant of round t;
Addition denotes addition modulo 232.

Examples

```
SHA1("The quick brown fox jumps over the lazy dog")
gives hexadecimal: 2fd4e1c67a2d28fced849ee1bb76e7391b93eb12
gives Base64 binary to ASCII text encoding: L9ThxnotKPzthJ7hu3bnORuT6xI=
```

```
SHA1("The quick brown fox jumps over the lazy cog")
gives hexadecimal: de9f2c7fd25e1b3afad3e85a0bd17d9b100db4b3
gives Base64 binary to ASCII text encoding: 3p8sf9JeGzr60+haC9F9mxANtLM=
```

Pseudocode

```
Note 1: All variables are unsigned 32-bit quantities and wrap modulo 2^32 when calculating,
except for
        ml, the message length, which is a 64-bit quantity, and
        hh, the message digest, which is a 160-bit quantity.
Note 2: All constants in this pseudo code are in big endian.
        Within each word, the most significant byte is stored in the leftmost byte position

Initialize variables:

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

ml = message length in bits (always a multiple of the number of bits in a character).

Pre-processing:
append the bit '1' to the message e.g. by adding 0x80 if message length is a multiple of 8
bits.
append 0 ≤ k < 512 bits '0', such that the resulting message length in bits
   is congruent to −64 ≡ 448 (mod 512)
append ml, the original message length, as a 64-bit big-endian integer. Thus, the total
length is a multiple of 512 bits.
```

Pseudocode

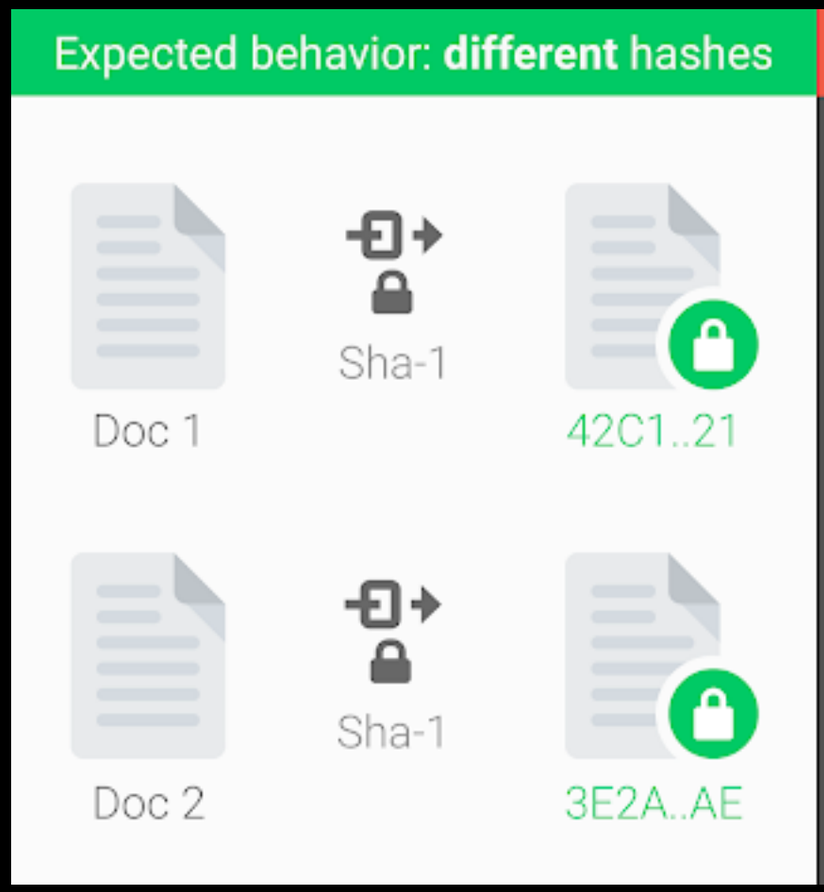
```
Process the message in successive 512-bit chunks:
break message into 512-bit chunks
for each chunk
    break chunk into sixteen 32-bit big-endian words w[i], 0 ≤ i ≤ 15

    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
```

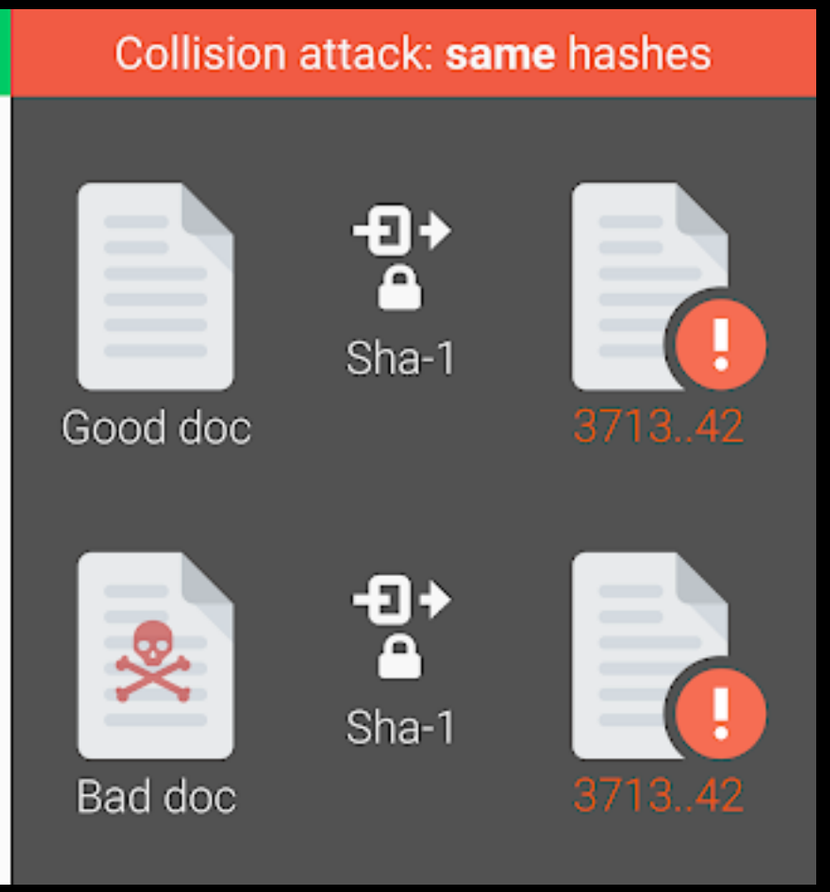
```
Main loop:[3][55]
for i from 0 to 79
    if 0 ≤ i ≤ 19 then
        f = (b and c) or ((not b) and d)
        k = 0x5A827999
    else if 20 ≤ i ≤ 39
        f = b xor c xor d
        k = 0x6ED9EBA1
    else if 40 ≤ i ≤ 59
        f = (b and c) or (b and d) or (c and d)
        k = 0x8F1BBCDC
    else 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

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

Produce the final hash value (big-endian) as a 160-bit number:
hh = (h0 leftshift 128) or (h1 leftshift 96) or (h2 leftshift 64) or (h3 leftshift 32) or h4
```

Attacks

- Several attacks were published in 2005, but the computing power was too much to consider the algorithm broken

- 2017 - Google announces the first public collision attack rendering the algorithm broken https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html





- Nine quintillion SHA1 computations
- 6,500 years of CPU computation to complete the first phase
- 110 years of GPU computation to complete the second phase

- Still 100,000 times faster than a brute force attack

SHA-256

- Designed by NSA
- 256-bit hash vs 128-bit
- https://passwordsgenerator.net/sha256-hash-generator/

References

- https://www.cs.cmu.edu/~adamchik/15-121/lectures/Hashing/hashing.html
- https://en.wikipedia.org/wiki/Cryptographic_hash_function
- https://en.wikipedia.org/wiki/Collision_attack
- https://en.wikipedia.org/wiki/SHA-1
- https://en.wikipedia.org/wiki/Preimage_attack