Introduction to Software Security

Hash Functions

(Chapter 5)

Seong-je Cho

Spring 2018

Computer Security & Operating Systems Lab, DKU

Sources / References

- Textbook, Chapter 5.
- An Illustrated Guide to Cryptographic Hashes
  - http://www.unixwiz.net/techtips/iguide-crypto-hashes.html
- Bitcoin Hash Functions Explained
  - https://www.coindesk.com/bitcoin-hash-functions-explained/

- N. Vlajic, CSE 3482: Introduction to Computer Security, Yorku
- Nicholas Weaver, Computer Science 161: Computer Security, Berkeley
- Myrto Arapinis, Computer Security: INFRA10067, University of Edinburgh

Contents

- What is a Hash Function?
  - Cryptographic hash function: SHA-2, SHA-3
  - For integrity
- The Birthday Problem
- Tiger Hash
- Uses of Hash Functions

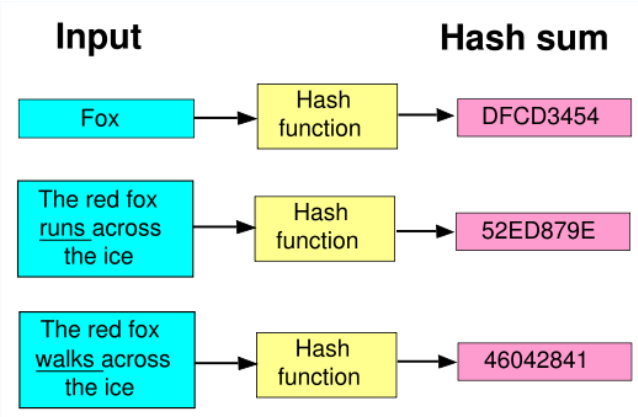
Security Threats

- MS threat model
  - STRIDE
- Security Properties (Security Goals)
  - CIA, Authentication, Non-repudiation, Authorization (Access Control)

Threats	Security Goals
Tampering • Modification, Alteration	**Integrity** • Cryptographic hash functions
Spoofing • Masquerade, Fabrication, Disguise	**Authentication** • Authenticity • Digital signature • (ID, password), Fingerprint

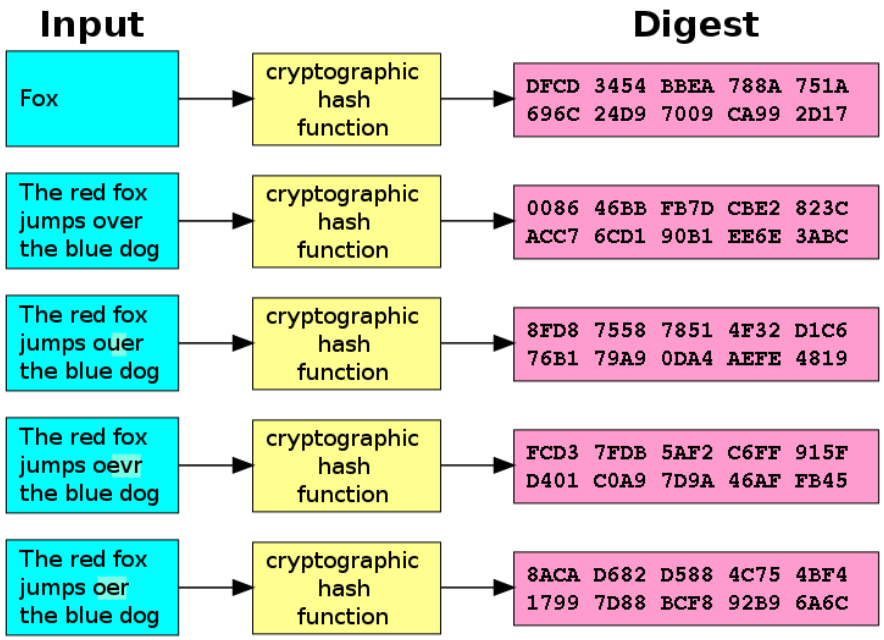
Hash function?

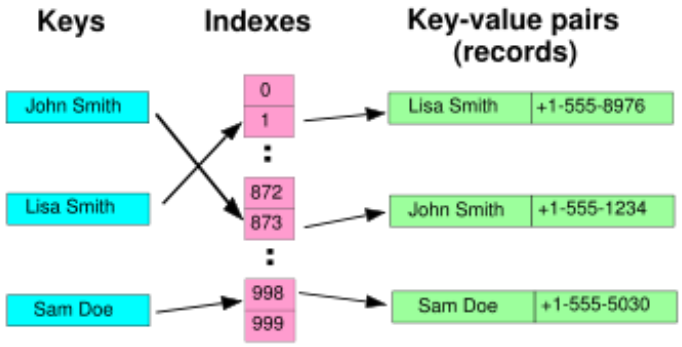
- A hash function **h** is a reproducible method of turning some kind of data into a (relatively) small number that may serve as a digital "fingerprint" of the data.
  - Hash = Digest

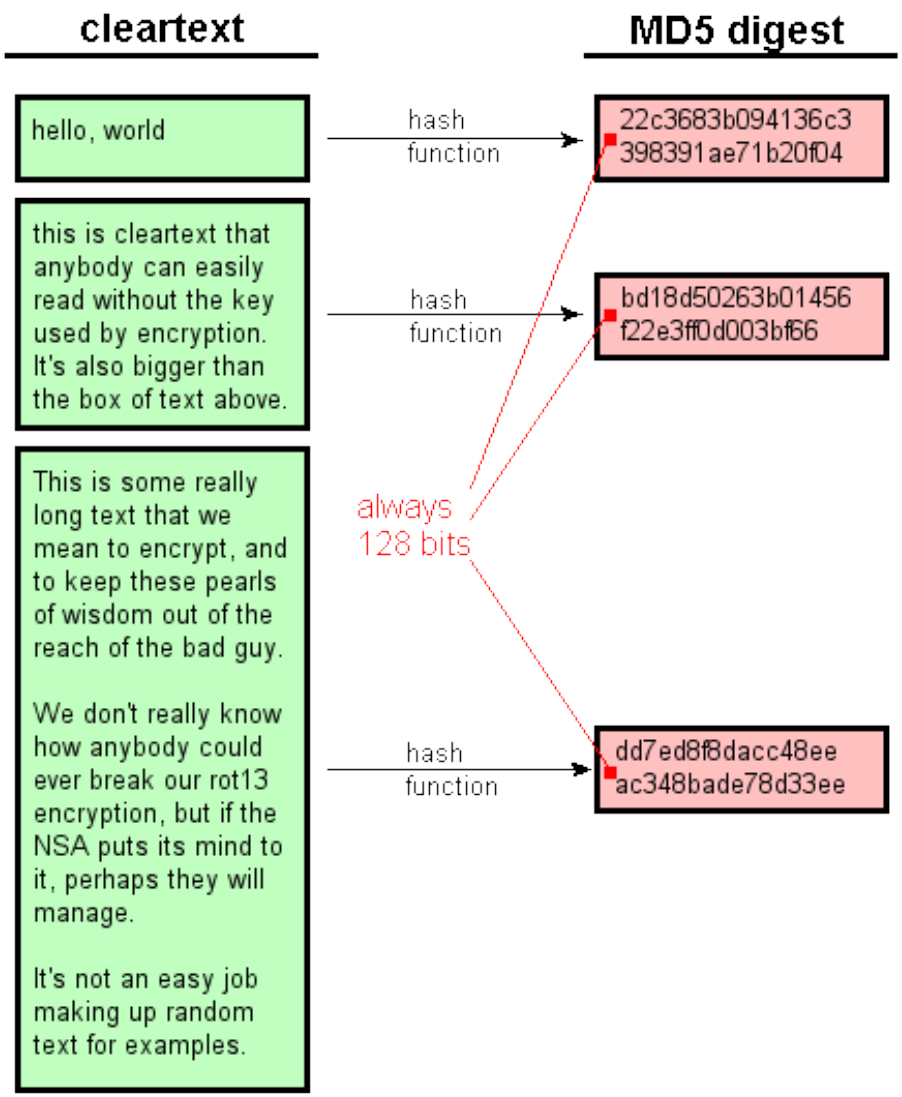


- Crypto Hash function: a hash function **h** with certain additional security properties to make it suitable for use as various info security applications
  - SHA-2: SHA-224, SHA-256, SHA-384, …

Hash Functions







http://en.wikipedia.org/wiki/Avalanche_effect

http://cse.csusb.edu/tong/courses/cs330/notes/hash.php

Cryptographic Hash

- Many Unix and Linux systems provide the **md5sum** program, which reads a stream of data and produces a fixed, 128-bit number that summarizes that stream using the popular "MD5" method.

```
$ cat smallfile
This is a very small file with a few characters

$ cat bigfile
This is a larger file that contains more characters.
This demonstrates that no matter how big the input
stream is, the generated hash is the same size (but
of course, not the same value). If two files have
a different hash, they surely contain different data.

$ ls -l empty-file smallfile bigfile linux-kernel
-rw-rw-r--    1 steve    steve           0 2004-08-20 08:58 empty-file
-rw-rw-r--    1 steve    steve          48 2004-08-20 08:48 smallfile
-rw-rw-r--    1 steve    steve         260 2004-08-20 08:48 bigfile
-rw-r--r--    1 root     root      1122363 2003-02-27 07:12 linux-kernel

$ md5sum empty-file smallfile bigfile linux-kernel
d41d8cd98f00b204e9800998ecf8427e  empty-file
75cdbfeb70a06d42210938da88c42991  smallfile
6e0b7a1676ec0279139b3f39bd65e41a  bigfile
c74c812e4d2839fa9acf0aa0c915e022  linux-kernel
```

Hash Function Motivation

- Suppose Alice signs M
  - Alice sends M and $S = [M]_{Alice}$ to Bob
  - Bob verifies that $M = \{S\}_{Alice}$
  - Aside: Is it OK to just send S?
- If M is big, $[M]_{Alice}$ is costly to compute

- Suppose instead, Alice signs h(M), where h(M) is much smaller than M
  - Alice sends M and $S = [h(M)]_{Alice}$ to Bob
  - Bob verifies that $h(M) = \{S\}_{Alice}$

Crypto Hash Function

- Crypto hash function h(x) must provide the following properties
- **Compression**
  - output length is small
- **Efficiency**
  - h(x) easy to computer for any x
- **One-way**
  - given a value y it is infeasible to find an x such that h(x) = y
- **Weak collision resistance**
  - *Given a plaintext x* and h(x), infeasible **to find y** with $y \neq x$ such that h(y) = h(x)
- **Strong collision resistance**
  - infeasible **to find any x and y,** with $x \neq y$ such that h(x) = h(y)

- Lots of collisions exist, but hard to find one

Birthday Paradox (=Birthday Problem)

- How large must a group of people be for there to be a 50% or greater chance of at least two of them to share a *given* birthday?

  Say, your birthday?

  - A kind of weak collision problem.

We can answer this question without much trouble. We will call the probability of finding a match p, but we will consider instead the probability that there *is not* a match: $\bar{p} = 1 - p$. Assuming that all birthdays in the year are equiprobable, we find

$$\bar{p} = \left(\frac{364}{365}\right)^n \tag{1}$$

for the probability that none of a group of n people share your birthday. With $\bar{p} = p = 0.5$, there must be $n = 253$ people. Now this is about large enough to seem plausible. As we noted though, this is a different problem. We are not interested in matching a specific birthday, but instead matching *any* birthday. This too isn't difficult, with [1]

.....

1. N명 중 특정한 사람을 선택하여 이 사람과 생일이 같은 사람이 (N-1)명 중에 있을 확률이 50%가 넘기 위한 N은?
2. N명 중 생일이 같은 두 명이 있을 확률이 50%가 넘기 위한 N은?

Source: The Birthday Problem and Hash Collisions, Brian Powell, 2015, http://tangentspace.info/docs/bday.pdf

Birthday Problem

- How many people must be in a room before probability is ≥ 1/2 that two or more have same birthday? (strong collision) = match any birthday
  - The probability that no two out of **n** people share a birthday

$$\bar{p} = \left(\frac{365}{365}\right)\left(\frac{364}{365}\right)\left(\frac{363}{365}\right)\cdots\left(\frac{365-n+1}{365}\right) = \frac{365!}{(365-n)!}\frac{1}{365^n}$$

  - The probability that any two students have the same birthday
    - » 1 – 365/365 · 364/365 · · ·(365–n+1)/365 = 1 – 365!/{(365-n)! · 365^n}
  - Set equal to 1/2 and solve: **n = 23**
- Surprising? A paradox?
- Maybe not: "Should be" about sqrt(365) since we compare all **pairs** x and y

19*19 = 361, 22*22=484, 23*23= 529, 27*27=729, sqrt(2)*365= 516.2, 2*365 = 730

Of Hashes and Birthdays

- If h(x) is N bits, then 2^N different hash values are possible
- sqrt(2^N) = $2^{N/2}$
- Therefore, hash about **$2^{N/2}$ random values** and you expect to find a collision
- **Implication:** secure N bit symmetric key requires 2^{N-1} work to "break" while secure N bit hash requires $2^{N/2}$ work to "break"

Non-Crypto Hash

Non-crypto Hash (1)

- Data $X = (X_0, X_1, X_2, \ldots, X_{n-1})$, each X_i is a byte
- Spse $hash(X) = X_0 + X_1 + X_2 + \ldots + X_{n-1} \bmod 256$
  - Output is always 8 bits

- Is this secure?
  - Example: (n = 2)

    X = (10101010,00001111) = 170+15 = 185 = 10111001

    Hash is 10111001

    But so is hash of Y = (00001111,10101010)

- Easy to find collisions, so **not** secure…

Non-crypto Hash (2)

- Data $X = (X_0, X_1, X_2, \ldots, X_{n-1})$
- Suppose hash is
  - $h(X) = nX_0 + (n-1)X_1 + (n-2)X_2 + \ldots + 1 \cdot X_{n-1} \bmod 256$
- Is this hash secure?
- At least
  - $h(10101010,00001111) \neq h(00001111,10101010)$

- But hash of (00000001,00001111) is same as hash of (00000000,00010001)
- Not one-way, but this hash is used in the (non-crypto) application rsync
  - **rsync** is a free software computer program for Unix systems which synchronizes files and directories from one location to another while minimizing data transfer using delta encoding when appropriate.

Non-crypto Hash (3)

- Cyclic Redundancy Check (CRC)
  - A CRC "checksum" is the remainder of a binary division with no bit carry (XOR used instead of subtraction), of the message bit stream, by a predefined (short) bit stream of length $n + 1$, which represents the coefficients of a polynomial with degree n.
  - Before the division, n zeros are appended to the message stream.
  - Read paper in http://www.repairfaq.org/filipg/LINK/F_crc_v31.html#CRCV_006 (A painless guide to CRC error detection alg)
  - CRCs and similar checksum methods are only designed to detect transmission errors
  - Not to detect intentional tampering with data
  - But CRC sometimes mistakenly used in crypto applications (WEP)
    - WEP(Wired Equivalent Privacy) uses the stream cipher RC4 and the CRC-32 checksum

Crypto Hash Function = Crypto Hash

(Cryptographic Hash Function)

```
$ cat file1
This is a very small file with a few characters

$ cat file2
this is a very small file with a few characters

$ md5sum file?
75cdbfeb70a06d42210938da88c42991  file1
6fbe37f1eea0f802bd792ea885cd03e2  file2
```

Crypto Hash Design

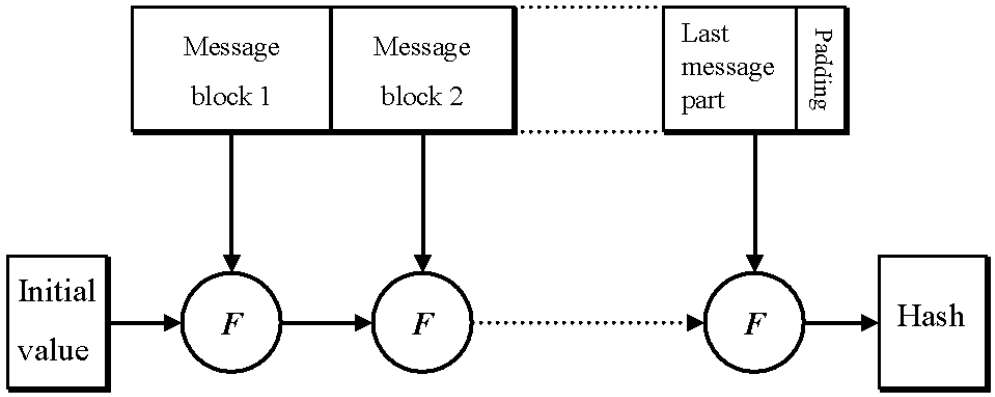
- No collisions
  - Then we say the hash function is secure
  - Change in input should not be correlated with output
- Desired property: avalanche effect
  - Change to 1 bit of input should affect about half of output bits
  - Avalanche effect should occur after few rounds
- Efficiency
  - No efficiency, no meaning for signing appl
- Crypto hash functions consist of some number of rounds
  - Analogous to design of block ciphers
  - Similar trade-offs as the iterated block cipher
  - Want security and speed but simple rounds

Popular Crypto Hashes

- MD(Message Digest) 5
  - invented by Rivest
  - 128 bit output
  - MD2 → MD4 → MD5
  - MD2 and MD4 are no longer secure, due to collision found
  - Note: even MD5, collision recently found
- SHA(Secure Hash Algorithm)-1
  - A US government standard (similar to MD5)
  - "The world's most popular hash function"
  - 160 bit output (= 20 byte output)
  - SHA-0 → SHA-1 → SHA-2 (256/512 bits output) → SHA-3
- Many others hashes, but MD5 and SHA-1 most widely used

How is Hashing Different form Encryption?

- Hashing:
  - One-way function
  - A fixed-length value for any inputs
  - …



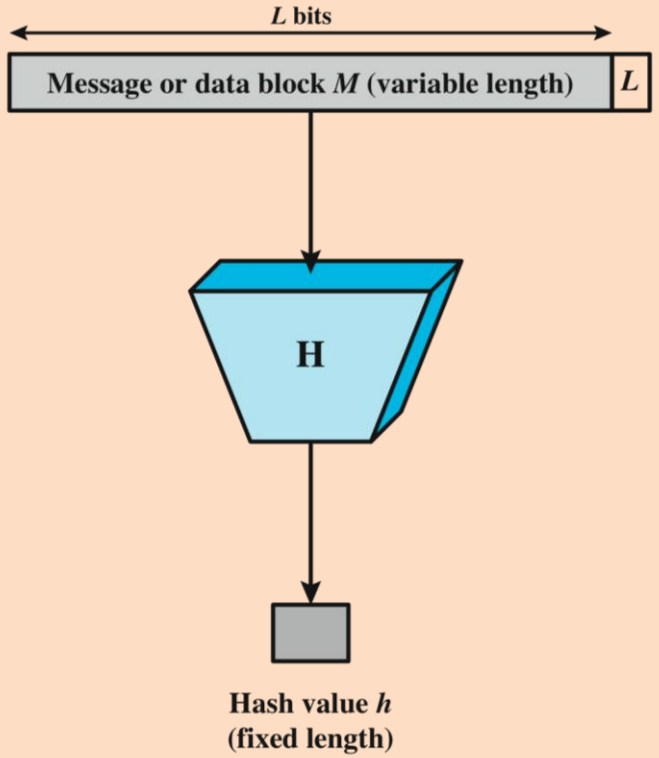


Figure 2.5 Block Diagram of Secure Hash Function; $h = H(M)$

- Encryption:
  - Two-way function
  - Result size proportional to input size
  - …

Hash Uses

- **Bitcoin**: Miners try to combine all of the inputs with their own arbitrary piece of input data in such a way that the resulting hash starts with a certain number of zeroes.
- https://www.coindesk.com/bitcoin-hash-functions-explained/

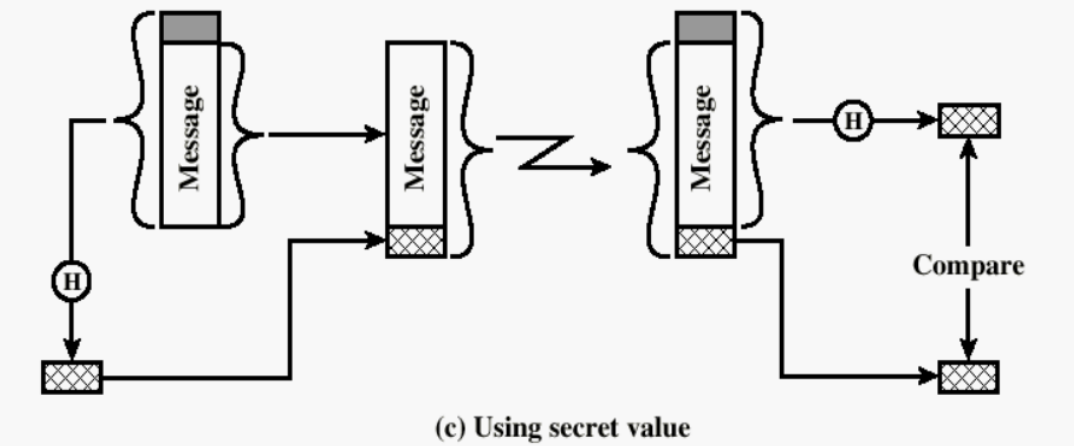
```
>>> hash("CoinDesk rocks!!")
66925f1da83c54354da73d81e013974d
>>> hash("CoinDesk rocks!!!")
c8de96b4cf781a6373766c668ceac0f0
>>> hash("CoinDesk rocks!!!!")
9ea367cea6a2cc4a6f5a1d9a334d0d9e
>>> hash("CoinDesk rocks!!!!!")
b8d43387d98f035e2f0ac49740a5af38
>>> hash("CoinDesk rocks!!!!!!")
0fe46518541f4739613b9ce29ecea6b6 => SOLVED!
```

Hash Uses

- Authentication (HMAC)
  - Password storage
- Message integrity (HMAC)
  - Blockchain
- Message fingerprint
- Data corruption detection
  - File integrity verification
- Digital signature efficiency

- Anything you can do with symmetric crypto ?

- ***Right figure***
  - Secret value is added before the hash, and removed before transmission
  - A secret shared key can be a Secret value



(c) Using secret value

Online Auction

- Suppose Alice, Bob and Charlie are bidders
- Alice plans to bid A, Bob B and Charlie C
- They don't trust that bids will stay secret

- Solution?
  - Alice, Bob, Charlie submit hashes h(A), h(B), h(C)
  - All hashes received and posted online
  - Then bids A, B and C revealed

- Hashes don't reveal bids (one way)
- Can't change bid after hash sent (collision)

Summary, Q & A

- Cryptographic Hash Functions for Integrity
- Birthday Paradox

Algorithm and variant		Output size (bits)	Internal state size (bits)	Block size (bits)	Max message size (bits)	Rounds	Operations	Security bits (Info)
MD5 (as reference)		128	128 (4 × 32)	512	Unlimited[2]	64	And, Xor, Rot, Add (mod 2^{32}), Or	<64 (collisions found)
SHA-0		160	160 (5 × 32)	512	$2^{64} - 1$	80	And, Xor, Rot, Add (mod 2^{32}), Or	<34 (collisions found)
SHA-1								<63 (collisions found[3])
SHA-2	*SHA-224*	224	256 (8 × 32)	512	$2^{64} - 1$	64	And, Xor, Rot, Add (mod 2^{32}), Or, Shr	112
	SHA-256	256						128
	SHA-384	384	512 (8 × 64)	1024	$2^{128} - 1$	80	And, Xor, Rot, Add (mod 2^{64}), Or, Shr	192
	SHA-512	512						256
	SHA-512/224	224						112
	SHA-512/256	256						128
SHA-3	*SHA3-224*	224	1600 (5 × 5 × 64)	1152	Unlimited[4]	24[5]	And, Xor, Rot, Not	112
	SHA3-256	256		1088				128