Chapter 11

Cryptographic Hash Functions

1

Hash Functions

- A hash function H accepts a variable-length block of data *M* as input and produces a fixed-size hash value
  - $h = H(M)$
  - Principal object is data integrity
- Cryptographic hash function
  - An algorithm for which it is computationally infeasible to find either:

    (a) a data object that maps to a pre-specified hash result (the one-way property)

    (b) two data objects that map to the same hash result (the collision-free property)

2

Hash Functions

- Cryptographic hash functions are keyless cryptographic algorithms
  - There are however some "keyed" hash functions.

3

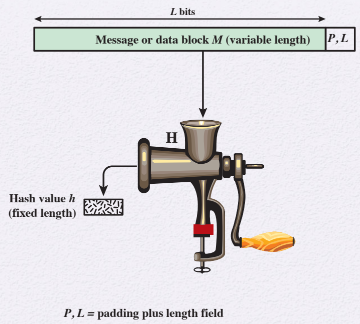


Figure 11.1 Cryptographic Hash Function; $h = \mathrm{H}(M)$

4

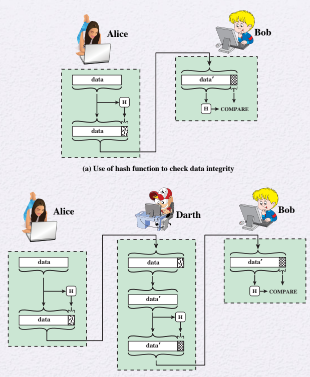


Figure 11.2 Attack Against Hash Function

5

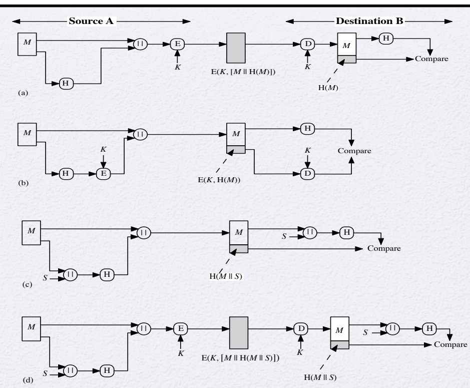


Figure 11.3 Simplified Examples of the Use of a Hash Function for Message Authentication

6

Message Authentication Code (MAC)

- Also known as a *keyed hash function*
- Typically used between two parties that share a secret key to authenticate information exchanged between those parties

Takes as input a secret key and a data block and produces a hash value (MAC) which is associated with the protected message

- If the integrity of the message needs to be checked, the MAC function can be applied to the message and the result compared with the associated MAC value
- An attacker who alters the message will be unable to alter the associated MAC value without knowledge of the secret key

7

Digital Signature

- Operation is similar to that of the MAC
- The hash value of a message is encrypted with a user's private key
- Anyone who knows the user's public key can verify the integrity of the message
- An attacker who wishes to alter the message would need to know the user's private key
- Implications of digital signatures go beyond just message authentication

8

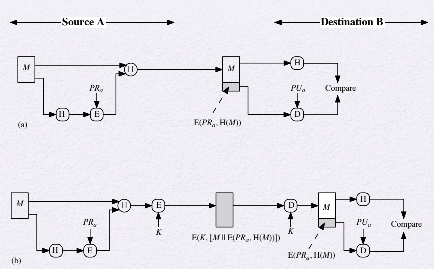


Figure 11.4 Simplified Examples of Digital Signatures

9

Other Hash Function Uses

Commonly used to create a one-way password file

- When a user enters a password, the hash of that password is compared to the stored hash value for verification
- This approach to password protection is used by most operating systems

Can be used for intrusion and virus detection

- Store H(F) for each file on a system and secure the hash values
- One can later determine if a file has been modified by recomputing H(F)
- An intruder would need to change F without changing H(F)

Can be used to construct a pseudorandom function (PRF) or a pseudorandom number generator (PRNG)

- A common application for a hash-based PRF is for the generation of symmetric keys

10

Two Simple Hash Functions

- Consider two simple insecure hash functions that operate using the following general principles:
  - The input is viewed as a sequence of *n*-bit blocks
  - The input is processed one block at a time in an iterative fashion to produce an *n*-bit hash function
- Bit-by-bit exclusive-OR (XOR) of every block
  - $C_i = b_{i1}$ *xor* b_{i2} *xor . . . xor* b_{im}
  - Produces a simple parity for each bit position and is known as a longitudinal redundancy check
  - Reasonably effective for random data as a data integrity check
- Perform a one-bit circular shift on the hash value after each block is processed
  - Has the effect of randomizing the input more completely and overcoming any regularities that appear in the input

11

Two Simple Hash Functions

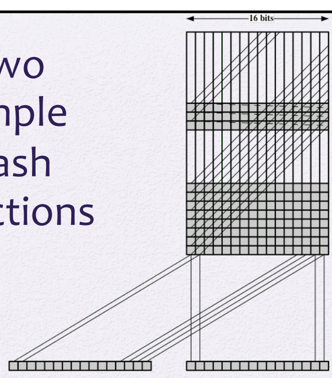


Figure 11.5 Two Simple Hash Functions

12

Requirements and Security

Preimage

- *x* is the preimage of *h* for a hash value *h* = H(*x*)
- Is a data block whose hash function, using the function H, is *h*
- Because H is a many-to-one mapping, for any given hash value *h*, there will in general be multiple preimages

Collision

- Occurs if we have *x* ≠ *y* and H(*x*) = H(*y*)
- Because we are using hash functions for data integrity, collisions are clearly undesirable

13

Table 11.1

Requirements for a Cryptographic Hash Function H

Requirement	Description
Variable input size	H can be applied to a block of data of any size.
Fixed output size	H produces a fixed-length output.
Efficiency	H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.
Preimage resistant (one-way property)	For any given hash value h, it is computationally infeasible to find y such that H(y) = h.
Second preimage resistant (weak collision resistant)	For any given block x, it is computationally infeasible to find $y \neq x$ with H(y) = H(x).
Collision resistant (strong collision resistant)	It is computationally infeasible to find any pair (x, y) such that H(x) = H(y).
Pseudorandomness	Output of H meets standard tests for pseudorandomness

(Table can be found on page 327 in textbook.)

14

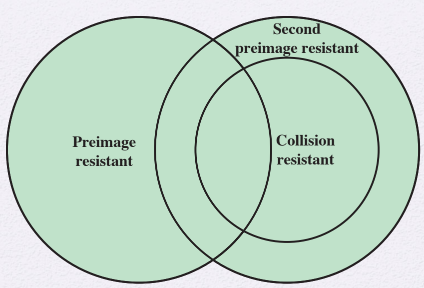


Figure 11.6 Relationship Among Hash Function Properties

15

Table 11.2

Hash Function Resistance Properties Required for Various Data Integrity Applications

	Preimage Resistant	Second Preimage Resistant	Collision Resistant
Hash + digital signature	yes	yes	yes*
Intrusion detection and virus detection		yes	
Hash + symmetric encryption			
One-way password file	yes		
MAC	yes	yes	yes*

* Resistance required if attacker is able to mount a chosen message attack

16

Attacks on Hash Functions

Brute-Force Attacks

- Does not depend on the specific algorithm, only depends on bit length
- In the case of a hash function, attack depends only on the bit length of the hash value
- Method is to pick values at random and try each one until a collision occurs

Cryptanalysis

- An attack based on weaknesses in a particular cryptographic algorithm
- Seek to exploit some property of the algorithm to perform some attack other than an exhaustive search

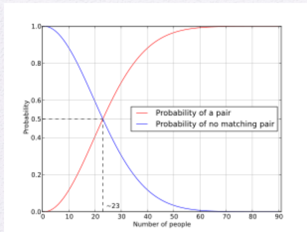
17

Collision Resistant Attacks

- For a collision resistant attack, an adversary wishes to find two messages or data blocks that yield the same hash function
  - The effort required is explained by a mathematical result referred to as the *birthday paradox*
- Yuval proposed the following strategy to exploit the birthday paradox in a collision resistant attack:
  - The source (A) is prepared to sign a legitimate message *x* by appending the appropriate *m*-bit hash code and encrypting that hash code with A's private key
  - Opponent generates $2^{m/2}$ variations *x'* of *x*, all with essentially the same meaning, and stores the messages and their hash values
  - Opponent prepares a fraudulent message *y* for which A's signature is desired
  - Opponent generates minor variations y' of *y*, all of which convey essentially the same meaning. For each y', the opponent computes H (y'), checks for matches with any of the H (x') values, and continues until a match is found. That is, the process continues until a y' is generated with a hash value equal to the hash value of one of the x' values
  - The opponent offers the valid variation to A for signature which can then be attached to the fraudulent variation for transmission to the intended recipient
    - Because the two variations have the same hash code, they will produce the same signature and the opponent is assured of success even though the encryption key is not known

18

Birthday Paradox



- 23 people – 50% chance
- 70 people – Over 99% chance

19

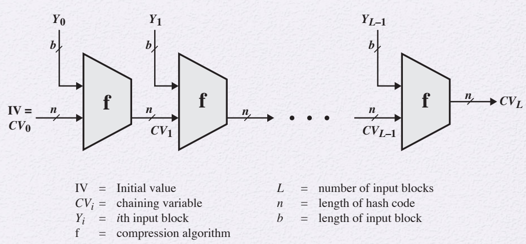


Figure 11.8 General Structure of Secure Hash Code

20

Secure Hash Algorithm (SHA)

- SHA was originally designed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993
- Was revised in 1995 as SHA-1
- Based on the hash function MD4 and its design closely models MD4
- Produces 160-bit hash values
- In 2002 NIST produced a revised version of the standard that defined three new versions of SHA with hash value lengths of 256, 384, and 512
  - Collectively known as SHA-2

21

Comparison of SHA functions

view · talk · edit

Algorithm and variant		Output size (bits)	Internal state size (bits)	Block size (bits)	Rounds	Operations	Security against collision attacks (bits)	Security against length extension attacks (bits)	Performance on Skylake (median cpb)[1] Long messages	8 bytes	First published
MD5 (as reference)		128	128 (4 × 32)	512	4 (16 operations in each round)	And, Xor, Or, Rot, Add (mod 2^{32})	≤ 18 (collisions found)[2]	0	4.99	55.00	1992
SHA-0		160	160 (5 × 32)	512	80	And, Xor, Or, Rot, Add (mod 2^{32})	< 34 (collisions found)	0	≈ SHA-1	≈ SHA-1	1993
SHA-1							< 63 (collisions found)[3]		3.47	52.00	1995
SHA-2	*SHA-224* *SHA-256*	224 256	256 (8 × 32)	512	64	And, Xor, Or, Rot, Shr, Add (mod 2^{32})	112 128	32 0	7.62 7.63	84.50 85.25	2004 2001
	SHA-384	384	512 (8 × 64)	1024	80	And, Xor, Or, Rot, Shr, Add (mod 2^{64})	192	128	5.12	135.75	2001
	SHA-512	512					256	0[4]	5.06	135.50	2001
	SHA-512/224 *SHA-512/256*	224 256					112 128	288 256	≈ SHA-384	≈ SHA-384	2012
SHA-3	*SHA3-224* *SHA3-256* *SHA3-384* *SHA3-512*	224 256 384 512	1600 (5 × 5 × 64)	1152 1088 832 576	24[5]	And, Xor, Rot, Not	112 128 192 256	448 512 768 1024	8.12 8.59 11.06 15.88	154.25 155.50 164.00 164.00	2015
	SHAKE128 *SHAKE256*	*d* (arbitrary) *d* (arbitrary)		1344 1088			min(*d*/2, 128) min(*d*/2, 256)	256 512	7.08 8.59	155.25 155.50	

22

Table 11.3
Comparison of SHA Parameters

Algorithm	Message Size	Block Size	Word Size	Message Digest Size
SHA-1	$< 2^{64}$	512	32	160
SHA-224	$< 2^{64}$	512	32	224
SHA-256	$< 2^{64}$	512	32	256
SHA-384	$< 2^{128}$	1024	64	384
SHA-512	$< 2^{128}$	1024	64	512
SHA-512/224	$< 2^{128}$	1024	64	224
SHA-512/256	$< 2^{128}$	1024	64	256

Note: All sizes are measured in bits.

23

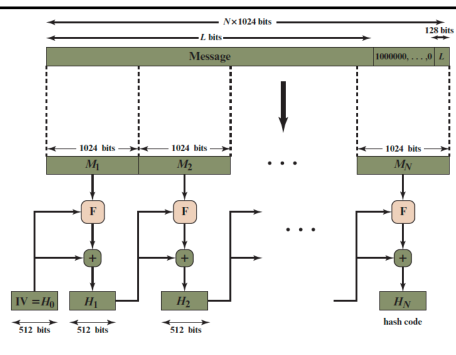


Figure 11.9 Message Digest Generation Using SHA-512

24

Summary

- Summarize the applications of cryptographic hash functions
- Explain why a hash function used for message authentication needs to be secured

- Understand the differences among preimage resistant, second preimage resistant, and collision resistant properties
- Present an overview of the basic structure of cryptographic hash functions
- Describe how cipherblock chaining can be used to construct a hash function

25