IT Security Hashing and MAC

Michael Claudius, Associate Professor, Roskilde

29.08.2024

Problems with Integrity and Authentication

- **Make sure the message/documents (m) has not been changed/tampered**
- **Make sure sender is sender (Bob) and receiver is receiver (Alice)**
- **Lets look a little deeper into the problem and solutions**
- **What we need is a fingerprint/DNA of message/document**
- **This is done by using a cryptographic hash function (H)**
- **H produce a hash value H(m)**

- **Special Issues**
- **Find a fast and secure hash function, i.e. high security strength**
- **Length extension attacks**
- **Playback attacks**

Security Strengths of Approved Hash Functions

- **An approved hash function is expected to have the following three properties:**

- **Collision resistance**: It is computationally infeasible to find two different inputs (m1, m2) to the hash function that have the same hash value.
- **Preimage resistance**: Given a randomly chosen hash value, it is computationally infeasible to find an input message that hashes to this hash value.
- **Second preimage resistance**: It is computationally infeasible to find a second input that has the same hash value as any other specified input.

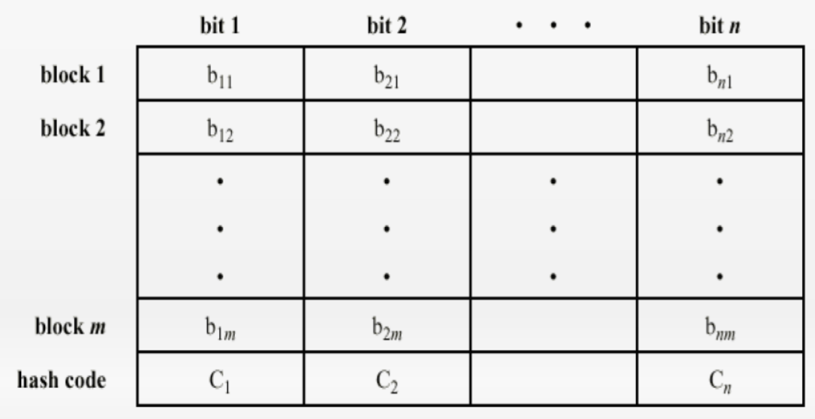
Resistance types

- **The difference the choice/knowledge of message m1, m2**

- In the first case (collision resistance), the attacker can **freely choose both messages** m1 and m2, with the only requirement that they are different (and hash to the same value).
- In the second case the attacker is handed a random H(m1) value and tries to find m1.
- In the third case (second preimage resistance), the attacker is **handed a fixed** m1, H(m1) to which he has to find a different m2 with equal hash. In particular, he **can't choose** m1.

Simple Hash Function

- **Simple Hash function using bit-wise XOR on one block**



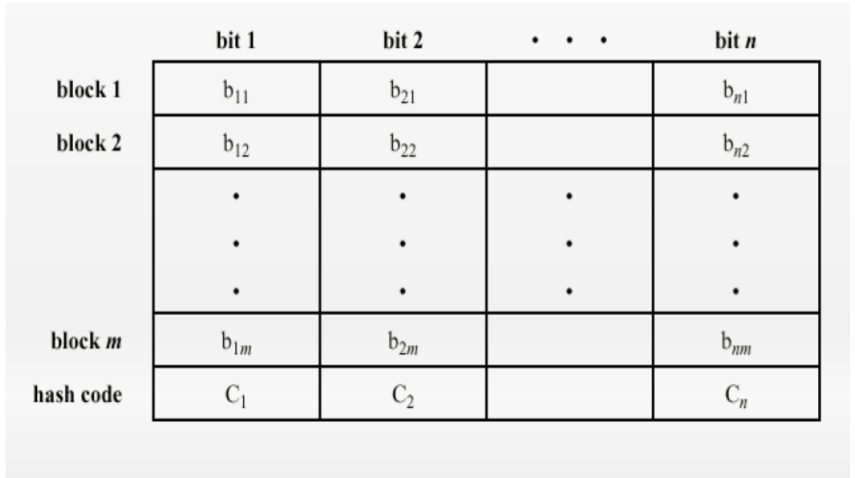
	bit 1	bit 2	• • •	bit n
block 1	b_{11}	b_{21}		b_{n1}
block 2	b_{12}	b_{22}		b_{n2}
	⋮	⋮	⋮	⋮
block m	b_{1m}	b_{2m}		b_{nm}
hash code	C_1	C_2		C_n

Figure 3.3 Simple Hash Function Using Bitwise XOR

- **Improve by one bit circular shift after the block has been processed**
- **For each n-bit block of data:**
  - **Rotate the current hash vqlue to the left by one bit**
  - **XOR the block into that hash.**

Simple Hash Function

- **Simple Hash function using bit-wise XOR on one block**



	bit 1	bit 2	• • •	bit n
block 1	b_{11}	b_{21}		b_{n1}
block 2	b_{12}	b_{22}		b_{n2}
	• • •	• • •	• • •	• • •
block m	b_{1m}	b_{2m}		b_{nm}
hash code	C_1	C_2		C_n

Figure 3.3 Simple Hash Function Using Bitwise XOR

- **Improve by one bit circular shift after the block has been processed**
- **For each n-bit block of data:**
  - **Rotate the current hash value to the left by one bit**
  - **XOR the block into that hash.**

SHA512 processing on a message

- **Message M is divided into blocks (M1, M2, ….) of 1024 bit length**
- **The last block is padded by 10000.. to adjust the size**
- **L is the length of the message 2^128 bit max**
- **Each block M1..Mn is manipulated by a function F**
- **Outcome is of F is added with previous hash (Hi-1)**
- **+ means word-by-word addition modulus 2^64**

- **Final output is the 512 bit message digest a hash value**

- **So whats inside the function ??**

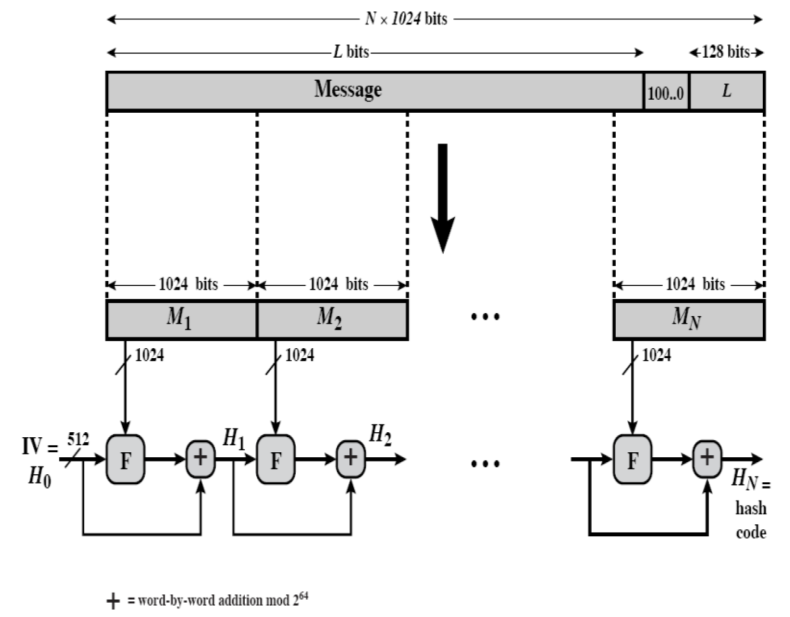


Figure 3.4 Message Digest Generation Using SHA-512

SHA512 function on one block

- **Message M is divided into blocks (M1, M2, ….) of 1024 bit length**
- **Each block is distributed into 80 words (W0…W79)**
- **Copy block (1024) into the first 16 words (16x64=1024) W0..W15**
- **Extend/manipulate the 16 words into the last W16..W79**
- **Previous hash (Hi-1) is placed in registers abcdef of 64 bit length**
- **80 rounds is done**
- **Each round using a constant Ki, a word Wi, abcdef**
  - **A round is a sequence of operations performed multiple times**
  - **to thoroughly mix up the data until it's unrecognizable**.
- **K0..K79 is the fractional cube root of the first 80 primes: 1,2,5,7…**
- **Outcome round 79 is added with previous hash (Hi-1)**
- **+ means word-by-word addition modulus 2^64**

- **Final output is the 512 bit message digest**
- **Every bit is a function of all input bits !**

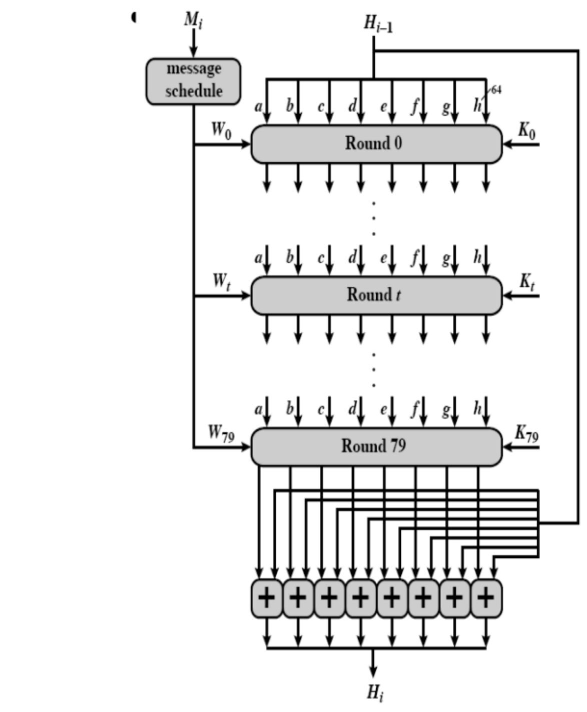


Figure 3.5 SHA-512 Processing of a Single 1024-Bit Block

SHA algorithms

- **Comparing of SHA parameters**

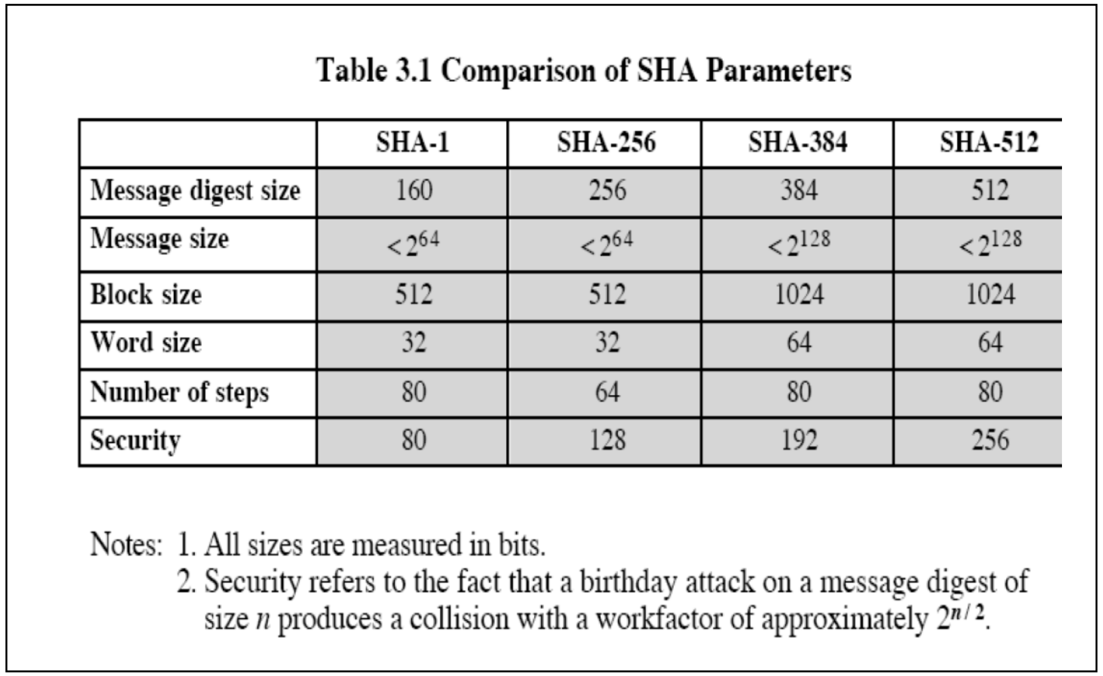


Table 3.1 Comparison of SHA Parameters

	SHA-1	SHA-256	SHA-384	SHA-512
Message digest size	160	256	384	512
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	1024	1024
Word size	32	32	64	64
Number of steps	80	64	80	80
Security	80	128	192	256

Notes: 1. All sizes are measured in bits.
2. Security refers to the fact that a birthday attack on a message digest of size n produces a collision with a workfactor of approximately $2^{n/2}$.

- **The higher digest size the better security and lower speed**
- **BUT sensitive to length extension attacks, lets look at that**

Zealand

MAC Message Authentication Code

- **If A sends m,H(m) to B**
- **Intruder Trudy can easily send another message m’,H(m‘) claiming she is A**
- **Need a secret key K: s**
- **Create a MAC = H(m+s)**

- **But its also sensitive to length attacks**
- **Lets look at that**

  - https://en.wikipedia.org/wiki/Length_extension_attack

- **Solution HMAC**

HMAC Hash Message Authentication Code

- **If A sends m,H(m) to B**
- **Intruder Trudy can easily send another message m’,H(m‘) claiming she is A**
- **Need a secret key K: s**
- **Create a MAC = H(m+s)**

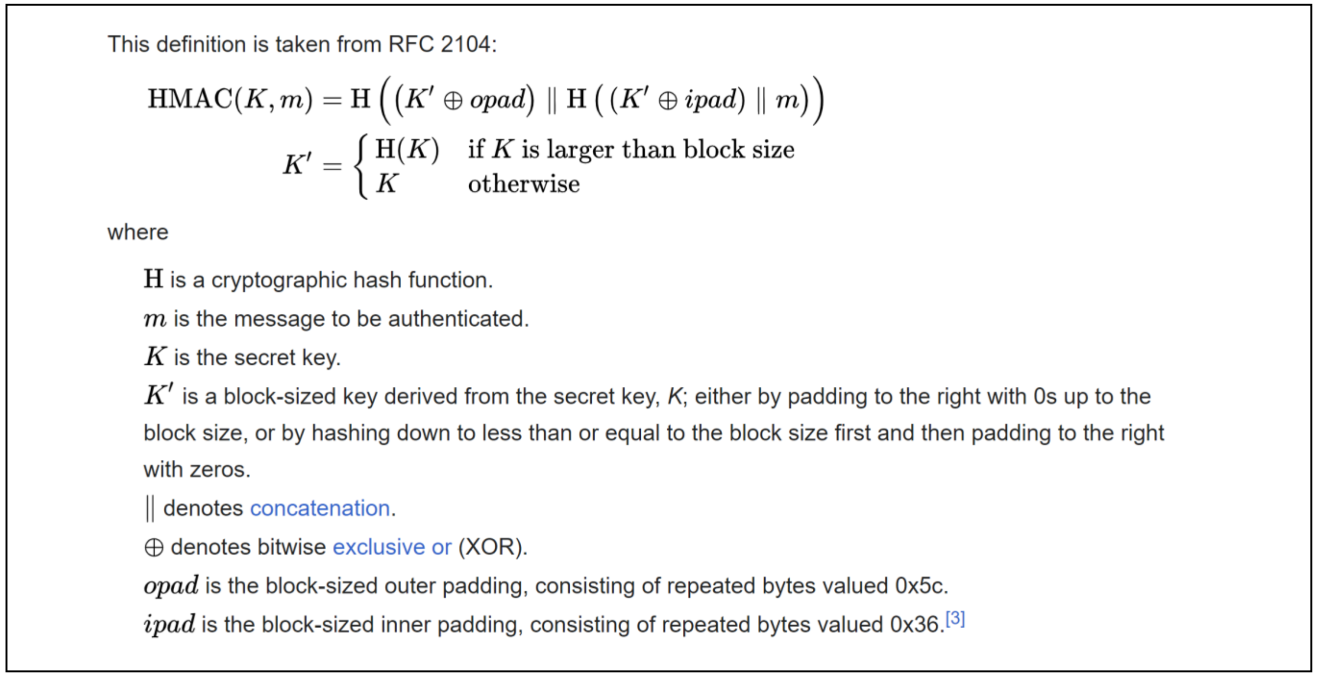
- **But its also sensitive to length attacks**
- **Lets look at that**

  - https://en.wikipedia.org/wiki/Length_extension_attack

- **Solution HMAC**

HMAC Hash Message Authentication Code

- **Definition of HMAC**



This definition is taken from RFC 2104:

$$\mathrm{HMAC}(K,m) = \mathrm{H}\Big((K' \oplus opad) \parallel \mathrm{H}\big((K' \oplus ipad) \parallel m\big)\Big)$$

$$K' = \begin{cases} \mathrm{H}(K) & \text{if } K \text{ is larger than block size} \\ K & \text{otherwise} \end{cases}$$

where

H is a cryptographic hash function.

m is the message to be authenticated.

K is the secret key.

K' is a block-sized key derived from the secret key, *K*; either by padding to the right with 0s up to the block size, or by hashing down to less than or equal to the block size first and then padding to the right with zeros.

$\parallel$ denotes concatenation.

$\oplus$ denotes bitwise exclusive or (XOR).

$opad$ is the block-sized outer padding, consisting of repeated bytes valued 0x5c.

$ipad$ is the block-sized inner padding, consisting of repeated bytes valued 0x36.[3]

- **BUT what about playback attacks?**
- **Solution is a adding a nounce R to m+s**
- **This is explained next week about length extension attacks, lets look at that**

Zealand

Exercise

- **It is time for discussion, applying hashing algorithms**
- **Also we will investigate the Homework 1 exercise !!**

- Lab Cryptool 2

  - ***Look at details, but don’t loose the overview ☺***
  - ***Just follow the ”right” track and you find the gold***