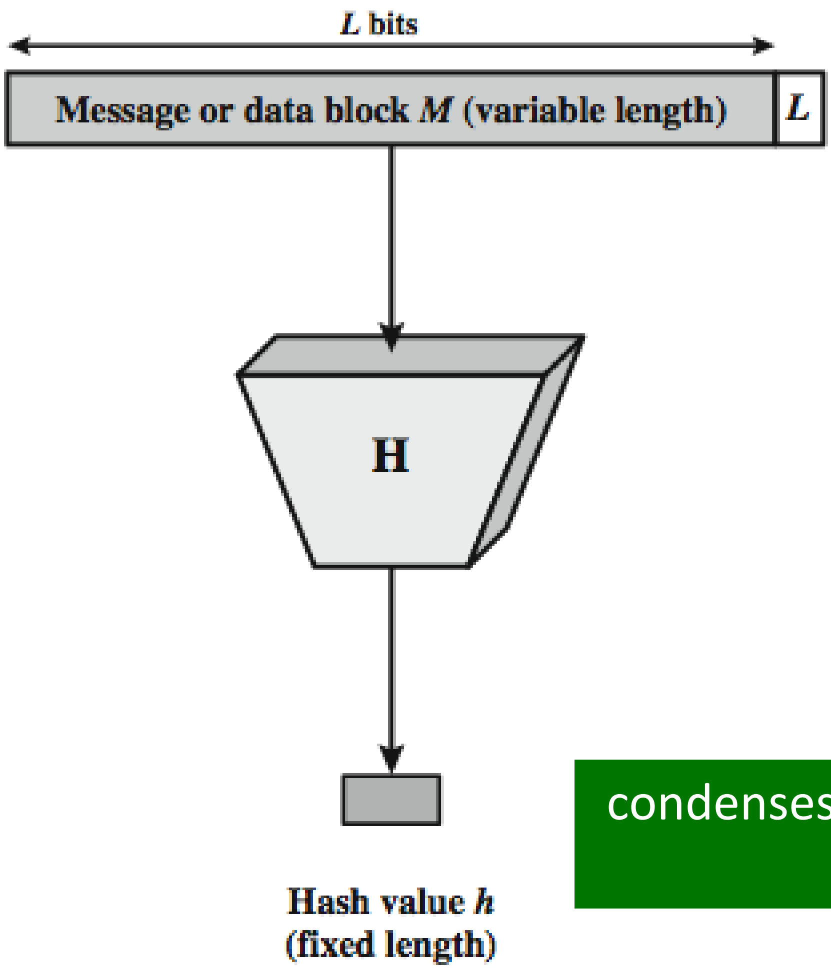
Hash Functions

BSCH4-NS
Jason Farina
Jason.farina@griffith.ie

Hash Function



- The hash value represents concisely the longer message
  - may called the *message digest*
- A message digest is as a ``digital fingerprint'' of the original document

condenses arbitrary message to fixed size

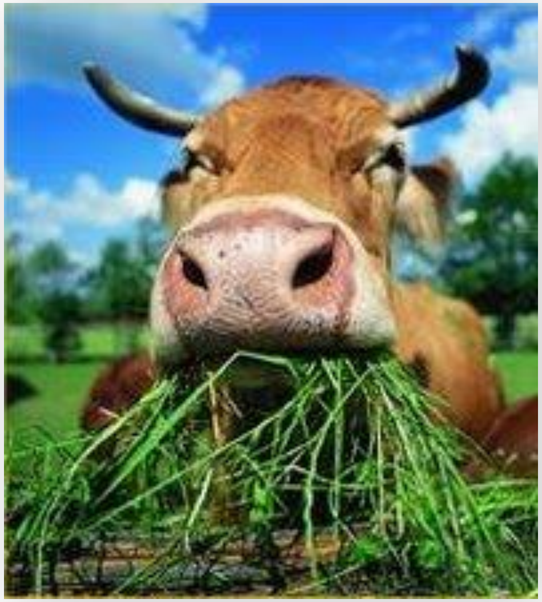
$$h = H(M)$$

Hashing

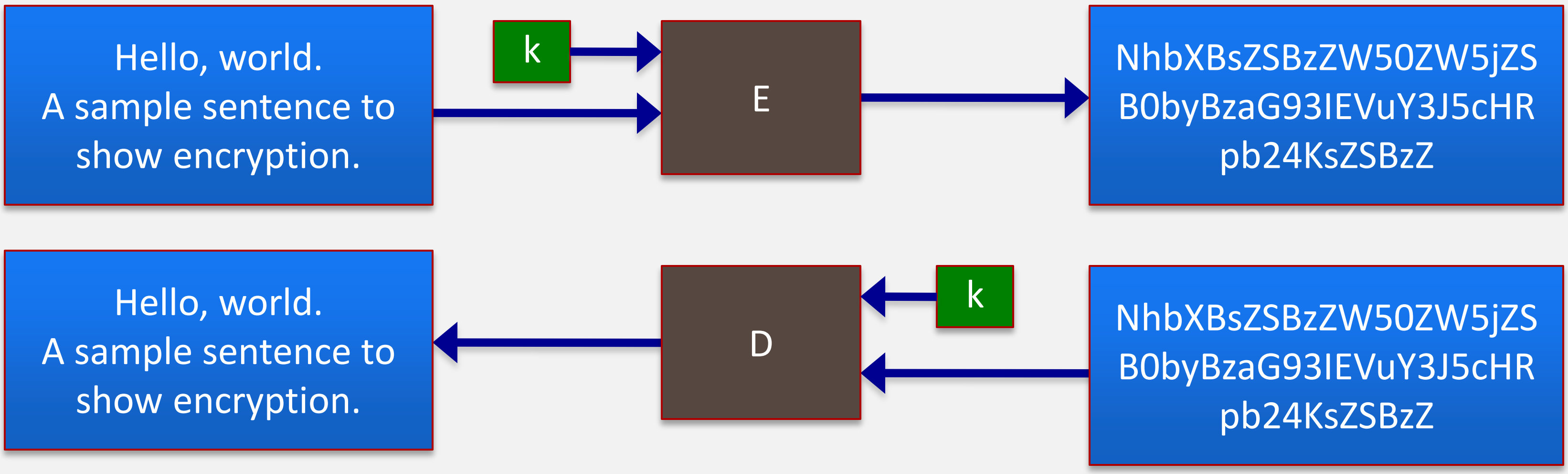
- A file hash is an algorithm that encodes a variable length message to a fixed length string.
- ABCDEFGHIJKLMNOPQRSTUVWXYZ => A
- INFOSEC IS FUN => B
- Wheeeeeeee! => C
- Given the ciphertext (A, B or C) we have no way to know how long the original plaintext message was.

Chewing functions

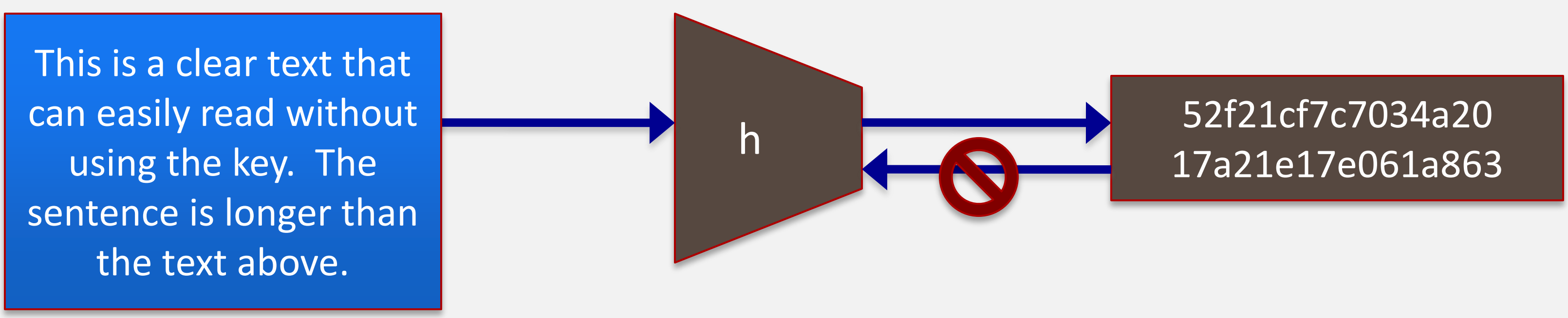
- Hashing function as “chewing” or “digest” function



Hashing V.S. Encryption



- Encryption is two way, and requires a key to encrypt/decrypt



– Hashing is one-way. There is no 'de-hashing’

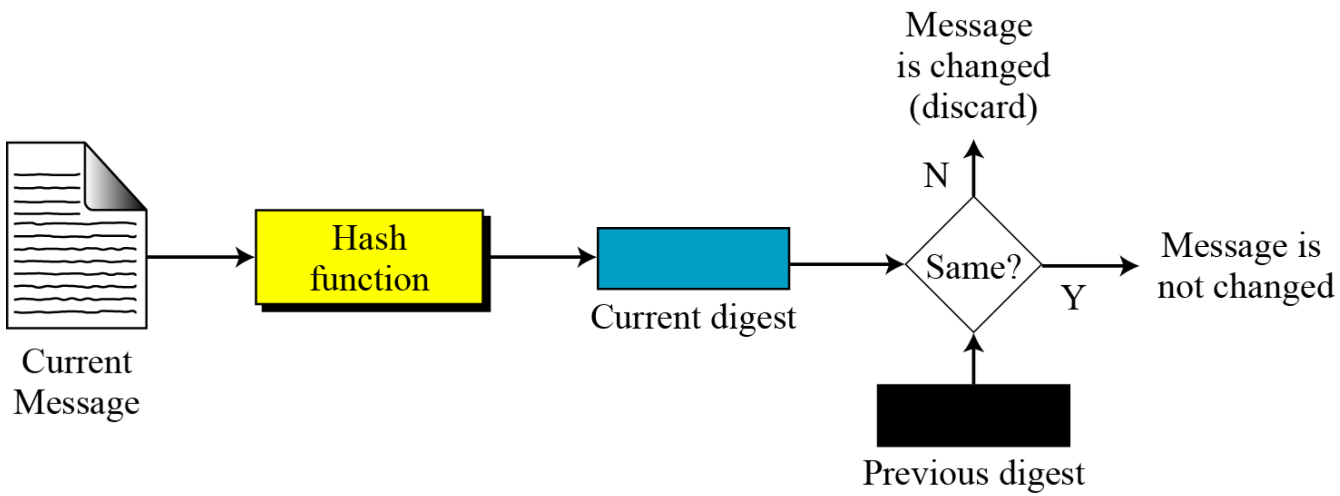
Motivation for Hash Algorithms

- Intuition
  – Re-examine the non-cryptographic checksum
  – Main Limitation
    - An attack is able to construct a message that matches the checksum
- Goal
  – Design a code where the original message can not be inferred based on its checksum
  – such that an accidental or intentional change to the message will change the hash value

Hash Function Applications

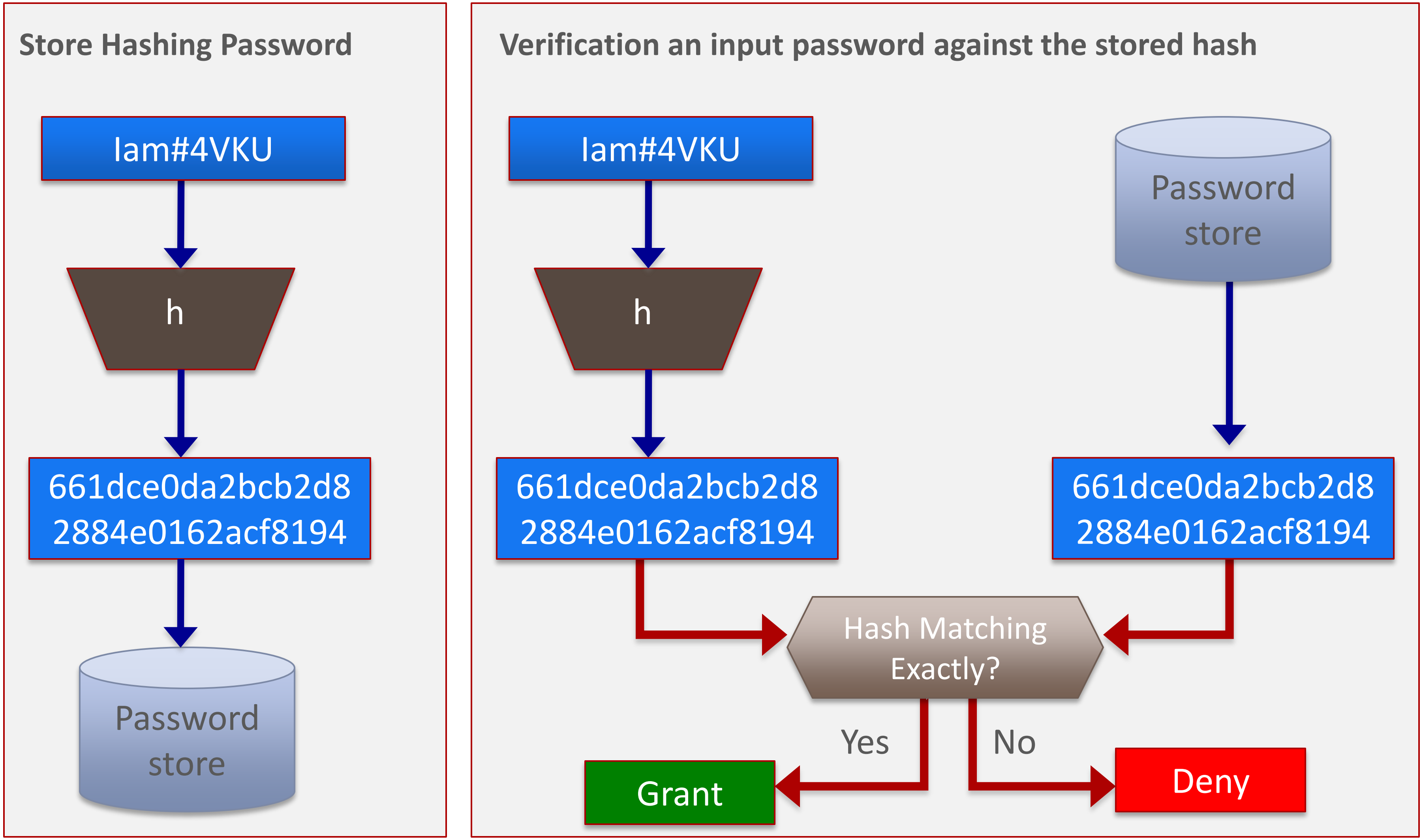
- Used Alone
  - Fingerprint -- file integrity verification, public key fingerprint
  - Password storage (one-way encryption)
- Combined with encryption functions
  - Message Authentication Code (MAC)
    - protects both a message's integrity as well as its authenticity
  - Digital signature
    - Ensuring Non-repudiation
    - Encrypt hash with private (signing) key and verify with public (verification) key

Integrity

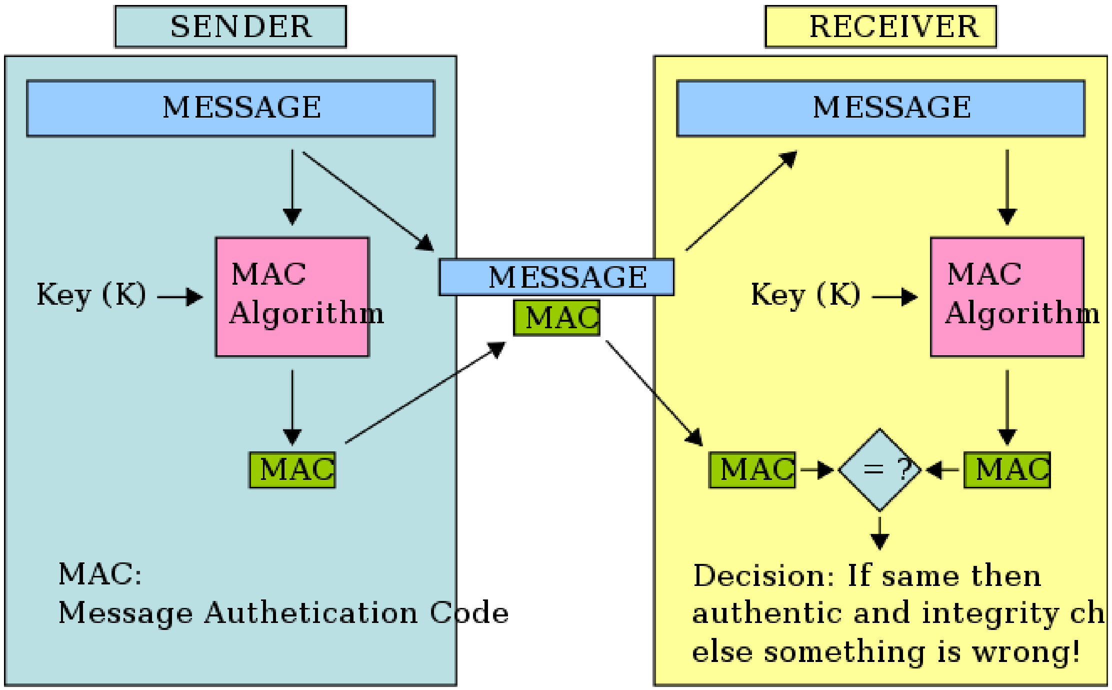


- to create a one-way password file
  - store hash of password not actual password
- for intrusion detection and virus detection
  - keep & check hash of files on system

Password Verification

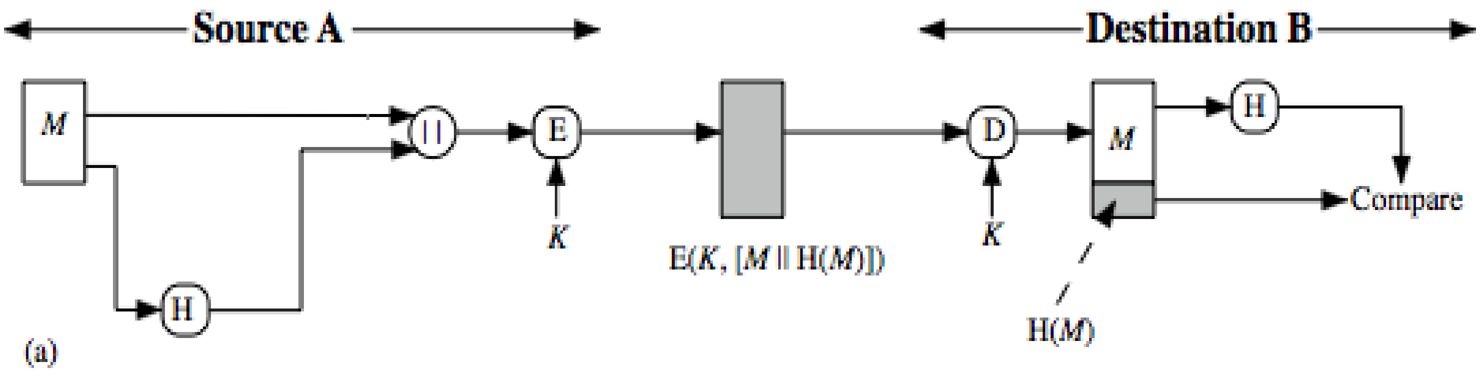


Authentication

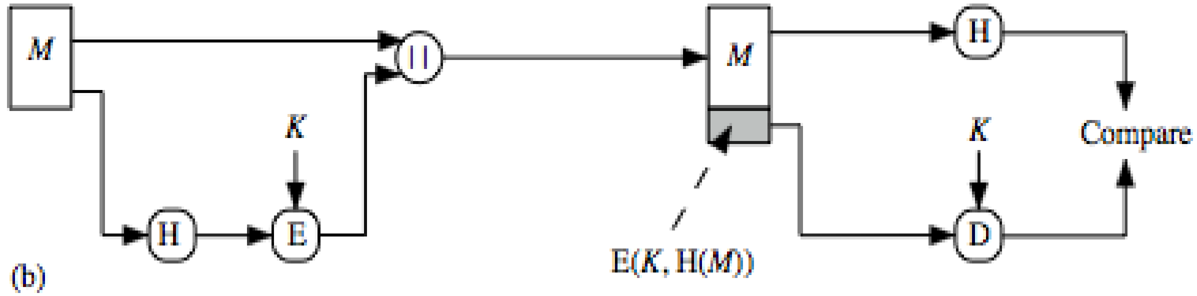


- protects both a message's integrity as well as its authenticity , by allowing verifiers (who also possess the secret key) to detect any changes to the message content

Hash Function Usages (I)

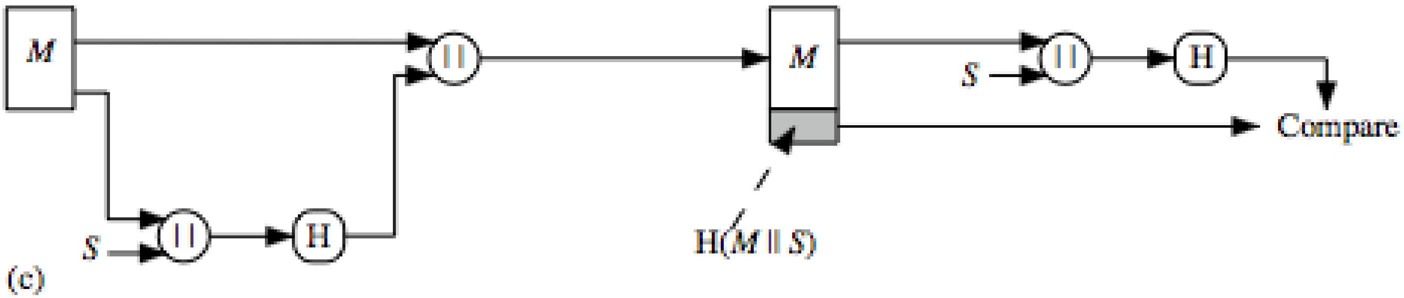


Message encrypted : Confidentiality and authentication

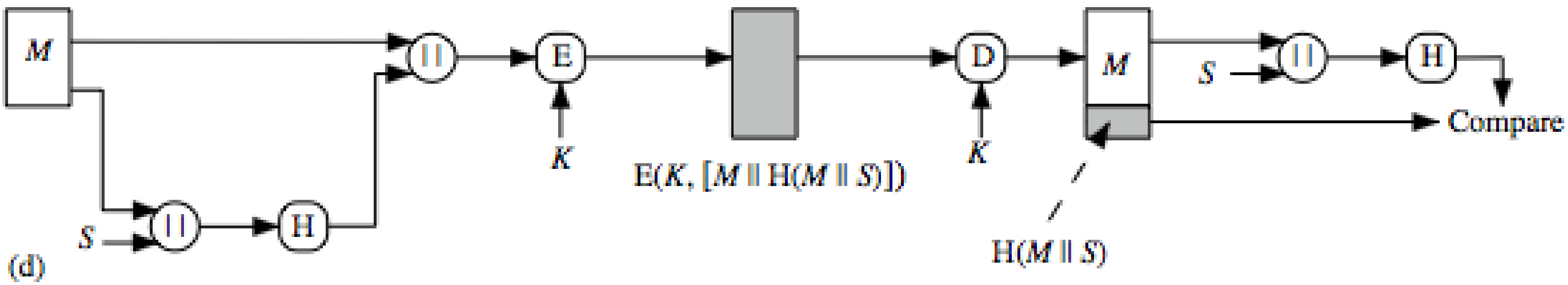


Message unencrypted: Authentication

Hash Function Usages (II)

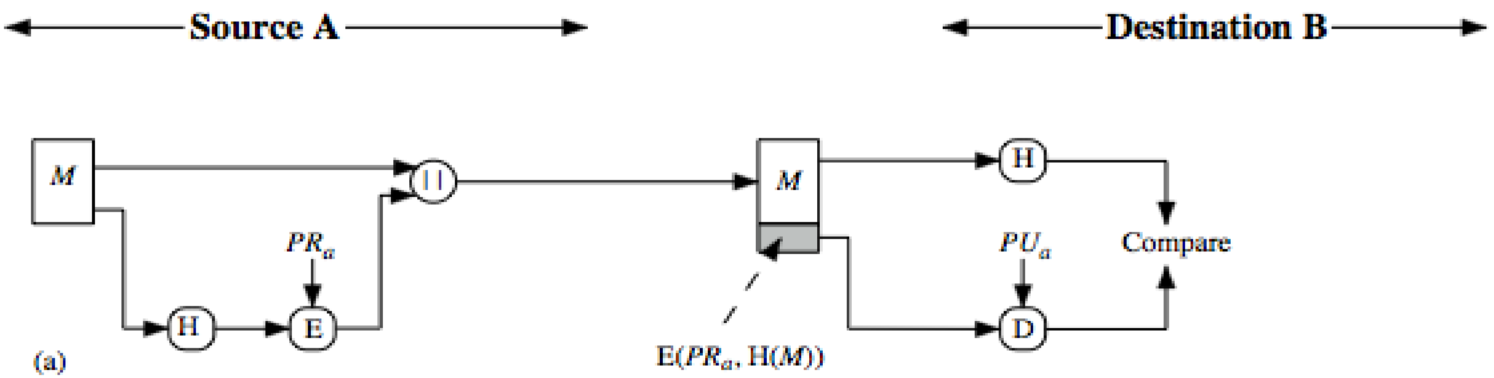


Message encrypted : Authentication (no encryption needed!)

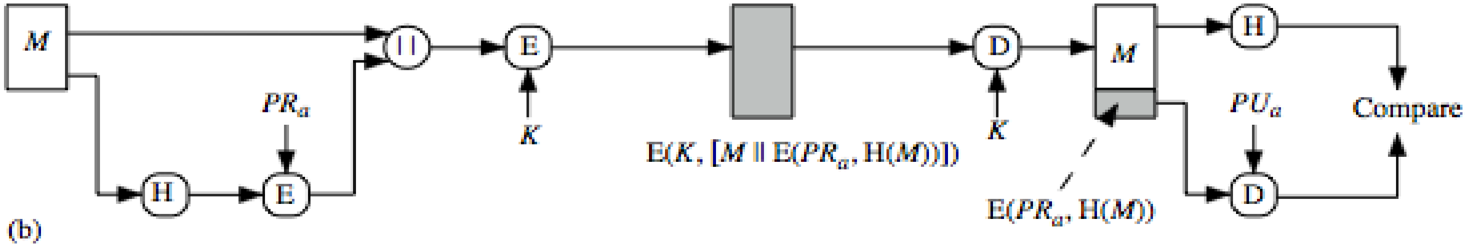


Message unencrypted: Authentication, confidentiality

Hash Function Usages (III)



Authentication, digital signature



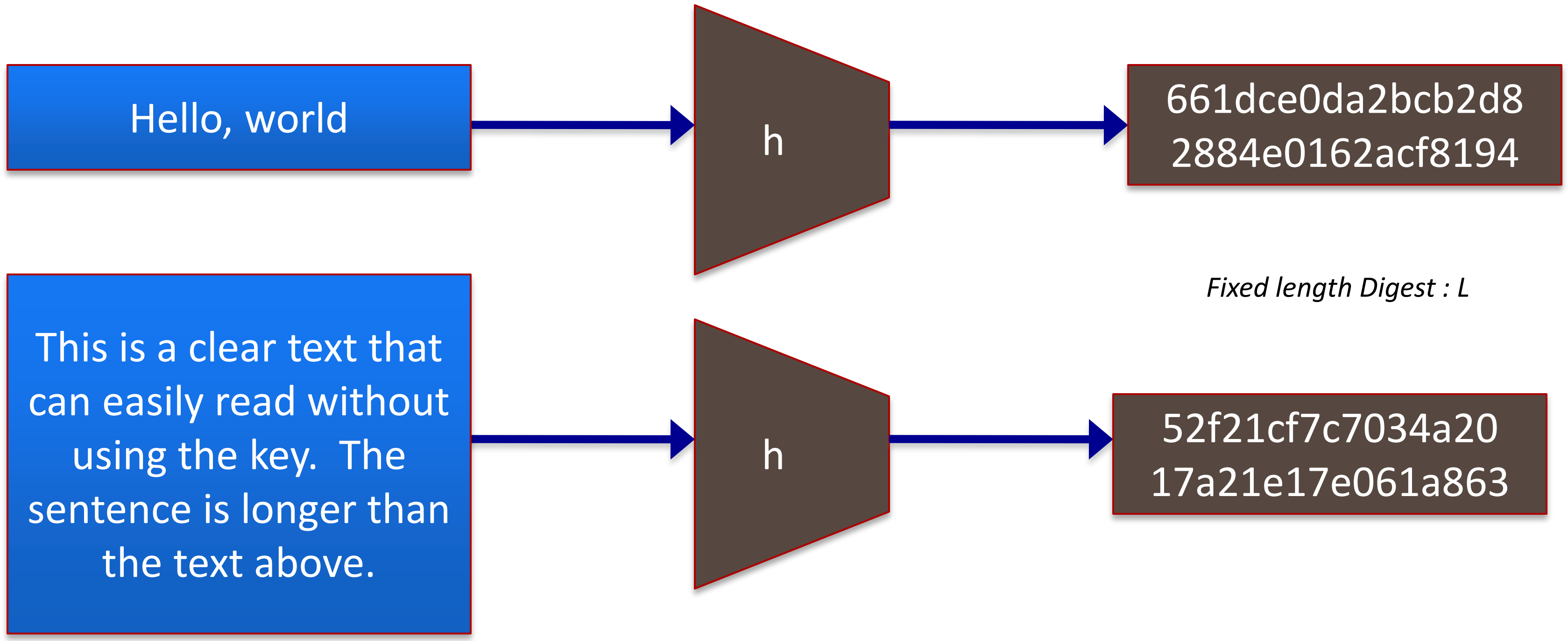
Authentication, digital signature, confidentiality

Hashing

- Common properties

1. Deterministic : the same input should produce the same output regardless of the system it is being run on.
2. Collision resistant: for any given message (m) there should not exist m1 such that h(m) = h(m1)
3. Given H(m) it should be realistically impossible to determine m
4. Any change in m should change the value of h(m)

Properties : Fixed length



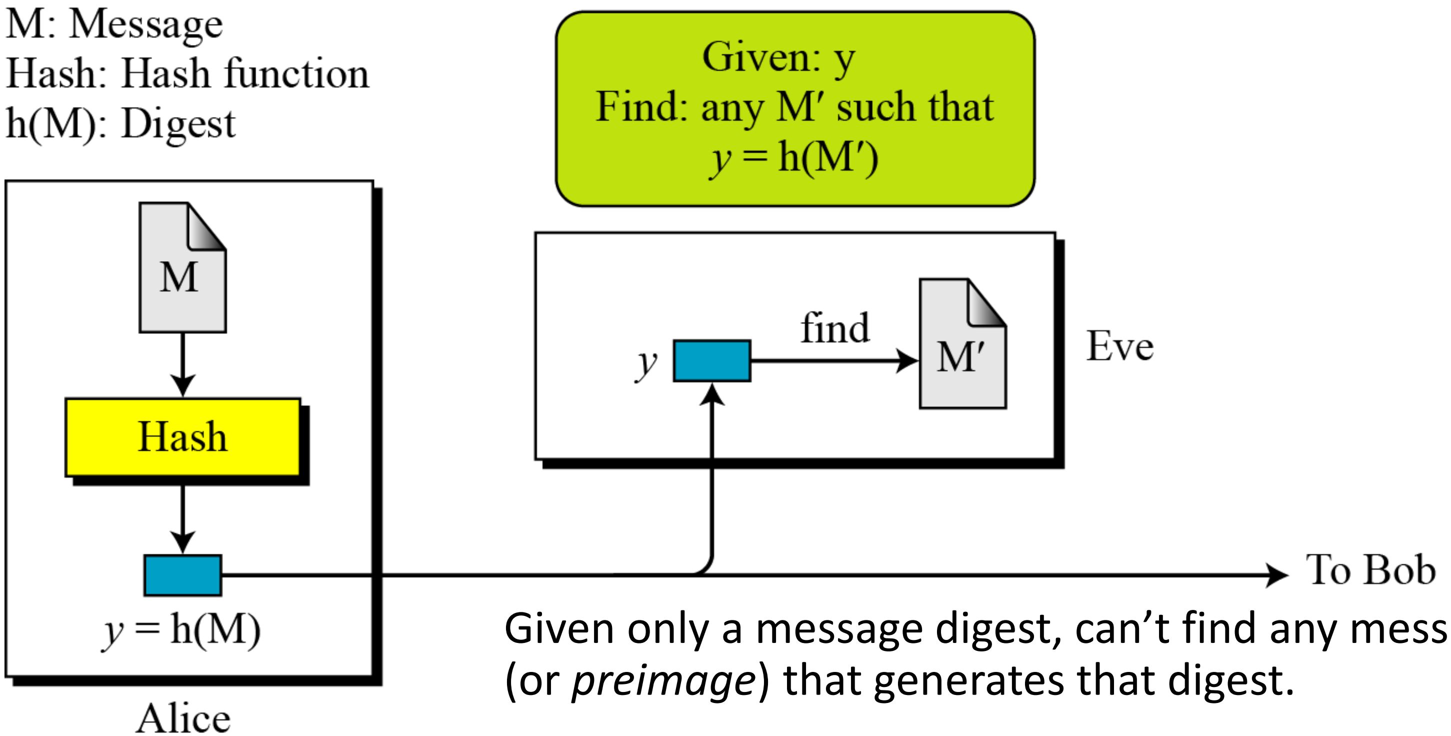
- Arbitrary-length message to fixed-length digest

Preimage resistant

- This measures how difficult to devise a message which hashes to the known digest
- Roughly speaking, the hash function must be one-way.

Preimage Attack

Given: $y = h(M)$ **Find:** M' such that $y = h(M')$



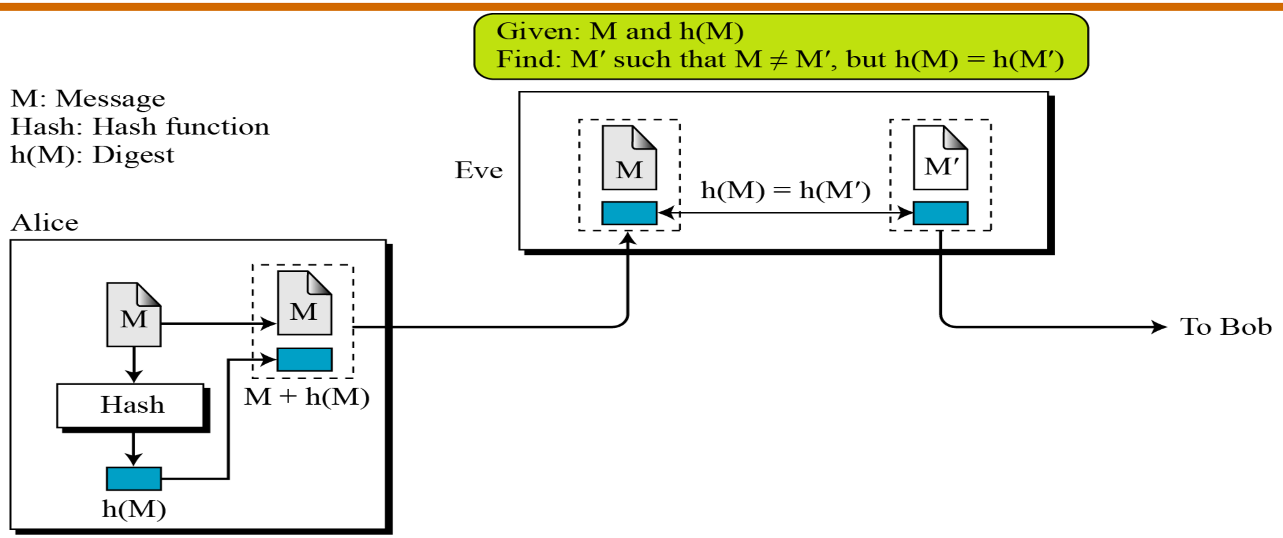
Given only a message digest, can’t find any message (or *preimage*) that generates that digest.

Second preimage resistant

- This measures how difficult to devise a message which hashes to the known digest and its message

Second Preimage Attack

Given: M and h(M) **Find: M′ ≠ M such that h(M) = h(M′)**



- Given one message, can't find another message that has the same message digest. An attack that finds a second message with the same message digest is a *second pre-image* attack.
    - It would be easy to forge new digital signatures from old signatures if the hash function used weren't second preimage resistant

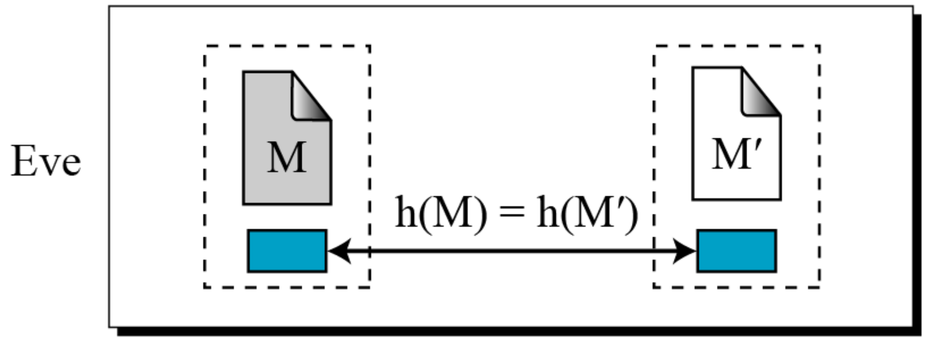
Collision Resistant

Collision Attack

Given: none | **Find: $M' \neq M$ such that $h(M) = h(M')$**

M: Message
Hash: Hash function
h(M): Digest

Find: M and M′ such that $M \neq M'$, but $h(M) = h(M')$



- Can’t find any two different messages with the same message digest
  - Collision resistance implies second preimage resistance
  - Collisions, if we could find them, would give signatories a way to repudiate their signatures

Hashing

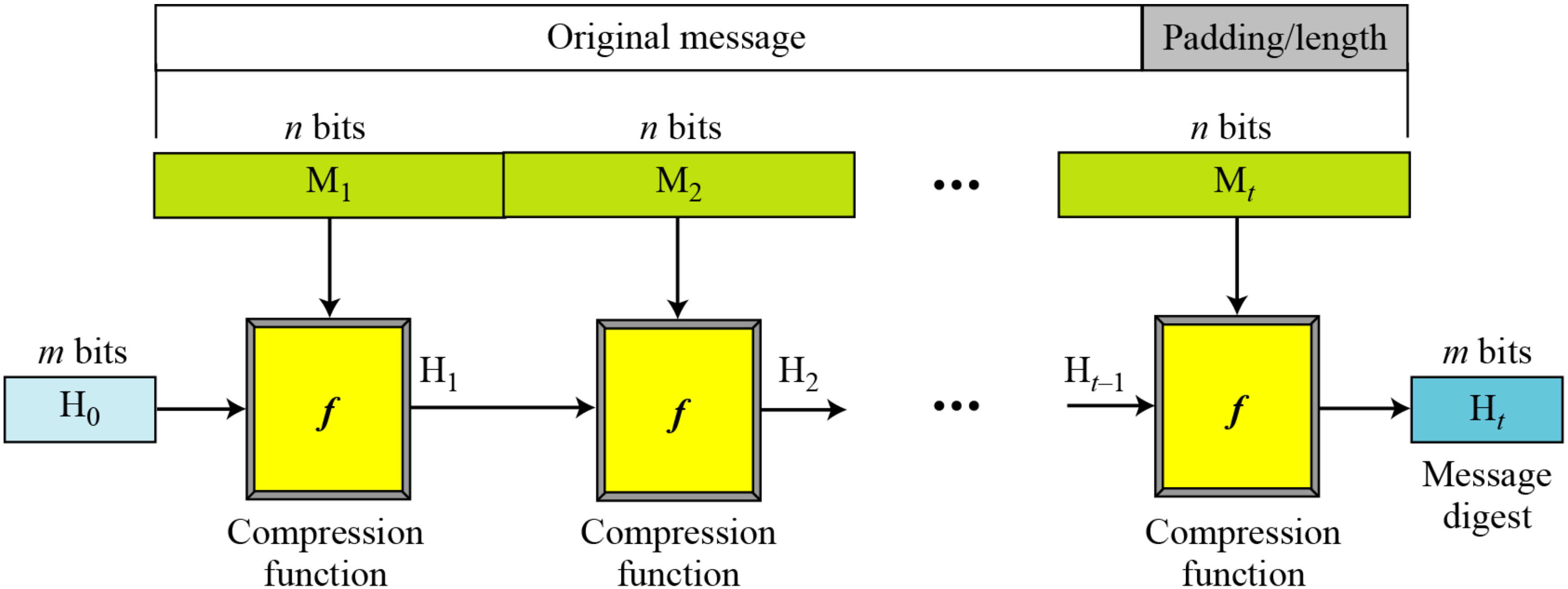
- MD5: Message digest5 takes an input of arbitrary length and produces a 128bit output (16 bytes usually represented by 32 Hex characters)
- DOG = b0e603b215aa2da0e6c605301d79efe4
- CAT = c01ae1a5f122f25ce5675f86028b536a
- dog = 06d80eb0c50b49a509b49f2424e8c805
- canis canem=02a35181a31072d1ac0436572ca77abe

Hashing

- canis canem=02a35181a31072d1ac0436572ca77abe

- canis caneM= 0c89f84deee731b74615d536dd7dabd4
- A minor change in the plaintext results in a drastic change to the message digest.
- Online tools such as: http://www.md5.cz/
- Offline such as md5sum , command line utility for windows or *nix
- Note: the DATA section of a file is hashed, NOT the system metadata.

Merkle-Damgard Scheme



- Well-known method to build cryptographic has function
- A message of arbitrary length is broken into blocks
  - length depends on the compression function *f*
  - padding the size of the message into a multiple of the block size.
  - sequentially process blocks , taking as input the result of the hash so far and the current message block, with the final fixed length output

Two Group of Compression Functions

- The compression function is made from scratch
  - Message Digest
- A symmetric-key block cipher serves as a compression function
  - Whirlpool

Hash Functions Family

- **MD (Message Digest)**
  - Designed by Ron Rivest
  - Family: MD2, MD4, MD5
- **SHA (Secure Hash Algorithm)**
  - Designed by NIST
  - Family: SHA-0, SHA-1, and SHA-2
    - SHA-2: SHA-224, SHA-256, SHA-384, SHA-512
    - SHA-3: New standard in competition
- **RIPEMD (Race Integrity Primitive Evaluation Message Digest)**
  - Developed by Katholieke University Leuven Team
  - Family : RIPEMD-128, RIPEMD-160, RIPEMD-256, RIPEMD-320,

MD5, SHA-1, and RIPEMD-160

	MD5	SHA-1	RIPEMD-160
Digest length	128 bits	160 bits	160 bits
Basic unit of processing	512 bits	512 bits	512 bits
Number of steps	64 (4 rounds of 16)	80 (4 rounds of 20)	160 (5 paired rounds of 16)
Maximum message size	∞	2^{64} - 1 bits	2^{64} - 1 bits
Primitive logical functions	4	4	5
Additive constants used	64	4	9
Endianness	Little-endian	Big-endian	Little-endian

MD2, MD4 and MD5

- Family of one-way hash functions by Ronald Rivest
  - All produces 128 bits hash value

- **MD2: 1989**
  - Optimized for 8 bit computer
  - Collision found in 1995
- **MD4: 1990**
  - Full round collision attack found in 1995
- **MD5: 1992**
  - Specified as Internet standard in RFC 1321
  - since 1997 it was theoretically not so hard to create a collision
  - Practical Collision MD5 has been broken since 2004
  - CA attack published in 2007

Hashing

- MD5 is produced using 5 steps
- **Step 1. Append Padding Bits**

Pad the message so it becomes 448 mod 512 bits long. The pad consists of 1+0000000...000

- **Step 2. Append Length**

The length is appended as (length of message) mod 64

- **Step 3. Initialize MD Buffer**

Initialises to

var *int* a0 := 0x67452301 //A

var *int* b0 := 0xefcdab89 //B

var *int* c0 := 0x98badcfe //C

var *int* d0 := 0x10325476 //D

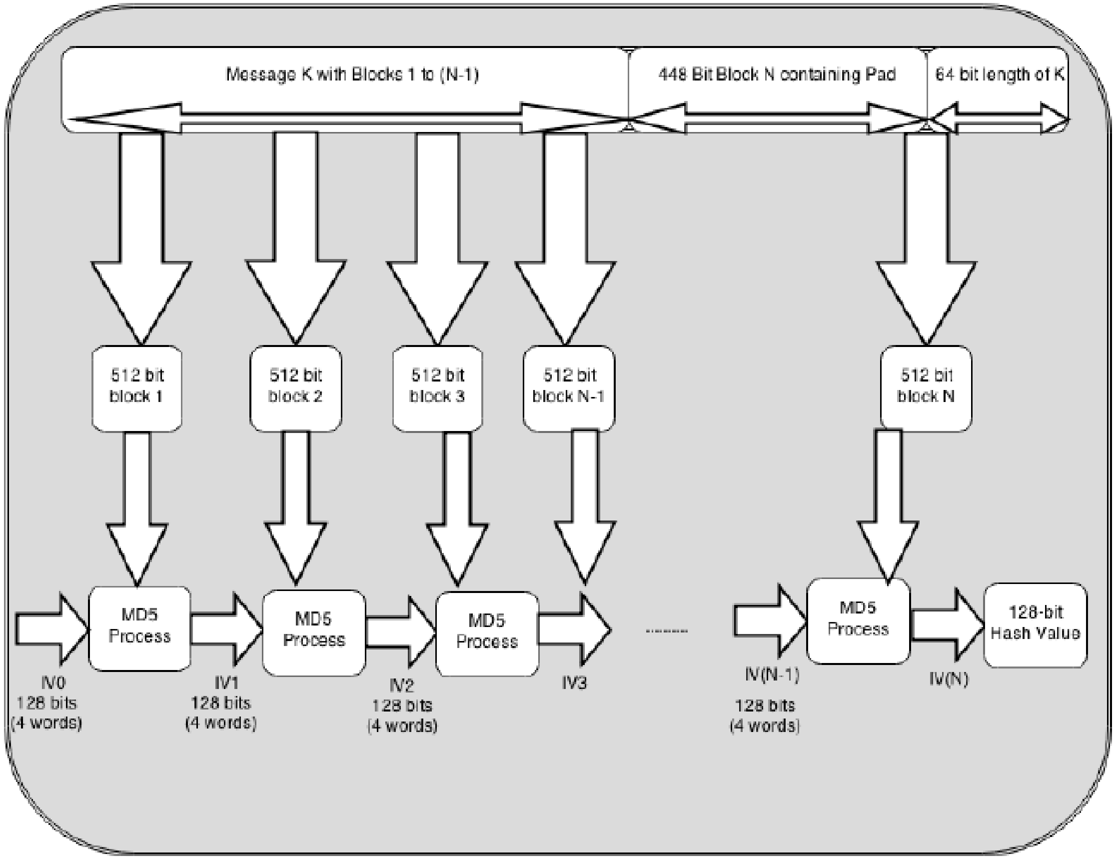
- **Step 4. Process Message in 16-Word Blocks**

Each word is 32 bits giving each message block 512 bits long.

- **Step 5. Output**

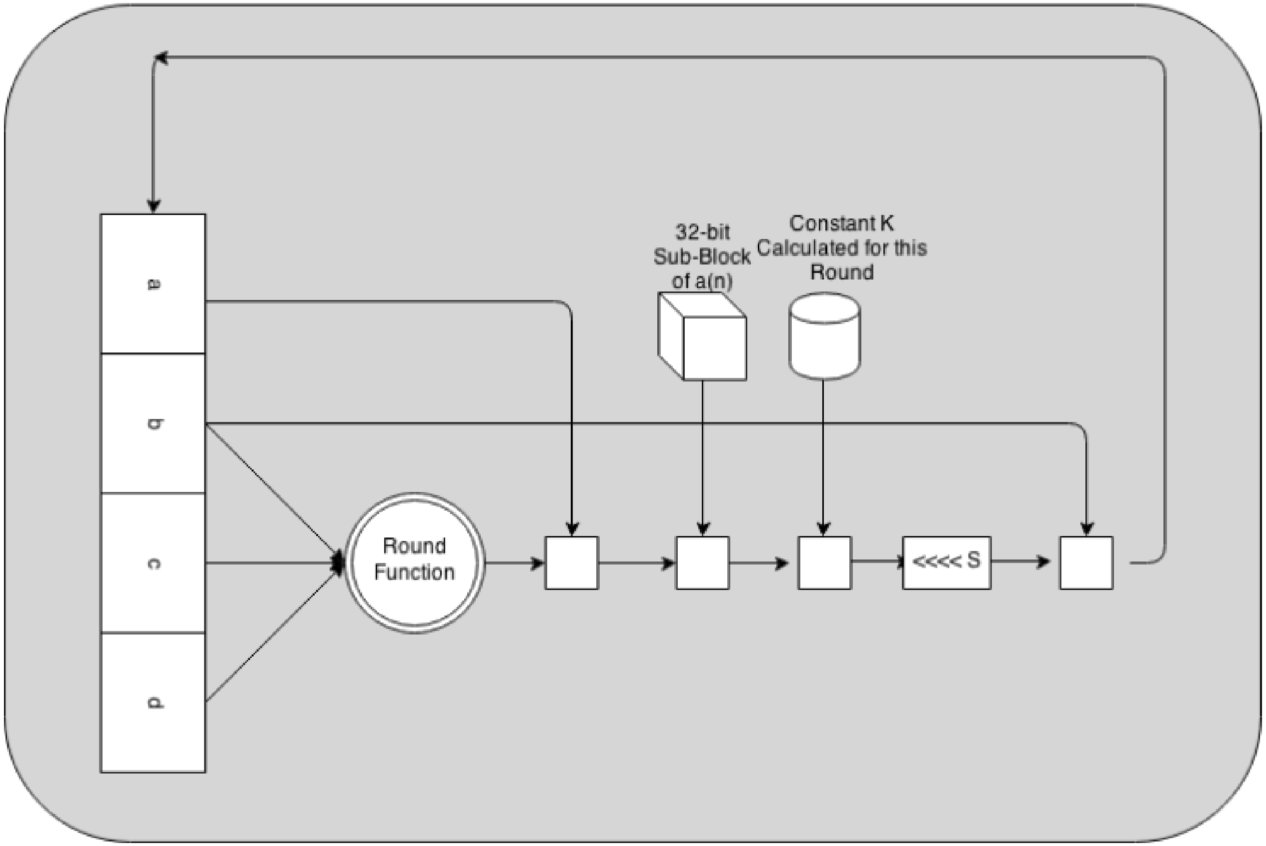
Outputs a single 128bit value that becomes the MD buffer for the next round

Hashing



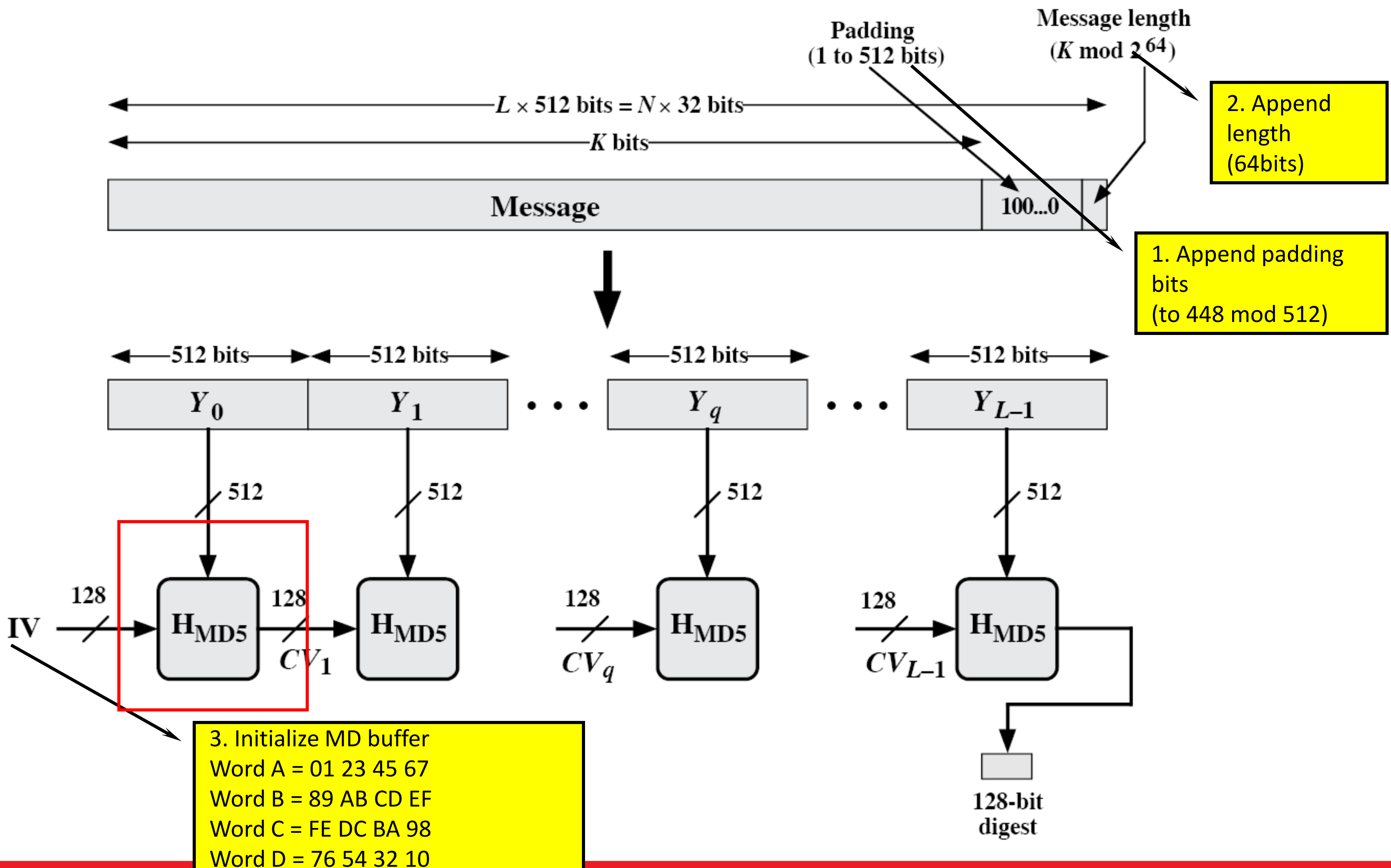
Message K with Blocks 1 to (N-1)
448 Bit Block N containing Pad
64 bit length of K
512 bit block 1
512 bit block 2
512 bit block 3
512 bit block N-1
512 bit block N
MD5 Process
MD5 Process
MD5 Process
MD5 Process
128-bit Hash Value
IV0 128 bits (4 words)
IV1 128 bits (4 words)
IV2 128 bits (4 words)
IV3
IV(N-1) 128 bits (4 words)
IV(N)

Hashing



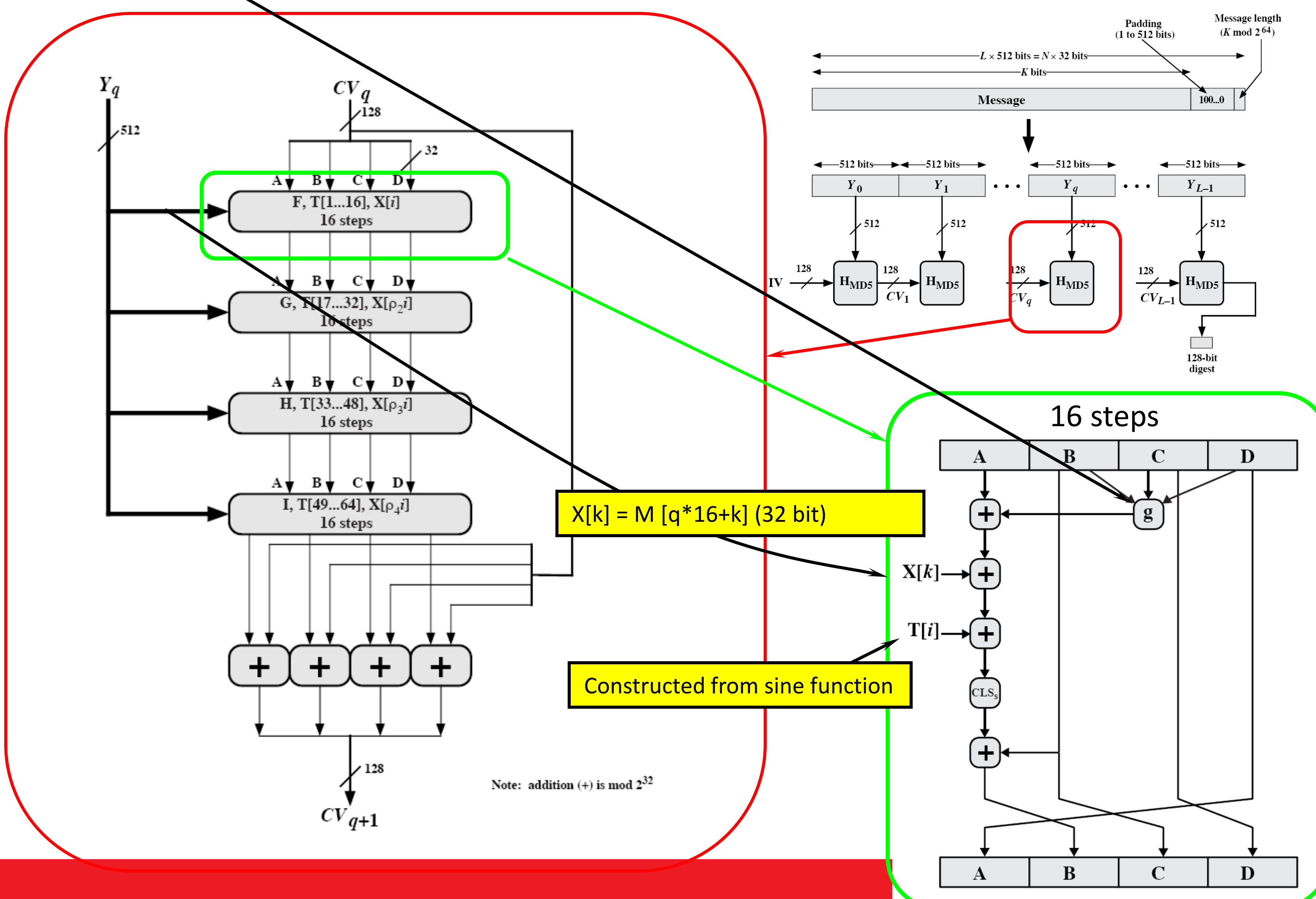
32-bit
Sub-Block
of a(n)
Constant K
Calculated for this
Round
a
b
c
d
Round
Function
<<<< S

MD5 Overview



b	c	d	F	G	H	I
0	0	0	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	0	1	1
1	0	1	0	1	0	1
1	1	0	1	1	0	0
1	1	1	1	1	1	0

Hash Algorithm Design – MD5



The ith 32-bit word in matrix T, constructed from the sine function

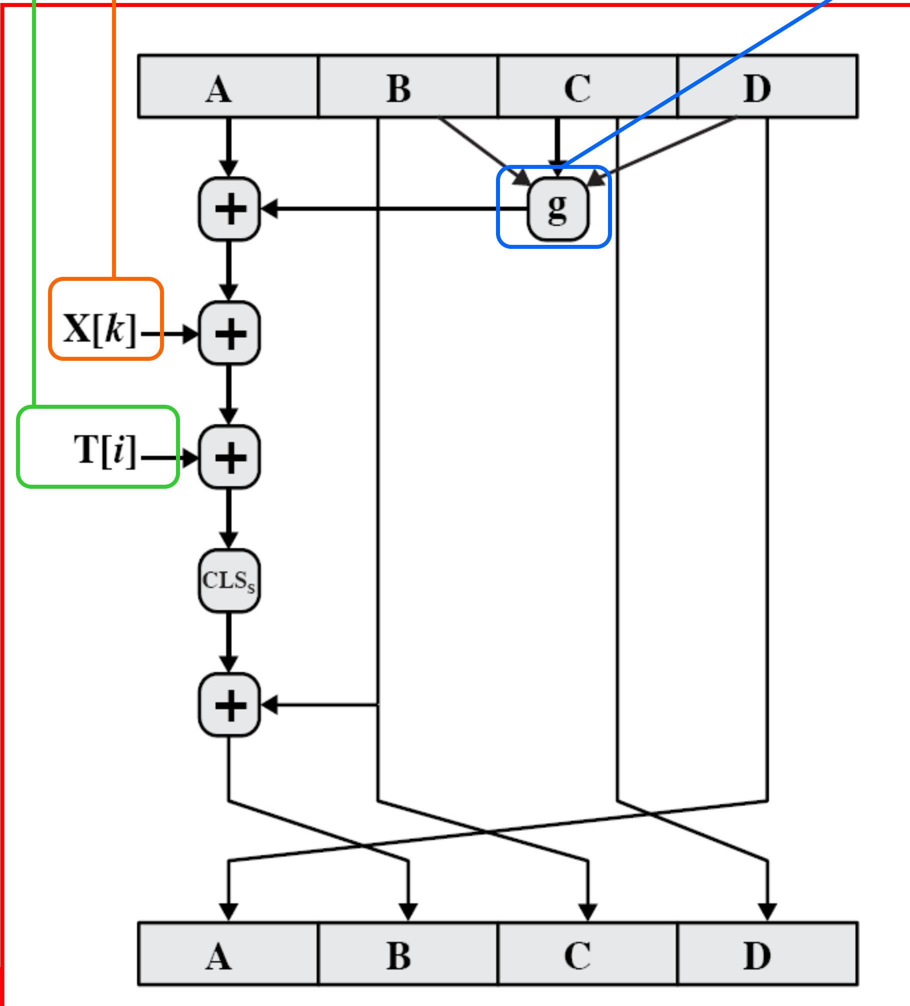
M [q*16+k] = the kth 32-bit word from the qth 512-bit block of the msg

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$

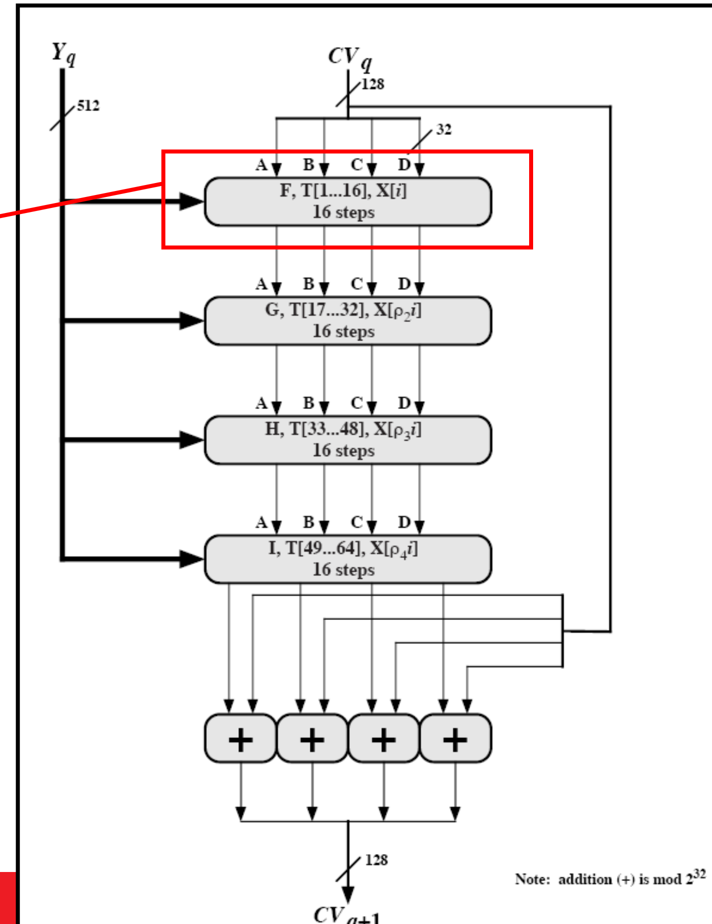
$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$

$$H(X,Y,Z) = X \oplus Y \oplus Z$$

$$I(X,Y,Z) = Y \oplus (X \vee \neg Z)$$



Single step



- Advantages: fast to compute

Still accepted as legal evidence in court.

Disadvantages:

Broken. Collisions can be generated.

Modern computing power enables brute force matching (theoretically)

Input vector 1:

d1	31	dd	02	c5	e6	ee	c4	69	3d	9a	06	98	af	f9	5c
2f	ca	b5	87	12	46	7e	ab	40	04	58	3e	b8	fb	7f	89
55	ad	34	06	09	f4	b3	02	83	e4	88	83	25	71	41	5a
08	51	25	e8	f7	cd	c9	9f	d9	1d	bd	f2	80	37	3c	5b
d8	82	3e	31	56	34	8f	5b	ae	6d	ac	d4	36	c9	19	c6
dd	53	e2	b4	87	da	03	fd	02	39	63	06	d2	48	cd	a0
e9	9f	33	42	0f	57	7e	e8	ce	54	b6	70	80	a8	0d	1e
c6	98	21	bc	b6	a8	83	93	96	f9	65	2b	6f	f7	2a	70

Input vector 2:

d1	31	dd	02	c5	e6	ee	c4	69	3d	9a	06	98	af	f9	5c
2f	ca	b5	07	12	46	7e	ab	40	04	58	3e	b8	fb	7f	89
55	ad	34	06	09	f4	b3	02	83	e4	88	83	25	f1	41	5a
08	51	25	e8	f7	cd	c9	9f	d9	1d	bd	72	80	37	3c	5b
d8	82	3e	31	56	34	8f	5b	ae	6d	ac	d4	36	c9	19	c6
dd	53	e2	34	87	da	03	fd	02	39	63	06	d2	48	cd	a0
e9	9f	33	42	0f	57	7e	e8	ce	54	b6	70	80	28	0d	1e
c6	98	21	bc	b6	a8	83	93	96	f9	65	ab	6f	f7	2a	70

Identical MD5 value, verified with WinHex: 79054025255fb1a26e4bc422aef54eb4

Secure Hash Algorithm

- SHA originally designed by NIST & NSA in 1993
- was revised in 1995 as SHA-1
- US standard for use with DSA signature scheme
  - standard is FIPS 180-1 1995, also Internet RFC3174
- based on design of MD4 with key differences
- produces 160-bit hash values
- recent 2005 results on security of SHA-1 have raised concerns on its use in future applications

Revised SHA

- NIST issued revision FIPS 180-2 in 2002
- adds 3 additional versions of SHA
  - SHA-256, SHA-384, SHA-512
- designed for compatibility with increased security provided by the AES cipher
- structure & detail is similar to SHA-1
- hence analysis should be similar
- but security levels are rather higher

GRIFFITH COLLEGE

SHA Versions

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Digest size	160	224	256	384	512
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	512	1024	1024
Word size	32	32	32	64	64
# of steps	80	64	64	80	80

- Sha-1 is currently the least secure cryptographic hash function supported by NIST

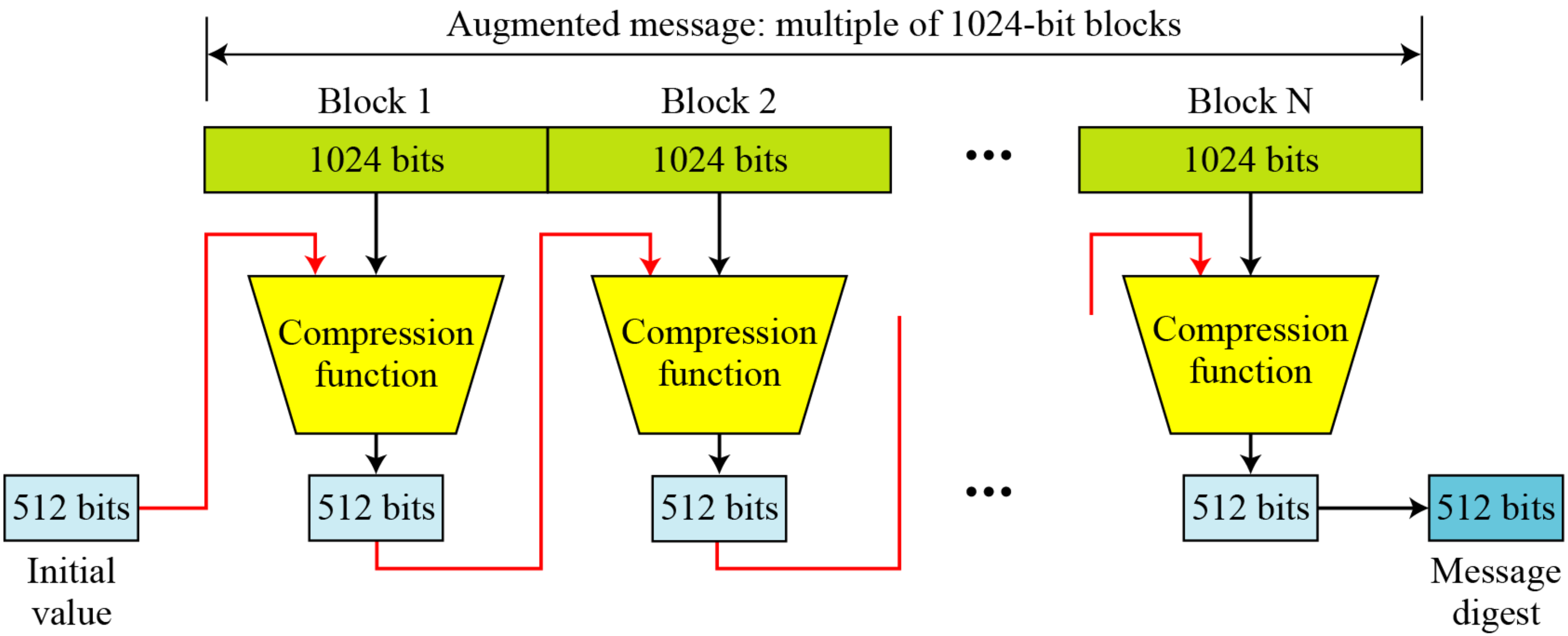
Algorithm and variant		Output size (bits)	Internal state size (bits)	Block size (bits)	Max message size (bits)	Rounds	Operations	Security (bits)	Example Performance (MiB/s)[45]
MD5 (as reference)		128	128 (4×32)	512	$2^{64} - 1$	64	add mod 2^{32}, and, or, xor, rot	<64 (collisions found)	335
SHA-0		160	160 (5×32)	512	$2^{64} - 1$	80	add mod 2^{32}, and, or, xor, rot	<80 (collisions found)	-
SHA-1		160	160 (5×32)	512	$2^{64} - 1$	80	add mod 2^{32}, and, or, xor, rot	<80 (theoretical attack[46] in 2^{51})	192
SHA-2	*SHA-224* *SHA-256*	224 256	256 (8×32)	512	$2^{64} - 1$	64	add mod 2^{32}, and, or, xor, shr, rot	112 128	139
	SHA-384 *SHA-512* *SHA-512/224* *SHA-512/256*	384 512 224 256	512 (8×64)	1024	$2^{128} - 1$	80	add mod 2^{64}, and, or, xor, shr, rot	192 256 112 128	154
SHA-3	*SHA3-224* *SHA3-256* *SHA3-384* *SHA3-512* *SHAKE128* *SHAKE256*	224 256 384 512 *d* (arbitrary) *d* (arbitrary)	1600 (5×5×64)	1152 1088 832 576 1344 1088	∞	24	and, xor, not, rot	112 128 192 256 min(*d*/2, 128) min(*d*/2, 256)	

Sample Processing

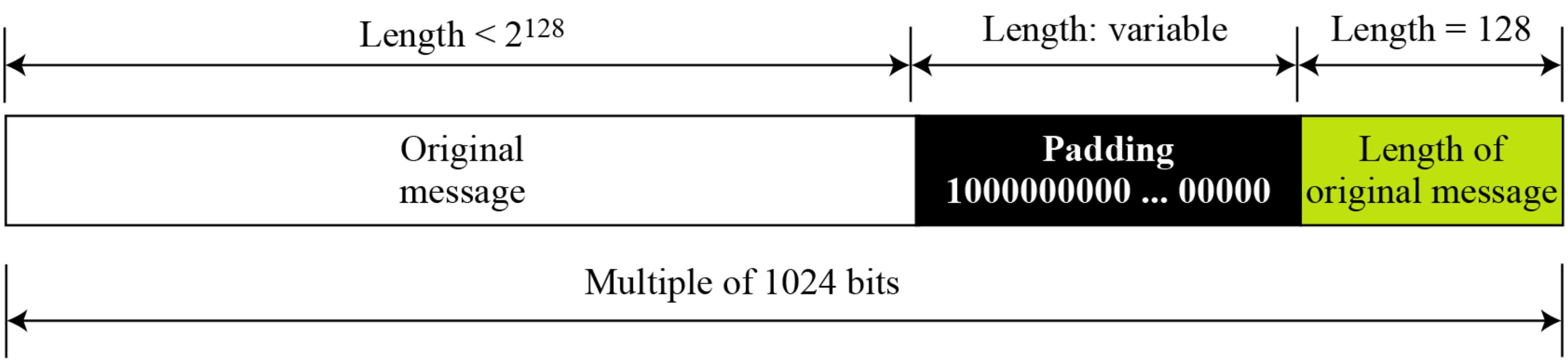
Type	bits	data processed
md5	128	469.7MB/s
sha1	160	339.4MB/s
sha512	512	177.7MB/s

- Mac Intel 2.66 Ghz core i7
- 1024 bytes block of data

SHA-512 Overview



Padding and length field in SHA-512



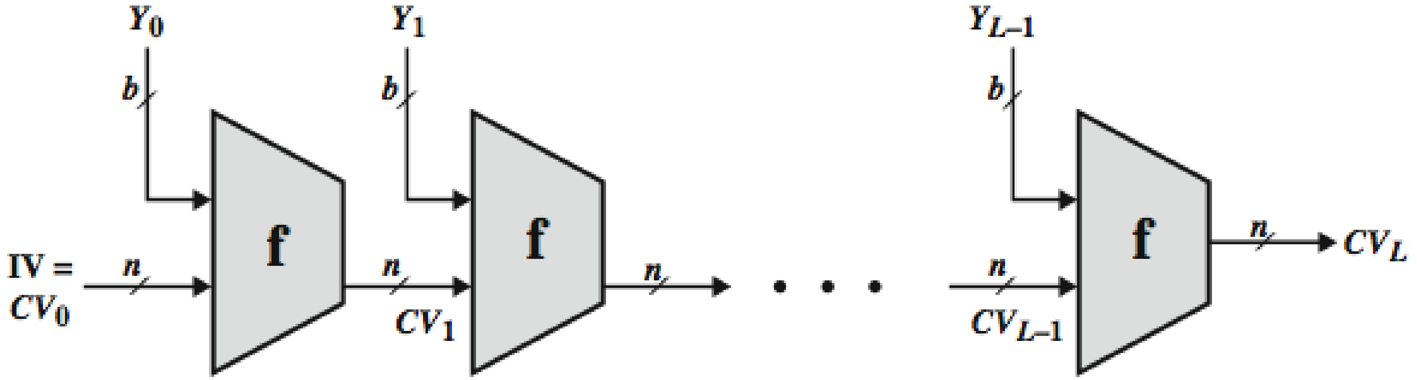
- **What is the number of padding bits if the length of the original message is 2590 bits?**
- We can calculate the number of padding bits as follows:

$$|P| = (-2590 - 128) \bmod 1024 = -2718 \bmod 1024 = 354$$

- The padding consists of one 1 followed by 353 0's.

Hash Function Cryptanalysis

- cryptanalytic attacks exploit some property of alg so faster than exhaustive search
- hash functions use iterative structure
  - process message in blocks (incl length)
- attacks focus on collisions in function f



Attacks on Hash Functions

- have brute-force attacks and cryptanalysis
- a preimage or second preimage attack
  - find y s.t. $H(y)$ equals a given hash value
- collision resistance
  - find two messages x & y with same hash so $H(x) = H(y)$

Birthday Attack

- How many people do you need so that the probability of having two of them share the same birthday is > 50% ?
- N distinct values, k randomly chosen ones
  - P(N,i) = prob(i randomly selected values from 1..N have at least one match)
  - P(N,2) = 1/N
  - P(N,i+1) = P(N,i)+(1-P(N,i))(i/N)
- For P(N,k)>0.5, need k ≈ $N^{1/2}$
- ➢ For m bits hash code, hence value $2^{m/2}$ determines strength of hash code against brute-force attacks
  - 128-bits inadequate, 160-bits suspect

Definition

Birthday attacks are a class of brute-force techniques that target the cryptographic hash functions. The goal is to take a cryptographic hash function and find two different inputs that produce the same output.

The Birthday Problem

What is the probability that at least two of k randomly selected people have the same birthday? (Same month and day, but not necessarily the same year.)

The Birthday Paradox

How large must *k* be so that the probability is greater than 50 percent?

The answer is 23

It is a paradox in the sense that a mathematical truth contradicts common intuition.

Birthday paradox

- **What's the chances that two people in a class of 43 have the same birthday?**
- Approximate solution:

$$p = 1 - e^{\frac{-k^2}{2N}} = 1 - e^{\frac{-43^2}{2*365}} \approx 0.92$$

Where *k* = 43 people, and *N* = 365 choices

Birthday Calendar Wall

- Equivalence to our hashing space

Jan	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Feb	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28			
Mar	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Apr	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
May	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
Jun	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
Jul	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Aug	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Sep	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
Oct	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Nov	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
Dec	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Calculating the Probability-1

- Assumptions
  - Nobody was born on February 29
  - People's birthdays are equally distributed over the other 365 days of the year

GRIFFITH COLLEGE

Calculating the Probability-2

- In a room of *k* people

  q: the prob. all people have different birthdays

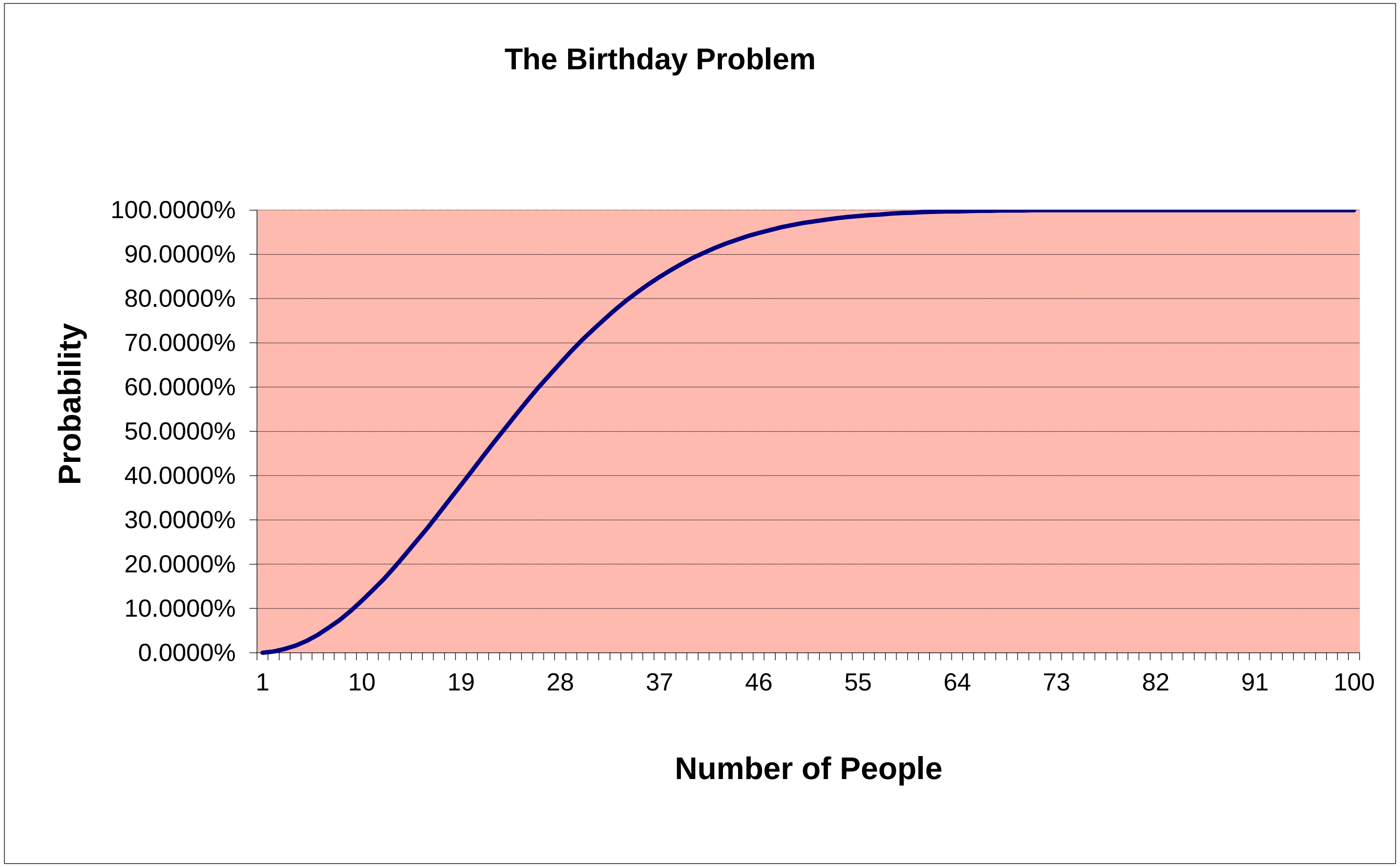
$$q = \frac{365}{365} \cdot \frac{364}{365} \cdot \frac{363}{365} \cdot \frac{362}{365} \cdots \frac{365-(k-1)}{365}$$

$$q = \frac{365!/(365-k)!}{365^k}$$

p: the prob. at least two of them have the same birthdays

$$p = 1 - q = 0.5 \Rightarrow k = 23$$

- Shared Birthday Probabalities



Attack Prevention

The important property is the length in bits of the message digest produced by the hash function.

If the number of *m* bit hash , the cardinality *n* of the hash function is

$$n = 2^m$$

The 0.5 probability of collision for m bit hash, expected number of operation *k* before finding a collision is very close to

$$k \approx \sqrt{n} = 2^{m/2}$$

m should be large enough so that it's not feasible to compute hash values!!!

The need of new Hash standard

- MD5 and SHA-0 already broken
- SHA-1 not yet fully "**broken**"
  - but similar to broken MD5 & SHA-0
  - so considered insecure and be fade out
- SHA-2 (esp. SHA-512) seems secure
  - shares same structure and mathematical operations as predecessors so have concern
- NIST announced in 2007 a competition for the SHA-3 next gen hash function
  - goal to have in place by 2012

SHA-3 Requirements

- replace SHA-2 with SHA-3 in any use
  - so use same hash sizes
- preserve the nature of SHA-2
  - so must process small blocks (512 / 1024 bits)
- evaluation criteria
  - security close to theoretical max for hash sizes
  - cost in time & memory
  - characteristics: such as flexibility & simplicity

COLLISION IMPROVEMENTS

- Rogue CA construction (<2048 bits)
  - Cluster of 215 PlayStation3s
    - Performing like 8600 pc cores
  - Complexity 2^{50} using 30GB:
    - 1 day on cluster
  - Complexity $2^{48.2}$ using a few TBs:
    - 1 day on 20 PS3s and 1 pc
    - 1 day on 8 NVIDIA GeForce GTX280s
    - 1 day on Amazon EC2 at the cost of $2,000

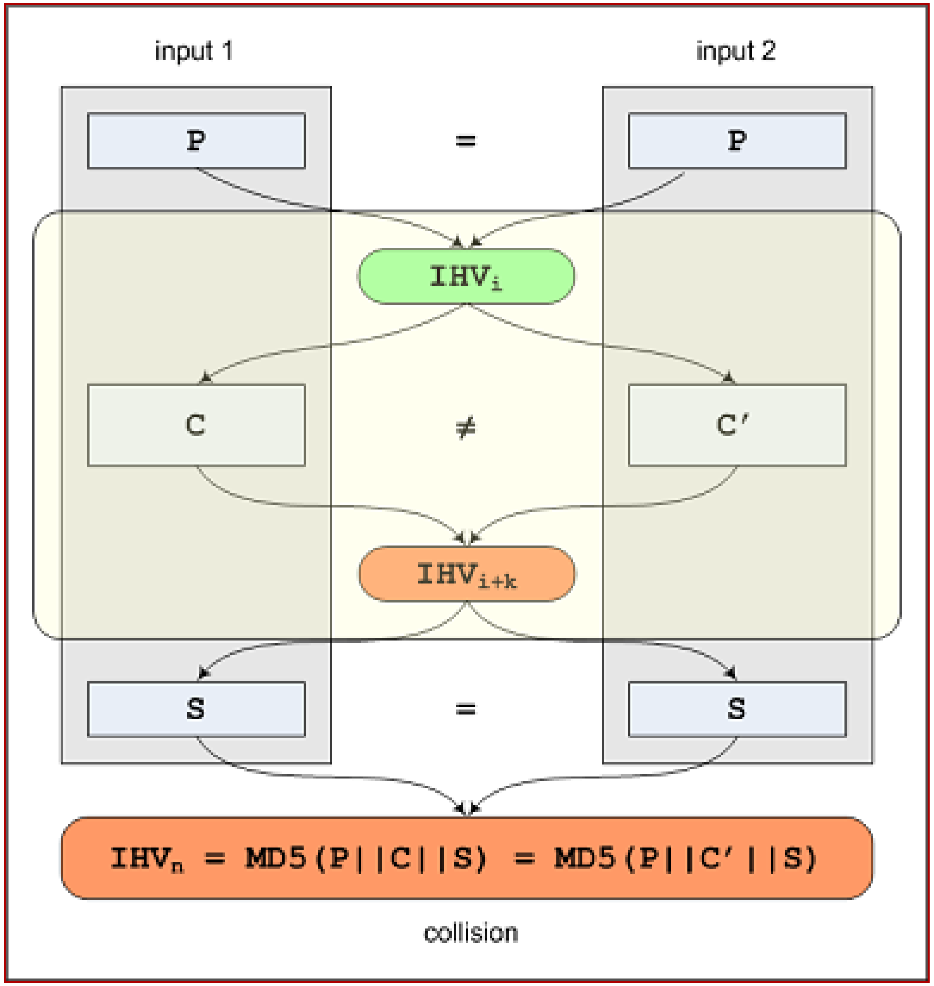
- Normal CPC
  - Complexity approx. 2^{39} (<1 day on quadcore pc)

MD5 Breakers

- Xiaoyun Wang (China)
  - collisions for MD5 in 2004
  - in a few hours on a big computer

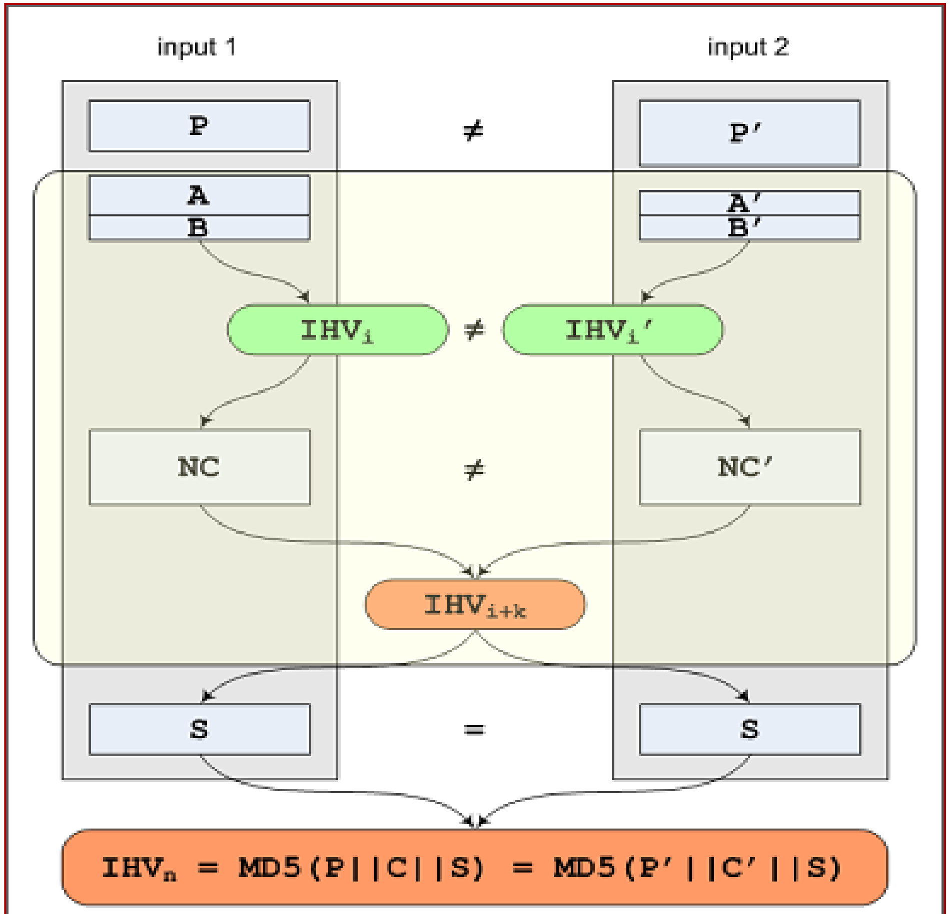
- Marc Stevens (Amsterdam)
  - MSc thesis 2007, TU/e
  - improved method, fully automated
  - collisions can now be found in about 1 second on a standard laptop

Wang's Collisions : Identical Prefix



- identical prefix P
- different collision blocks C, C'
- identical suffix S

Steven's Collisions : Chosen Prefix



chosen-prefix collision

- Different prefixes P, P'
- different collision blocks NC, NC'
- identical suffix S

SHA-0 Attack

- 1998
  - Possible collisions attack with 2^{61} operations
- 2004
  - Full collisions found with 2^{51} operations
  - 80,000 CPU hours with Itanium2
- 2004
  - Collisions with 2^{40} operations for SHA-0, MD5 and other
- 2005
  - Collisions with 2^{39} operations

GRIFFITH COLLEGE

SHA-1 Attack

- 2005
  - Collisons found in 2^{80} operatons of reduced version of SHA-1--53 out of 80 rounds
- 2006
  - SHA-1-64 with 2^{35} operations
- 2010
  - SHA-1-73 with 2^{35} operations
  - Project HashClash : claim fully near collision attack with estimated complexity of $2^{57.5}$

Progress of Collision Attacks

Attack complexities for MD5, SHA-1 and SHA-

jaar	MD5 identical prefix	MD5 chosen prefix	SHA-1 identical prefix	SHA-1 chosen prefix	SHA-2(256) identical prefix	SHA-2(256) chosen prefix
– 2003	64	64	80	80	128	128
2004	40		69			
2005	37		63			
2006	32	49		80 - ε		
2007	25	42	61			
2008	21					
2009	16 New !	39	52	New ?		

MD5 vectors

-
```
d1 31 dd 02 c5 e6 ee c4 69 3d 9a 06 98 af f9 5c
2f ca b5 87 12 46 7e ab 40 04 58 3e b8 fb 7f 89
55 ad 34 06 09 f4 b3 02 83 e4 88 83 25 71 41 5a
08 51 25 e8 f7 cd c9 9f d9 1d bd f2 80 37 3c 5b
d8 82 3e 31 56 34 8f 5b ae 6d ac d4 36 c9 19 c6
dd 53 e2 b4 87 da 03 fd 02 39 63 06 d2 48 cd a0
e9 9f 33 42 0f 57 7e e8 ce 54 b6 70 80 a8 0d 1e
c6 98 21 bc b6 a8 83 93 96 f9 65 2b 6f f7 2a 70
```
-
```
d1 31 dd 02 c5 e6 ee c4 69 3d 9a 06 98 af f9 5c
2f ca b5 07 12 46 7e ab 40 04 58 3e b8 fb 7f 89
55 ad 34 06 09 f4 b3 02 83 e4 88 83 25 f1 41 5a
08 51 25 e8 f7 cd c9 9f d9 1d bd 72 80 37 3c 5b
d8 82 3e 31 56 34 8f 5b ae 6d ac d4 36 c9 19 c6
dd 53 e2 34 87 da 03 fd 02 39 63 06 d2 48 cd a0
e9 9f 33 42 0f 57 7e e8 ce 54 b6 70 80 28 0d 1e
c6 98 21 bc b6 a8 83 93 96 f9 65 ab 6f f7 2a 70
```

- Each of these blocks has MD5 hash
  79054025255fb1a26e4bc422aef54eb4

MD5 Collision demo

```
$ ls -al
total 16
drwxr-xr-x  4 admin  staff  136 Jul 16 17:05 .
drwxr-xr-x  9 admin  staff  306 Jul 16 16:40 ..
-rwxr--r--  1 admin  staff  128 Jul 14 11:34 v1
-rwxr--r--  1 admin  staff  128 Jul 14 11:35 v2

$ md5 v*; openssl dgst -sha1 v*
MD5 (v1) = 79054025255fb1a26e4bc422aef54eb4
MD5 (v2) = 79054025255fb1a26e4bc422aef54eb4
SHA1(v1)= a34473cf767c6108a5751a20971f1fdfba97690a
SHA1(v2)= 4283dd2d70af1ad3c2d5fdc917330bf502035658
```

Concat File Equivalence

```
$ ls >f1
$ cat v1 f1 >w1
$ cat v2 f1 >w2
$ ls -al
total 40
drwxr-xr-x  7 admin  staff  238 Jul 16 17:07 .
drwxr-xr-x  9 admin  staff  306 Jul 16 16:40 ..
-rw-r--r--  1 admin  staff    9 Jul 16 17:06 f1
-rwxr--r--  1 admin  staff  128 Jul 14 11:34 v1
-rwxr--r--  1 admin  staff  128 Jul 14 11:35 v2
-rw-r--r--  1 admin  staff  137 Jul 16 17:07 w1
-rw-r--r--  1 admin  staff  137 Jul 16 17:07 w2
$ md5 w*; openssl dgst -sha1 w*
MD5 (w1) = e9dc7f025001005370d9140168895489
MD5 (w2) = e9dc7f025001005370d9140168895489
SHA1(w1)= d867ab657437652d1cd9df9b4c89d9810f35fc24
SHA1(w2)= 2e05a71ff6c16f57d6ca935a47360de6aefcfad5
```

But how’s about

```
$ md5 -s  windows
MD5 ("windows") = 0f4137ed1502b5045d6083aa258b5c42
```

http://md5.rednoize.com/