Message Authentication Code
Hash Function and Message Digest

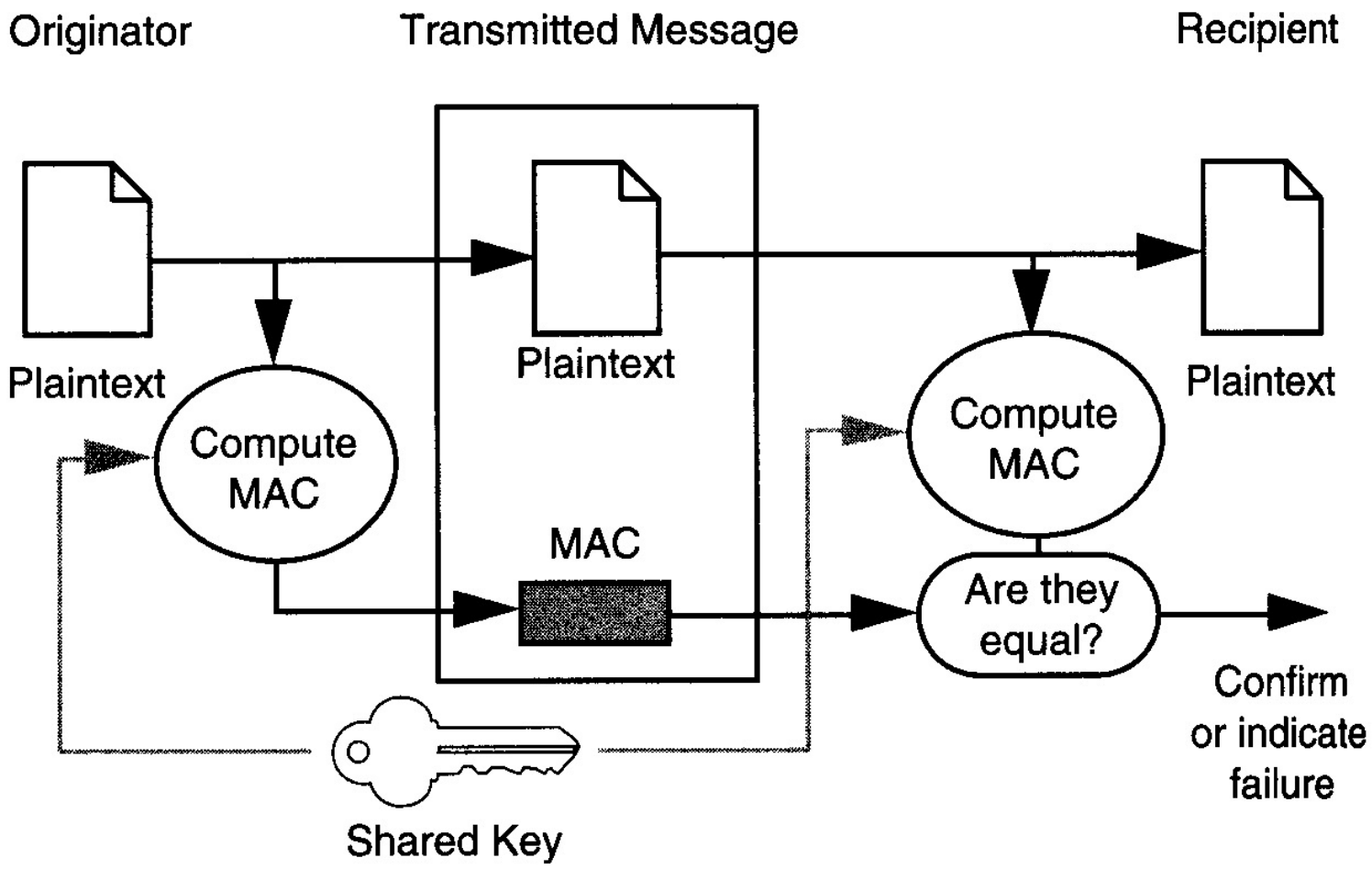
What is Message Authentication ?

- Procedure that allows communicating parties to verify that received messages are authentic, namely
  - source is authentic – not from masquerading
  - contents unaltered – message has not been modified
  - timely sequencing – the message is not a replay of a previously sent one

Ways to provide Message Authentication

- Message Authentication via Conventional Encryption
  - Only the sender and receiver should share a key ;
  - Include a time-stamp or “nonce” to prevent replay attack
  - Implicitly assume the receiver can recognize if the output from the decryption unit is garbage or not ;
    - easy if they know the message has some specific format, e.g. English
    - May be difficult if the original plaintext are random binary data =>need to impose some structure, e.g. Checksum
- Message Authentication without Message Encryption (thus no message confidentiality)
  - An authentication tag (aka Message Authentication Code or MAC) is generated and appended to each message where
    - the MAC is computed as a publicly known function F, of the message M and a shared secret key K:
      - MAC = F(K, M)
  - A *one-way* Hash function can be used as F

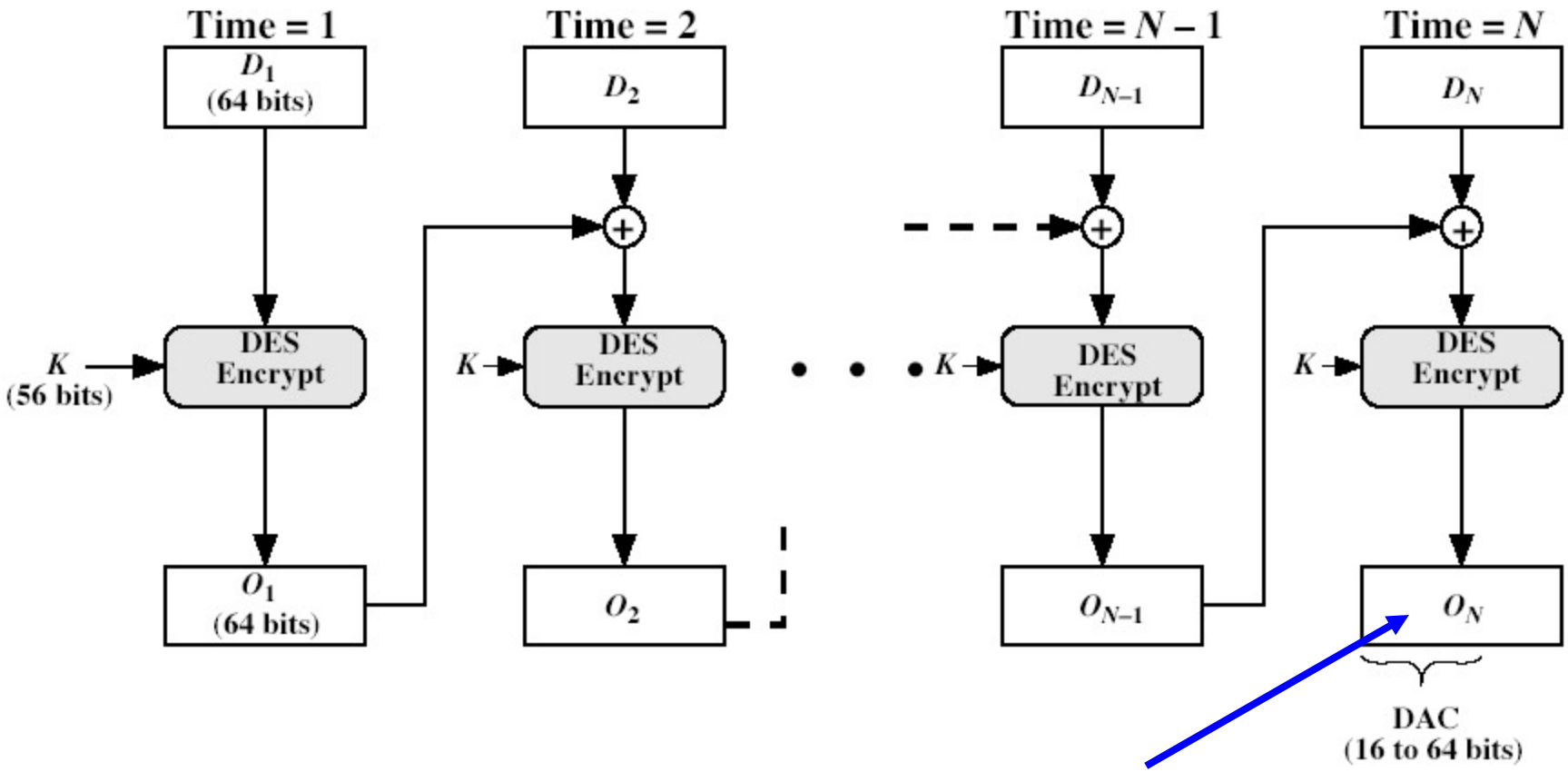
Ensuring Message Authenticity using a MAC



Message Authentication Code

- Receiver assured that message is not altered – no modification
- Receiver assured that the message is from the alleged sender – no masquerading
- Include a sequence number, assured proper sequence – no replay

CBC-residue as MAC



CBC-residue, aka DAC = Data Authentication Code

CBC-residue as MAC (cont'd)

- The last encrypted block, aka the CBC residue, can be used as a "Message Authentication Code" (MAC) for a message as follows:

1. The sender transmits the original message in plaintext together with the the CBC residue (but NOT the key, of course)
2. The receiver, who knows the key in advance, can then encrypt the plaintext upon its arrival using CBC mode. If the message has been tampered with during transmission, the CBC residue won't match !

- Notice in this case, CBC is used for MAC purpose and does NOT provide secrecy at all ;
- If both secrecy and message-authenticity (tamper-proof) is required, we need to do CBC twice in 2 passes with 2 different keys:
  - 1st pass for encryption,
  - 2nd pass to generate the CBC-residue for MAC.
- Why is it insufficent to just append the CBC residue of the 1st pass as the MAC ?

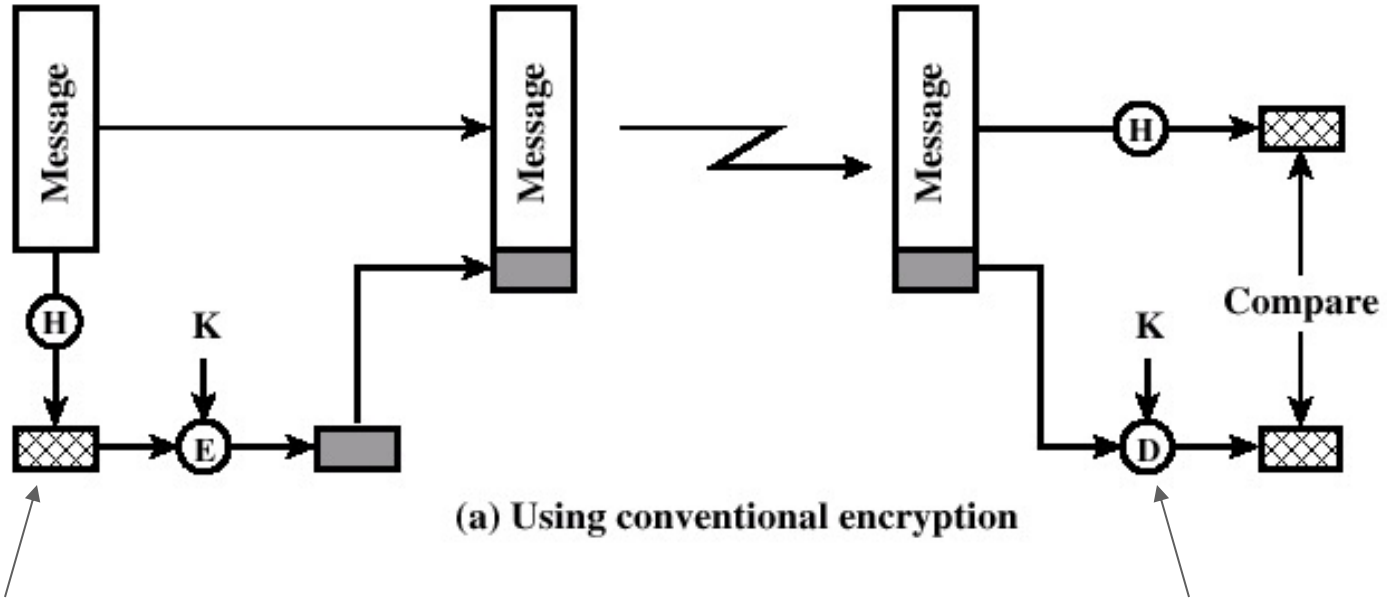
Drawbacks of using Encryption for MAC

- Encryption software is slow
- Encryption hardware costs aren't cheap
- Hardware optimized toward large data sizes
- Encryption Algorithms are usually covered by patents
- Algorithms subject to US export control

One-Way Hash Function

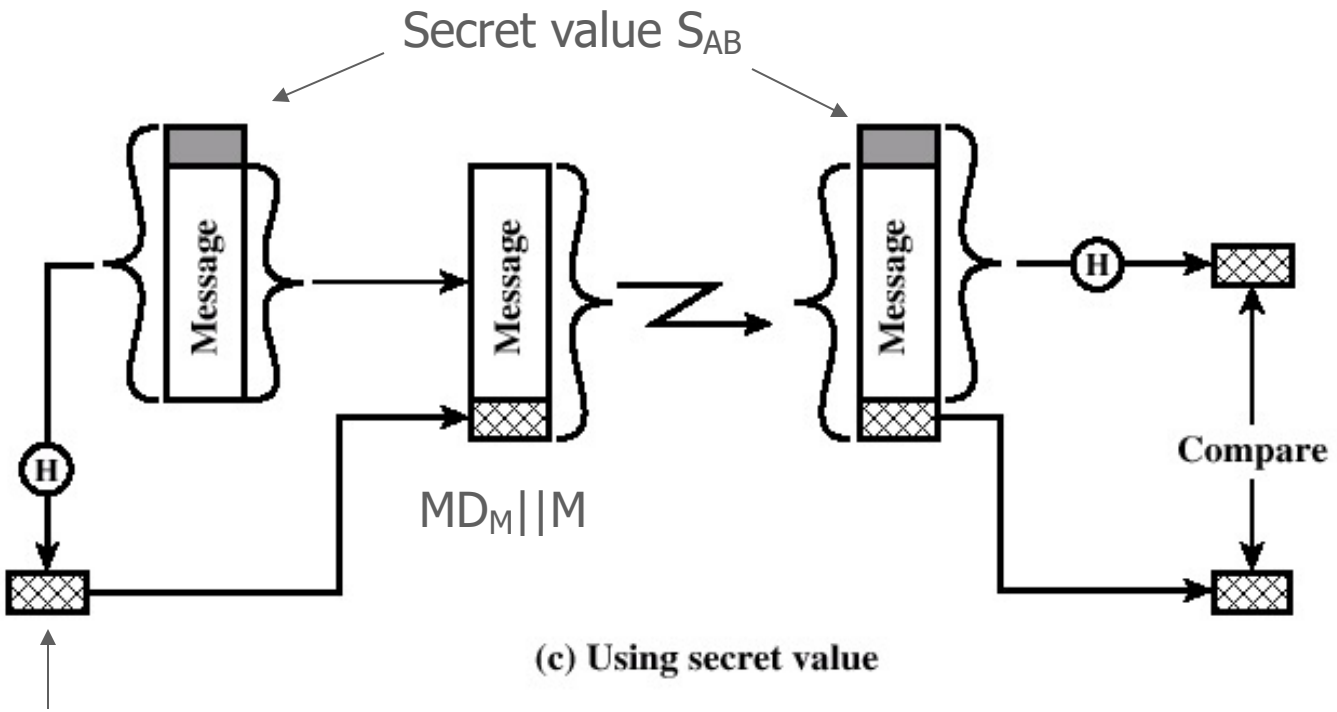
- Hash function accepts a *variable* size message M as input and produces a *fixed-size* message digest $H(M)$ as output
- Message digest is sent with the message for authentication
- Produces a fingerprint of the message
- No secret key is involved

MAC generation using One Way Hash + Conventional encryption



Authenticity is assured ; no confidentiality is provided;
Still need Encryption algorithm ; but faster because the hash function computation is quicker than encrypting the entire message ; now only need to encrypt
The much shorter message digest instead

Use only One Way Hash Function to compute MAC



$MD_M = H(M \,||\, S_{AB})$

Would $MD_M = H(S_{AB} \,||\, M)$ work as well ? The Answer is NO for some One-Way Hash Functions

No encryption for message authentication

Secret value never sent; can't modify the message

One-way Hash Function Requirements

weak

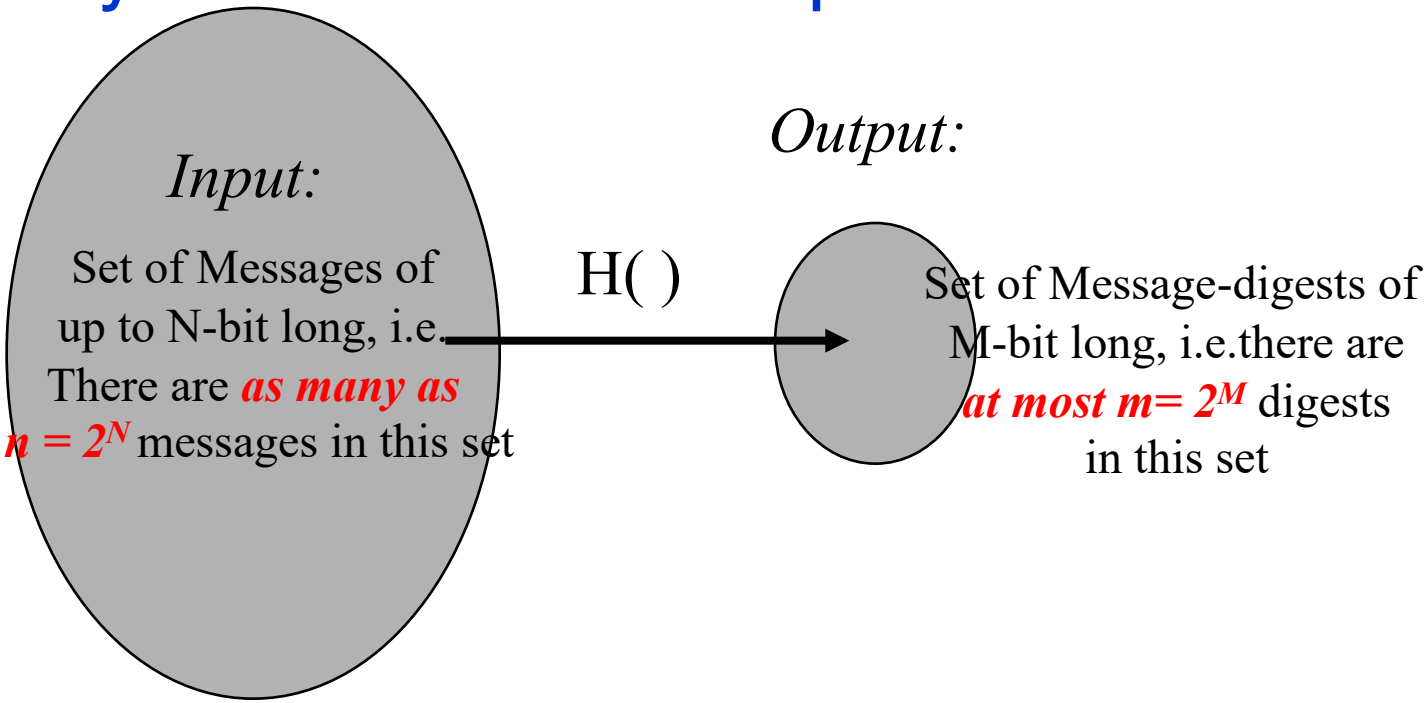
1. H can be applied to a block of data of any size
2. H produces a fixed length output
3. $H(x)$ is relatively easy to compute
4. For any given code h, it is computationally infeasible to find x such that $H(x) = h$ (i.e. safe against the so-called 1st preimage attack)
5. For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$ (i.e. safe against the so-called 2nd preimage attack)
6. It is computationally infeasible to find any pair (x,y) such that $H(x) = H(y)$

one way

strong collision resistance
birthday attack

weak collision resistance

How likely to have Hash output collisions ?



- Since $N >> M$, (and therefore) $n >> m$, collisions are inevitable no matter how secure the one-way function H( ) is.

The Birthday Paradox

- In a room with ***n*** people, what is the probability that we will find at least 2 people who have the same birthday (there are $m = 365$ possible choices of birthday)?

An *approximate* analysis:

- Assuming birthdays are uniformly distributed over the entire year. For any given pair of people, the possibly of them having the same birthday is $1/m = 1/365$;
- There are ${}_nC_2 = n(n-1)/2$ ways to select a pair out of n people
- Let $P_{collision}$ be the Probability of at least one collision,
  - $P_{collision}$ approx. $= n(n-1)/2 * 1/m = n(n-1)/2m$;
- $P_{collision} > ½$ when $n >= 20$
- In general, $P_{collision} > ½$ when n becomes $>= \sqrt{m}$
- The approximation is not good when n approaches m

- Where is the approximation ?

The Birthday Paradox (cont'd)

- In a room with ***n*** people, what is the probability that we will find at least 2 people who have the same birthday (there are m = 365 possible choices of birthday)?

An *exact* analysis:

- Assuming birthdays are uniformly distributed over the entire year. For any given pair of people, the possibly of them having the same birthday is $1/m = 1/365$;
- Probability of zero collision

= Probability that all of the n people have different birthdays

= $\boldsymbol{m * (m-1) * (m-2) *\ldots* (m-n+1) / m^n}$

$\boldsymbol{= 1 - n(n-1)/2m}$ ***approximately when m>>n***

- $P_{collision}$ = 1 – Probability of zero collision
  = $n(n-1)/2m$ approximately

How difficult to find a Hash collision ?

How secure is a one-way hash with 64-bit output, e.g. CBC-DES ?

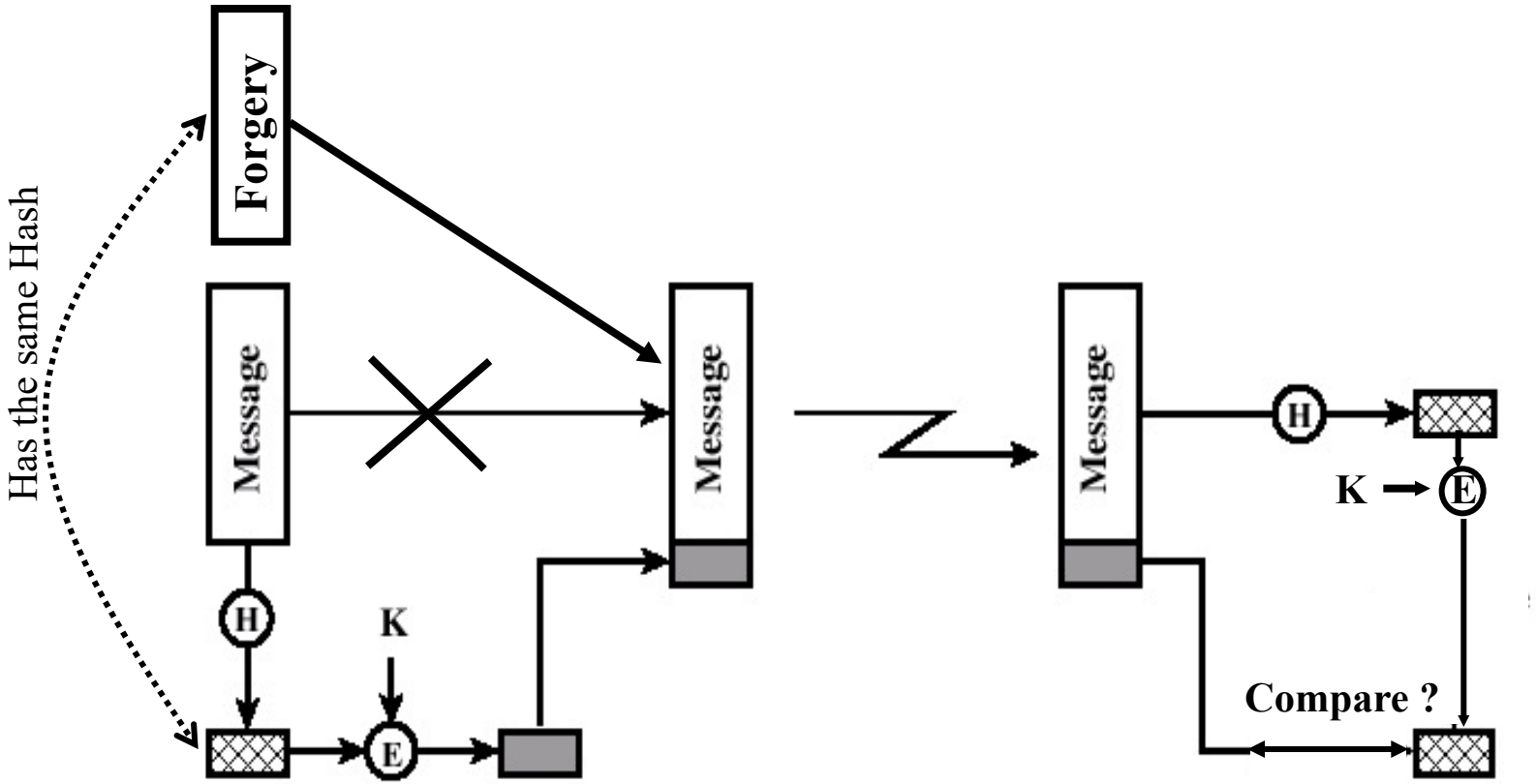
- Based on the property of a good hash function, the hash output of any input string should be uniformly distributed over the hash output space of size $m=2^{64}$
  - This is analogous to the fact that the birthday of any given person is uniformly distributed over any days within a year (i.e. output space of size m = 365)

- Thus, according to the Birthday Paradox, if no. of all possible outcomes = m, we only need to try about $n = \sqrt{m}$ inputs to the hash function to have a good chance to find a collision, e.g.

  For, a hash function with 64-bit output, $m=2^{64}$

=> it only takes about $\sqrt{m} = 2^{32}$ tries to find a pair of inputs which will produce the same hash output, i.e. a collision

Birthday Attack on Message Digest



Using CBC-residue as Message Authentication Code

Birthday Attacks

- **Birthday attack can proceed as follows:**
  - opponent generates 2^{32} variations of a valid message, all with essentially the same meaning ; this is “doable” given current technology.
  - opponent also generates 2^{32} variations of a desired fraudulent message
  - two sets of messages are compared to find a pair with same hash output (by argument similar to the Birthday paradox, this probability > 0.5)
  - have user (the victim) sign the valid message, but sent the forgery message which will have a valid message digest

- Conclusion is that we need to use longer MACs

- BTW, how can we generate 2^{32} variations of a letter carrying the same meaning ?

  Just 2 choices of wording at 32 different places.

How to generate large no. of messages of each type to get the necessary message digest collision to pull off a B-day attack ?

Type 1 message

I am writing {this memo | } to {demand | request | inform you} that {Fred | Mr. Fred Jones} {must | } be {fired | terminated} {at once | immediately}. As the {July 11 | 11 July} {memo | memorandum} {from | issued by} {personnel | human resources} states, to meet {our | the corporate} {quarterly | third quarter} budget {targets | goals}, {we must eliminate all discretionary spending | all discretionary spending must be eliminated}.

{Despite | Ignoring} that {memo | memorandum | order}, Fred {ordered | purchased} {PostIts | nonessential supplies} in a flagrant disregard for the company's {budgetary crisis | current financial difficulties}.

Type 2 message

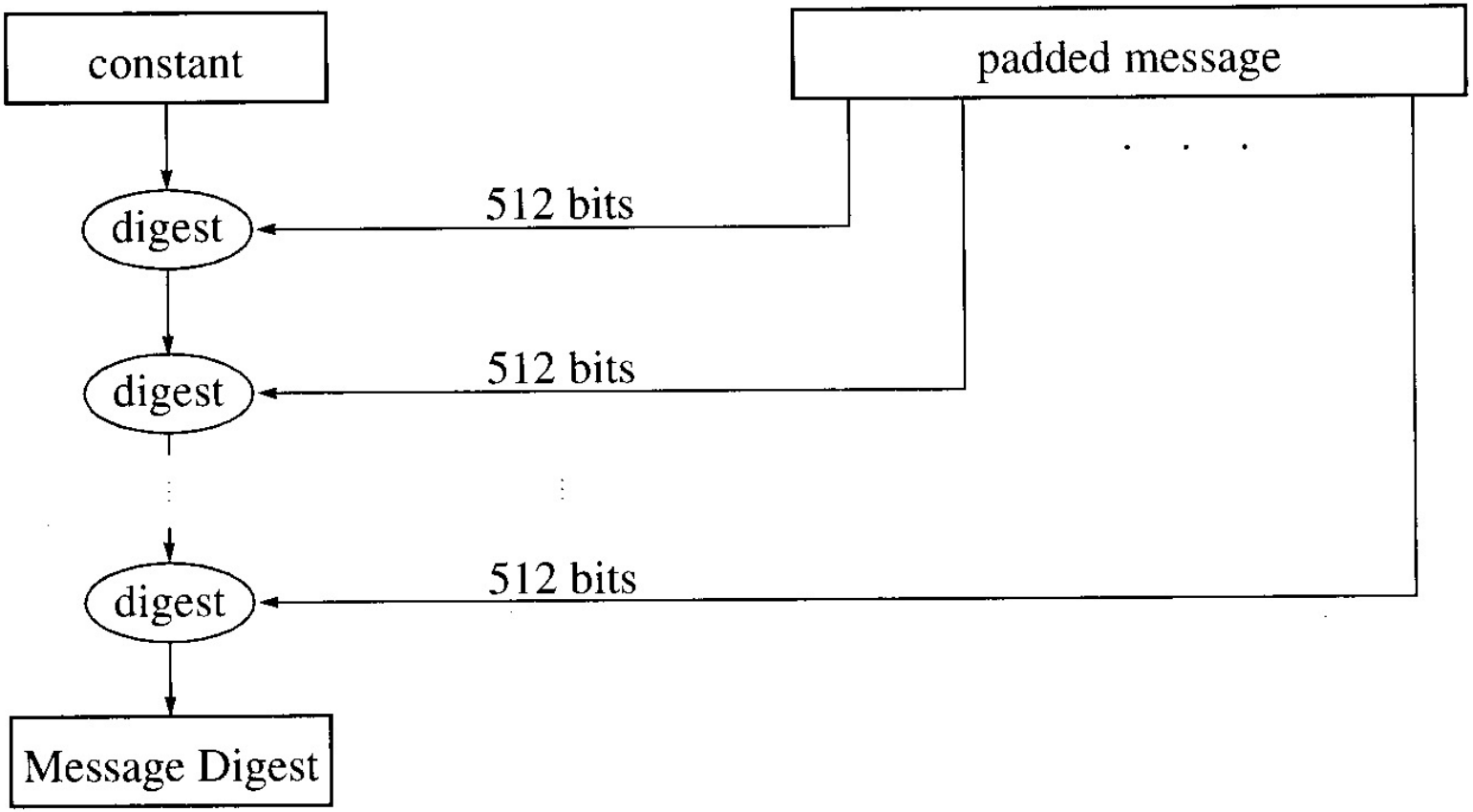
I am writing {this letter | this memo | this memorandum | } to {officially | } commend Fred {Jones | } for his {courage and independent thinking | independent thinking and courage}. {He | Fred} {clearly | } understands {the need | how} to get {the | his} job {done | accomplished} {at all costs | by whatever means necessary}, and {knows | can see} when to ignore bureaucratic {nonsense | impediments}. I {am hereby recommending | hereby recommend} {him | Fred} for {promotion | immediate advancement} and {further | } recommend a {hefty | large} {salary | compensation} increase.

MD5 Message Digest

- Ron Rivest - 1992
- RFC 1321
- Input: arbitrary Output: 128-bit digest
- Most widely used secure hash algorithm – until 2004
- MD5 shows significant crack in summer 2004 by a Chinese Team including: Wang Xiao Yun
  - they had successfully constructed a pair of input messages which can produce collision, i.e. the same MD5 hash output.
- After several years of further effort by many other researchers, MD5 was totally broken by Dec. 30 2008 (these are all b-day collision attacks, no successful preimage attacks so far) :
  - “MD5 considered harmful TODAY”, http://www.win.tue.nl/hashclash/rogue-ca/

The General Structure of MD5 and SHA-1

The so-called Merkle–Damgård construction



- Note the possibility of attacking by “appending” at the end of the original message if the shared secret is placed at the beginning of the input message ; what should we do ?

SHA-1 Secure Hash Function

- SHA was designed by NIST & NSA in 1993, revised 1995 as SHA-1 ; again, design criteria were not disclosed
- US standard for use with DSA signature scheme
  - standard is FIPS 180-1 1995, also Internet RFC3174
  - The algorithm is SHA, the standard was SHS
- Input is processed in 512-bit blocks ;
- Produce as output a 160-bit message digest
- But slower than MD5
- Was the generally preferred hash algorithm (than MD5)

Insecurity of SHA-1 (It's Dead !)

- Was considered to be Very Secure – Only until Feb 2005 ;
  - The same Chinese Team who broke MD5 in summer 2004 found a way to reduce the complexity of finding SHA-1 hash collisions from 2^{80} to 2^{68} => i.e. a speed up of 4096 times
- 1st Full collision for full SHA-1 discovered by Marc Stevens
  - https://marc-stevens.nl/research/
  - https://shattered.io
  - Won CRYPTO 2017 Best Paper Award and
  - Received Blackhat USA 2017 Pwnie Award for Best Crypto Attack
- Google announced the SHAttered attack in Feb 2017, which successfully constructed 2 different input messages to produce the same SHA1 hash !! (using ~ 110 GPU years), still 100K times faster than brute-force search for collisions
  - https://elie.net/static/files/how-we-created-the-first-sha1-collision-and-what-it-means-for-hash-security/how-we-created-the-first-sha1-collision-and-what-it-means-for-hash-security-slides.pdf
- But many legacy software, e.g. GiT will be stuck with SHA-1 for the foreseeable future
  - Mitigate risk by performing Counter-Cryptanalysis by scanning incoming files for patterns which facilitating collision-generating attacks.

Computational Cost Comparison

MD5

1 smartphone
30 sec

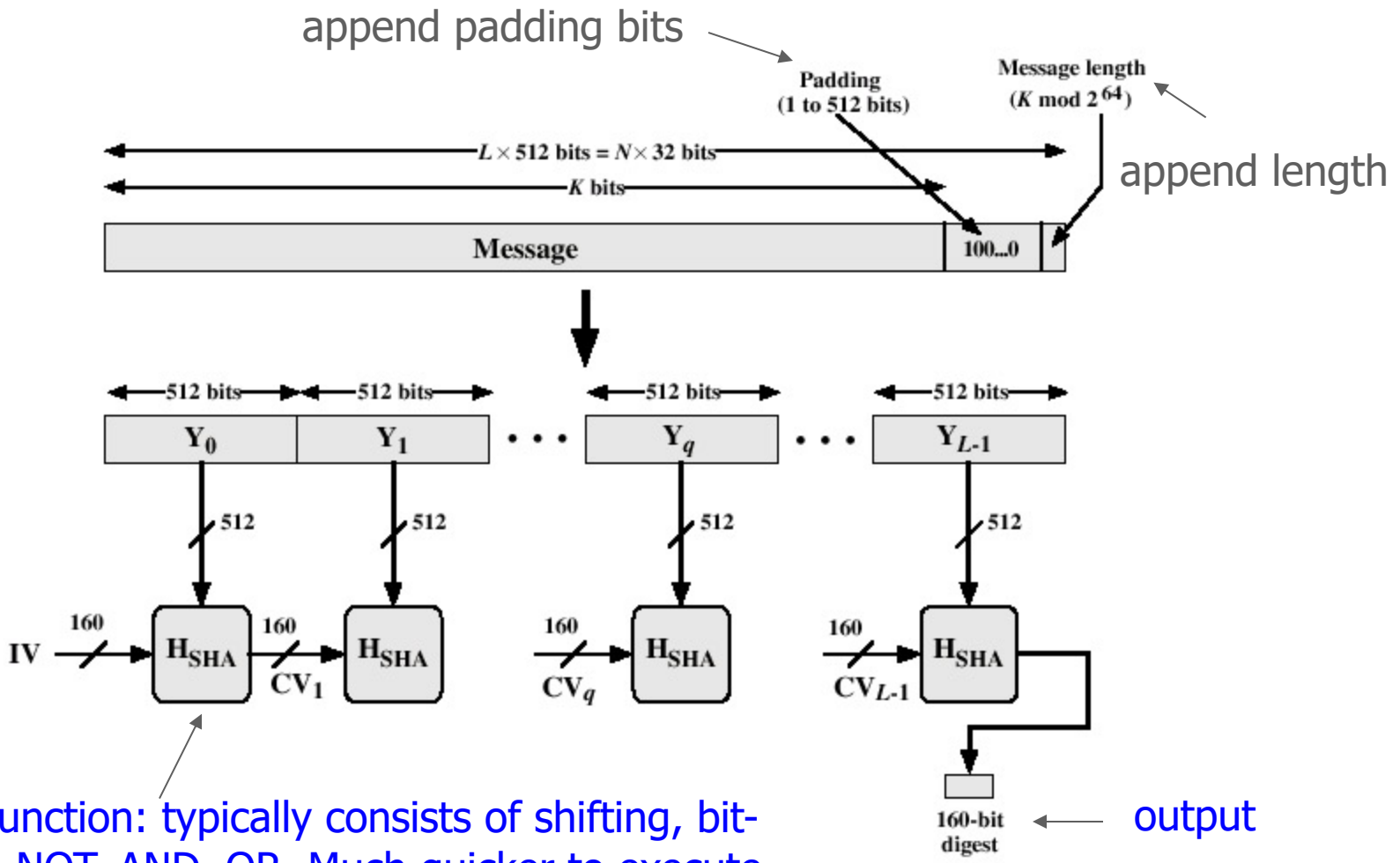
SHA-1 shattered

110 GPU
1 year

SHA-1 bruteforce

12,000,000 GPU
1 year

SHA-1 Secure Hash Function



compression function: typically consists of shifting, bit-rotation, XOR, NOT, AND, OR. Much quicker to execute than encryption

Every bit of the hash code is a function of every bit of the input!

RIPEMD-160

- European RIPE Project – 1997
- Same group launched an attack on MD5
- Extended from 128 to 160-bit message digest

Comparison of Secure HASH functions

	SHA-1	MD5	RIPEMD-160
Digest length	160 bits	128 bits	160 bits
Basic unit of processing	512 bits	512 bits	512 bits
Number of steps	80 (4 rounds of 20)	64 (4 rounds of 16)	160 (5 paired rounds of 16)
Maximum message size	2^{64}-1 bits	∞	∞
Sample relative Speed (on 90MHz Pentium) http://www.esat.kuleuven.ac.be/~bosselae/fast.html	6.88 Mbyte/sec	17.09 Mbyte/sec	5.69 Mbyte/sec

The SHA-2 Family

- SHA-2 is a set of cryptographic hash functions:
  - SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224,SHA-512/256
- Designed by NSA and published by NIST in 2001 as a U.S. FIPS (Federal Information Processing Standard).
- SHA-2 bears some similarities with SHA-1 but contains some key changes.
  - Attacks on SHA-1 cannot be readily extended to SHA-2.

Algorithm and variant		Output size (bits)	Internal state size (bits)	Block size (bits)	Max message size (bits)	Word size (bits)	Rounds	Operations	Collisions found	Example Performance (MiB/s)[13]
MD5 (as reference)		128	128	512	$2^{64} - 1$	32	64	+,and,or,xor,rot	Yes	255
SHA-0		160	160	512	$2^{64} - 1$	32	80	+,and,or,xor,rot	Yes	-
SHA-1		160	160	512	$2^{64} - 1$	32	80	+,and,or,xor,rot	Theoretical attack (2^{51})[14]	153
SHA-2	*SHA-224* *SHA-256*	224 256	256	512	$2^{64} - 1$	32	64	+,and,or,xor,shr,rot	Yes	111
	SHA-384 *SHA-512* *SHA-512/224* *SHA-512/256*	384 512 224 256	512	1024	$2^{128} - 1$	64	80	+,and,or,xor,shr,rot	None	99
SHA-3		224/256/384/512	1600 (5x5 array of 64 bit words)			64	120 (default)		None	

Recent Results on SHA-2 Attacks

Published in	Year	Attack method	Attack	Variant	Rounds	Complexity
New Collision Attacks Against Up To 24-step SHA-2[32][33]	2008	Differential	Collision	SHA-256	24/64	$2^{15.5}$
				SHA-512	24/80	$2^{22.5}$
Preimages for step-reduced SHA-2[34]	2009	Meet-in-the-middle	Preimage	SHA-256	42/64	$2^{251.7}$
					43/64	$2^{254.9}$
				SHA-512	42/80	$2^{502.3}$
					46/80	$2^{511.5}$
Advanced meet-in-the-middle preimage attacks[35]	2010	Meet-in-the-middle	Preimage	SHA-256	42/64	$2^{248.4}$
				SHA-512	42/80	$2^{494.6}$
Higher-Order Differential Attack on Reduced SHA-256[2]	2011	Differential	Pseudo-collision	SHA-256	46/64	2^{178}
					33/64	2^{46}
Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 family[1]	2011	Biclique	Preimage	SHA-256	45/64	$2^{255.5}$
				SHA-512	50/80	$2^{511.5}$
			Pseudo-preimage	SHA-256	52/64	2^{255}
				SHA-512	57/80	2^{511}

Recent Results on SHA-2 Attacks (cont'd)

Published in	Year	Attack method	Attack	Variant	Rounds	Complexity
Improving Local Collisions: New Attacks on Reduced SHA-256[36]	2013	Differential	Collision	SHA-256	31/64	$2^{65.5}$
			Pseudo-collision	SHA-256	38/64	2^{37}
Branching Heuristics in Differential Collision Search with Applications to SHA-512[37]	2014	Heuristic differential	Pseudo-collision	SHA-512	38/80	$2^{40.5}$
Analysis of SHA-512/224 and SHA-512/256[38]	2016	Differential	Collision	SHA-256	28/64	practical
				SHA-512	27/80	practical
			Pseudo-collision	SHA-512	39/80	practical
New Records in Collision Attacks on SHA-2[39]	2023	Differential	Collision	SHA-256	31/64	$2^{49.8}$
				SHA-512	31/80	$2^{115.6}$
			Pseudo-collision	SHA-256	39/64	practical

The NIST SHA-3 Competition (2006-2012)

http://csrc.nist.gov/groups/ST/hash/sha-3/index.html

- On Dec. 9, 2010, the Final FIVE candidates for the Round 3 of the competition were announced:
  - http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/documents/Email_Announcing_Finalists.pdf
  - http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/submissions_rnd3.html
- The Winning algorithm: Keccak, (pronounced “catch-ack”) was announced on Oct 2, 2012, to be called SHA-3 in Standards ;
  - Designed by a team of researchers from Belgium and Italy
  - http://keccak.noekeon.org
  - NSA believes both SHA-2 and SHA-3 are secure and can be used in practice.
    - Since SHA-2 and SHA-3 differ substantially in their designs and theory, this diversity can provide system designers a fallback solution in case one of them is broken in the future.
- SHA-3 approved as a new hashing standard by NIST of U.S..
  - Published as FP202 on Aug. 5, 2015.

The NIST SHA-3 Competition Timeline

Date	Event	Candidates Left
11/2/2007	Call for Proposals published, competition began	
10/31/2008	SHA3 submission deadline	***64***
12/10/2008	*First-round candidates announced*	*51*
2/25/2009	First SHA3 workshop in Leuven, Belgium	51
7/24/2009	*Second-round candidates announced*	*14*
8/23/2010	Second SHA3 workshop in Santa Barbara, CA	14
12/9/2010	*SHA3 finalists announced*	*5*
3/22/2012	Third SHA3 workshop in Washington, DC	5
10/2/2012	*Keccak announced as the SHA3 winner*	***1***

The 5 Finalists for SHA-3 Competition

BLAKE, Grostl, JH, Keccak, Skein

- Published selection in Dec 2010
- Cryptanalytic results were harder to interpret
- Often distinguishers of no apparent relevance
- All five finalists made tweaks for third round
  - BLAKE and JH increased number of rounds
  - Grostl changed internals of Q permutation
  - Keccak changed padding rules
  - Skein changed key schedule constant

Choosing a Winner: Security

- Nobody was knocked out by cryptanalysis
- Different algorithms got different depth of cryptanalysis
  - Grostl, BLAKE, Skein, Keccak, JH
- Keccak and Blake had best security margins
- Domain extenders (aka chaining modes) all had security proofs
- Grostl had a very big tweak, Skein a significant one
- ARX vs non-ARX designs
  - ARX = Addition (mod 2^n), Rotation, XOR

Keccak looks very strong, and seems to have been analyzed in sufficient depth to give the Judging Panel confidence.

Choosing a Winner: Performance

- All five finalists have acceptable performance
- ARX designs (BLAKE and Skein) are excellent on high-end software implementations
- JH and Grostl fairly slow in software
  - Slower than SHA2
- Keccak is very hardware friendly
  - High throughput per area

Keccak performs well everywhere, and very well in hardware.

Complementing SHA2

- SHA3 is expected to deployed into a world full of SHA2 implementations
- SHA2 still looks strong
- NIST expect the standards to coexist.
- SHA3 should *complement* SHA2.
  - Good in different environments
  - Susceptible to different analytical insights

Keccak is fundamentally different from SHA2. Its performance properties and implementation tradeoffs have little in common with SHA2.

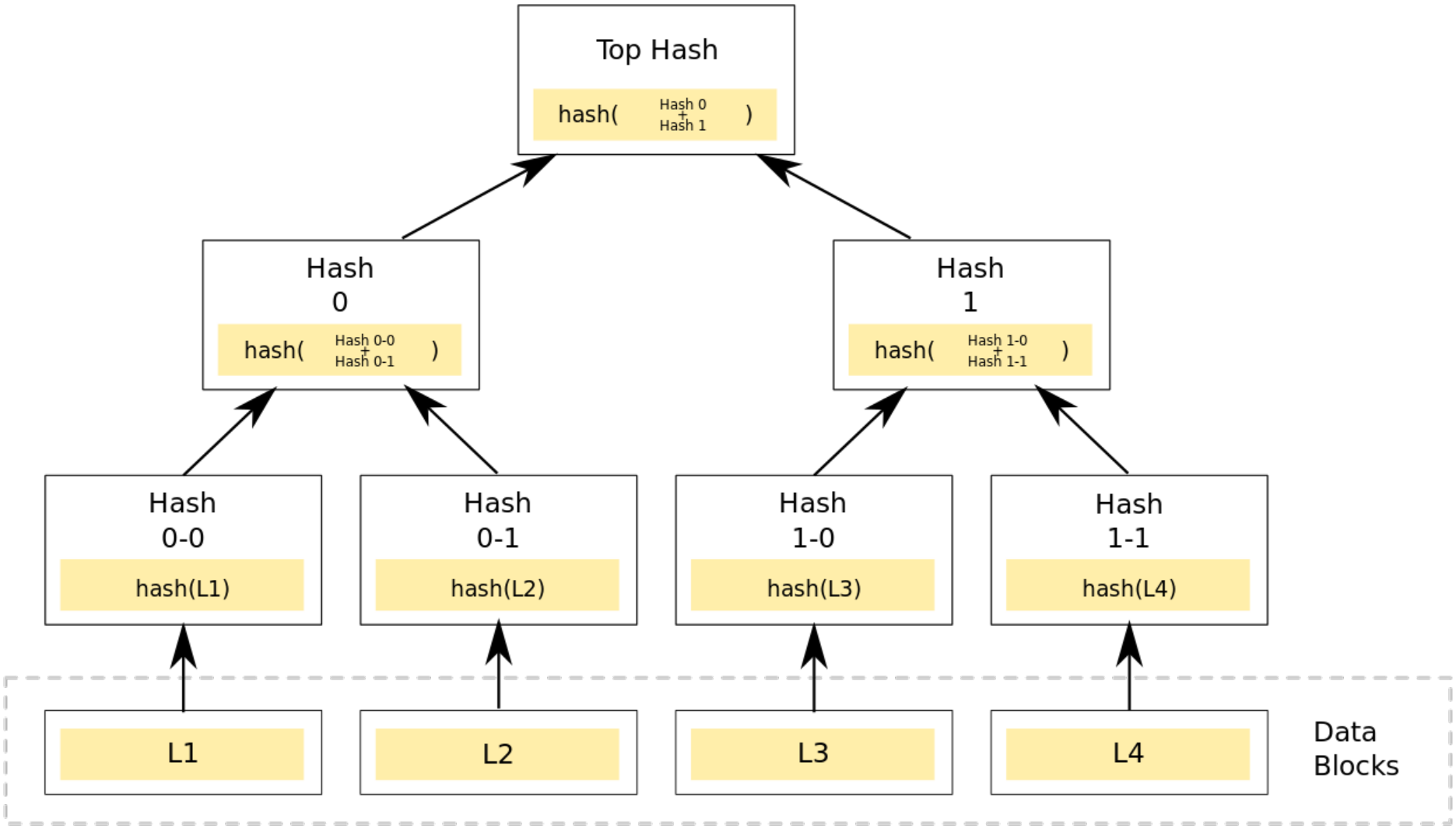
Reasons for Keccak selected as the Winner

- High security margin
- Fairly high quality, in-depth analysis
- Elegant, clean design
- Excellent hardware performance
- Good overall performance
- Flexibility: rate is readily adjustable
- Design diversity from SHA2

Taking Keccak as SHA3: Goals/ Requirements

- Play well with existing applications
  - DRBGs (Deterministic Random Bit Generators), KDFs (Key Derivation Functions), HMAC, signatures
- Drop-in replacements
  - SHA-224, -256, -384, -512, and even SHA1 and MD5
- Fast and efficient everywhere
- Benefit from Tree Hashing
- Benefit from Keccak extras
  - Variable output length, efficient PRF, authenticated encryption, DRBG

A Hash Tree (Merkle Tree)



New Attacks on SHA-2 discovered during SHA-3 Competition

Published in	Year	Attack method	Attack	Variant	Rounds	Complexity
New Collision attacks Against Up To 24-step SHA-2 [25]	2008	Deterministic	Collision	SHA-256	24/64	$2^{28.5}$
				SHA-512	24/80	$2^{32.5}$
Preimages for step-reduced SHA-2 [26]	2009	Meet-in-the-middle	Preimage	SHA-256	42/64	$2^{251.7}$
					43/64	$2^{254.9}$
				SHA-512	42/80	$2^{502.3}$
					46/80	$2^{511.5}$
Advanced meet-in-the-middle preimage attacks [27]	2010	Meet-in-the-middle	Preimage	SHA-256	42/64	$2^{248.4}$
				SHA-512	42/80	$2^{494.6}$
Higher-Order Differential Attack on Reduced SHA-256 [2]	2011	Differential	Pseudo-collision	SHA-256	46/64	2^{178}
					46/64	2^{46}
Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 family [1]	2011	Biclique	Preimage	SHA-256	45/64	$2^{255.5}$
				SHA-512	50/80	$2^{511.5}$
			Pseudo-preimage	SHA-256	52/64	2^{255}
				SHA-512	57/80	2^{511}
Improving Local Collisions: New Attacks on Reduced SHA-256 [28]	2013	Differential	Collision	SHA-256	31/64	$2^{65.5}$
			Pseudo-collision	SHA-256	38/64	2^{37}
Branching Heuristics in Differential Collision Search with Applications to SHA-512 [29]	2014	Heuristic Differential	Pseudo-collision	SHA-512	38/80	$2^{40.5}$

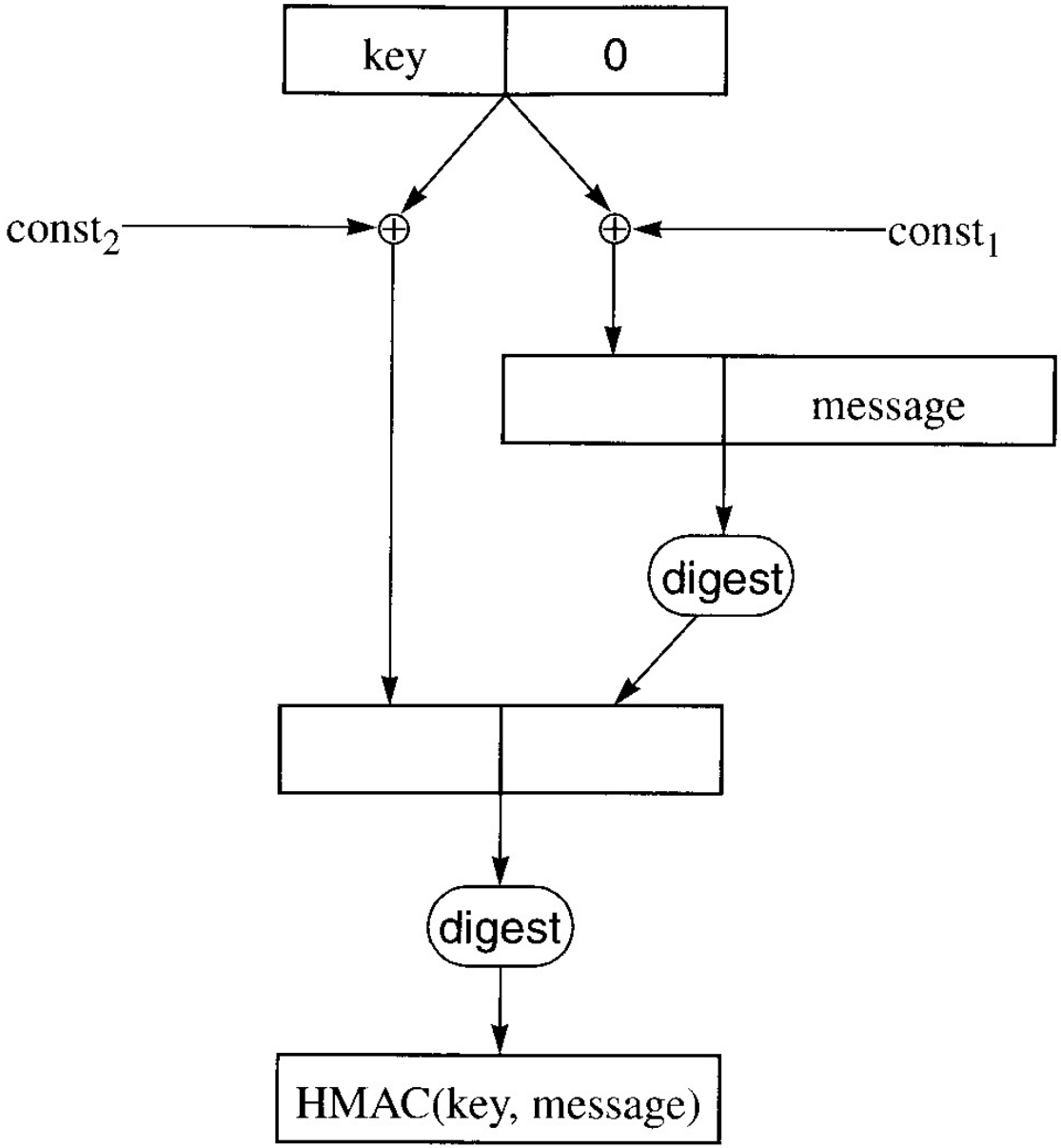
Comparison of SHA functions

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) [41]		First published
									Long messages	8 bytes	
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)[42]	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)[43]		3.47	52.00	1995
SHA-2	*SHA-224*	224	256 (8 × 32)	512	64	And, Xor, Or, Rot, Shr, Add (mod 2^{32})	112	32	7.62	84.50	2004
	SHA-256	256					128	0	7.63	85.25	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[44]	5.06	135.50	2001
	SHA-512/224	224					112	288	≈ SHA-384	≈ SHA-384	2012
	SHA-512/256	256					128	256			
SHA-3	*SHA3-224*	224	1600 (5 × 5 × 64)	1152	24[45]	And, Xor, Rot, Not	112	448	8.12	154.25	2015
	SHA3-256	256		1088			128	512	8.59	155.50	
	SHA3-384	384		832			192	768	11.06	164.00	
	SHA3-512	512		576			256	1024	15.88	164.00	
	SHAKE128	*d* (arbitrary)		1344			min(*d*/2, 128)	256	7.08	155.25	
	SHAKE256	*d* (arbitrary)		1088			min(*d*/2, 256)	512	8.59	155.50	

The Future of Hash Security is Diversity

	Security Claim	Fixed prefix	Chosen attack
~~SHA-1~~	MD		
SHA-256	MD	2^{128}	
SHA-3	Sponge	2^{128}	2^{128}
BLAKE	HAIFA	2^{128}	2^{256}

HMAC



By XORing key with const1 and const2, we have pseudo-randomly generated two new keys from the original key

HMAC

- Effort to develop a MAC derived from a cryptographic hash codes such as SHA-256
- Executes faster in software
- No export restrictions
- Relies on a secret key
- RFC 2104 list design objectives and
- Provable security properties
- Used in IPsec, TLS
- Can use different digest functions as a component, e.g.
  - HMAC-SHA256 , HMAC-SHA3 ;
- Informational RFC6151 (circa 2011) concluded that: Although the security of MD5 hash function itself is severely compromised, the currently known “attacks on HMAC-MD5 do not seem to indicate a practical vulnerability when used as a message authentication code,” but “for new protocol design, a ciphersuite with HMAC-MD5 should NOT be included.”

The Nostradamus Project (circa 2007)

- https://marc-stevens.nl/research/hashclash/Nostradamus/index.html

- Predicting the 2008 US Presidential Election Result using PS3 and MD5:

Announcement	**We have used a Sony Playstation 3 to correctly predict the outcome of the 2008 US presidential elections. In order not to influence the voters we keep our prediction secret, but commit to it by publishing its cryptographic hash on this website. The document with the correct prediction and matching hash will be revealed after the elections.**

- To illustrate another Common Application of Secure Hash Function:
  - To Commit a Secret