Hash Functions

EJ Jung

Hash Functions

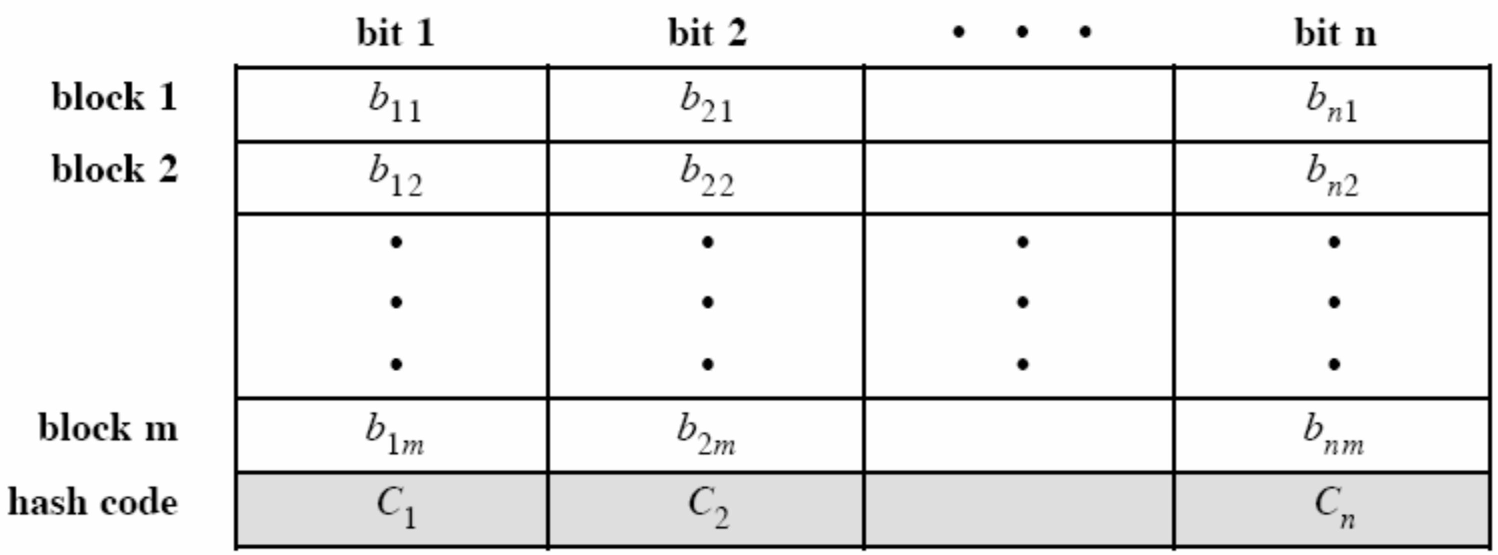
- Purpose of the HASH function is to produce a ”fingerprint”.
- But, what do you mean by fingerprint??

Secure Hash Functions

- Properties of a HASH function H :

1. H can be applied to a block of data at any size
2. H produces a fixed length output
3. H(x) is easy to compute for any given x.
4. For any given block x, it is computationally infeasible to find x such that H(x) = h
5. For any given block x, it is computationally infeasible to find $y \neq x$ with H(y) = H(x).
6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y)

Simple hash function

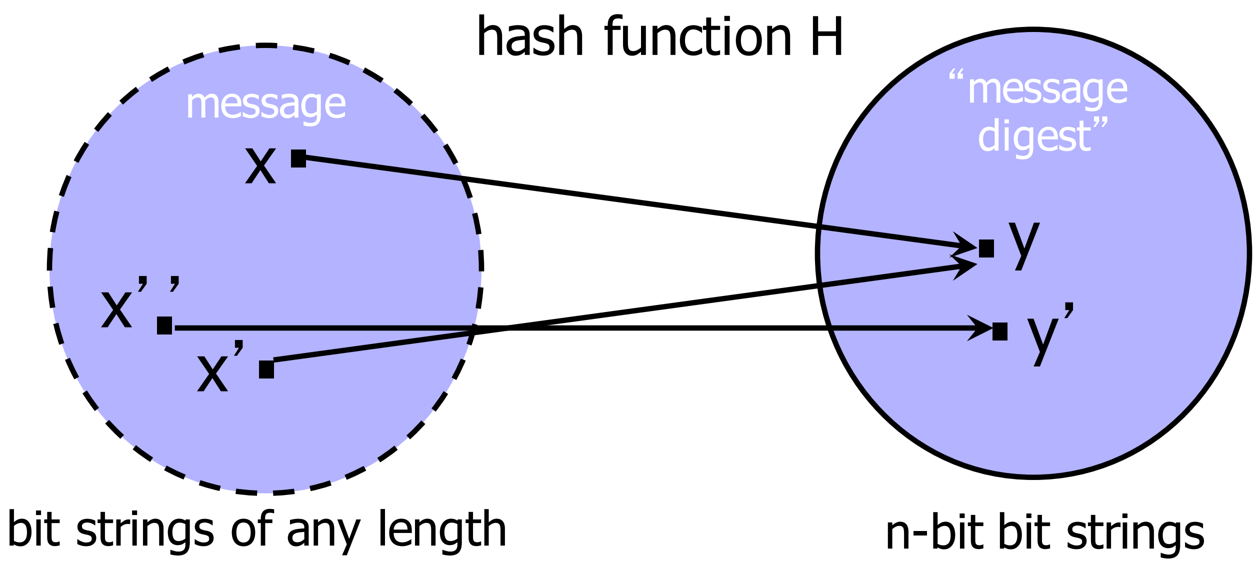


	bit 1	bit 2	• • •	bit n
block 1	b_{11}	b_{21}		b_{n1}
block 2	b_{12}	b_{22}		b_{n2}
	⋮	⋮	⋮	⋮
block m	b_{1m}	b_{2m}		b_{nm}
hash code	C_1	C_2		C_n

Figure 3.3 Simple Hash Function Using Bitwise XOR

Example

Hash Functions: Main Idea



- H is a lossy compression function
  - Collisions: h(x)=h(x') for some inputs x, x'
  - Result of hashing should "look random" (make this precise later)
    - Intuition: half of digest bits are "1"; any bit in digest is "1" half the time
- Cryptographic hash function needs a few properties…

One-Way

- Intuition: hash should be hard to invert
  - "Preimage resistance"
  - Let $h(x')=y \in \{0,1\}^n$ for a random x'
  - Given y, it should be hard to find any x such that $h(x)=y$
- How hard?
  - Brute-force: try every possible x, see if $h(x)=y$
  - SHA-1 (common hash function) has 160-bit output
    - Suppose have hardware that'll do 2^{30} trials a pop
    - Assuming 2^{34} trials per second, can do 2^{89} trials per year
    - Will take 2^{71} years to invert SHA-1 on a random image

"Birthday Paradox"

- T people
- Suppose each birthday is a random number taken from K days (K=365) – how many possibilities?
  - K^T (samples with replacement)
- How many possibilities that are all different?
  - $(K)_T = K(K-1)\ldots(K-T+1)$ samples without replacement
- Probability of no repetition?
  - $(K)_T/K^T \approx 1 - T(T-1)/2K$
- Probability of repetition?
  - $O(T^2)$

Collision Resistance

- Should be hard to find x, x' such that h(x)=h(x')
- Brute-force collision search is $O(2^{n/2})$, <u>not</u> $O(2^n)$
  - n = number of bits in the output of hash function
  - For SHA-1, this means $O(2^{80})$ vs. $O(2^{160})$
- Reason: birthday paradox
  - Let T be the number of values x,x',x'' … we need to look at before finding the first pair x,x' s.t. h(x)=h(x')
  - Assuming h is random, what is the probability that we find a repetition after looking at T values? $O(T^2)$
  - Total number of pairs? $O(2^n)$
  - Conclusion: T [illegible] $O(2^{n/2})$

One-Way vs. Collision Resistance

- One-wayness does not imply collision resistance
  - Suppose g is one-way
  - Define h(x) as g(x') where x' is x except the last bit
    - h is one-way (to invert h, must invert g)
    - Collisions for h are easy to find: for any x, h(x0)=h(x1)
- Collision resistance does not imply one-wayness
  - Suppose g is collision-resistant
  - Define h(x) to be 0x if x is n-bit long, 1g(x) otherwise
    - Collisions for h are hard to find: if y starts with 0, then there are no collisions, if y starts with 1, then must find collisions in g
    - h is not one way: half of all y's (those whose first bit is 0) are easy to invert (how?); random y is invertible with probab. 1/2

Weak Collision Resistance

- Given randomly chosen x, hard to find x' such that h(x)=h(x')
  - Attacker must find collision for a specific x. By contrast, to break collision resistance, enough to find any collision.
  - Brute-force attack requires O(2^n) time
- Weak collision resistance does not imply collision resistance (why?)

Which Property Do We Need?

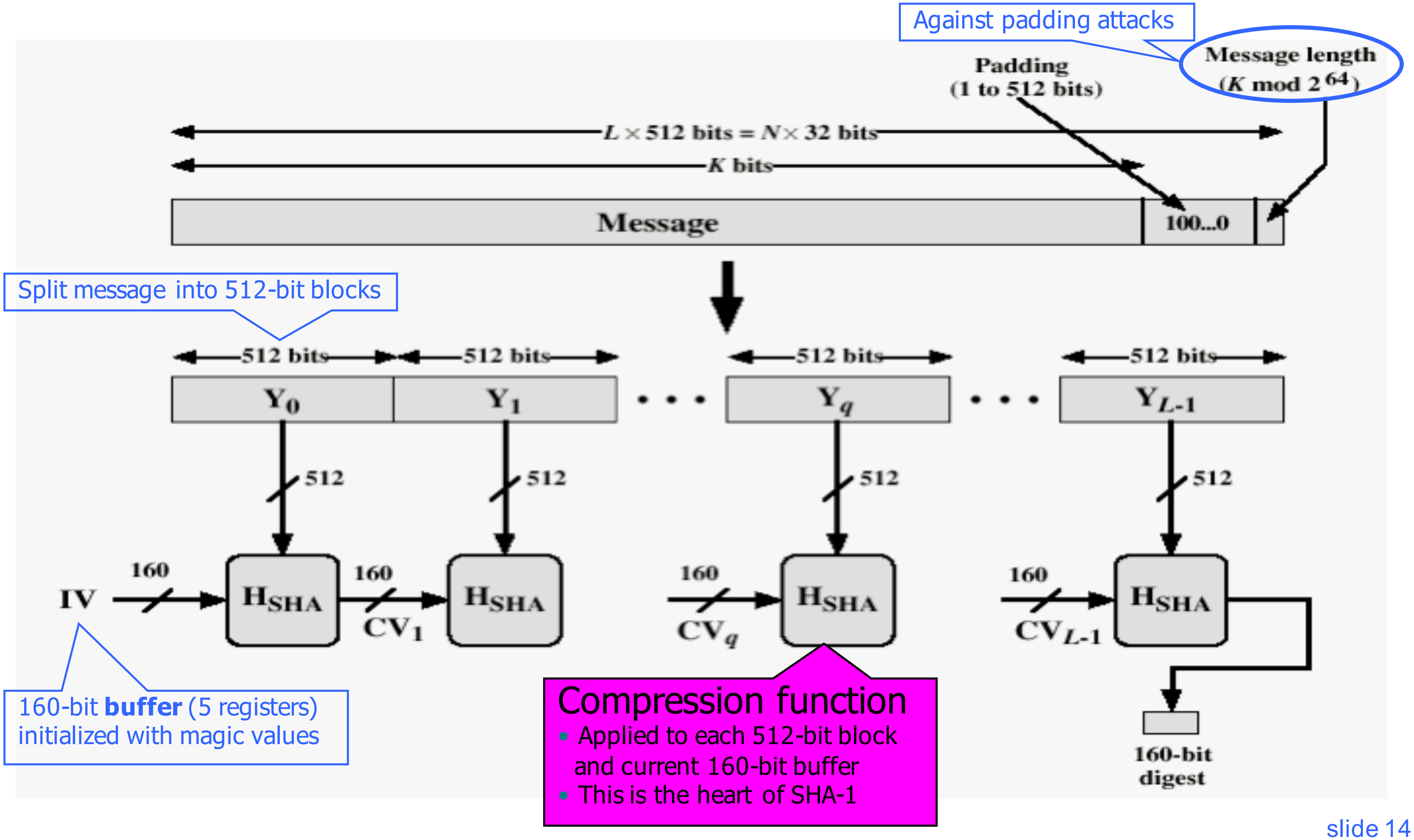
- UNIX passwords stored as hash(password)
  - One-wayness: hard to recover password
- Integrity of software distribution
  - Weak collision resistance
  - But software images are not really random… maybe need full collision resistance
- Auction bidding
  - Alice wants to bid B, sends H(B), later reveals B
  - One-wayness: rival bidders should not recover B
  - Collision resistance: Alice should not be able to change her mind to bid B' such that H(B)=H(B')

Common Hash Functions

- MD5
  - 128-bit output
  - Still used very widely
  - Completely broken by now
- RIPEMD-160
  - 160-bit variant of MD-5
- SHA-1 (Secure Hash Algorithm)
  - 160-bit output
  - US government (NIST) standard as of 1993-95
    - Also the hash algorithm for Digital Signature Standard (DSS)

UNIVERSITY of SAN FRANCISCO
department of computer science

Basic Structure of SHA-1

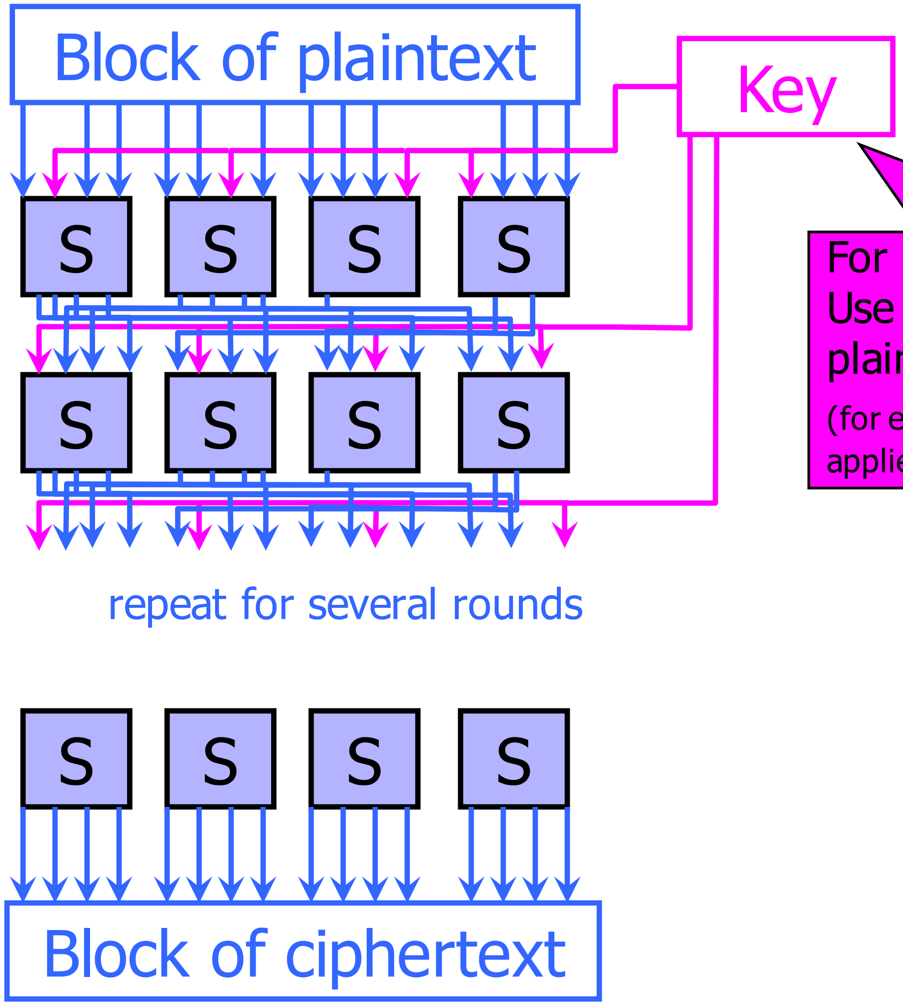


Against padding attacks
Padding (1 to 512 bits)
Message length (K mod 2^{64})
$L \times 512$ bits = $N \times 32$ bits
K bits
Message
100...0
Split message into 512-bit blocks
512 bits
Y_0
Y_1
Y_q
Y_{L-1}
512
IV
160
H_{SHA}
CV_1
CV_q
CV_{L-1}
160-bit digest
160-bit buffer (5 registers) initialized with magic values

Compression function

- Applied to each 512-bit block and current 160-bit buffer
- This is the heart of SHA-1

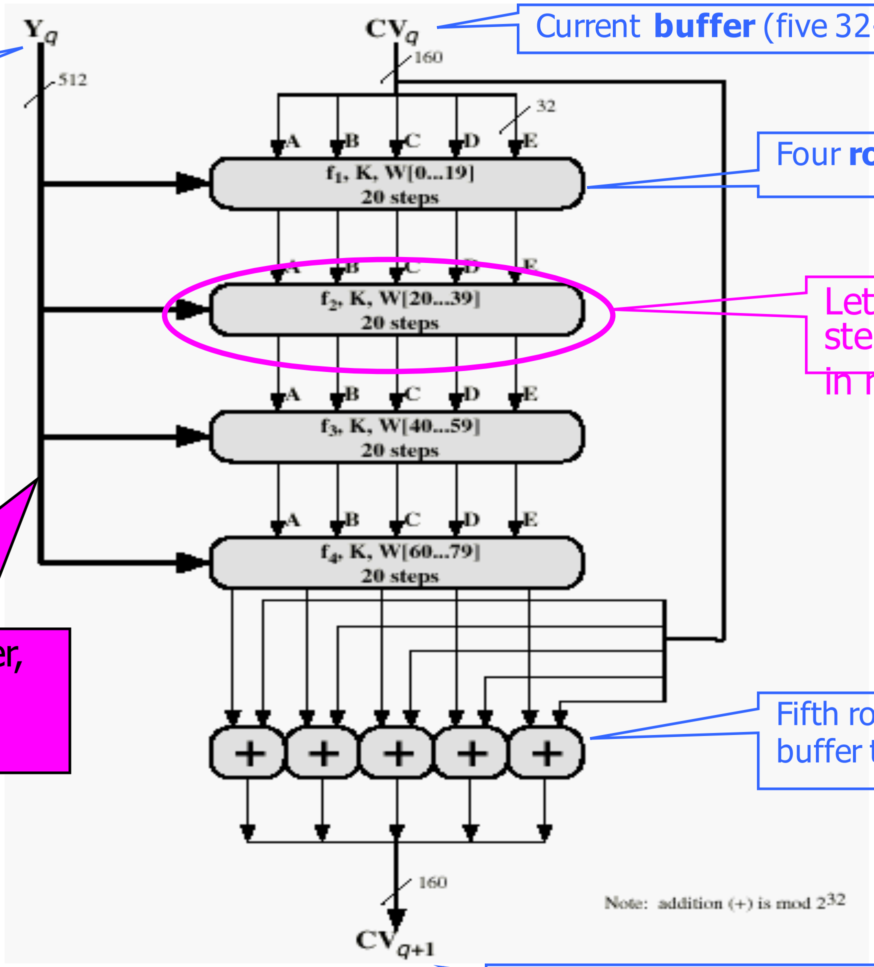
Block Ciphers



For hashing, there is no KEY. Use message as key and replace plaintext with a fixed string.

(for example, Unix password hash is DES applied to NULL with password as the key)

SHA-1 Compression Function



Current **message block**

Current **buffer** (five 32-bit registers A,B,C,D,E)

Four **rounds**, 20 steps in each

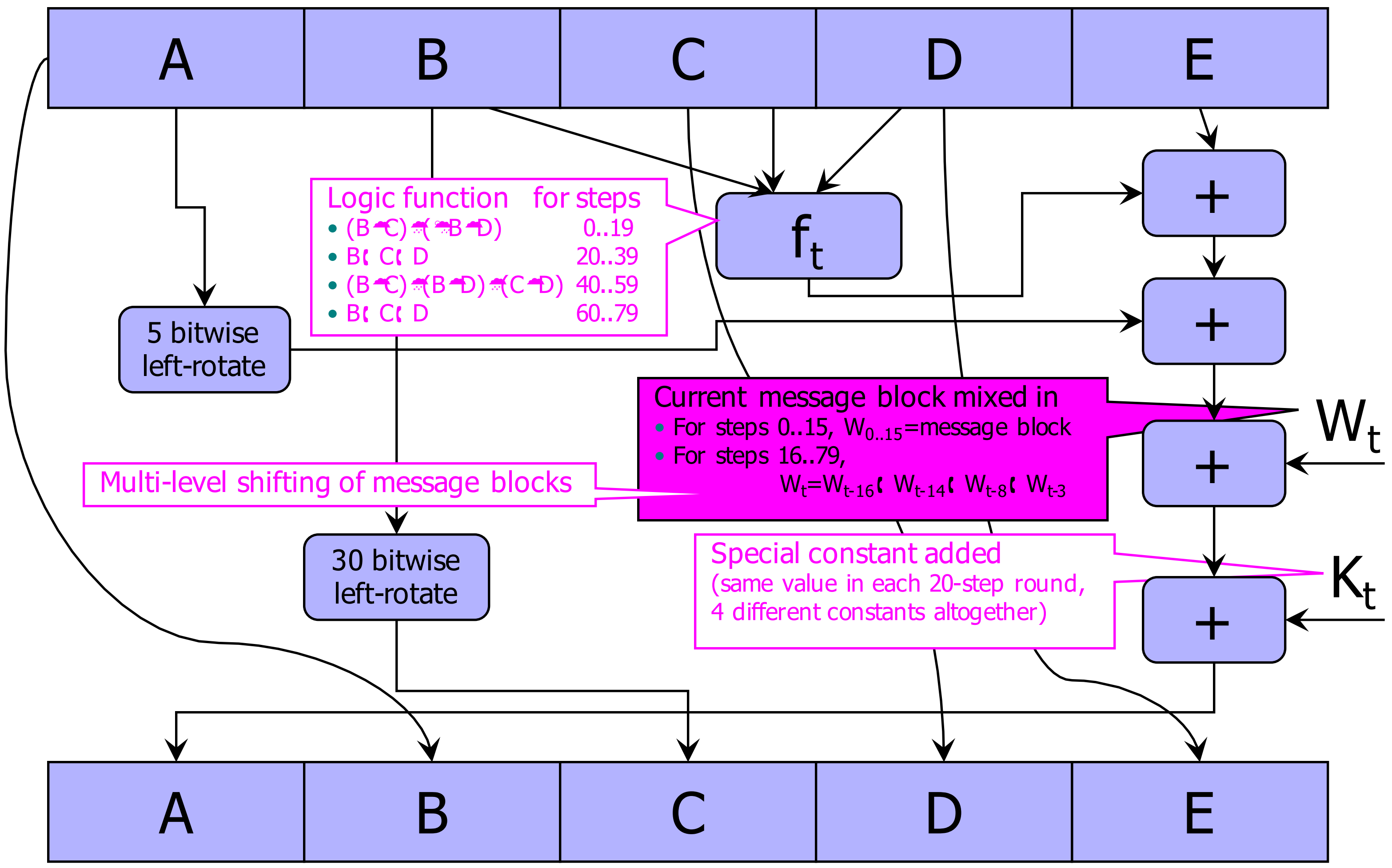
Let's look at each step
in more detail…

Very similar to a block cipher,
with message itself used
as the key for each round

Fifth round adds the original buffer to the result of 4 rounds

Buffer contains final hash value

One Step of SHA-1 (80 steps total)



Logic function	for steps
• $(B \wedge C) \vee (\neg B \wedge D)$	0..19
• $B \oplus C \oplus D$	20..39
• $(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$	40..59
• $B \oplus C \oplus D$	60..79

Current message block mixed in
- For steps 0..15, $W_{0..15}$=message block
- For steps 16..79, $W_t = W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3}$

Multi-level shifting of message blocks

Special constant added
(same value in each 20-step round, 4 different constants altogether)

How Strong Is SHA-1?

- Every bit of output depends on every bit of input
  - Very important property for collision-resistance
- Brute-force inversion requires 2^{160} ops, birthday attack on collision resistance requires 2^{80} ops
- Some recent weaknesses (2005)
  - Collisions can be found in 2^{63} ops

SHA-512 big picture

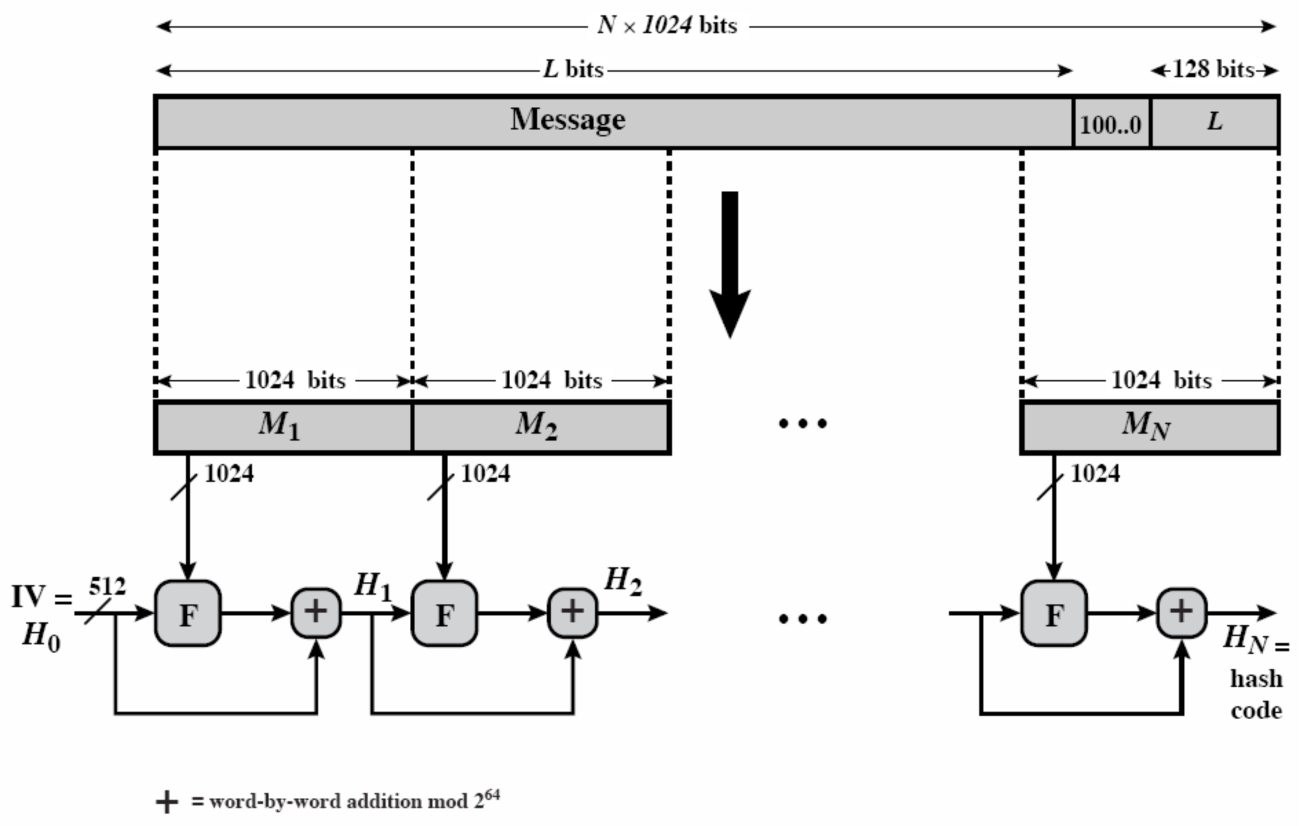


Figure 3.4 Message Digest Generation Using SHA-512

SHA-512 zoom in

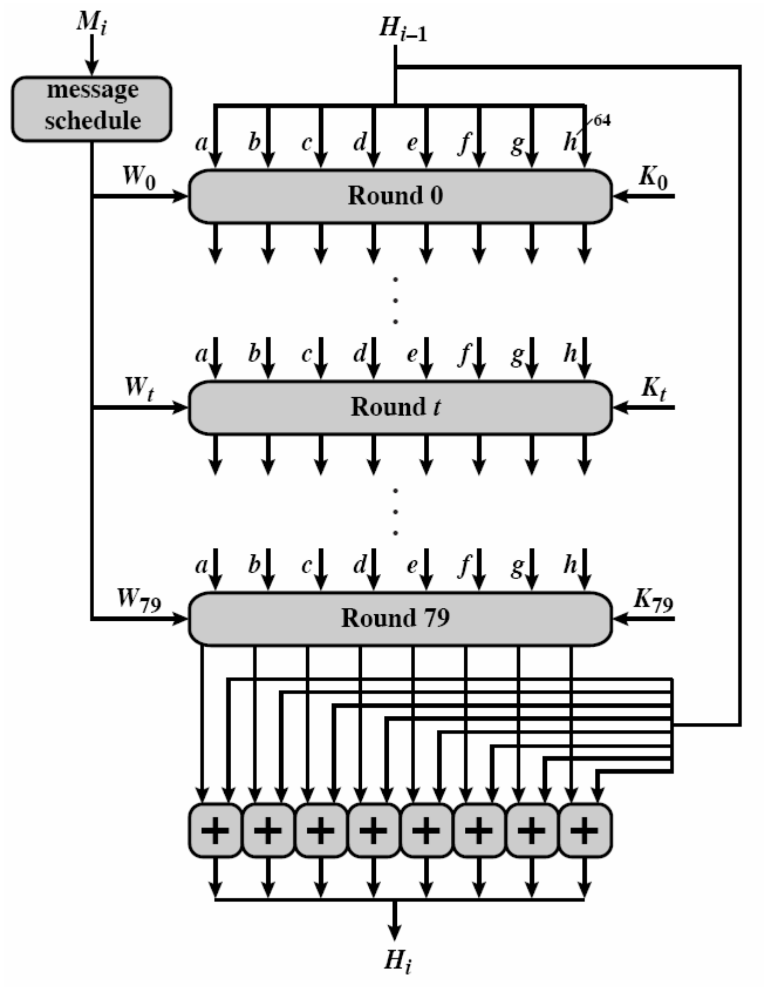


Figure 3.5 SHA-512 Processing of a Single 1024-Bit Block

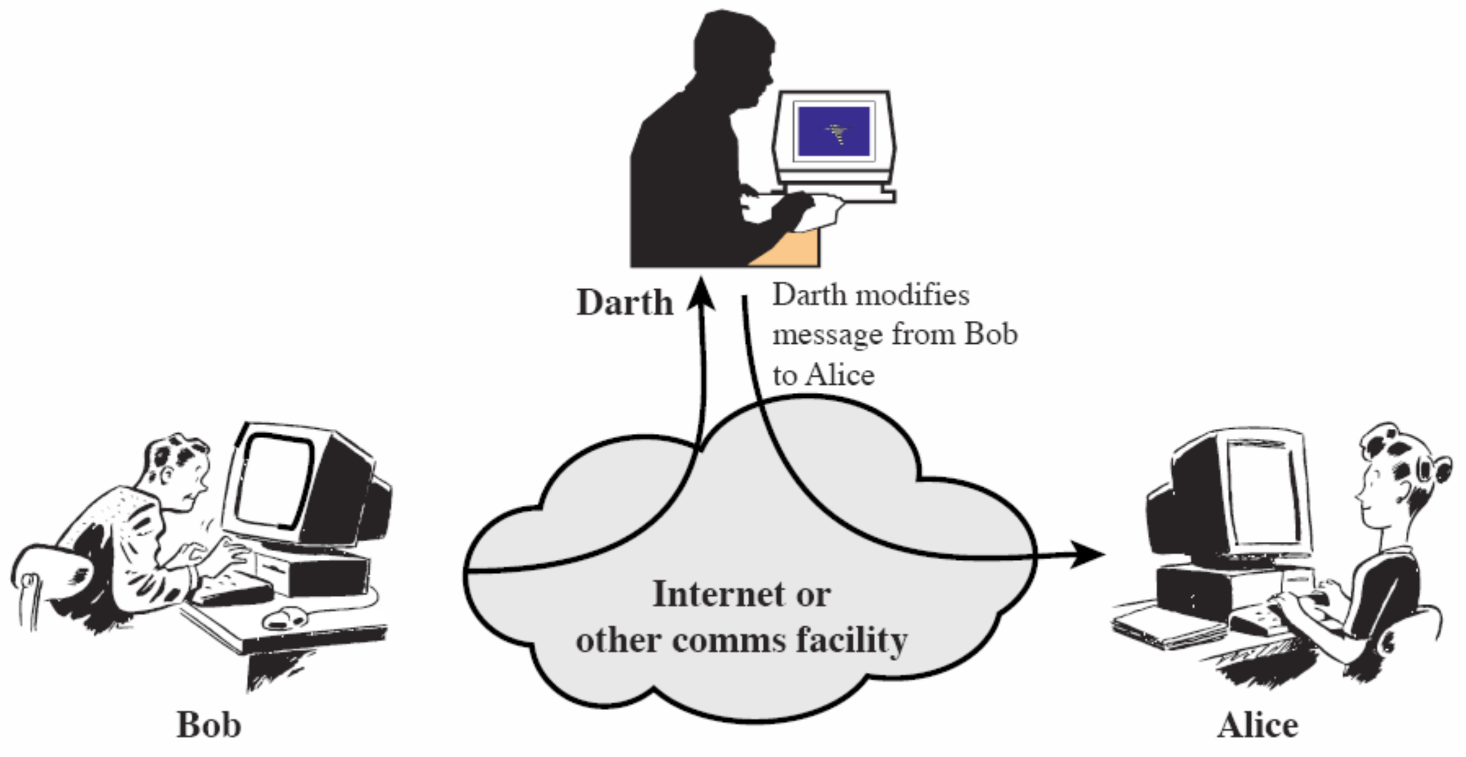
What is “Security”?

- Confidentiality
- Privacy
- Authentication
- Authorization
- anything else?

Where to use hash functions

- Cookie
  - H(server's secret, client's unique information, timestamp)
- Password storage
  - safe against server problems

Integrity checks

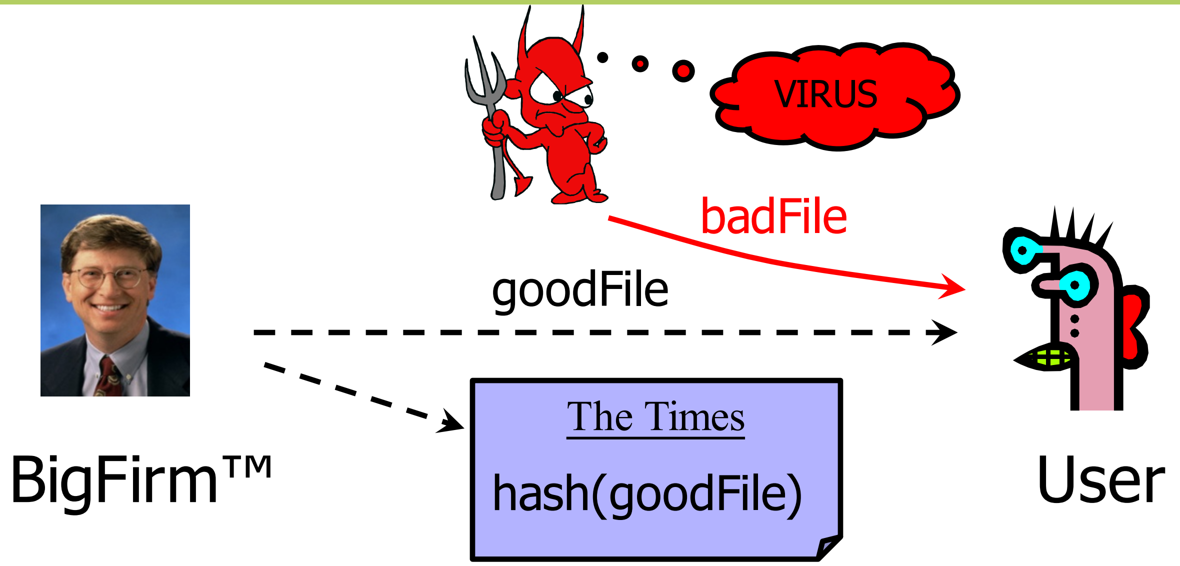

Darth
Darth modifies message from Bob to Alice
Internet or other comms facility
Bob
Alice

(c) Modification of messages

Integrity vs. Confidentiality

- ➢ Integrity: attacker cannot tamper with message
- ➢ Encryption may not guarantee integrity!
  - Intuition: attacker may able to modify message under encryption without learning what it is
    - – Given one-time key K, encrypt M as M⊕K… Perfect secrecy, but can easily change M under encryption to M⊕M' for any M'
    - – Online auction: halve competitor's bid without learning its value
  - This is recognized by industry standards (e.g., PKCS)
    - – "RSA encryption is intended primarily to provide confidentiality… It is not intended to provide integrity"
  - Many encryption schemes provide secrecy AND integrity

More on Integrity



Software manufacturer wants to ensure that the executable file
is received by users without modification…

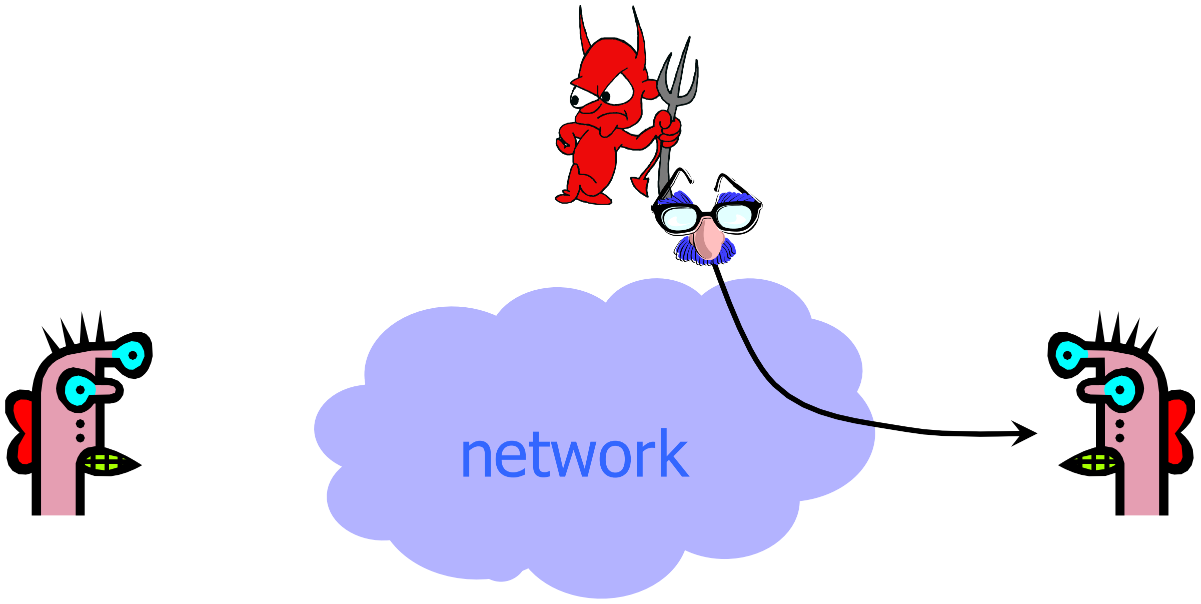
Sends out the file to users and publishes its hash in NY Times

The goal is integrity, not confidentiality

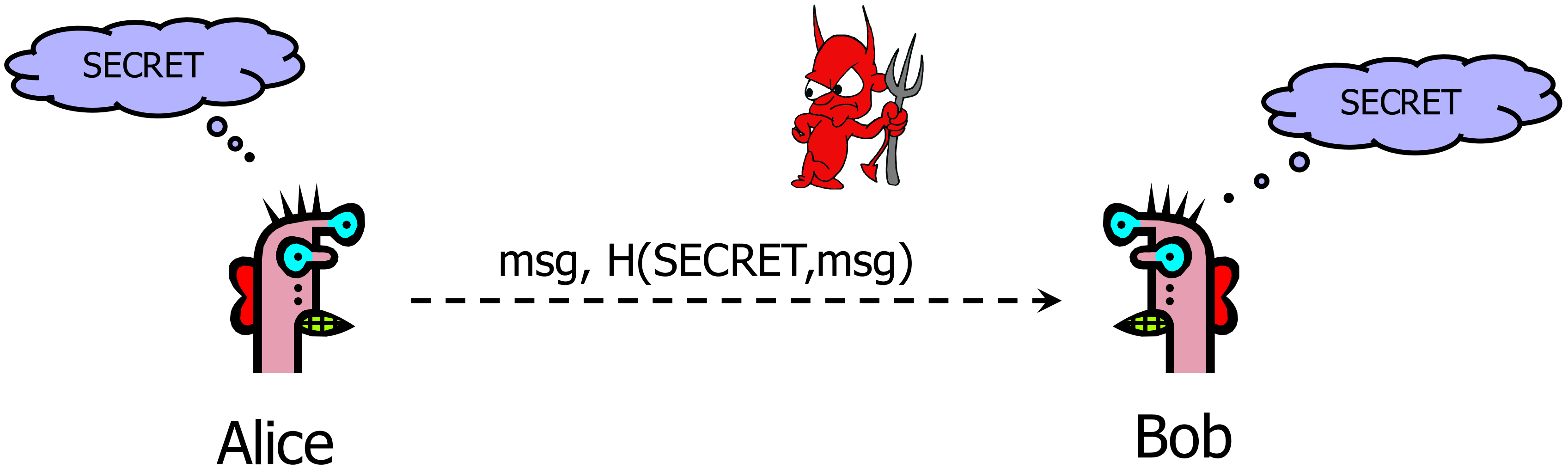
Idea: given goodFile and hash(goodFile),
very hard to find badFile such that hash(goodFile)=hash(badFile)

Authentication

- Authenticity is identification and assurance of origin of information
  - We'll see many specific examples in different scenarios



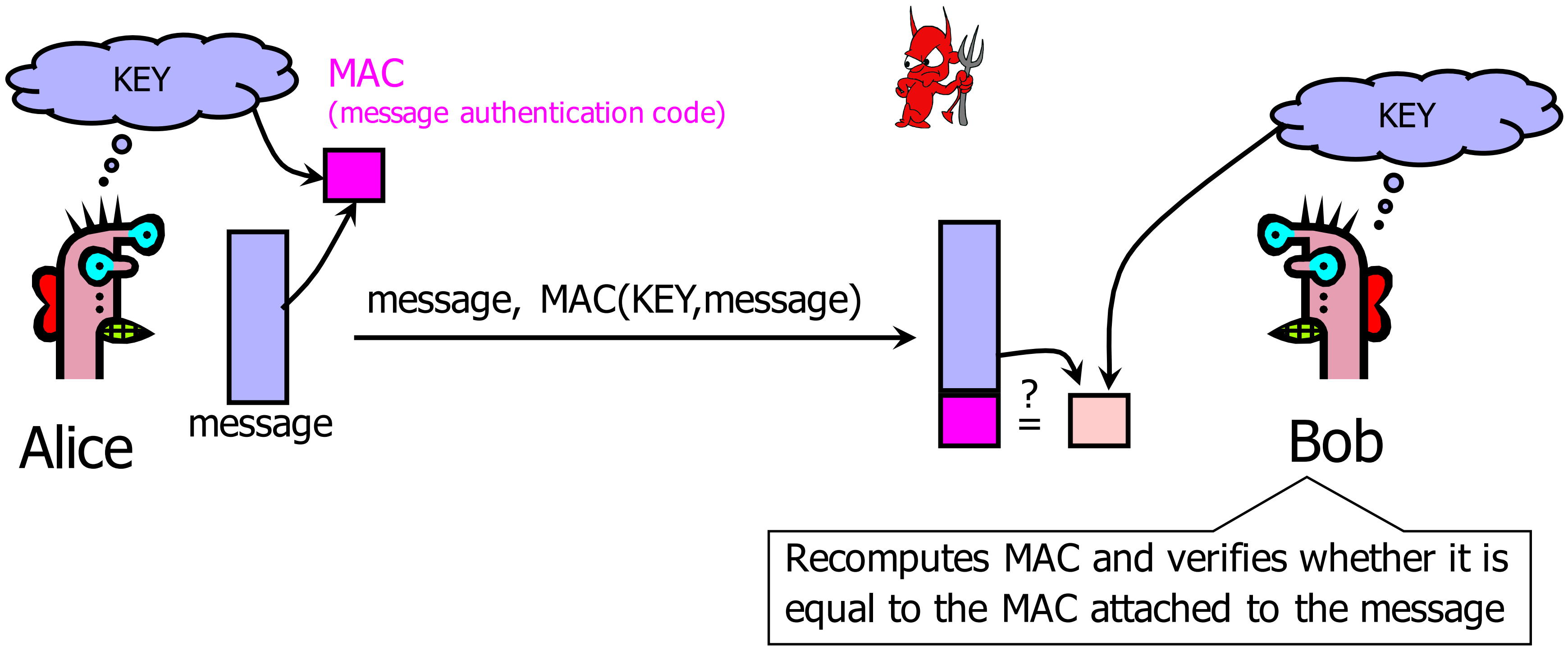
Authentication with Shared Secrets



Alice wants to ensure that nobody modifies message in transit (both integrity and authentication)

Idea: given msg,
very hard to compute H(SECRET, msg) without SECRET;
easy with SECRET

Authentication Without Encryption



KEY
MAC
(message authentication code)
KEY
message, MAC(KEY,message)
Alice
message
?
=
Bob
Recomputes MAC and verifies whether it is equal to the MAC attached to the message

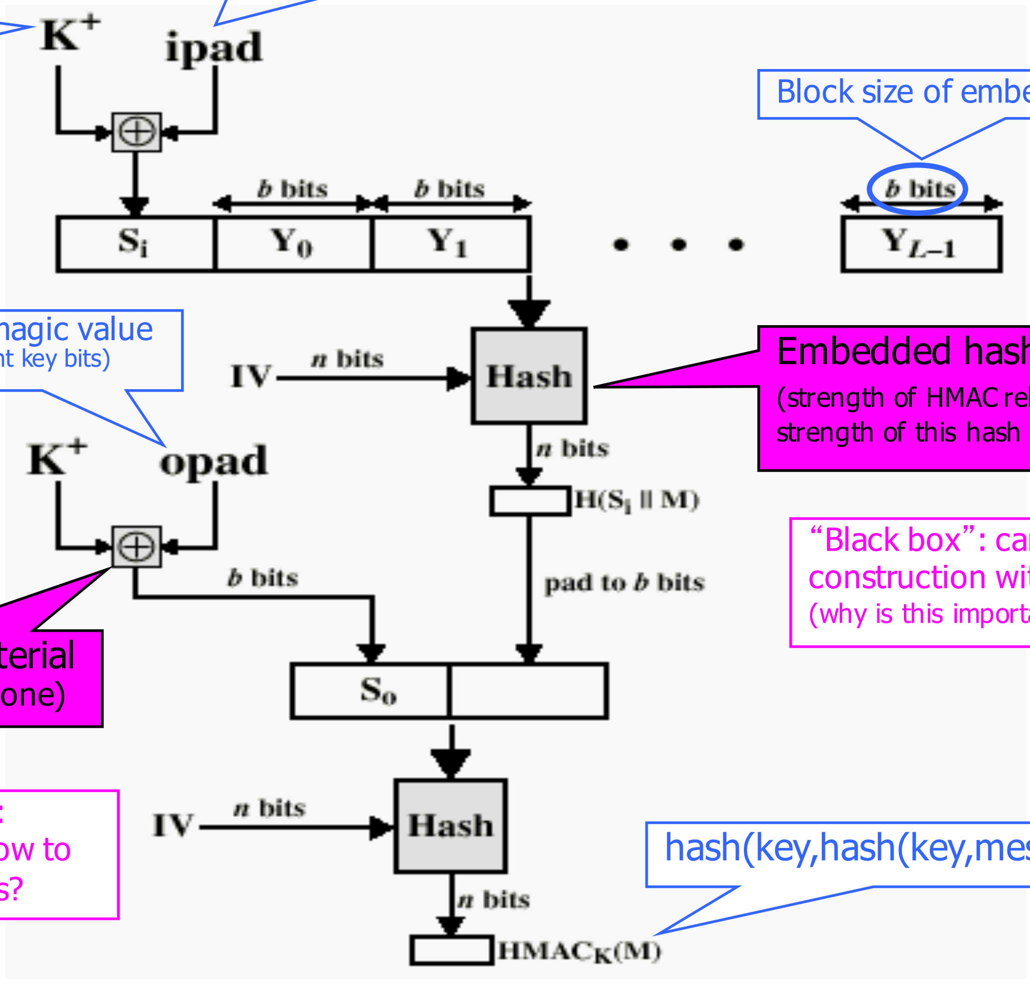
Integrity and authentication: only someone who knows KEY can compute MAC for a given message

usfCS
UNIVERSITY of SAN FRANCISCO
department of computer science

HMAC

- Construct MAC by applying a cryptographic hash function to message and key
  - Could also use encryption instead of hashing, but…
  - Hashing is faster than encryption in software
  - Library code for hash functions widely available
  - Can easily replace one hash function with another
  - There used to be US export restrictions on encryption
- Invented by Bellare, Canetti, and Krawczyk (1996)
- Mandatory for IP security, also used in SSL/TLS

Structure of HMAC



magic value (flips half of key bits)

Secret key padded to block size

Block size of embedded hash function

another magic value (flips different key bits)

Embedded hash function (strength of HMAC relies on strength of this hash function)

"Black box": can use this HMAC construction with any hash function (why is this important?)

"Amplify" key material (get two keys out of one)

Very common problem: given a small secret, how to derive a lot of new keys?

hash(key,hash(key,message))