CS 361S

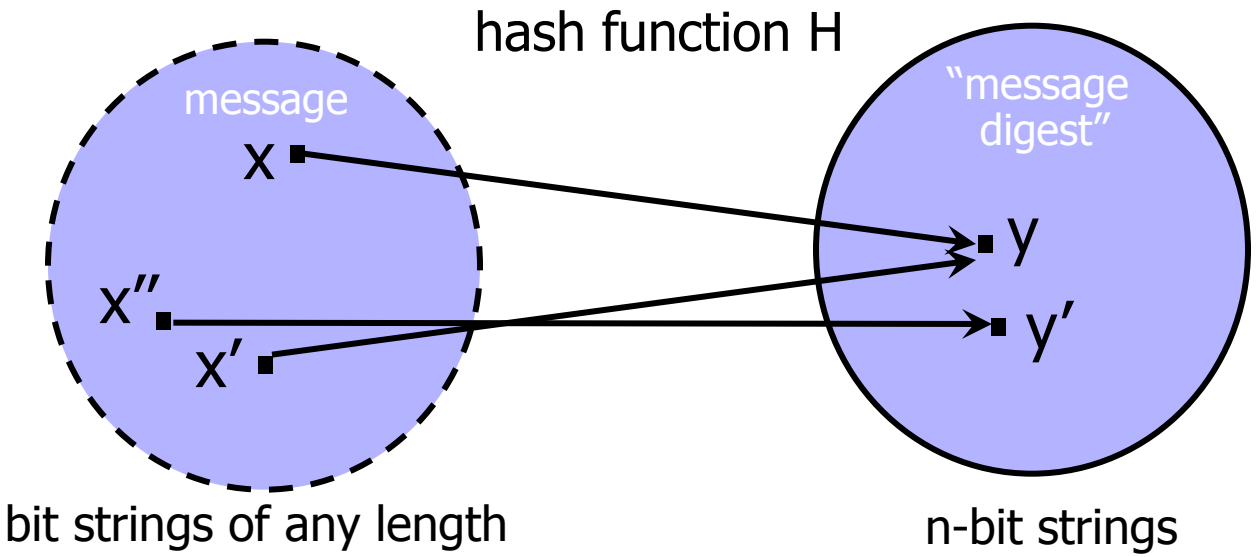
Cryptographic Hash Functions

Vitaly Shmatikov

Reading Assignment

- Read Kaufman 5.1-2 and 5.6-7

Hash Functions: Main Idea



- Hash function H is a lossy compression function
  - Collision: H(x)=H(x′) for some inputs x≠x′
- H(x) should look "random"
  - Every bit (almost) equally likely to be 0 or 1
- A cryptographic hash function must have certain properties

One-Way

- Intuition: hash should be hard to invert
  - "Preimage resistance"
  - Given a random, it should be hard to find any x such that h(x)=y
    - y is an n-bit string randomly chosen from the output space of the hash function, ie, y=h(x') for some x'
- How hard?
  - Brute-force: try every possible x, see if h(x)=y
  - SHA-1 (a common hash function) has 160-bit output
    - Suppose we have hardware that can 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 $x \neq x'$ such that $h(x)=h(x')$
- Birthday paradox
  - Let T be the number of values x, x', x''… we need to look at before finding the first pair $x \neq x'$ 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? $O(2^n)$
    - n = number of bits in the output of hash function
  - Conclusion: $T \approx O(2^{n/2})$
- Brute-force collision search is $O(2^{n/2})$, <u>not</u> $O(2^n)$
  - For SHA-1, this means $O(2^{80})$ vs. $O(2^{160})$

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 (cannot invert h without inverting 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-1)-bit long, else 1g(x)
    - 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?), thus random y is invertible with prob. 1/2

Weak Collision Resistance

- Given a 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?)

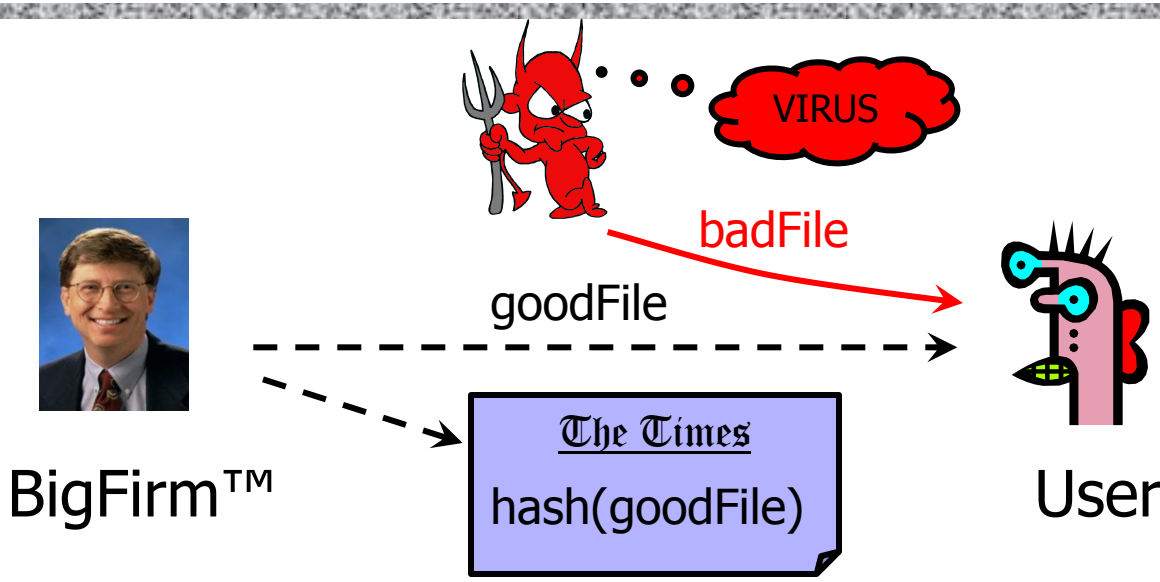
Hashing vs. Encryption

- Hashing is one-way. There is no "uh-hashing"!
  - A ciphertext can be decrypted with a decryption key… hashes have no equivalent of "decryption"
- Hash(x) looks "random", but can be compared for equality with Hash(x')
  - Hash the same input twice → same hash value
  - Encrypt the same input twice → different ciphertexts
- Cryptographic hashes are also known as "cryptographic checksums" or "message digests"

Application: Password Hashing

- Instead of user password, store hash(password)
- When user enters a password, compute its hash and compare with the entry in the password file
  - System does not store actual passwords!
  - Cannot go from hash to password!
- Why is hashing better than encryption here?
- Does hashing protect weak, easily guessable passwords?

Application: Software 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 the NY Times

The goal is integrity, not secrecy

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

Which Property Is Needed?

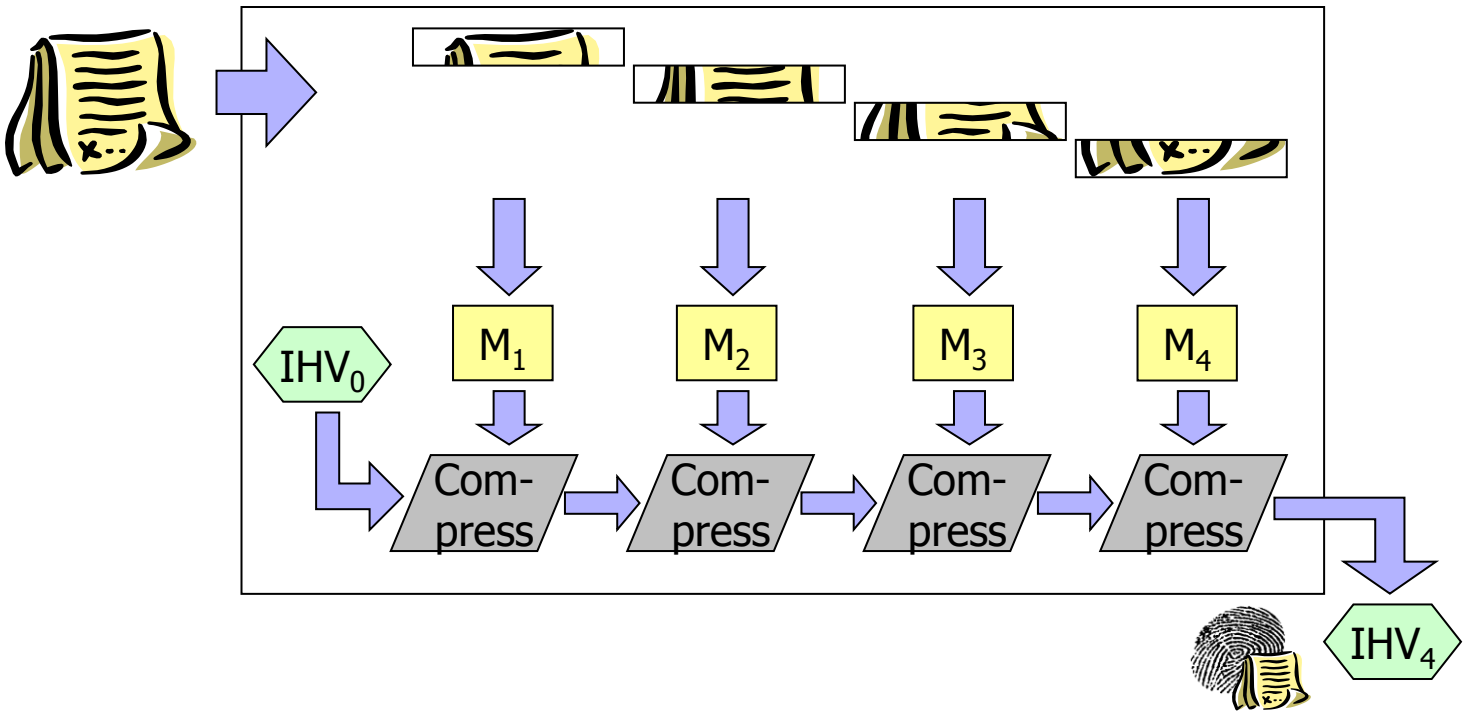
- Passwords stored as hash(password)
  - One-wayness: hard to recover entire password
  - Passwords are not random and thus guessable
- Integrity of software distribution
  - Weak collision resistance?
  - But software images are not random… maybe need full collision resistance
- Auctions: to bid B, send H(B), later reveal B
  - One-wayness… but does not protect B from guessing
  - Collision resistance: bidder should not be able to find two bids B and B′ such that H(B)=H(B′)

Common Hash Functions

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

Overview of MD5

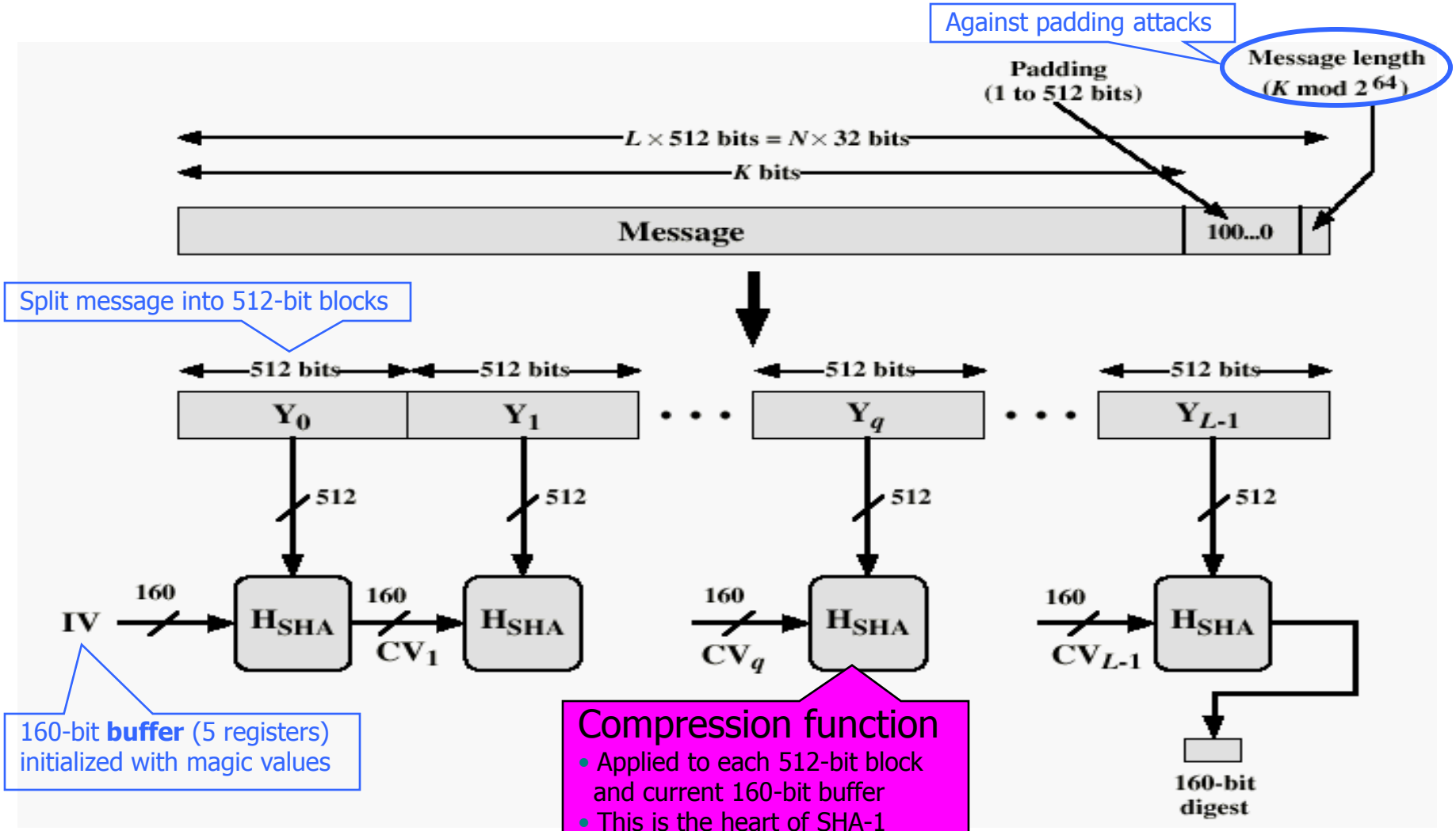
- Designed in 1991 by Ron Rivest
- Iterative design using compression function



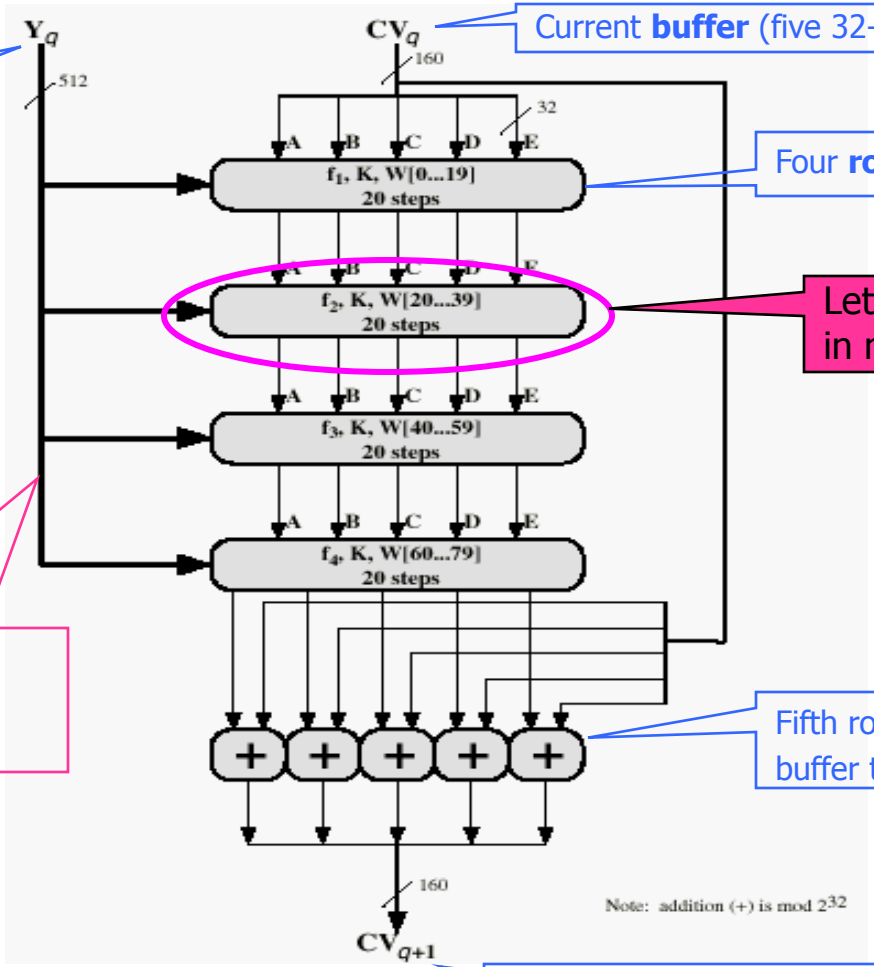
History of MD5 Collisions

- 2004: first collision attack
  - The only difference between colliding messages is 128 random-looking bytes
- 2007: chosen-prefix collisions
  - For any prefix, can find colliding messages that have this prefix and differ up to 716 random-looking bytes
- 2008: rogue SSL certificates
  - Talk about this in more detail when discussing PKI
- 2012: MD5 collisions used in cyberwarfare
  - Flame malware uses an MD5 prefix collision to fake a Microsoft digital code signature

Basic Structure of SHA-1



SHA-1 Compression Function



Current **message block**

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

Four **rounds**, 20 steps in each

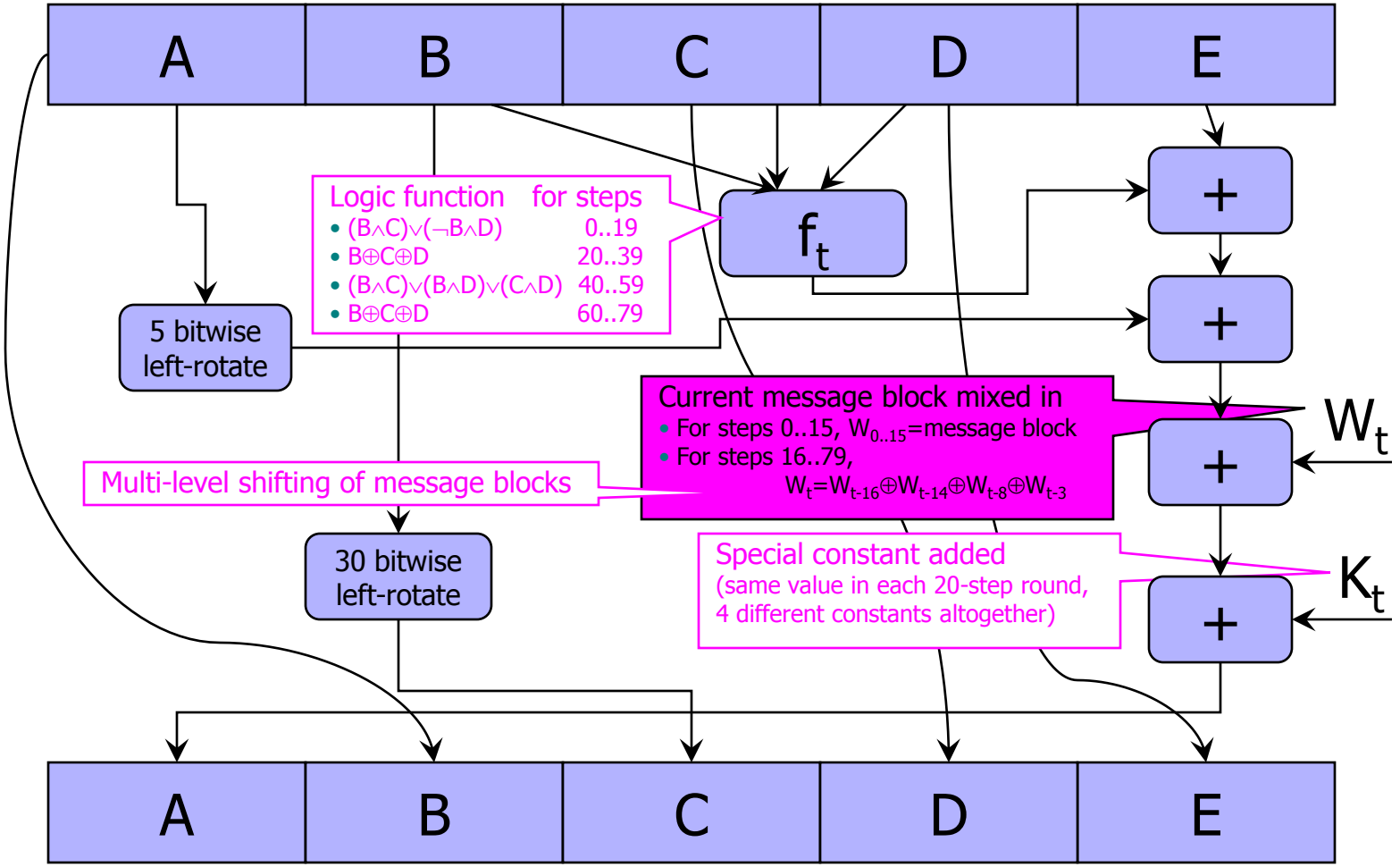
Let's look at each step in more detail...

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)



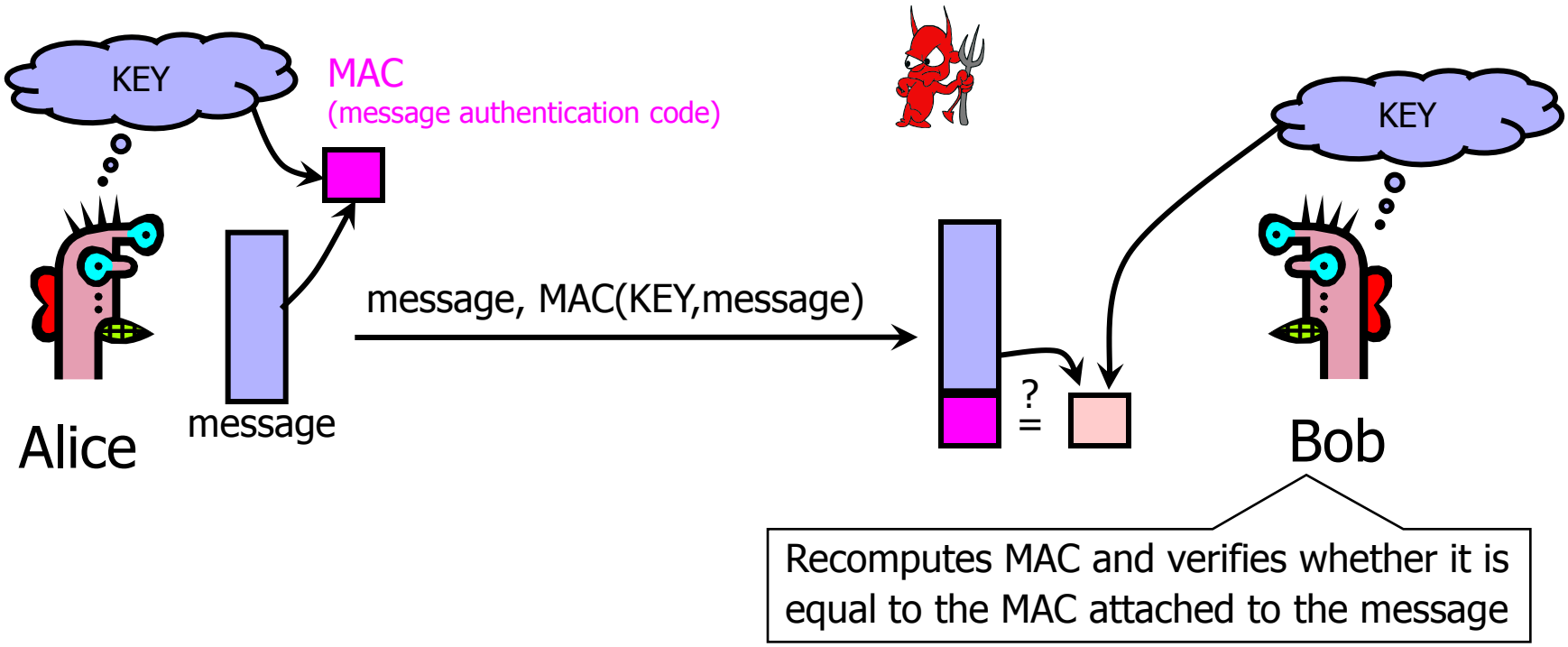
How Strong Is SHA-1?

- 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 weaknesses discovered in 2005
  - Collisions can be found in 2^{63} ops

NIST Competition

- A public competition to develop a new cryptographic hash algorithm
  - Organized by NIST (read: NSA)
- 64 entries into the competition (Oct 2008)
- 5 finalists in 3rd round (Dec 2010)
- Winner: Keccak (Oct 2012)
  - Will be standardized as SHA-3

Integrity and Authentication



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

HMAC

- Construct MAC from a cryptographic hash function
  - Invented by Bellare, Canetti, and Krawczyk (1996)
  - Used in SSL/TLS, mandatory for IPsec
- Why not encryption?
  - Hashing is faster than encryption
  - Library code for hash functions widely available
  - Can easily replace one hash function with another
  - There used to be US export restrictions on encryption

Structure of HMAC