CS 4173/5173
COMPUTER SECURITY

The Length of Hash

OUTLINE LAST TIME

Hash functions:

1) Performance: Easy to compute H(*m*)

2) One-way property: Given H(*m*) but not *m*, it's computationally infeasible to find *m*

3) Weak collision resistance: Given H(*m*), it's computationally infeasible to find *m'* such that H(*m'*) = H(*m*).

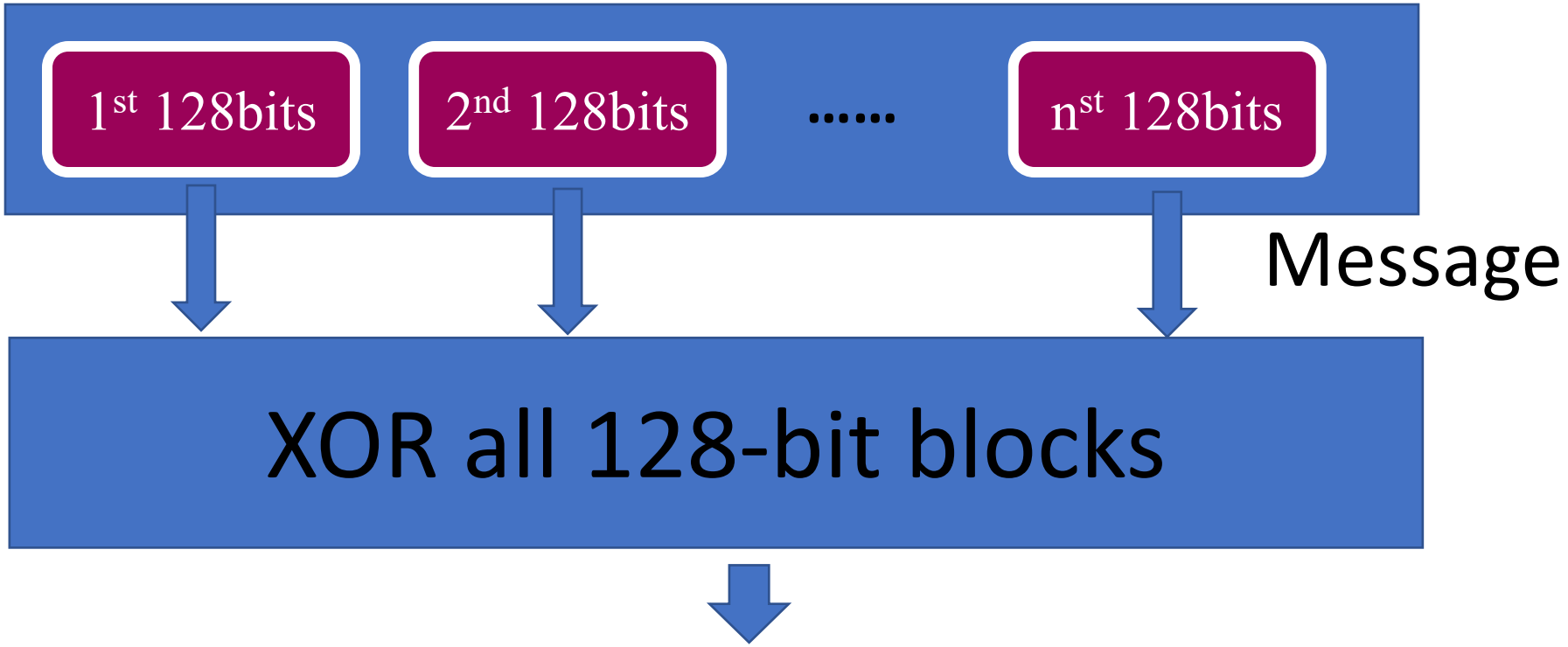
4) Strong collision resistance: computationally infeasible to find m_1, m_2 such that H(m_1) = H(m_2)

EXERCISE I (SECRET DEAL)

- A and B have a financial agreement on something.
  - They want to keep the agreement confidential, no third-party should know the content of the agreement.
  - But they also want to have a third-party to certify the agreement, just in case…

EXERCISE II (A NEW HASH?)

- Bob designed a very fast hash function, which is to XOR every 128 bits in a message.



What is the flaw?

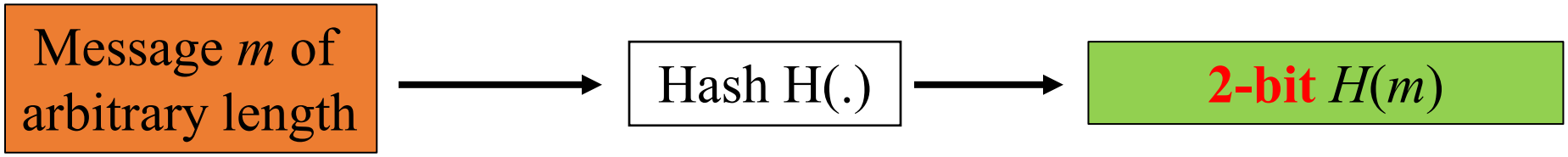
EXERCISE III (FILE MANAGEMENT)

- A file system contains millions of data files.
- Design a scheme to detect if a given file exists in the system.
  - i.e., find out if there already exists a file that has the same content with the given file.
- Specify your design
  - Cost in terms of storage and computation
  - Efficiency in file finding and comparison

ESSENCE IN HASH-BASED SECURE DESIGN

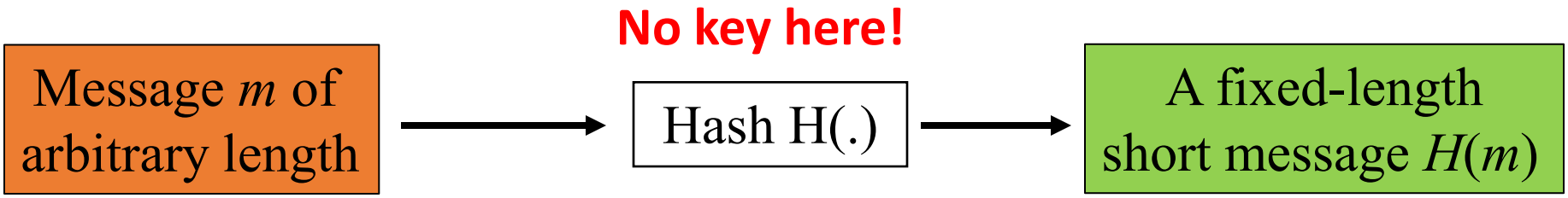
- Make sure the input has a very large value space so it is computationally difficult to do exhaust search.
  - Use salt
  - Use random padding if necessary
  - ……

HASH FUNCTION WITH SHORT OUTPUT



Does this hash function have the strong collision resistance property? Why?

LENGTH OF HASH



- Question
  - Why do we have 128 bits,160 bits, 256 bits in the output of a hash function?
  - If it is too long
    - Unnecessary overhead
  - If it is too short
    - Loss of strong collision free property

BIRTHDAY PARADOX: SIMPLE QUESTION

- Question:
  - What is the probability that two persons have same birthday?
  - (Equivalent question: what is the probability the hashes of two different inputs are the same?)
    - Assume 365 days a year, and all birthdays are equally likely
  - Probability is 1 - 365 * (365-1)/(365*365).

BIRTHDAY PARADOX: GENERAL QUESTION

- Question:
  - What is the probability that at least two persons among *k* persons have the same birthday?
    - Assume 365 days a year, and all birthdays are equally likely
  - Probability is

$1 - 365*(365-1)*(365-2)*\ldots*(365-k+1)/365^k$

$= 1 - 365!/(365-k)!/365^k$

BIRTHDAY PARADOX

- Question:
  - What is the probability that at least two persons among *k* persons have the same birthday?
    - Assume 365 days a year, and all birthdays are equally likely
    - Probability is $1 - 365!/(365-k)!/365^k$
    - The larger *k*, the larger the probability.

- New Question: Find the values of *k* such that the probability is greater than 0.5.
  - Why 0.5? Just an example to show when the value of k could make the collision probability non-negligible
    - i.e., collision is really likely to happen.
  - $1 - 365!/(365-k)!/365^k >= 0.5$ ➔ ***k* >= 23**.

BIRTHDAY PARADOX: GENERALIZATION

- Generalization of birthday paradox
  - Given
    - random variable with uniform distribution between 1 and *n*
      - *n*=365 in the birthday paradox
    - a selection of *k* random variables
      - *k* persons in birthday paradox
  - What is the least value of *k* such that
    - There will be at least two variables with the same value with probability P(*n*,*k*) > 0.5 ?

BIRTHDAY PARADOX: GENERALIZATION

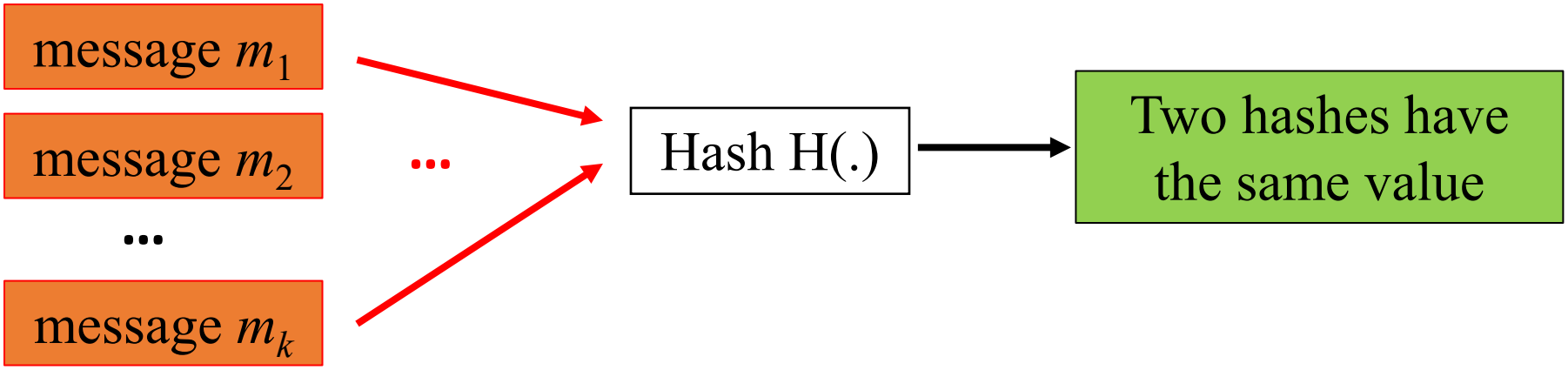
- Generalization of birthday paradox
  - $P(n,k) = 1-\{n!/(n-k)!n^k\} \approx 1 - e^{-k*(k-1)/2n}$
  - For large n and k, to have $P(n,k) > 0.5$ with the smallest k, we have

$$k = \sqrt{2(\ln 2)n} = 1.18\sqrt{n} \approx \sqrt{n}$$

  - Example
    - Find the values of k such that the probability that two persons have the same birthday is greater than 0.5. (n=365)
      - $1.18*(365)^{1/2} = 22.54$

BIRTHDAY PARADOX (CONT'D)

- Implication for hash function H of bit length *m*
  - The hash value of an arbitrary input message is randomly distributed between 1 and 2^m, i.e., ($n=2^m$)



  - What is the least value of *k* such that
    - If we hash *k* messages, the probability that at least two of them have the same hash (i.e., a collision) is larger than 0.5.

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

Design objective: make sure *k* is very large so an attacker sees the computational difficulty

BIRTHDAY PARADOX (CONT'D)

- Typical hash length: 128 (MD5), 160 bits (SHA-1), 256 (SHA-256), 512 (SHA-512)

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

  - 2^{64} =18446744073709551616
  - 2^{80} =1208925819614629174706176
    - It takes 38,334,786 years for a computer with 1 billion hashes per second.

SUMMARY

- Design of Hash should balance the tradeoff between overhead and security
  - If it is too long
    - Unnecessary overhead
  - If it is too short
    - Loss of strong collision free property
    - Birthday paradox

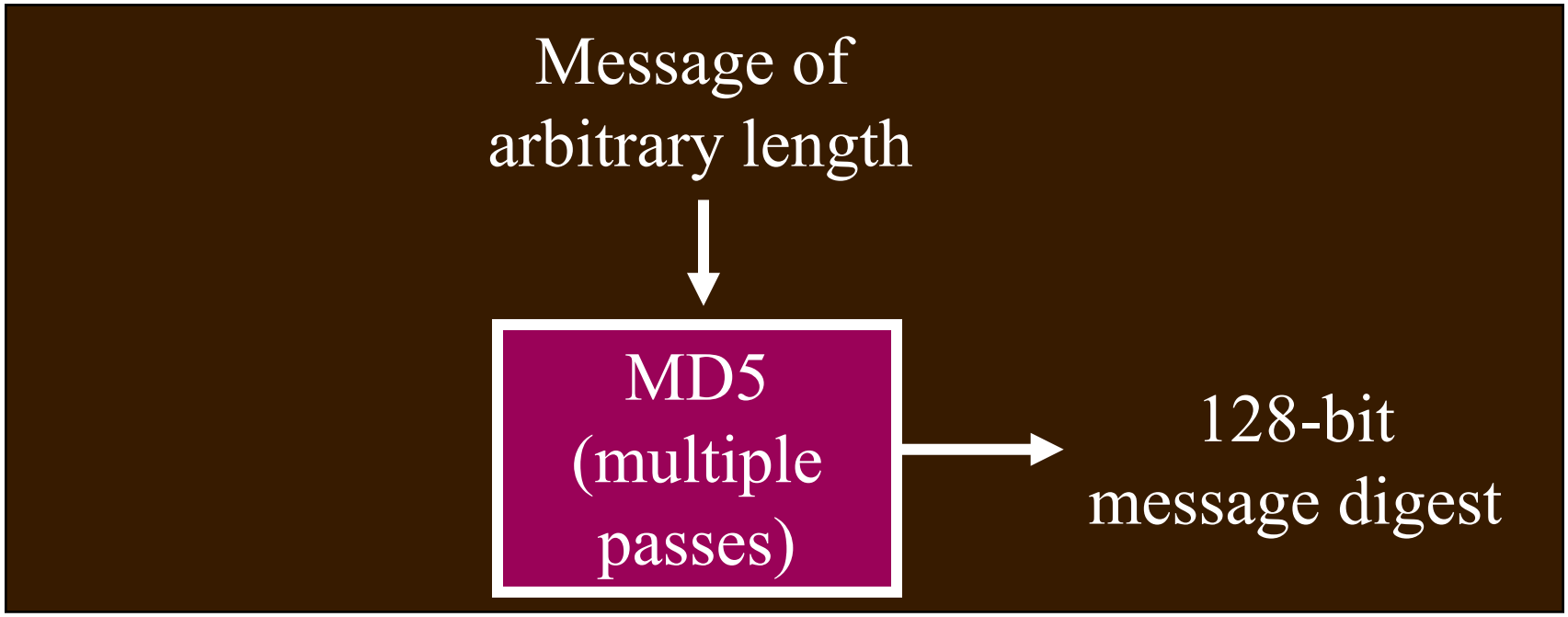
- MD5: 128 bits – not secure any more
- SHA1: 160 bits – not (very) secure any more
- SHA256: 256 bits
- SHA384, SHA512, …

CS 4173/5173
COMPUTER SECURITY

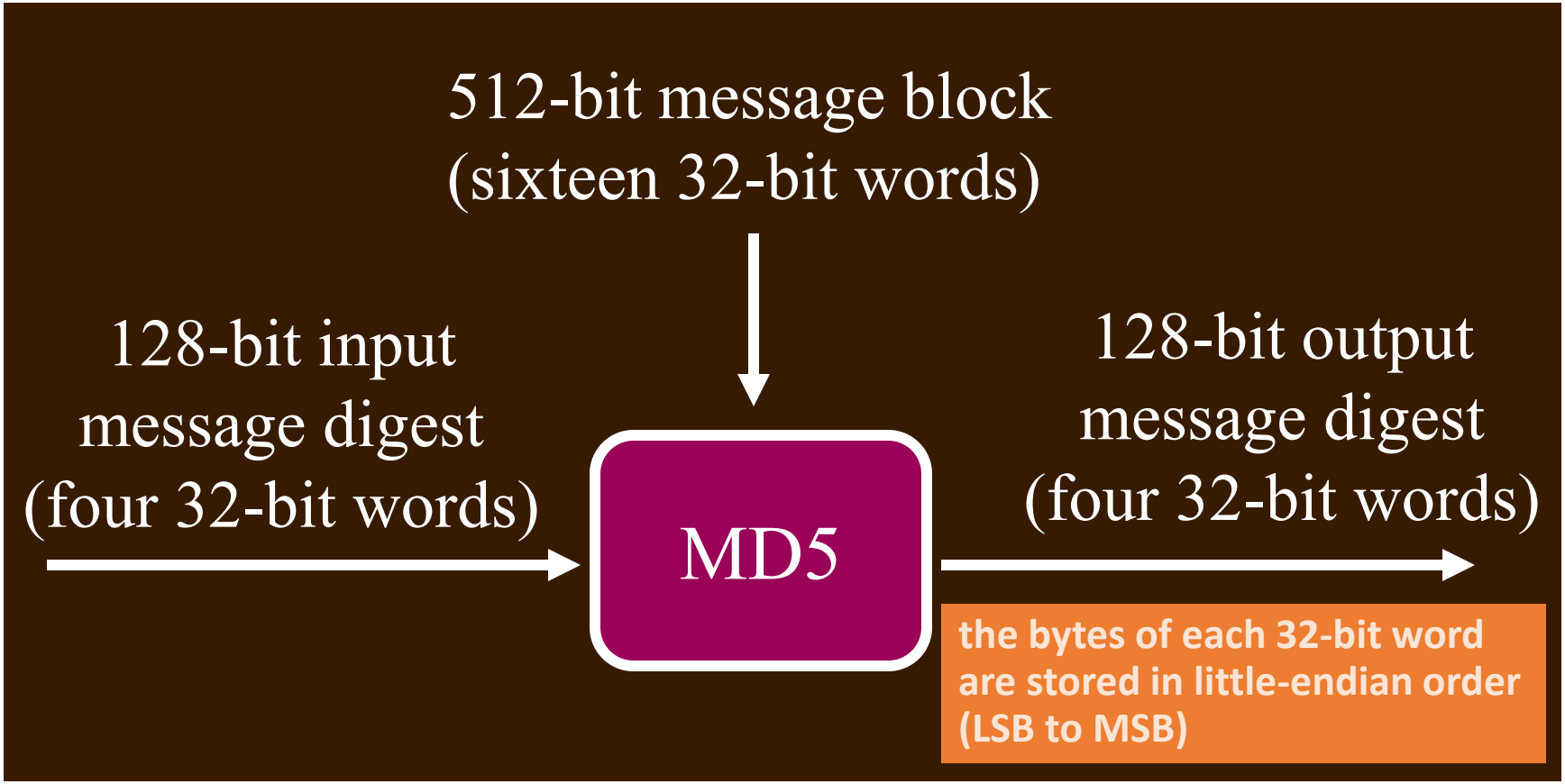
The MD5 Hash Function

MD5: MESSAGE DIGEST VERSION 5

- MD5 at a glance

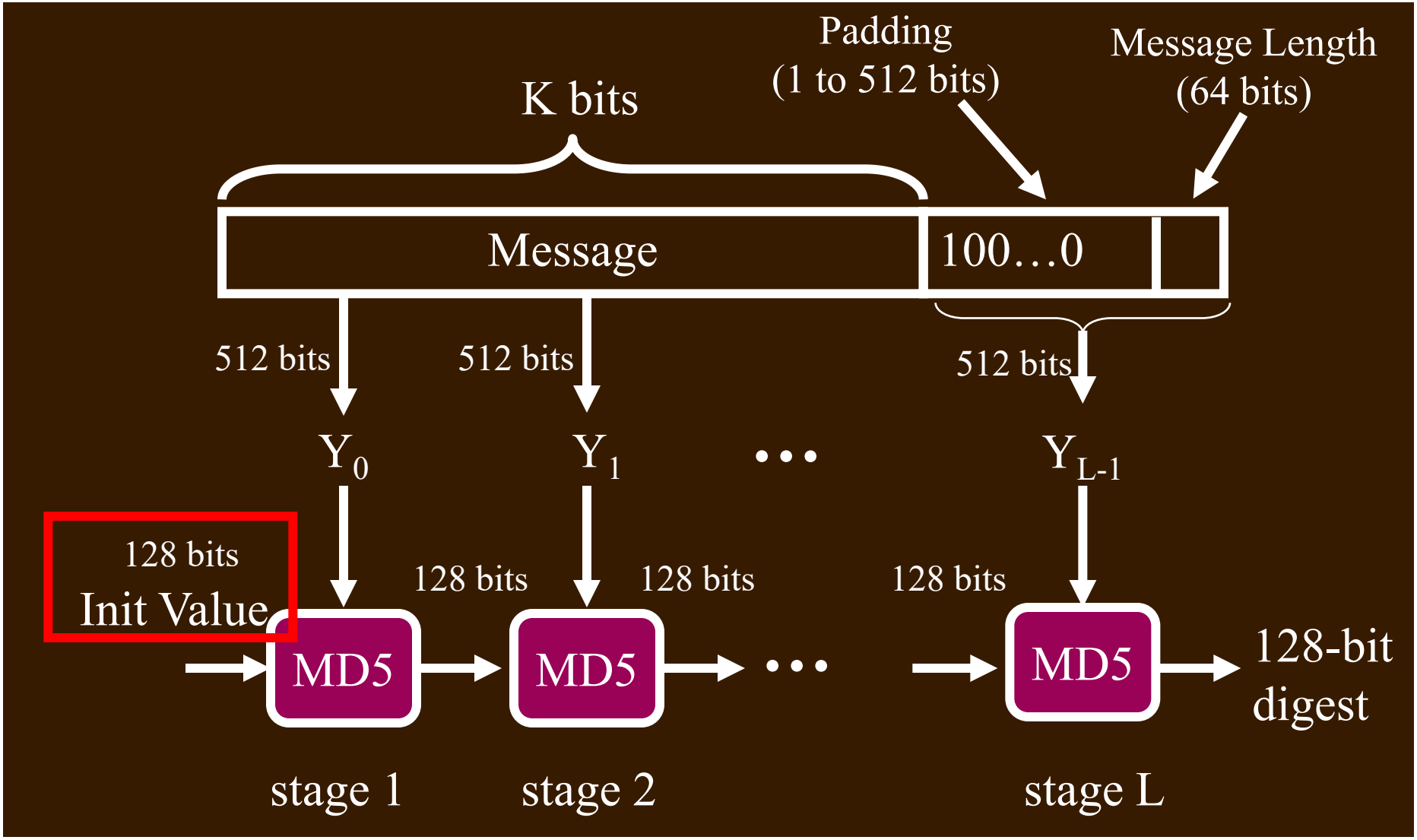


PROCESSING OF A SINGLE BLOCK



1 word = 4 bytes = 32 bits

MD5: A HIGH-LEVEL VIEW



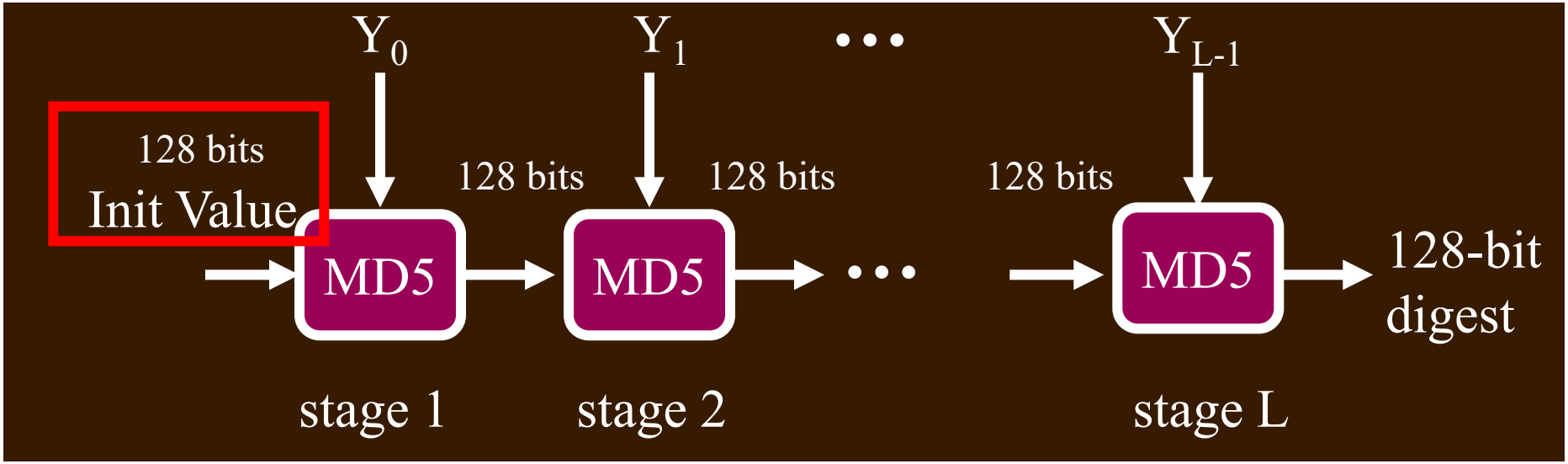
PADDING

- To original message M, add padding bits "10…0"
  - enough 0's so that resulting total length is 64 bits less than a multiple of 512 bits

- Append L (the original length of M), represented in 64 bits, to the padded message

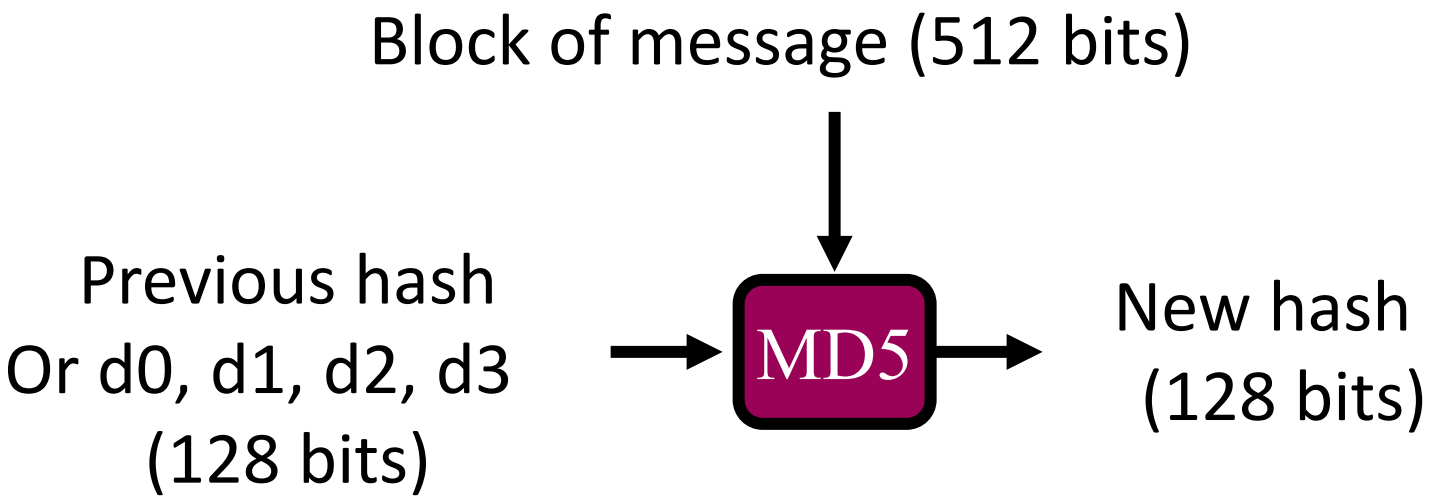
INITIAL VALUES

- The four 32-bit words of the output (the *digest*) are referred to as **d0, d1, d2, d3**
  - Initial values (in little-endian order)
  - **d0** = 0x67452301, **d1** = 0xEFCDAB89
  - **d2** = 0x98BADCFE, **d3** = 0x10325476



INSIDE AN MD5 BLOCK

- Bits operations: fully specified:
  - AND, OR, XOR, bit shifting, …



(IN)SECURITY OF MD5

- A few methods can find collisions in a few hours
  - Birthday attack: need hash 2^{64} messages to find a collision with probability 0.5.
    - Recall from birthday paradox, we need to compute $2^{m/2}$ hashes to find two with the same value with probability 0.5.
    - For MD5, m = 128 bits, so brute-force attack needs $2^{128/2} = 2^{64}$.
  - A few collisions were published in 2004
  - Can find many collisions for 1024-bit messages.

CS 4173/5173
COMPUTER SECURITY

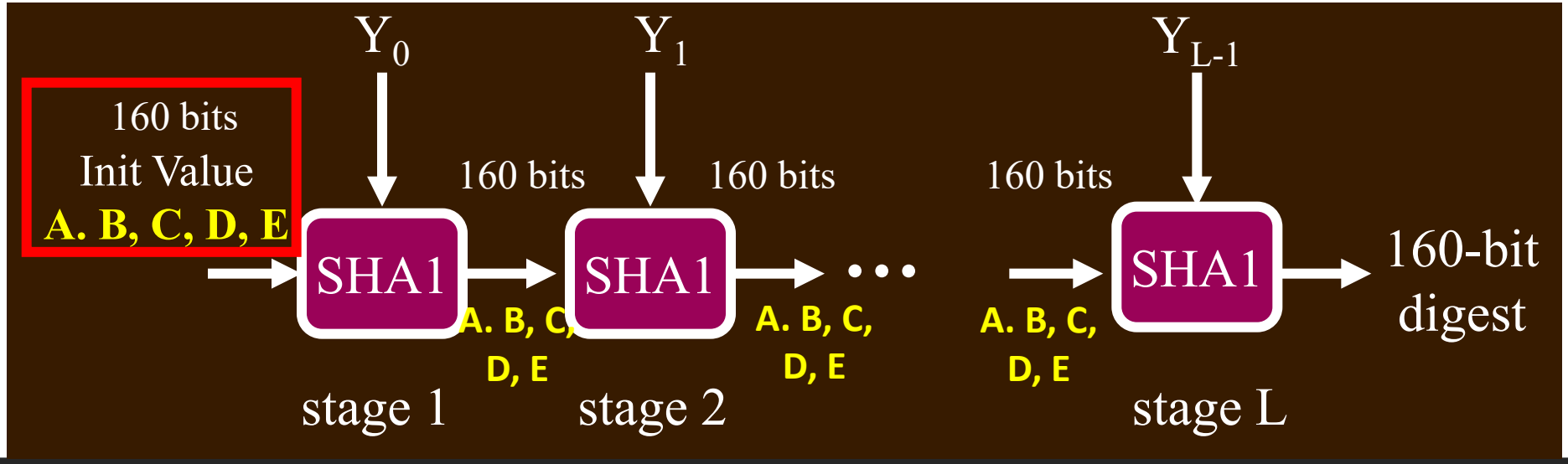
The SHA-1 Hash Function

SECURE HASH ALGORITHM (SHA)

- Developed by NIST, specified in the Secure Hash Standard, 1993
- SHA is specified as the hash algorithm in the Digital Signature Standard (DSS)
- SHA-1: revised (1995) version of SHA

SHA-1 PARAMETERS

- Input message must be < 2^{64} bits
- Input message is processed in 512-bit blocks, with the same padding as MD5
- Message digest output is 160 bits long
  - Referred to as five 32-bit words **A, B, C, D, E**
  - IV: **A** = 0x67452301, **B** = 0xEFCDAB89, **C** = 0x98BADCFE, **D** = 0x10325476, **E** = 0xC3D2E1F0 (big-endian order)

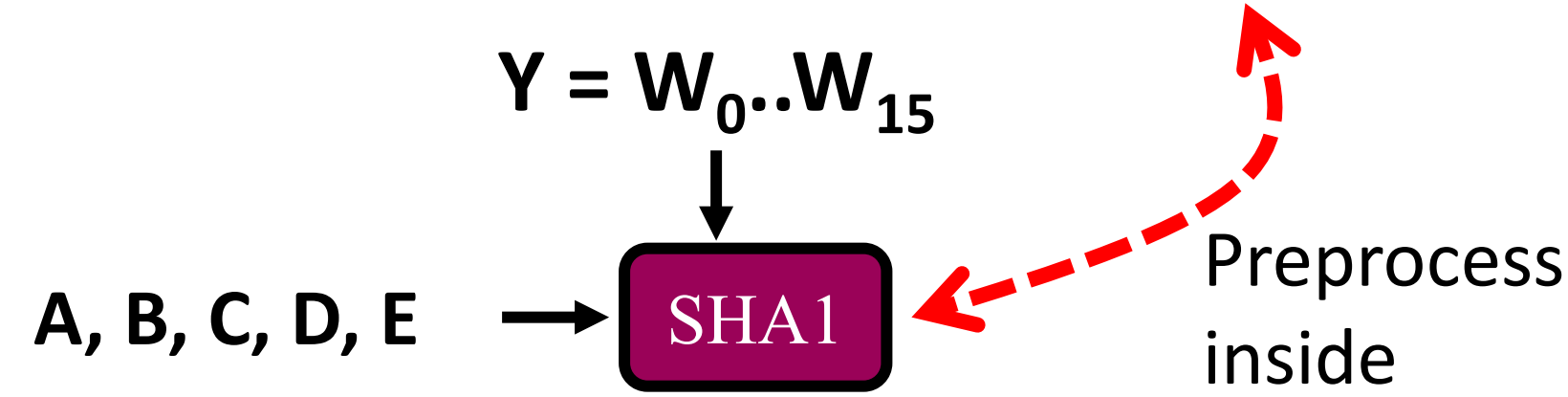


PREPROCESSING OF A BLOCK

- Let 512-bit block be denoted as sixteen 32-bit words $\mathbf{W_0..W_{15}}$
- Preprocess $\mathbf{W_0..W_{15}}$ to derive an additional sixty-four 32-bit words $\mathbf{W_{16}..W_{79}}$, as follows:
  for $16 \leq t \leq 79$

$$W_t = (W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3}) << 1$$

Note: << is "circular shift"



SHA1 BLOCK PROCESSING

- Consists of 80 steps! (vs. 64 for MD5)
- Inputs for each step $0 \leq t \leq 79$:
  - $\mathbf{W}_t$
  - K_t – a constant
  - **A**,**B**,**C**,**D**,**E**: current values to this point
- Outputs for each step:
  - **A**,**B**,**C**,**D**,**E** : new values

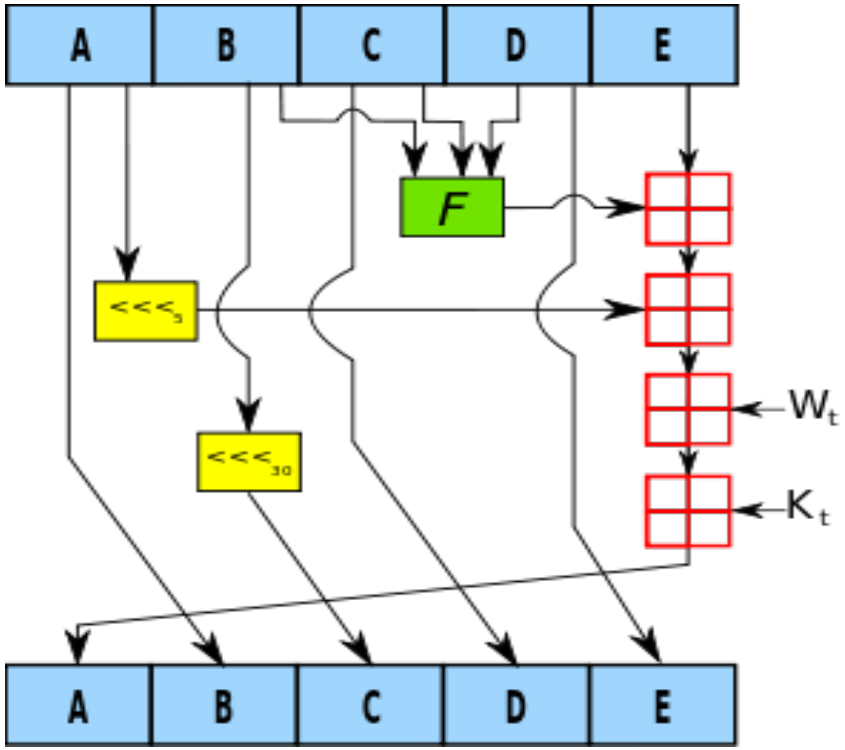
PROCESSING PER STEP

- Everything to right of "=" is input value to this step (i.e., old values)

```
for t = 0 to 79
    A = E + (A << 5) + W_t + K_t + f(t,B,C,D)
    B = A
    C = B << 30
    D = C
    E = D
end
```

Note: << is "circular shift"

FIGURE FOR ONE STEP



from wiki
http://en.wikipedia.org/wiki/SHA-1

CONSTANTS K_T

- Only 4 values (represented in 32 bits), derived from $2^{30} * i^{1/2}$, for i = 2, 3, 5, 10
  - for $0 \leq t \leq 19$: K_t = 0x5A827999
  - for $20 \leq t \leq 39$: K_t = 0x6ED9EBA1
  - for $40 \leq t \leq 59$: K_t = 0x8F1BBCDC
  - for $60 \leq t \leq 79$: K_t = 0xCA62C1D6

FUNCTION F(*T*,B,C,D)

- 3 different functions are used in SHA-1 processing

Round	Function f(t,B,C,D)
$0 \le t \le 19$	$(B \wedge C) \vee (\sim B \wedge D)$
$20 \le t \le 39$	$B \oplus C \oplus D$
$40 \le t \le 59$	$(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
$60 \le t \le 79$	$B \oplus C \oplus D$

- $\wedge$ - AND
- $\vee$ - OR

COMPARISON: SHA-1 VS. MD5

- SHA-1 is a stronger algorithm
  - brute-force attacks require on the order of 2^{80} operations vs. 2^{64} for MD5
- SHA-1 is about twice as expensive to compute
- Both MD-5 and SHA-1 are much faster to compute than DES

SECURITY OF SHA-1

- SHA-1
  - Still more secure than MD5
    - SHA-1 has 160 bits, so $2^{160/2} = 2^{80}$ for brute-force attacks
    - MD5 has 128 bits, so $2^{128/2} = 2^{64}$ for brute-force attacks, and much less for other smart attacks.

FINDING SHA-1 COLLISION

- https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html

- In 2013, Marc Stevens published a paper that outlined a theoretical approach to create a SHA-1 collision.
- In 2017, first collision announced by Google:
  - Nine quintillion (9,223,372,036,854,775,808/2^{63}) SHA1 computations in total
  - 6,500 years of CPU computation to complete the attack first phase
  - 110 years of GPU computation to complete the second phase

MODERN HASH FUNCTIONS

- MD6, SHA-256, SHA-384, SHA-512…
- https://shattered.io/

USING HASH FUNCTIONS IN LINUX

- MD5
  - Command: md5sum [filename]
    - Compute the MD5 hash of the content of a file
  - Command: echo “[content]” | md5sum
    - Compute the MD5 hash of [content]
- SHA1
  - Command: sha1sum [filename]
- SHA256
  - Command: sha256sum [filename]

SUMMARY OF HASH

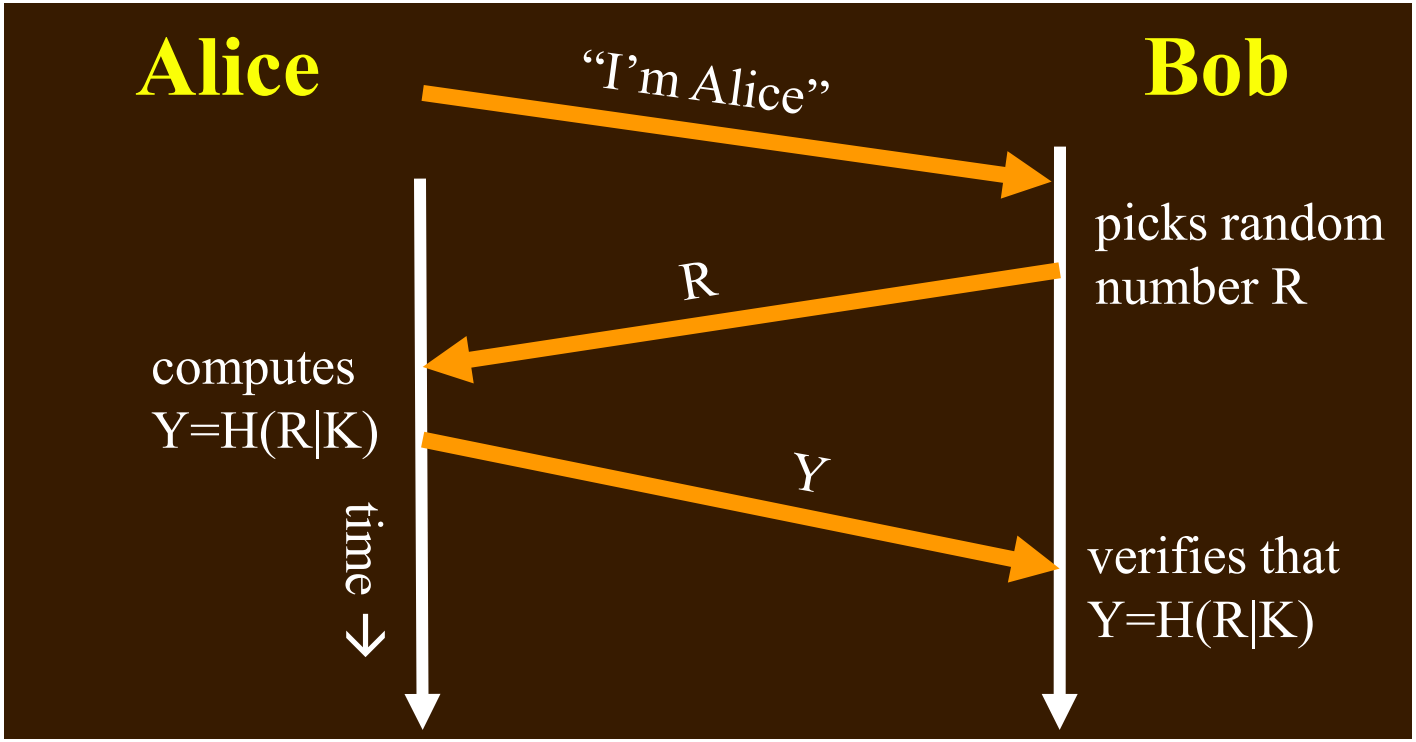
- Four important properties
- Applications
  - Detecting file change
  - Storing password
  - Message authentication, HMAC
  - …
- Hash Functions:
  - MD5: 128 bits – not secure at all
  - SHA1: 160 bits – not very secure
  - SHA256, SHA384, SHA512 – secure for now
  - …

APPLICATION

- Alice wants to prove to Bob that she is indeed Alice
  - Alice and Bob shares a common secret key
  - But the communication between them is on a public channel.
  - What Alice should do to prove to Bob that she is indeed Alice without actually disclosing the key itself?

APPLICATION: USER AUTHENTICATION

- Alice wants to authenticate herself to Bob
  - assuming they already share a secret key K
- Protocol:



What should be the range of random number R?

CS 4173/5173
COMPUTER SECURITY

High Level Introduction to Public Key Cryptography

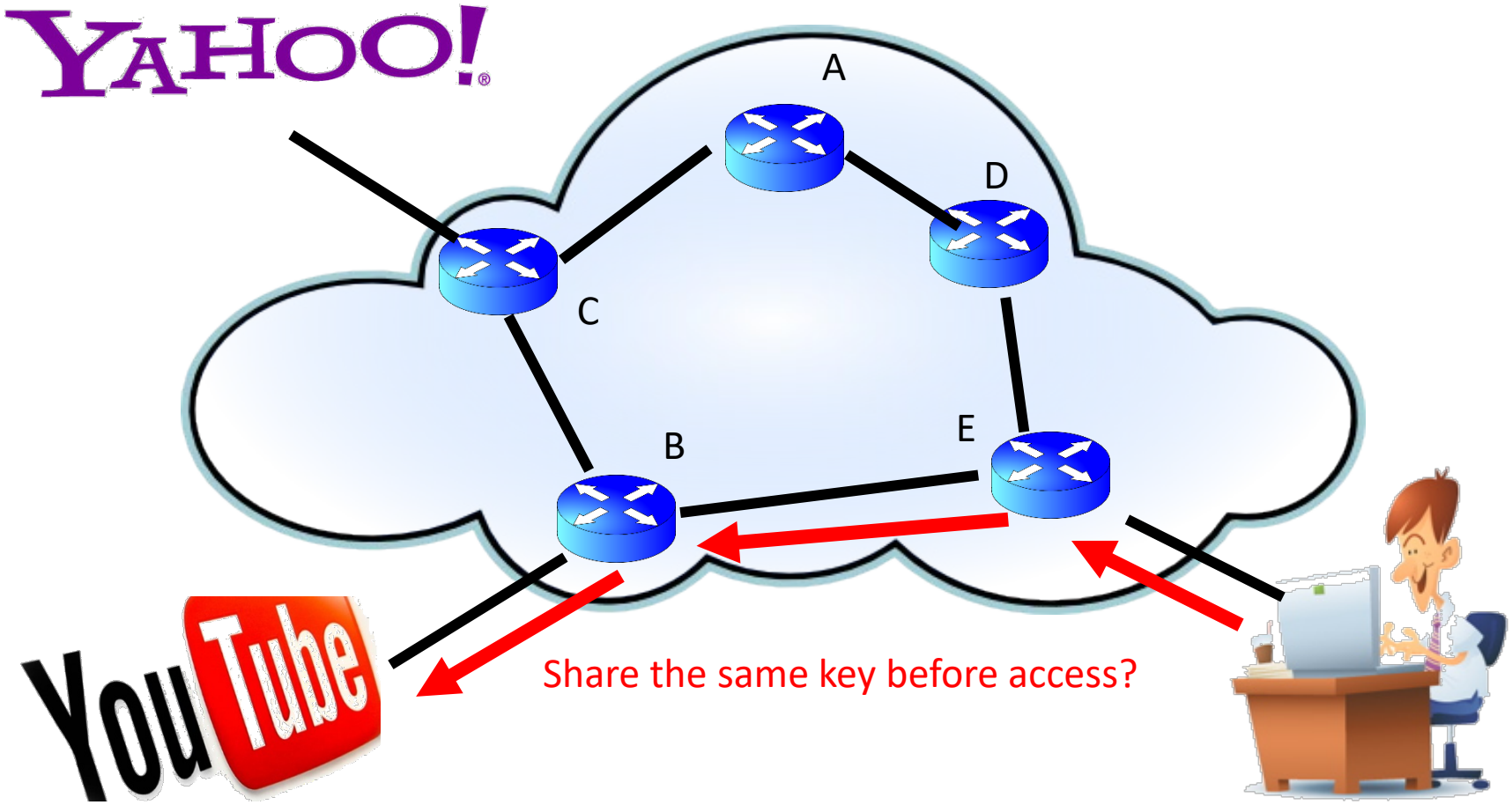
COMPANY A'S PROBLEM I

- Company A is a big web service company with over 10,000 employees.
- The president Bob wants to make sure that all employees can verify the authenticity of the announcement emails that he sends.
- Q: How to ensure authenticity of these emails.

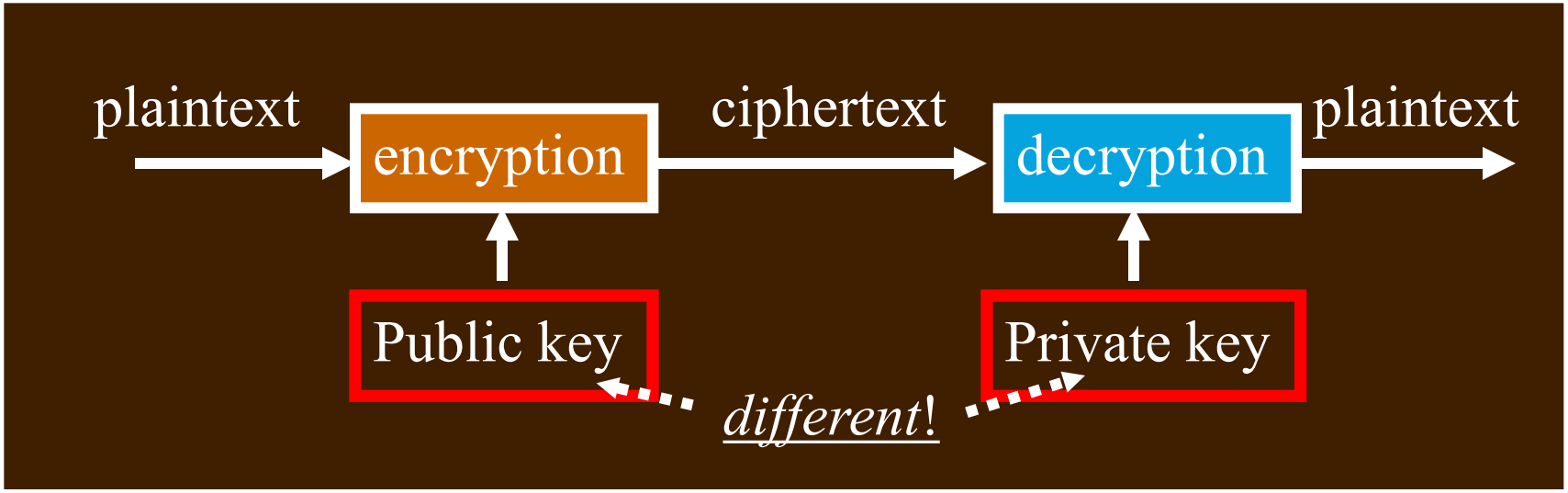
COMPANY A'S PROBLEM II

- Company A is accepting vulnerability report of their web system from the public.
- They need a design that someone can successfully send the report of a potential vulnerability via email to them.
- Q: How to ensure the confidentiality of reports?

HOW TO SECURELY SURF INTERNET



PUBLIC KEY CRYPTOGRAPHY

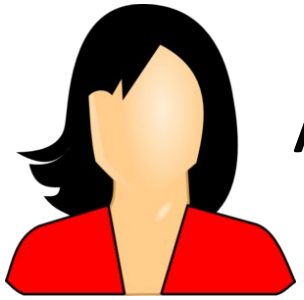


- Invented and published in 1975
- A *public / private key pair* is used
- Also known as *asymmetric* cryptography
- Much slower to compute than secret key cryptography

PUBLIC/PRIVATE KEY

- Public key – encrypt
- Private key – decrypt

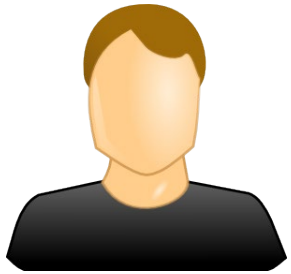
- How does the secret communication look like?



Alice

wants to send a message

Bob



PUBLIC/PRIVATE KEY

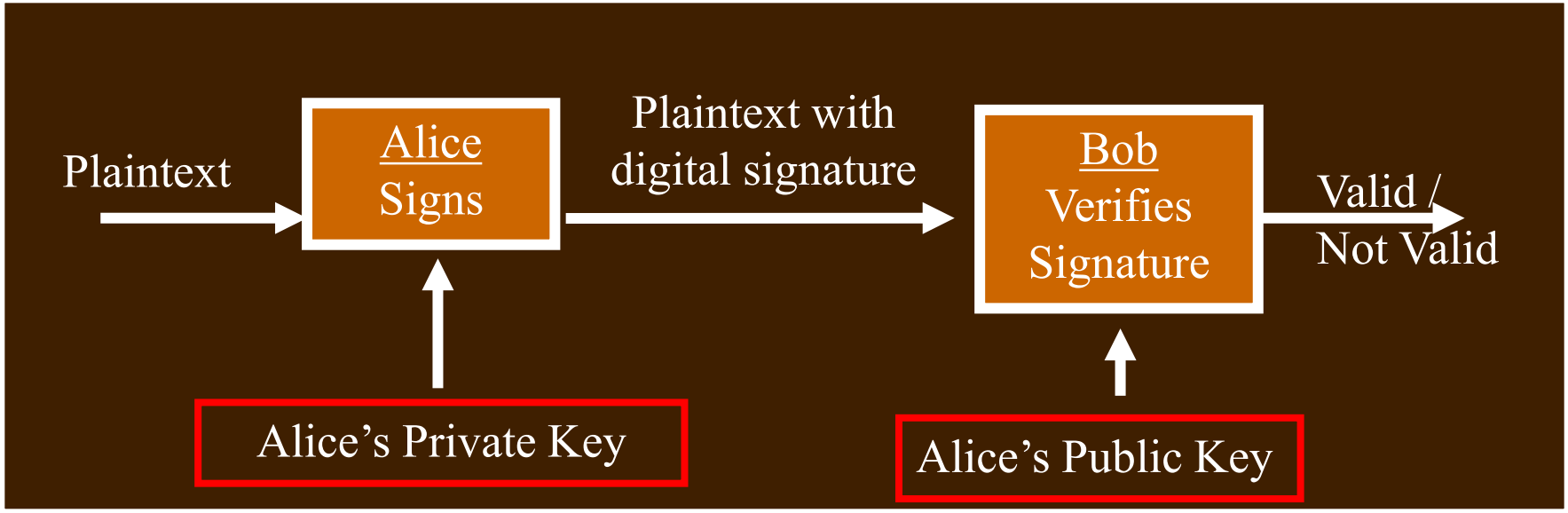
- Alice has her own public and private key pair
- Bob also has his own public and private key pair

- Public key
  - Can be released to the public

- Private Key
  - Must be kept secret.

AUTHENTICATION IN PUBLIC KEY CRYPTO

- Message integrity with digital signatures
- Alice computes hash, signs with her private key (no one else can do this without her key)
- Bob verifies hash on receipt using Alice's public key using the verification equation



AUTHENTICATION (CONT'D)

- Authentication in public key crypto:
  - Hash function to hash the message into a digest
  - The action of sign the digest with (private key)
  - The action of verify the digest with (public key)

PUBLIC-KEY REQUIREMENTS

- It must be computationally
  - easy to generate a public / private key pair
  - hard to determine the private key, given the public key
- It must be computationally
  - easy to encrypt using the public key
  - easy to decrypt using the private key
  - hard to recover the plaintext message from just the ciphertext and the public key

PUBLIC KEY ALGORITHMS

- Public key algorithms covered in this class, and their applications

System	Encryption / Decryption?	Digital Signatures?	Key Exchange?
RSA	Yes	Yes	Yes
Diffie-Hellman			Yes
DSA		Yes	

SOLVING COMPANY A'S PROBLEM I

- Company A is a big web service company with over 10,000 employees.
- The president Bob wants to make sure that all employees can verify the authenticity of the announcement emails that he sends.
- Answer:
  - Everyone knows Bob's public key.
  - Bob signs the email using his private key.
  - Everyone can verify the signed email using Bob's public key

SOLVING COMPANY A'S PROBLEM II

- Company A is accepting vulnerability report of their web system from the public.
- They need a design that someone can successfully send the report of a potential vulnerability via email to them.
- Answer:
  - Company A generates a key pair, then releases the public key to the public for vulnerability report.
  - Everyone uses the public key to encrypt the report.

PUBLIC KEY VS. SYMMETRIC KEY

Symmetric key	Public key
Two parties MUST trust each other	Two parties DO NOT need to trust each other
Both share same key	Two separate keys: a public and a private key
Typically faster	Typically slower
Examples: DES, RC5, AES, …	Examples: RSA, DSA, ECC…

COMPANY A'S PROBLEM III

- Company A provides an on-line chat service for vulnerability report.
  - Requirement 1: confidentiality.
  - Requirement 2: efficiency because there will be a number of message exchanges.

- Q: How to satisfy both requirements?

DIGITAL ENVELOPE: SYMMETRIC+ASYMMETRIC

1. Generate a secret key (called a session key) at random.
2. Encrypt the message using the session key and symmetric algorithm.
3. Encrypt the session key with the recipient's public key. This becomes the "digital envelope".
4. Send the encrypted message and the digital envelope to the recipient.

DIGITAL ENVELOPE (CONT'D)

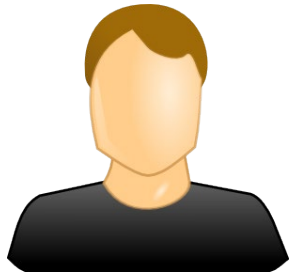
Alice (finds a vulnerability)

Bob (company representative)

Knows the public key: Kp

Generate a session key Ks

Encrypt message M using Ks
Encrypted Ks using Kp



Public key: Kp
Private key: Ki

1. Use Ki to decrypt Ks
2. Use Ks to decrypt M

Send a response using Ks

All remaining communication is based on Ks