WK01: Crypto Building Blocks

Securing Wireless Networks, COMP4337/9337

UNSW AUSTRALIA

Never Stand Still

Professor Sanjay K. Jha

School of Computer Science and Engineering, UNSW

Today's Agenda

- Learn Crypto building blocks for
- Confidentiality and Authentication :Secret Keys
- Integrity: Message Digests (Hash)
- Integrity and Authentication: HMAC, MAC
- Non-Repudiation: Digital Signature
- Block Ciphers

Security Fundamentals (revision)

confidentiality:

only sender, intended receiver should "understand" message contents

- sender encrypts message
- receiver decrypts message

authentication:

sender, receiver want to confirm identity of each other

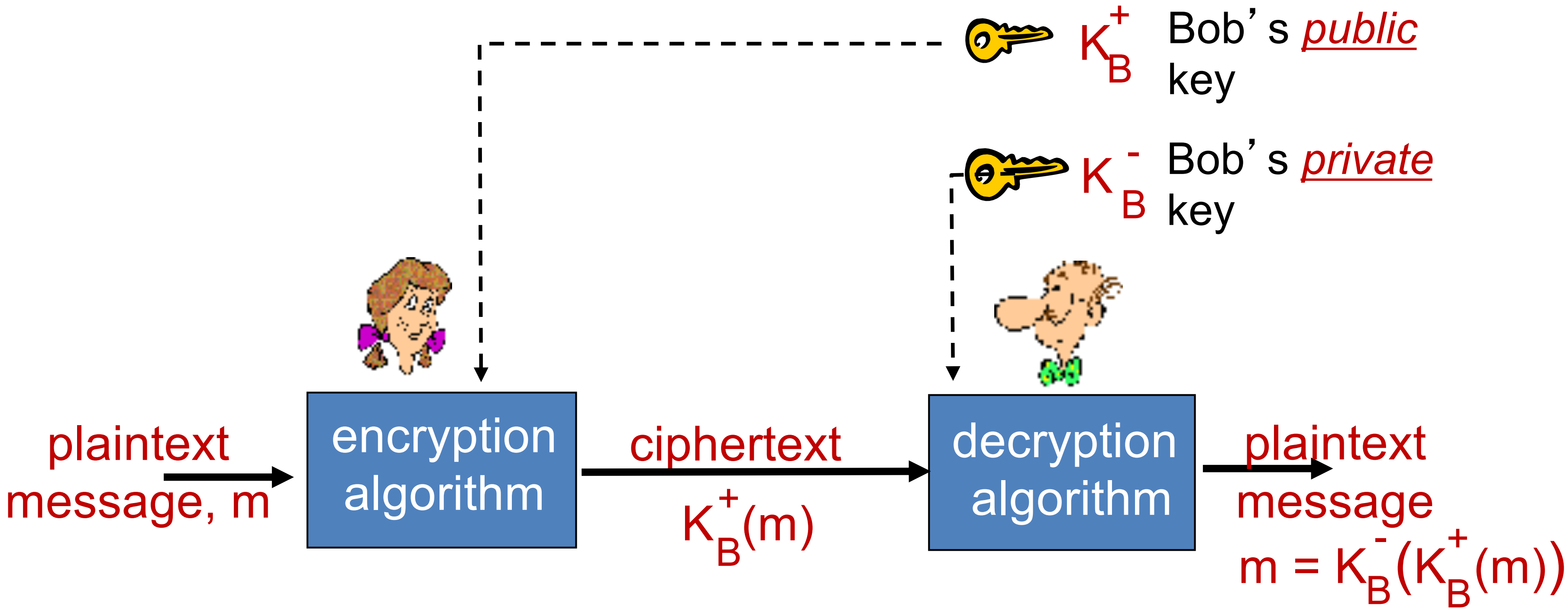
message integrity:

ensure message not altered (in transit, or afterwards) without detection

Non-repudiation:

associating actions or changes to a unique individual

Public Key Cryptography



Public key encryption algorithms

- Need K_B^+ (.) and K_B^- (.) such that

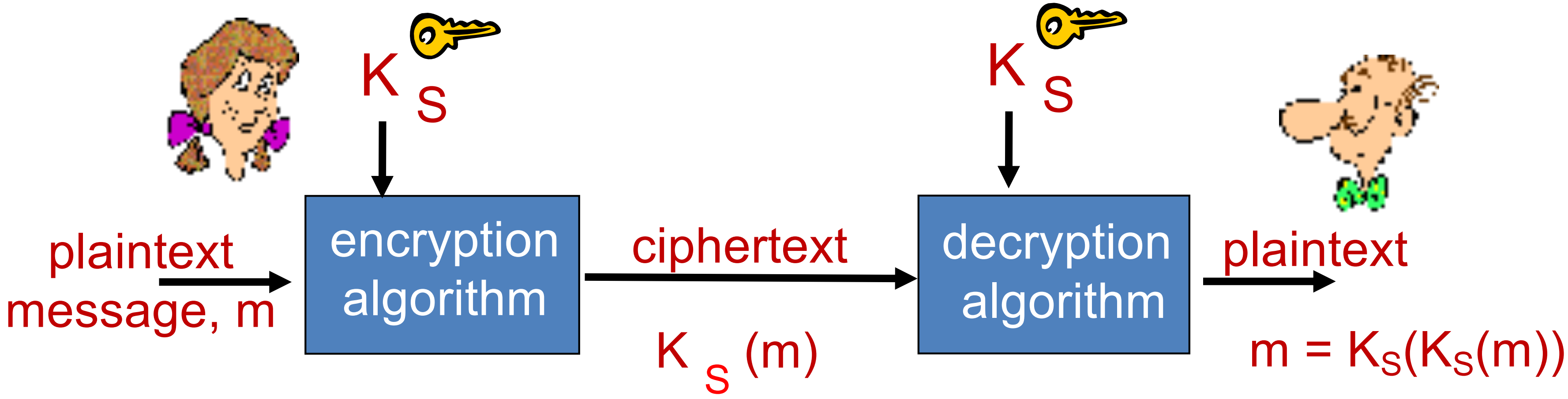
$$K_B^-(K_B^+(m)) = m$$

- given public key, it should be impossible to compute private key

RSA: Rivest, Shamir, Adelson algorithm

More on Public Key in later week

Symmetric Key Cryptography



symmetric key crypto: Bob and Alice share same (symmetric) key: K_S

Q: how do Bob and Alice agree on key value?

Session Keys

- Public-key cryptography is computationally intensive as compared to symmetric key crypto
- Exponentiation is computationally intensive
- Symmetric Algorithms much faster than RSA

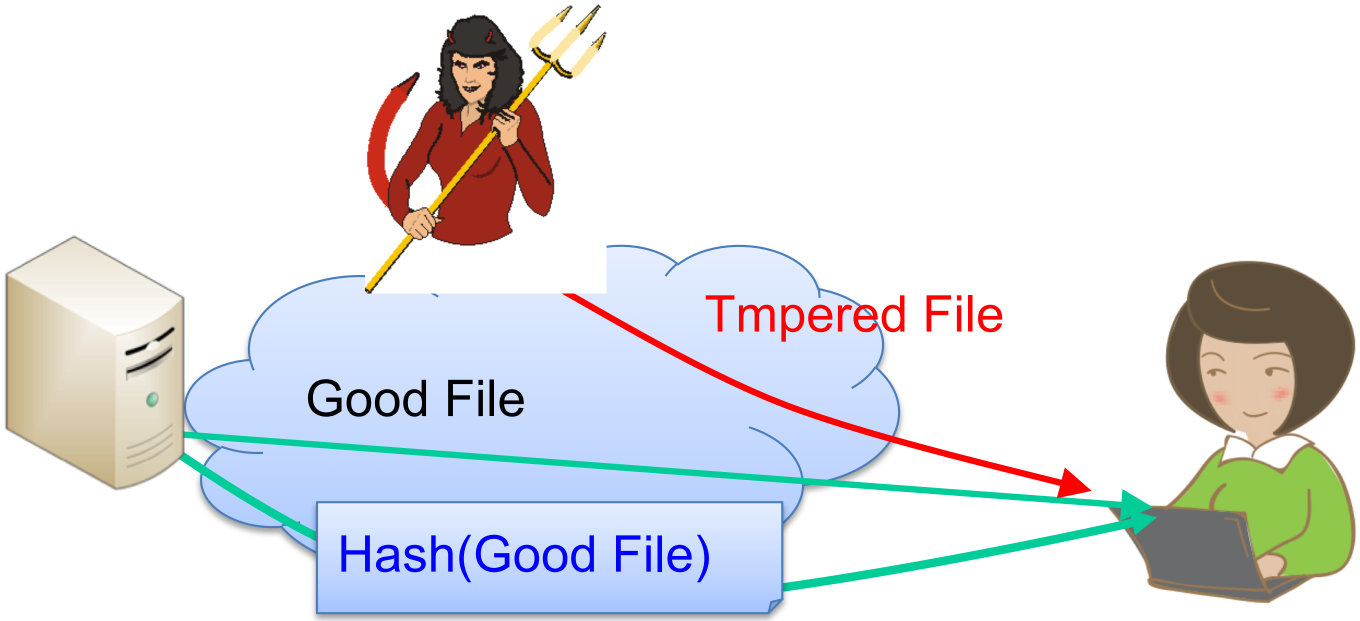
Session key, K_S

- Bob and Alice use RSA to exchange a symmetric key K_S
- Once both have K_S, they use symmetric key cryptography

Confidentiality vs Integrity

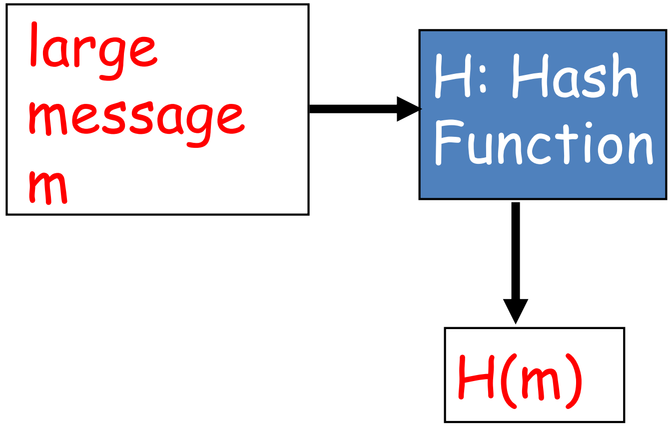
- Confidentiality: message private and secret
- Integrity: protection against message tempering
- Encryption alone may not guarantee integrity
  - Attacker can modify message under encryption without learning what it is
- Public Key Crypto Standards (PKCS)
  - "RSA encryption is intended primarily to provide confidentiality… It is not intended to provide integrity"
- Both Confidentiality and integrity are needed for security

Example on Integrity



- Software distribution protection:
  - For a Good File and Hash(Good File), it is infeasible to find a Tampered File (containing rootkit or Trojan) such that Hash(Good File) = Hash(Tampered File)

Hash/Message Digests (MD)



- Function H( ) that takes as input an arbitrary length message and outputs a fixed-length string: "message signature"
- The values returned H( ) are called **hash values**, **hash codes**, **digests**, or simply **hashes**.
- Note that H( ) is a many-to-1 function
- H( ) is often called a "hash function"

- Desirable properties:
  - Easy to calculate
  - Irreversibility: Can't determine m from H(m)
  - Collision resistance: Computationally difficult to produce m and m' such that H(m) = H(m')
  - Seemingly random output

MD Randomness

Message digest of a hashing should have a random pattern

- Randomness: any bit in digest is “1” half the time
- Change input only one bit, and the hash will change half of the digest bits
- Diffusion: if hash function does not exhibit the avalanche effect to a slight change of input, then it has poor randomization, and thus a cryptanalyst can make predictions about the input, being given only the output

Poor Crypto Hash Example

Internet checksum has some properties of hash function:

- produces fixed length digest (16-bit sum) of message
- is many-to-one

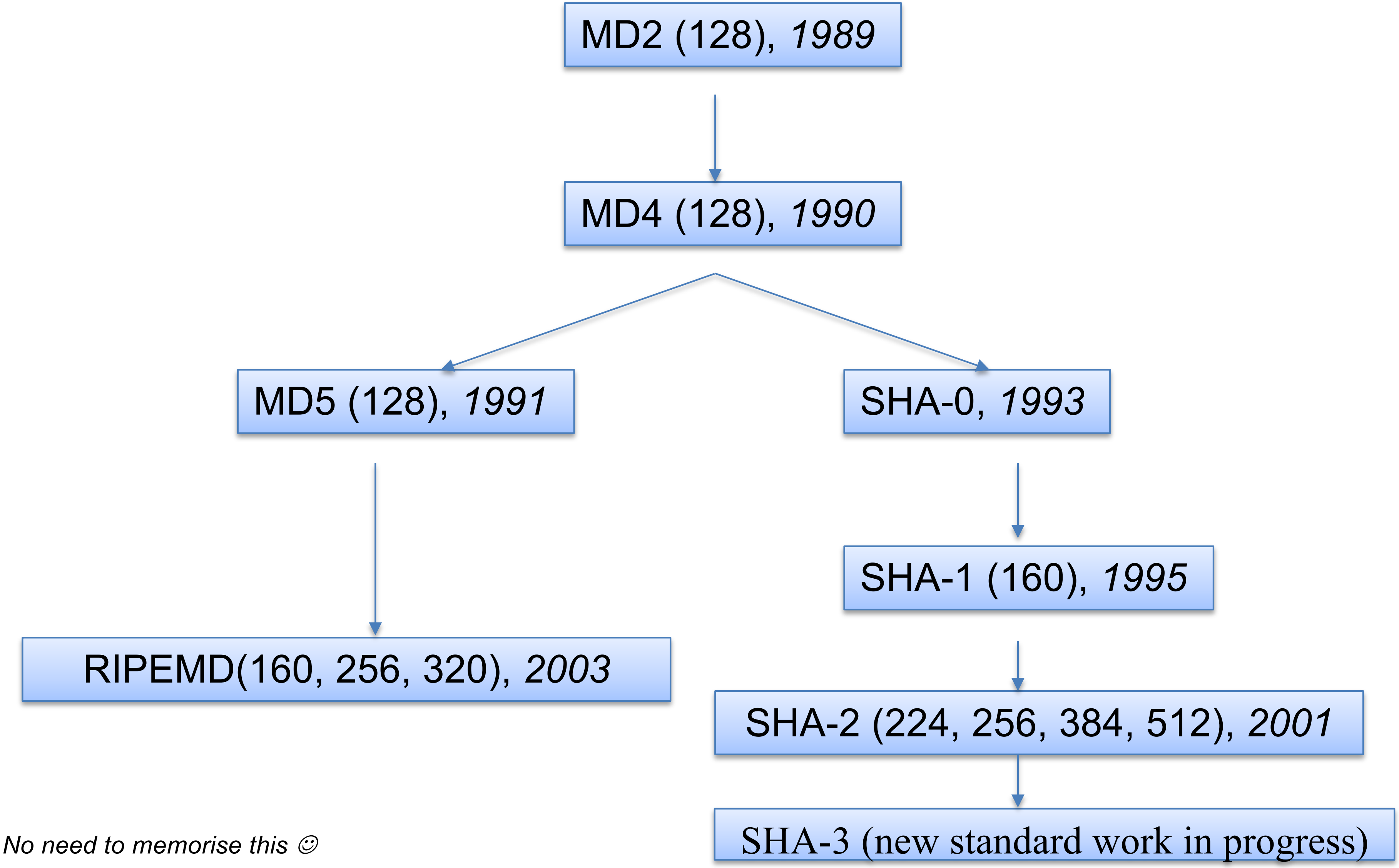
But given message with given hash value, it is easy to find another message with same hash value:

message	ASCII format
I O U 1	49 4F 55 31
0 0 . 9	30 30 2E 39
9 B O B	39 42 D2 42
	B2 C1 D2 AC

message	ASCII format
I O U 9	49 4F 55 39
0 0 . 1	30 30 2E 31
9 B O B	39 42 D2 42
	B2 C1 D2 AC

different messages but identical checksums!

Standard Hash Function History



No need to memorise this ☺

Example: SHA-256

- Plaintext

  Phishing Explained

  Phishing scams are typically fraudulent e-mail messages appearing to come from legitimate sources like your bank, your Internet Service Provider, eBay, or PayPal, for example. These messages usually direct you to a fake web site and ask you for private information (e.g., password, credit card, or other account updates). The perpetrators then use this private information to commit identity theft.

  Warning Signs

  ……….

  URLs Don't Match - Place your mouse over the link in the e-mail message. If the URL displayed in the window of your browser is not exactly the same as the text of the link provided in the message, run. It's probably a fake. Sometimes the URLs do match and the URL is still a fake.

- Digest in BASE64

Tx9S2IwqlrGI7hhNJ4s5K7qiYt3PjQSD6vWH4QBl7
yg =

- The corresponding digest in binary format is listed as follows:

1010110001010110011110011101000000011101000001111101101110100101001011011111101101000100001
0001001101100000101100110101010110110010011011111000100110011011001010100110100011101001101
1011111010110000000011011000101111010011011111111110100110111110000110010100 1

Example: Diffusion in SHA-256

- Plaintext: only change the first letter from P to Q

  Qhishing Explained

  Phishing scams are typically fraudulent e-mail messages appearing to come from legitimate sources like your bank, your Internet Service Provider, eBay, or PayPal, for example. These messages usually direct you to a fake web site ……….

  URLs Don't Match - Place your mouse over the link in the e-mail message. If the URL displayed in the window of your browser is not exactly the same as the text of the link provided in the message, run. It's probably a fake. Sometimes the URLs do match and the URL is still a fake.

- Digest in BASE64 changes from

  `Tx9S2IwqlrGI7hhNJ4s5K7qiYt3PjQSD6vWH4QBl7yg =`

  to

  `AAHew/sW6W4X39EQq7ctFEb3PyHxka+T3D1UQJhDkw =`

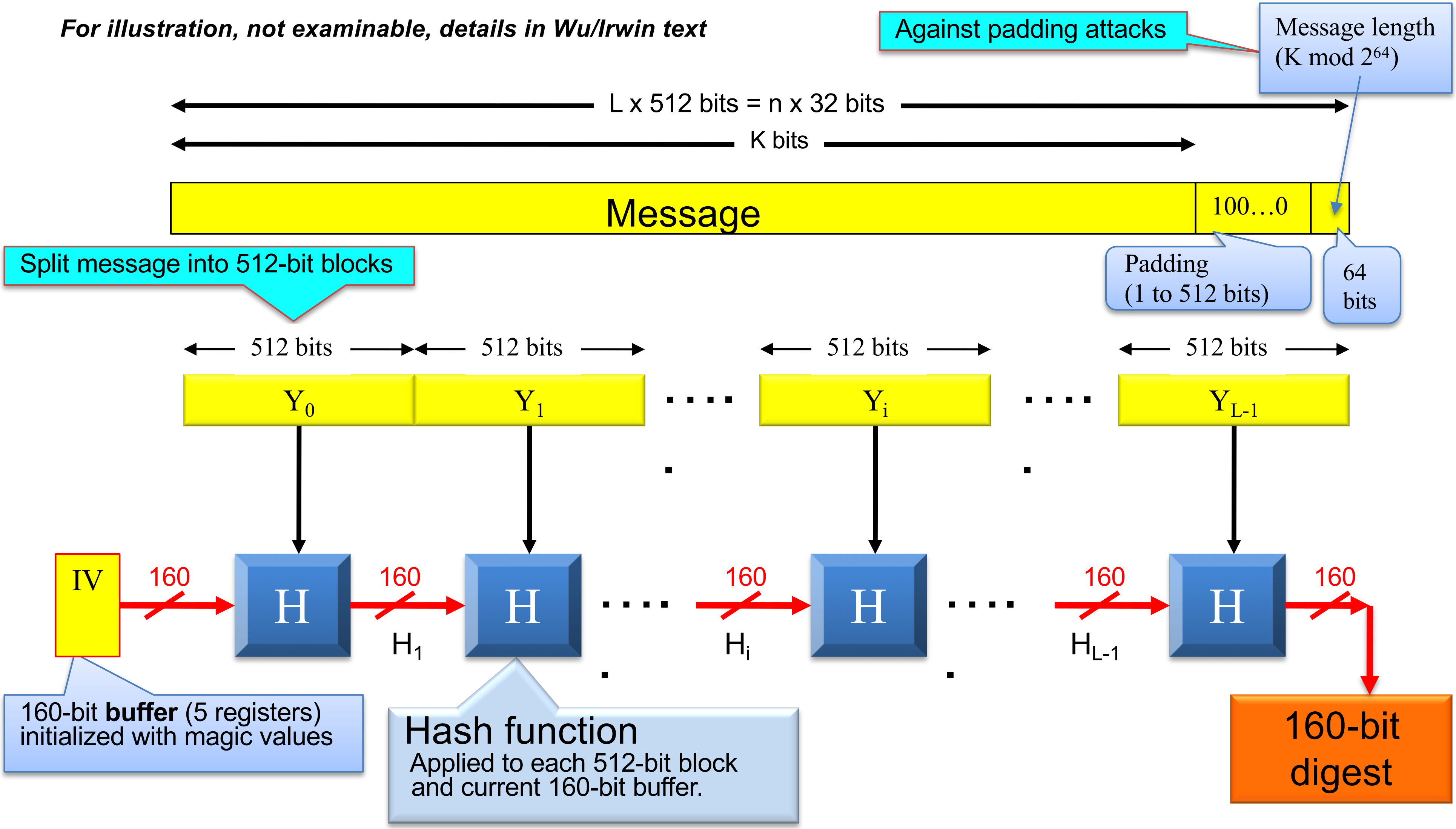
- The corresponding digest in binary format is listed as follows:

```
101001001100101110001011010100010010110011001111101000000100101101101110010
110001011000001001100111001100101100111001000001000010100010101000011011111
010100101101101111111111111110111011111011010011000001000001111001011101 10
0110010110101111111010000010000
```


CySPri Laboratory

Basic Structure of SHA-1

For illustration, not examinable, details in Wu/Irwin text



SHA properties

Algorithm Name	Max. Message Size (bits)	Block Size (bits)	Word size (bits)	Digest size (bits)	Collision resistance (bits)
SHA-1	2^{64}	512	32	160	80
SHA-256	2^{64}	512	32	256	128
SHA-384	2^{128}	1024	64	384	192
SHA-512	2^{128}	1024	64	512	256

No need to memorise this ☺, appreciate and consult table as needed. Details on collusison resistance is beyond scope of this course. We will see some related practical issues in WEP protocol later. Basically determines stregnth of scheme under bruteforce attack i.e. how quickly a collision can be found on a fast machine.

160 bit digest is consider unsafe with fast computers.

Attacks on Hash (Self study/no-exam)

- MD5 is one of the most widely used cryptographic hash functions
- Attack on MD5 to find collisions efficiently
  - About 15 minutes up to an hour computation time
  - Finding a collision for MD5 is easily feasible
- This attack is also able to break hash functions efficiently, including HAVAL-128, MD4, RIPEMD, SHA-0 and SHA-1
- SHA-1 near-collision attack needs $2^{57.5}$ hashes
  - Marc Stevens: HashClash on 21/2/2019
  - https://marc-stevens.nl/p/hashclash/
  - SHA-1 Chosen-prefix collision attack: $2^{77.06}$

- Ref:
  - Xiaoyun Wang, Hongbo Yu: How to Break MD5 and Other Hash Functions. EUROCRYPT 2005: 19-35
  - Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, Xiuyuan Yu: Cryptanalysis of the Hash Functions MD4 and RIPEMD. EUROCRYPT 2005: 1-18
  - Xiaoyun Wang, Yiqun Lisa Yin, Hongbo Yu: Finding Collisions in the Full SHA-1. CRYPTO 2005: 17-36
  - Xiaoyun Wang, Hongbo Yu, Yiqun Lisa Yin: Efficient Collision Search Attacks on SHA-0. CRYPTO 2005: 1-16
  - Hongbo Yu, Gaoli Wang, Guoyan Zhang, Xiaoyun Wang: The Second-Preimage Attack on MD4. CANS 2005: 1-12

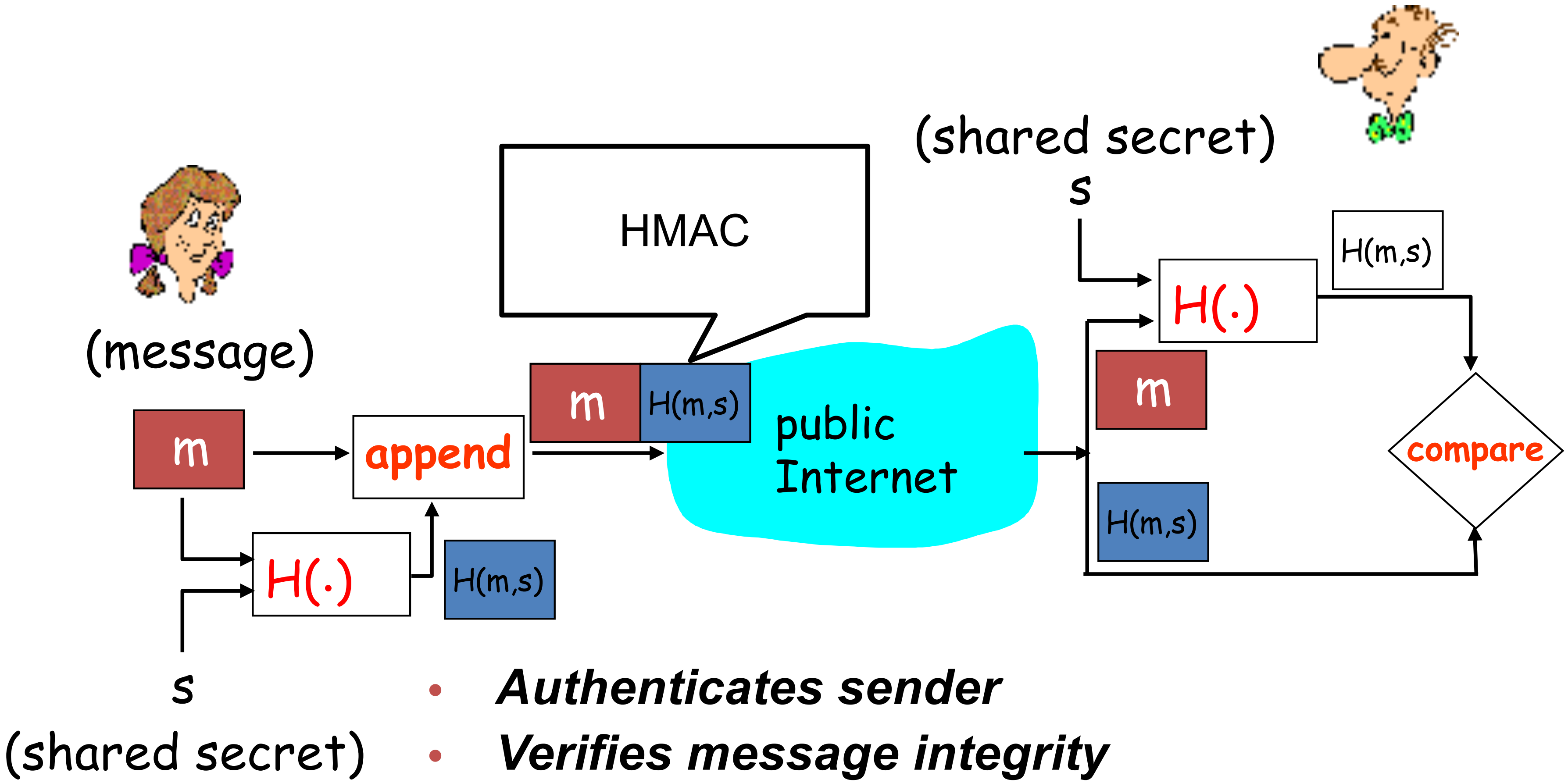
Integrity vs Authentication

- suppose Alice creates msg m, and calculates hash H(m) using a hash function (e.g. SHA-1)
- Alice send the extended message (m, H(m)) to Bob
- Bob upon receipt of this message independently calculates H(m') from the message
- If H(m) = H(m'), message integrity verified.
- *What if Trudy intercepts the original message, replaces it with a new message n and its hash H(n) ?*

HMAC

- FIPS 198, RFC 2104
  - Keyed-hash message authentication code: a message authentication code that uses a cryptographic key in conjunction with a hash function
    - Runs data and authentication key through hash function twice
  - Hashing is faster than encryption in software
  - Allow the use of any hash function, e.g., SHA-256, or SHA-512
  - No US export restrictions on HMAC and hash
- Used in TLS and IPSec (later weeks)

HMAC: Integrity and Authentication



- ***Authenticates sender***
- ***Verifies message integrity***
- No encryption ! – Confidentiality not always needed
- Also called “keyed hash”
- Please note that HMAC take both message and key as input, not a simple concatenation.

Integrity of files using HMAC

Example:

- Use the following key in ASCII format.

  `TigersFootballWarEagleBCSChampio`

- The corresponding Hex format for the key is

  546967657273466F6F7462616C6C5761724561676C65424353436
  8616D70696F

Example: HMAC-SHA-256

- Plaintext

  Phishing Explained

  Phishing scams are typically fraudulent e-mail messages appearing to come from legitimate sources like your bank, your Internet Service

  …….

- HMAC Digest in BASE64

  `HadCmwGX9EGKBon0C+6XEinXCI8 =`

- Only change the first char from *Tiger* to *Siger* in key, HMAC Digest in BASE64

  `Ar89wBq+6rxq4Eenho53TMiHvSw =`

Hash, MAC and HMAC

- One way Hash Function: Doesn't take any secret key or its operation
  - Easy to compute (both hardware and software solutions possible)
  - Concatenation of a Secret Key and Message can be passed through a hash function without any need for encryption/decryption for efficiency.
- Message Authentication Code: MAC
  - Can use functions like DES, last bits of the cipher-text can be used as code. However, encryption software is slow, hence may be avoided.
- HMAC: Key is XORed with ipad (part of pad) and then concatenated with the message m. This mix is run through a hash function. The result is then concatenated with XOR of Key and opad (part of pad). Now, this whole mix is run through hash function one more time.

Digital signatures

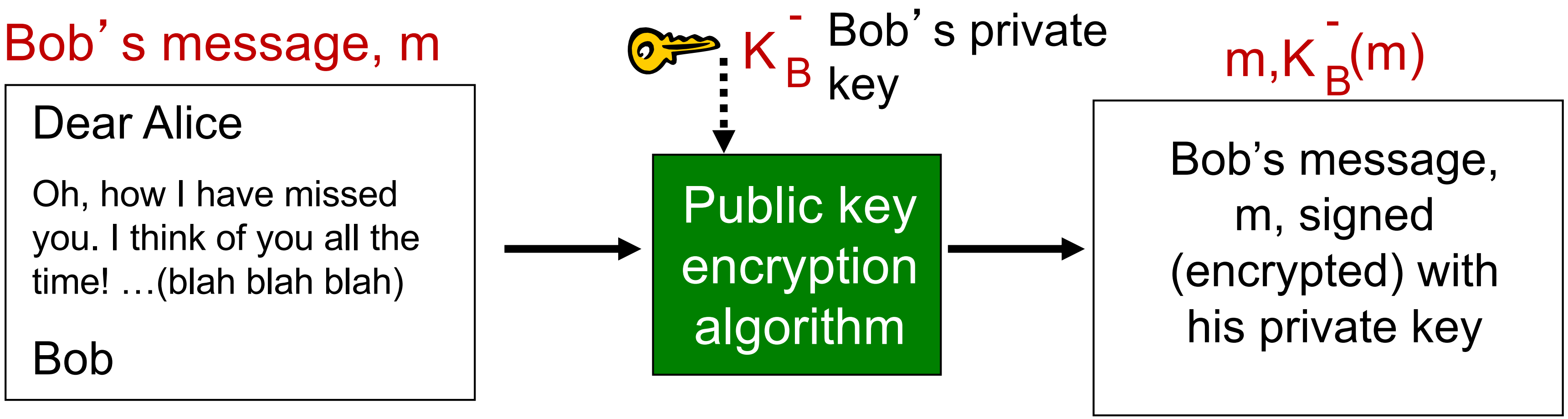
cryptographic technique analogous to hand-written signatures:

- sender (Bob) digitally signs document, establishing he is document owner/creator.
- *verifiable, nonforgeable:* recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document

Digital signatures

simple digital signature for message m:

- Bob signs m by encrypting with his private key K_B^-, creating "signed" message, $K_B^-(m)$



Digital signatures

- suppose Alice receives msg m, with signature: m, $K_B^-(m)$
- Alice verifies m signed by Bob by applying Bob's public key K_B^+ to $K_B^-(m)$ then checks $K_B^+(K_B^-(m)) = m$.
- If $K_B^+(K_B^-(m)) = m$, whoever signed m must have used Bob's private key.

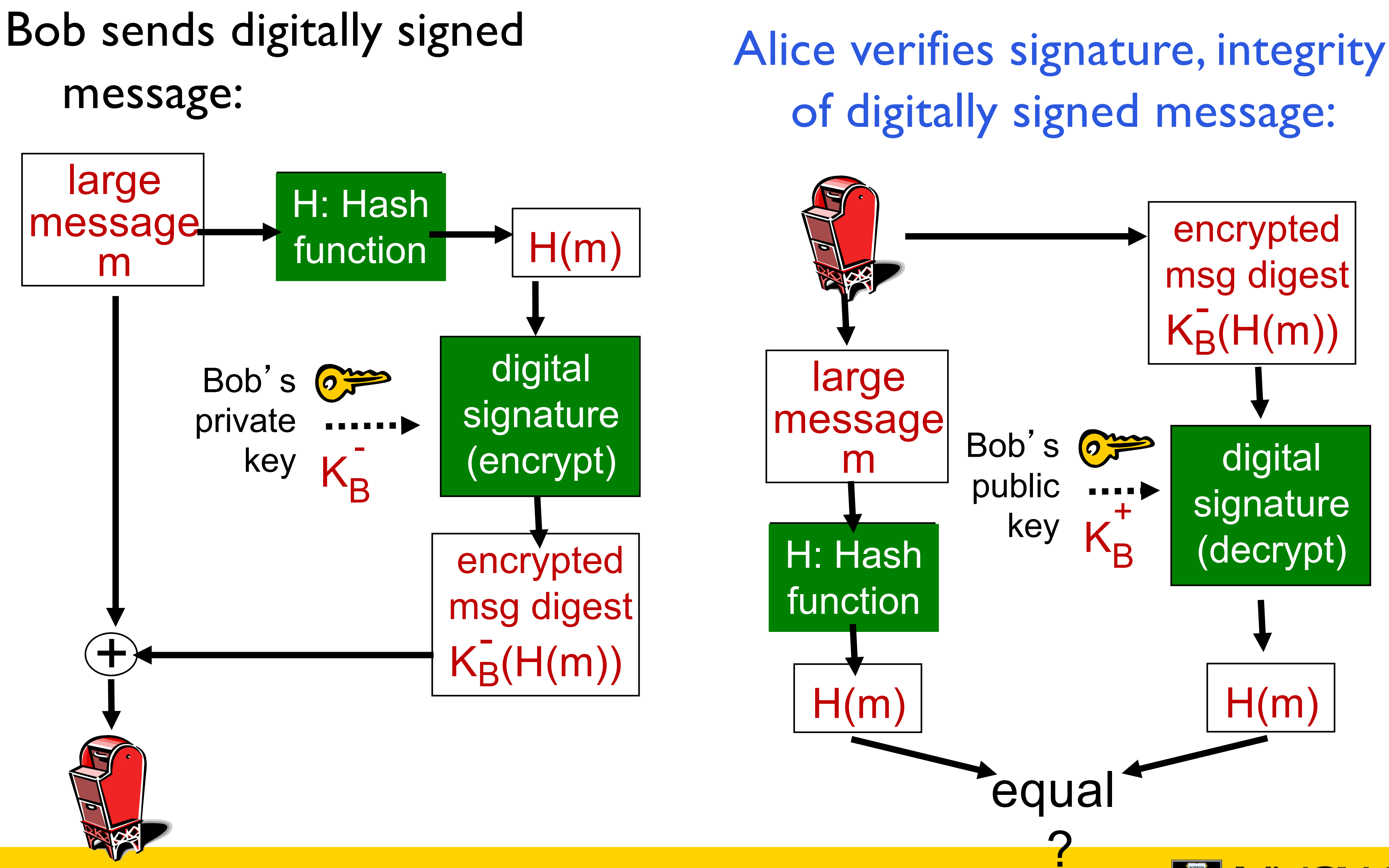
Alice thus verifies that:

- Bob signed m
- no one else signed m
- Bob signed m and not m'

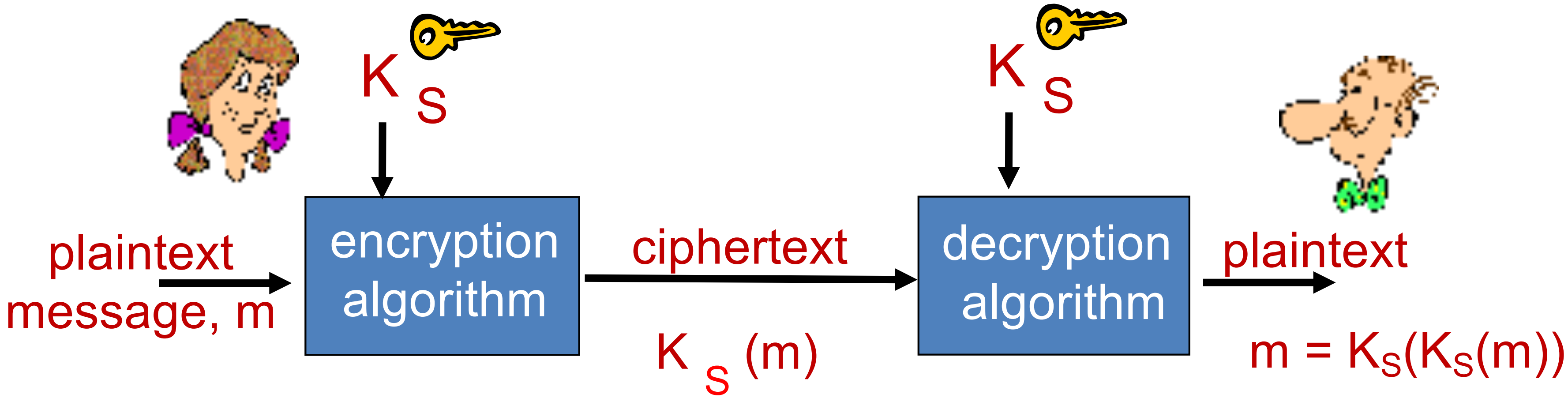
non-repudiation:

✓ Alice can take m, and signature $K_B^-(m)$ to court and prove that Bob signed m

Digital signature = signed message digest



Symmetric Key Cryptography (contd)



Lets look deeper into symmetric ciphers

Two types of symmetric ciphers

- Block ciphers
  - Break plaintext message in equal-size blocks
  - Encrypt each block as a unit
  - Used in may Internet protocols (PGP-secure email, SSL (secure TCP), IPsec (secure net-transport layer)
- Stream ciphers
  - encrypt one bit at time
  - Used in secure WLAN

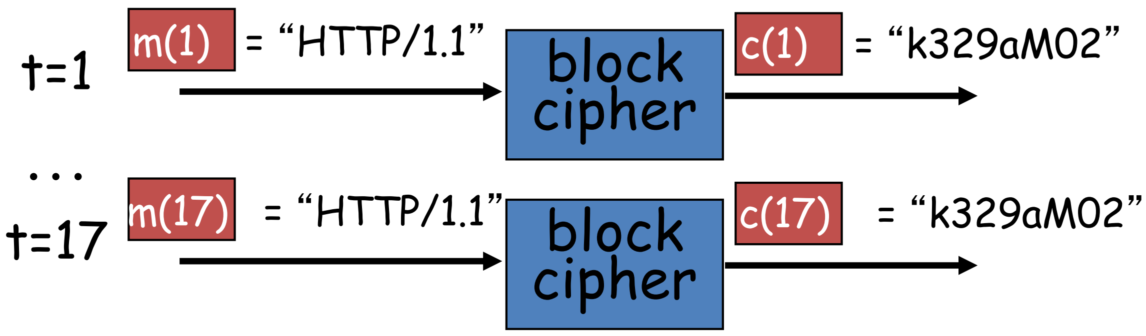
Block Cipher

- Ciphertext processed as *k* bit blocks
- 1-to-1 mapping is used to map k-bit block of plaintext to k-bit block of ciphertext
- E.g: k=3 (see table)
  - 010110001111 => 101000111001
- Possible permutations = 8! (40,320)
- To prevent brute force attacks
  - Choose large k (64, 128, etc)
- Full-block ciphers not scalable
  - E.g., for k = 64, a table with 2^{64} entries required
  - instead use function that simulates a randomly permuted table

Input	Output
000	110
111	001
001	111
010	101
011	100
100	011
101	010
110	000

Cipher Block Chaining

- cipher block: if input block repeated, will produce same cipher text:



- Sender creates a random *k*-bit number *r(i)* for *i*th block and calculates
  - $c(i) = K_S(\ m(i) \oplus r(i)\ )$
  - Sends ***c(1), r(1), c(2), r(2), c(3), r(3)***….
    - *r(i)* sent in clear but *Ks* not known to attackers.

010 XoR 001 = 011
From table 011 encrypted to -> 100

CBC Example

- Example: sent text 010010010 if no CBC, sent txt = *101101101*
  - *The sent text is result of 1-to-1 mapping is used to map k-bit block of plaintext to k-bit block of ciphertext (see table earlier)*
- *Lets use the following random bits*
  - *r(001), r2(111), r3(100)*
- *First we XOR the plain text with the above random bits:*
  - *E.g 010 XOR 001 = 011*
  - *Now do table lookup for 011 -> 100*
- Use above technique to generate cipher text *c(1) = 100, c(2) = 010, c(3) = 000*: NOTE all three output different even though same plain text 010.
- Inefficient: twice as many bits sent

- NOTE: Try other bits in the example to make sure you understand how this works.

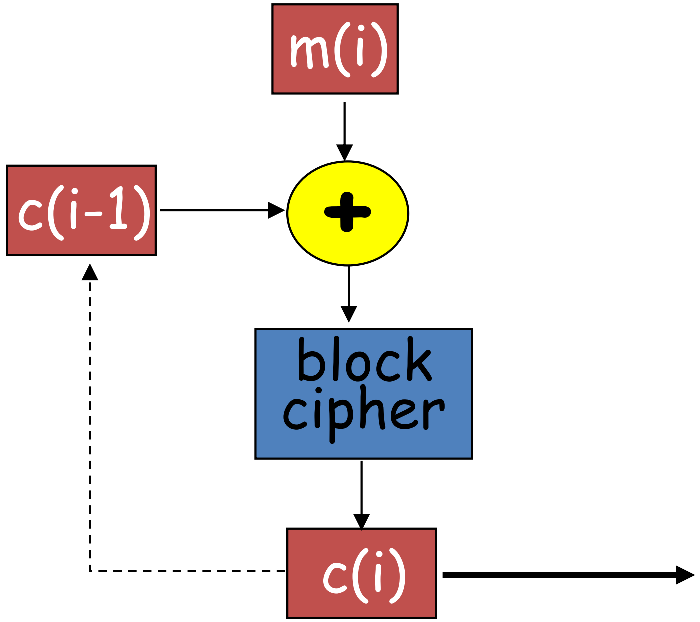
CBC: Sender

- *cipher block chaining:* XOR ith input block, $m(i)$, with previous block of cipher text, $c(i-1)$
  - $c(0)$ is an Initialisation Vector transmitted to receiver in clear
  - First block:

$$c(1) = K_S(m(i) \oplus c(0))$$

  - *Subsequent blocks:*

$$c(i) = K_S(m(i) \oplus c(i-1))$$



CBC: Receiver

- How to recover $m(i)$?
  - Decrypt with *Ks* to get $s(i) = Ks(c(i)) = m(i) \oplus c(i-1)$
  - Now the receiver knows $c(i-1)$, it can get
    $m(i) = s(i) \oplus c(i-1)$
  - **IV** sent only once
  - Intruder can't do much with **IV** since it doesn't have Ks
  - CBC has important consequence for designing secure network protocols

Block Ciphers

- Block cipher needs to wait for one block before processing it
  - Suitable for storage
- Operates on a single block of plaintext
  - 64 bits for Data Encryption Standard (DES), 128 bits for Advanced Encryption Standard (AES)
    - AES much (6x) faster than DES
- Computationally infeasible to break block cipher by brute-force by cracking the key
  - Brute force decryption (try each key)

Symmetric key crypto: DES

DES: Data Encryption Standard (US encryption standard [NIST 1993])

- 56-bit symmetric key, 64-bit plaintext input
- Block cipher with cipher block chaining
- How secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase decrypted (brute force) in less than a day
    - DES Challenge III: Distributed. Net worked with EFF's supercomputer and a world-wide network of 100,000 PCs to crack it within 22 hours and 15 minutes, Testing 2.45 billion keys per second
  - no known good analytic attack
- Making DES more secure:
  - 3DES: encrypt 3 times with 3 different keys

Symmetric key crypto: DES

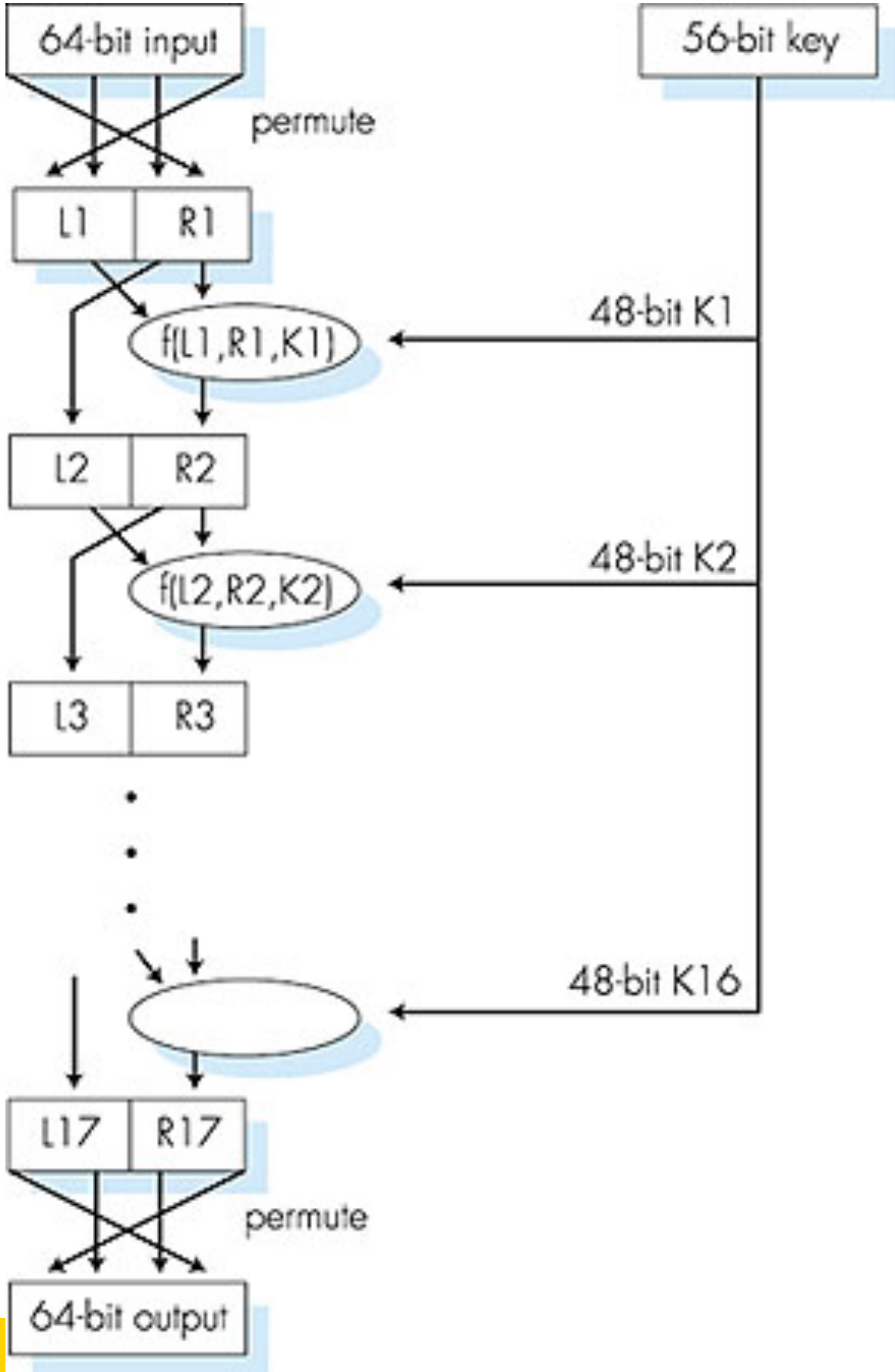
DES operation

initial permutation

16 identical "rounds" of function application, each using different 48 bits of key

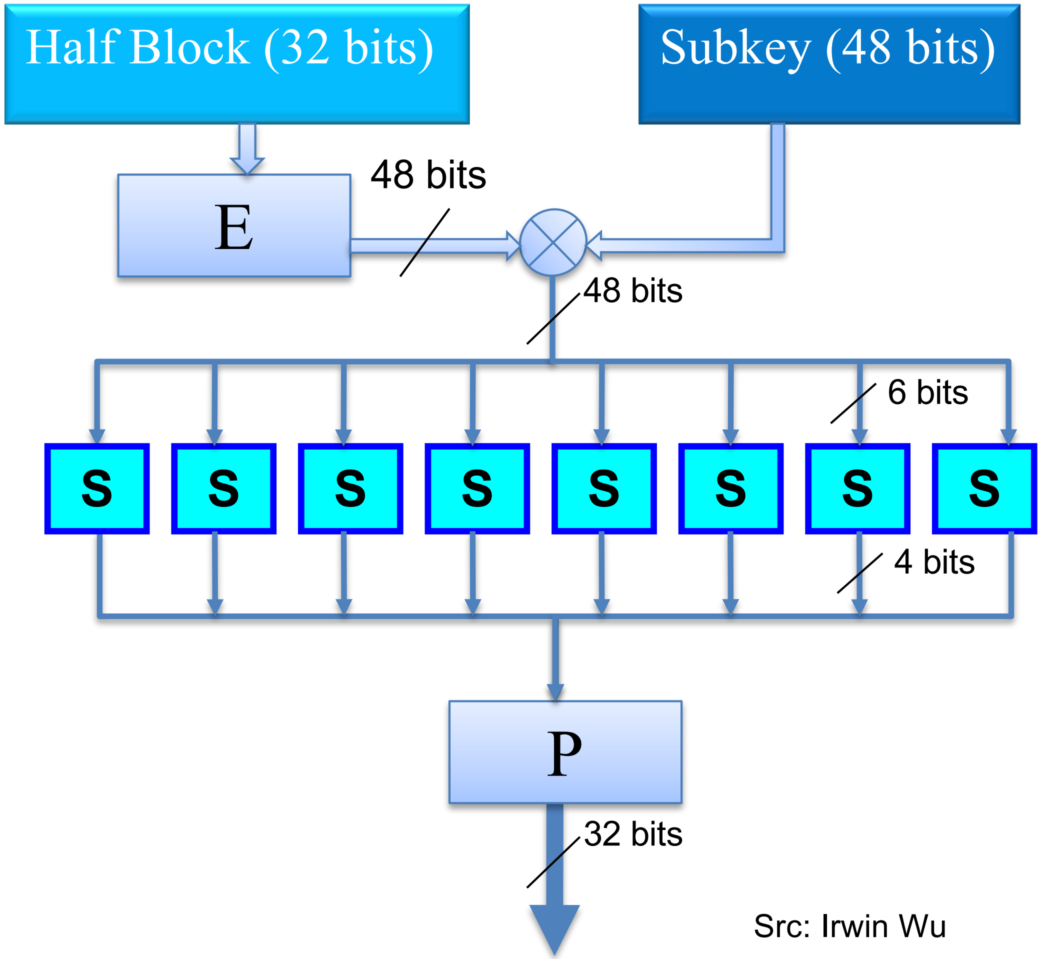
final permutation

No need to memorise this



Feistel (F) function

- Feistel (F) function
  - Expansion (E): the 32-bit half-block is expanded to 48 bits using the expansion permutation
  - Key mixing:
    - Sixteen 48-bit subkeys: one for each round from 56 bit key
    - Derived from the main key using the key schedule
    - E is combined with a subkey using an XOR operation
  - P: permutation function
    - P yields a 32-bit output from a 32-bit input by permuting the bits of the input block
  - S-box: Substitution
    - Transforms input bits using substitution tables to provide diffusion
    - Spread plaintext bits throughout ciphertext
    - Small change in either the key or the plaintext should cause a drastic change in the ciphertext (avalanche effect)



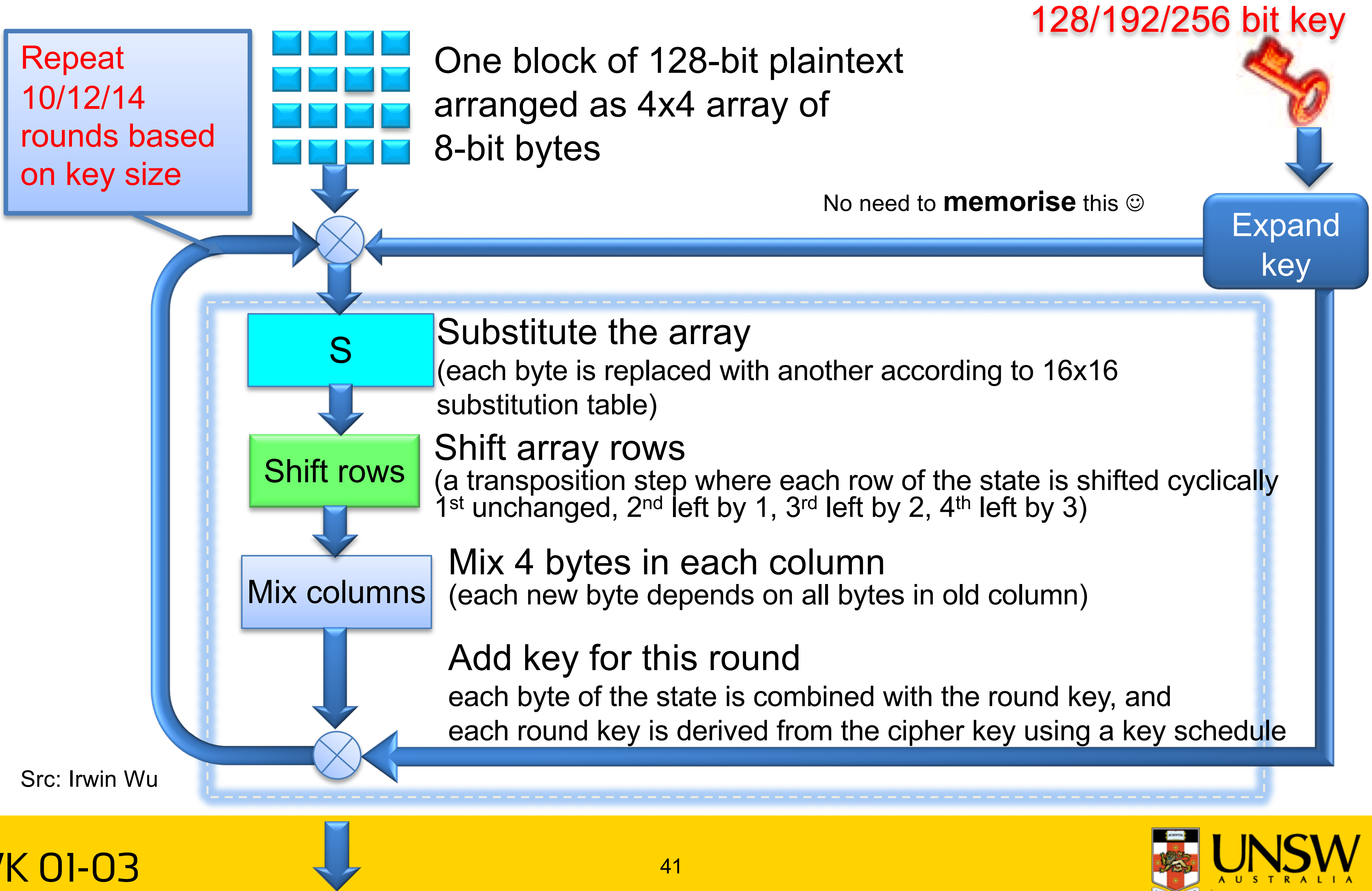
Src: Irwin Wu

No need to memorise this ☺

AES: Advanced Encryption Standard

- Symmetric-key NIST standard, replaced DES (Nov 2001)
- processes data in 128 bit blocks
- 128, 192, or 256 bit keys
- Brute force decryption (try each key) taking few secs on DES, takes 149 trillion years for AES
    - Universe lifetime: 100 billion years

AES -Structure

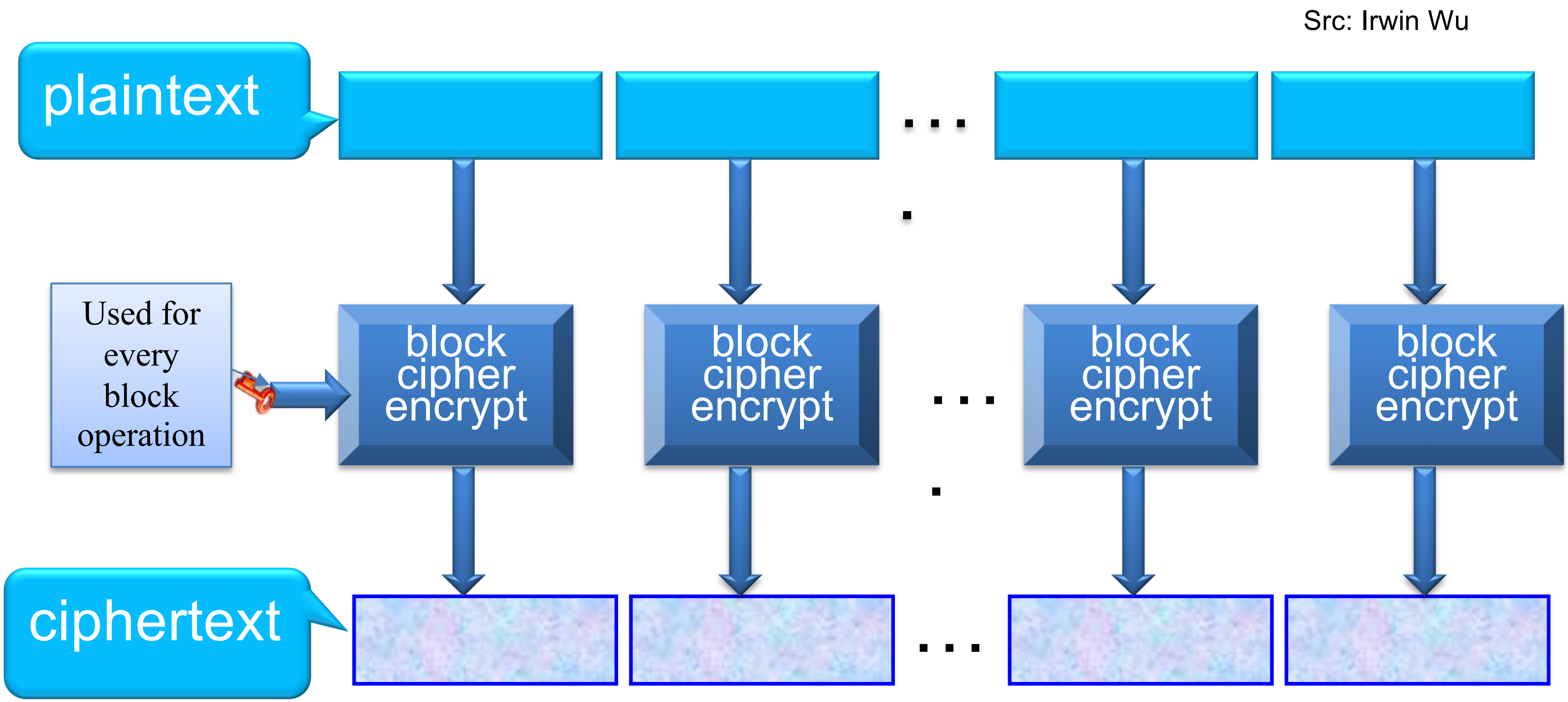


Src: Irwin Wu

AES Confidentiality Modes

- Five confidentiality modes of operation for symmetric key block cipher algorithms, such as AES
- Electronic Codebook (ECB),
  - Split plaintext into blocks, encrypt each one separately using the block cipher
- Cipher Block Chaining (CBC) mode
  - Split plaintext into blocks, XOR each block with the result of encrypting previous blocks
- Cipher Feedback (CFB), Output Feedback (OFB), and Counter (CTR) modes: stream ciphers based on block cipher (will discuss as needed in future lectures)

ECB Mode Encryption



- Problem: Identical blocks of plaintext produce identical blocks of ciphertext
- No integrity checks: can mix and match blocks

Weakness of ECB

- Message repetitions may show in ciphertext
- Weakness is due to the encrypted message block uniquely mapped to the plaintext block
- Main use is sending only a few blocks of short data

- *If longer messages, use Cipher Block Chaining (as discussed earlier)*

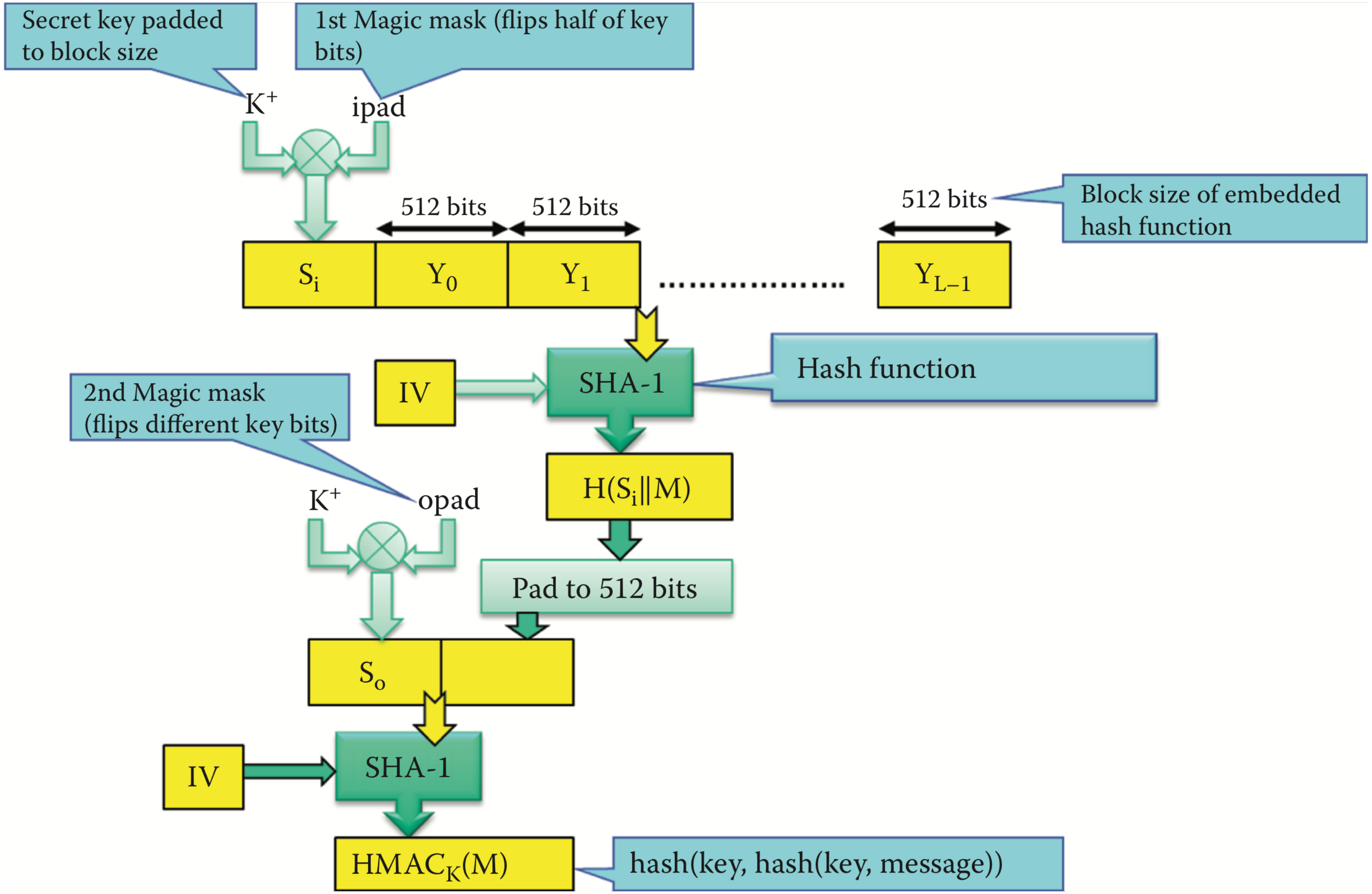
Stream Ciphers

- NEXT WEEK
  - Stream Ciphers
  - How to design a flawed Security Protocol:
    - WEP Case Study
    - Fixing a flawed Protocol: WPA, WPA2

Summary

- Lot of these crypto mechanism are building block for security protocols: both wired and wireless
- Acknowledgement:
  - Adaptation of foils from Kurose/Ross (revision from basic networking subject COMP3331)
  - Some material from Wu & Irwin book has lot more on various protocols and standards in chapter 20.
    - Details of algorithms beyond scope, you can consult many texts on crypto if interested.
  - Stallings: Network Security Essentials, Chapter 2 and 3 has good summary of Symmetric Encryption, Hash, HMAC, MAC etc.
  - Also refer to *Cryptography KA* *from https://www.cybok.org/*

HMAC



Src: Irwin Wu: figure 20.6