Cryptography?

- Art of writing or solving (secret) codes
- Modern cryptography provides mechanisms for confidentiality, integrity, authentication, non-repudiation, privacy, …
- Very broad subject
- We focus primarily on using it as a (black box) tool for secure communication

Principles of Cryptography

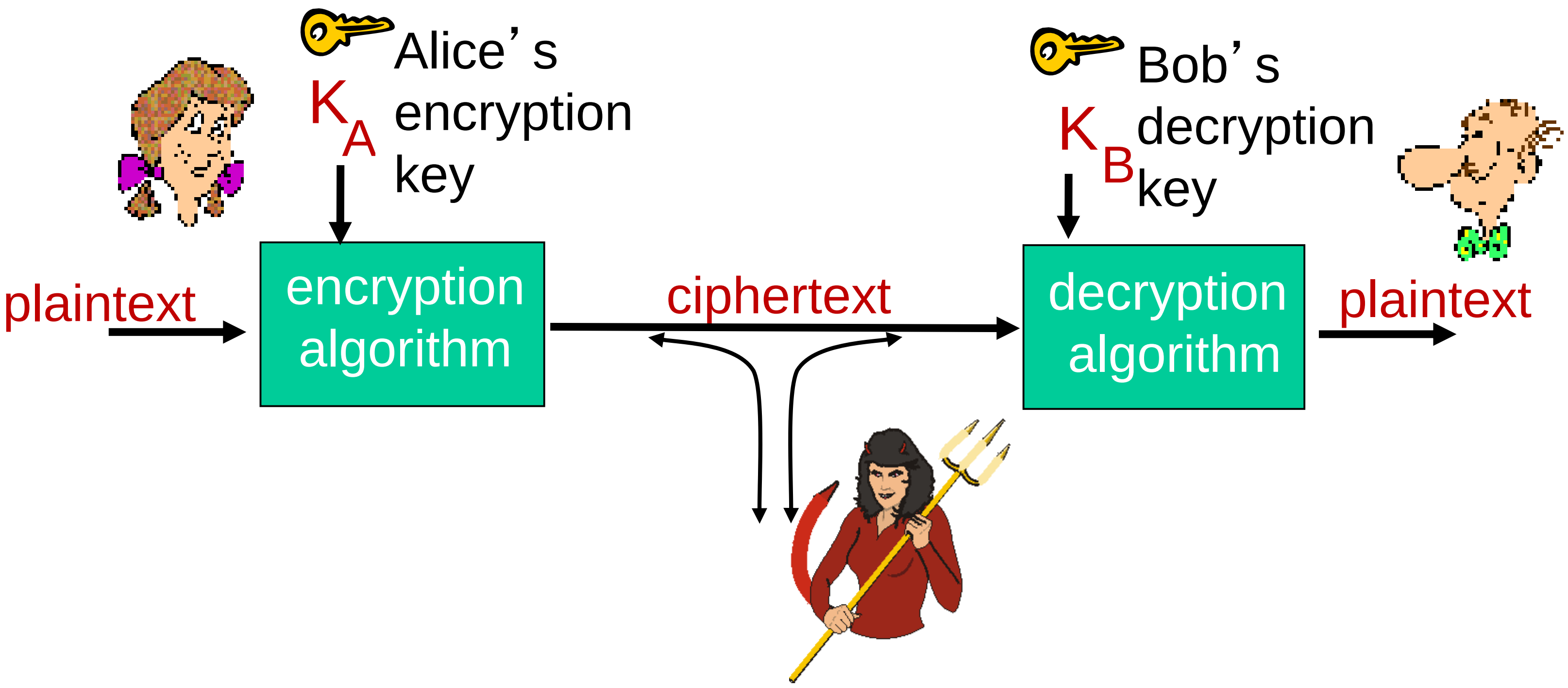
Bheemarjuna Reddy Tamma
IIT HYDERABAD

Credits: *Adapted from Kurose and Ross textbook on Computer Networking and based on slides from Stefan Savage, Steven Bellovin and a host of others and Internet sources*

Encryption & Decryption

- Plaintext (*m*): unencrypted message to be sent by Alice
  - Binary string of arbitrary length
- Ciphertext (*c*): encrypted version of message by using encryption function E
  - $c = E(m)$
  - *c* is also a binary string (may not be same length as plaintext)
- Bob decrypts c by using decryption function D
  - $m = D(c)$
- A **cipher** is an algorithm for transforming plaintext to/from ciphertext
  - Encryption and decryption functions should be parameterized by a key
  - Only the key is secret, but ciphers are considered public knowledge!

The language of cryptography



m plaintext message

$K_A(m)$ ciphertext, encrypted with key K_A

$m = K_B(K_A(m))$

Simple encryption scheme

substitution cipher: substituting one thing for another

- **monoalphabetic cipher**: substitute one letter for another

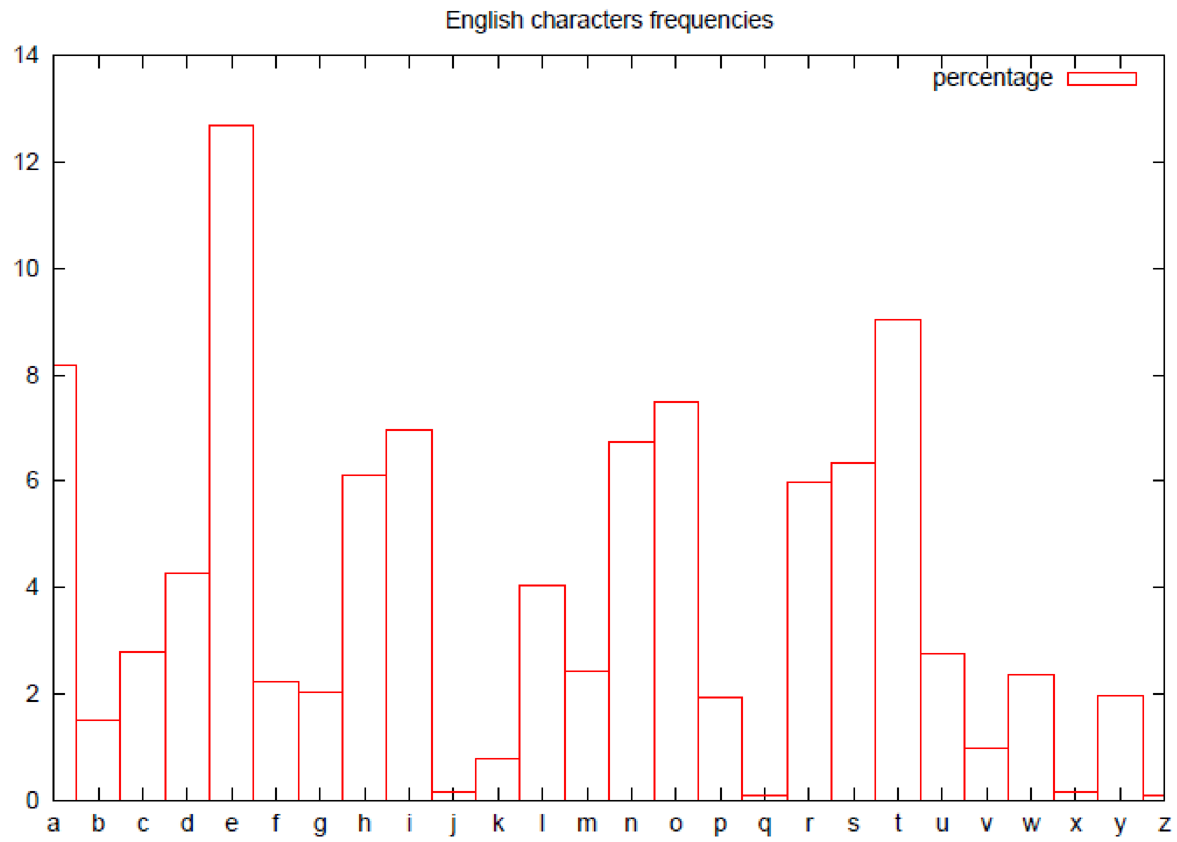
```
plaintext:   abcdefghijklmnopqrstuvwxyz

ciphertext:  mnbvcxzasdfghjklpoiuytrewq
```

e.g.:
```
Plaintext:  bob. i love you. alice
ciphertext: nkn. s gktc wky. mgsbc
```

Encryption key: mapping from set of 26 letters to set of 26 letters

Attacks on monoalphabetic cipher

- Easy to learn patterns
- Frequency analysis



A more sophisticated encryption approach

- n substitution ciphers, $M_1, M_2, \ldots, M_n$
- example, n=4 with cycling pattern of length 5
  - M_1, M_3, M_4, M_3, M_2;
  - for each new plaintext symbol, use subsequent subsitution pattern in cyclic pattern
  - drink: d from M_1, r from M_3, i from M_4, n from M_3, k from M_2

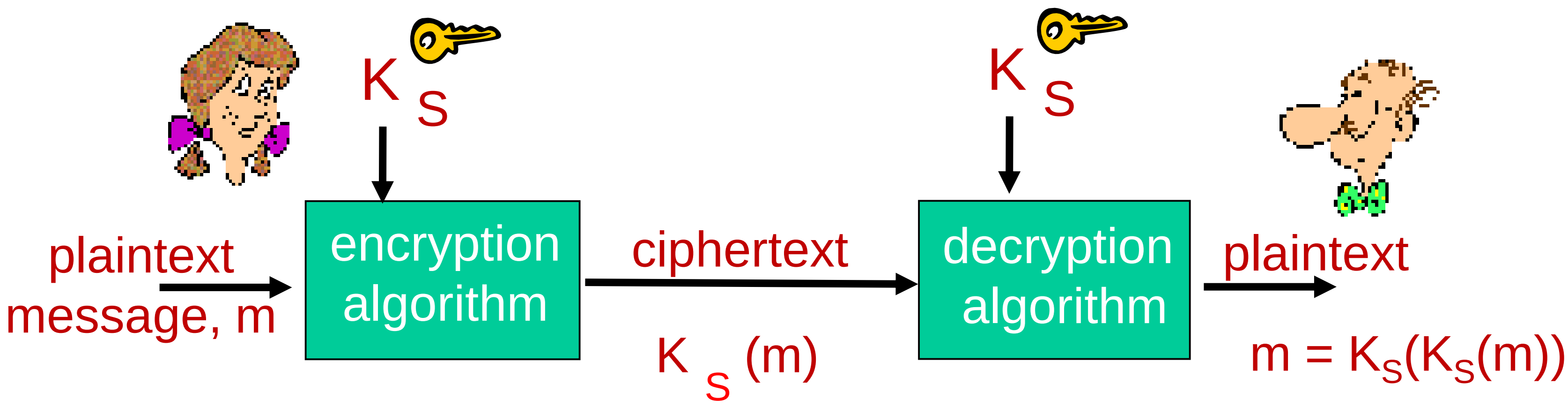
Encryption key: n substitution ciphers, and cyclic pattern

- key need not be just n-bit pattern

Cryptanalysis: Breaking an encryption scheme

- cipher-text only attack: Trudy has ciphertext she can analyze
- two approaches:
  - brute force: search through all keys
  - statistical analysis
- known-plaintext attack: Trudy has plaintext corresponding to ciphertext
  - e.g., in monoalphabetic cipher, Trudy determines pairings for a,l,i,c,e,b,o,
- chosen-plaintext attack: Trudy can get ciphertext for a chosen plaintext

Symmetric key cryptography



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

- e.g., key is knowing substitution pattern in monoalphabetic substitution cipher

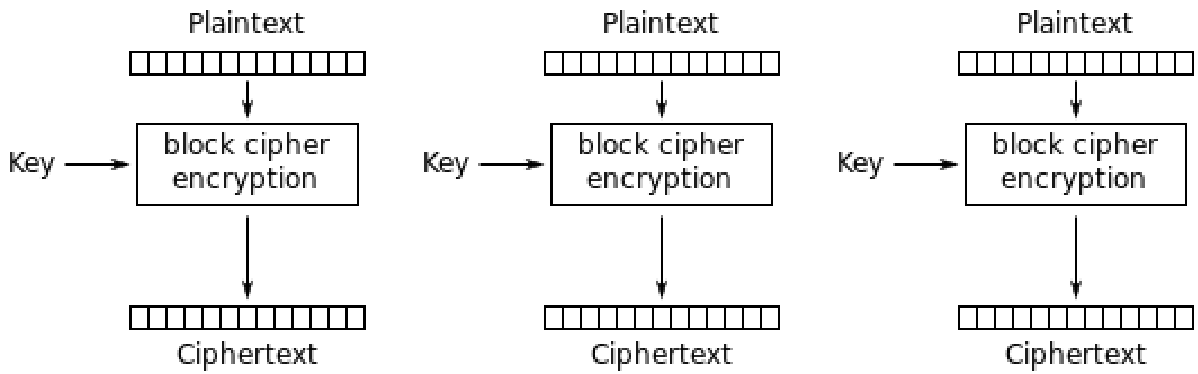
Q: how do Bob and Alice agree on key value?

A: Diffie-Hellman Symmetric Key Exchange Protocol, Public key crypto

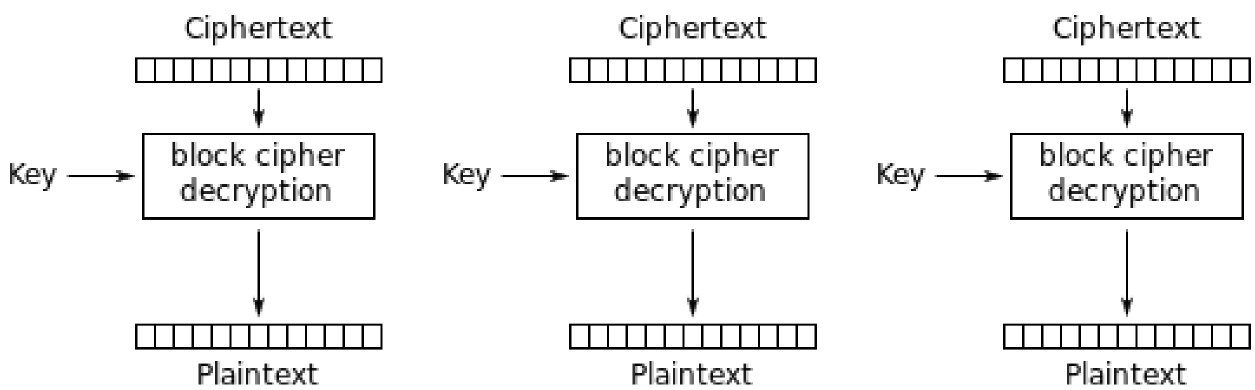
AES: Advanced Encryption Standard

- symmetric-key NIST standard, replaced DES which used 56-bit keys (Nov 2001)
- processes data in 128 bit blocks (Block Cipher)
- 128, 192, or 256 bit keys
- brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES!

AES: Electronic Codebook (ECB) Mode



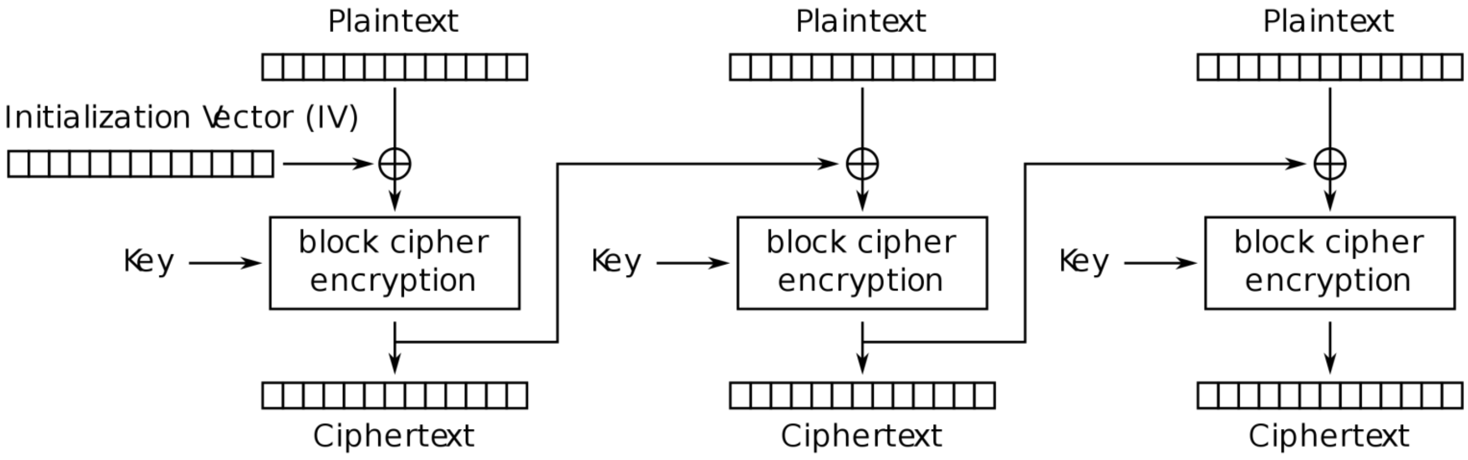
Electronic Codebook (ECB) mode encryption



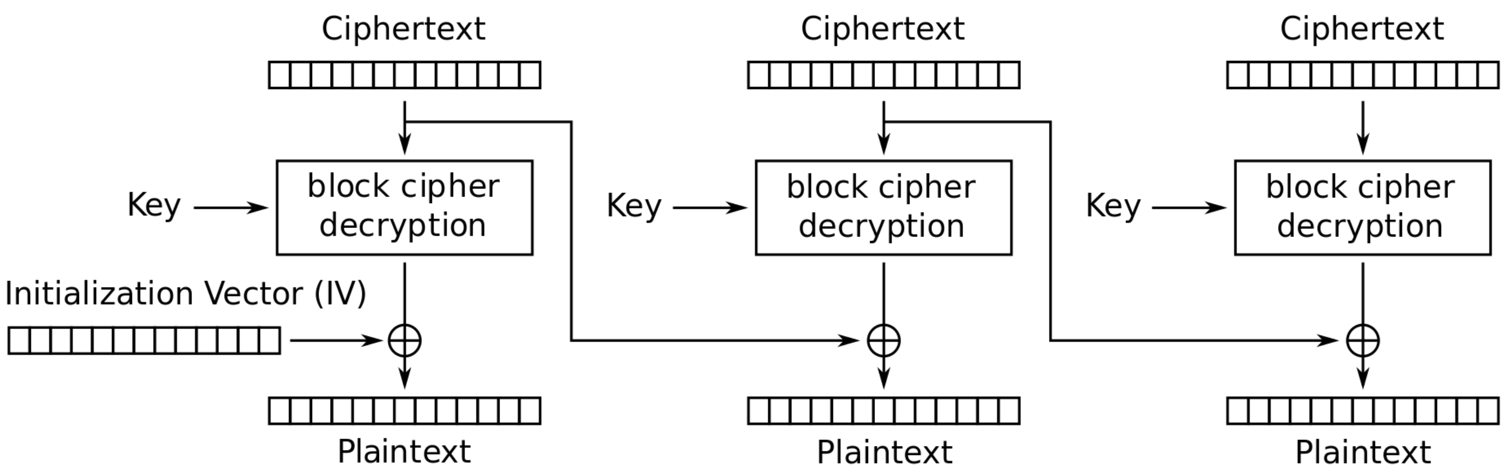
Electronic Codebook (ECB) mode decryption

ECB encrypts identical plaintext blocks into identical ciphertext blocks

AES: Cipher Block Chaining (CBC) Mode



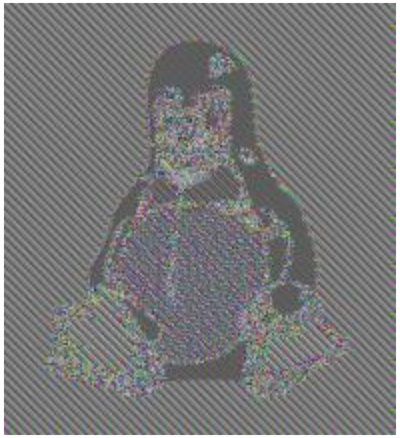
Cipher Block Chaining (CBC) mode encryption



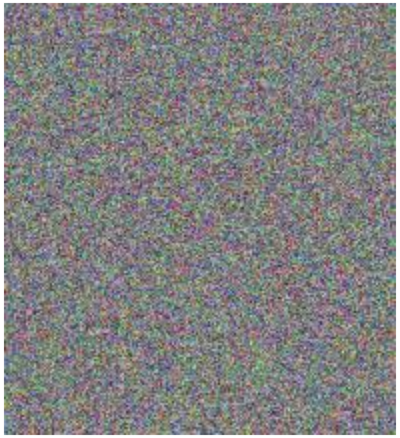
Cipher Block Chaining (CBC) mode decryption

AES, in pictures!

Original image



Encrypted using ECB mode



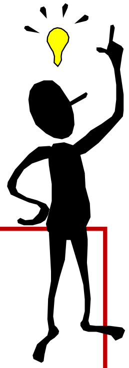
Modes other than ECB like CBC result in pseudo-randomness

Source: Wikipedia

Public Key Cryptography

symmetric key crypto

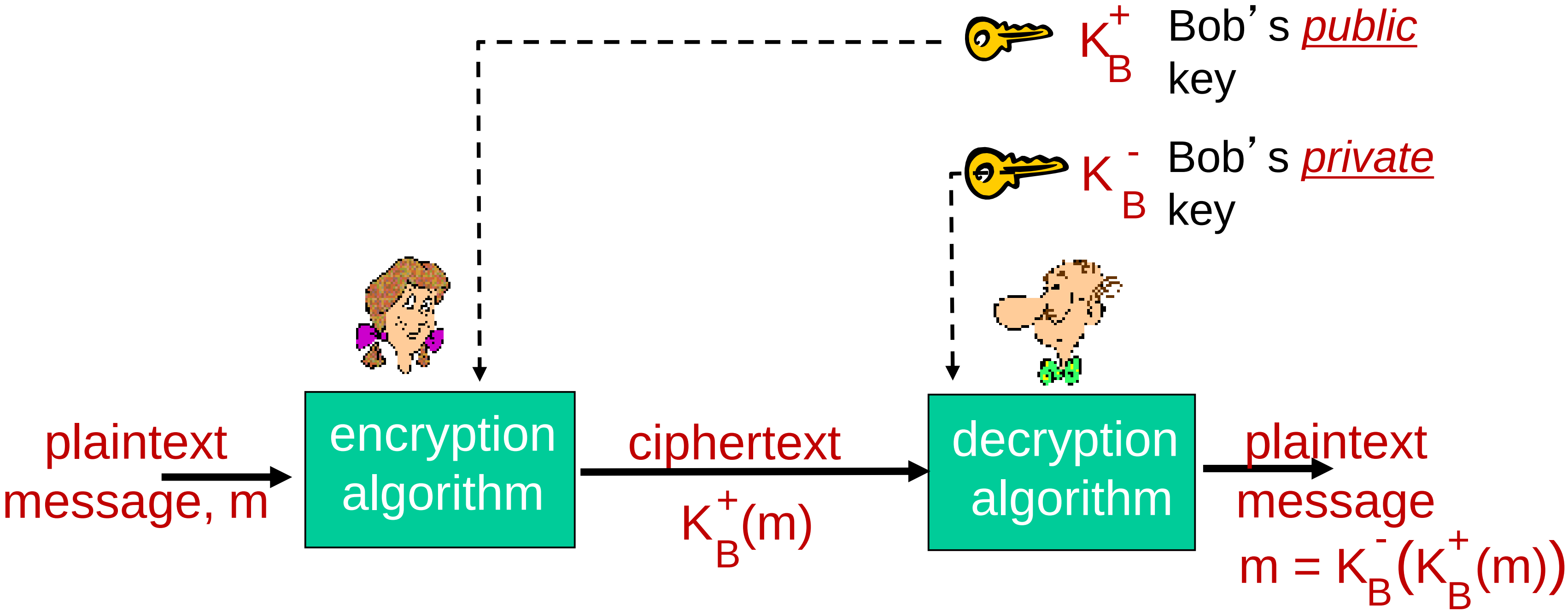
- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never "met")?



public key crypto

- radically different approach [Diffie-Hellman76, RSA78]
- sender, receiver do *not* share secret key
- *public* encryption key known to *all*
- *private* decryption key known only to receiver

Public Key Cryptography



Wow - public key cryptography revolutionized 2000-year-old (previously only symmetric key) cryptography!

- similar ideas emerged at roughly same time, independently in US and UK (classified)

Public key encryption algorithms

requirements:

1. need $K_B^+(\cdot)$ and $K_B^-(\cdot)$ such that

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

2. given public key K_B^+, it should be impossible to compute private key K_B^-

RSA: Rivest, Shamir, Adelson algorithm

RSA: Creating public/private key pair

1. choose two large prime numbers p, q. (e.g., 1024 bits each)
2. compute $n = pq$, $z = (p-1)(q-1)$
3. choose e (with $e<n$) that has no common factors with z (e, z are "relatively prime").
4. choose d such that $ed-1$ is exactly divisible by z. (in other words: $ed \bmod z = 1$).
5. *public* key is $\underbrace{(n,e)}_{K_B^+}$. *private* key is $\underbrace{(n,d)}_{K_B^-}$.

RSA: encryption, decryption

0. given (n,e) and (n,d) as computed above

1. to encrypt message m $(<n)$, compute

   $c = m^e \bmod n$

2. to decrypt received bit pattern, c, compute

   $m = c^d \bmod n$

magic happens! $m = (\underbrace{m^e \bmod n}_{c})^d \bmod n$

RSA example:

Bob chooses *p=5, q=7*. Then *n=35, z=24*.

e=5 (so *e, z* relatively prime).

d=29 (so *ed-1* exactly divisible by z).

encrypting 8-bit messages.

encrypt: $\underbrace{\text{bit pattern}}_{\text{0000II00}} \quad \underbrace{m}_{12} \quad \underbrace{m^e}_{248832} \quad \underbrace{c = m^e \bmod n}_{17}$

decrypt: $\underbrace{c}_{17} \quad \underbrace{c^d}_{481968572106750915091411825223071697} \quad \underbrace{m = c^d \bmod n}_{12}$

http://fringe.davesource.com/Fringe/Crypt/RSA/Example.html

Why is RSA secure?

- suppose you know Bob's public key (n,e). How hard is it to determine d in his private key (n,d)?
- essentially need to find factors of n without knowing the two factors p and q
  - n=p*q is easy to calculate but hard to reverse
    - fact: factoring a large number is really hard
    - So, uses relatively large keys (1024 bits) and relies on the high computational cost of factoring large numbers

RSA in practice: session keys

- exponentiation in RSA is computationally intensive
- DES (sym key crypto) is at least 100 times faster than RSA (asym key crypto)
- use public key cryto to establish secure connection, then establish second key – symmetric session key – for encrypting application data faster

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

Comparison

Symmetric Key Crypto

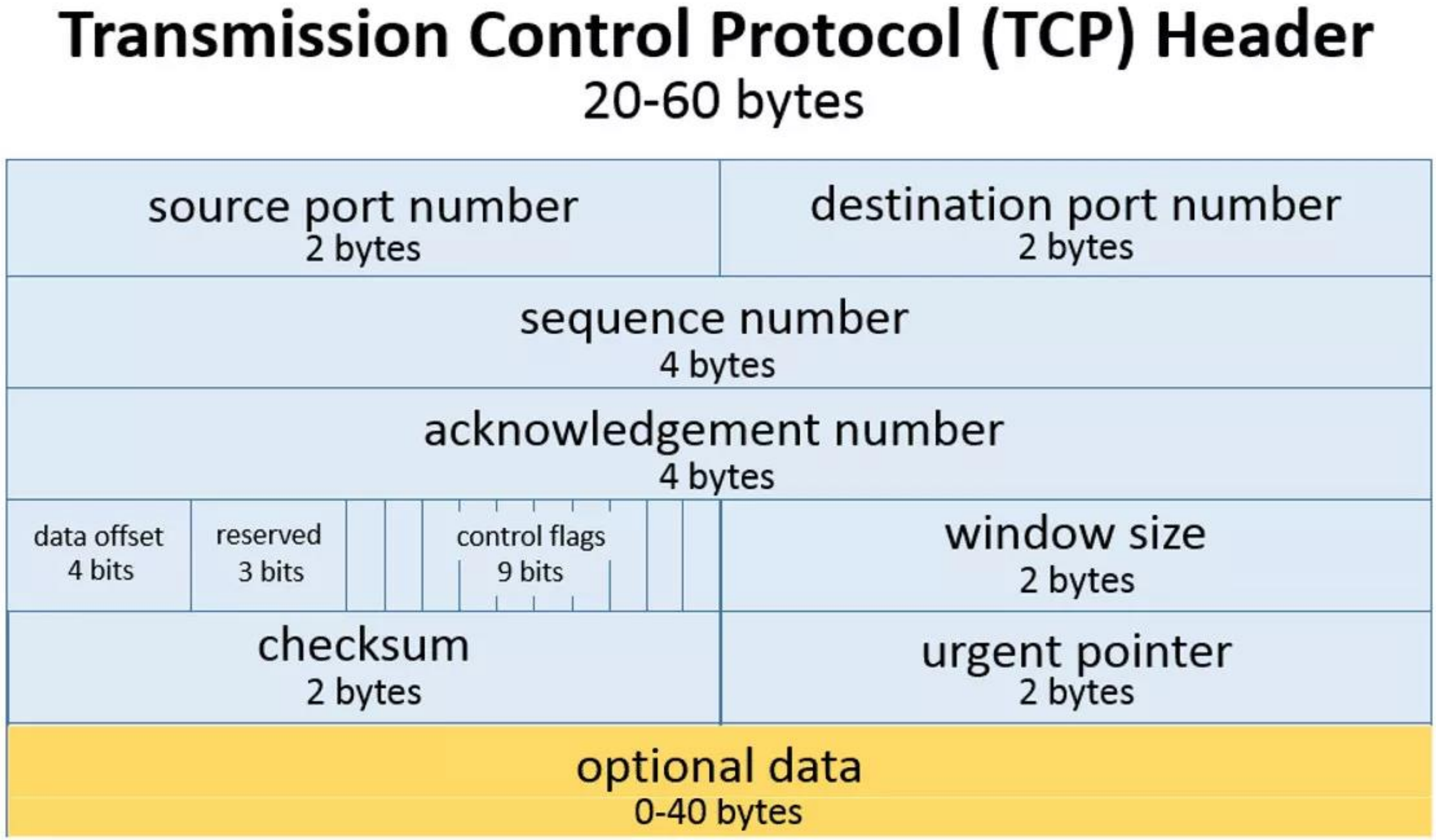
- Single (secret) key for encryption and decryption
- For communication among *n* people, it needs $n*(n-1)/2$ secret keys
  - Each person has to maintain n-1 keys secretly
- DES, 3DES, AES, RC4
- Encryption process is very fast
  - Large data transfer
- Provides only confidentiality, authenticity and integrity

Asymmetric/Public Key Crypto

- Public key for encryption and private key for decryption
- It needs only *n number of* <*public, private*> key pairs
  - Each person has to maintain only one private key secretly
- RSA, DSA, ECC, Diffie-Hellman, El Gamal
- Encryption process is slow
  - Small data transfer
- Provides confidentiality, authenticity, integrity, and non-repudiation

Integrity Check

- Checksum (CRC)
- Message Authentication Code (MAC) aka Message Integrity Code (MIC)
- Digital Signatures



Source: Lifewire

Ethernet and Wi-Fi Frame Formats

Ethernet (802.3) Frame Format

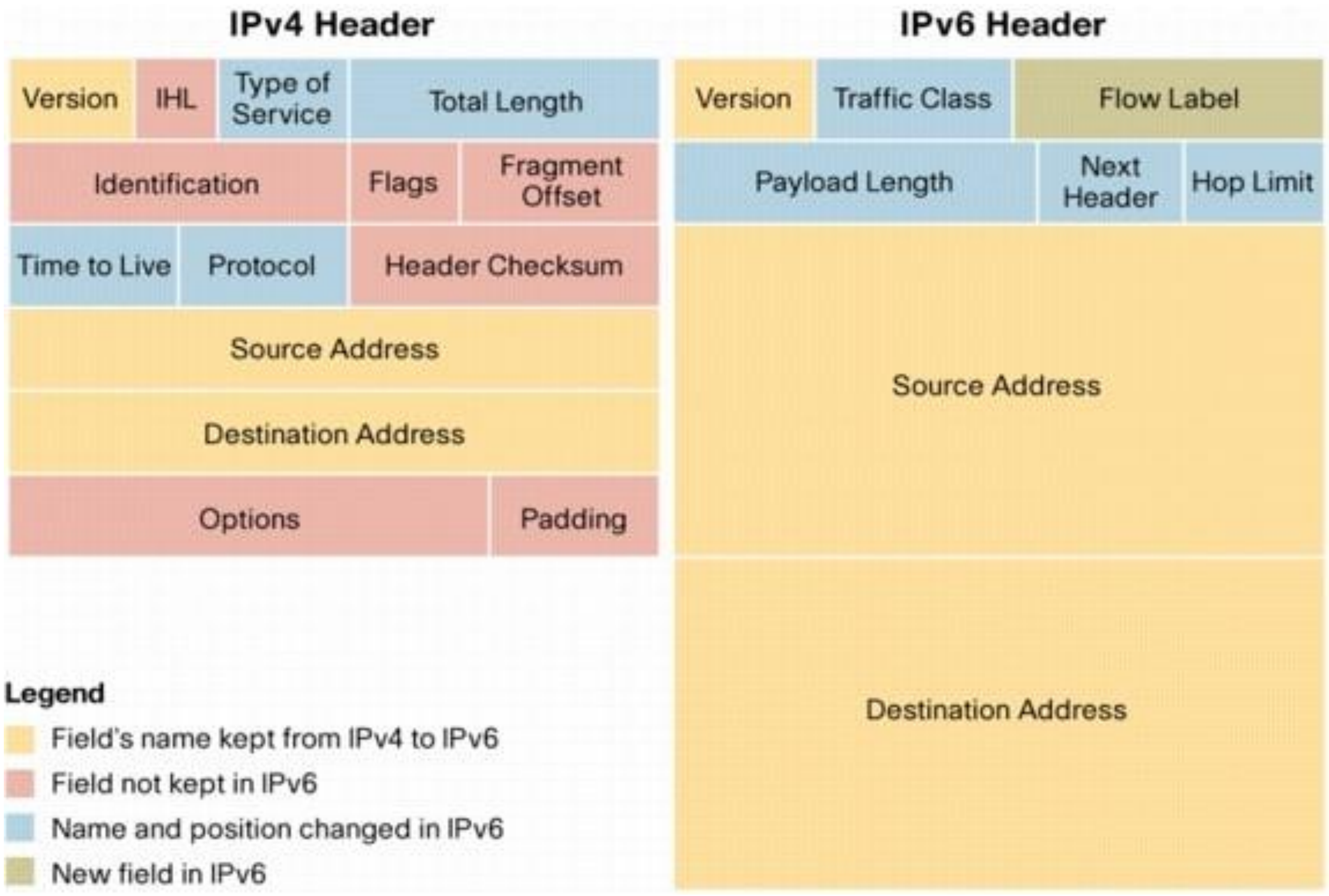
7 bytes	1 byte	6 bytes	6 bytes	2 bytes	42 to 1500 bytes	4 bytes	12 bytes
Preamble	Start of Frame Delimiter	**Destination MAC Address**	**Source MAC Address**	Type	Data (payload)	CRC	Inter-frame gap

For TCP/IP communications, the payload for a frame is a packet

WiFi (802.11) Frame Format

2 bytes	2 bytes	6 bytes	6 bytes	6 bytes	2 bytes	6 bytes	0 to 2312 bytes	4 bytes
Frame Control	Duration	MAC Address 1 (Destination)	MAC Address 2 (Source)	MAC Address 3 (Router)	Seq Control	MAC Address 4 (AP)	Data (payload)	CRC

IPv4 and IPv6 Headers



Source: Cisco

Internet checksum: poor crypto hash function

Internet checksum has some properties of hash function:

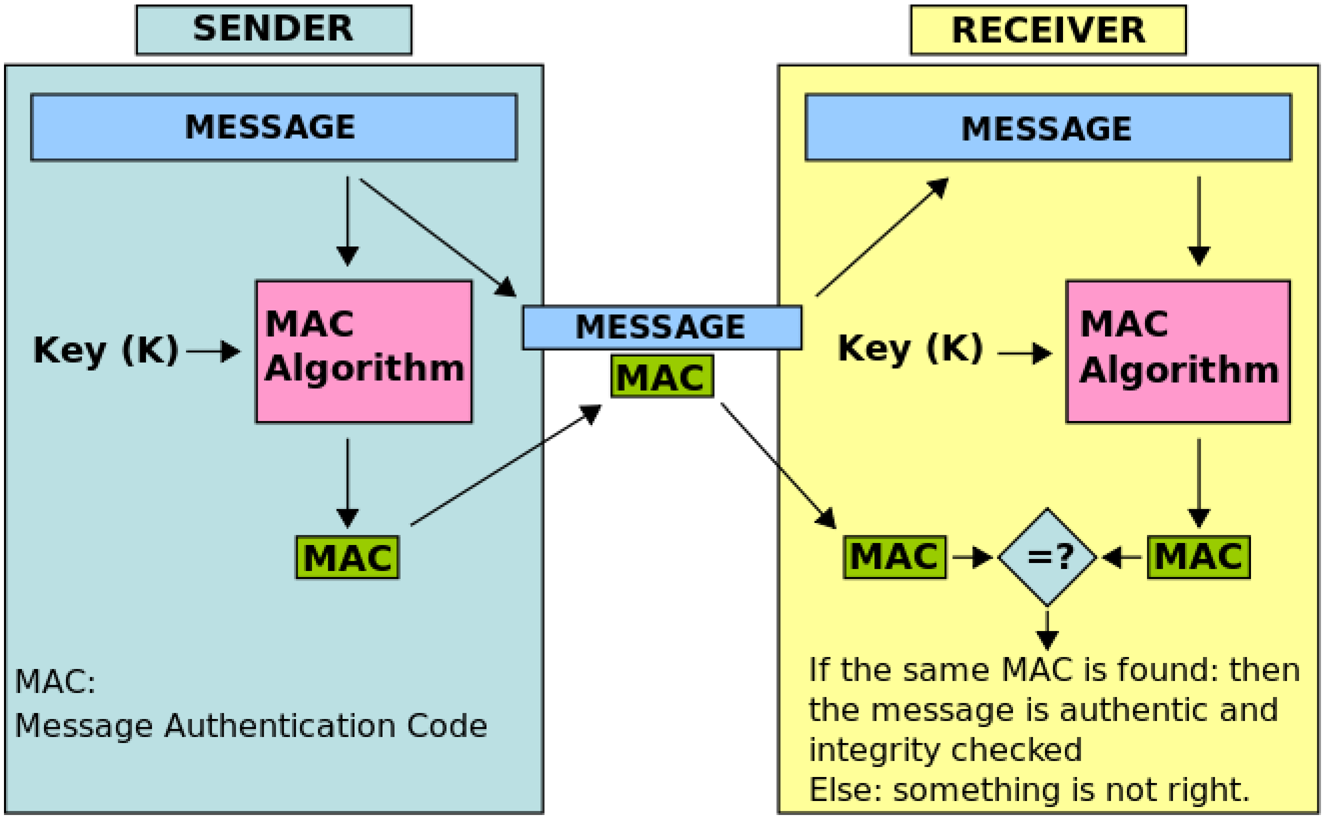
- produces fixed length digest (e.g., 16-bit checksums in IPv4 and TCP/UDP Headers) of message for error checking. Ethernet/Wi-Fi use 32-bit checksums
- is many-to-one
- but given message with given hash value, it is easy to find another message with same hash value:

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!

Message Authentication Code (MAC)



- Keyed hashing using symmetric keys
- Single key for generation of MAC (of fixed length) and its verification
- MAC/MIC protects a message's data integrity and its authenticity
- But does not offer non-repudiation

Source: Wikipedia

RSA: another important property

The following property is *very* useful for Digital Signatures:

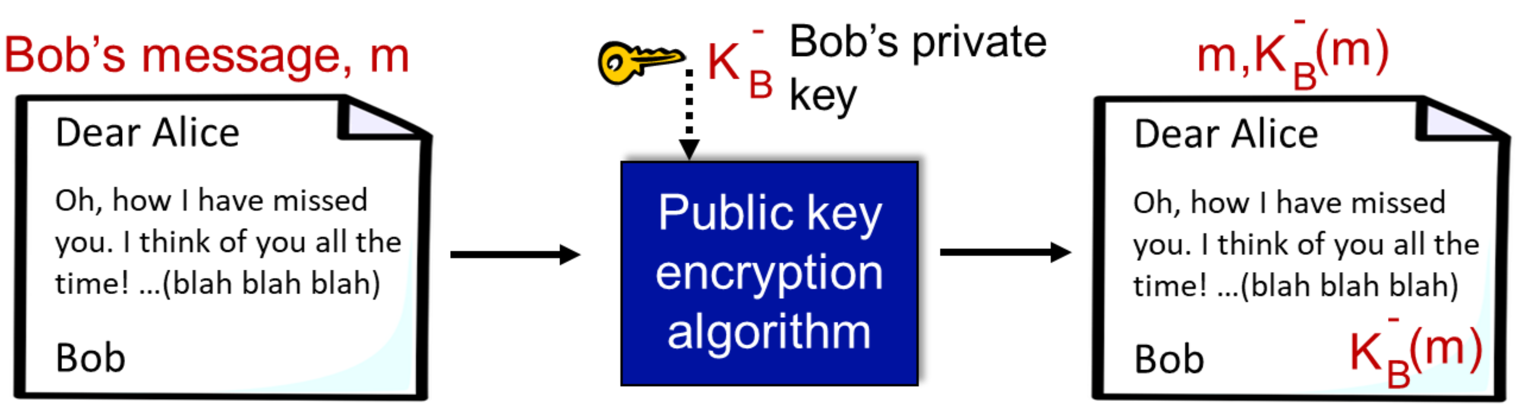
$$\underbrace{K_B^-(K_B^+(m))}_{\substack{\text{use public key first,}\\ \text{followed by}\\ \text{private key}}} = m = \underbrace{K_B^+(K_B^-(m))}_{\substack{\text{use private key}\\ \text{first, followed by}\\ \text{public key}}}$$

result is the same!

Digital signatures

Analogous to hand-written signatures:

- sender (Bob) digitally signs document: 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
- 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 $K^+_B(K_B^-(m)) = m$.
- $K^+_B(K_B^-(m)) = m$, whoever signed m must have used Bob's private key

Alice thus verifies that:

- Bob signed m, no one else signed m (Authenticity)
- Bob signed m and not m' (integrity)

\+ non-repudiation:

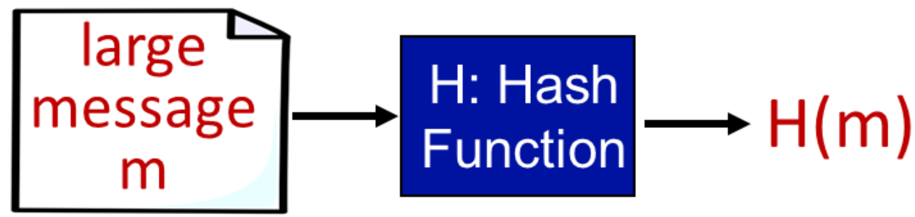
✓ Alice can take m, and signature $K_B^-(m)$ to court and prove that Bob signed m

Message digests

computationally expensive to public-key-encrypt (i.e., digitally sign) long messages

goal: fixed-length, easy-to-compute digital "fingerprint"

- apply hash function H to *m*, get fixed size message digest, *H(m)*



Secure/Crypto Hash function properties:

- one-way function (no keys) unlike encryption which is 2-way with keys
- produces fixed-size msg digest (fingerprint) using m: many-to-one
- *given message digest x, computationally infeasible to find preimage m such that x = H(m)*
- computationally infeasible to find 2nd preimage and collisions

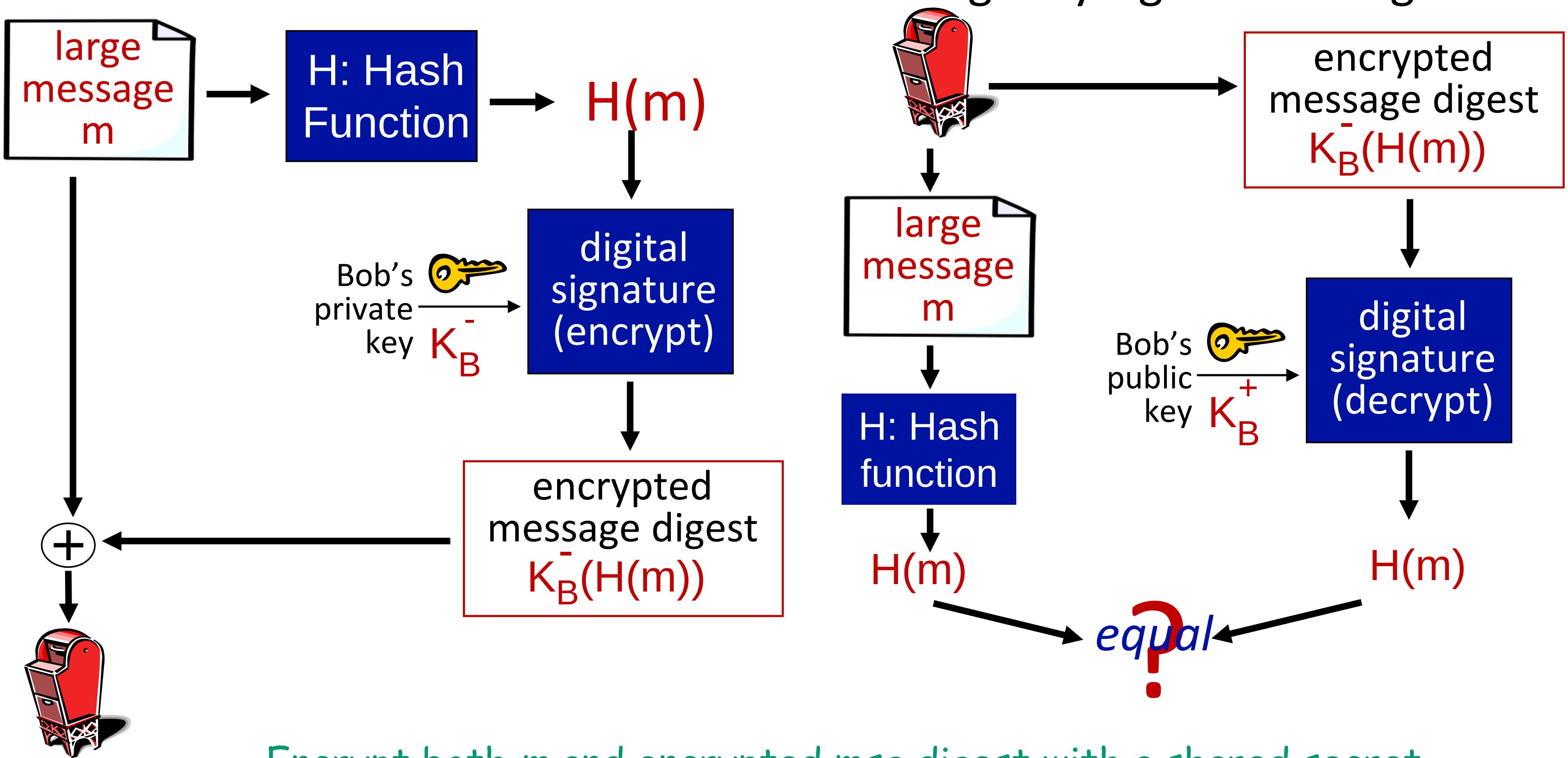
Secure hash function algorithms

- MD5 hash function widely used (RFC 1321)
  - computes 128-bit message digest in 4-step process.
  - arbitrary 128-bit string x, appears difficult to construct msg m whose MD5 hash is equal to x
  - Found to suffer from extensive vulnerabilities in 2005: collision attacks to find collisions in seconds on Pentium PC
- SHA-2: Secure Hash Algorithm 2
  - Designed by NSA
  - Do not use older, obsolete SHA-1 (2017)
  - Output (digest) length: 224, 256, 384, or 512 bits
  - Recommended for use today for password hashing, digital signatures, etc
  - SHA-3 is out, but yet deployed

Digital signature = signed message digest

Bob sends digitally signed message:

Alice verifies signature, integrity of digitally signed message:



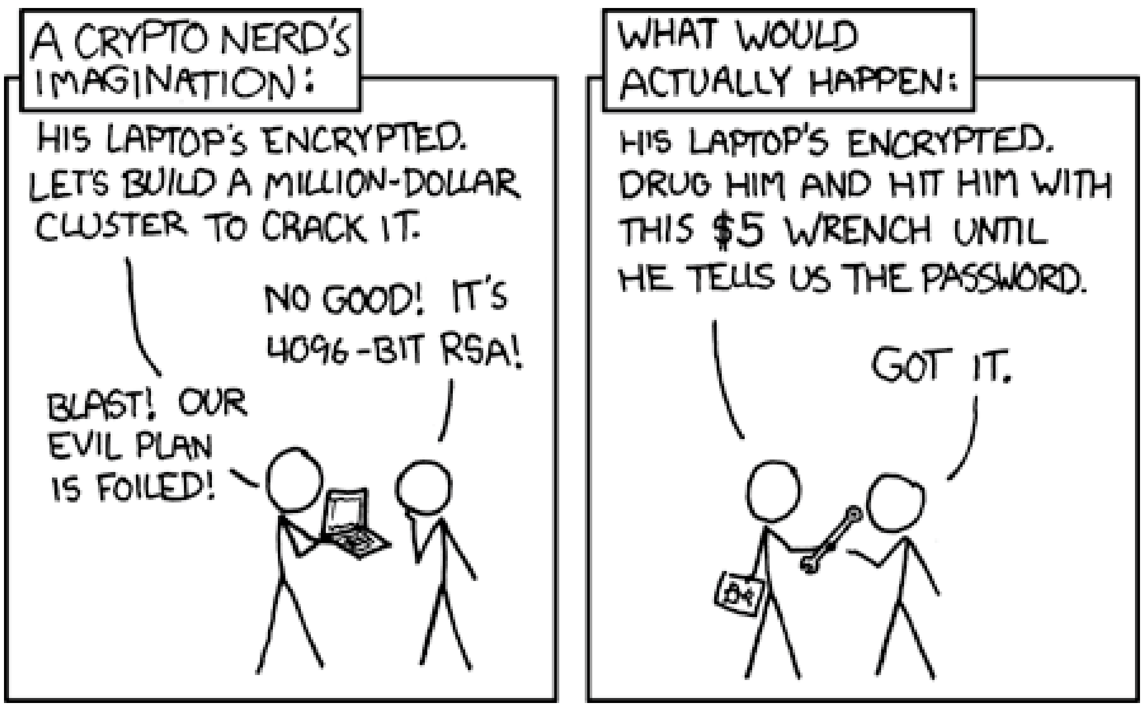
Encrypt both *m* and encrypted msg digest with a shared secret key (of a symmetric-key cipher) to also achieve confidentiality

Limitations of cryptography

Cryptography works when used correctly !!

… but is not the solution to all security problems

“Crypto will not be broken. It will be bypassed.” – Adi Shamir



XKCD 538

References

- https://book.systemsapproach.org/security/crypto.html
- Crypto book: http://web.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf
- Cryptographic Standards and Guidelines | CSRC (nist.gov)
- Latacora - Cryptographic Right Answers
- A Few Thoughts on Cryptographic Engineering by Matthew Green: https://blog.cryptographyengineering.com/