# Network Security

## Public Key Cryptography

# Public Key Cryptography

*Agenda:*

- **Message authentication** – authentication codes and hash functions
- **Public key encryption** – principles and algorithms
- **Exchange** of conventional keys
- **Digital signatures**
- Revisit **key management**

# Recall Security Services

- **Confidentiality** – protection from passive attacks
- **Authentication** – you are who you say you are
- **Integrity** – received as sent, no modifications, insertions, shuffling or replays

# Security Attacks

**Passive threats**

Release of message contents

Traffic analysis

- eavesdropping, monitoring transmissions
- conventional encryption helped here

# Security Attacks

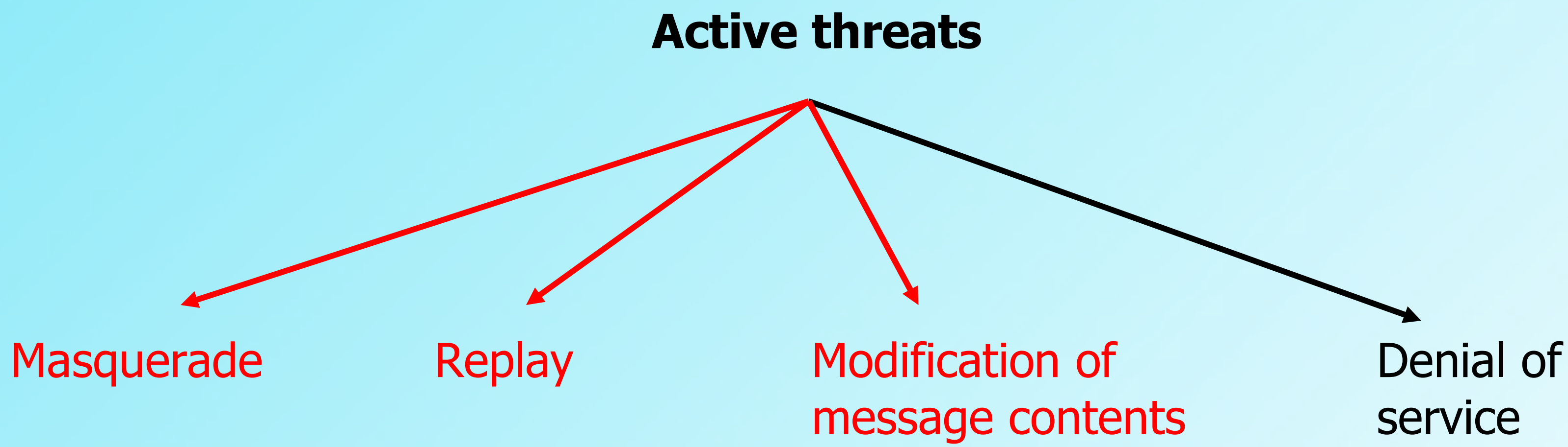
NEW YORKER


"On the Internet, nobody knows you're a dog."

**On the Internet, nobody knows you’re a dog**

- by Peter Steiner, New York, July 5, 1993

# Security Attacks



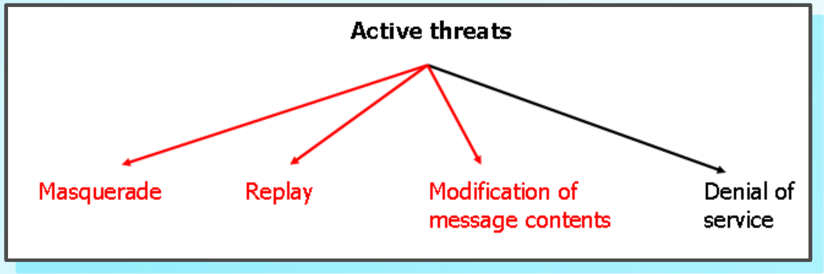

- Message authentication helps prevents these!

# What Is Message Authentication

- It's the "source," of course!
- Procedure that allows communicating parties to verify that received messages are authentic
- Characteristics:
  - source is authentic – *masquerading*
  - contents unaltered – *message modification*
  - timely sequencing – *replay*

# Can We Use Conventional Encryption?

- Only sender and receiver share a key
- Include a time stamp
- Include error detection code and sequence number



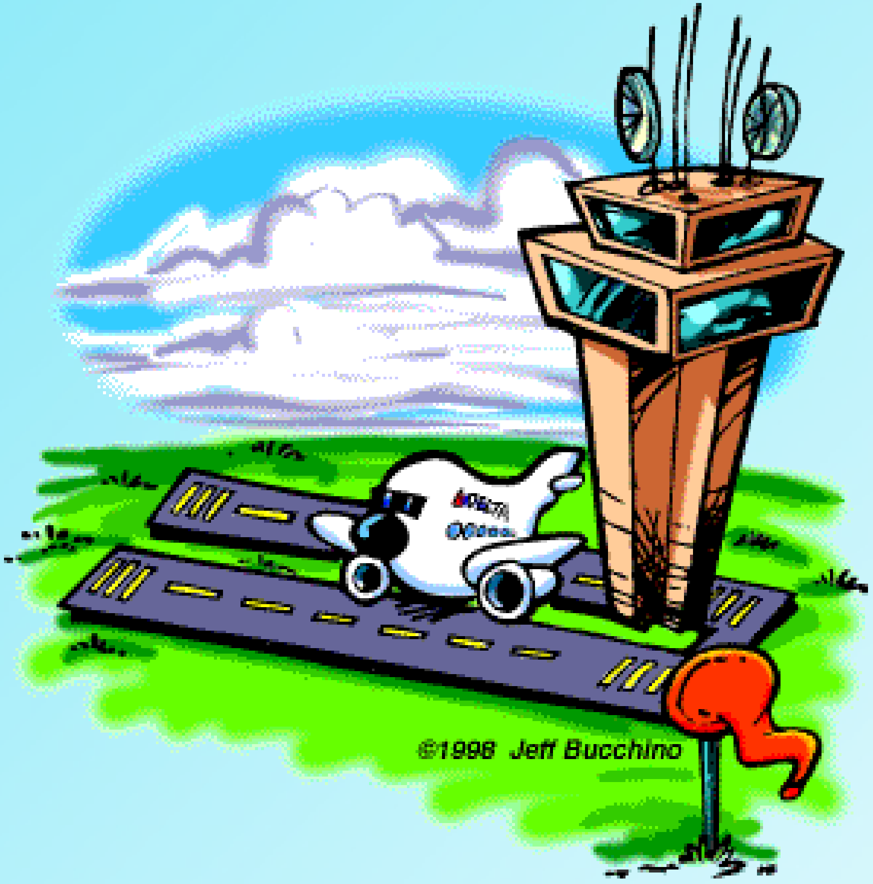
# Message Authentication Sans Encryption

- Append an authentication tag to a message
- Message read independent of authentication function
- No message confidentiality

# Message Authentication w/o Confidentiality

- *Application that broadcasts a message* – only one destination needs to monitor for authentication
- *Too heavy a load to decrypt* – random authentication checking
- *Computer executables and files* – checked when assurance required

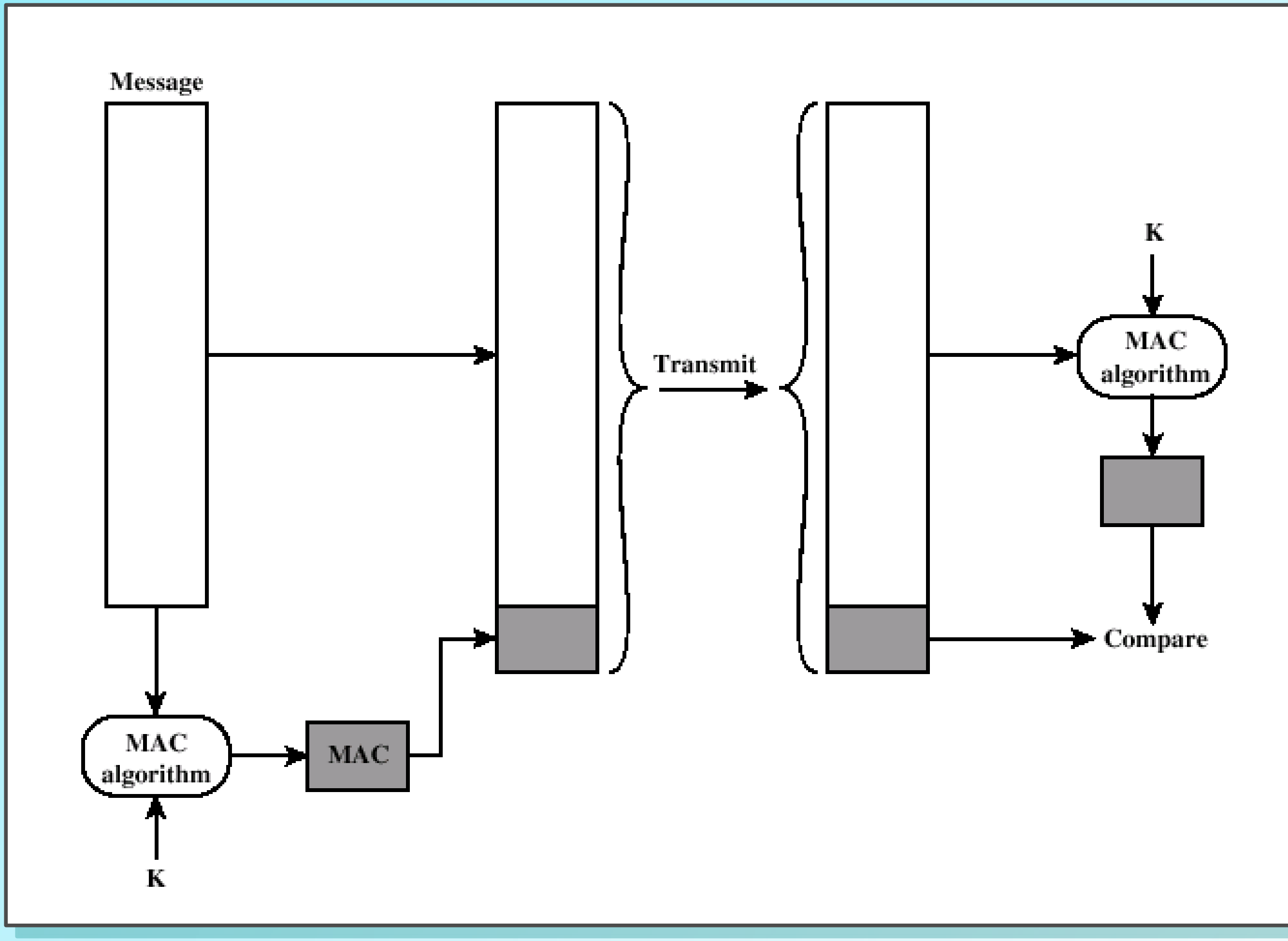
# Life Without Authentication




# Message Authentication Code

- Message Authentication Code (MAC) – use a secret key to generate a small block of data that is *appended* to the message
- *Assume*: A and B share a common secret key $K_{AB}$
- $MAC_M = F(K_{AB}, M)$

# Message Authentication Code



# Message Authentication Code

- Receiver assured that message is not altered – no modification
- Receiver assured that the message is from the alleged sender – no masquerading
- Include a sequence number, assured proper sequence – no replay

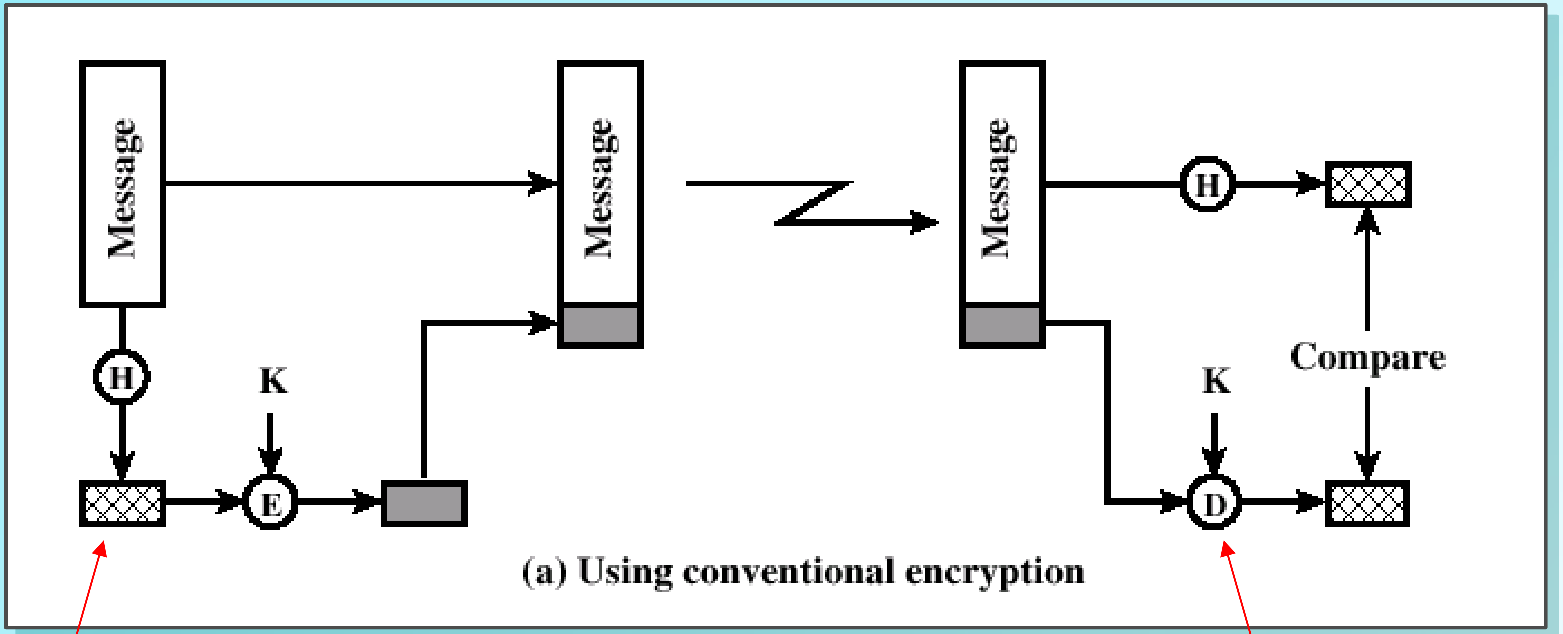
# Message Authentication Code

- DES is used
- Need not be reversible
- Checksum
- Stands up to attack
- But there is an alternative...

# One Way Hash Function

- Hash function accepts a *variable* size message $M$ as input and produces a *fixed-size* message digest $H(M)$ as output
- No secret key as input
- Message digest is sent with the message for authentication
- Produces a fingerprint of the message

# One Way Hash Function

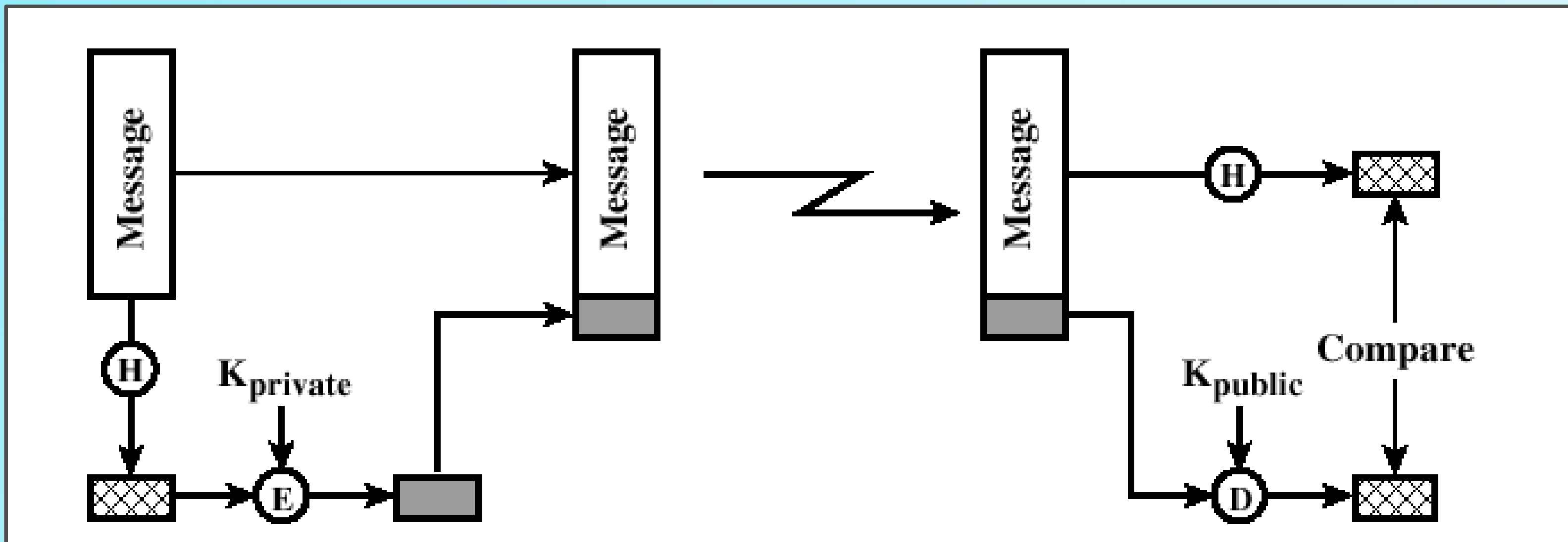



Message digest H(M)

Shared key

Authenticity is assured

# One Way Hash Function




(b) Using public-key encryption

Digital signature

No key distribution

Less computation since message does not have to be encrypted

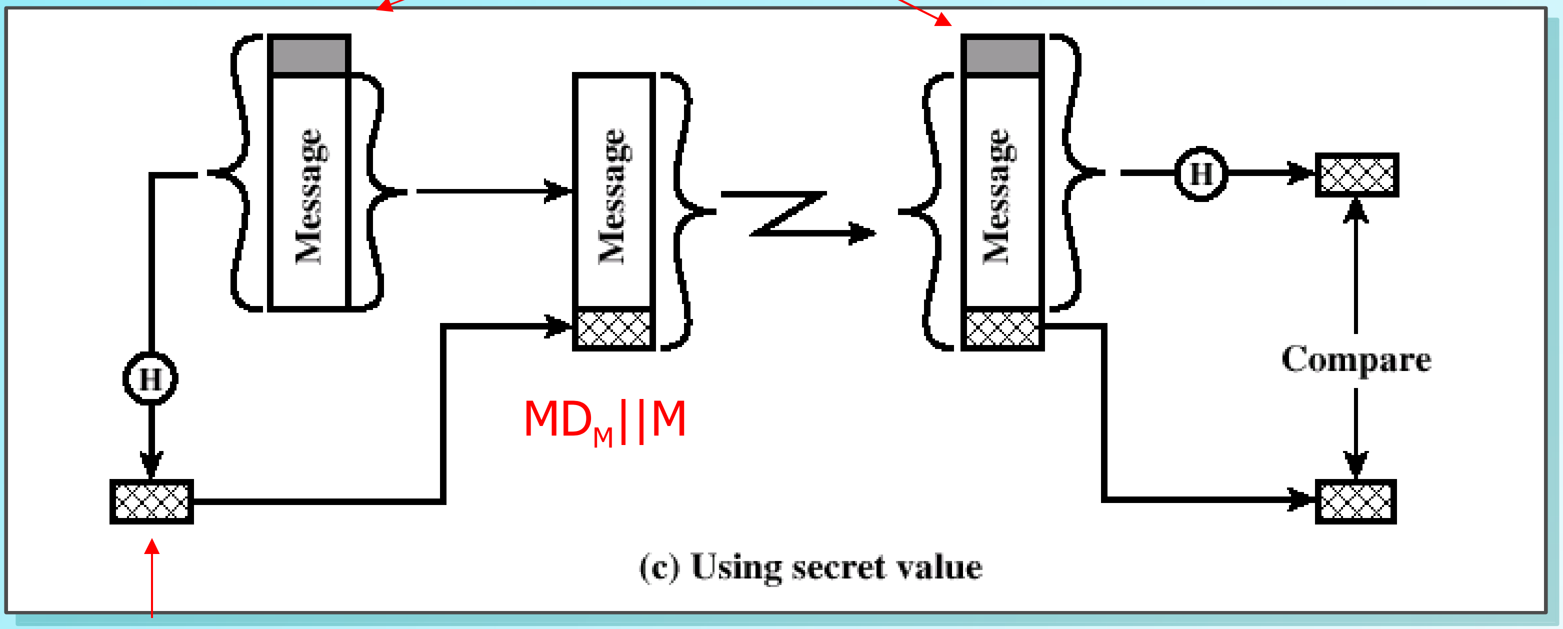
# One Way Hash Function

*Ideally We Would Like To Avoid Encryption*

- Encryption software is slow
- Encryption hardware costs aren't cheap
- Hardware optimized toward large data sizes
- Algorithms covered by patents
- Algorithms subject to export control

# One Way Hash Function

Assumes secret value $S_{AB}$




$MD_M = H(S_{AB}||M)$

No encryption for message authentication
Secret value never sent; can’t modify the message
Important technique for **Digital Signatures**

# Hash Function Requirements

weak

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

# Simple Hash Functions

- Input: sequence of $n$-bit block
- Processed: one block at a time producing an $n$-bit hash function
- Simplest: Bit-by-bit XOR of every block $C_i = b_{i1} \oplus b_{i2} \oplus \cdots \oplus b_{im}$
- Longitudinal redundancy check

# Bitwise XOR

| | 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** | $C_1$ | $C_2$ | | $C_n$ |

- Problem: Eliminate predictability of data
- One-bit circular shift for each block is used to randomize the input

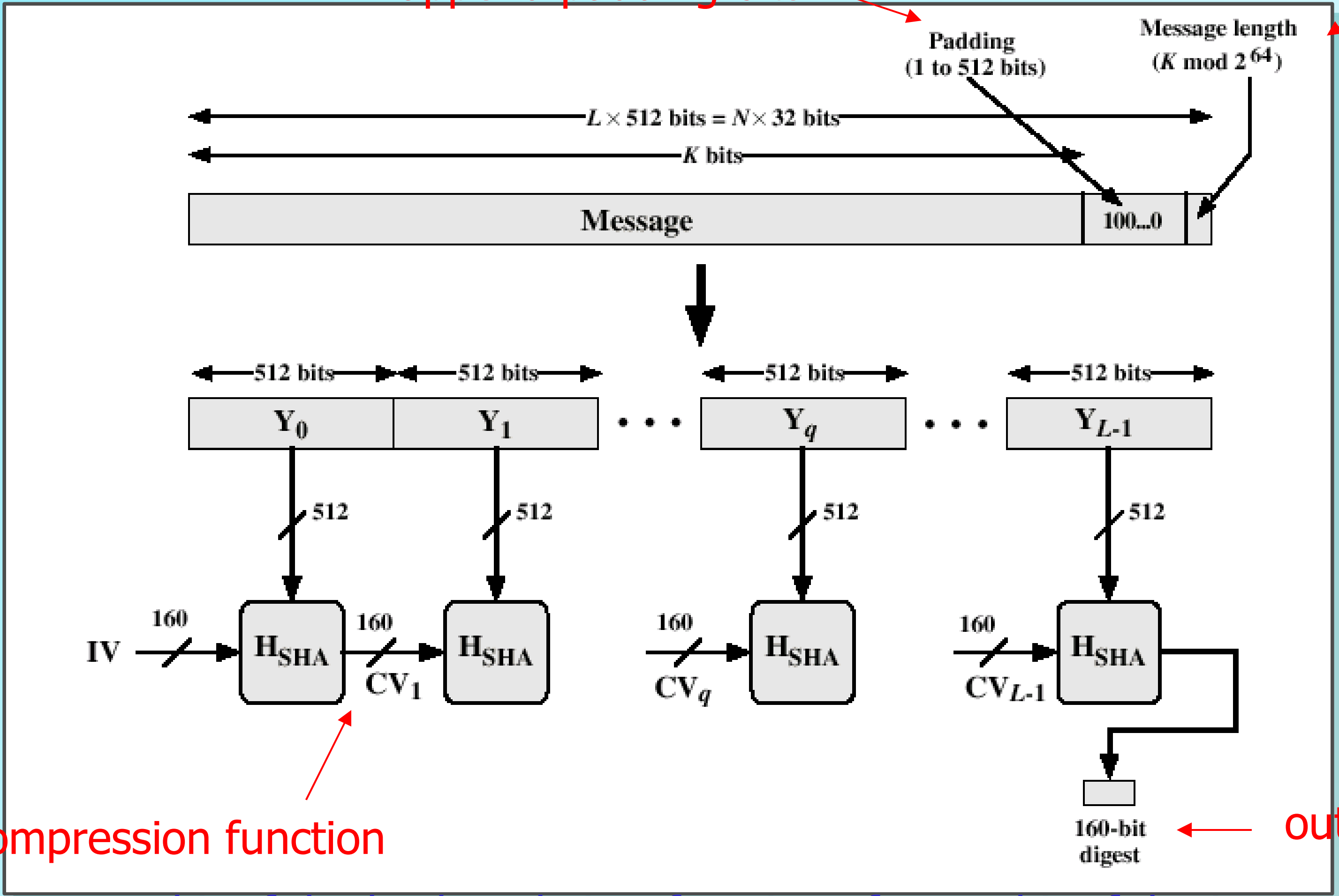
# SHA-1 Secure Hash Function

- Developed by NIST in 1995
- Input is processed in 512-bit blocks
- Produces as output a 160-bit message digest
- *Every bit of the hash code is a function of every bit of the input*
- Very secure – so far!

# SHA-1 Secure Hash Function

append padding bits

append length

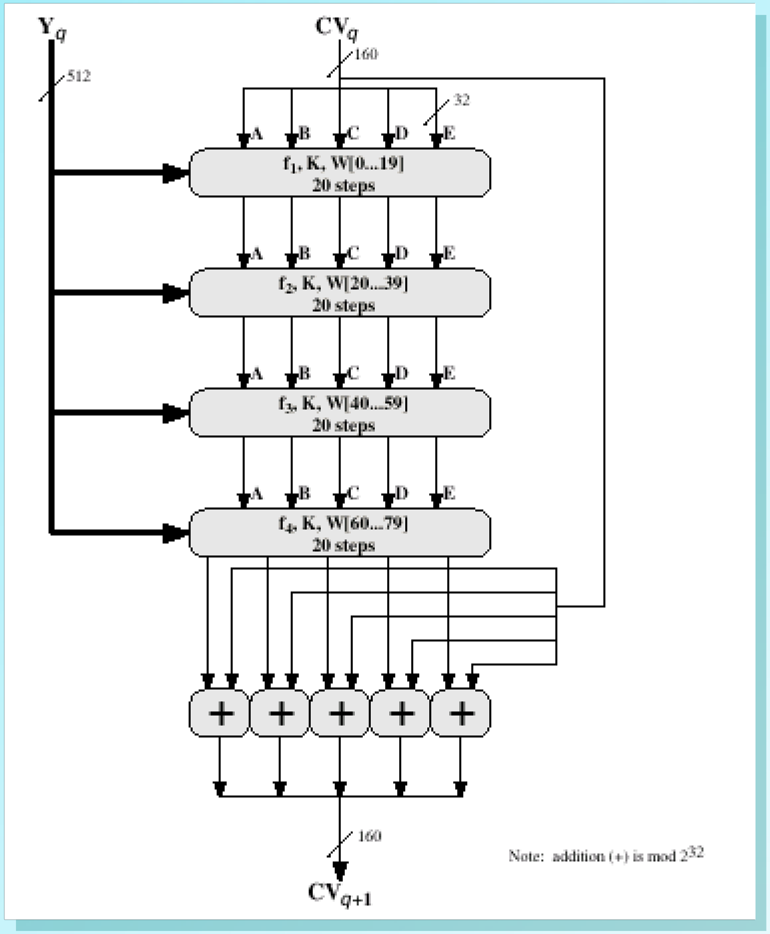



compression function

output

*Every bit of the hash code is a function of every bit of the input!*

# SHA-1 Secure Hash Function



# Other Hash Functions

- Most follow basic structure of SHA-1
- This is also called an iterated hash function – Ralph Merkle 1979
- *If the compression function is collision resistant, then so is the resultant iterated hash function*
- Newer designs simply refine this structure

# MD5 Message Digest

- Ron Rivest - 1992
- RFC 1321
- Input: arbitrary  Output: 128-bit digest
- Most widely used secure hash algorithm – until recently
- Security of 128-bit hash code has become questionable (1996, 2004)

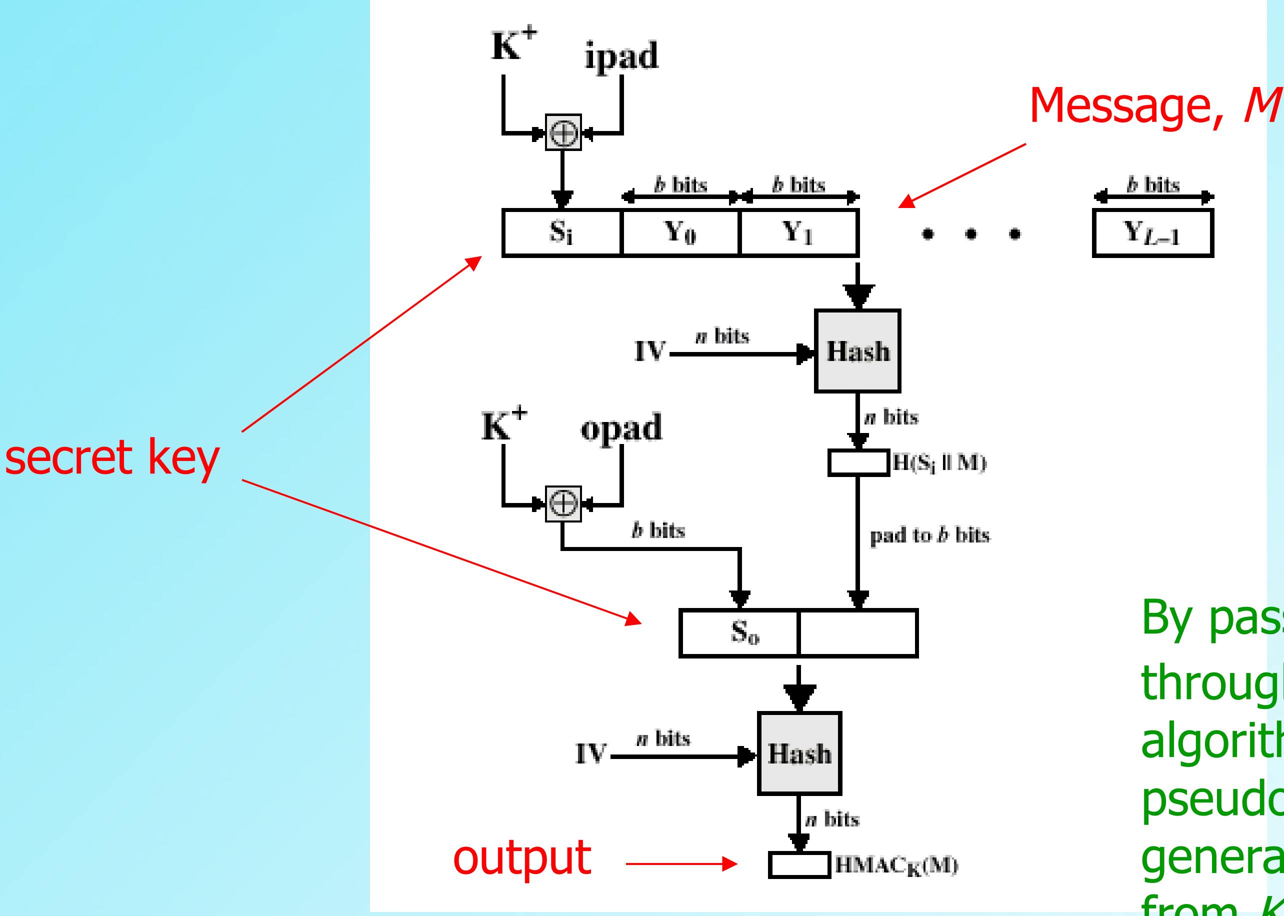
# RIPEMD-160

- European RIPE Project – 1997
- Same group launched an attack on MD5
- Extended from 128 to 160-bit message digest

# HMAC

- Effort to develop a MAC derived from a cryptographic hash code
- Executes faster in software
- No export restrictions
- Relies on a secret key
- RFC 2104 list design objectives
- Used in Ipsec
- Simultaneously verify integrity and authenticity

# HMAC Structure



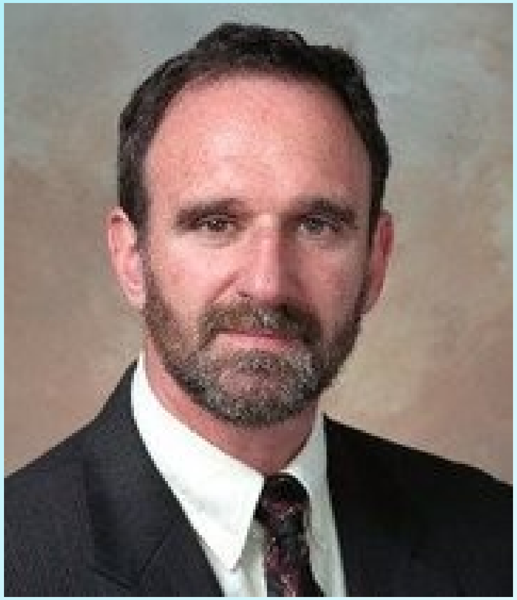

By passing $S_i$ and $S_o$ through the hash algorithm, we have pseudoradomly generated two keys from *K*.

# Public Key Encryption

- Diffie and Hellman – 1976
- First revolutionary advance in cryptography in thousands of years
- Based on mathematical functions not bit manipulation
- Asymmetric, two separate key
- Profound effect on confidentiality, key distribution and authentication

# Public Key Encryption

Whitfield Diffie



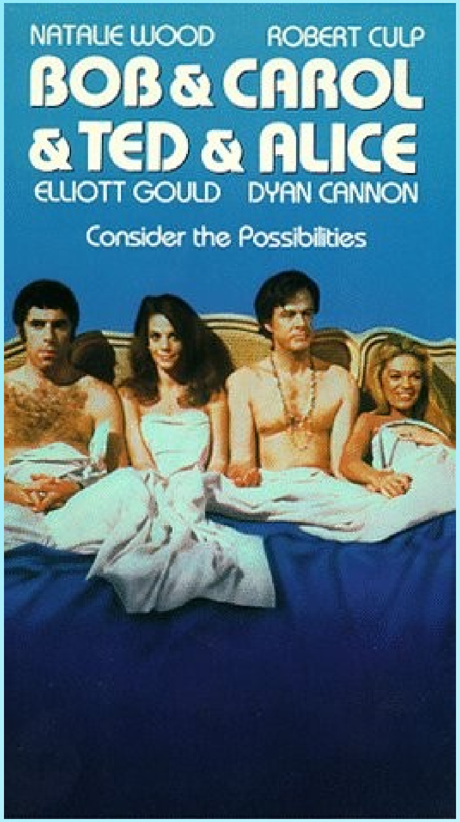
Martin Hellman

*Famous Paper:*
New Directions In Cryptography - 1976

# Public Key Structure

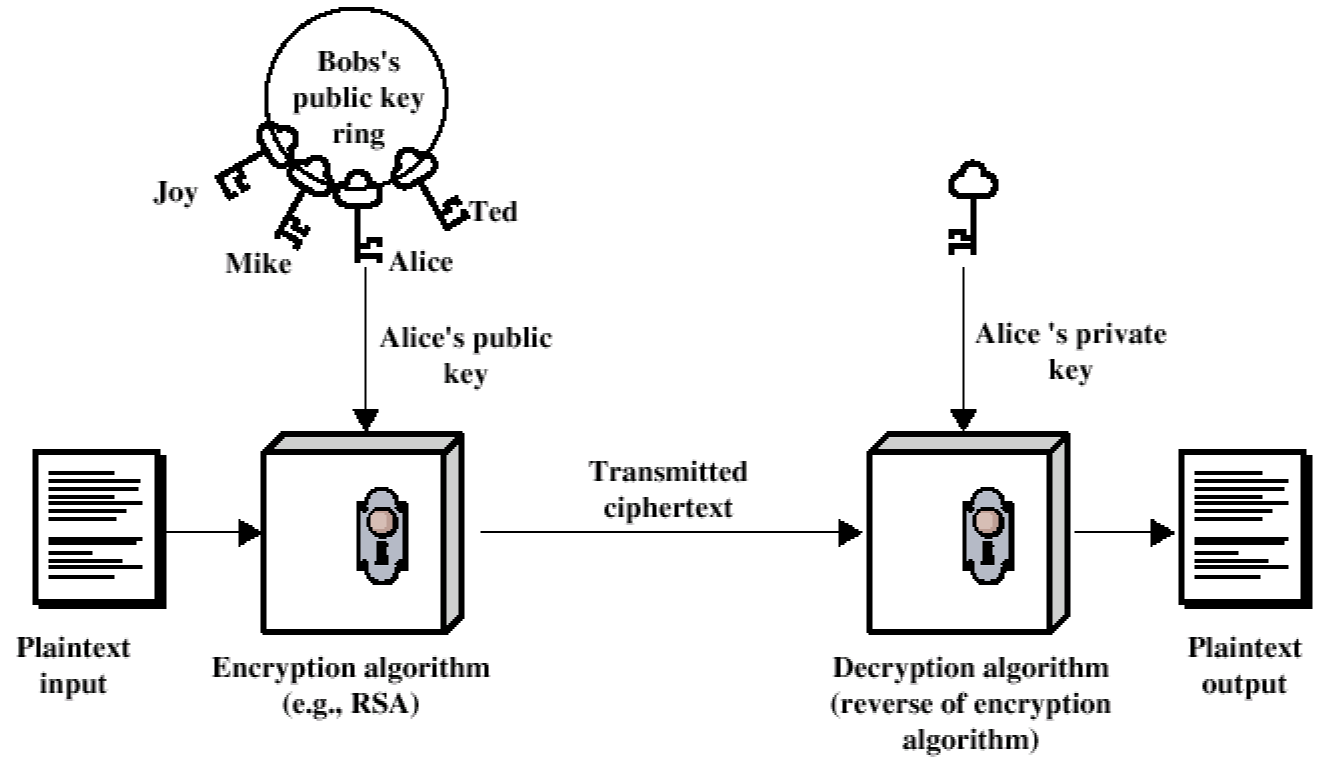
- Plaintext: message input into the algorithm
- Encryption algorithm: transformations on plaintext
- Public & Private Key: pair of keys, one for encryption; one for decryption
- Ciphertext: scrambled message
- Decryption algorithm: produces original plaintext

# Folklore




- 1969 Alternative Culture Film
- The names have stuck
- This is meaningless trivia!!!

# Public Key Encryption

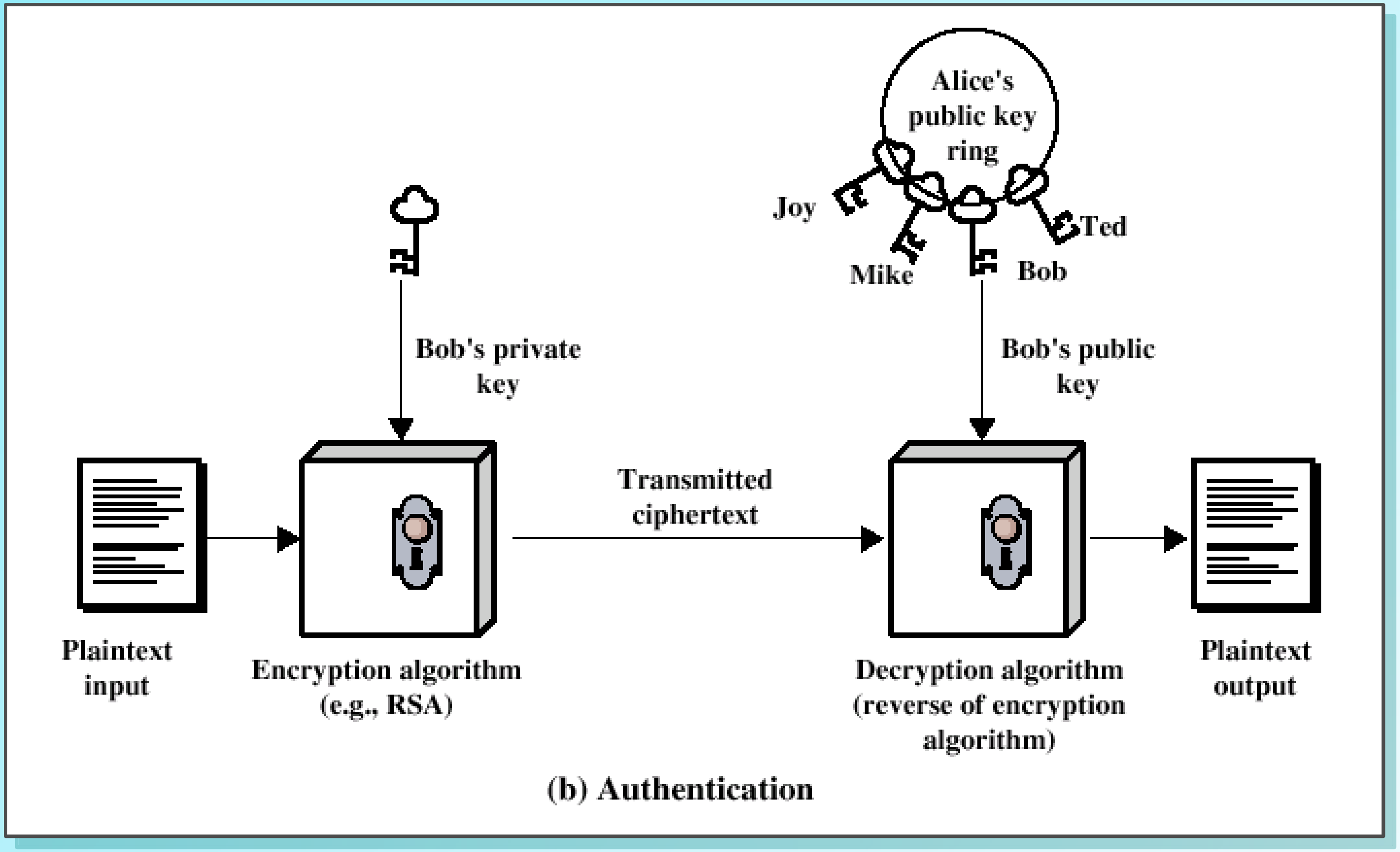



(a) Encryption

# The Basic Steps

- Each user generates a pair of keys
- The public key goes in a public register
- The private key is kept private
- If Bob wishes to send a private message to Alice, Bob *encrypts* the message using Alice's *public* key
- When Alice receives the message, she *decrypts* using her *private* key

# Public Key Authentication



# Public Key Applications

- Encryption/decryption – encrypts a message with the recipient's public key
- Digital signature – sender *signs* a message with private key
- Key Exchange – two sides cooperate to exchange a session key

# Requirements For Public Key

- Easy for party $B$ to generate pairs: public key $KU_b$ ; private key $KR_b$
- Easy for sender $A$ to generate cipertext using public key:

$$C = E_{KUb}(M)$$

- Easy for receiver $B$ to decrypt using the private key to recover original message

$$M = D_{KRb}(C) = D_{KRb}[E_{KUb}(M)]$$

*HINT:*
PUBLIC
PRIVATE

# Requirements For Public Key

- It is computationally infeasible for an opponent, knowing the public key $KUb$ to determine the private key $KR_b$
- It is computationally infeasible for an opponent, knowing the public key $KUb$ and a ciphertext, $C$, to recover the original message, $M$
- Either of the two related keys can be used for encryption, with the other used for decryption

$$M = D_{KRb}[E_{KUb}(M)] = D_{KUb}[E_{KRb}(M)]$$

# RSA Algorithm

- Ron Rivest, Adi Shamir, Len Adleman – 1978
- Most widely accepted and implemented approach to public key encryption
- Block cipher where $M$ and $C$ are integers between $0$ and $n-1$ for some $n$
- Following form:

$$C = M^e \bmod n$$

$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

# RSA Algorithm

- Sender and receiver know the values of $n$ and $e$, but only the receiver knows the value of $d$
- Public key: $KU = \{e,n\}$
- Private key: $KR = \{d,n\}$

# RSA Requirements

- It is possible to find values of $e$, $d$, $n$ such that $M^{ed} = M \bmod n$ for all $M<n$
- It is relatively easy to calculate $M^e$ and $C$ for all values of $M<n$
- It is infeasible to determine $d$ given $e$ and $n$

Here is the magic!

# RSA Algorithm

**Key Generation**

| | |
|---|---|
| Select $p, q$ | $p$ and $q$ both prime |
| Calculate $n = p \times q$ | |
| Calculate $\phi(n) = (p - 1)(q - 1)$ | |
| Select integer $e$ | $\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$ |
| Calculate $d$ | $d = e^{-1} \bmod \phi(n)$ |
| Public key | $KU = \{e, n\}$ |
| Private key | $KR = \{d, n\}$ |

# RSA Algorithm

**Encryption**

| | |
|---|---|
| Plaintext: | $M < n$ |
| Ciphertext: | $C = M^e \pmod{n}$ |

**Decryption**

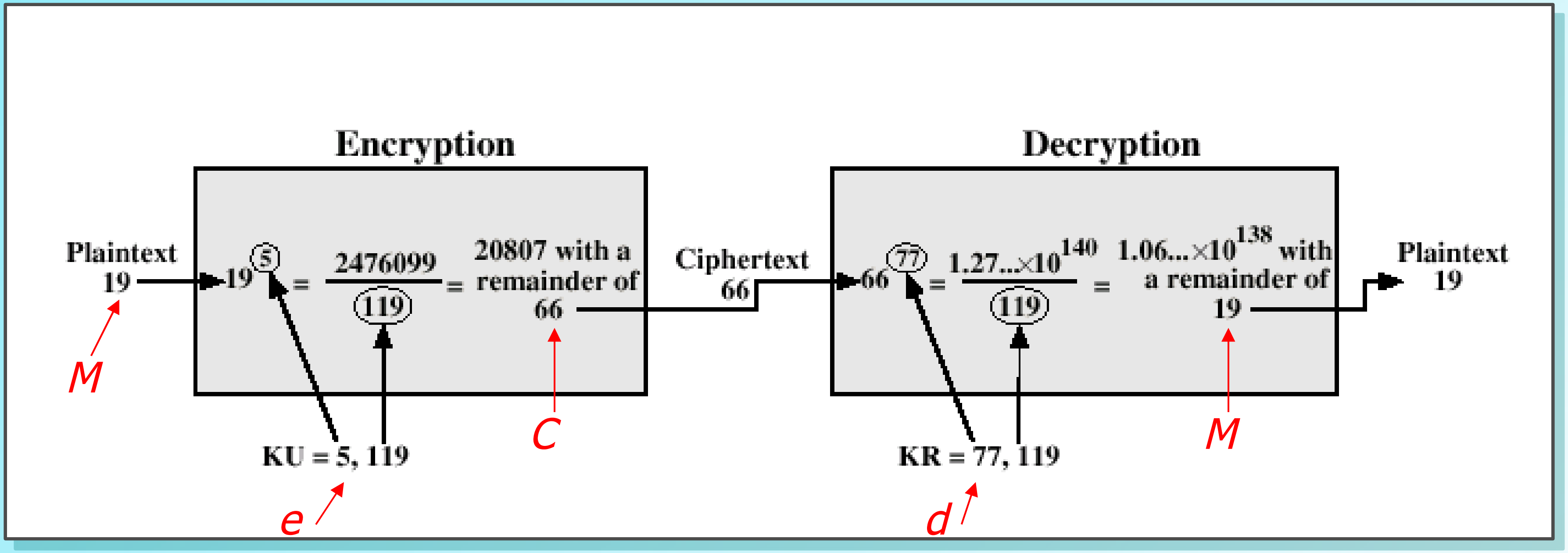
| | |
|---|---|
| Ciphertext: | $C$ |
| Plaintext: | $M = C^d \pmod{n}$ |

# RSA Example

- Select two prime numbers, $p=7$ and $q=17$
- Calculate $n = pq = 7 \times 17 = 119$ ← this is the modulus
- Calculate $\phi(n) = (p-1)(q-1) = 96$ ← Euler totient
- Select $e$ such that $e$ is relatively prime to $\phi(n) = 96$ and less than $\phi(n)$ ; in this case, $e= 5$
- Determine $d$ such that $de = 1 \bmod 96$ and $d<96$. The correct value is $d = 77$, because $77 \times 5 = 385 = 4 \times 96 + 1$

↑ multiplicative inverse of *e*

# RSA Example


Encryption
Decryption
Plaintext 19
19 5
2476099
119
20807 with a remainder of 66
Ciphertext 66
66 77
1.27...×10 140
119
1.06...×10 138 with a remainder of 19
Plaintext 19
KU = 5, 119
KR = 77, 119
M
C
M
e
d

# RSA Strength

- Brute force attack: try all possible keys – the larger $e$ and $d$ the more secure
- The larger the key, the slower the system
- For large $n$ with large prime factors, factoring is a hard problem
- Cracked in 1994 a 428 bit key; $100
- Currently 1024 key size is considered strong enough

# Diffie-Hellman Key Exchange

**Global Public Elements**

| | |
|---|---|
| $q$ | prime number |
| $\alpha$ | $\alpha < q$ and $\alpha$ a primitive root of $q$ |

**User A Key Generation**

| | |
|---|---|
| Select private $X_A$ | $X_A < q$ |
| Calculate public $Y_A$ | $Y_A = \alpha^{X_A} \bmod q$ |

**User B Key Generation**

| | |
|---|---|
| Select private $X_B$ | $X_B < q$ |
| Calculate public $Y_B$ | $Y_B = \alpha^{X_B} \bmod q$ |

Enables two users to exchange a secret key securely.

# Diffie-Hellman Key Exchange

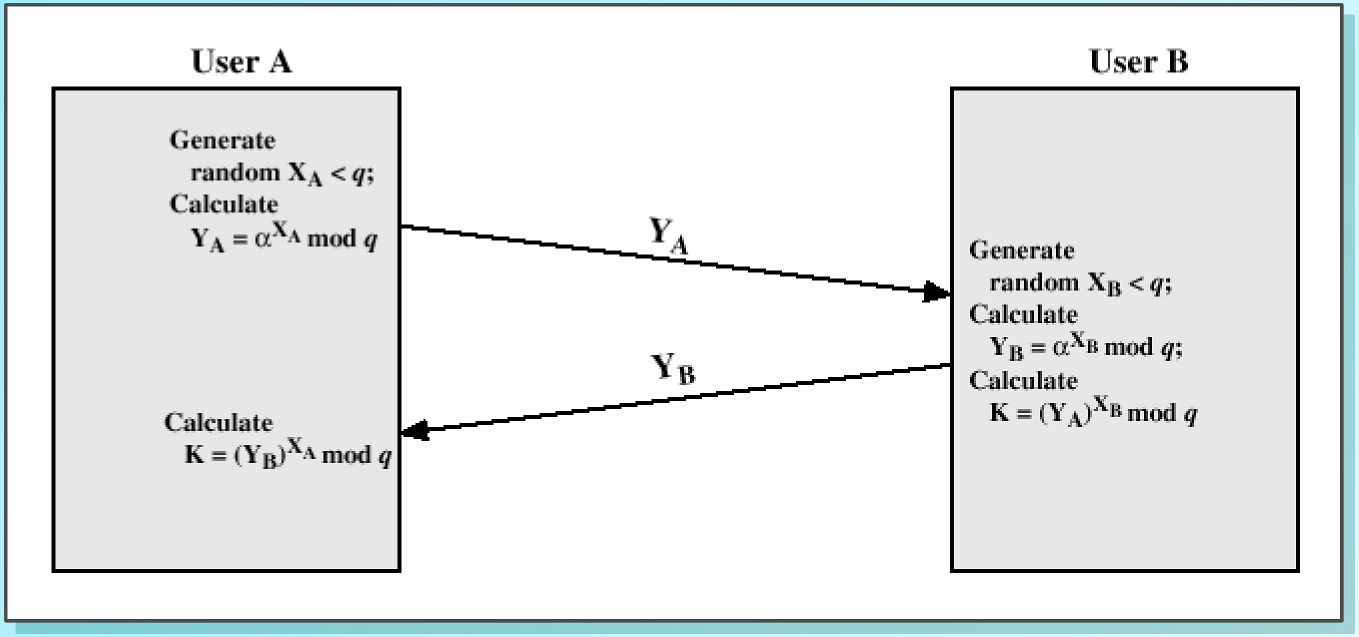
**Generation of Secret Key by User A**

$$K = (Y_B)^{X_A} \bmod q$$

**Generation of Secret Key by User B**

$$K = (Y_A)^{X_B} \bmod q$$

# Diffie-Hellman Key Exchange



# Other Public Key Algorithms

- Digital Signature Standard (DSS) – makes use of SHA-1 and presents a new digital signature algorithm (DSA)
- Only used for digital signatures not encryption or key exchange

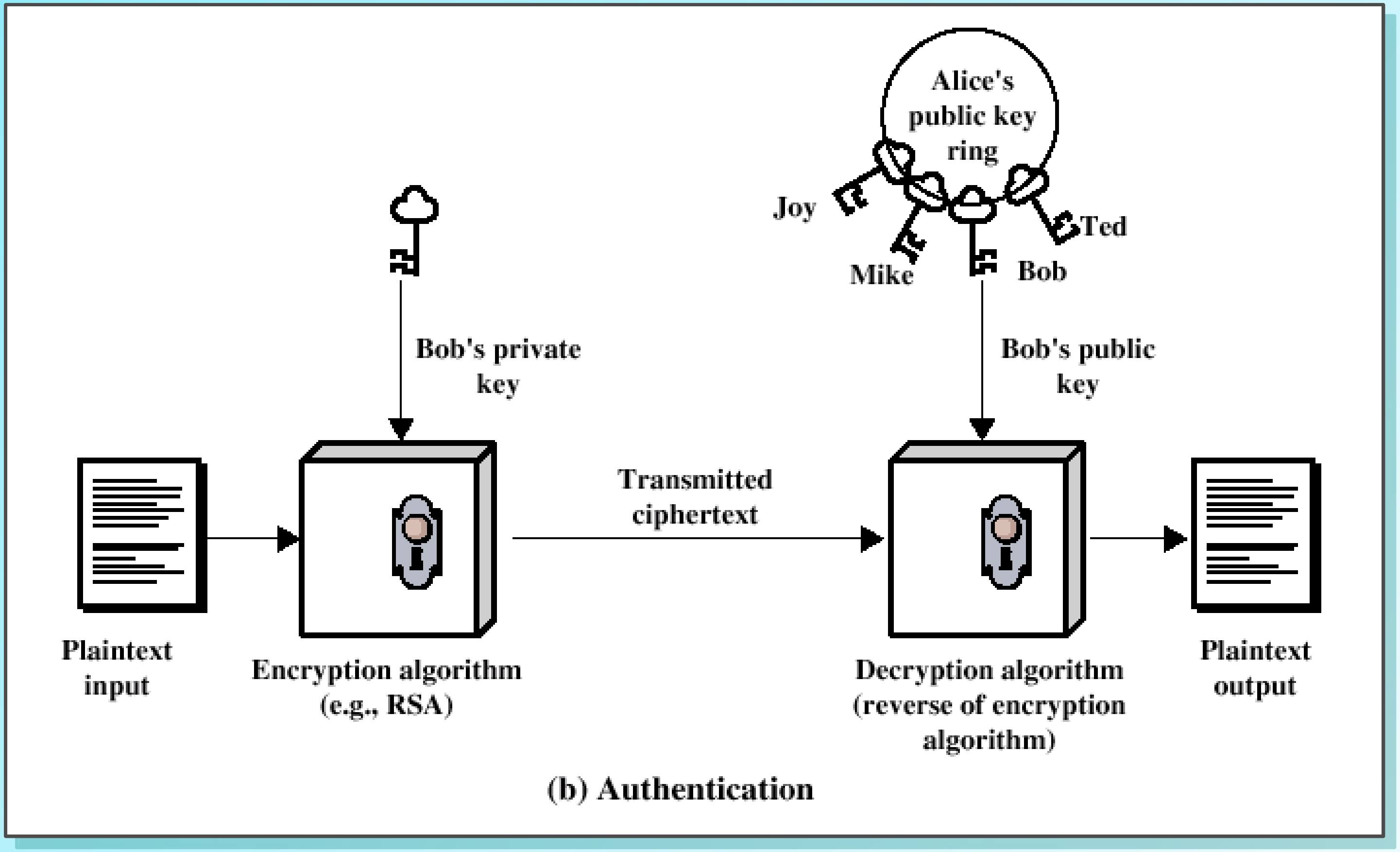
# Other Public Key Algorithms

- Elliptic Curve Cryptography (ECC) – it is beginning to challenge RSA
- Equal security for a far smaller bit size
- Confidence level is not as high yet

# Digital Signatures

- Use the private key to encrypt a message
- Entire encrypted message serves as a digital signature
- Encrypt a small block that is a function of the document, called an authenticator (e.g., SHA-1)

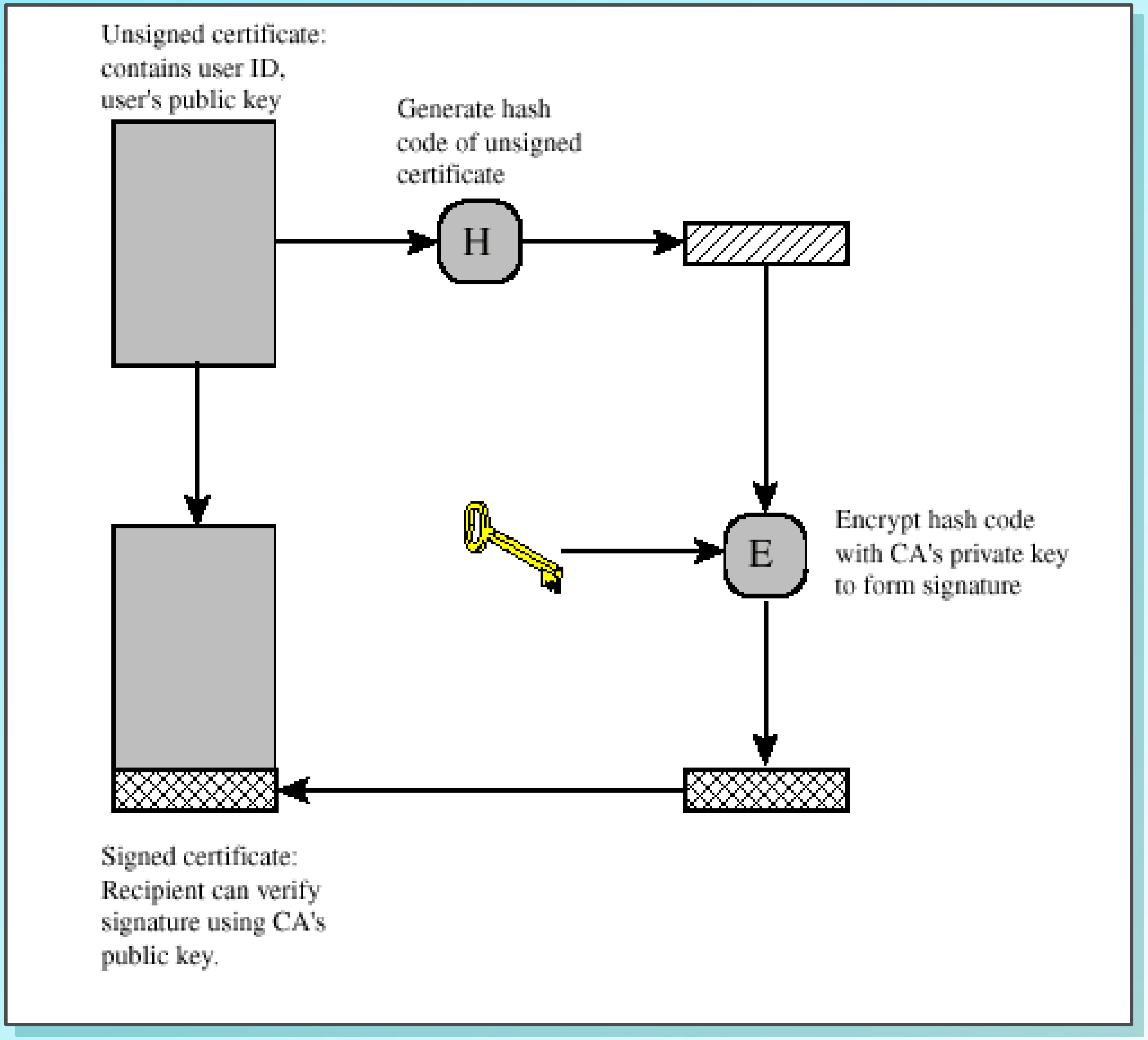
# Public Key Authentication



# Digital Certificate

- Certificate consists of a *public key* plus a *user ID* of the key owner, with the whole block signed by a trusted third party, the certificate authority (CA)
- X.509 standard
- SSL, SET and S/MIME
- Verisign is primary vendor

# Public Key Certificate Use



# Important URLs

- http://www.abanet.org/scitech/ec/isc/dsg-tutorial Discusses the legal implications of **digital signature** usage. (American Bar Association)
- http://www.rsasecurity.com/rsalabs/cryptobytes/ Take a look at **Volume 2, No. 1 - Spring 1996** for the “Aysmmetric Encryption: Evolution and Enhancements”

# Homework

- Read Chapter Three
- Scan Appendix 3A

# Assignment 1

- Pick sun.com and one other site. Using whois and ARIN, get as much information as possible about the IP addressing, the DNS and the site (location, owner, etc.)
- Problems (p83): 3.5,c and 3.6
- Due next class March 6