ECE 463
Lecture: Internet Security

Sanjay Rao

Security Services

Assume A is sending data to B across a network
What security properties are desirable to preserve?

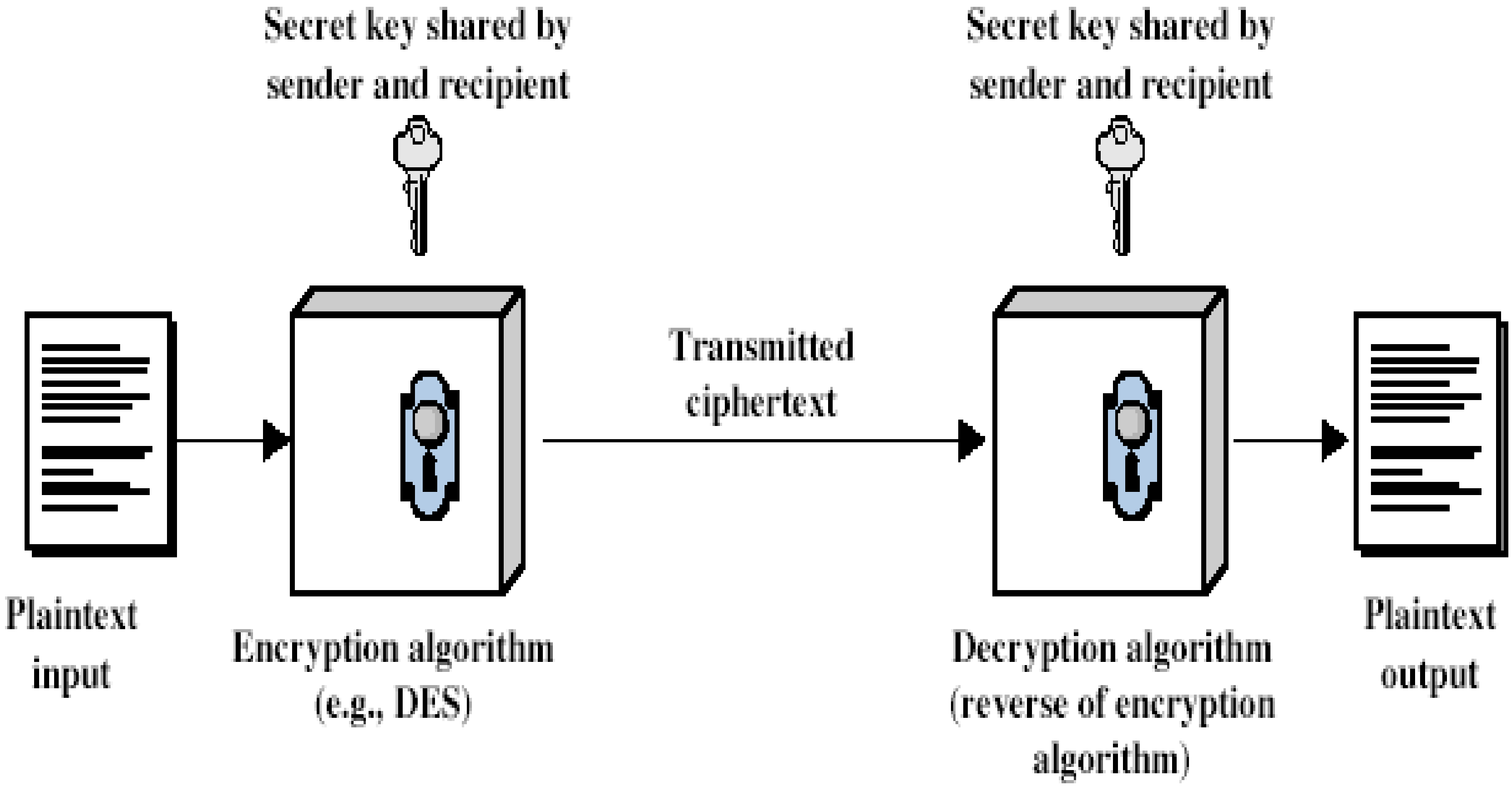
Key Security Services

- Confidentiality
  - Keep information from all except authorized
- Data Integrity
  - Detect unauthorized alteration of data
- Authentication
  - Confirms identity of peer entity during communication
- Non-repudiation
  - Prevent entities from denying previous actions

This Class

- Classes of cryptographic functions
  - Symmetric Key
  - Public/Private Key
  - One-way Hash
- Pros and cons of each class
- Use in SSL

Symmetric Key Cryptography



Caesar Cipher

- Shift by a few characters
- can define transformation as:

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

- mathematically give each letter a number

```
a b c d e f g h i j k  l  m
0 1 2 3 4 5 6 7 8 9 10 11 12
n  o  p  q  r  s  t  u  v  w  x  y  Z
13 14 15 16 17 18 19 20 21 22 23 24 25
```

- then have Caesar cipher as:

$C = \mathrm{E}(p) = (p + k) \bmod (26)$

$p = \mathrm{D}(\mathrm{C}) = (\mathrm{C} - k) \bmod (26)$

Example

- Caesar Cipher (Substitution-based)
  - Algorithm: “letter shift”
  - Key: “Amount of shift”
    - $C_i=E(p_i) = p_i+3$

  - PlainText: A B C D ..
  - CipherText: d e f g …

  - Raw Message: TREATY
  - Encrypted Message: WUHDWB

Cryptanalysis of Caesar Cipher

- only have 26 possible ciphers
  - A maps to A,B,..Z
- could simply try each in turn
- a **brute force search**
- given ciphertext, just try all shifts of letters
- do need to recognize when have plaintext

Monoalphabetic Cipher

- rather than just shifting the alphabet
- each plaintext letter maps to a different random ciphertext letter

```
Plain:  abcdefghijklmnopqrstuvwxyz
Cipher: DKVQFIBJWPESCXHTMYAUOLRGZN
Plaintext:  ifwewishtoreplaceletters
Ciphertext: WIRFRWAJUHYFTSDVFSFUUFYA
```

How many keys in all ?

How easy is a brute force search?

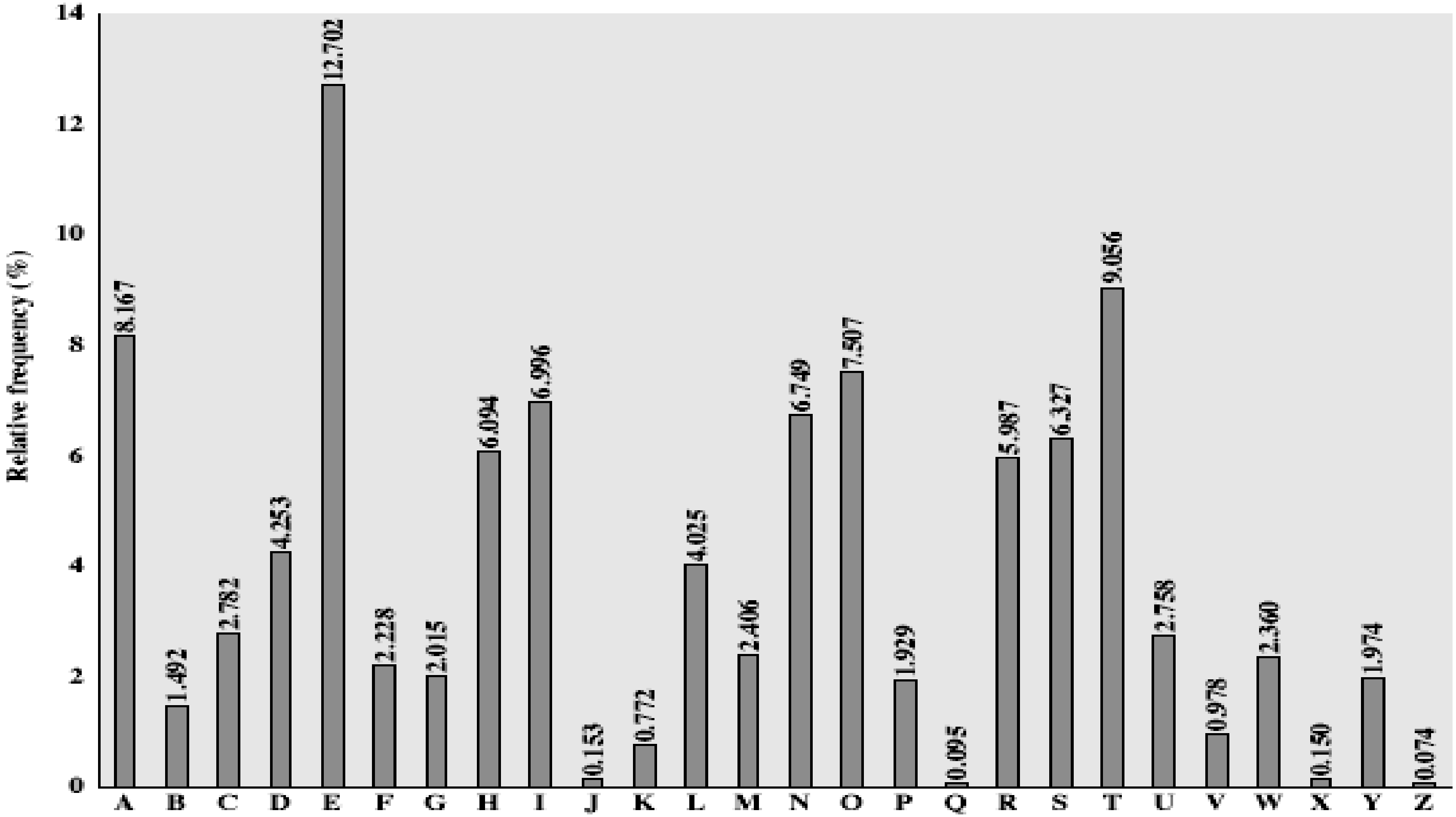
- always possible to simply try every key
- most basic attack, proportional to key size
- assume either know / recognise plaintext

Key Size (bits)	Number of Alternative Keys	Time required at 1 encryption/μs	Time required at 10^6 encryptions/μs
32	$2^{32} = 4.3 \times 10^9$	2^{31} μs = 35.8 minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	2^{55} μs = 1142 years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	2^{127} μs = 5.4×10^{24} years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	2^{167} μs = 5.9×10^{36} years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	2×10^{26} μs = 6.4×10^{12} years	6.4×10^6 years

CryptAnalysis of Monoalphabetic cipher

- No of keys: (26!)
  - Brute force search not as easy
- However, easy to cryptanalyze
  - Letters are not equally commonly used
  - E.g. **e** is by far the most common letter
- Have tables of single, double & triple letter frequencies

English Letter Frequencies



Transposition Ciphers

- Ciphers discussed so far:
  - “Substitution” based
  - Alter letters used based on keys
- Another class:
  - **Transposition** or **permutation** ciphers
  - “Permute” characters of the original text

Permutations

- Rearrange letters of a word.
- E.g. Columnar transposition
  - Plain text: HAPPY HOLIDAYS

    H A P P Y
    H O L I D
    A Y S

  - Encrypted Text: HHAAOYPLSPIYD

What’s used in practice?

- Data Encryption Standard (DES)
- Careful and complex combination of:
  - Substitutions
  - Permutations
- Text encrypted as blocks of 64 bits.
- DES Key is 64 bits long
  - Effectively 56

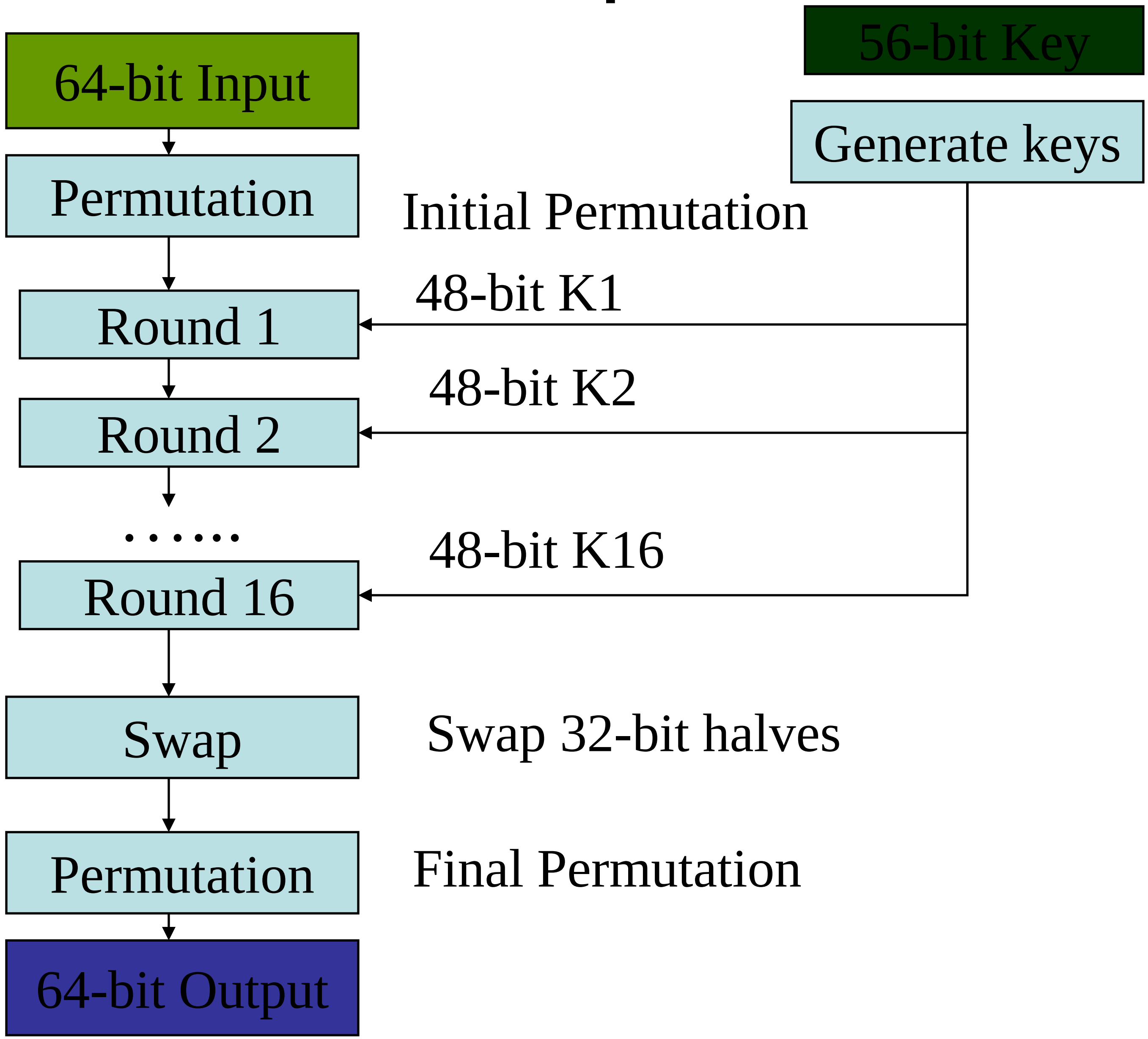
DES: Objectives

- Input: 64 bits   Output: 64 bits
  - Looks “random”
- Confusion/Diffusion:
  - “Dissipate” statistical structure of plaintext
  - 1 bit input change must affect many op bits
  - Every cipher bit affected by several input bits

DES Design

- Building blocks
  - Substitutions
  - Permutations
- DES:
  - Cycles of substitutions & permutations
- Design process not made public
  - Not always clear why certain choices were made

DES Top View

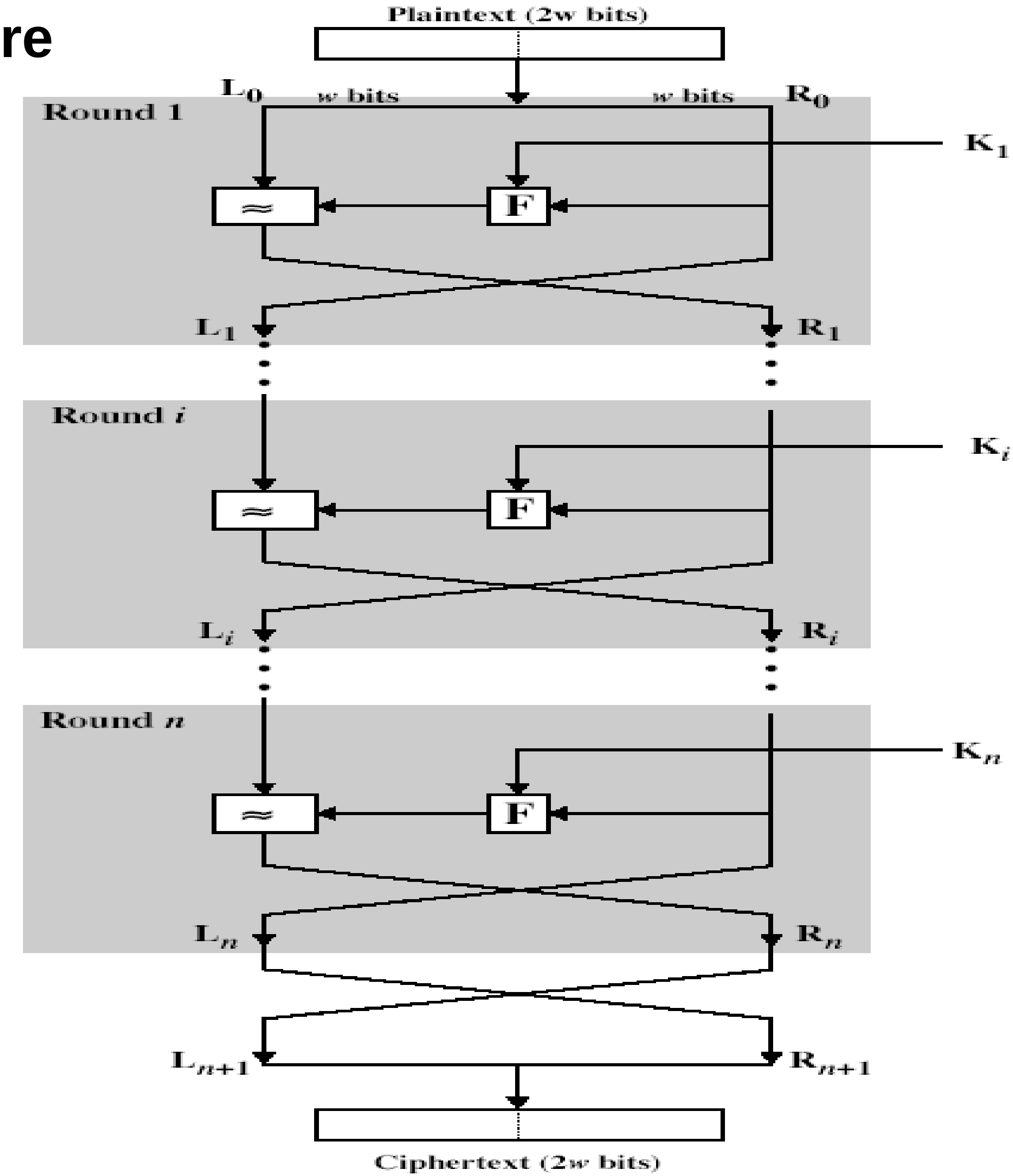

56-bit Key
64-bit Input
Generate keys
Permutation
Initial Permutation
48-bit K1
Round 1
48-bit K2
Round 2
……
48-bit K16
Round 16
Swap
Swap 32-bit halves
Permutation
Final Permutation
64-bit Output

Feistel Cipher Structure

F: “Mangler Function”

$L_i = R_{i-1}$

$R_i = L_{i-1}$ XOR $F(R_{i-1}, K_i)$



DES Box Summary

- Simple, easy to implement:
  - Hardware/gigabits/second, software/megabits/second
- 56-bit key DES acceptable for non-critical applications
- Achieving greater security: Triple DES : 112 bit effective key.
- More recent trends:
  - AES (Advanced Encryption Standard)

Achieving Security Services

- Symmetric cryptography may be used to achieve…
  - Confidentiality ? Yes!
  - Authentication ? Yes!
  - Data Integrity ? Yes!
  - Non-repudiation ???

Non-Repudiation

- A sends B a message
  - (e.g. ordering equipment)
- A refuses to accept this at a later point.
- How does B prove to a judge C that A indeed placed the order?

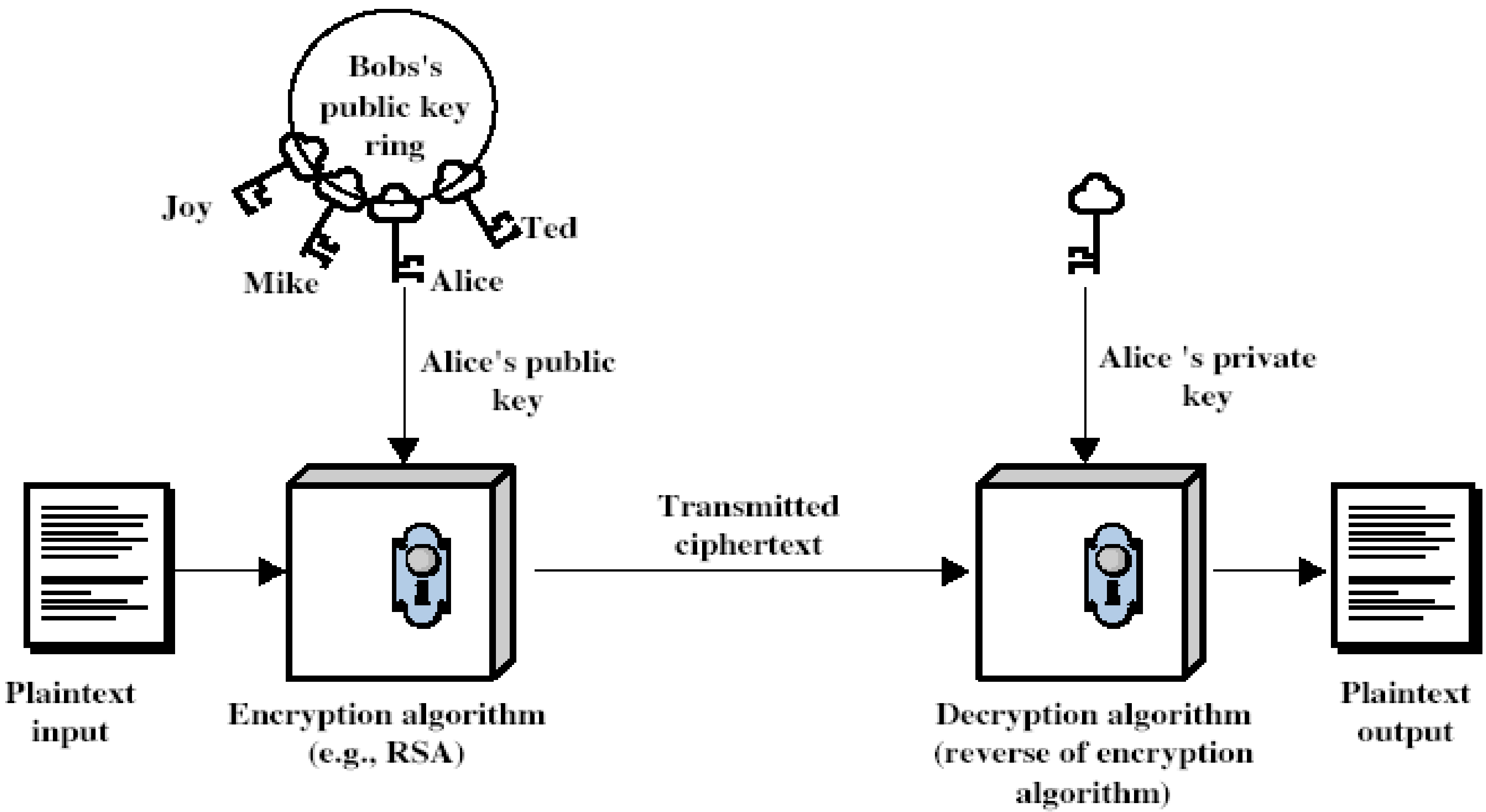
- Cannot be achieved with symmetric key systems.

Another issue with Symm key

- Assume a set of N people
- Assume every pair wants to communicate

- Number of symmetric keys needed:
  - $N * (N-1)/2$.

Public-Key Cryptography



Public-Key Cryptography

- Two keys
  - Public key: globally known
    - Anyone can send by encrypting using public
  - **Private-key**: known only to the recipient
    - Only recipient can **decrypt messages**
- is **asymmetric** because
  - those who encrypt messages **cannot** decrypt messages

Public-Key Characteristics

- Public-Key algorithms rely on two keys with the characteristics that it is:
  - computationally infeasible to find decryption key knowing only algorithm & encryption key
  - computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known

RSA (Rivest, Shamir, Adleman)

- The most popular one.
- Support both public key encryption and digital signature.
- Assumption/theoretical basis:
  - Factoring a big number is hard.
- Variable key length (usually 512 bits).

Factoring

- Creating database of all primes impractical
  - Knowing a few large primes easier…
  - Knowing **all** primes less than (large) number harder!

- Number of primes with 1024 bit RSA:
  - Approx 10 ^ 297

- Analogies:
  - Atoms in the universe: 10^{77}
  - Seconds in a year: 10^{7}
  - Age of the universe: 10^{10} *years*

Other public-key cryptosystems

- Many other public-key cryptosystems
  - Diffie Hellman
  - DSS
  - ECC (Elliptic Curve)

Non-Repudiation

- Feasible with public-key.
- “Digital Signatures”.

Dual use of public key systems

- A sends data to B
  - Encrypt with B’s pub key
  - Sign with A’s private key
- On receiving data:
  - B decrypts with its private key
  - Verifies signature with A’s public key

- Clarification:
  - RSA allow both encryption/signatures
  - Some public-key systems allow only one or the other

Another win with public-key

- Assume N nodes
- Every pair communicates
  - Symmetric key: $(N * (N-1)) / 2$ keys
  - Public key: $2*N$ keys

Question

- Why use symmetric key at all?

Example Application

- Public key systems are much slower.
- Public key crypto & Symmetric Crypto combined
  - Assume C wants to talk to S
  - C uses S's public key to encrypt "secret key" used for the session.
  - "Secret key" used to encrypt messages

  - This concept used in SSL, PGP etc.

Hash Functions

- Applied to any sized message M
- Produces a fixed-length output `h`

- Easy to compute `h=H(M)` for any M
- Hard to "invert":
  - Given h, hard to find M such that `H(M)=h`

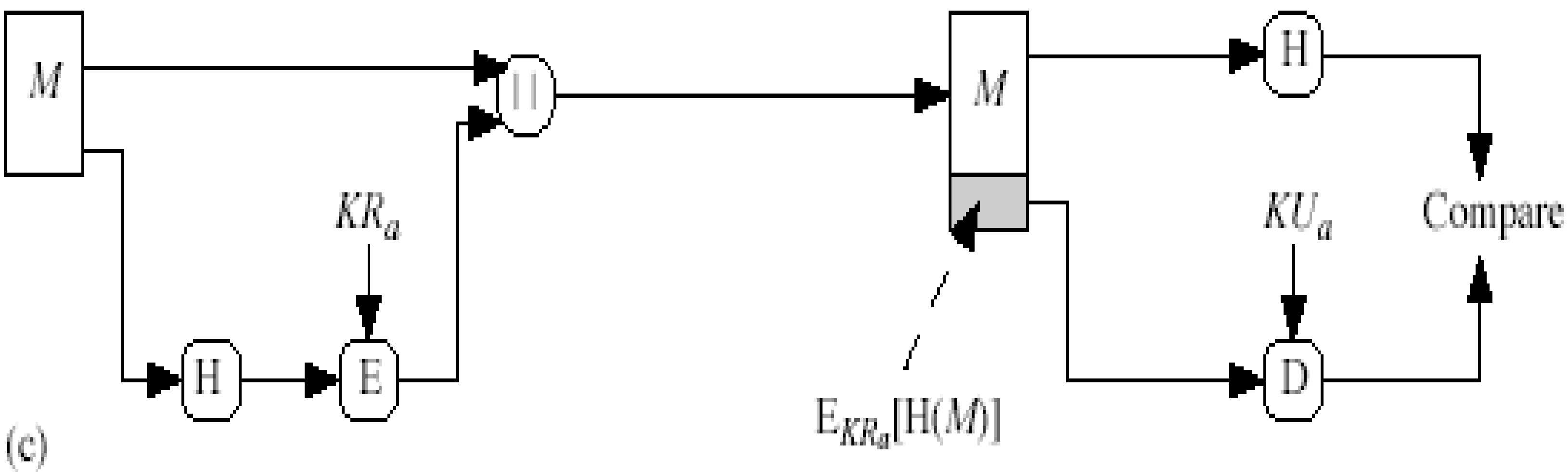
- All algorithms are public!

Hash Functions

- One-way hash functions
  - Given message M, produce h(m) (smaller)
  - Given h(m), hard to find m (or alternate m1)

- Application to digital signature:
  - Produce “hash” of message m
  - Sign the hash using sender’s private key

Hash Functions & Digital Signatures



Hash Function Properties

- Hash function algorithms public
- Can be applied to any sized message M
- Produces fixed-length output `h`
- Is easy to compute `h=H(M)` for any message `M`
- `Popular Example : MD5`

Requirements for Hash Functions

1. Given `h` is infeasible to find `x` s.t. `H(x)=h`

   One-way property

2. given `x` is infeasible to find `y` s.t. `H(y)=H(x)`

   - Weak collision resistance

3. is infeasible to find any `x,y` s.t. `H(y)=H(x)`

   - Strong collision resistance

SSL and TLS History

- SSL was originated by Netscape
  - SSLv1 never deployed
  - SSLv2 deployed in Netscape 1.1 in 1995
  - SSLv3 fixed several security issues
- TLS: standardization process
  - http://www.ietf.org/html.charters/tls-charter.html

When invoked?

- Using https:// (instead of http://)

Obtaining a Certificate

- E.g. Server generates RSA key pair
- Obtains “certificate” for public key
- Issued by one of certificate authorities
- Key of CA must be present in client browser
- Typically browsers shipped with keys
  - Several CA’s
  - Many issues due to this dependency

Key Generation

- Client receives server certificate
- Verifies certificate using public key of CA
  - Extracts server's public key
- Client generates 48 byte pre-master secret
  - Used to produce master secret
  - Encrypted with server public-key, sent back