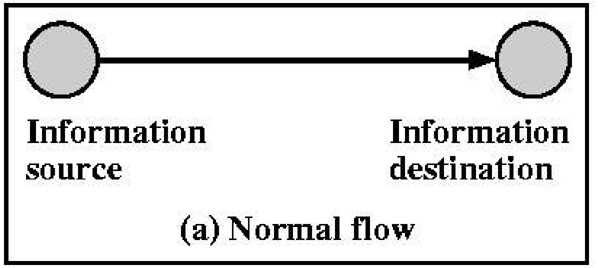
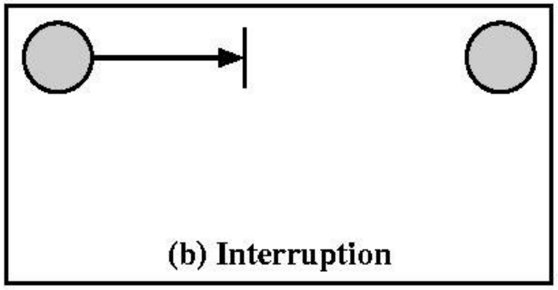
Security

- Communication security
  - security of data channel
  - typical assumption: adversary has access to the physical link over which data is transmitted
  - cryptographic separation is necessary
- System Security
  - security at the end points
  - information cannot be encrypted, as it needs to be accessed by applications on the end system
  - logical separation is typically the basis

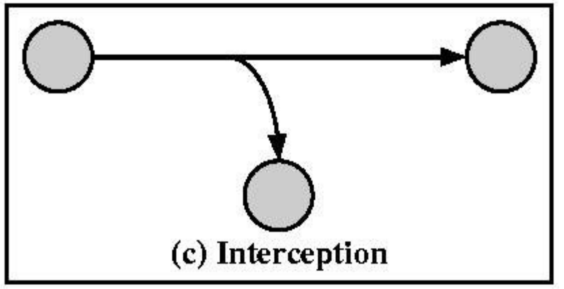
Security Concerns



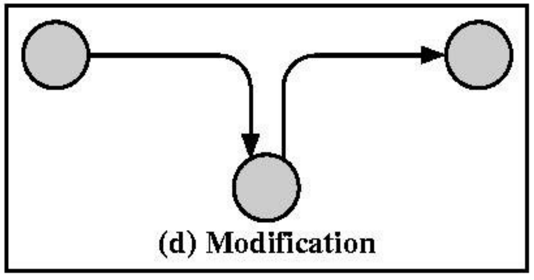
(a) Normal flow


(b) Interruption

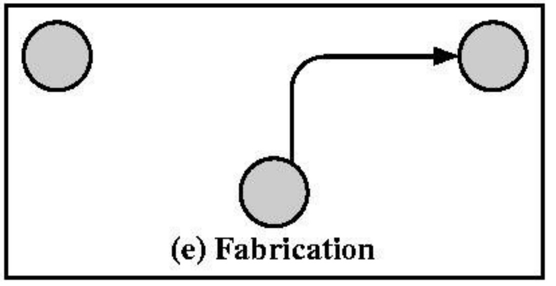
Availability


(c) Interception

Confidentiality


(d) Modification

Authenticity, integrity


(e) Fabrication

Nonrepudiability

How to achieve security

- Basis is separation
  - Separate adversarial entities
- How to separate adversaries?
  - Physical separation
  - Temporal separation
  - Cryptographic separation
  - Logical separation
- Security vs Functionality
  - Controlled sharing

Cryptography

- Encode the data in a manner that makes it accessible only to authorized parties
  - Encryption algorithm
  - Encryption key
- Why it is not a good idea to rely on secrecy of algorithm
  - Hard to develop good encryption algorithm
  - Does not scale beyond a few users
  - Security by obscurity
- Key point: need to preserve secrecy of key

Key concepts and terminology

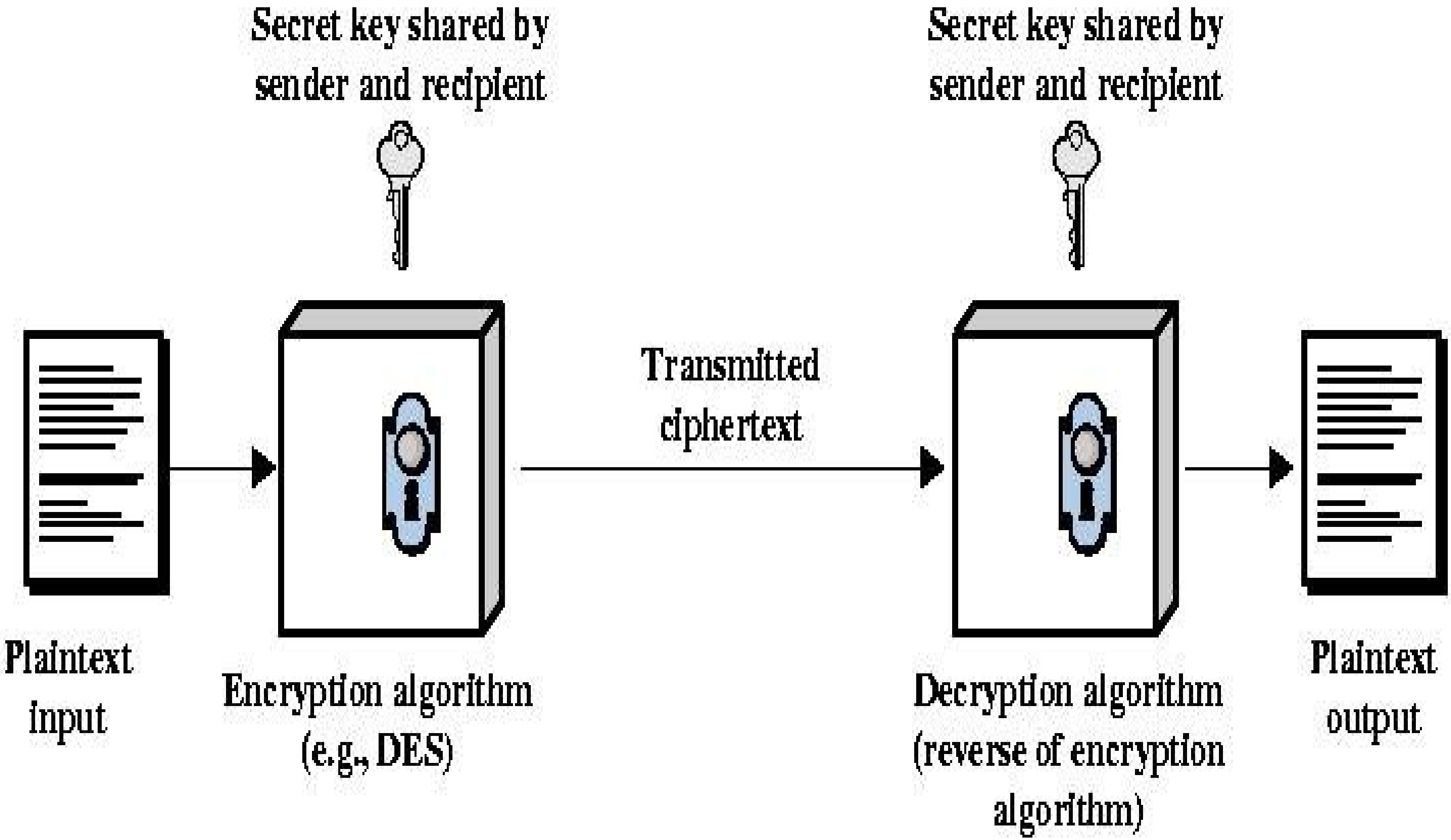
- Plaintext (“unencrypted”)
- Ciphertext (“encrypted”)
- Encryption ($E_k(X)$) Vs Decryption ($D_k(X)$)
- Key Vs Algorithm
- Cryptanalysis: Discover k, X or both

Type of Attack	Known to Cryptanalyst
Ciphertext only	• Encryption algorithm • Ciphertext to be decoded
Known plaintext	• Encryption algorithm • Ciphertext to be decoded • One or more plaintext-ciphertext pairs formed with the secret key
Chosen plaintext	• Encryption algorithm • Ciphertext to be decoded • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key
Chosen ciphertext	• Encryption algorithm • Ciphertext to be decoded • Purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key
Chosen text	• Encryption algorithm • Ciphertext to be decoded • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key • Purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

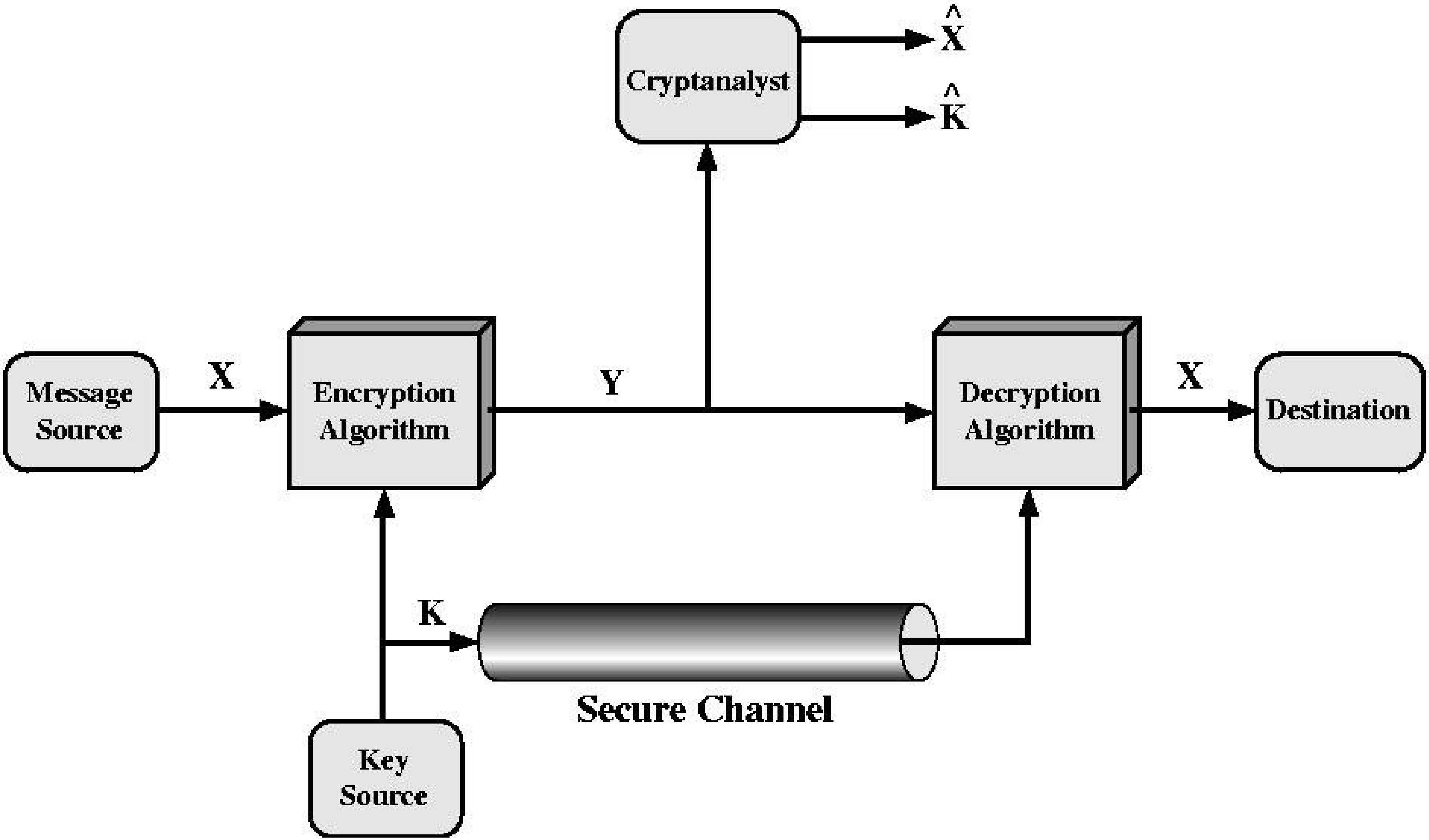
Steganography

- Hiding presence of information
- Use normal-looking messages/pictures that conceal secret data
- Useful if communication is monitored for “suspicious content” by someone
- Also used for copyright protection
  - ***Watermark:*** invisible data encoded in messages that is retained in copies, and is robust in the face of typical image transformation operations

Symmetric Crypto



Model of Symmetric Crypto

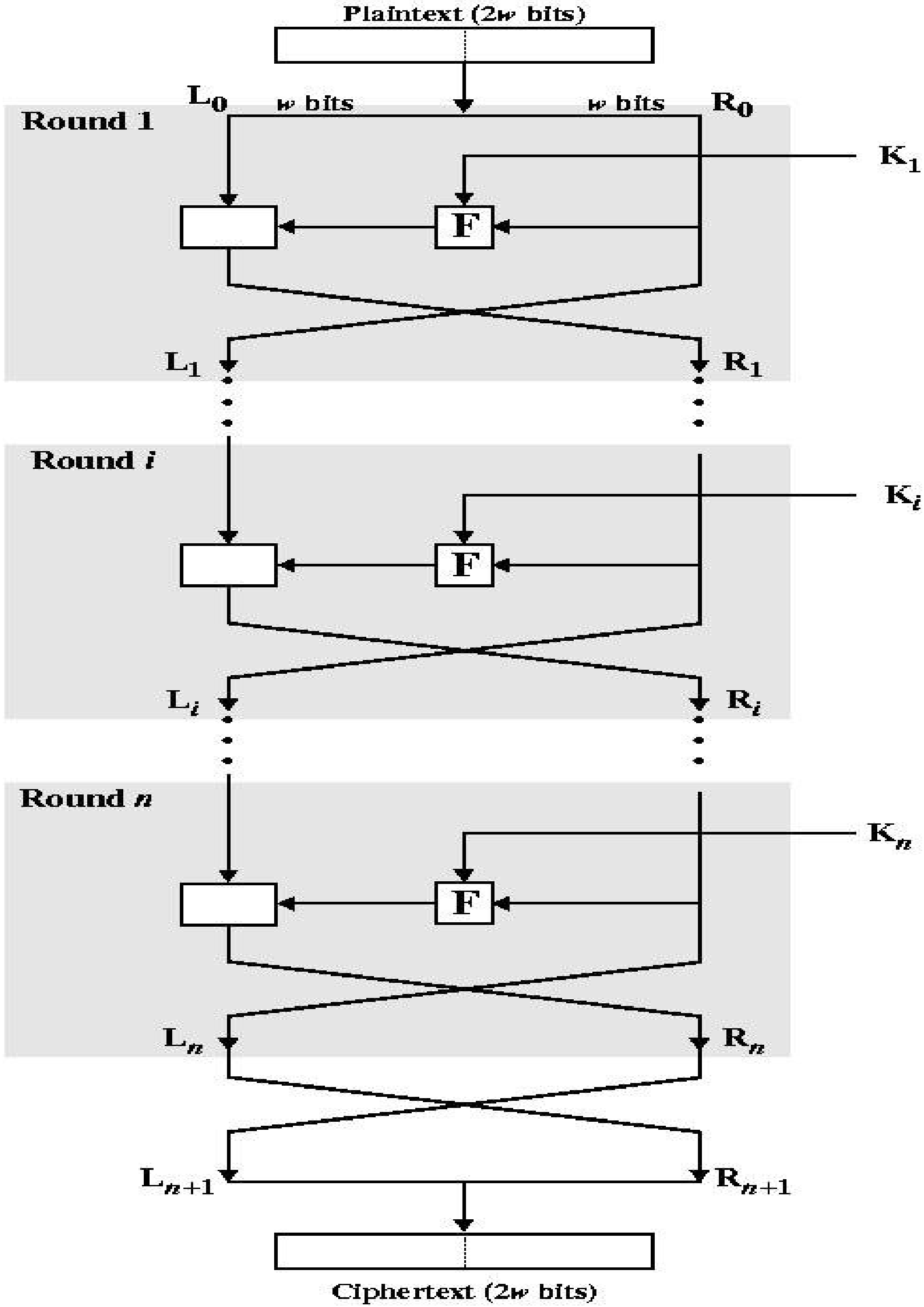


Stream and Block Ciphers

- Stream cipher: used to encrypt digital streams of data, one bit or a byte at a time
- Block cipher: data is partitioned into blocks (typically 64 or 128 bits), and encryption operates on these blocks.
- Stream ciphers can be constructed from block ciphers
  - For this reason, crypto algorithms are developed almost exclusively for block ciphers

Structure of Symmetric Crypto

- Needs to produce a reversible mapping that maps 64-bit blocks onto other 64-bit blocks
- Good ciphers are based on Shannon's concepts of "diffusion" and "confusion"
  - Diffusion: disperse bit-patterns within each block of data
  - Confusion: "mix-up" the order of bits within a block. In practice, use permutations specified by a key.
- In principle, good ciphers can be implemented using a table of mappings
  - Encryption key selects which mapping to use
  - Approach impractical for all except smallest block sizes
- Feistel structure: a way to build more complex ciphers from simpler ones


Plaintext (2w bits)
L0
w bits
w bits
R0
Round 1
K1
F
L1
R1
Round i
Ki
F
Li
Ri
Round n
Kn
F
Ln
Rn
Ln+1
Rn+1
Ciphertext (2w bits)

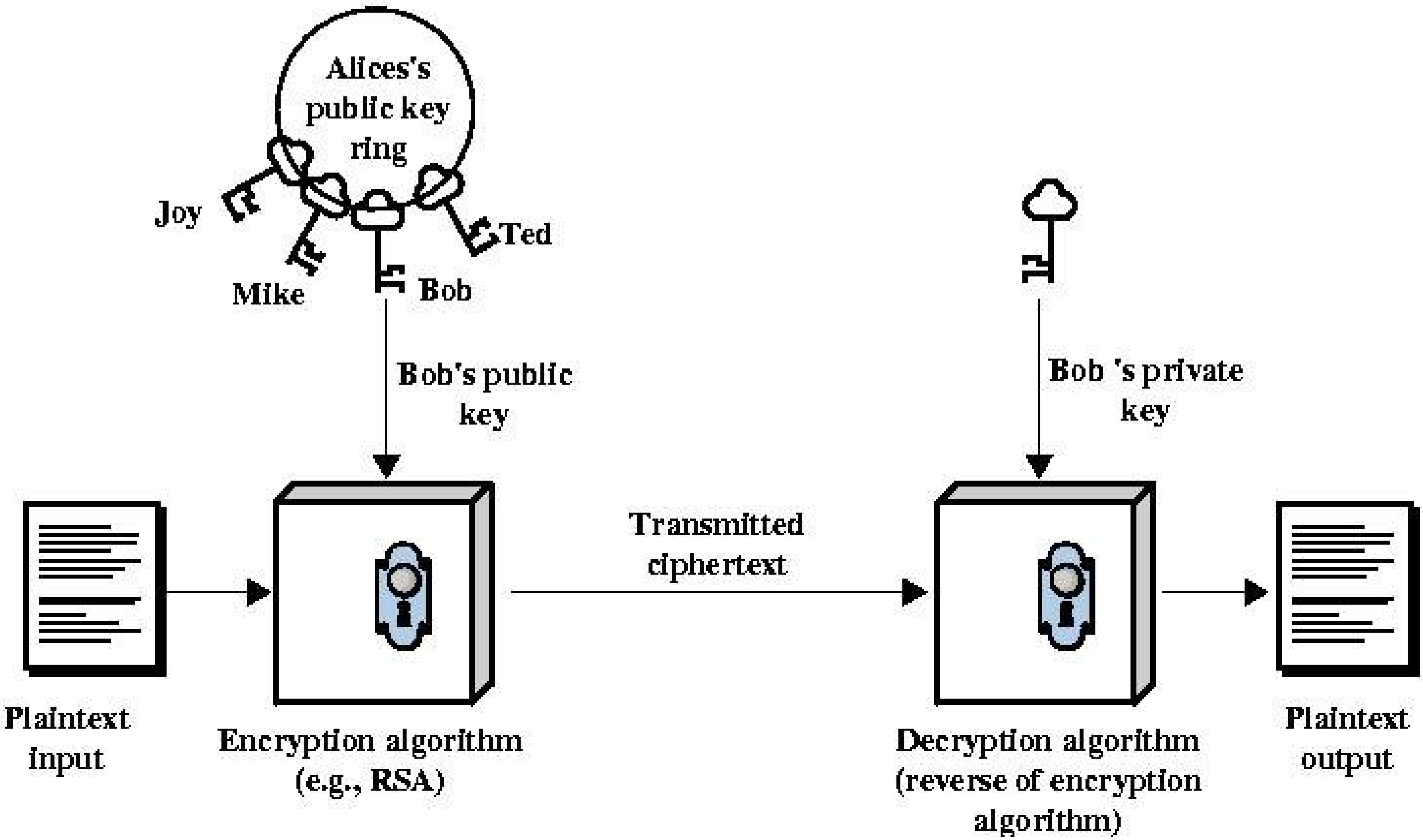
Symmetric Crypto Algorithms

- DES
  - Not considered very secure (key length of 56 bits)
- Triple DES with two keys (128 bits)
- AES (128 bits)
- IDEA (128 bits)
- Blowfish (up to 448 bits)
- RC5 (up to 2040 bits)
- CAST-128 (40 to 128 bits)
- RC2 (8 to 1024 bits)

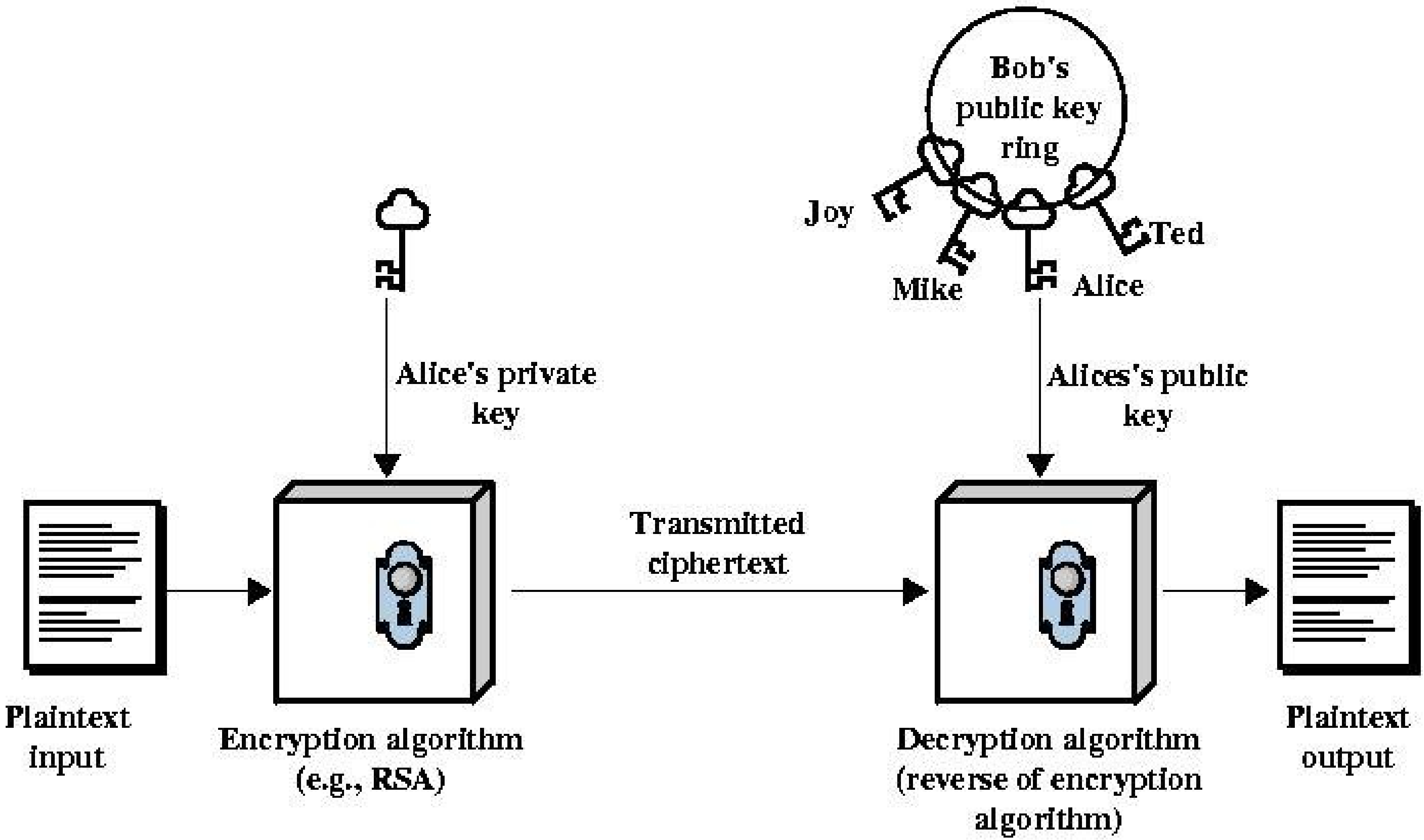
Public Key (Asymmetric) Crypto

- Uses one key for encryption and another one for decryption
  - Requires that it be computationally infeasible to compute one of the keys based on the other
- One of the two keys is private to a *principal*; the other key can be freely distributed to any one
  - Each principal generates his/her own pair of public/private keys, and the private key need not be revealed to any one.
- Some public key algorithms (e.g., RSA) permit both keys to be used for encryption and decryption
  - What is encrypted with one key can be decrypted with the other

Encryption in Public Key Crypto



Authentication in Public Key Crypto



Encryption Vs Signing

- When the encoding operation is performed using someone's public key, the results are accessible only to that person
  - This operation can be used to ensure confidentiality of data --- hence called "encryption"
- When the encoding operation is done using someone's private key, the results are accessible to every one.
  - But one can be sure that the message came only from the person whose public key is used for decoding --- hence called "signing"

RSA Algorithm

- Alphabet = {0,...,n-1}
  - in practice, $\{0,\ldots,2^k\}$ for $2^k < n <= 2^{k+1}$
- Encryption: $C = M^e \bmod n$
- Decryption: $M = C^d \bmod n$

  $= (M^e)^d \bmod n$

  $= M^{ed} \bmod n$
- Need: $M^{ed} = M \bmod n$
- Both sender and receiver know n.
- Sender knows e, while only the receiver knows d.
- ie, KU = (e, n), KR = {d, n}

RSA Algorithm Requirements

- It is possible to find d,e, n s.t.

  $M^{ed} = M \bmod n$, for all $M < n$

- It is easy to calculate M^e and C^d
- It is infeasible to determine d from e

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	$KR = \{d, n\}$

Encryption

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

Decryption

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

Miller-Rabin Test

- Pick an odd number n, note $n\text{-}1 = 2^k q$, where q is odd
- Pick a number $1 < a < n\text{-}1$, compute $a^q, a^{2q}, \ldots, a^{2^k q}$
- If n is prime, by Fermat's theorem

  $a^{2^k q} \bmod n = a^{n-1} \bmod n = 1$

  Hence, for some $0 \le j \le k$, $a^{2^j q} \bmod n = 1$

  - Case 1: $j = 0$: this means $a^q \bmod n = 1$
  - Case 2: $j > 0$, $a^{2^{j-1} q} \bmod n \neq 1$, $a^{2^j q} \bmod n = 1$

    i.e., $(a^{2^{j-1} q} - 1) * (a^{2^{j-1} q} + 1) \bmod n = 0$

    Since the first factor is nonzero, we have

    $(a^{2^{j-1} q} + 1) \bmod n = 0$, or, $a^{2^{j-1} q} \bmod n = n\text{-}1$
- The algorithm tests for case 1 or case 2.
  - If the test fails, that means n is composite
  - If it succeeds, n is not guaranteed to be prime
    - but the probability of success for a nonprime is less than 0.25
    - repeat the test for t different a's to get a prime with probability $1\text{-}(0.25)^t$

Conventional Vs Public Key Crypto

- Conventional crypto is fast
  - Software implementations on current PCs can perform encryption at the rate of few MB/s
- Public key crypto is much slower
  - At least 3 orders of magnitude slower
- Key distribution is easier with public keys
  - Need to ensure authenticity of public keys
  - For conventional keys, confidentiality is needed
- Solution
  - Use conventional crypto for encrypting bulk data
  - Use public key crypto to exchange keys for such encryption.
    - conventional keys are encrypted using public keys and sent to the recipient.
  - Use certificates and certification authorities (CAs) to establish authenticity of public keys

Uses of Random Numbers

- Nonces (to protect against replay attacks)
- Session key generation
- RSA key generation

- Need cryptographically strong random number generator
  - Not enough if we had "random" numbers in a statistical sense
  - Need unpredictability

Pseudorandom number generators

- Linear congruential method
  - $\boldsymbol{X_{n+1} = (aX_n + x) \bmod m}$
  - Not good for crypto applications, as it is predictable
- Cyclic encryption
  - $\boldsymbol{E_k(n)}$, for n = 0,1,2,…
  - Problem: what happens after a system restart? Does n go back to zero? If so, the random number sequence becomes predictable (seen before)
  - Solutions:
    - use something like a real-time clock plus sequence number
    - use “true random” source

Natural Random Noise

- Best source is natural randomness in real world
  - radiation counters
  - radio noise
  - Keystroke intervals
  - Network packet arrival characteristics

Digital Signatures

- Required properties
  - receiver can verify who sends
  - sender can not repudiate
  - receiver can not generate
- Conventional crypto is not very useful
  - Sender and recipient share key, so nonrepudiability is a problem
- Public-key signature
  - Originator simply encrypts the message using private key
  - When the receiver gets the message decrypted using the originator's public key, then we can be sure about who sent the message
- Note that the encrypted message can be produced only by the originator, so all of the above properties are satisfied.

Message Digests

- Encrypting the whole message for signature purposes is impractical (too inefficient)
- Solution
  - use one-way hash functions: compute a fixed-size (e.g., 128-bit) hash on the message
  - Encrypt the hash using private key
- One-way hash code:
  - Given P, it is easy to compute H(P)
  - Given H(P), it is impossible find P
  - No one can generate two messages that have the same message digest
- Common hash functions
  - MD5
  - SHA-1
  - RIPEMD-160

Digital Certificates

- Certificates are issued by a CA
- Every one knows the public keys of the CA
- A certificate for a principal A is simply A's public key that is encrypted with CA's private key
  - Only the CA could have produced such a message, so the recipient of the certificate knows that the CA vouches for A's public key
  - If the recipient trusts CA, then the certificate provides a simple way to authenticate the public key of A.

Public-Key Certificates

- certificates allow key exchange without real-time access to public-key authority
- a certificate binds **identity** to **public key**
  - usually with other info such as period of validity, rights of use etc
- with all contents **signed** by a trusted Public-Key or Certificate Authority (CA)
- can be verified by anyone who knows the public-key authorities public-key