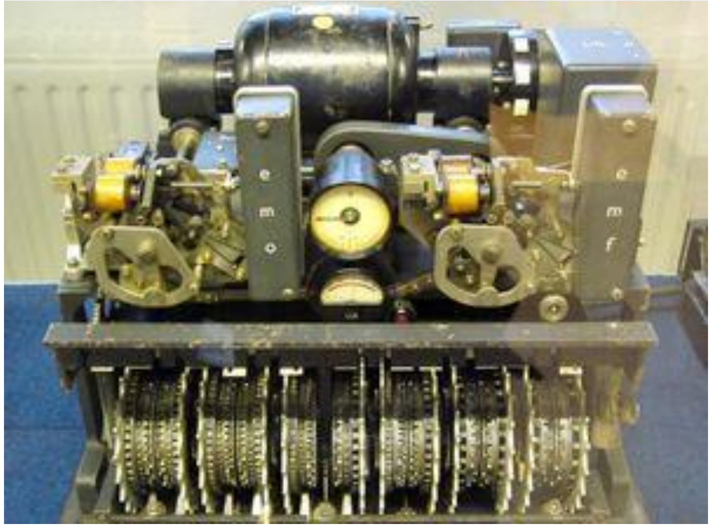


Cryptography

COS 461: Computer Networks
Princeton University

Overview

- Network security and definitions
- Brief introduction to cryptography
  - Cryptographic hash functions
  - Symmetric-key crypto
  - Public-key crypto
  - Hybrid crypto

Internet’s Design: Insecure

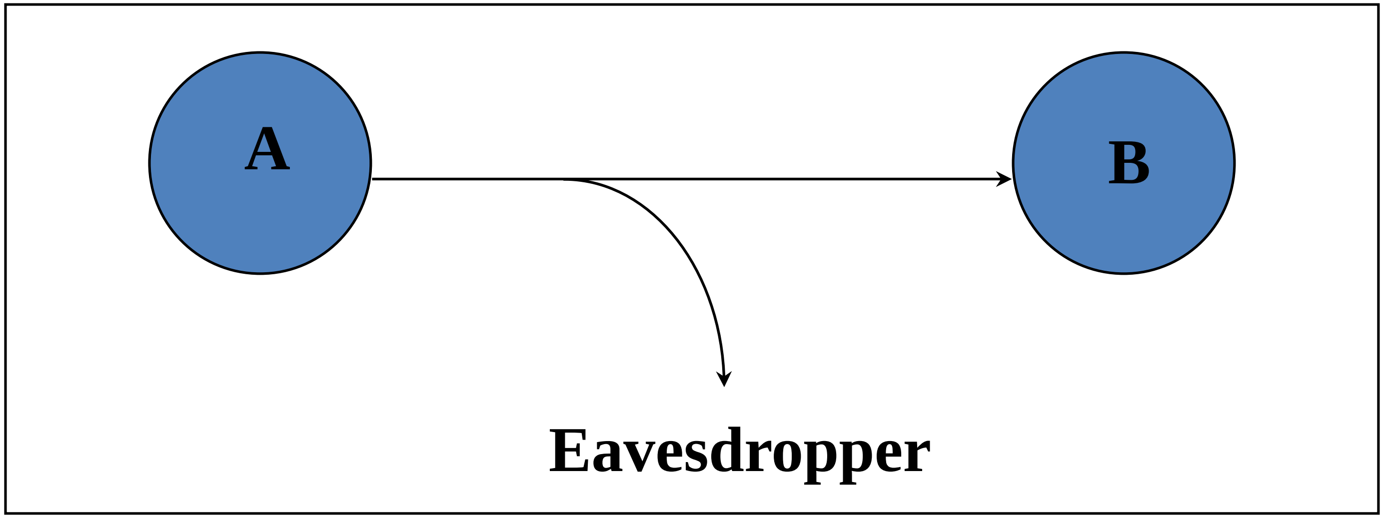
- Designed for simplicity
- “On by default” design
- Readily available zombie machines
- Attacks look like normal traffic
- Internet’s federated operation obstructs cooperation for diagnosis/mitigation

Basic Concepts

- **Confidentiality:** Concealment of information or resources
- **Authenticity:** Identification and assurance of origin of info
- **Integrity:** Trustworthiness of data or resources in terms of preventing improper and unauthorized changes
- **Availability:** Ability to use desired info or resource
- **Non-repudiation:** Offer of evidence that a party indeed is sender or a receiver of certain information
- **Access control:** Facilities to determine and enforce who is allowed access to what resources (host, software, network, …)

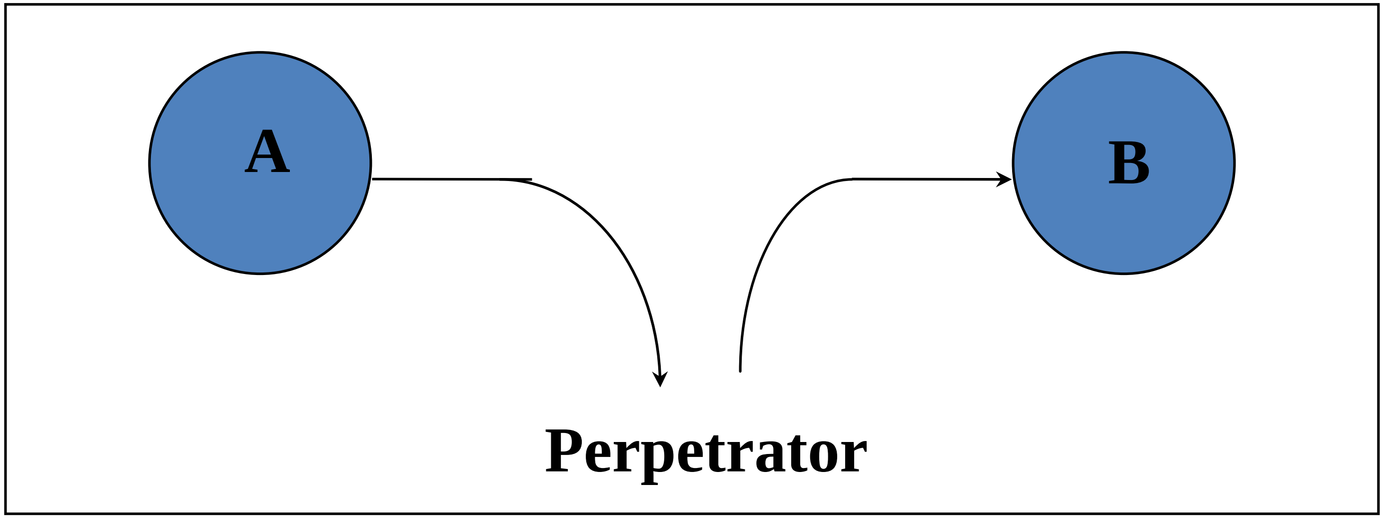
Eavesdropping - Message Interception (Attack on Confidentiality)

- Unauthorized access to information
- Packet sniffers and wiretappers (e.g. tcpdump)
- Illicit copying of files and programs



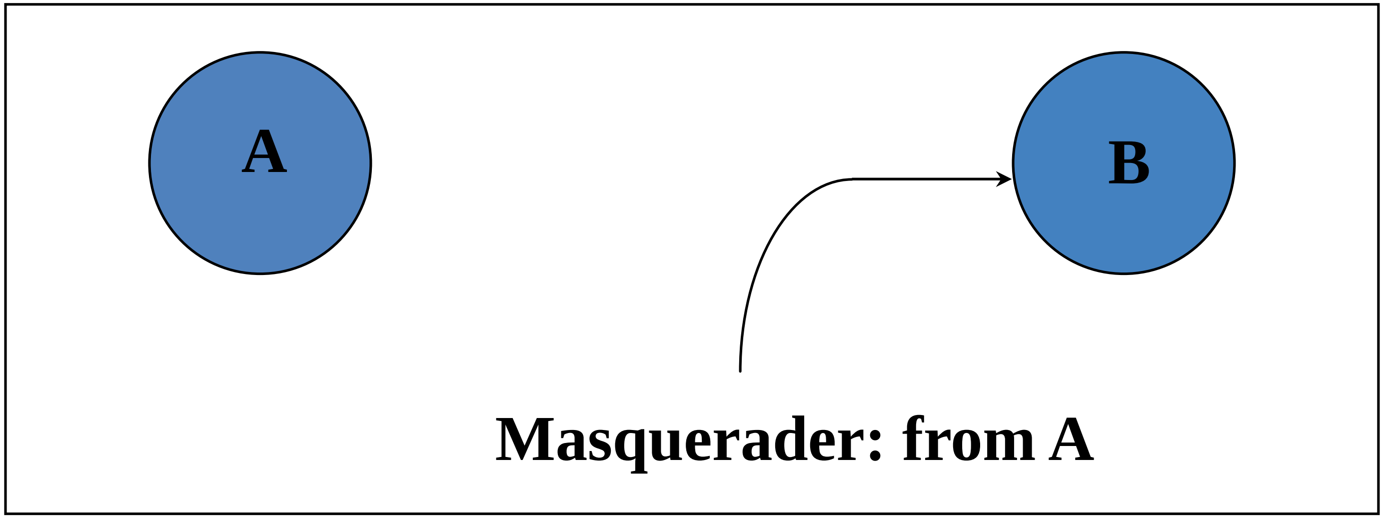
Integrity Attack - Tampering

- Stop the flow of the message
- Delay and optionally modify the message
- Release the message again



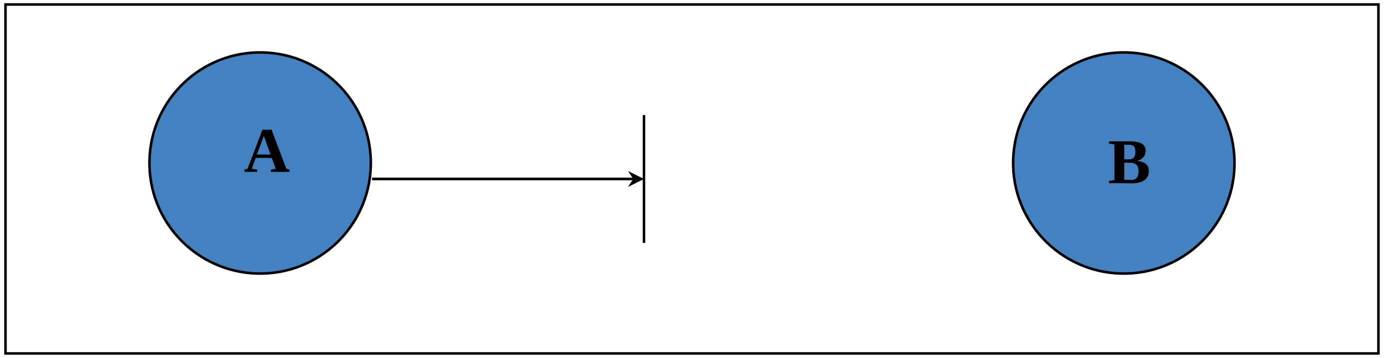
Authenticity Attack - Fabrication

- Unauthorized assumption of other's identity
- Generate and distribute objects under identity



Attack on Availability

- Destroy hardware (cutting fiber) or software
- Modify software in a subtle way
- Corrupt packets in transit



- Blatant *denial of service* (DoS):
  - Crashing the server
  - Overwhelm the server (use up its resource)

Introduction to Cryptography

What is Cryptography?

- Comes from Greek meaning "secret writing"
  - Primitives also can provide integrity, authentication
- Cryptographers invent secret codes to attempt to hide messages from unauthorized observers

plaintext —**encryption**→ **ciphertext** —**decryption**→ **plaintext**

- Modern encryption:
  - *Algorithm* public, *key* secret and provides security
  - May be symmetric (secret) or asymmetric (public)

Cryptographic Algorithms: Goal

- One-way functions: cryptographic hash
  - Easy to compute hash
  - Hard to invert
- "Trapdoor" functions: encryption/signatures
  - Given ciphertext alone, hard to compute plaintext (invert)
  - Given ciphertext and key (the "trapdoor"), relatively easy to compute plaintext
  - "Level" of security often based on "length" of key

Cryptographic hash functions

Cryptography Hash Functions

- Take message, *m*, of arbitrary length and produces a smaller (short) number, *h(m)*
- One-way function
  - Easy to compute *h(m)*
  - Hard to find an *m*, given *h(m)*
  - Often assumed: output of hash fn's "looks" random
- Collision resistance:
  - Strong: Find any m != m' such that h(m) == h(m')
  - Weak: Given m, find m' such that h(m) == h(m')
  - For 160-bit hash (SHA-1)
    - Strong collision are birthday paradox: 2^{160/2} = 2^80
    - Weak collision requires 2^160

Example use #1: Passwords

- Can't store passwords in a file that could be read
  - Concerned with insider attacks!

- Must compare typed passwords to stored passwords
  - Does hash (typed) == hash (password) ?

- Actually, a "salt" is often used: hash (input || salt)

Example use #2: Self-certifying naming

- File-sharing software (LimeWire, BitTorrent)
  - File named by F_{name} = hash (data)
  - Participants verify that hash (downloaded) == F_{name}
    - If check fails, reject data
    - To successfully "forge" data', must find weak collision
- Recursively applied…
  - BitTorrent file has many chunks
  - Control file downloaded from tracker includes:
    - forall chunks, $F_{chunk\ name}$ = hash (chunk)
  - BitTorrent client verifies each individual chunk

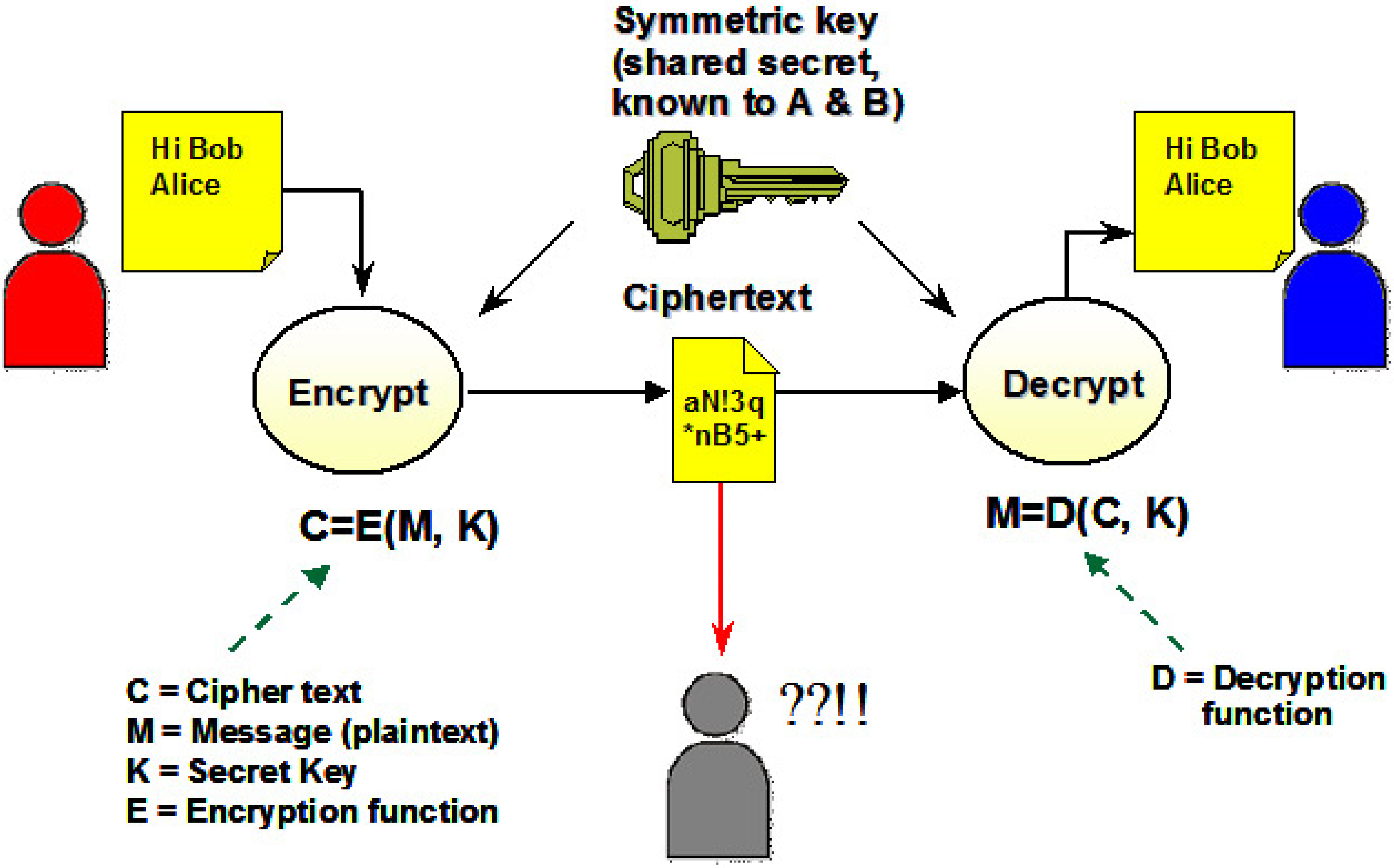
Symmetric (Secret) Key Cryptography

Symmetric Encryption

- Also: "conventional / private-key / single-key"
  - Sender and recipient share a common key
  - All classical encryption algorithms are private-key
  - Dual use: confidentiality or authentication/integrity
    - Encryption vs. msg authentication code (MAC)

- Was only type of encryption prior to invention of public-key in 1970's
  - Most widely used
  - (Much) more computationally efficient than "public key"

Symmetric Cipher Model



Distribution of Symmetric Keys

- Manual delivery is challenging…
- The number of keys grows quadratically with the number of endpoints *(n*(n-1)/2)*
  - Further complexity for application/user level encryption
- Key distribution center (KDC) a good alternative
  - Only *n* master keys required
  - KDC generate session key for Alice and Bob

Public-Key Cryptography

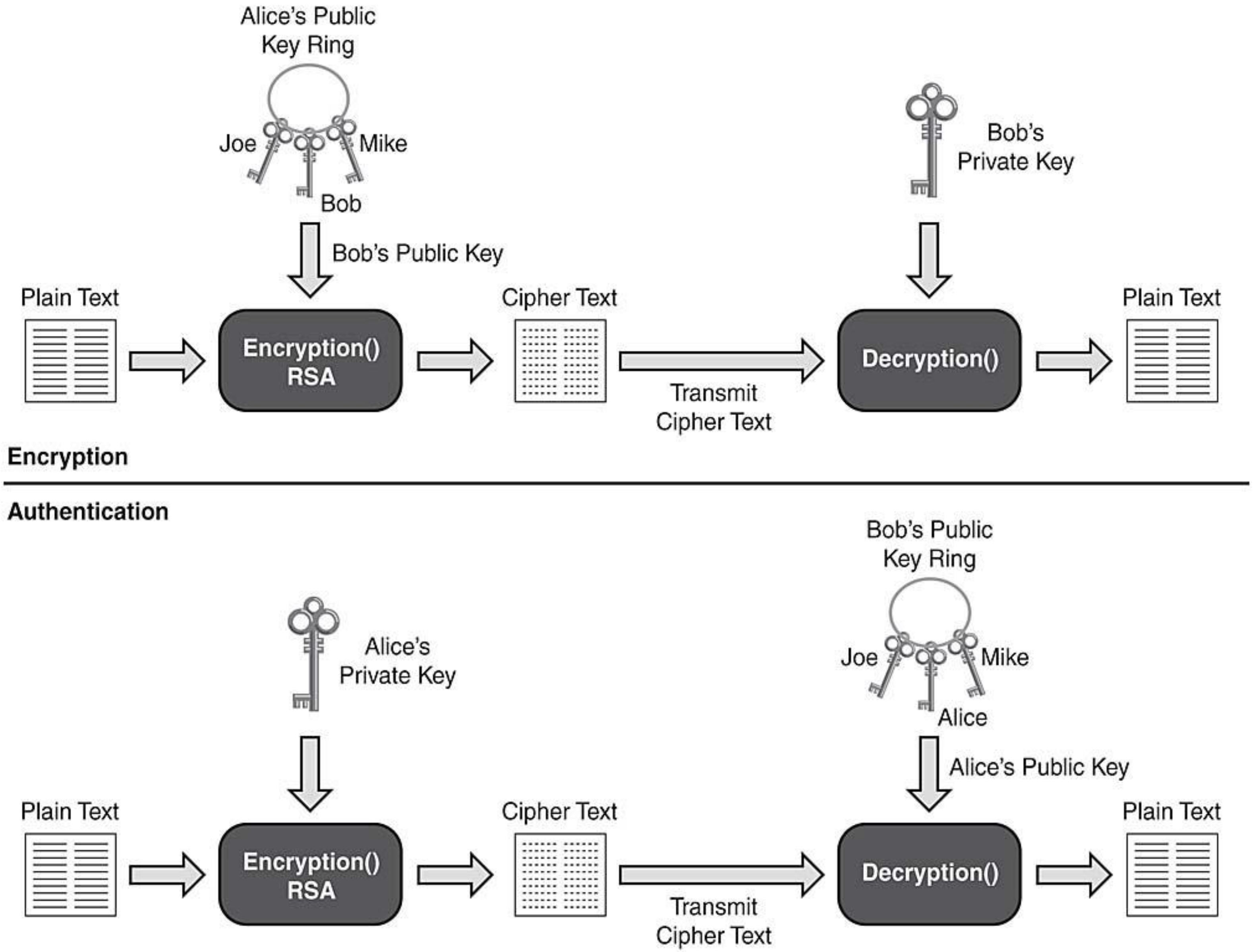
Why Public-Key Cryptography?

- Developed to address two key issues:
  - Key distribution: Secure communication w/o having to trust a key distribution center with your key
  - Digital signatures: Verify msg comes intact from claimed sender (w/o prior establishment)
- Public invention due to Whitfield Diffie & Martin Hellman in 1976
  - Known earlier in classified community

Public-Key Cryptography

- **Public-key:** Known by anybody, and can be used to encrypt messages and verify signatures
- **Private-key:** Known only to recipient, used to decrypt messages and sign (create) signatures
- Can encrypt messages or verify signatures w/o ability to decrypt messages or create signatures

Public-Key Cryptography



Security of Public Key Schemes

- Public-key encryption is a "trap-door" function:
  - Easy to compute $c \leftarrow F(m, k_p)$
  - Hard to compute $m \leftarrow F^{-1}(c)$ without knowing k_s
  - Easy to compute $m \leftarrow F^{-1}(c, k_s)$ by knowing k_s
- Like private key schemes, brute force search possible
  - But keys used are too large (e.g., >= 2048 bits)
  - Hence is slow compared to private key schemes

(Simple) RSA Algorithm

- **Security** due to cost of factoring large numbers
  - Factorization takes $O(e^{\log n \log \log n})$ operations (hard)
  - Exponentiation takes $O((\log n)^3)$ operations (easy)

- To encrypt a message M the sender:
  - Obtain public key `{e,n};` compute `C = M`^{e} `mod n`

- To decrypt the ciphertext C the owner:
  - Use private key `{d,n};` computes `M = C`^{d} `mod n`

- Note that msg M must be smaller than the modulus n
  - Otherwise, hybrid encryption:
    - Generate random symmetric key *r*
    - Use public key encryption to encrypt *r*
    - Use symmetric key encryption under *r* to encrypt *M*

Symmetric vs. Asymmetric

- Symmetric Pros and Cons
  - Simple and really very fast (order of 1000 to 10000 faster than asymmetric mechanisms)
  - Must agree/distribute the key beforehand
  - AES/CBC (256-bit) → 80 MB / s
  - (*for 2048 bits, that's .003 ms)*
- Public Key Pros and Cons
  - Easier predistribution for public keys
    - Public Key Infrastructure (in textbook)
  - Much slower
  - 2048-RSA → 6.1ms Decrypt, 0.16ms Encrypt