Cryptography Tools

CSE 565 - Fall 2025
Computer Security

Hongxin Hu (hongxinh@buffalo.edu)

Updates

- **AI Quiz**
  - Deadline: Monday, September 10
- **Assignment 1**
  - Deadline: Monday, September 16
- **Project 1 Secret-Key Encryption**
  - Deadline: Thursday, September 18
  - **Environment Setup**
    - Mac OS (M1/M2) users please check the new setup document: https://docs.google.com/presentation/d/1EY0cLCB5-1yMwqHT9NS8McxTpZI-gTLzm7Irha3Y8bM/edit?usp=sharing
    - Other users: https://piazza.com/class_profile/get_resource/llmmx2es9cn5pv/lm4e2j8ed3u6w1
- **One question from each project in midterm (projects 1/2) or final exam (projects 3/4/5)**

Cryptography Issues

Confidentiality: only sender, intended receiver should "understand" message contents

- Sender encrypts message
- Receiver decrypts message

End-Point Authentication: sender, receiver want to confirm identity of each other

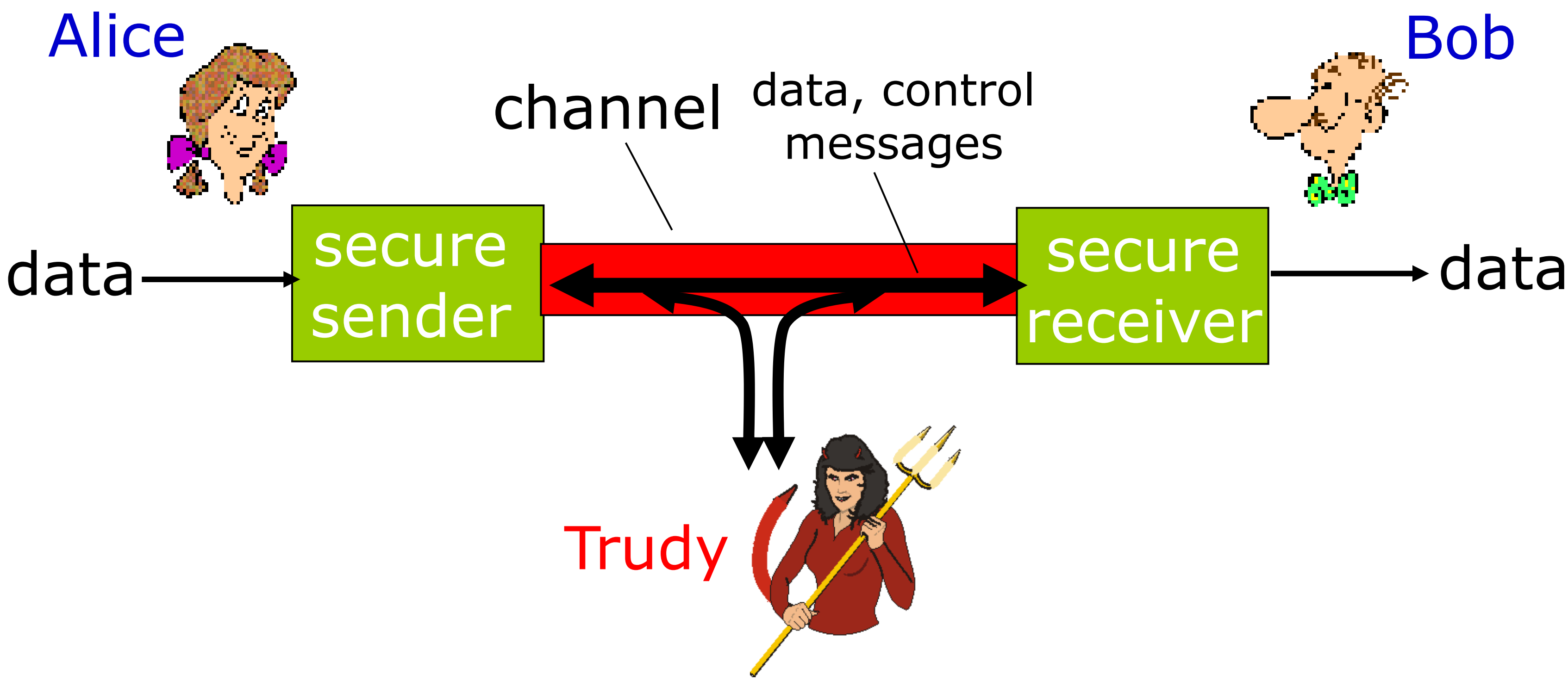
Message Integrity: sender, receiver want to ensure message not changed (in transit, or afterwards) without detection

Cryptography

- Overview
- Symmetric Key Cryptography
- Public Key Cryptography
- Message Integrity and Digital Signatures

Friends and enemies: Alice, Bob, Trudy

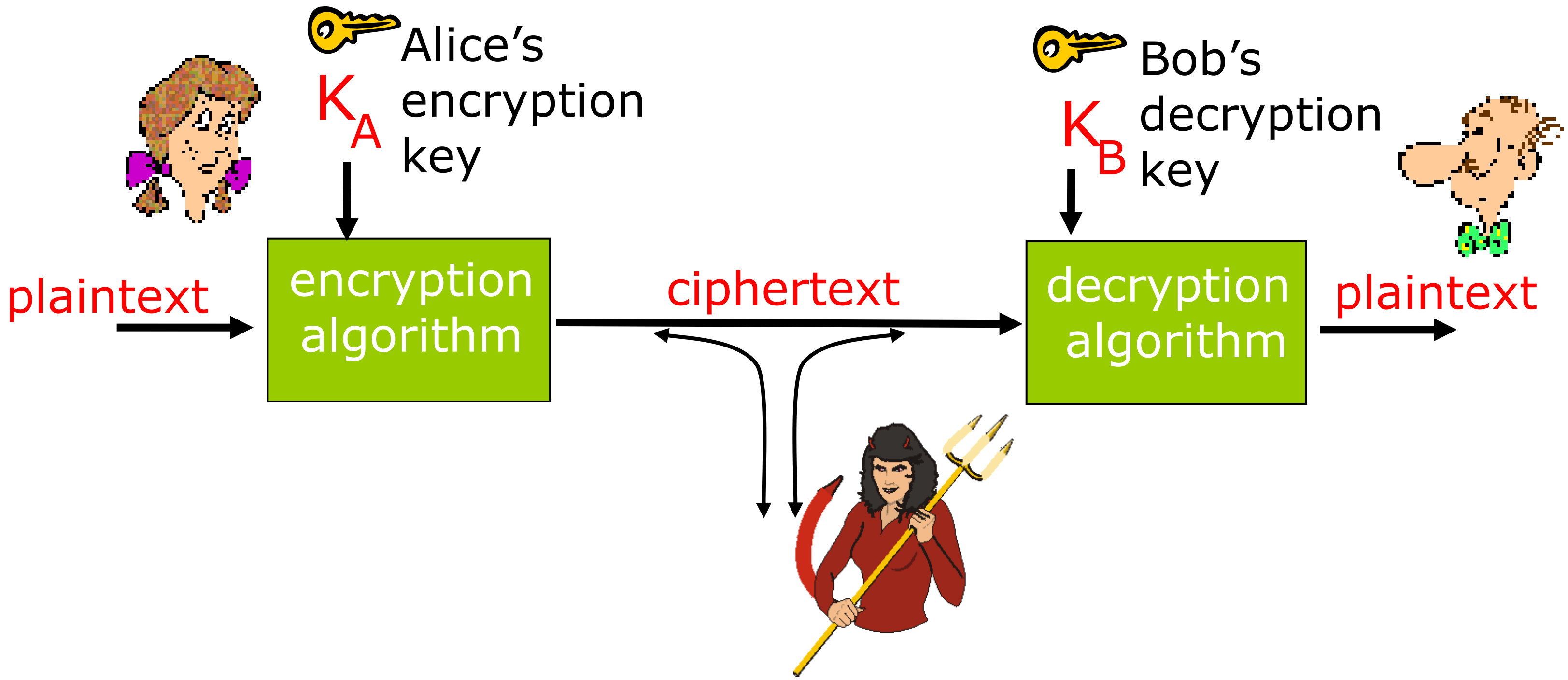
- Well-known in network security world
- Bob, Alice (lovers!) want to communicate “securely”
- Trudy (intruder) may intercept, delete, add messages



Who might Bob, Alice be?

- … well, *real-life* Bobs and Alice!
  - Web browser/server for electronic transactions (e.g., on-line purchases)
  - On-line banking client/server
  - DNS servers
  - Routers exchanging routing table updates

The language of cryptography



m plaintext message

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

$m = K_B(K_A(m))$ decrypted with key K_B

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
```

Key: the **mapping** from the set of 26 letters to the set of 26 letters

Breaking an encryption scheme

- Cipher-text only attack: Trudy only has ciphertext that she can analyze
  - Two approaches:
    - Search through all keys: must be able to differentiate resulting plaintext from gibberish
    - Statistical analysis
- Known-plaintext attack: trudy has some plaintext corresponding to some ciphertext
- Chosen-plaintext attack: trudy can get the cyphertext for some chosen plaintext

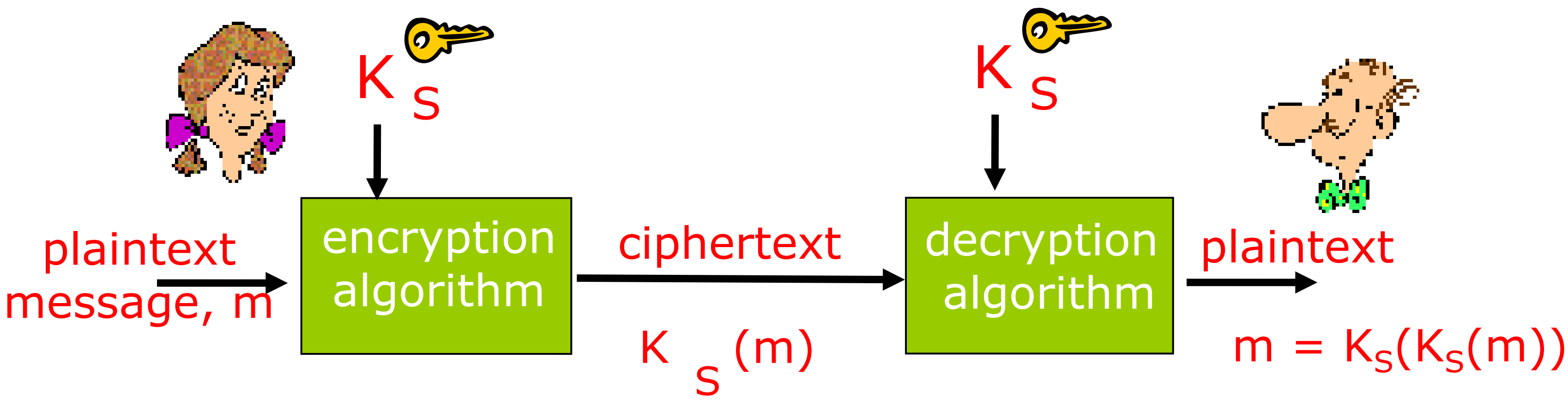
Types of Cryptography

- Crypto often uses keys:
  - Algorithm is known to everyone
  - Only “keys” are secret
- Symmetric key cryptography
  - Involves the use one key
- Public key cryptography
  - Involves the use of two keys
- Hash functions
  - Involves the use of no keys
  - Nothing secret: How can this be useful?

Cryptography

- Overview
- Symmetric Key Cryptography
- Public Key Cryptography
- Message integrity and digital signatures

Symmetric key cryptography



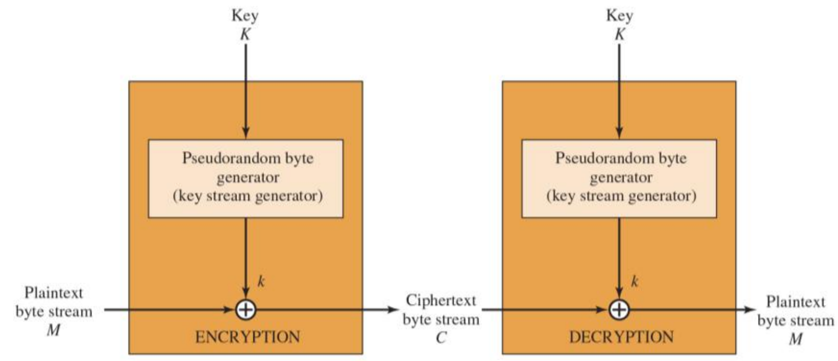
symmetric key crypto: Bob and Alice share same (symmetric) key: K_S

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

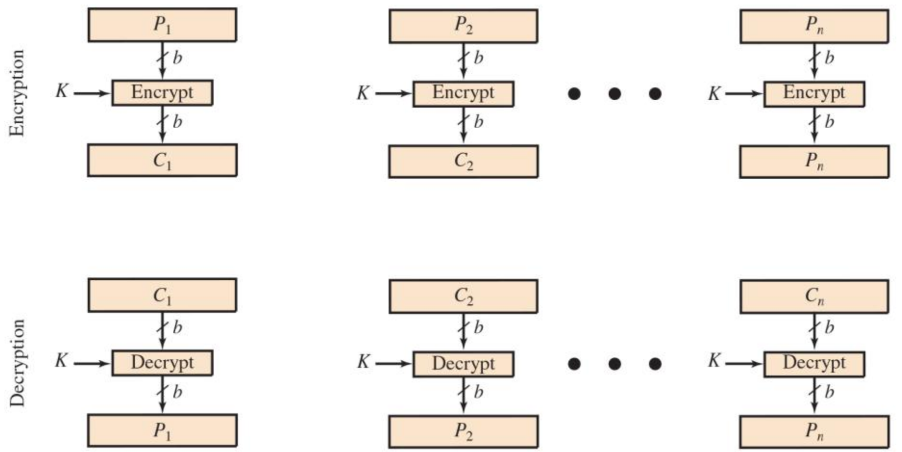
Q: how do Bob and Alice agree on key value?

Two types of symmetric ciphers

- Stream ciphers
  - Encrypt one bit at time
- Block ciphers
  - Break plaintext message in equal-size blocks
  - Encrypt each block as a unit

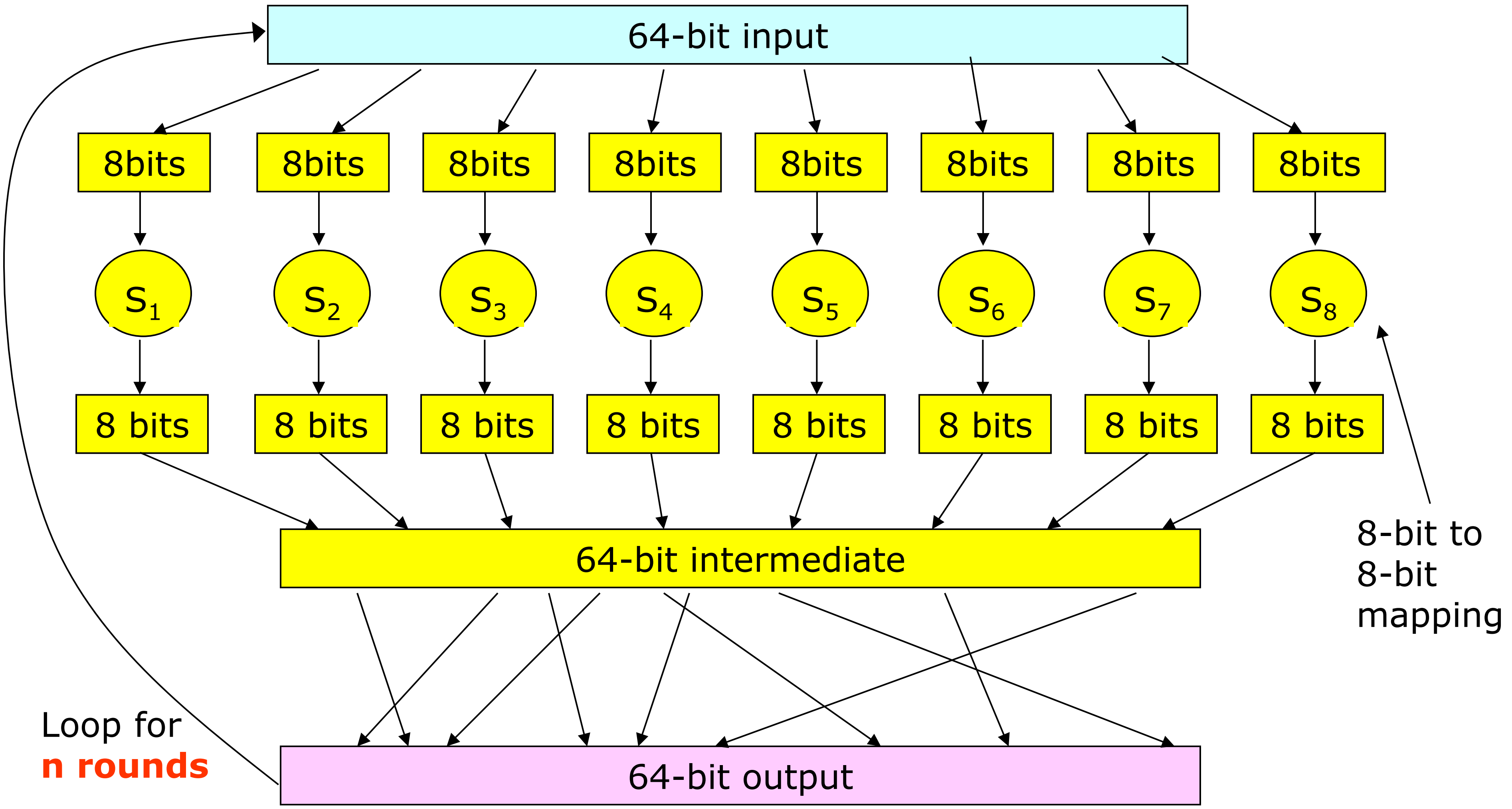


(b) Stream encryption



(a) Block cipher encryption (electronic codebook mode)

Prototype function



Why rounds in prototype?

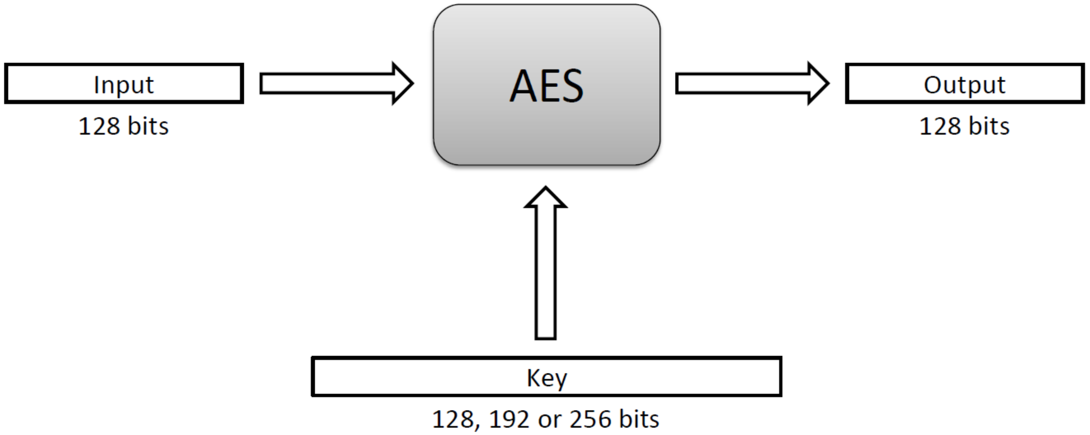
- If only a single round, then one bit of input affects at most 8 bits of output.
- In 2nd round, the 8 affected bits get scattered and inputted into multiple substitution boxes.
- How many rounds?
  - How many times do you need to shuffle cards
  - Becomes less efficient as n increases

Block Ciphers in Practice

- **Data Encryption Standard (DES)**
  - Developed by IBM and adopted by NIST in 1977
  - 64-bit blocks and 56-bit keys
  - Small key space makes exhaustive search attack feasible since late 90s
- **Triple DES (3DES)**
  - Nested application of DES with three different keys KA, KB, and KC
  - Effective key length is 168 bits, making exhaustive search attacks unfeasible
- **Advanced Encryption Standard (AES)**
  - Selected by NIST in 2001 through open international competition and public discussion
  - 128-bit blocks and several possible key lengths: 128, 192 and 256 bits
  - Exhaustive search attack not currently possible
  - AES-256 is the symmetric encryption algorithm of choice
    - E.g., CryptoLocker Virus

Advanced Encryption Standard (AES)

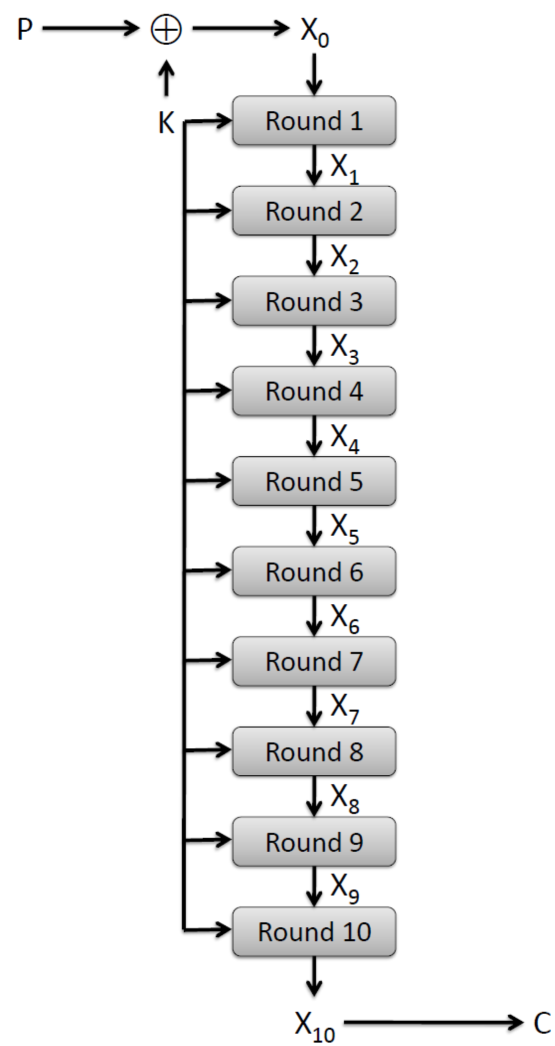
- In 1997, the U.S. National Institute for Standards and Technology (NIST) put out a public call for a replacement to DES.
- It narrowed down the list of submissions to five finalists, and ultimately chose an algorithm that is now known as the **Advanced Encryption Standard** (**AES**).
- AES is a block cipher that operates on 128-bit blocks. It is designed to be used with keys that are 128, 192, or 256 bits long, yielding ciphers known as AES-128, AES-192, and AES-256.



AES Round Structure

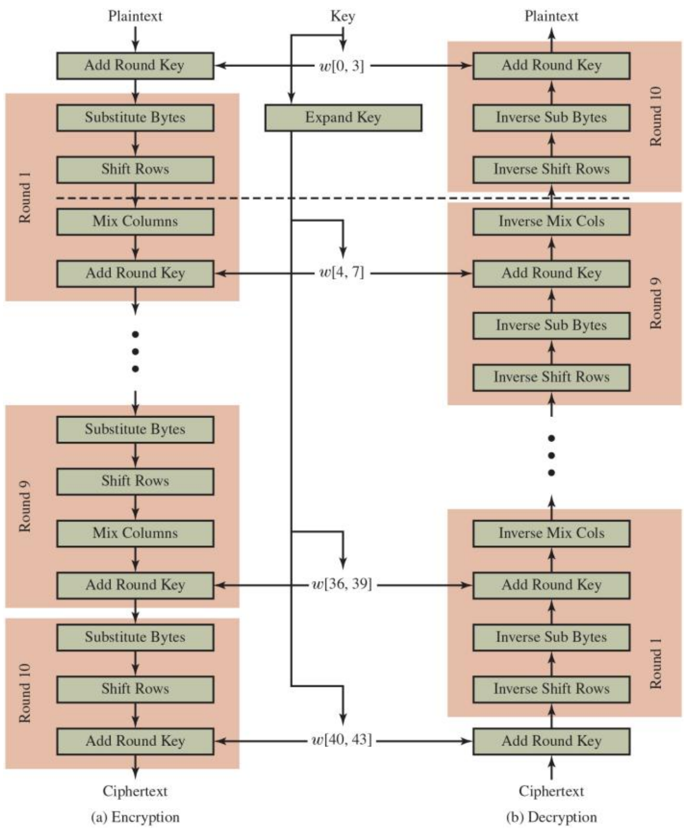
- The 128-bit version of the AES encryption algorithm proceeds in ten rounds.
- Each round performs an invertible transformation on a 128-bit array, called **state**.
- The initial state X_0 is the XOR of the plaintext P with the key K:

  X_0 = P XOR K.
- Round i (i = 1, …, 10) receives state X_{i-1} as input and produces state X_i.
- The ciphertext C is the output of the final round: $C = X_{10}$.



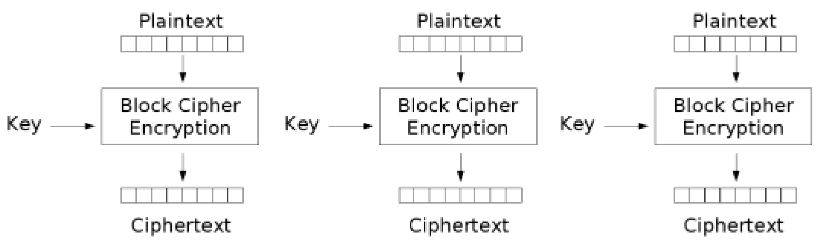
AES Rounds

- Each round is built from four basic steps:
  1. **SubBytes step**: an S-box substitution step
  2. **ShiftRows step**: a permutation step
  3. **MixColumns step**: a matrix multiplication step
  4. **AddRoundKey step**: an XOR step with a **round key** derived from the 128-bit encryption key

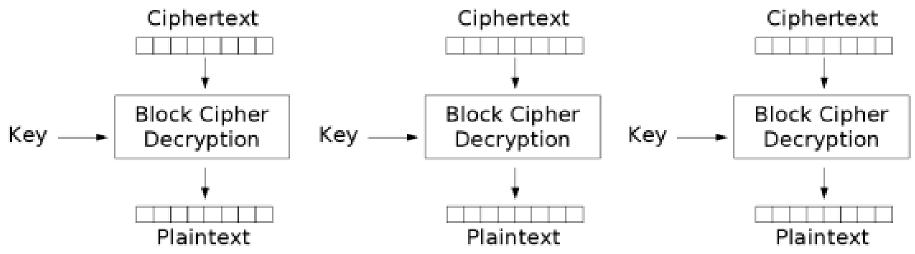


Block Cipher Modes

- A block cipher mode describes the way a block cipher encrypts and decrypts a sequence of message blocks.
- Electronic Code Book (ECB) Mode (is the simplest):
  - Block P[i] encrypted into ciphertext block C[i] = E_K(P[i])
  - Block C[i] decrypted into plaintext block M[i] = D_K(C[i])



Electronic Codebook (ECB) mode encryption



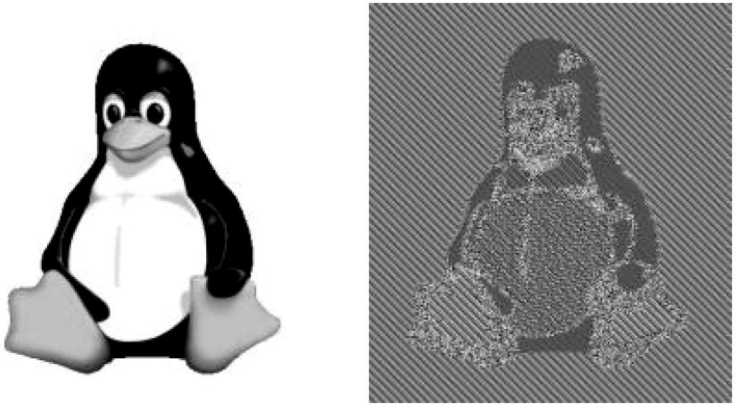
Electronic Codebook (ECB) mode decryption

Public domain images from http://en.wikipedia.org/wiki/File:Ecb_encryption.png and http://en.wikipedia.org/wiki/File:Ecb_decryption.png

Strengths and Weaknesses of ECB

- Strengths:
  - Is very simple
  - Allows for parallel encryptions of the blocks of a plaintext
  - Can tolerate the loss or damage of a block

- Weakness:
  - Documents and images are not suitable for ECB encryption since patterns in the plaintext are repeated in the ciphertext:

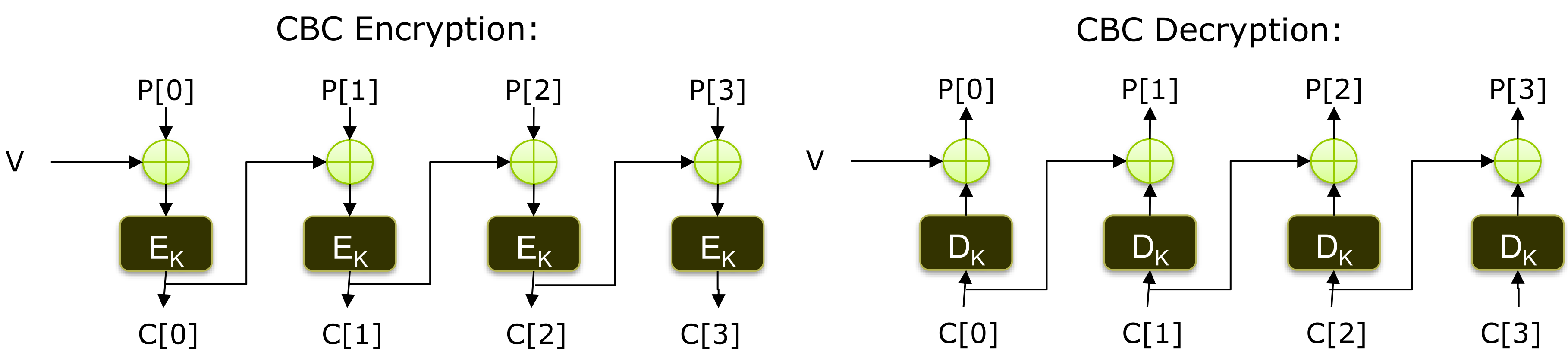


(a) (b)

Figure 8.6: How ECB mode can leave identifiable patterns in a sequence of blocks: (a) An image of Tux the penguin, the Linux mascot. (b) An encryption of the Tux image using ECB mode. (The image in (a) is by Larry Ewing, lewing@isc.tamu.edu, using The Gimp; the image in (b) is by Dr. Juzam. Both are used with permission via attribution.)

Cipher Block Chaining (CBC) Mode

- In Cipher Block Chaining (CBC) Mode
  - The previous ciphertext block is combined with the current plaintext block $C[i] = E_K(C[i-1] \oplus P[i])$
  - $C[-1] = V$, a random block separately transmitted encrypted - known as the Initialization Vector (IV)
  - Decryption: $P[i] = C[i-1] \oplus D_K(C[i])$



Strengths and Weaknesses of CBC

- Strengths:
  - Doesn't show patterns in the plaintext
  - Is the most common mode
- Weaknesses:
  - CBC requires the reliable transmission of all the blocks sequentially
  - CBC is not suitable for applications that allow packet losses (e.g., music and video streaming)

Java AES Encryption Example

- Source

  http://java.sun.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html

- Generate an AES key

```
KeyGenerator keygen = KeyGenerator.getInstance("AES");
SecretKey aesKey = keygen.generateKey();
```

- Create a cipher object for AES in ECB mode and PKCS5 padding

```
Cipher aesCipher;
aesCipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
```

- Encrypt

```
aesCipher.init(Cipher.ENCRYPT_MODE, aesKey);
byte[] plaintext = "My secret message".getBytes();
byte[] ciphertext = aesCipher.doFinal(plaintext);
```

- Decrypt

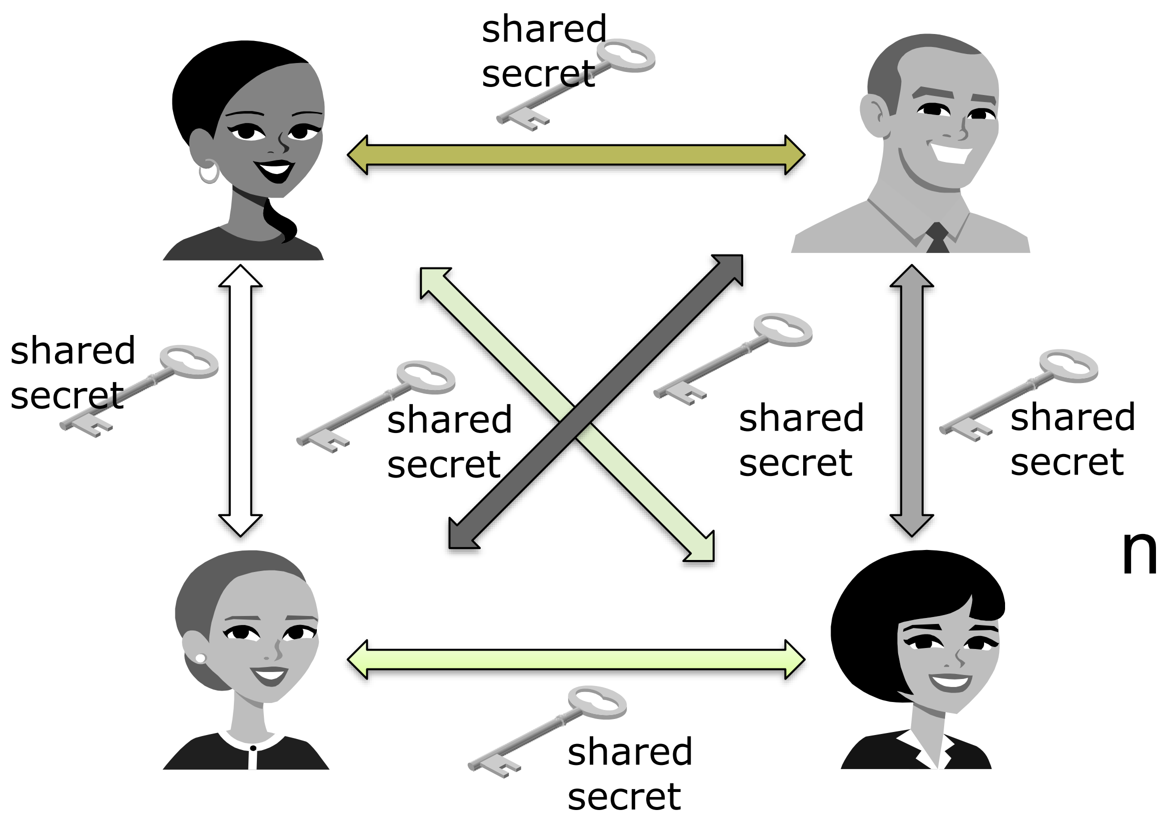
```
aesCipher.init(Cipher.DECRYPT_MODE, aesKey);
byte[] plaintext1 = aesCipher.doFinal(ciphertext);
```

Cryptography

- Overview
- Symmetric Key Cryptography
- Public Key Cryptography
- Message integrity and digital signatures

Symmetric Key Distribution

- Requires each pair of communicating parties to share a (separate) secret key.



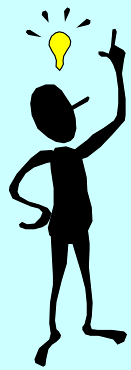
n (n–1)/2 keys

Public Key Cryptography

Symmetric key crypto

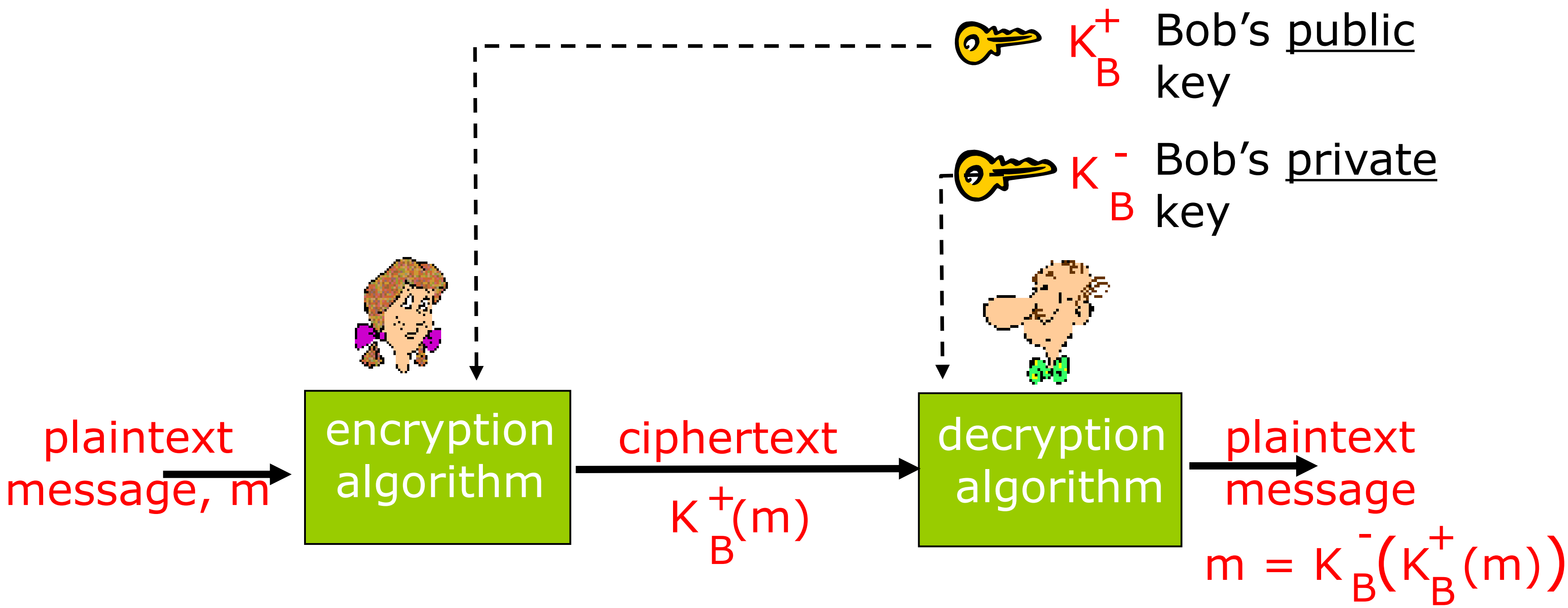
- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never "met")?
  - Challenging for Distributed systems

Public key cryptography



- 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



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: another important property

The following property will be *very* useful:

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

Result is the same!

Session keys

- Exponentiation (RSA) is computationally intensive
- DES is at least **100** times faster than RSA

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

Cryptography

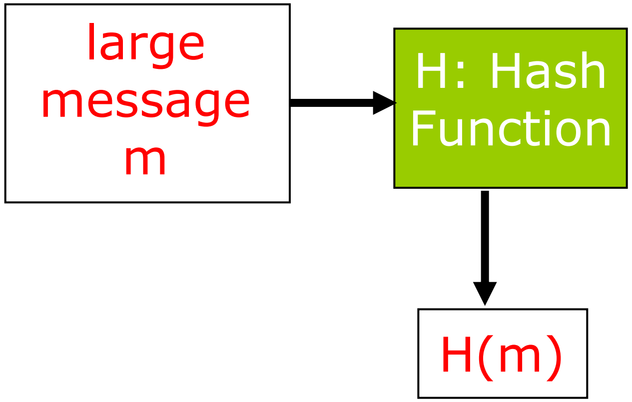
- Overview
- Symmetric Key Cryptography
- Public Key Cryptography
- Message integrity and digital signatures

Message Integrity

- Allows communicating parties to verify that received messages are authentic
  - Content of message has not been changed
  - Sequence of messages is maintained

Message Digests

- Function H( ) that takes as input an arbitrary length message and outputs a fixed-length string: "message signature"
- Note that H( ) is a many-to-1 function
- H( ) is often called a "hash function"



- Desirable properties:
  - Easy to calculate
  - Irreversibility: Can't determine m from H(m)
  - Collision resistance: Computationally difficult to produce m and m' such that H(m) = H(m')
  - Seemingly random output

Hash Function Algorithms

- MD5 hash function widely used (RFC 1321)
  - computes 128-bit message digest in 4-step process.
- SHA-1 is also used.
  - US standard [NIST, FIPS PUB 180-1]
  - 160-bit message digest

Digital Signatures

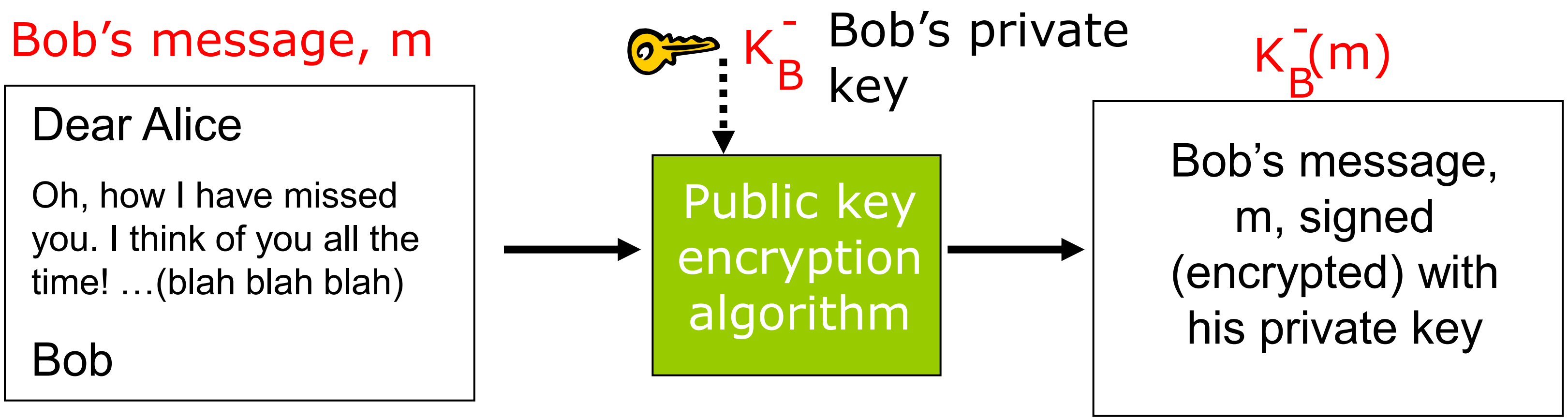
Cryptographic technique analogous to hand-written signatures.

- Sender (Bob) digitally signs document, establishing he is document owner/creator.
- Goal is similar to that of a MAC, except now use public-key cryptography
- verifiable: recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document

Digital Signatures

Simple digital signature for message m:

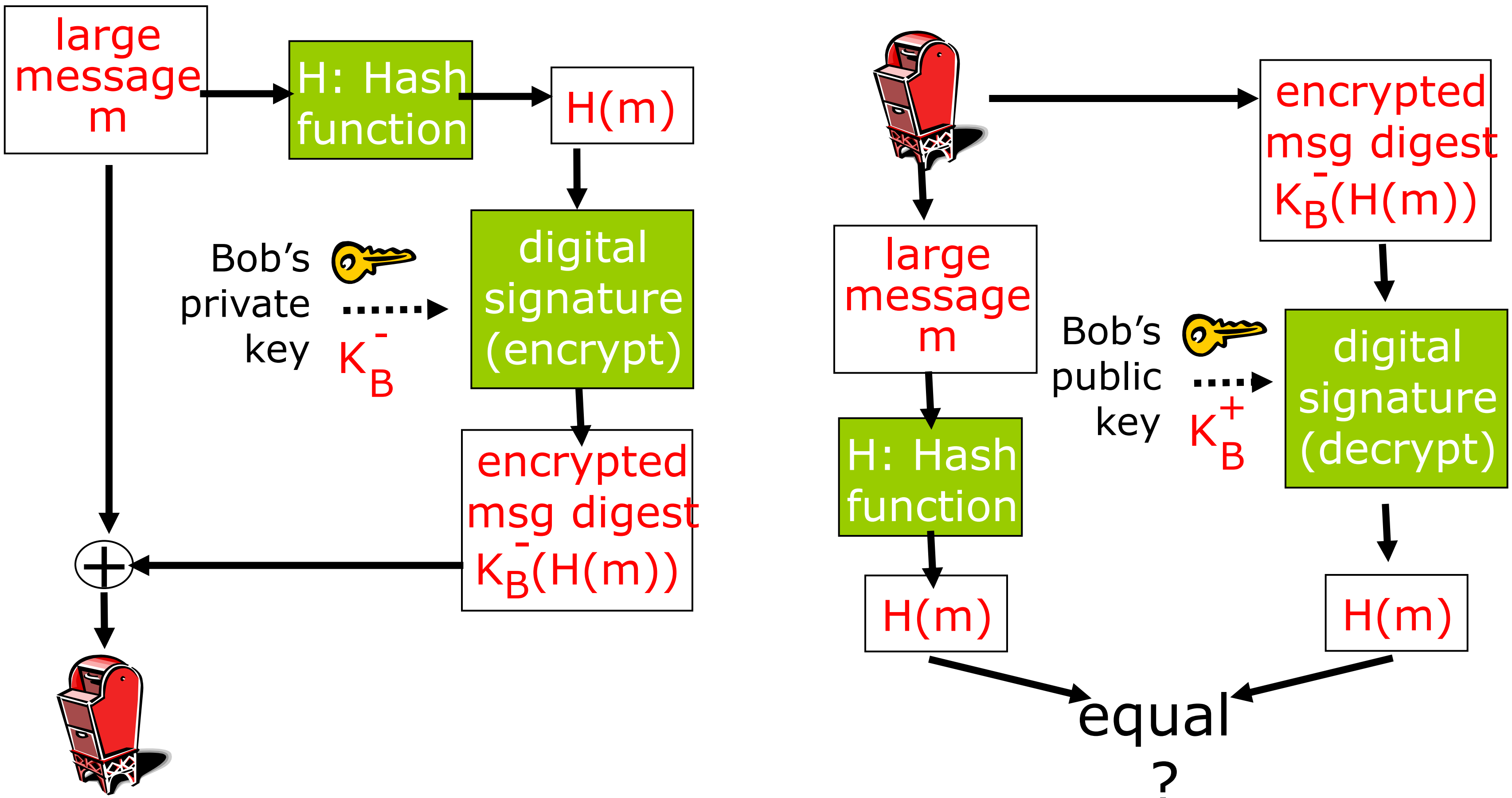
- Bob signs m by encrypting with his private key K_B^-, creating "signed" message, $K_B^-(m)$



Digital signature = signed message digest

Bob sends digitally signed message:

Alice verifies signature and integrity of digitally signed message:



Digital Signatures (more)

- Suppose Alice receives msg m, digital signature $K_B^-(m)$
- Alice verifies m signed by Bob by applying Bob's public key K_B^+ to $K_B^-(m)$ then checks $K_B^+(K_B^-(m)) = m$.
- If $K_B^+(K_B^-(m)) = m$, whoever signed m must have used Bob's private key.

Alice thus verifies that:

- ✓ Bob signed m.
- ✓ No one else signed m.
- ✓ Bob signed m and not m'.

Non-repudiation:

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

Summary

- **Three types of cryptography: secret-key, public key, and hash function**

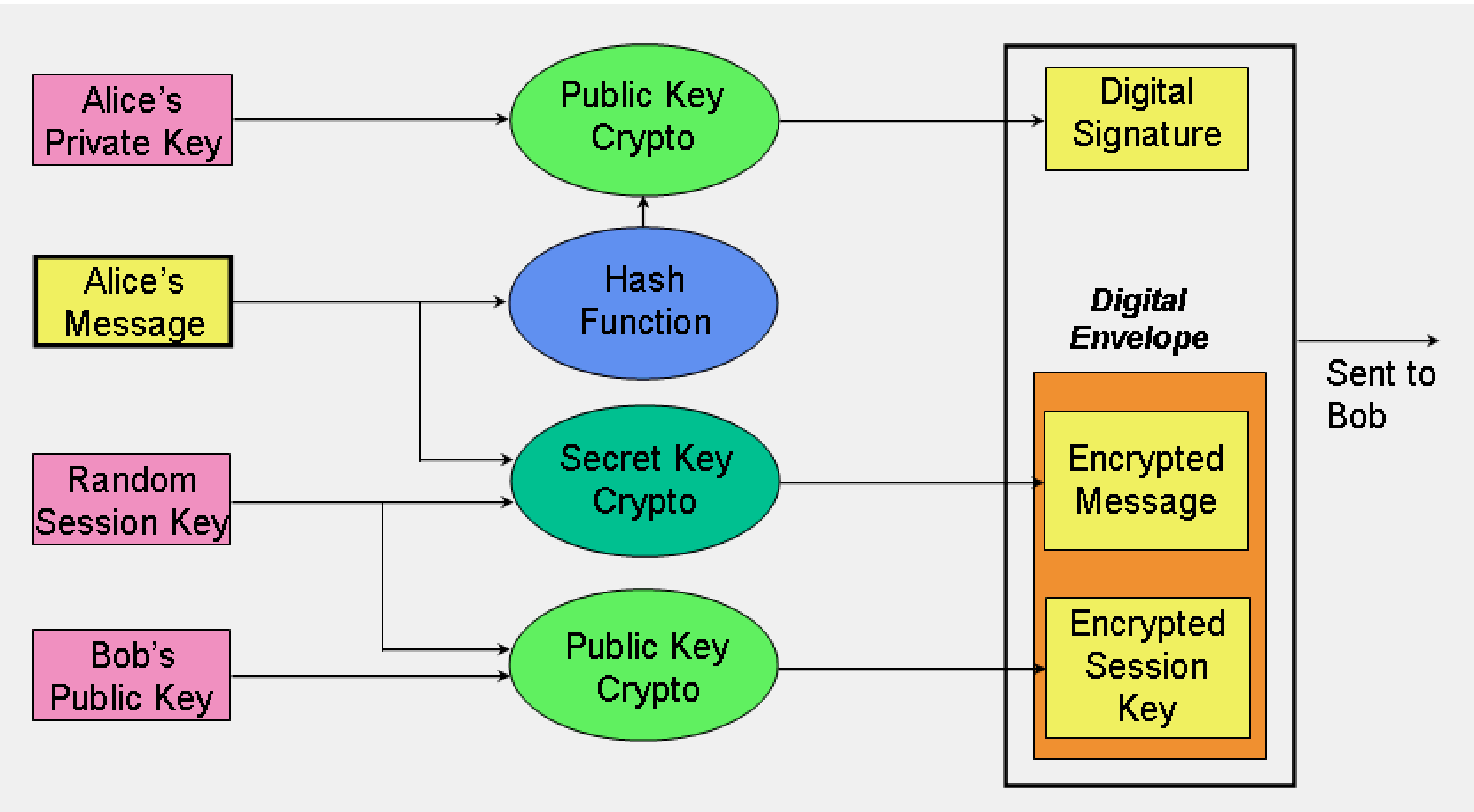
A) Secret key (symmetric) cryptography. SKC uses a single key for both encryption and decryption.

B) Public key (asymmetric) cryptography. PKC uses two keys, one for encryption and the other for decryption.

C) Hash function (one-way cryptography). Hash functions have no key since the plaintext is not recoverable from the ciphertext.

Summary

- **Application of the three cryptographic techniques for secure communication**



Summary

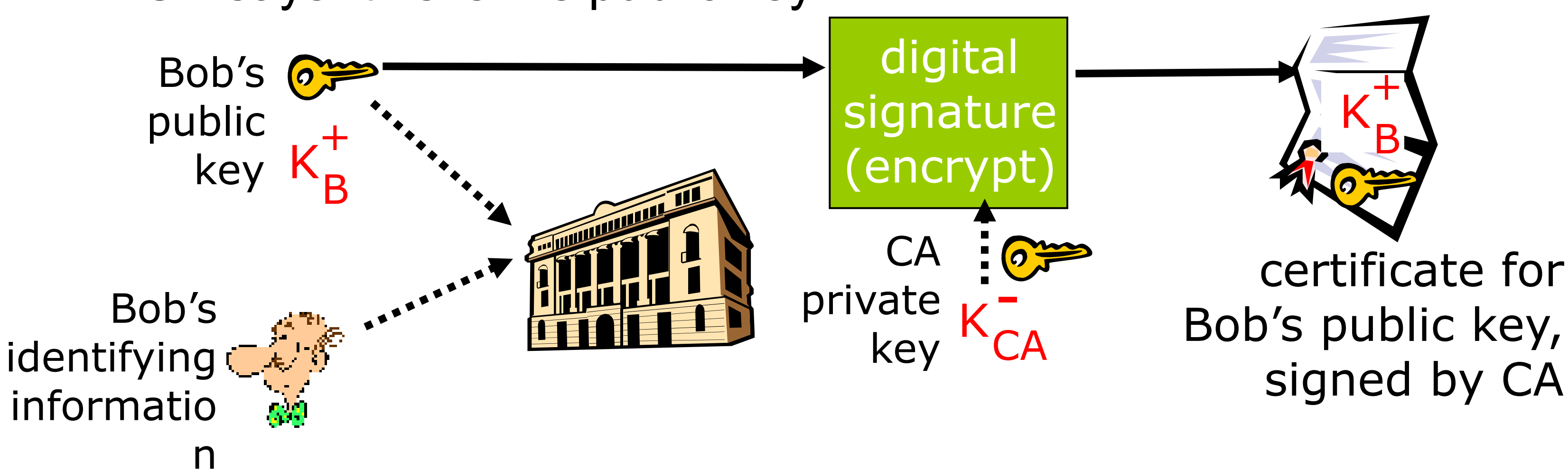
- **Application of the three cryptographic techniques for secure communication**
  - Confidentiality
    - Encrypted message
  - End-Point Authentication (Both Alice and Bob)
    - Secure Key exchange: only Bob can decrypt session key
    - Digital signature: decrypting the digital signature with Alice's public key
      - Message was sent by Alice
  - Message Integrity
    - Hash value of her message

Public-key certification

- Motivation: Trudy plays pizza prank on Bob
  - Trudy creates e-mail order:
    Dear Pizza Store, Please deliver to me four pepperoni pizzas. Thank you, Bob
  - Trudy signs order with her private key
  - Trudy sends order to Pizza Store
  - Trudy sends to Pizza Store her public key, but says it's Bob's public key.
  - Pizza Store verifies signature; then delivers four pizzas to Bob.
  - Bob doesn't even like Pepperoni

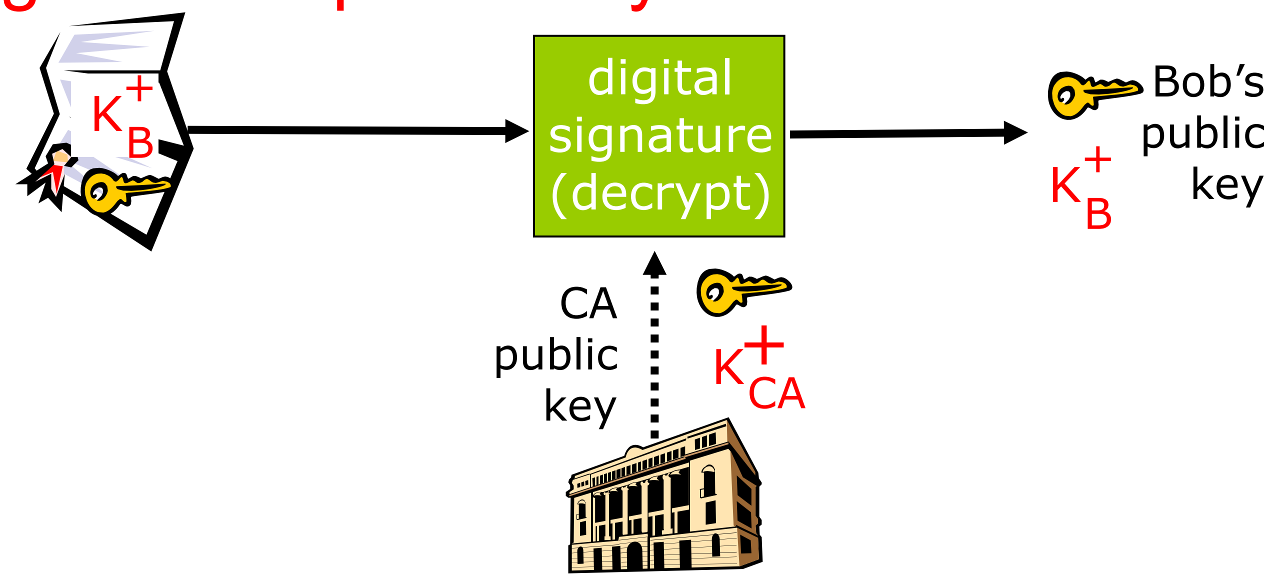
Certification Authorities

- Certification authority (CA): binds public key to particular entity, E.
- E (person, router) registers its public key with CA.
  - E provides "proof of identity" to CA.
  - CA creates certificate binding E to its public key.
  - certificate containing E's public key digitally signed by CA – CA says "this is E's public key"



Certification Authorities

- When Alice wants Bob's public key:
  - gets Bob's certificate (Bob or elsewhere).
  - apply CA's public key to Bob's certificate, get Bob's public key



Certificates: summary

- Primary standard X.509 (RFC 2459)
  - Certificate contains:
    - Issuer name
    - Entity name, address, domain name, etc.
    - Entity's public key
    - Digital signature (signed with issuer's private key)
- Public-Key Infrastructure (PKI)
  - Certificates and certification authorities
  - Often considered "heavy"

Questions?

Attacking Symmetric Encryption

Cryptanalytic Attacks

- Rely on:
  - Nature of the algorithm
  - Some knowledge of the general characteristics of the plaintext
  - Some sample plaintext-ciphertext pairs
- Exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or the key being used
  - If successful, all future and past messages encrypted with that key are compromised

Brute-Force Attacks

- Try all possible keys on some ciphertext until an intelligible translation into plaintext is obtained
  - On average half of all possible keys must be tried to achieve success

AES

- simple design but resistant to known attacks
- very efficient on a variety of platforms including 8-bit and 64-bit platforms
- highly parallelizable
- had the highest throughput in hardware among all AES candidates
- well suited for restricted-space environments (very low RAM and ROM requirements)
- optimized for encryption (decryption is slower)

Average Time Required for Exhaustive Key Search

Key Size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 decryptions / μs	Time Required at 10^{13} decryptions / μs
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	$2^{55}\ \mu\text{s} = 1.125$ years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	$2^{127}\ \mu\text{s} = 5.3 \times 10^{21}$ years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	$2^{167}\ \mu\text{s} = 5.8 \times 10^{33}$ years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	$2^{191}\ \mu\text{s} = 9.8 \times 10^{40}$ years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	$2^{255}\ \mu\text{s} = 1.8 \times 10^{60}$ years	1.8×10^{56} years