Cryptographic Building Blocks

Reading: Ch. 2, Van Oorschot

Furkan Alaca

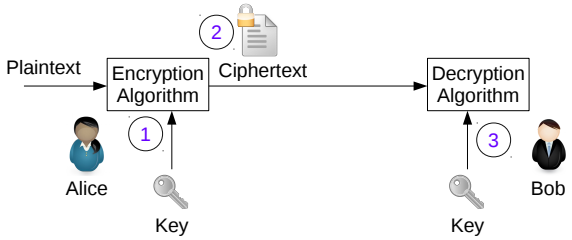
University of Toronto Mississauga

CSC347H5, Fall 2018

Learning Objectives

- Overview of common cryptographic operations
- Different categories of cryptographic algorithms
- Important properties/characteristics of cryptographic operations
- How to **use** cryptography (more on this throughout the course)
- Some practical challenges

Symmetric-Key Encryption



- An encryption algorithm transforms a plaintext message into ciphertext
- Ciphertext should be **meaningless** to any eavesdropping party without possession of the corresponding key
- In symmetric-key encryption, both encryption and decryption are done with a **secret key** shared between the sender and receiver

Symmetric Encryption: Requirements

There are two requirements for the secure use of symmetric encryption:

1. A strong encryption algorithm: An attacker should be unable to decrypt ciphertext or discover the key even when in possession of a collection of ciphertext-plaintext mappings. There are two general attack categories which the algorithm must be secure against:
   - **Cryptanalysis:** Tries to exploit the characteristics of the algorithm to deduce the plaintext or key
   - **Brute-force attacks:** Tries to decrypt the ciphertext using all possible keys, until an intelligible translation into plaintext is obtained
2. The sender and receiver must have obtained copies of the secret key in a secure fashion, and must keep the key secure.

Symmetric Encryption: Algorithms

Block Ciphers (e.g., AES; older schemes: DES, 3DES)

- Processes the input one block (e.g., 64 bits for DES, 128 bits for AES) at a time
- Produces an equal-sized output block for each input block
- Can reuse keys
- More common

Stream Ciphers (e.g., ChaCha20; older scheme: RC4)

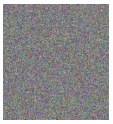
- Processes a stream of input elements, typically one byte at a time
- Simpler and faster than block ciphers
- Performs XOR of input data with a pseudorandom stream
- Pseudorandom stream appears random to an observer without knowledge of the secret key

Symmetric Encryption: Block Cipher Modes of Operation

- The **mode of operation** specifies how to repeatedly apply a block cipher algorithm to encrypt a message which is larger than the size of one block
- **Electronic Code Book (ECB)** divides the message into blocks and encrypts each block separately. Not recommended (see **(b)** on the right)
- **Cypher-Block Chaining (CBC)** XORs each block of plaintext with the previous block of ciphertext before encrypting it
- **Counter (CTR)** turns a block cipher into a stream cipher

(a) Original image

(b) Encrypted with ECB mode



(c) Other modes (e.g., CBC) result in pseudorandomness

Source of images: Wikipedia. Picture of Tux (Linux mascot) created by Larry Ewing.

Symmetric Encryption: CBC Mode of Operation

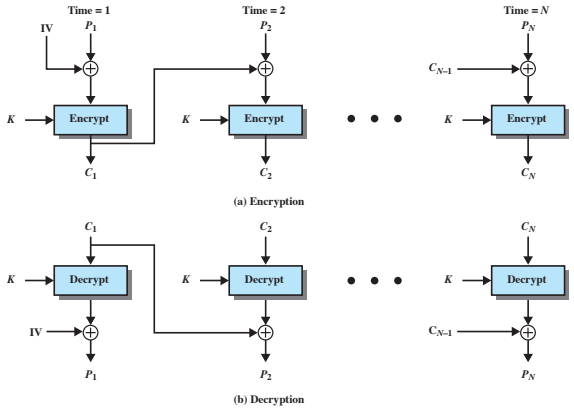


Figure 20.6 Cipher Block Chaining (CBC) Mode

Symmetric Encryption: CTR Mode of Operation

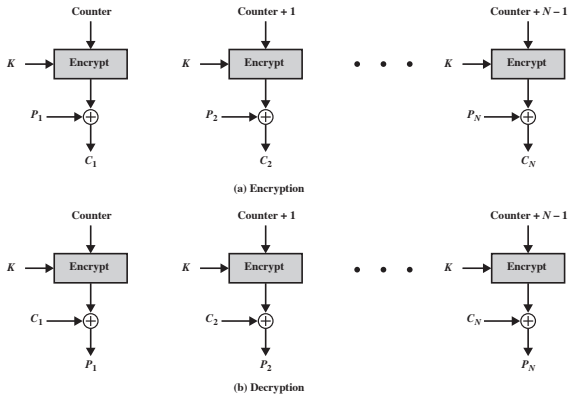


Figure 20.8 Counter (CTR) Mode

Symmetric Encryption: Exhaustive Key Search

Table 2.2 Average Time Required for Exhaustive Key Search

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

Age of the universe: 13.8 billion (1.38×10^{10}) years

Random and Pseudorandom Numbers

- Many cryptographic algorithms rely on random numbers
- Traditional requirements for randomness:
  - Uniform distribution: Frequency of occurrence of each number should be approximately equal
  - Independence: No value in the sequence can be inferred from the other
- Pseudorandom numbers: Not truly random
- **Cryptographically secure** pseudorandom number generators: Infeasable to predict
- True random number generators rely on obtaining measurements from nondeterministic sources such as radiation or thermal noise

Cryptographic Hash Functions

- A hash function H accepts an input message m of any size, and produces a fixed-length output $h = H(m)$
- Cryptographic hash functions must possess the following properties:
  - **Pre-image resistant (one-way):** Given any h, it is infeasible to find m such that $H(m) = h$
  - **Collision-resistant:** Infeasible to find a distinct pair m_1 and m_2 such that $H(m_1) = H(m_2)$
  - **Second preimage resistant (weak collision resistant):** Given m_1, infeasible to find a distinct m_2 such that $H(m_1) = H(m_2)$
- Ideally, a hash function should be computationally efficient
- Applications:
  - Password storage: Protects confidentiality
  - Intrusion detection: Store hash values of files to detect tampering
  - Can be used in message authentication codes and digital signatures
- Algorithms: SHA-3, SHA-2 (SHA-256, SHA-384, SHA-512)
  - MD5, SHA, SHA-1: Not recommended

Hash Collisions

In Feb. 2017, Google used the "SHAttered" attack to construct 2 distinct PDF files that have the same SHA-1 hash. The attack required the equivalent of 1 year of computation time with 110 GPUs. The attack is about 100,000 times faster than a brute-force attack.

MD5 has long been known to be insecure, but check out this blog post from Oct. 2014 by Nathaniel McHugh, showing how he modified the below two images to produce the same MD5 hash: It cost 65 cents for 10 hours of computation time on an Amazon Web Services GPU instance.

Source: `http://natmchugh.blogspot.ca/2014/10/how-i-created-two-images-with-same-md5.html`


Message Authentication

- Encryption protects against passive attacks (eavesdropping)
- **Message authentication** protects against active attacks (falsification of data)
- A message is **authentic** if (1) it came from its claimed source and (2) its contents (or possibly its timeliness) have not been altered
- Encryption on its own is not sufficient for message authentication, e.g., blocks could be re-ordered or purposely delayed
- Some applications where confidentiality is not needed may require message authentication
- **Message Authentication Codes (MACs)** are the most popular mechanism for message authentication, and can be computed in various ways:
  - Encrypt the message with a block cipher and use the last few bytes
  - Encrypt the hash of the message
  - Hash the message concatenated with secret key (both prefix and suffix)
  - HMAC: Also relies on hash functions, but involves more steps

MAC Example with Keyed Hash Function

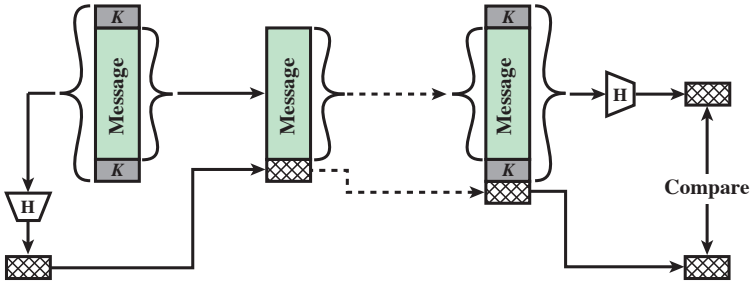
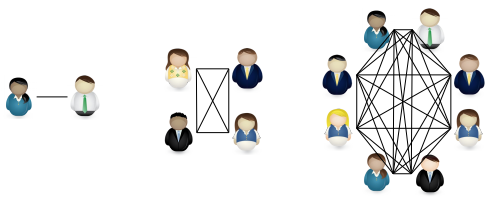


Figure 2.5c: Message authentication using a cryptographic hash function with a secret key

Symmetric-Key Encryption Challenges

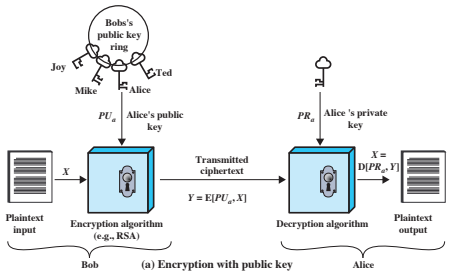
- Symmetric key encryption requires a secret key to be securely agreed upon **in advance** between any two communicating parties
- Each pair of communicating parties needs to establish a unique shared secret



1 key 6 keys 28 keys

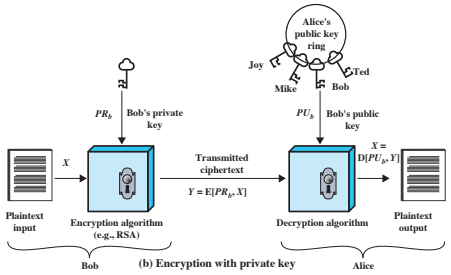
Public-Key Cryptography: Encryption

- Each party requires a **private key** and a **public key**
- Relies on mathematical functions rather than simple bit operations
- For **confidentiality**: The sender encrypts the message with the receiver's public key, and the receiver decrypts the ciphertext with its own private key:



Public-Key Cryptography: Encryption (2)

- For **authentication** and/or **data integrity**: The sender encrypts the message with its private key, and the receiver decrypts the ciphertext with the sender's public key:



- Public-key cryptography can also be used to establish a shared secret key for subsequent symmetric encryption of messages
  - Symmetric-key encryption is much more computationally efficient than public-key encryption

Public-Key Cryptography: Requirements

- It is computationally easy for a party B to generate a public-private key pair
- It is computationally easy for a sender A, knowing the receiver's public key, to encrypt a message using the public key
- It is computationally easy for the receiver B to decrypt the received ciphertext using its private key
- It is computationally infeasible for an attacker with knowledge of a public key to determine its corresponding private key
- It is computationally infeasible for an attacker, knowing a public key and the ciphertext of a message encrypted with that public key, to recover the original message
- Either of the two related keys can be used for encryption, with the other used for decryption

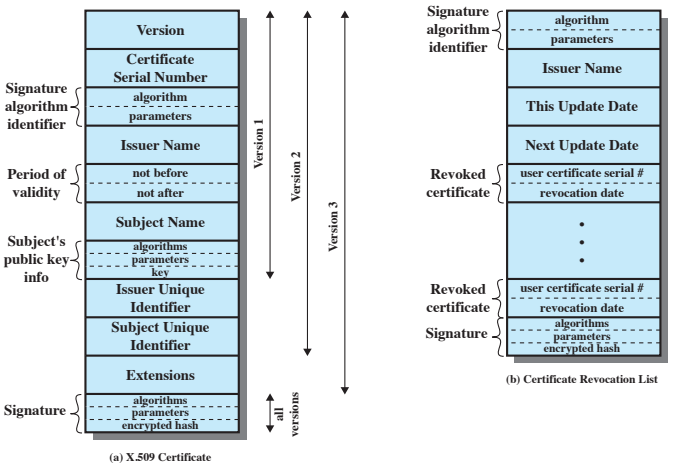
Public-Key Cryptography: Digital Signatures

- The receiver may want to verify the source and data integrity of a message
  - e.g., e-mails, documents, software updates
- The sender must sign the message with their private key, and the receiver must verify the signature with the sender's public key
- Signing a message typically involves encrypting its cryptographic hash with the sender's private key
- Does not provide confidentiality

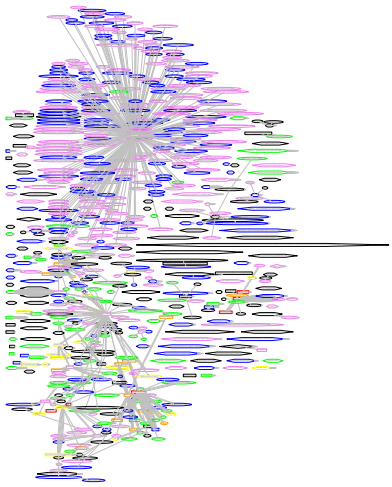
Public-Key Certificates

- A mechanism is needed for the two communicating parties to exchange public keys
  1. Meet in person to exchange or validate keys, or
  2. Rely on a trusted third-party to certify and validate keys
- A certificate consists of a public key, the identification of the key owner, and some additional information such as the validity period
- The certificate is signed by a **trusted third party** – typically, a **Certificate Authority (CA)**
- The user must be in possession of the CA's public key in order to verify the signature

X.509 Certificates: Structure



Graph of 650 CAs Trusted by Mozilla and Microsoft



Source: EFF SSL Observatory

Public-Key Cryptosystems

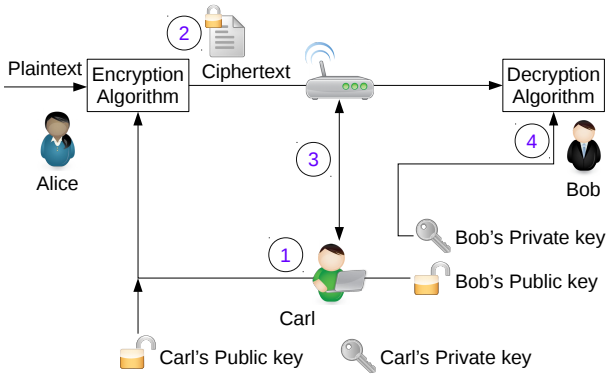
Table 2.2 Applications for Public-Key Cryptosystems

Algorithm	Digital Signature	Symmetric Key Distribution	Encryption of Secret Keys
RSA	Yes	Yes	Yes
Diffie-Hellman	No	Yes	No
DSS	Yes	No	No
Elliptic Curve	Yes	Yes	Yes

Diffie-Hellman Key Exchange

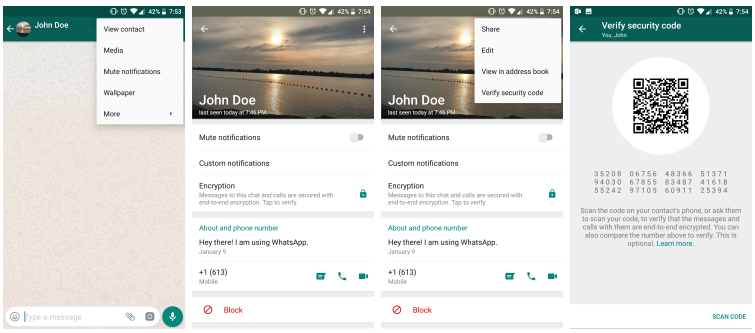
- Provides **perfect forward secrecy:** If an eavesdropper records all network traffic, they will be unable to decrypt the data even in the event of a future compromise of the server's private key
- Simple example to illustrate the concept (adapted from Wikipedia):
  - Alice and Bob agree to use a prime number $p = 23$ and base $g = 5$
  - Alice chooses a secret integer $a = 6$, and sends Bob:
    $$A = 5^6 \mod 23 = 8$$
  - Bob chooses a secret integer $b = 15$, and sends Alice:
    $$B = 5^{15} \mod 23 = 19$$
  - Alice computes the shared secret:
    $$s = 19^6 \mod 23 = 2$$
  - Bob computes the shared secret:
    $$s = 8^{15} \mod 23 = 2$$
- In practice, variations of the Diffie-Hellman key exchange are used which digitally sign the exchanged messages in order to protect **integrity** and **authenticity**
  - No need to protect **confidentiality** of the messages, since the shared key is never transmitted over the network

Man-In-The-Middle Attacks



- A man-in-the-middle (MITM) attack involves intercepting messages (e.g., via an insecure WiFi network) between two hosts

Case Study and Class Discussion: WhatsApp Encryption



Figures Credit

Figures and Tables on slides 7, 8, 9, 14, 16, 17, 21, 23 are taken from Computer Security: Principles and Practice 3e by Stallings & Brown.