

**Theory of Computing**  
**CSE 30151 Spring 2008**

**Cryptography: Basic Concepts**

**Department of Computer Science and Engineering**  
**University of Notre Dame**

# What is Cryptography?

- **Cryptography** is the study of mathematical techniques for achieving security objectives
- It uses **complexity theory**, as well as number theory and probability theory
- It puts to use the fact that **efficient solutions to certain problems are not known**
- **Encryption** is the best known cryptographic tool

## Historical Background

- **Ciphers** date back 2500+ years
  - the oldest cipher known to date is **Caesar cipher**
  - it is a **shift cipher** where each letter is “encrypted” by shifting right by 3 positions, i.e.,  $E(m) = (m + 3) \bmod 26$

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  
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

CAESAR → FDHVDU

- decryption consists of shifting left:  $D(c) = (c - 3) \bmod 26$
- very easy to break
- The techniques are driven by the current communication and computation technology

# Encryption

- The goal of encryption is to **communicate data in a private manner**
  - someone intercepting the ciphertext should not be able to recover information about the message or the key
- We divide ciphers into two groups: **symmetric encryption** and **public-key encryption**
- In **symmetric (or secret-key) encryption**, encryption and decryption is performed using the same key  $k$ 
  - $c = E_k(m)$  and  $m = D_k(c)$
  - both parties need to agree on the key and keep it secret

# Encryption

- A cipher that provably leaks no information about the message is called **one time pad**
  - the **key** is a randomly generated string at least as long as the message
  - **encryption** is performed by **XORing** a message with the key

key:	011001010110110001	DKFJWORIUJLAD
message:	0011001100110010	ENCRYPTION
ciphertext:	010101100101111001	HYIBVELRJXLAD

- the key must be truly random and cannot be reused
  - not very practical
- We want to be able to use a short key to encrypt lots of data

# Encryption

- It is **not possible to unconditionally hide lots of data with a short key**
- A solution is to make information recovery **computationally hard** rather than impossible in principle
  - given enough resources, an attacker can succeed
- High worst case complexity is not good enough
  - breaking must be **hard on average**, in the majority of cases
- Like with NP problems, absence of an efficient algorithm for breaking a scheme is not a proof of its security
  - confidence in an encryption algorithm is **built over time** with many people trying (and unable) to break it

# Symmetric Encryption

- Such algorithms are often **block ciphers**
  - the message is partitioned into blocks prior to encryption
  - each block is encrypted and decrypted separately
- **Most modern block ciphers**
  - use more than one type of operation to hide the data
  - proceed in iterations (called **rounds**)
  - use fast operations such as permutations and substitutions
  - are very efficient
- **Example: Digital Encryption Standard (DES)**

# DES

- **DES** was developed by IBM and **adopted as a standard in 1977**
- It was expected to be used as a standard for 10–15 years, but was replaced only in 2001 with **Advanced Encryption Standard (AES)**
- **DES characteristics:**
  - key size is 56 bits
  - block size is 64 bits
- **No practical attack on DES is known to date!**
  - the best attack on DES is brute force search of the key
  - with 56 bit key, it takes on average  $2^{55}$  tries to find the key
  - several DES challenges have been solved (<http://distributed.net/des>)

# AES

- In 2001 **Rijndael** was adopted as the **Advanced Encryption Standard**
  - invented by Belgian researchers **Daemen** and **Rijmen**
  - simple design but resistant to known attacks
  - very efficient in hardware and software implementations on a wide range of platforms
  - highly parallelizable and has very low memory requirements
  - supports different block sizes (128, 192, and 256 bits)
  - support keys of different length (128, 192, and 256 bits)

## Symmetric vs. Public-Key Encryption

- **Symmetric encryption is very fast, but both parties must have prior knowledge of the shared key**
  - not always possible
- **Public-key encryption** allows for secure communication between parties with no prior relationship or shared secret
  - the key now is a pair  $(pk, sk)$
  - the public key  $pk$  is used for encryption only:  $c = E_{pk}(m)$
  - the secret key  $sk$  is used for decryption:  $m = D_{sk}(c)$
- **Public-key cryptography is much more powerful, but less efficient**

# Secure Communication Channels

- **Communication using public-key encryption**
  - Alice creates a public-private key pair  $(pk, sk)$  and publishes  $pk$
  - everyone can securely send a message  $E_{pk}(m)$  to her
  - Alice is the only person who will be able to decrypt
  - user Bob can now securely send his credit card number to server Alice
- **Sending large volumes of data with public-key encryption is not efficient**
  - for that reason, powerful **public-key encryption** and fast **symmetric encryption** are used together
  - public key encryption is used to establish the shared secret key
  - symmetric encryption is used to communicate the data

# Complexity of Public-Key Algorithms

- Public-key cryptography uses **completely different techniques** from symmetric encryption
- It heavily relies on **number theoretic problems**
  - e.g., the fact that factoring large numbers is hard
  - arithmetic is modulo a 1024-bit or larger number
  - inefficiency comes from the need to perform modulo exponentiations with large moduli
- The **choice of the modulus** is driven by the best known algorithms for solving the hard problem
  - factoring a product of two prime numbers  $n = pq$  of length 1024 is roughly equivalent to  $2^{80}$  work

# Modern Cryptography

- Most of **modern cryptography** is proof-driven
  - a security proof gives a good evidence that the proposed solution is likely to withstand the time
- **Security proofs are often reductions**
  - assuming the difficulty of some problem, the cryptographic solution is shown to be secure
  - the proof shows that, given that someone can break our scheme, we will violate the hardness assumption

# Security Objectives

- **Confidentiality:** information is available to authorized parties only
- **Authentication:**
  - **entity authentication:** the entity is who it claims it is
  - **data authentication:** the data is coming from an authorized party
- **Integrity:** any unauthorized change to the data is detected
- **Access control:** only authorized parties can use specific resources
- **Availability:** resources are available to authorized parties
- **Non-repudiability (repudiability):** inability (ability) to deny communication

# Cryptography and Its Uses

- Cryptography is only **one tool for realizing security objectives**
  - others include software, hardware, physical security, etc.
- But it is **very powerful and has many uses**
- The need for cryptographic techniques often arises when parties **need to communicate remotely**
  - Example: fair coin flipping over the phone
- Well known uses of cryptographic techniques include **encryption, digital signatures, integrity checking, certificates**

# Cryptography and Its Uses

- **Cryptography also allows us to realize:**
  - **secure voting systems and auctions**
  - **e-cash**
  - **contract negotiation and fair contract signing**
  - **anonymous authentication (e.g., using hidden credentials and/or hidden policies)**
  - **trusted computing and data modification**
  - **usage of untrusted storage (e.g., searches on encrypted data) or untrusted computational power (e.g., uncheatable grid computing)**
  - **privacy-preserving computation (including private information retrieval and privacy-preserving data mining)**

# Attacker Models

- We often refer to parties participating in a cryptographic protocol as **Alice and Bob**
- An adversary **Eve/Carl/Mallory** eavesdrops on the communication or tries to disrupt the protocol
- Thus, security attacks can be **passive or active**
  - a **passive attacker** does not modify the data, but only monitors the communication
  - an **active attacker** can interfere with communications by modifying, deleting, inserting data and even possibly corrupting and controlling some participants

# Attacker Models

- **Outsider:** the attacker does not access to the data internal to the system
- **Insider:** the attacker is part of the system and has access to internal information
  - e.g., access to cryptographic keys
- A cryptographic system often
  - precisely **defines the goals and behavior of an attacker**
  - **formally shows** resilience to such adversarial behavior