

ENEE 459-C

Computer Security

Introduction

(continue from previous lecture)



UNIVERSITY OF
MARYLAND

Netflix incident



- See paper
 - http://www.cs.utexas.edu/~shmat/shmat_oak08netflix.pdf

Google chrome saving passwords

- In Google chrome, if you save your password so that you do not have to put it in every time you log into a website (a bad idea in general), one can retrieve the password with three clicks
- No need for the password to be displayed in plaintext
- <http://www.itpro.co.uk/security/20363/google-chrome-password-access-bug-discovered>



The human factor

- In 2010, Google fired an employee because he was caught snooping on user's data
- A 27-year-old employee, allegedly accessed information about four teenagers he met through a Seattle technology group, according to gossip website Gawker, which reported the incident Tuesday

Basic Security Principles



Economy of mechanism

- This principle stresses **simplicity** in the **design** and **implementation** of security measures
 - Example: Avoid multiple interconnecting software modules (running in different machines) to implement a security property (e.g., input of password in one machine, checking of password in another)

Complete mediation

- The idea behind this principle is that every access to a resource must be checked for **compliance with a protection scheme**
 - As a consequence, one should be wary of performance improvement techniques that save the results of previous authorization checks, since permissions can change over time
 - For example, an online banking web site should require users to sign on again after a certain amount of time, say, 15 minutes, has elapsed

Open design

- According to this principle, the security architecture and **design** of a system should be made **publicly available**
 - Security should rely only on keeping cryptographic keys secret
 - Open design allows for a system to be scrutinized by multiple parties, which leads to the early discovery and correction of security vulnerabilities caused by design errors
 - The open design principle is the opposite of the approach known as **security by obscurity**, which tries to achieve security by keeping cryptographic algorithms secret and which has been historically used without success by several organizations

Separation of privilege

- Try to isolate software components to limit the damage that can be caused in a computer system
- E.g., when one runs a virtual machine, then an attack on some software running in the virtual machine cannot affect files in the host machine (these are different machines)

Least privilege

- Each program and user of a computer system should operate with the bare **minimum privileges necessary** to function properly
 - If this principle is enforced, abuse of privileges is restricted, and the damage caused by the compromise of a particular application or user account is minimized
 - The military concept of **need-to-know** information is an example of this principle

Least common mechanism

- In systems with multiple users, mechanisms allowing resources to be **shared by more than one user should be minimized**

Psychological acceptability

- This principle states that user interfaces should be **well designed and intuitive**, and all security-related settings should adhere to what an ordinary user might expect

Work factor

- According to this principle, the **cost of circumventing** a security mechanism should be compared with the resources of an attacker when designing a security scheme
 - A system developed to protect student grades in a university database, which may be attacked by snoopers or students trying to change their grades, probably needs less sophisticated security measures than a system built to protect military secrets, which may be attacked by government intelligence organizations

Compromise recording

- This principle states that sometimes it is more desirable to **record the details** of an intrusion than to adopt more sophisticated measures to prevent it
 - Internet-connected surveillance cameras are a typical example of an effective compromise record system that can be deployed to protect a building in lieu of reinforcing doors and windows
 - The servers in an office network may maintain logs for all accesses to files, all emails sent and received, and all web browsing sessions

ENEE 459-C

Computer Security

Symmetric encryption



UNIVERSITY OF
MARYLAND

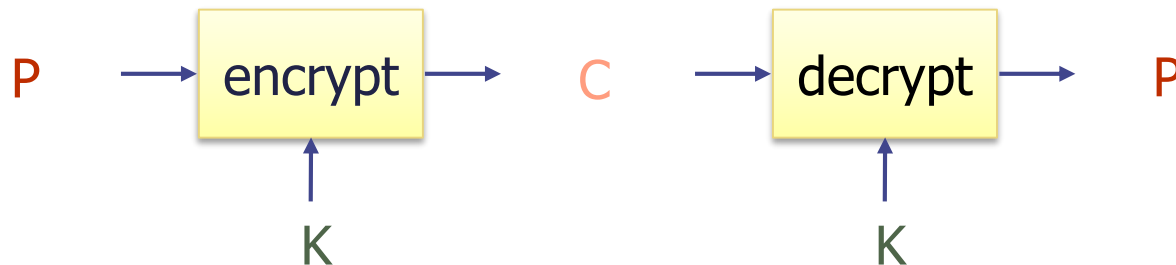
Symmetric Cryptosystem

- Scenario

- Alice wants to send a message (plaintext P) to Bob
- The communication channel is insecure and can be eavesdropped
- If Alice and Bob have previously agreed on a symmetric encryption scheme and a secret key K , the message can be sent encrypted (ciphertext C)

- Issues

- What is a good symmetric encryption scheme?
- What is the complexity of encrypting/decrypting?
- What is the size of the ciphertext, relative to the plaintext?

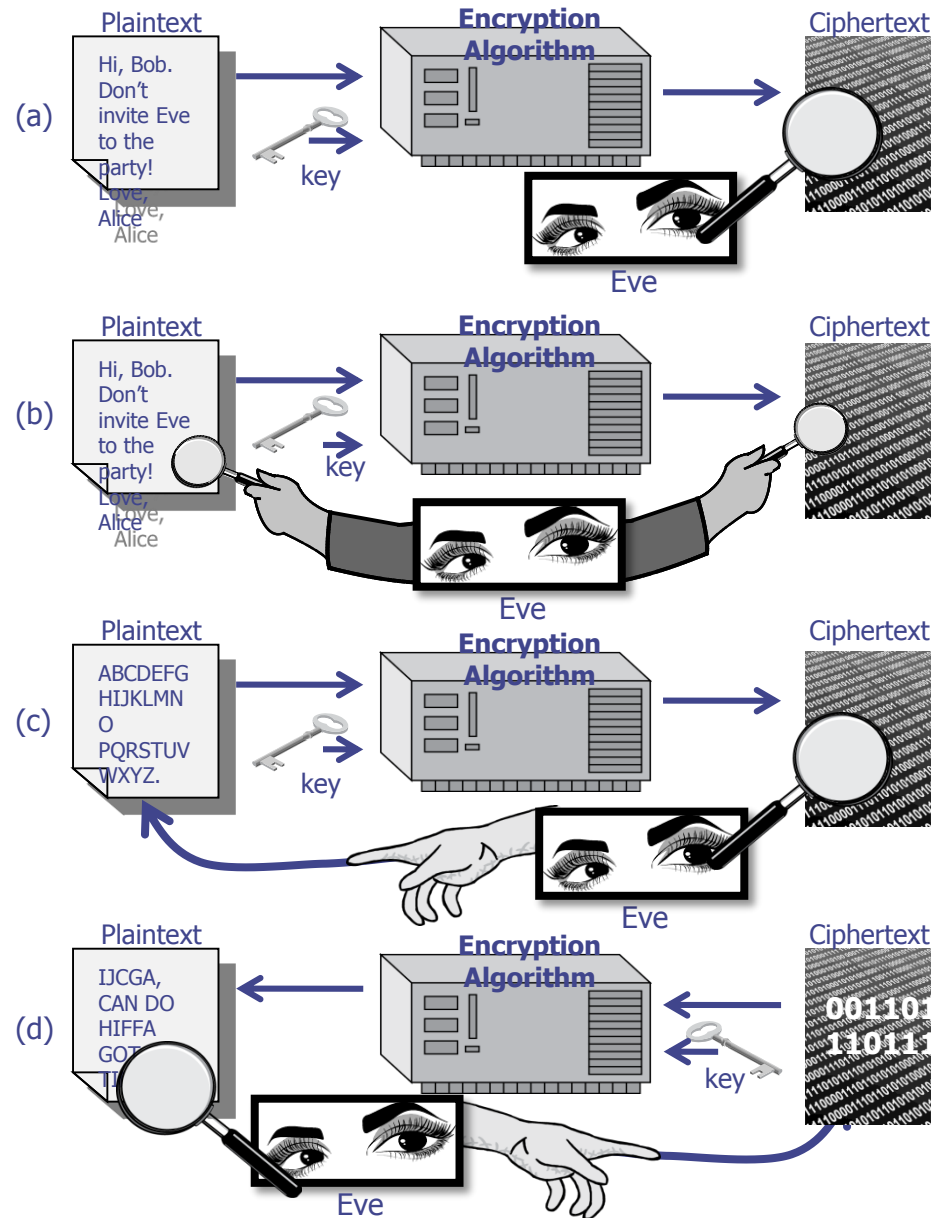


Basics

- Notation
 - Secret key K
 - Encryption function $E_K(P)$
 - Decryption function $D_K(C)$
 - Plaintext length typically the same as ciphertext length
 - Encryption and decryption are **permutation functions (bijections)** on the set of all n -bit arrays
- Efficiency
 - functions E_K and D_K should have efficient algorithms
- Consistency
 - Decrypting the ciphertext yields the plaintext
 - $D_K(E_K(P)) = P$

Attacks

- Attacker may have
 - a) collection of ciphertexts (ciphertext only attack)
 - b) collection of plaintext/ciphertext pairs (known plaintext attack)
 - c) collection of plaintext/ciphertext pairs for plaintexts selected by the attacker (chosen plaintext attack)
 - d) collection of plaintext/ciphertext pairs for plaintexts and ciphertexts selected by the attacker (chosen ciphertext attack or lunchtime attack)



Randomized encryption

- Encryption should be randomized
 - For the same plaintext, it should output different ciphertexts
- How can we turn a deterministic encryption scheme into a randomized one?
 - Padding input with randomness
- Decryption should however always work

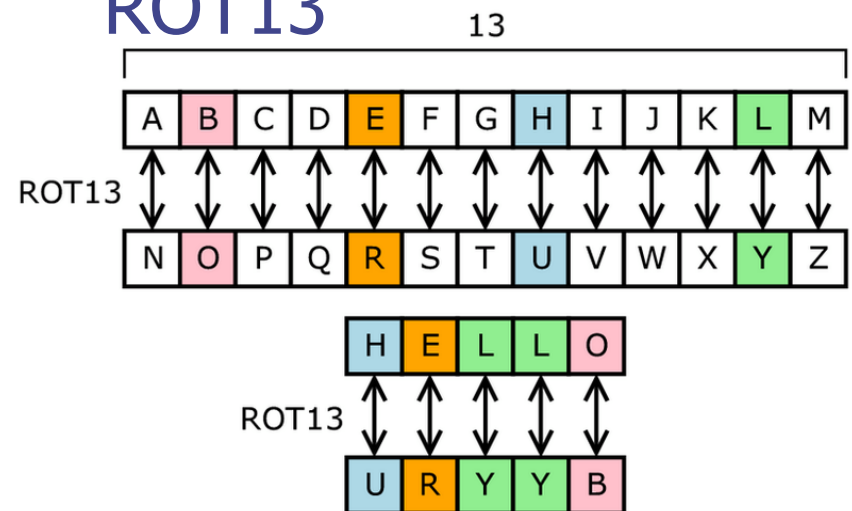
Brute-Force Attack

- Try all possible keys K and determine if $D_K(C)$ is a likely plaintext
 - Requires some knowledge of the structure of the plaintext (e.g., PDF file or email message)
- Key should be a sufficiently long random value to make exhaustive search attacks unfeasible



Substitution Ciphers

- Each letter is uniquely replaced by another
- There are $26!$ possible substitution ciphers
- One popular substitution “cipher” for some Internet posts is ROT13



Substitution Boxes

- Substitution can also be done on binary numbers.
- Such substitutions are usually described by substitution boxes, or S-boxes.

	00	01	10	11		0	1	2	3
00	0011	0100	1111	0001	0	3	8	15	1
01	1010	0110	0101	1011	1	10	6	5	11
10	1110	1101	0100	0010	2	14	13	4	2
11	0111	0000	1001	1100	3	7	0	9	12
(a)					(b)				

Figure 8.3: A 4-bit S-box (a) An S-box in binary. (b) The same S-box in decimal.

Frequency Analysis

- Letters in a natural language, like English, are not uniformly distributed
- Knowledge of letter frequencies, including pairs and triples can be used in cryptologic attacks against substitution ciphers

a: 8.05%	b: 1.67%	c: 2.23%	d: 5.10%
e: 12.22%	f: 2.14%	g: 2.30%	h: 6.62%
i: 6.28%	j: 0.19%	k: 0.95%	l: 4.08%
m: 2.33%	n: 6.95%	o: 7.63%	p: 1.66%
q: 0.06%	r: 5.29%	s: 6.02%	t: 9.67%
u: 2.92%	v: 0.82%	w: 2.60%	x: 0.11%
y: 2.04%	z: 0.06%		

Letter frequencies in the book *The Adventures of Tom Sawyer*, by Twain.

Semantic security

- I give you a symmetric encryption scheme $(\text{Enc}, \text{Dec}, K)$
- What do you need to prove in order to say that it is secure?
- A strong notion used is “semantic security”
- Informally:
 - An attacker picks message m_0 and m_1 and sends them to the bank that has the secret key
 - The bank flips a coin b and computes $t_b = \text{Enc}_K(m_b)$
 - The bank sends t_b to the attacker
 - The scheme is secure if the attacker has no better chance of finding whether t_b corresponds to m_0 or m_1 than just guessing!
- This should hold even if it is repeated many (polynomial) times

One-Time Pads

- There is one type of substitution cipher that is absolutely unbreakable
 - The **one-time pad** was invented in 1917 by Joseph Mauborgne and Gilbert Vernam
 - We use a block of shift keys, (k_1, k_2, \dots, k_n) , to encrypt a plaintext, M , of length n , with each shift key being chosen uniformly at random
- Since each shift is random, every ciphertext is equally likely for any plaintext

Algorithms

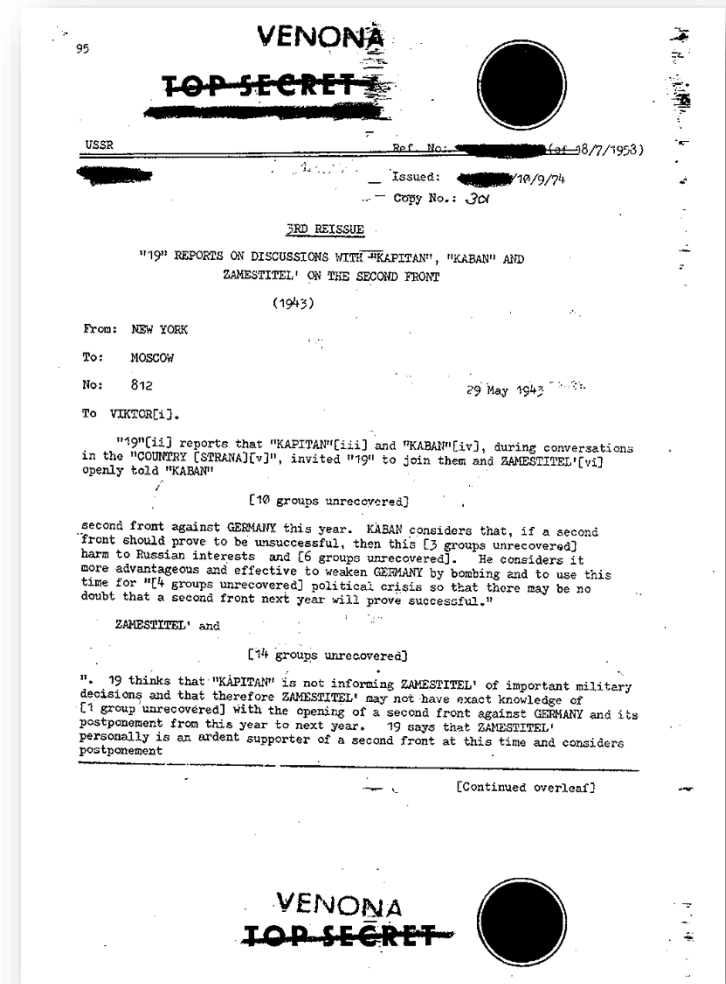
- $K \leftarrow \text{KeyGen}(n)$: Pick a random key K of n bits
- $E_K(A)$: On input plaintext A , compute ciphertext $B = A \text{ XOR } K$
- $D_K(B)$: On input ciphertext B , compute plaintext $A = B \text{ XOR } K$
- **Correctness**: $B \text{ XOR } K = (A \text{ XOR } K) \text{ XOR } K = A \text{ XOR } 0 = A$
- **Security?**

Perfect security

- For all messages m_1 and m_2 and for all ciphertexts c
- $\Pr[K \leftarrow \text{KeyGen}(n): E_K(m_1)=c] = \Pr[K \leftarrow \text{KeyGen}(n): E_K(m_2)=c]$
- Proof
 - Note that $\text{Enc}_K(m_1)=c$ is the event $m_1 \text{ XOR } K = c$ which is the event $K = m_1 \text{ XOR } c$
 - K is chosen at random (irrespective of m_1 and m_2 , and therefore the probability is 2^{-n})
 - Namely ciphertext does not reveal anything about the plaintext

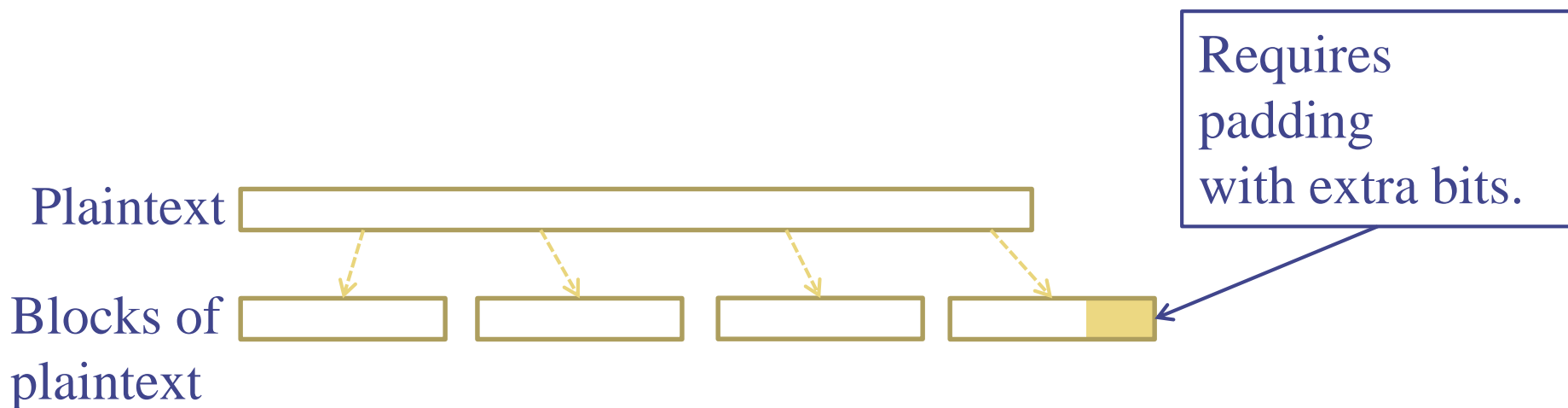
Weaknesses of the One-Time Pad

- In spite of their perfect security, one-time pads have some weaknesses
- The key has to be as long as the plaintext
- Keys can never be reused
 - Repeated use of one-time pads allowed the U.S. to break some of the communications of Soviet spies during the Cold War



Block Ciphers

- In a **block cipher**:
 - Plaintext and ciphertext have fixed length b (e.g., 128 bits)
 - A plaintext of length n is partitioned into a sequence of m **blocks**, $P[0], \dots, P[m-1]$, where $n \leq bm < n + b$
- Each message is divided into a sequence of blocks and encrypted or decrypted in terms of its blocks



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 K_A , K_B , and K_C
 - Effective key length is 168 bits, making exhaustive search attacks unfeasible
 - $C = E_{K_C}(D_{K_B}(E_{K_A}(P)))$; $P = D_{K_A}(E_{K_B}(D_{K_C}(C)))$
 - Equivalent to DES when $K_A=K_B=K_C$ (backward compatible)
- 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**

A perfect encryption of a block

- Say you have a block of n bits
- You want to encrypt it
- You want to use the same key all the time
- Consider a bijective mapping T from $\{0,1\}^n$ to $\{0,1\}^n$
- The pairs are computed uniformly at random
- To encrypt x , just output $T[x]$
- To decrypt y , just output $T^{-1}[y]$
- Your secret key is T
- Problem with this approach: T has size $\sim n 2^n$
- Can you make it randomized?

