# Lecture 2

## Cryptography:

## History + Simple Encryption Methods and Preliminaries



## Cryptography can be used at different levels

- algorithms: encryption, signatures, hashing, RNG
- protocols (2 or more parties): key distribution, authentication, identification, login, payment, etc.
- systems: electronic cash, secure filesystems, smartcards, VPNs, e-voting, etc.
- attacks: on all the above

# Some applications of cryptography

- **network, operating system security**
- **protect Internet, phone, space communication**
- **electronic payments (e-commerce)**
- **database security**
- **software/content piracy protection**
- **pay TV (e.g., satellite)**
- **military communications**
- **voting**

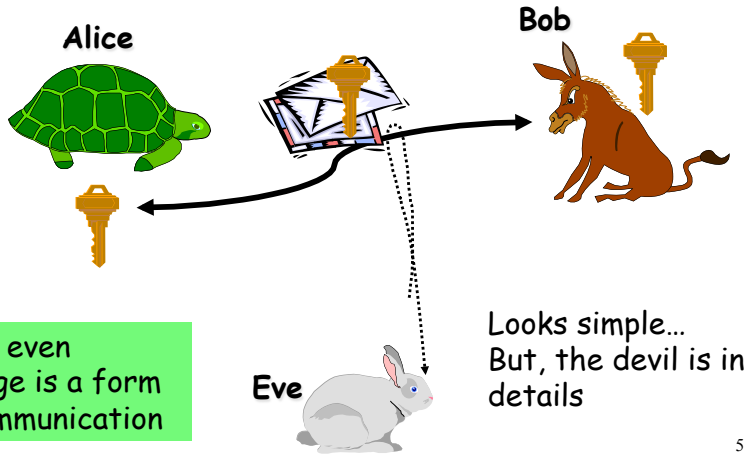


# Open vs. closed design model

• **Open design**: algorithm, protocol, system design (and even possible plaintext) are public information. Only key(s) are kepp secret.

• **Closed design**: as much information as possible is kept secret.

# Core issue in network security : how to communicate securely?




Note: even storage is a form of communication

Looks simple…
But, the devil is in the details



# The biggest "headache" is that…

## Good security must be

Effective

Yet

Unobtrusive

Because security is not a service in and of itself, but a burden!

# Cryptography has been around…

- Most CS sub-fields are fairly new:
  - Graphics, compilers, software, CSCW, OS, architecture
- And, a few are quite old:
  - Cryptography, database, networking



# Some history: Caesar's cipher

**Homo**
**Hominem**
**Lupus!**

**Krpr**
**Krplqhp**
**Oxsxv!**

# Some history: Rosetta Stone



# Some history: Enigma

# Historical (Primitive) Ciphers

- **Shift (e.g., Caesar):** $Enc_k(x) = x+k \bmod 26$
- **Affine:** $Enc_{k1,k2}(x) = k1 * x + k2 \bmod 26$
- **Substitution:** $Enc_{perm}(x) = perm(x)$
- **Vigenere':** $Enc_K(x) = ( X[0]+K[0], X[1]+K[1], \ldots )$
- **Vernam: one-time pad (OTP)**



# Shift (Caesar) Cipher

Example:

$K = 11$

| W | E | W | I | L | L | M | E | E | T | A | T | M | I | D | N | I | G | H | T |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 22 | 4 | 22 | 8 | 11 | 11 | 12 | 4 | 4 | 19 | 0 | 19 | 12 | 8 | 3 | 13 | 8 | 6 | 7 | 19 |

| 7 | 15 | 7 | 19 | 22 | 22 | 23 | 15 | 15 | 4 | 11 | 4 | 23 | 19 | 14 | 24 | 19 | 17 | 18 | 4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| H | P | H | T | W | W | X | P | P | E | L | E | X | T | O | Y | T | R | S | E |

- How many keys are there?
- How many trials are needed to find the key?

# Substitution Cipher

Example:

*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*

*X N Y A H P O G Z Q W B T S F L R C V M U E K J D I*

KEY

*W E W I L L M E E T A T M I D N I G H T*

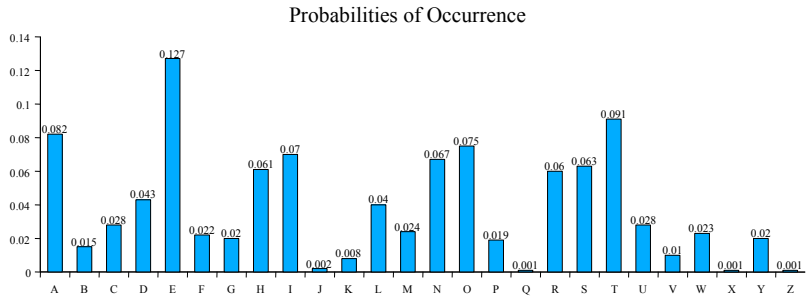
*K H K Z B B T H H M X M T Z A S Z O G M*

- How many keys are there?
- How many trials are needed to find the key?



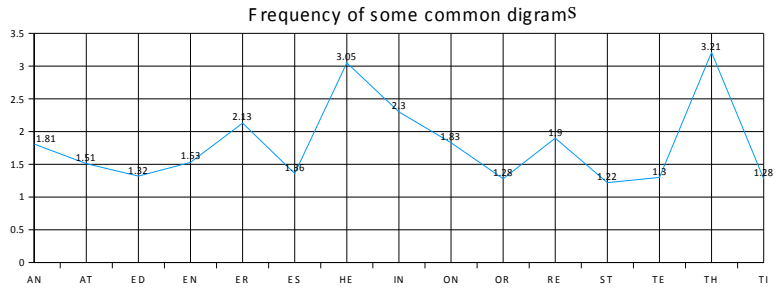
# Substitution Cipher

Cryptanalysis



# Substitution Cipher

Cryptanalysis






# VERNAM One-Time Pad: world's best cipher

$$\text{Plaintext} = \{p_0, \ldots, p_{n-1}\}$$

$$\text{One - time pad stream} = \{otp_0, \ldots, otp_{n-1}\}$$

$$\text{Ciphertext} = \{c_0, \ldots, c_{n-1}\}$$

$$where:$$

$$c_i = p_i \oplus otp_i \forall 0 < i < n$$

$$C = A \oplus B$$

$$C \oplus B = A$$

# VERNAM One-Time Pad: world's best cipher

❖ Vernam offers perfect information-theoretic security, but:

❖ How long does the OTP keystream needs to be?

❖ How do Alice and Bob exchange the keystream?



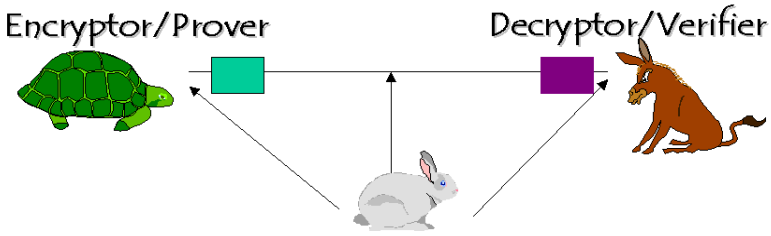
## Encryption Principles

- A cryptosystem has (at least) five ingredients:
  - Plaintext
  - Secret Key
  - Ciphertext
  - Encryption algorithm
  - Decryption algorithm
- Security usually depends on the secrecy of the key, not the secrecy of the algorithm

# Crypto Basics

Crypto Attacks:
- ciphertext only
- known plaintext
- chosen plaintext
- chosen ciphertext

Cryptosystem:
- *P -- plaintext*
- *C -- ciphertext*
- *K -- keyspace*
- *E -- encryption rules*
- *D -- decryption rules*






# Average time required for exhaustive key search (for brute force attacks)

| Key Size (bits) | Number of Alternative Keys | Time required at $10^6$ Decr/$\mu$s |
|---|---|---|
| 32 | $2^{32} = 4.3 \times 10^9$ | 2.15 milliseconds |
| 56 | $2^{56} = 7.2 \times 10^{16}$ | 10 hours |
| 128 | $2^{128} = 3.4 \times 10^{38}$ | $5.4 \times 10^{18}$ years |
| 168 | $2^{168} = 3.7 \times 10^{50}$ | $5.9 \times 10^{30}$ years |

# Types of Attainable Security

• Perfect, unconditional or "information theoretic": the security is evident free of any assumptions

• Reducible or "provable": security can be shown to be based on some common (often unproven) assumptions, e.g., the conjectured difficulty of factoring large integers

• Ad hoc: the security seems good → often "snake oil"…

Take a look at:

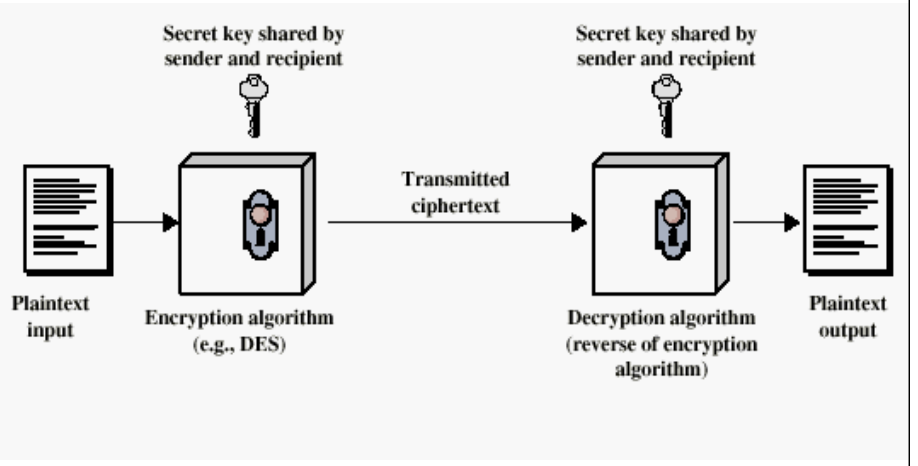
http://www.ciphersbyritter.com/GLOSSARY.HTM



# Computational Security

- Encryption scheme is *computationally secure* if
  - cost of breaking it (via brute force) exceeds the value of the encrypted information; or
  - time required to break it exceeds useful lifetime of the encrypted information
- Most modern schemes we will see are considered computationally secure
  - Usually rely on very large key-space, impregnable to brute force
- Most advanced schemes rely on lack of knowledge of effective algorithms for certain hard problems, not on a proven inexistence of such algorithms → reducible security!
  - Such as: factorization, discrete logarithms, etc.

# Conventional Encryption Principles






# Cryptosystems

Classified along three dimensions:

- Type of operations used for transforming plaintext into ciphertext
  - Binary arithmetic: shifts, XORs, ANDs, etc.
    - Typical for **conventional** encryption
  - Integer arithmetic
    - Typical for **public key** encryption
- Number of keys used
  - Symmetric or conventional (single key used)
  - Asymmetric or public-key (2 keys: 1 to encrypt, 1 to decrypt)
- How plaintext is processed:
  - One bit at a time
  - A string of any length
  - A block of bits

# Complexity reminder/re-cap

- P: problems that can be solved in polynomial time, i.e., problems that can be solved/decided "efficiently"
- NP: broad set of problems that includes P;
  - answers can be verified "efficiently";
  - solutions can't always be efficiently found.
- NP-complete: the believed-to-be-hard decision problems in NP, they appear to have no efficient solution; answers are efficiently verifiable, solution to one is never much harder than a solution to another
- NP-hard: hardest; cannot be solved by a non-deterministic TM. Many computation version of NP-complete problems are NP-hard.
- Examples:
  - Factoring, discrete log are in NP, not know if in NP-complete or in P
  - Primality testing was recently shown to be in P
  - Knapsack is in NP-complete

For more info, see: **http://www.nist.gov/dads/HTML**



# Suggested readings:

Chapters 1 and 2 in KPS book

Optional: Ch 1 in Stinson

Don't forget to check the website! Did you do it before this lecture?