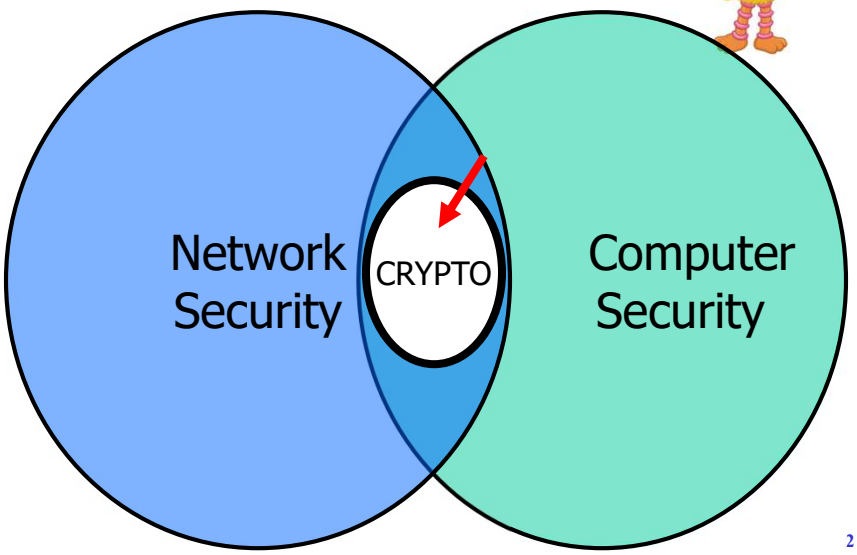
Cryptography:
History, Simple Encryption Methods
and Preliminaries

Bird's eye view





Cryptography

The word cryptography comes from the Greek words κρυπτός (hidden or secret) and γράφειν (writing).

Thus, historically cryptography has been:

The **art** of secret writing

Most of today's cryptography is well-grounded in mathematics and it's unclear whether there's still an "art" aspect to it.

3

Cryptography can be used at several different levels

- Algorithms: encryption, digital signatures, hashing, Random Number Generators (RNGs), secure erasure

- Protocols (2 or more parties): key distribution, authentication, identification, log-in, e-payment, etc.

- Systems: electronic cash, secure file-systems, smartcards, VPNs, e-voting, crypto-currencies, etc.

- Attacks: on all the above

4

Some Applications of Cryptography

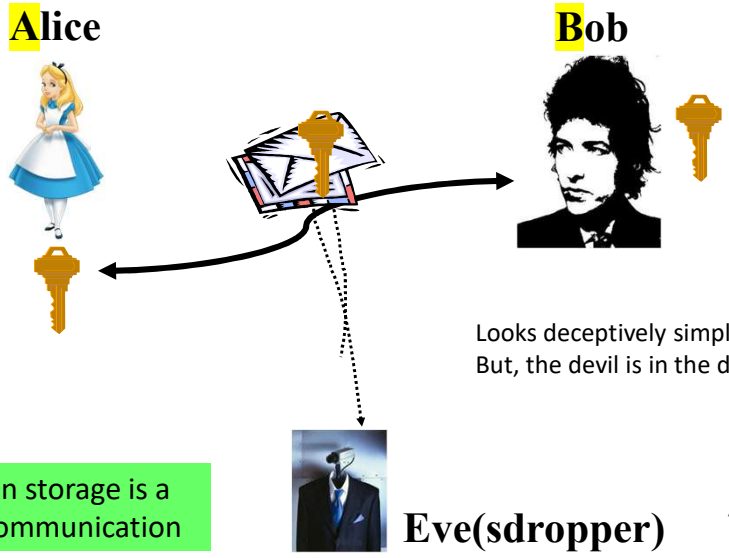
- Network, operating system security
- Protect Internet, phone, space communication
- Electronic payments (e-commerce)
- Database security
- Software/content piracy protection
- Pay TV (e.g., satellite)
- Military communications
- Voting

Open vs. Closed Design Model

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

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

Core Issue in Network security : How to Communicate Securely?



Looks deceptively simple …
But, the devil is in the details

Note: even storage is a form of communication

7

7

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=overhead!

8

8

Cryptography is Old …

- Most sub-fields in CS are fairly new (20-30 years):
  - Graphics, compilers, software, OS, architecture
- And, some are quite old:
  - Predate computing and electronic comm.
  - Cryptography, databases, networking, compilers, languages, numerical methods

9

Some History: Caesar's Cipher

Homo
Hominem
Lupus!



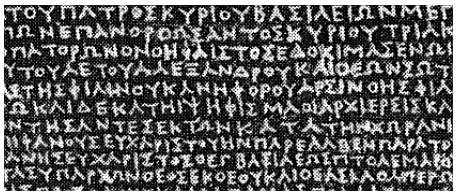
Krpr
Krplqhp
Oxsxv!

10

Some History: Rosetta Stone







11

11

Some History: Enigma





Alan Turing
(1912-1954)

12

12

Encryption Principles

- A cryptosystem has (at least) five ingredients:
  - Plaintext
  - Secret Key k
  - Ciphertext
  - Encryption Algorithm: Plaintext $\xrightarrow{k}$ Ciphertext
    - Can be denoted as $Enc_k()$
  - Decryption Algorithm: Ciphertext $\xrightarrow{k}$ Plaintext
    - Can be denoted as $Dec_k()$
- Security usually depends on the **secrecy of the key** k, not the secrecy of the algorithms

13

Historical (Primitive) Ciphers

Letter

- **Shift (e.g., Caesar):** $Enc_k(x) = x+k \bmod 26$

14

Shift (Caesar) Cipher

$Enc_k(x) = x+k \bmod 26$
$Dec_k(x) = x-k \bmod 26$

Example:

$k = 11$ ← Secret key $k \in K$, K is key space

Plaintext

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

Ciphertext

- How many possible keys are there ($|K|$)? $O(|K|/2)$
- How many average trials are needed to find the key?

15

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:** $Enc_{perm}(x) = perm(x)$

16

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 k

Plaintext

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

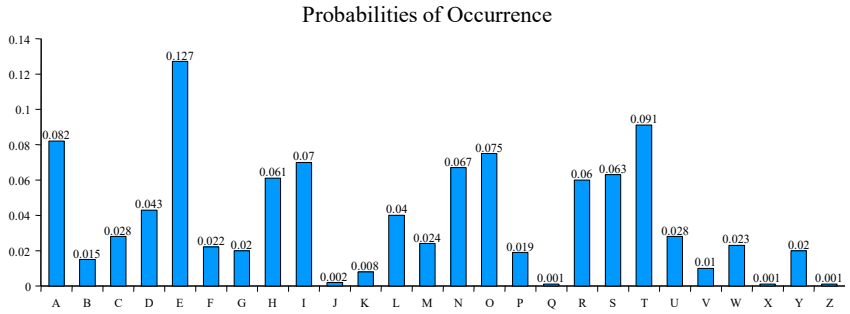
Ciphertext

- How many possible keys are there ($|K|$)?
- How many average trials are needed to find the key?

17

Substitution Cipher

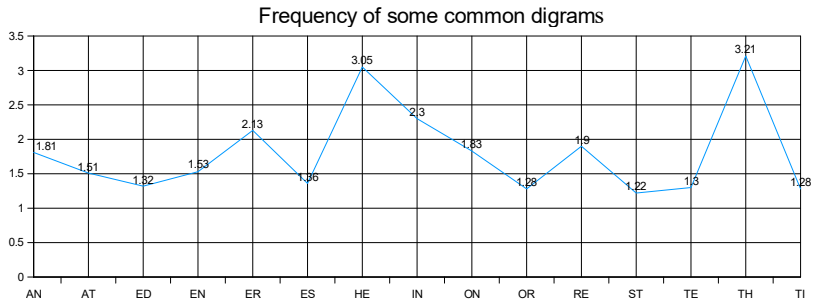
Cryptanalysis



18

Substitution Cipher

Cryptanalysis



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], …, X[n]+K[N] $)$ — "W" "E" … "T"
- **Vernam:** One-Time Pad (OTP)

VERNAM One-Time Pad (OTP): World's Best Cipher

$$\text{Plaintext} = \{p_0, \ldots, p_{n-1}\}$$
$$\text{One - time pad stream} = \{otp_0, \ldots, otp_{n-1}\}$$
$$\text{Ciphertext} = \{c_0, \ldots, c_{n-1}\}$$
$$where:$$
$$c_i = p_i \oplus otp_i \forall 0 < i < n$$

bit-level XOR

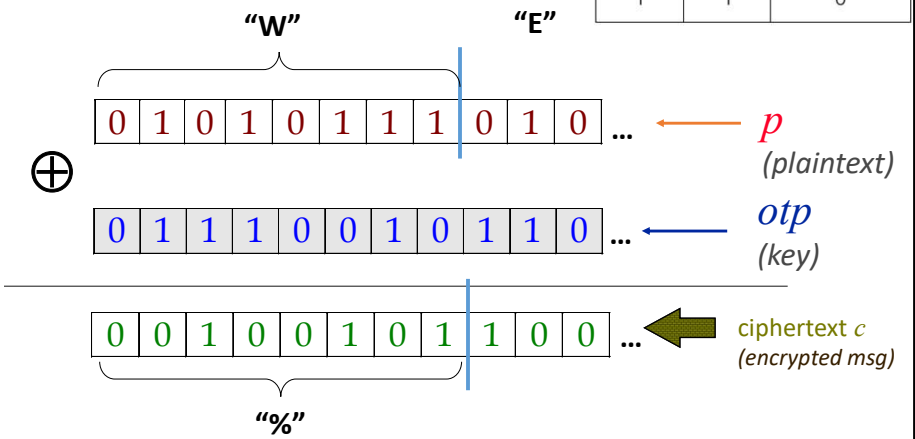
21

21

One-Time Pad (OTP)

A	B	A **XOR** B
0	0	0
0	1	1
1	0	1
1	1	0

For each **bit**:



70

22

VERNAM One-Time Pad (OTP): Cipher

A	B	A **XOR** B
0	0	0
0	1	1
1	0	1
1	1	0

$$C = A \oplus B$$
$$C \oplus B = A$$

Enc() ⟶ $c_i = p_i \oplus otp_i \forall 0 < i < n$

Dec()? $p_i = c_i \oplus otp_i$

23

VERNAM One-Time Pad (OTP): World's Best Cipher

- Vernam offers perfect <u>information-theoretic</u> security,
  - For any p, p', c: $Pr[E(k, p) = c] = Pr[E(k', p') = c]$
  - Intuitively: ciphertext reveal no "info" about plaintext

but:

- How long does the OTP keystream need to be?
- How do Alice and Bob exchange the keystream?

24

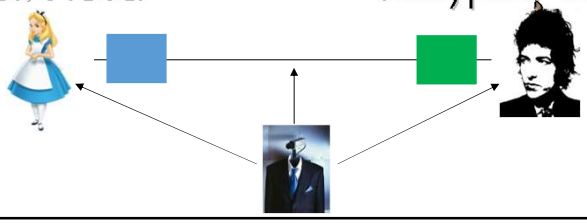
Crypto Basics

Cryptosystem:

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

Encryptor/Prover

Decryptor/Verifier



25

Crypto Basics

Crypto Attacks:

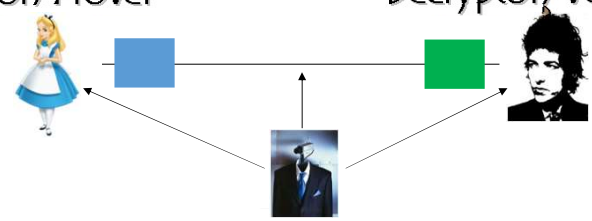
- ciphertext only
- known plaintext
- chosen plaintext
- chosen ciphertext
- brute force

Cryptosystem:

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

Encryptor/Prover

Decryptor/Verifier

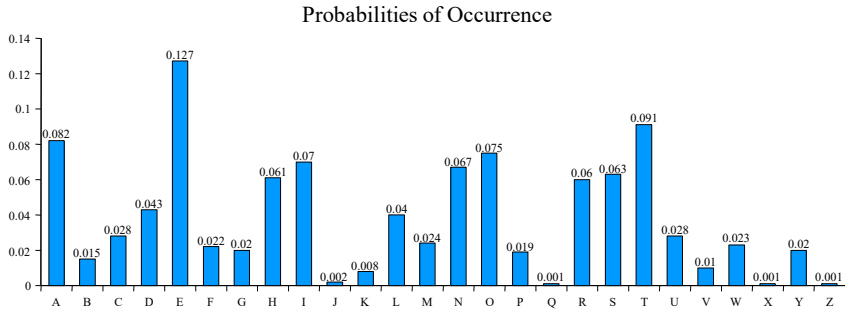


26

Substitution Cipher

Cryptanalysis

Ciphertext-only attack



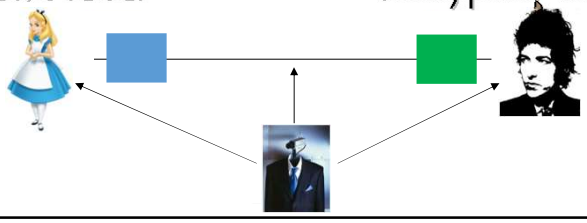
Crypto Basics

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

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

Encryptor/Prover

Decryptor/Verifier



Average Time for Exhaustive Key Search (for Brute-Force Attacks)

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

Today, **> 80 bits** can be generally considered "secure"

(but things are changing rapidly due to GPUs and special hardware; better to treat this only as the least length under not-so-rich/resourceful attackers)

29

29

Types of Attainable Security

• Perfect, unconditional or "information theoretic": the security is evident free of any (computational/hardness) assumptions

• Reducible: 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 → "snake oil"…

Take a look at:

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

30

30

Computational Security

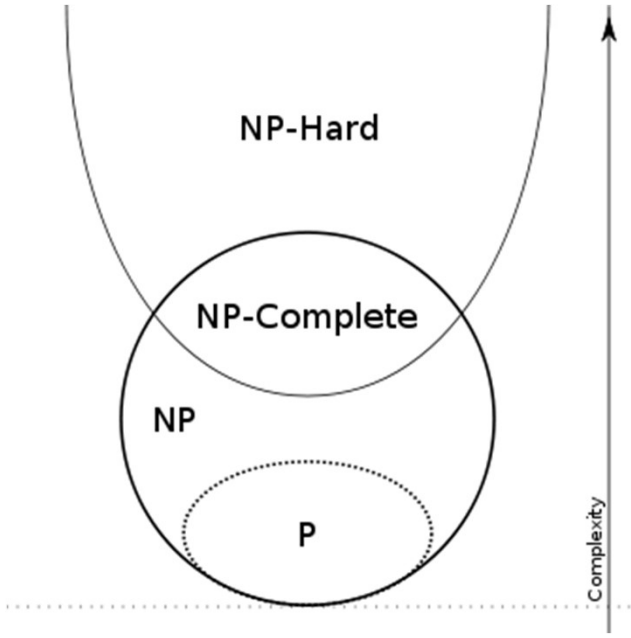
- Encryption scheme is *computationally secure* if
  - cost of breaking it (via brute force) exceeds the value of the encrypted information; or
  - time required to break it exceeds useful lifetime of the encrypted information
- Most modern schemes we will see are considered computationally secure
  - Usually rely on having a very large key-space, impregnable to brute force attacks
- 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.

31

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" (in polynomial time);
  - solutions cannot always be efficiently found (as far as we know).
- **NP-complete**: 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; some of them may not be solved by a non-deterministic TM. Many computational version of NP-complete problems are NP-hard.

32



33

33

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" (in polynomial time);
  - solutions cannot always be efficiently found (as far as we know).
- **NP-complete**: 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; some of them may not be so
  TM. Many computational version of NP-complet
- **Examples:**
  - Factoring, discrete log are in NP, not known if NP-complete or P
  - Primality testing was "recently" (2002) shown to be in P
  - Knapsack is NP-complete

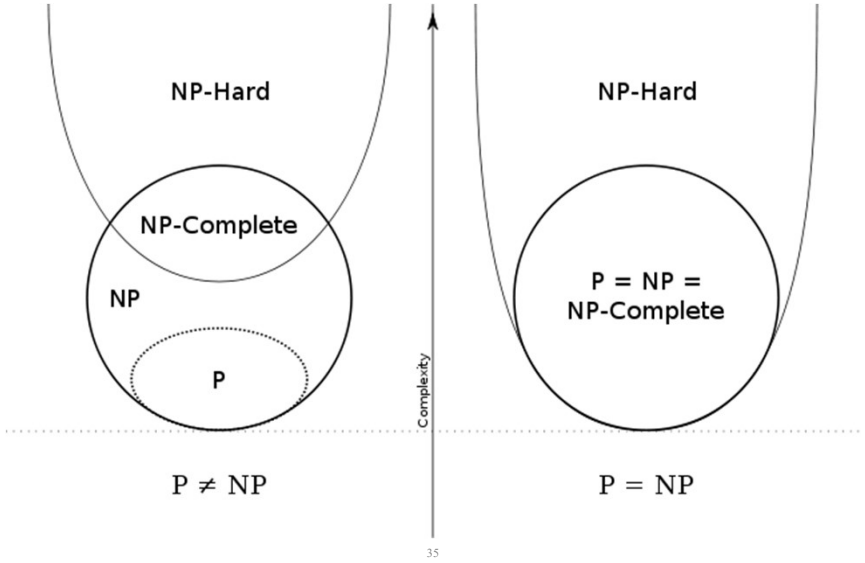
Relied upon by today's encryption schemes!

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

34

34

P vs NP



Suggested Readings:

Chapters 1 and 2 in KPS book

Optional: Ch 1 in Stinson