Introduction to applied cryptography

– Lecture 1

Omar Hasan

Books

- The content in the lectures is drawn from the following books:
  - **Applied Cryptography**, Second Edition – Bruce Schneier
  - **An Introduction to Cryptography**, version 8.0 – PGP Corporation

Security

“If I take a **letter**, lock it in a safe [or some **box**], hide the safe somewhere in New York [or any large **city**], then tell you to read the letter, that’s not security. That’s obscurity.

On the other hand, if I take a **letter** and lock it in a **safe**, and then give you the safe along with the **design specifications** of the safe and a **hundred identical safes** with their **combinations** so that you and the world’s best safecrackers can **study the locking mechanism**—and you still can’t open the safe and read the letter—that’s security.”

– Bruce Schneier

Some objectives of security

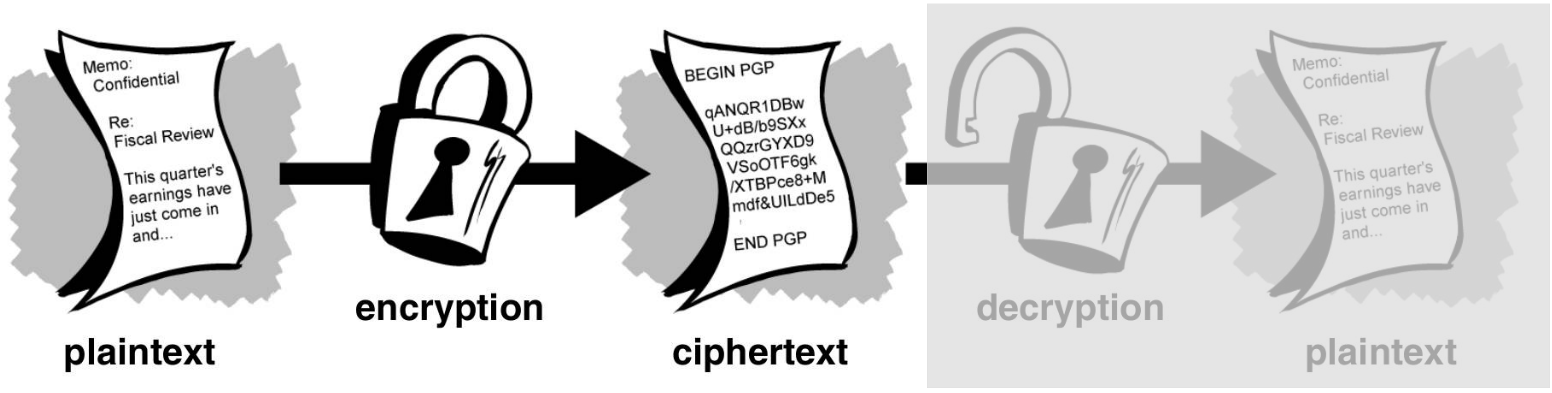
- **Confidentiality.** This is a necessary element for privacy. Confidentiality is an attribute of our capacity to protect our information from un-authorised access by intruders.
- **Authentication.** It should be possible for the receiver of a message to ascertain its origin; an intruder should not be able to masquerade as someone else.
- **Integrity.** It should be possible for the receiver of a message to verify that it has not been modified in transit; an intruder should not be able to substitute a false message for a legitimate one.
- **Non-repudiation.** A sender should not be able to falsely deny later that he sent a message.

Cryptography

- Cryptography is a **building block** for achieving **security objectives** such as confidentiality, authentication, integrity, non-repudiation, as well as others.
- **Cryptography** is the science of using mathematics to **encrypt** and **decrypt** data.
- Cryptography enables you to store **sensitive information** or transmit it across insecure networks so that it **cannot be read by anyone** except the intended recipient.

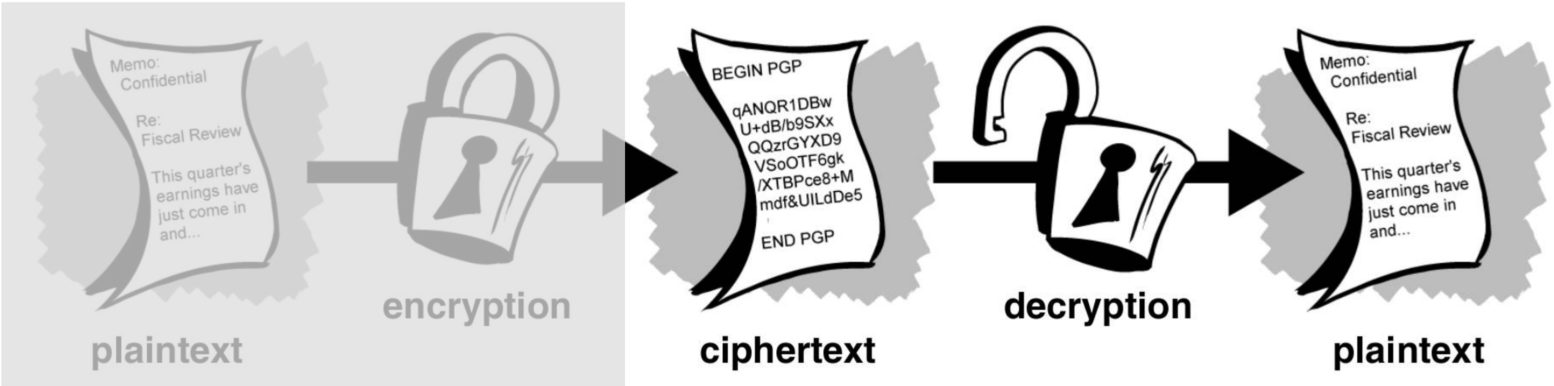
Encryption

- Data that can be read and understood without any special measures is called **plaintext**.
- The method of disguising plaintext in such a way as to hide its substance is called **encryption**.
- Encrypting plaintext results in unreadable gibberish called **ciphertext**.



Decryption

- The process of reverting ciphertext to its original plaintext is called **decryption**.



Keys

- In modern cryptography, both the encryption and decryption operations use **keys**.
- A key might be any one of a large number of values. The range of possible values of the key is called the **keyspace**.
- All of the **security** in the encryption and decryption algorithms is **based on the keys**.
- None of the security is based in the **details of the algorithm**. The algorithm can be published and analyzed.
- It doesn’t matter if an **eavesdropper** knows your **algorithm**; if she doesn’t know your particular **key**, she **can’t read** your messages.

Some notation

- Plaintext is denoted by **M**, for **message**, or **P**, for **plaintext**. It can be a stream of bits, a text file, a bitmap, etc.
- **Ciphertext** is denoted by **C**.
- An **encryption** function **E**, operates on M to produce C.

  E(M) = C
- A **decryption** function **D** operates on C to produce M.

  D(C) = M
- Decrypting a message recovers the **original plaintext**.

  D(E(M)) = M

Some notation

- A **key** is denoted by **K**.
- Encryption and decryption operations that use this key are denoted as follows:

  $E_K(M) = C$

  $D_K(C) = M$

- Algorithms that use a **different encryption key** and **decryption key** are denoted below. The encryption key, K1, is different from the corresponding decryption key, K2.

  $E_{K1}(M) = C$

  $D_{K2}(C) = M$

Cryptosystem

- A **cryptographic algorithm**, or **cipher**, is a mathematical function used in the encryption and decryption process.
- A cryptographic algorithm works in combination with a **key** to encrypt the plaintext.
- The same plaintext encrypts to **different ciphertext** with **different keys**.

Cryptosystem

- The **security** of encrypted data is dependent on two things: the strength of the **cryptographic algorithm** and the secrecy of the **key**.
- A cryptographic algorithm, plus all possible keys and all the protocols that make it work, comprise a **cryptosystem**. Example: PGP, RSA.

Types of cryptosystems

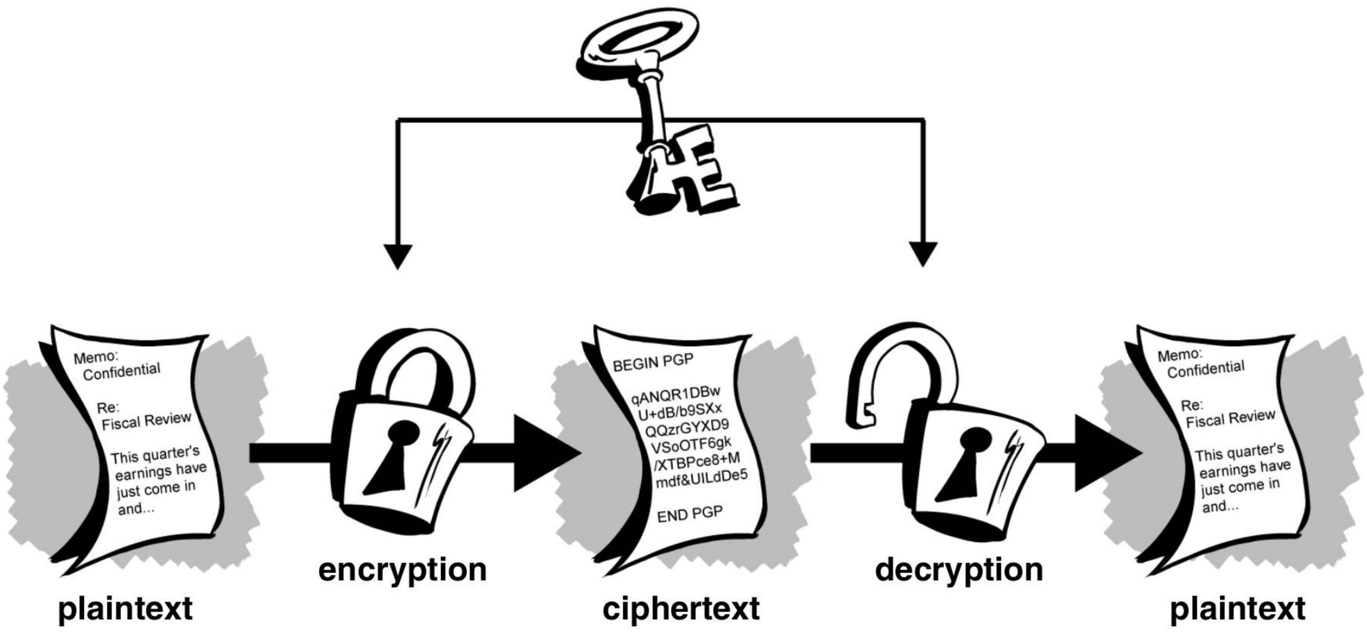
- There are three major types of cryptosystems:
  1) Cryptosystems based on **symmetric-key cryptography**
  2) Cryptosystems based on **public-key cryptography**
  3) **Hybrid** cryptosystems

Types of cryptosystems

- There are three major types of cryptosystems:
  1) Cryptosystems based on **symmetric-key cryptography**
  2) Cryptosystems based on **public-key cryptography**
  3) **Hybrid** cryptosystems

Symmetric-key cryptography

- In conventional cryptography, also called **symmetric-key** or **secret-key** encryption, **one key** is used both for encryption and decryption.
- An example of a conventional cryptosystem is **Data Encryption Standard (DES)**.



Symmetric-key cryptography

Advantage:

- Symmetric-key encryption and decryption is **fast**.

Disadvantage:

- For a sender and recipient to communicate securely using symmetric-key encryption, they must **agree upon a key** and keep it secret between themselves.
- If they are in different physical locations, they must trust some **secure communications medium** to prevent the disclosure of the secret key during transmission.
- Thus, secure **key distribution** is a challenge.

Substitution cipher

- A simple type of symmetric or secret-key cryptography is a **substitution cipher**.
- A substitution cipher substitutes one piece of information for another. This is most frequently done by **offsetting letters of the alphabet**.
- An example is **Caesar’s cipher**. The algorithm is to offset the alphabet and the key is the number of characters to offset it.
- For example, if we encode the word **“SECRET”** using Caesar’s **key value of 3**, we offset the alphabet so that the **3rd letter down (D)** begins the alphabet.

Substitution cipher

- Starting with:

  ABCDEFGHIJKLMNOPQRSTUVWXYZ

  and sliding everything **up by 3**, you get

  DEFGHIJKLMNOPQRSTUVWXYZABC

  where **D=A, E=B, F=C**, and so on.
- Using this scheme, the plaintext, **“SECRET”** encrypts as **“VHFUHW”**.
- To allow someone else **to read the ciphertext**, you tell them that the **key is 3**.

Substitution cipher

Some exercises

1) MRWEPCSR, key = 4
2) KNXWXLLIHKML, key = 19
3) ERFVQRAPRP, key = 13
4) GVMJOJIYZ, key = 21
5) DPSKLJDVWRQEHUJHU, key = 3

Substitution cipher

Solutions

1) MRWEPCSR, key = 4, INSALYON
2) KNXWXLLIHKML, key = 19, RUEDESSPORTS
3) ERFVQRAPRP, key = 13, RESIDENCEC
4) GVMJOJIYZ, key = 21, LAROTONDE
5) DPSKLJDVWRQEHUJHU, key = 3, AMPHIGASTONBERGER

https://www.dcode.fr/caesar-cipher

Types of cryptosystems

- There are three major types of cryptosystems:
  1) Cryptosystems based on **symmetric-key cryptography**
  2) Cryptosystems based on **public-key cryptography**
  3) **Hybrid** cryptosystems

Public-key cryptography

- Public-key cryptography is an **asymmetric scheme** that uses a **pair of keys** for encryption: **a public key**, which encrypts data, and a corresponding **private key** (secret key) for decryption.

Advantage:

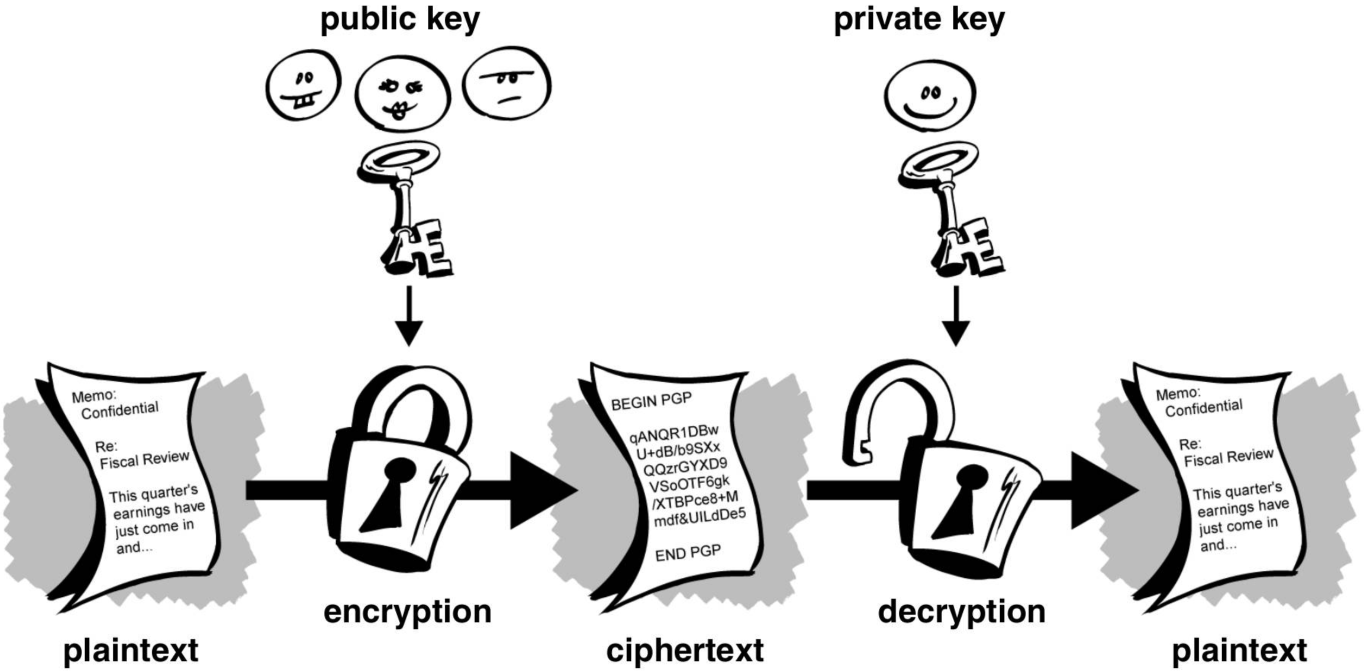
- The problem of **key distribution** is solved by public-key cryptography.

Disadvantage:

- Public-key encryption is about **1,000 times slower** than secret-key encryption.

Public-key cryptography

- You **publish your public key** to the world while keeping your **private key secret**.
- Anyone with a copy of your **public key** can then **encrypt information** that only you can read.



Public-key cryptography

- It is **computationally infeasible** to deduce the private key from the public key.
- Anyone who has a **public key** can **encrypt** information but **cannot decrypt** it.
- Only the person who has the corresponding **private key** can **decrypt** the information.

Public-key cryptography

- The advantage of public key cryptography is that it allows people who have **no preexisting security arrangement** to exchange messages securely.
- The need for sender and receiver to share secret keys via some **secure channel is eliminated**; all communications involve **only public keys**, and no private key is ever transmitted or shared.
- Two examples of public-key cryptosystems are **RSA** and **Diffie-Hellman**.

Diffie-Hellman

- Diffie-Hellman is a **public-key cryptosystem**, created in 1976, and named after its inventors Whitfield Diffie and Martin Hellman.
- Diffie–Hellman is a type of **key exchange algorithm** that enables two parties to agree upon a **shared secret key** by communicating over a channel that may be insecure.
- This is possible even if the two parties have **no prior knowledge** of each other.
- It gets its security from the difficulty of calculating **discrete logarithms** in a finite field.

Diffie-Hellman

1) Two users **Alice** and **Bob**, publicly share a modulus ***p*** and a base ***g***.

2) Alice generates a secret random integer ***A***.
Bob generates a secret random integer ***B***.

3) Alice then calculates $a = g^A \ (mod\ p)$.
Bob calculates $b = g^B \ (mod\ p)$.

4) Alice sends ***a*** (Alice's public key) to Bob.
Bob sends ***b*** (Bob's public key) to Alice.

5) Alice computes: $K = b^A \ (mod\ p)$

6) Bob computes: $K = a^B \ (mod\ p)$

Where:

$$K = b^A \ (mod\ p) = a^B \ (mod\ p) = g^{AB} \ (mod\ p)$$

Diffie-Hellman

Diffie-Hellman Key Exchange

Alice	Bob
Parameters: p, g	
$A = \text{random}()$ $a = g^A \pmod p$	$\text{random}() = B$ $g^B \pmod p = b$
$a \longrightarrow$ $\longleftarrow b$	
$K = g^{BA} \pmod p = b^A \pmod p$	$a^B \pmod p = g^{AB} \pmod p = K$
$\longleftarrow E_K(data) \longrightarrow$	

Source: https://www.rankred.com/common-encryption-techniques/

Diffie-Hellman

Example

1) Two users **Alice** and **Bob**, publicly share a modulus ***p = 43*** and a base ***g = 7***.

2) Alice generates a secret random integer ***A = 8***.
Bob generates a secret random integer ***B = 11***.

3) Alice then calculates $a = g^A \pmod{p} = \mathbf{7^8 \pmod{43} = 6}$.
Bob calculates $b = g^B \pmod{p} = \mathbf{7^{11} \pmod{43} = 37}$.

4) Alice sends ***a*** (Alice's public key) to Bob.
Bob sends ***b*** (Bob's public key) to Alice.

5) Alice computes: $K = b^A \pmod{p} = \mathbf{37^8 \pmod{43} = 36}$

6) Bob computes: $K = a^B \pmod{p} = \mathbf{6^{11} \pmod{43} = 36}$

Secret key = K = ***36***

Diffie-Hellman

- Diffie-Hellman is a public domain algorithm, which is well suited for use in **data communication** over insecure networks.
- However, Diffie-Hellman has **some shortcomings** when compared to other public-key cryptosystems such as **RSA**.
- For example, it **does not support authentication** of the participants (through digital signatures) and is thus susceptible to man-in-the-middle attacks.

Diffie-Hellman

Exercise

1) Two users **Alice** and **Bob**, publicly share a modulus ***p* = *43*** and a base ***g* = *7***.

2) Alice generates a secret random integer ***A* = *?***.
Bob generates a secret random integer ***B* = *?***.

3) Alice then calculates $a = g^A \ (mod\ p)$ = ***?***
Bob calculates $b = g^B \ (mod\ p)$ = ***?***

4) Alice sends ***a*** (Alice's public key) to Bob.
Bob sends ***b*** (Bob's public key) to Alice.

5) Alice computes: $K = b^A \ (mod\ p)$ = ***?***

6) Bob computes: $K = a^B \ (mod\ p)$ = ***?***

   – Secret key = *K* = ***?***,   Attacker's knowledge?

Types of cryptosystems

- There are three major types of cryptosystems:
  1) Cryptosystems based on **symmetric-key cryptography**
  2) Cryptosystems based on **public-key cryptography**
  3) **Hybrid** cryptosystems

Hybrid cryptosystem

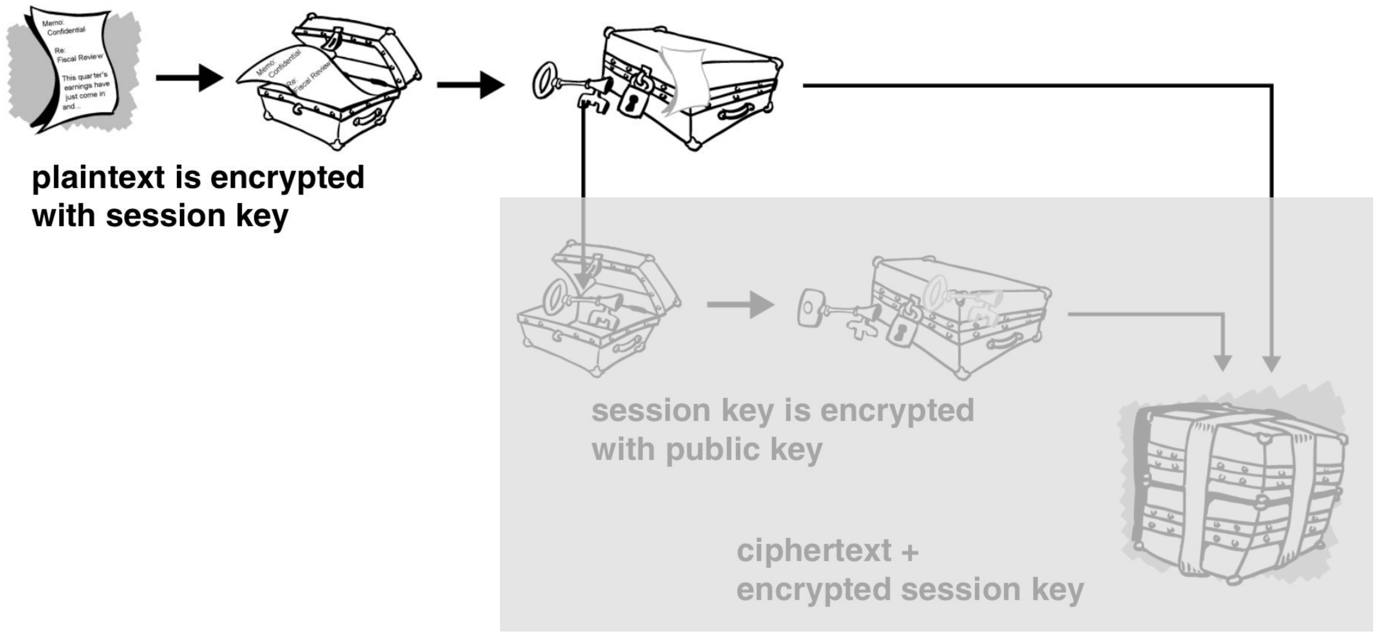
- A **hybrid cryptosystem** combines the features of both secret and public-key cryptography. An example: **PGP** (Pretty Good Privacy).
- **Secret-key** encryption is about 1,000 times **faster** than public-key encryption.
- Whereas, **public-key** encryption provides a solution to **key distribution** and data transmission issues.
- Used together, **performance** and **key distribution** are improved without any sacrifice in **security**.

PGP

Encryption:

1) PGP creates a **session key**, which is a one-time-only **secret key**. This key is a random number.

2) The session key works with a secure and fast symmetric **encryption algorithm** to encrypt the plaintext; the result is the **ciphertext**.

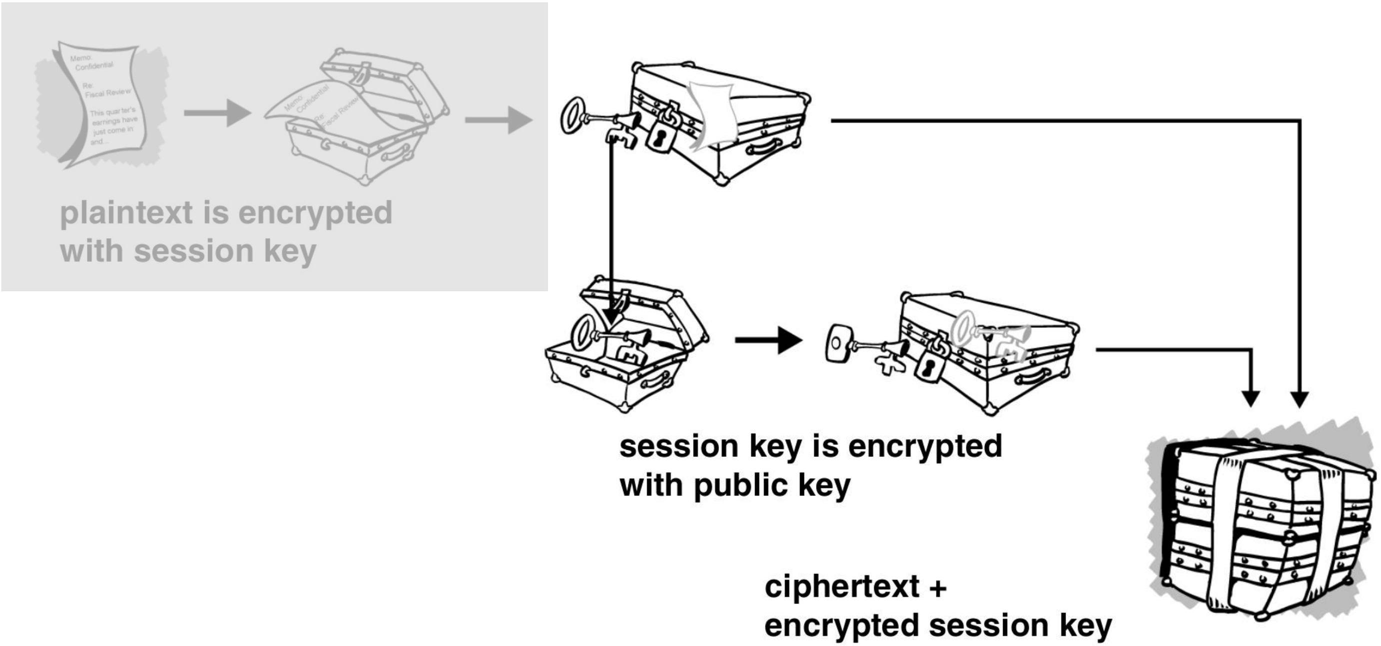


PGP

Encryption:

3) The **session key** is then **encrypted** to the recipient's **public key**.

4) This public key-encrypted **session key** is **transmitted** along with the **ciphertext** to the recipient.

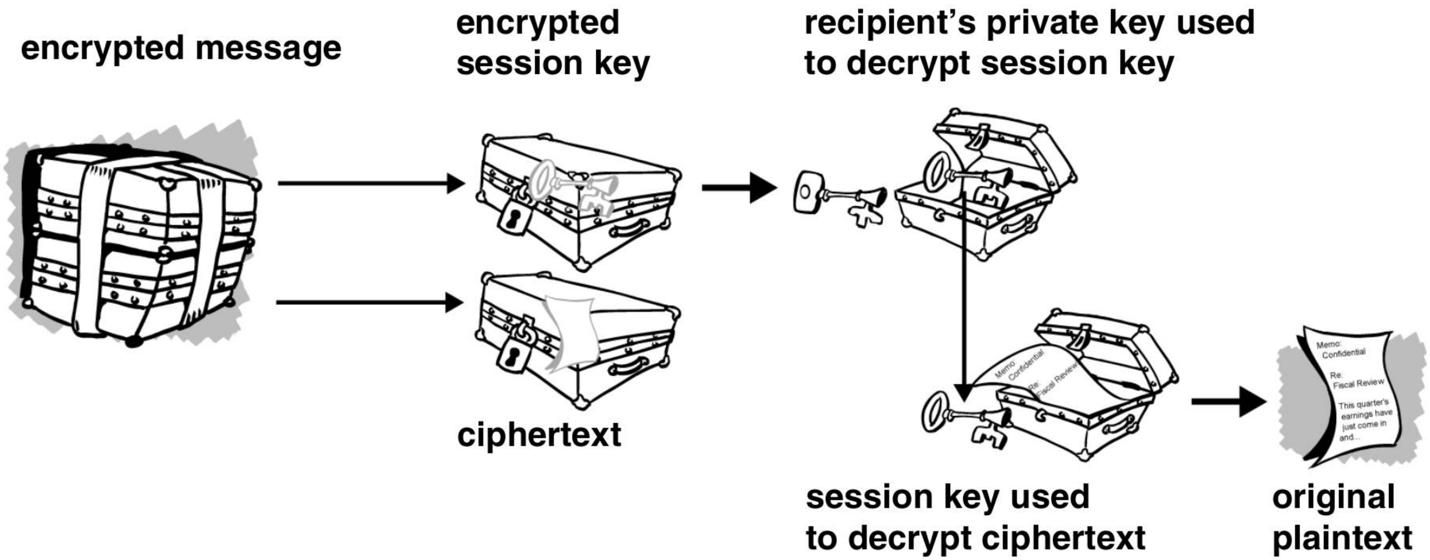


PGP

Decryption:

1) Decryption works in the **reverse**. The recipient's PGP uses his or her **private key** to recover the **session key**.

2) PGP then uses the session key to **decrypt** the conventionally encrypted **ciphertext**.



Hybrid cryptosystem

Review of the advantages of a hybrid cryptosystem

- A **hybrid cryptosystem** combines the features of both secret and public-key cryptography.
- **Secret-key** encryption is **faster** than public-key encryption.
- Whereas, **public-key** encryption provides a solution to **key distribution** and data transmission issues.
- Used together, **performance** and **key distribution** are improved without any sacrifice in **security**.

More about cryptographic keys

- A **key** is a value that works with an encryption algorithm to produce a ciphertext.
- Keys are essentially **large integers**.
- Key size is measured in **bits** e.g. 2048-bits. The **bigger** the key, the **more secure** the ciphertext.
- **Public key** size and **conventional** cryptography's secret key size are **unrelated**.
- A conventional **80-bit** key has the equivalent strength of a **1024-bit** public key.

Cryptanalysis

- The purpose of cryptography is to keep the **plaintext** and the **keys** secret from an adversary (also called eavesdropper or attacker).
- Attackers are assumed to have **complete access to the communications** between the sender and receiver.
- **Cryptanalysis** is the science of recovering the **plaintext** of a message **without access to the key**.
- A fundamental assumption in cryptanalysis is that the cryptanalyst has complete details of the **cryptographic algorithm and implementation**.
- The **secrecy** must reside entirely in the **key**.

Cryptanalytic attacks

- There are four general types of cryptanalytic attacks:
  1) **Ciphertext-only** attack
  2) **Known-plaintext** attack
  3) **Chosen-plaintext** attack
  4) **Adaptive-chosen-plaintext** attack

Ciphertext-only attack

- The cryptanalyst has the **ciphertext of several messages**, all of which have been encrypted using the same encryption algorithm.
- **Recover the plaintext** of as many messages as possible, or better yet **deduce the key** (or keys) used to encrypt the messages.
- **Given**: $C_1 = E_k(P_1)$, $C_2 = E_k(P_2)$,... $C_i = E_k(P_i)$
- **Deduce**: Either P_1, P_2, ... P_i; k; or an algorithm to infer P_{i+1} from $C_{i+1} = E_k(P_{i+1})$

Known-plaintext attack

- The cryptanalyst has access not only to the **ciphertext of several messages**, but also to the **plaintext of those messages**.
- **Deduce the key** used to encrypt the messages or an **algorithm to decrypt** any new messages.
- **Given**: $P_1, C_1 = E_k(P_1), P_2, C_2 = E_k(P_2), \ldots P_i, C_i = E_k(P_i)$
- **Deduce**: Either k, or an algorithm to infer P_{i+1} from $C_{i+1} = E_k(P_{i+1})$

Chosen-plaintext attack

- The cryptanalyst not only has access to the **ciphertext** and associated **plaintext** for **several messages**, but he also **chooses the plaintext** that gets encrypted.
- **Deduce the key** used to encrypt the messages or an **algorithm to decrypt** any new messages encrypted.
- **Given**: P_1, $C_1 = E_k(P_1)$, P_2, $C_2 = E_k(P_2)$,... P_i, $C_i = E_k(P_i)$, where the cryptanalyst gets to choose P_1, P_2 ,... P_i.
- **Deduce**: Either *k*, or an algorithm to infer P_{i+1} from $C_{i+1} = E_k(P_{i+1})$

Adaptive-chosen-plaintext attack

- This is a special case of a **chosen-plaintext attack**.
- Not only can the cryptanalyst choose the plaintext that is encrypted, but he can also **modify his choice** based on the results of previous encryption.
- In a chosen-plaintext attack, a cryptanalyst might just be able to choose **one large block of plaintext** to be encrypted.
- Whereas, in an adaptive-chosen-plaintext attack he can choose a **smaller block of plaintext** and then **choose another** based on the results of the first, and so forth.

Security of algorithms

- An algorithm is considered **computationally secure** if it cannot be broken with resources available to the attacker under consideration.
- You can measure the **complexity** of an attack in different ways:
  - **Data complexity.** The amount of data needed as input to the attack.
  - **Processing complexity.** The time needed to perform the attack.
  - **Storage requirements.** The amount of memory needed to do the attack.

Cryptanalysis of the Caesar cipher

- **Ciphertext-only attack**.
- **'Brute force'** solution: Decrypt the ciphertext using **each key** and determine the **fitness** of each decryption.
- **25** possible keys.
- Determining fitness:
  - Obtain the **statistics of standard English** text (e.g., **Quadgram** Statistics).
  - Calculate the **statistics of the deciphered text**.
  - **Compare** these statistics to those from standard English text.

Source for the content on cryptanalysis of the Caesar cipher:
http://practicalcryptography.com/cryptanalysis/stochastic-searching/cryptanalysis-caesar-cipher/

Cryptanalysis of the Caesar cipher

- This method works by first determining the **statistics of english text**, then calculating the **likelyhood** that the ciphertext comes from the **same distribution**.
- An **incorrectly deciphered** (i.e. using the wrong key) message will probably contain **sequences e.g. 'QKPC'** which are very **rare in normal English**.
- In this way we can **rank different decryption keys**, the decryption key we want is the one that produces deciphered text with the **fewest rare sequences**.

Cryptanalysis of the Caesar cipher

An example

- **Ciphertext:**

```
YMJHFJXFWHNUMJWNXTSJTKYMJJFWQNJXYPSTBSFSIXNRUQJXYHNUMJWX
```

- **Cryptanalysis:**

```
key          plaintext                      fitness
-----------------------------------------------------
1 : XLIGEIWEVGMTLIVMWSRISJXLIIEVPM... , -442.22
2 : WKHFDHVDUFLSKHULVRQHRIWKHHDUOL... , -495.20
3 : VJGECGUCTEKRJGTKUQPGQHVJGGCTNK... , -484.13
4 : UIFDBFTBSDJQIFSJTPOFPGUIFFBSMJ... , -490.73
5 : THECAESARCIPHERISONEOFTHEEARLI... , -246.02
6 : SGDBZDRZQBHOGDQHRNMDNESGDDZQKH... , -485.69
7 : RFCAYCQYPAGNFCPGQMLCMDRFCCYPJG... , -481.17
8 : QEBZXBPXOZFMEBOFPLKBLCQEBBXOIF... , -478.19
9 : PDAYWAOWNYELDANEOKJAKBPDAAWNHE... , -415.66
10: OCZXVZNVMXDKCZMDNJIZJAOCZZVMGD... , -488.75
11: NBYWUYMULWCJBYLCMIHYIZNBYYULFC... , -490.46
12: MAXVTXLTKVBIAXKBLHGXHYMAXXTKEB... , -490.82
13: LZWUSWKSJUAHZWJAKGFWGXLZWWSJDA... , -483.63
14: KYVTRVJRITZGYVIZJFEVFWKYVVRICZ... , -475.01
15: JXUSQUIQHSYFXUHYIEDUEVJXUUQHBY... , -466.90
16: IWTRPTHPGRXEWTGXHDCTDUIWTTPGAX... , -458.49
17: HVSQOSGOFQWDVSFWGCBSCTHVSSOFZW... , -474.67
18: GURPNRFNEPVCUREVFBARBSGURRNEYV... , -460.86
19: FTQOMQEMDOUBTQDUEAZQARFTQQMDXU... , -467.13
20: ESPNLPDLCNTASPCTDZYPZQESPPLCWT... , -454.29
21: DROMKOCKBMSZROBSCYXOYPDROOKBVS... , -461.91
22: CQNLJNBJALRYQNARBXWNXOCQNNJAUR... , -479.58
23: BPMKIMAIZKQXPMZQAWVMWNBPMMIZTQ... , -473.52
24: AOLJHLZHYJPWOLYPZVULVMAOLLHYSP... , -474.57
25: ZNKIGKYGXIOVNKXOYUTKULZNKKGXRO... , -494.13
```

Cryptanalysis of the Caesar cipher

Python code

```
import re
from ngram_score import ngram_score
fitness = ngram_score('quadgrams.txt') # load our quadgram statistics
from pycipher import Caesar

def break_caesar(ctext):
    # make sure ciphertext has all spacing/punc removed and is uppercase
    ctext = re.sub('[^A-Z]','',ctext.upper())
    # try all possible keys, return the one with the highest fitness
    scores = []
    for i in range(26):
        scores.append((fitness.score(Caesar(i).decipher(ctext)),i))
    return max(scores)

# example ciphertext
ctext = 'YMJHFJXFWHNUMJWNXTSJTKYMJJFWQNJXYPSTBSFSIXNRUQJXYHNUMJWX'
max_key = break_caesar(ctext)

print 'best candidate with key (a,b) = '+str(max_key[1])+':'
print Caesar(max_key[1]).decipher(ctext)
```

Quadgram statistics

- Quadgrams: **groups of 4 letters** in a text (with spacing and punctuation removed)
  - E.g. the quadgrams in the text **ATTACK** are: **ATTA**, **TTAC**, and **TACK**.
- Quadgrams from a large English text:

Quadgram	Count	Quadgram	Count
		...	
TION	13168375	AAOZ	2
NTHE	11234972	AAJQ	2
THER	10218035	ZZZV	1
THAT	8980536	ZZZJ	1
OFTH	8132597	ZZXW	1
...		...	

Quadgram statistics as a fitness measure

- To compute the probability of a piece of text being English, **extract all the quadgrams**, then **multiply each of the quadgram probabilities**.
- For the text **ATTACK**, the quadgrams are **ATTA**, **TTAC**, and **TACK**. The total probability is:

  *p(ATTACK) = p(ATTA) * p(TTAC) * p(TACK)*

  Where:

  p(ATTA) = count(ATTA) / N,

  N = total number of quadgrams.

- A **higher number** means it is **more likely to be English**, while a lower number means it is less likely to be English.

Quadgram statistics as a fitness measure

Example

- Let’s assume:
  - **Total** number of quadgrams: **1000**
  - Quadgram **frequencies: ATTA: 7, TTAC: 3, TACK: 11**

- Computing the **fitness measure** for the text **ATTACK**:

*p(ATTACK) = p(ATTA) * p(TTAC) * p(TACK)*

*= (count(ATTA)/N) * (count(TTAC)/N) * (count(TACK)/N)*

*= (7 / 1000) * (3 / 1000) * (11 / 1000)*

= 0,000000231

Quadgram statistics as a fitness measure

- **Log probability:**

  log(p(ATTACK)) = log(p(ATTA)) + log(p(TTAC)) + log(p(TACK))

- Due to, *log(a*b) = log(a)+log(b)*
- Log probability is used as the fitness measure since the **probabilities may be very small**.
- From a sample English text with 2500000 total quadgrams:

Quadgram	Count	Log Probability
AAAA	1	-6.40018764963
QKPC	0	-9.40018764963
YOUR	1132	-3.34634122278
TION	4694	-2.72864456437
ATTA	359	-3.84509320105

Cryptanalysis of the Caesar cipher

Exercise

- Ciphertext: **VHFUHW**
- Keys to try: **3, 5**
- For each key, calculate the fitness of the deciphered text
- Quadgram statistics:
  - **Total**: 4224127912
- Some quadgram frequencies:
  - **SECR**: 203226, **ECRE**: 480393
  - **CRET**: 226466, **QCAP**: 103
  - **CAPC**: 3895, **APCR**: 1290

Cryptanalysis of the Caesar cipher

Solution

- Ciphertext: **VHFUHW**
- Plaintext: key as **3: SECRET**; key as **5: QCAPCR**
- *log(p(**SECRET**))*

  = *log(p(**SECR**)) + log(p(**ECRE**)) + log(p(**CRET**))*

  = *log(203226 / 4224127912) + log(480393 / 4224127912) + log(226466 / 4224127912)*

  = ***−12.5326***
- *log(p(**QCAPCR**))*

  = *log(103 / 4224127912) + log(3895 / 4224127912) + log(1290 / 4224127912)*

  = ***−20.1633***