



**Higher Diploma in Science – Software Development**  
**Higher Diploma in Science – Data Analytics**

*Computational Thinking with Algorithms*



1

**Cryptography and Reliable  
Interaction**

-

**An Algorithmic View**

**Sean Duignan, M.Sc, Ph.D**

(sean.duignan@gmit.ie)



2

## Overview of this session....

- Cryptography (terms and definitions)
- Types of cryptosystems (symmetric and asymmetric)
- Examples of systems in practice

*– Through a "thinking" and "algorithmic" lens.....*

## Introduction to Cryptography

- **Cryptography is the science of secrecy** and is concerned with the need to communicate in secure, private and reliable ways.
- From a computational thinking / algorithmics perspective, a novel feature is the fact (as we will see) that **modern methods** used to solve cryptographic problems exploit the difficulty of solving other problems.

This is somewhat surprising.....

***problems for which no good algorithms are known are crucial here.***

***(More on this anon)***

## Cryptography (Problem Statements)

- The basic problem to be solved is that of **encrypting** and **decrypting** data.

– How should we encode an important message in such a way that the receiver should be able to decipher it, but not an eavesdropper?

– Moreover, can then message be **signed** by the sender so that:

- (1) The receiver can be sure that only the sender could have sent it.
- (2) The sender cannot later deny having sent it
- (3) The receiver, having received the signed message, cannot sign a message in the senders name, not even additional versions of the very message that has just been received.



## Cryptography (Some Definitions)

- Data that can be read and understood without any special measures is called **plaintext** (or clear-text). Plaintext, **P**, is the input to an encryption process (algorithm).
- **Encryption** is the **process** of disguising plaintext in such a way as to hide its substance.
- The result (output....) of the encryption process is **ciphertext, C**

A general encryption procedure (left) and decryption procedure (right) is as follows:

$$C = \text{Encr}(P)$$

and

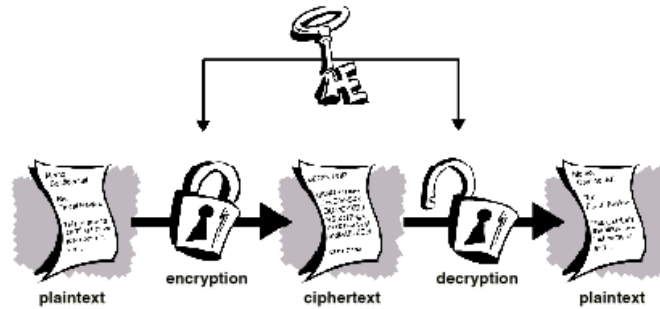
$$P = \text{Decr}(C)$$



## Symmetric Cryptography

- In conventional cryptography, also called **secret-key** or **symmetric-key** encryption, **one key** is used both for encryption and decryption.

**EG :** The Data Encryption Standard (DES) cryptosystem.



## Symmetric Cryptography – Simple Example

### Substitution Cipher

- The *Caesar cipher* shifted the alphabet by 3 characters.

A	B	C	D.....	W	X	Y	Z
↓	↓	↓	↓	↓	↓	↓	↓
D	E	F	G.....	Z	A	B	C

**EG:**           Hello    →    Khoor

- Caesar cipher is a one to one mapping. **A monoalphabetic substitution!**
- Cryptanalysis exploits statistical properties of the English language.
- "E" is the most frequently occurring letter.



## Symmetric Cryptography – Simple Example

L	21*	12.28%
V	17*	9.94%
Y	16*	9.36%
O	15*	8.77%
U	15*	8.77%
P	14*	8.19%
K	12*	7.02%
Z	10*	5.85%
H	10*	5.85%
A	9*	5.26%
T	7*	4.09%
M	7*	4.09%
S	4*	2.34%
N	4*	2.34%
J	3*	1.75%
B	3*	1.75%
D	2*	1.17%
C	1*	0.58%
R	1*	0.58%

Pypzotlu huk Pypzodvtlu: Pu  
aol uhtl vm Nvk huk vm aol  
klhk nlulyhapvuz myvt dopjo  
zol yljlpclz oly vsk ayhkpapvu  
vm uhapvuovvk, Pylshuk,  
aoyvbno bz, zbtvuz oly  
jopskylu av oly mshn huk  
zayprlz mvy oly myllkvt.



- 7

Irishmen and Irishwomen: In  
the name of God and of the  
dead generations from which  
she receives her old tradition of  
nationhood, Ireland, through  
us, summons her children to  
her flag and strikes for her  
freedom.



Q. Is L or I really E or e.....?



## Symmetric Cryptography – Examples cntd...

### *Polyalphabetic Substitution:*

- The key is a **simple phrase** of fixed length (which can repeat).
- Add the value of the letter in the key to the value of the letter in the plaintext to get ciphertext, and "wrap around" if necessary (modulo 36)

Plaintext	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
Numeric Value	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	26
Plaintext	0	1	2	3	4	5	6	7	8	9																
Numeric Value	27	28	29	30	31	32	33	34	35	36																

- This is essentially multiple Caesar-type ciphers
- Main advantage is that **same plaintext** gets mapped onto **different ciphertext**.



# Symmetric Cryptography – Examples cntd...

## Polyalphabetic Substitution cntd....

Plaintext	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
Numeric Value	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	26

Plaintext	0	1	2	3	4	5	6	7	8	9
Numeric Value	27	28	29	30	31	32	33	34	35	36

### Example

Plaintext	23	E	W	I	L	L	A	R	R	I	V	E	A	T	5	A	M
Key (phrase)	20	H	I	S	T	H	E	D	A	Y	T	H	I	S	T	H	E
Ciphertext	43	M	5	1	5	T	F	V	S	7	F	M	J	B	C	I	R

- This is essentially multiple Caesar-type ciphers
- Main advantage is that same plaintext gets mapped onto different ciphertext.



# Polyalphabetic Substitution cntd....

Plaintext	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
Numeric Value	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	26

Plaintext	0	1	2	3	4	5	6	7	8	9
Numeric Value	27	28	29	30	31	32	33	34	35	36

Plaintext	W	E	W	I	L	L	A	R	R	I	V	E	A	T	5	A	M
Key (phrase)	T	H	I	S	T	H	E	D	A	Y	T	H	I	S	T	H	E
Ciphertext	G	M	5	1	5	T	F	V	S	7	F	M	J	B	C	I	R

**Encryption Algorithm:**

$$C_v = (P_v + K_v) \text{ mod } 36$$

**Example:** W + T =

$$(23 + 20) \text{ mod } 36 = 7 = \mathbf{G}$$

**mod (modulo)** refers to the remainder resulting from a division operation.

**e.g.** 10 mod 7 = 3  
(7 divides 10 and leaves a *remainder* of 3)

**e.g.** 50 mod 9 = 5  
(9 divides 50 and leaves a *remainder* of 5)



Plaintext	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
Numeric Value	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	26

Plaintext	0	1	2	3	4	5	6	7	8	9
Numeric Value	27	28	29	30	31	32	33	34	35	36

Plaintext	W	E	W	I	L	L	A	R	R	I	V	E	A	T	5	A	M
Key (phrase)	T	H	I	S	T	H	E	D	A	Y	T	H	I	S	T	H	E
Ciphertext	G	M	5	1	5	T	F	V	S	7	F	M	J	B	C	I	R


**Decryption Algorithm:**

$$P_V = (C_V - K_V) \bmod 36$$



**Example:**  $G - T = (7 - 20) \bmod 36$   
 $= -13 \bmod 36 = 23 = W$

**mod (modulo) of negative numbers....:**  
**Remember,**  $10 \bmod 7 = 3$   
 (1 multiple of 7 "fits in to / is smaller than" 10 and leaves a *remainder* of 3)  
**And....**  $50 \bmod 9 = 5$   
 (5 multiples of 9 "fit in to / is smaller than" 50 and leaves a *remainder* of 5)  
**And....**  $-13 \bmod 36 = 23$   
 (-1 multiples of 36 (= -36) "fits in to / is smaller than" -13 and leaves a *remainder* of 23)



## Cryptography: Algorithms and Keys

- The strength of a cryptosystem is a function of:
  - The strength of the algorithm
  - The length (or size) of the key
  
- The assumption is that the general method of encryption (*i.e.* the algorithm is known), and that the security of the system lies in the **secracy of the key**.
  - *EG:* General idea of a combination pad lock is known, strength is in secret combination.
  
- The larger the key, the higher the work factor for the cryptanalyst.
  
- A **brute-force attack** means trying all possible key values.

## Cryptographic Keys

- In our earlier example, we had a phrase as our key to unlocking the secret message:

T	H	I	S	T	H	E	D	A	Y
---	---	---	---	---	---	---	---	---	---

... and each letter of the phrase could take on 1 of 26 possible **numeric values**.

**Ours are (T= 20, H= 8 etc....)**

20	08	09	19	20	08	05	04	01	25
----	----	----	----	----	----	----	----	----	----

- As we had 10 characters in our phrase, and each character could be any 1 of 26 possible values, our potential **key space** is  $26^{10}$  *i.e.* there are 141,167,095,653,376 possible keys or "phrases".
- Possible key value range: 00000000000000000000 – 26262626262626262626  
(ours is 20080919200805040125)

## Cryptographic Keys

- Lets think of keys as multiples of bits or bytes in a computer system
- If a key is 8 bits long, then there are  $2^8 = 256$  possible keys.
- A key that is 56-bits long has  $2^{56}$  possible key values. **A very large number!**  
**72,057,594,037,927,936** possible values in fact!
- **If a computer could try one million keys per second, it would take over 2000 years to try all key values.**

**72,057,594,037,927,936** possible values @ **1,000,000 tests per second....**

- $72,057,594,037,927,936 / 1,000,000 = 72057594037$  **seconds** required to test all values
- $72057594037 / 60 = 1,200,959,900$  **minutes** required to test all values
- $1,200,959,900 / 60 = 20,015,998.33$  **hours** required to test all values
- $20,015,998.33 / 24 = 833,999.93$  **days** required to test all values. / 365 = **2,284 YEARS**



## Cryptographic Keys

- If a **64-bit key** were used, it would take **600,000 years** to try all possible key values (at a test rate 1 million keys per second)
- For a **128-bit key**, it would take  $10^{25}$  years. The universe is only  $10^{10}$  years old.
- **When trying a brute-force attack need to consider:**
  - Number of keys to be tested
  - The speed of each test.

## Some Limitations of Symmetric Cryptography

- Consider our problem statement from earlier (which outlined what our desired solution should solve...)

The basic problem to be solved is that of **encrypting and decrypting data**.

– How should we encode an important message in such a way that the receiver should be able to decipher it, but not an eavesdropper?

– Moreover, can then message be **signed** by the sender so that:

- (1) The receiver can be sure that only the sender could have sent it.
- (2) The sender cannot later deny having sent it
- (3) The receiver, having received the signed message, cannot sign a message in the senders name, not even additional versions of the very message that has just been received.

## Some Limitations of Symmetric Cryptography

- Does not address the *signature* issue.....
  - Receiver could make up fake messages?
  - Sender could deny having sent authentic messages?
- Another major drawback (particularly in a distributed environment like the Internet) is the issue of key distribution and management
- Symmetric systems require parties to cooperate in the generation and distribution of key pairs. Not scalable (even where it might be “culturally” feasible).

## Cryptography – Key Distribution

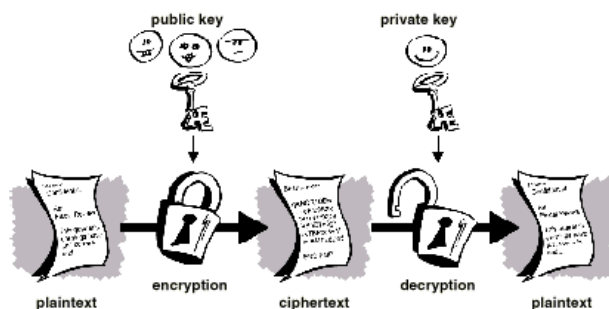
- Ciphers have improved significantly in terms of complexity since the days of Caesar. However.....
- As noted, a big issue with symmetric cryptographic systems is that of *key management*, specifically with respect to *transferring keys*.
- How do the sender and receiver agree on the same key?
- Split the key into several parts?
- Other....?

## Asymmetric Cryptography: Public Key Cryptography

- The problems of key distribution are solved by **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, or secret key** for decryption.
- 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.
- Crucially, it is **computationally infeasible** to deduce the private key from the public key.

## Public Key Cryptography

- The primary benefit of public key cryptography is that it allows people who have no pre-existing 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.

## A little bit of number theory

**Prime number:** A number is *prime* if it is greater than 1 and if its only factors are itself and 1.

1 is not prime (1 is not greater than 1....)

2 is prime (the ONLY even prime)

3 is prime

4 is not prime (no other even number – except 2 – is prime) ....  $4 \times 1$  ,  $2 \times 2$

5 is prime (but being odd does not necessarily make you prime)

6 is not prime  $6 \times 1$  ,  $3 \times 2$

7 is prime (..... But, again, being odd does not necessarily make you prime)

8 is not prime ....  $8 \times 1$  ,  $2 \times 4$

9 is NOT prime (but it is odd.....).....  $9 \times 1$  ,  $3 \times 3$

## RSA Algorithm

- First published in 1978 (and still secure).  
(Researchers: Ron **R**ivest, Adi **S**hamir, Len **A**dleman)
- A block ciphering scheme, where the plaintext and ciphertext are integers between 0 and  $n-1$ , for some value  $n$ .

**(Foundation is in number theory in Mathematics, based on Euler's generalisation of Fermat's Theorem).**

### **Format**

A block of plaintext (message), **M**, gets transformed into a cipher block **C**

$$\mathbf{C} = \mathbf{M}^e \bmod n \quad (\text{Enciphering Transformation})$$

$$\mathbf{M} = \mathbf{C}^d \bmod n = (\mathbf{M}^e)^d \bmod n = \mathbf{M}^{ed} \bmod n \quad (\text{Deciphering})$$

## RSA Algorithm

- Both sender and receiver know the value of **n** ← **Public**
- Sender / everyone knows the value of **e** ← **Public**
- Receiver (only) knows the value of **d** ← **Private**

Thus the public key is  $\{e, n\}$   
..... and the private key is  $\{d, n\}$

### Requirements:

Values exist for  $e, d$  and  $n$  such that  $M^{ed} = M \pmod n$  for all  $M < n$   
 $M^e$  and  $C^d$  can be calculated relatively easily for all values of  $M < n$   
It is infeasible to determine  $d$  given  $e$  and  $n$



25

25

## RSA: A (simple) example

- Suppose I wish to send my access number securely across an "open" network.  
*e.g.* PIN = 2345
- I don't want my account to be hacked; neither does my employer!
- My employer implements a security system (based on the RSA algorithm, for instance) and tells me (and everyone else).....:

"When sending us private data such as your PIN, encrypt it using RSA. This is the relevant public key information:  **$e = 7, n = 33$** "

26

26

## Encrypting my "plaintext" PIN: 2 3 4 5

$$C = P^e \text{ mod } n$$

$$e = 7, n = 33$$

### Encrypting "2":

$$C = P^e \text{ mod } n$$

$$C = 2^7 \text{ mod } 33$$

$$C = 128 \text{ mod } 33 = 29$$

So (plaintext) **2** encrypts to  
(ciphertext) **29**

### Encrypting "3":

$$C = P^e \text{ mod } n$$

$$C = 3^7 \text{ mod } 33$$

$$C = 2187 \text{ mod } 33 = 9$$

So (plaintext) **3** encrypts  
to (ciphertext) **9**

### Encrypting "4":

$$C = 4^7 \text{ mod } 33$$

$$C = 16384 \text{ mod } 33 = 16$$

So (plaintext) **4** encrypts  
to (ciphertext) **16**

### Encrypting "5":

$$C = 5^7 \text{ mod } 33$$

$$C = 78125 \text{ mod } 33 = 14$$

So (plaintext) **5** encrypts  
to (ciphertext) **14**

27

27

## Encrypting my "plaintext" PIN: 2345

Plaintext:            2     3     4     5

Ciphertext:         29    9     16    14

- An "attacker" listening in on the connection might capture your encrypted PIN (*i.e.* they could discover **29, 9, 16, 14**)
- Furthermore they would know that this is the output of the known RSA algorithm ( $C = P^e \text{ mod } n$ )
- .... They would know that  $e = 7$  and that  $n = 33$  as these are publicly available)
- **BUT.....** They won't be able to figure out **P** (the plaintext) in a **reasonable amount of time.**

28

28

## Decrypting 29, 9, 16, 14

- Can only be done (in a reasonable amount of time) if you know the private / secret key, **d**.
- **d** is not publicly available and is jealously guarded by the owner.
- Decryption is straightforward if you know the value of d.

$$C^d \bmod n \rightarrow P$$

- Raising the ciphertext to the power of d, modulo n, will give you back the plaintext.

29

29

## Decrypting Ciphertext: 29 9 16 14

$$P = C^d \bmod n$$

$$d = 3, n = 33$$

### Decrypting "29":

$$P = C^d \bmod n$$

$$P = 29^3 \bmod 33$$

$$P = 24389 \bmod 33 = 2$$

So (ciphertext) **29** decrypts to (plaintext) **2**

### Decrypting "9":

$$P = C^d \bmod n$$

$$P = 9^3 \bmod 33$$

$$P = 729 \bmod 33 = 3$$

So (ciphertext) **9** decrypts to (plaintext) **3**

### Decrypting "16":

$$P = 16^3 \bmod 33$$

$$P = 4096 \bmod 33 = 4$$

So (ciphertext) **16** decrypts to (plaintext) **4**

### Decrypting "14":

$$P = 14^3 \bmod 33$$

$$P = 2744 \bmod 33 = 5$$

So (ciphertext) **14** decrypts to (plaintext) **5**

30

30

## Decryption of my PIN

Ciphertext:        29    9    16    14

Plaintext:        2    3    4    5

- Note that a knowledge of the encryption algorithm, a knowledge of the encryption key and a copy of the ciphertext **is not sufficient** to get back to the plaintext (in a reasonable amount of time).
- You must also know the value of  $d$ , the private key.
- The beauty of RSA is that the encryption key and process can be made public without compromising the security of the system. This solves the “key distribution” problem outlined earlier.

31

31

## Next Steps

- Future session to look at RSA in more detail including some of the underpinnings of the algorithm.
- Authentication and signatures using RSA
- Secure Sockets Layer (SSL) Algorithm
- RSA performance
- Attacking RSA

### **To do (by you):**

- Contribute to the discussion forum on Moodle / LearnOnline
- Further reading in this area

32

32