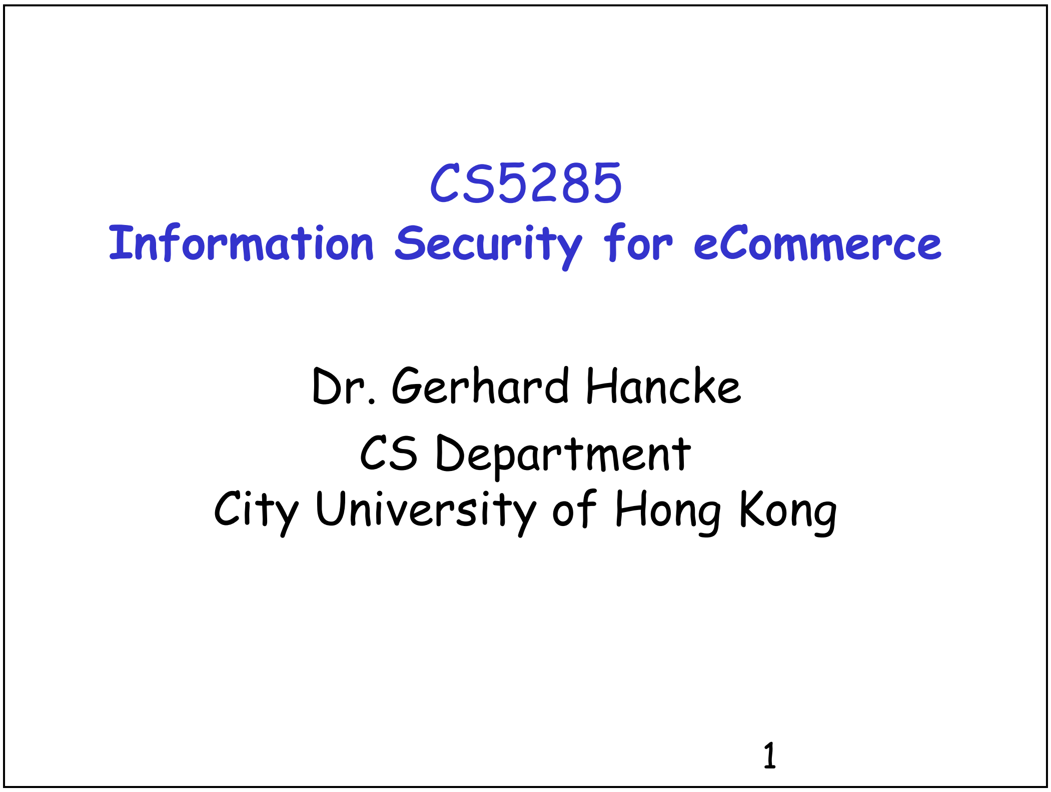
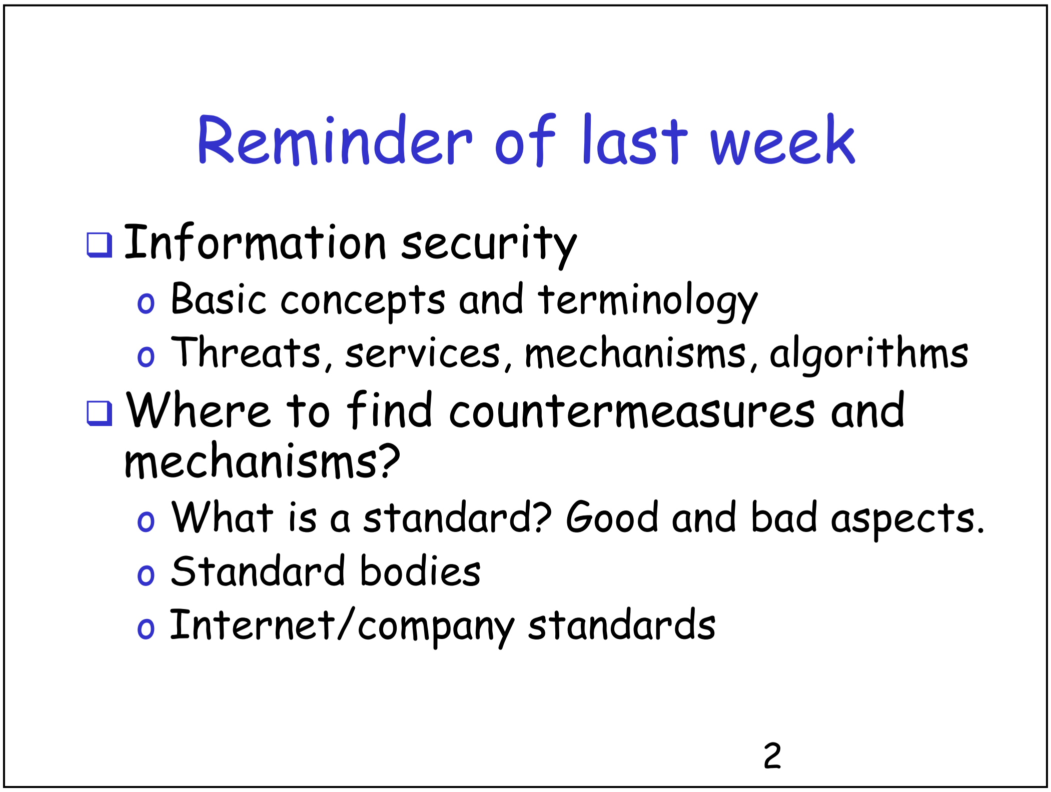


CS5285
Information Security for eCommerce

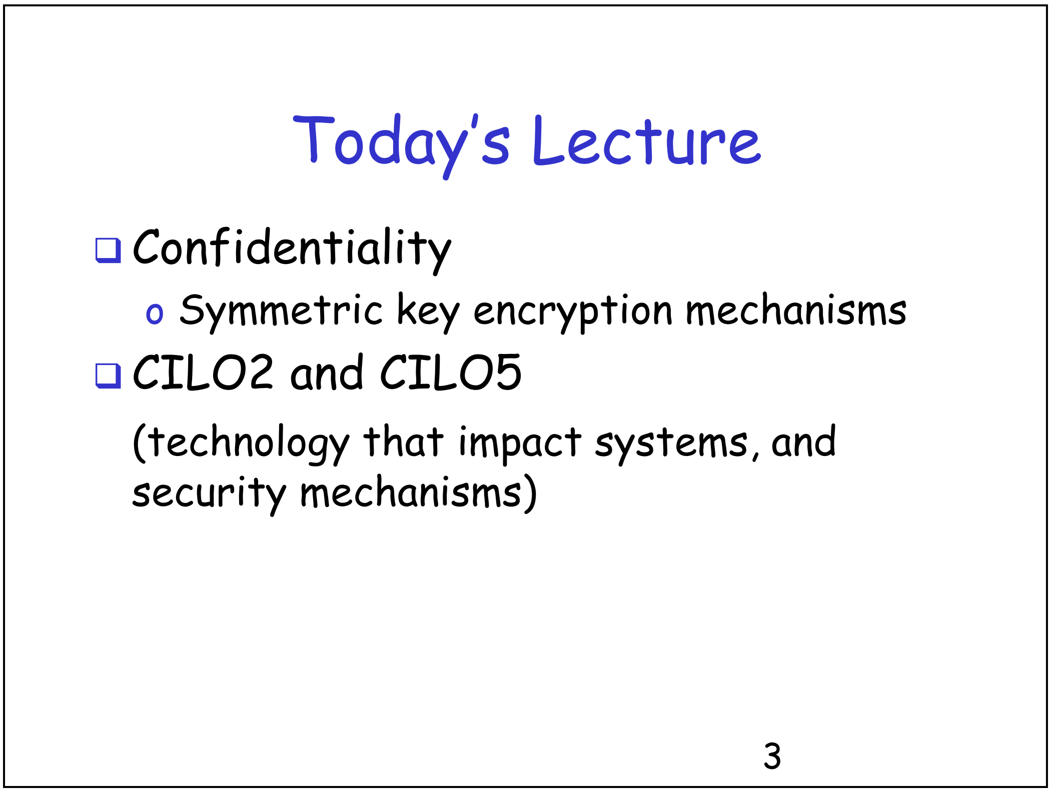
Dr. Gerhard Hancke

CS Department
City University of Hong Kong



Reminder of last week

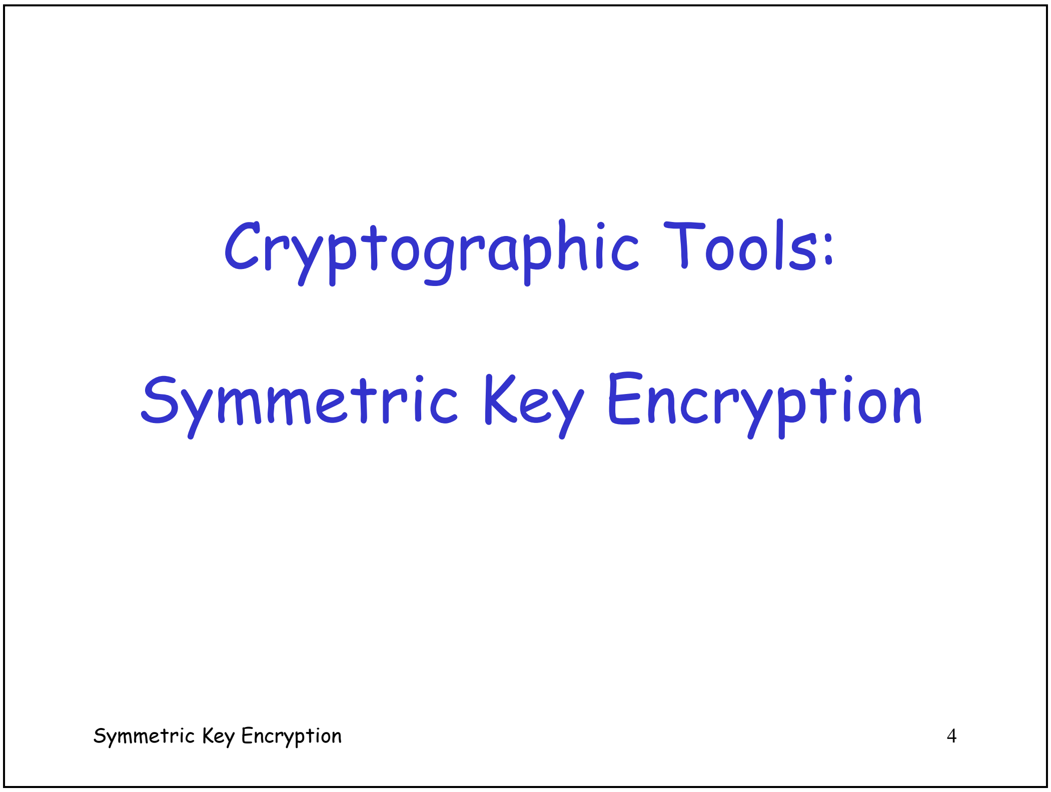
- Information security
  - Basic concepts and terminology
  - Threats, services, mechanisms, algorithms
- Where to find countermeasures and mechanisms?
  - What is a standard? Good and bad aspects.
  - Standard bodies
  - Internet/company standards



Today's Lecture

- Confidentiality
  - Symmetric key encryption mechanisms
- CILO2 and CILO5

  (technology that impact systems, and security mechanisms)



Cryptographic Tools:

Symmetric Key Encryption

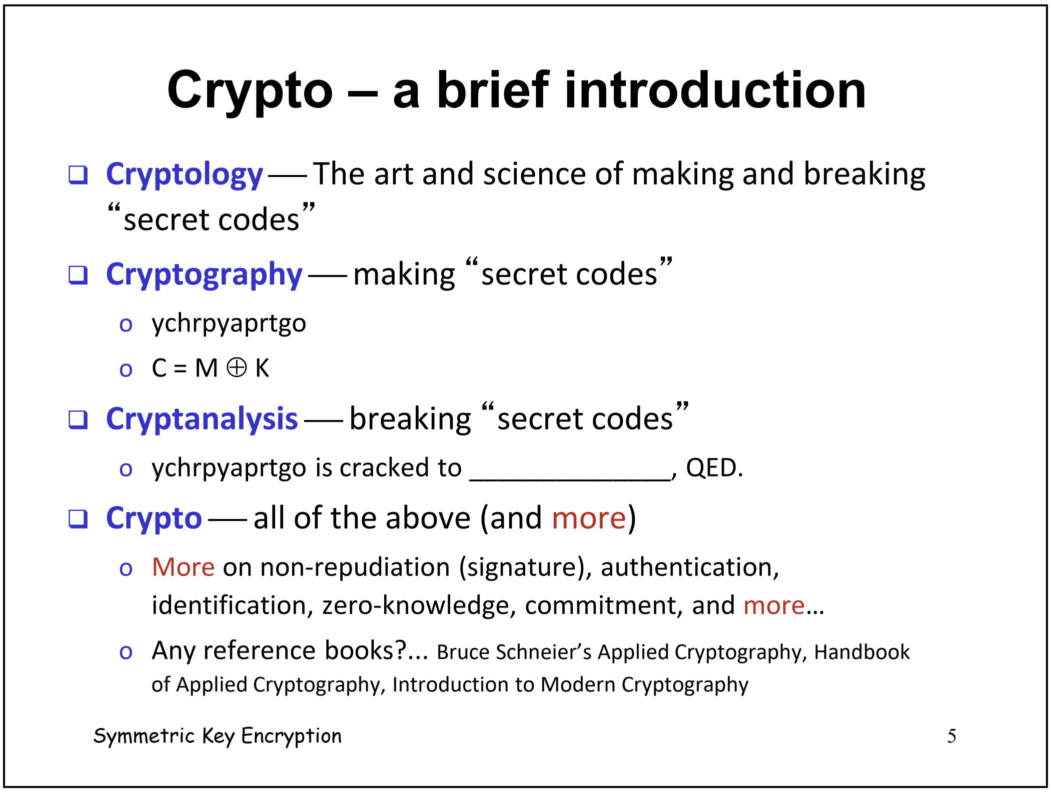
Symmetric Key Encryption 4

Encryption has fascinated people for centuries….this lecture gives us a little bit of history of security and encryption

Looking at symmetric or secret key cryptography (this simple means the sender and recipient uses the same key)

The other approach we will be looking at later is asymmetric or public key cryptography.

Note the we are dealing specifically with encryption mechanisms (symmetric key crypto can also be used for other purposes.)



Crypto – a brief introduction

- ❑ **Cryptology** — The art and science of making and breaking "secret codes"
- ❑ **Cryptography** — making "secret codes"
  - o ychrpyaprtgo
  - o C = M ⊕ K
- ❑ **Cryptanalysis** — breaking "secret codes"
  - o ychrpyaprtgo is cracked to ______________, QED.
- ❑ **Crypto** — all of the above (and more)
  - o More on non-repudiation (signature), authentication, identification, zero-knowledge, commitment, and more…
  - o Any reference books?... Bruce Schneier's Applied Cryptography, Handbook of Applied Cryptography, Introduction to Modern Cryptography

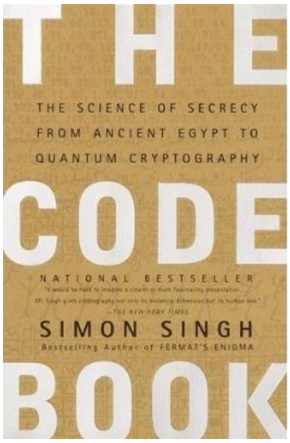
Reference slides

Cryptography was originally mostly encryption

Ychrpyaprtgo is cracked to 'Cryptography'

Allocate 12 spaces for the ciphertext (1-12) – start writing the letters from C in the even spaces until o is the final letter in space 12, now start filling in the remaining letters into the odd spaces starting with space 11.

These days 'Crypto' - and Cryptography is much more than just secret codes and confidentiality.

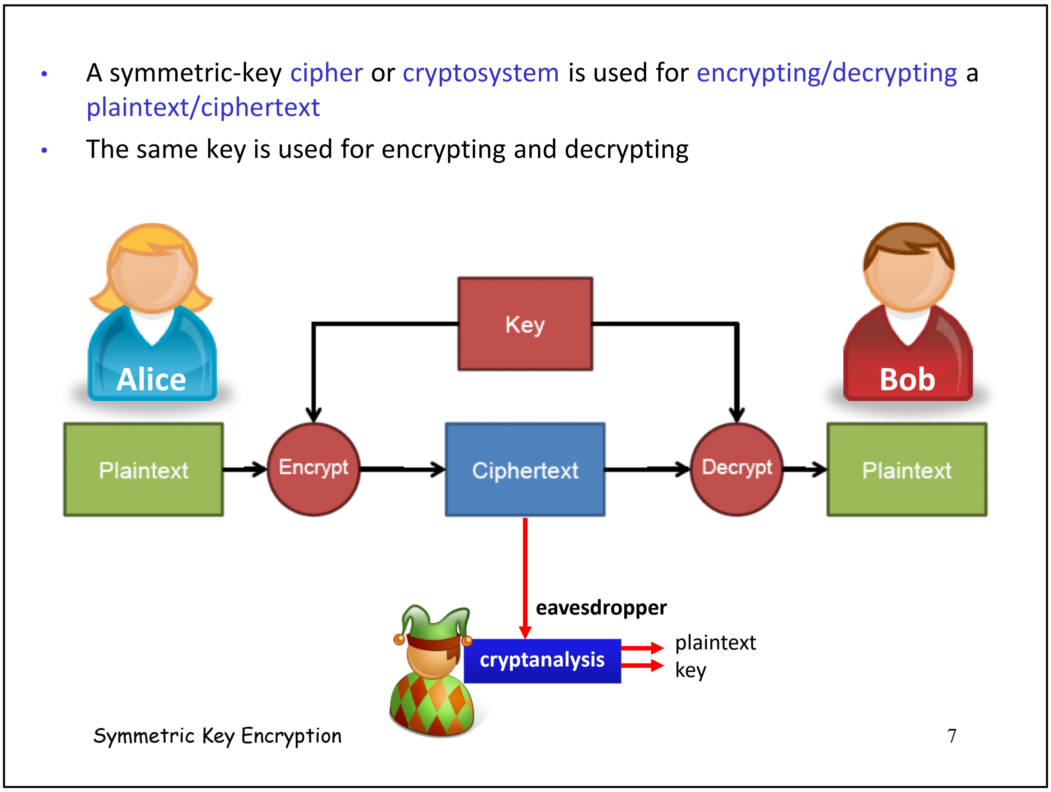

THE
THE SCIENCE OF SECRECY FROM ANCIENT EGYPT TO QUANTUM CRYPTOGRAPHY
CODE
NATIONAL BESTSELLER
—THE NEW YORK TIMES
SIMON SINGH
Bestselling Author of FERMAT'S ENIGMA
BOOK

"The history of codes and ciphers is the story of the centuries-old battle between codemakers and codebreakers, an intellectual arms race that has had a dramatic impact on the course of history."
– Simon Singh, The Code Book

References slide

This is essentially the same remarks one can make about anything in security – it is an 'arms race' between those who design secure systems and those who wish to circumvent security countermeasures.

Still consider historical methods to give us an idea of how this cycle works, and why the need was there to improve.



- A symmetric-key cipher or cryptosystem is used for encrypting/decrypting a plaintext/ciphertext
- The same key is used for encrypting and decrypting

Symmetric Key Encryption 7

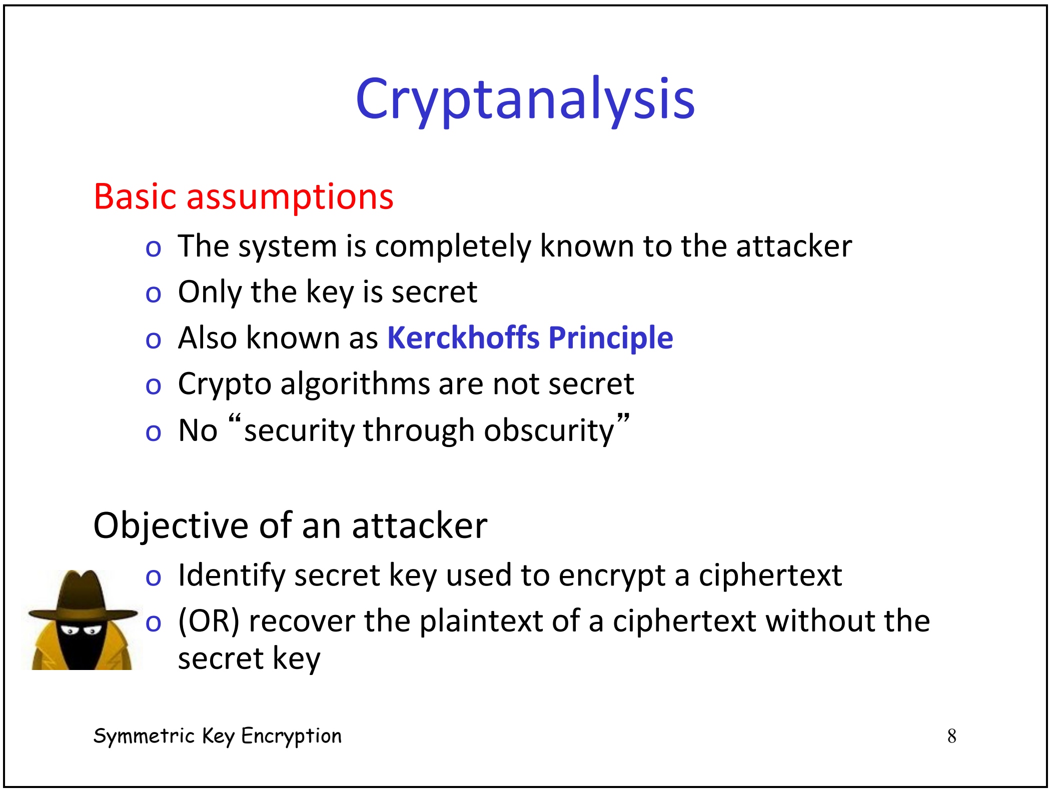
You need to know this model and be able to explain the similarities and differences between symmetric and asymmetric mechanisms.

In a symmetric system the encryption and decryption key is the same!

The message is plaintext, which becomes ciphertext after encryption function.

The decryption function takes the ciphertext and makes it into plaintext.

An attacker's goals it to either get the key or the plaintext – from the observed ciphertext.



Cryptanalysis

Basic assumptions

- The system is completely known to the attacker
- Only the key is secret
- Also known as **Kerckhoffs Principle**
- Crypto algorithms are not secret
- No “security through obscurity”

Objective of an attacker

- Identify secret key used to encrypt a ciphertext
- (OR) recover the plaintext of a ciphertext without the secret key

Symmetric Key Encryption 8

You need to be able to explain Kerckhoffs principles and know the objectives of an attacker that would result in encryption being ‘broken’

Look at the previous slide again -

Which of the following need to be kept secret when doing symmetric encryption?

a) Encryption/decryption key

b) Encryption/decryption algorithm

Only a – keeping b secret does not adhere to Kerckhoffs principle…

In modern cryptography we are mostly concerned with the key being found – more so than the message – many times an attacker can guess the basic messages (e.g. repetitive network commands)

For interest

System is known to the attacker – what does he know about our plaintext/ciphertext

There are different approaches

Ciphertext only (attacker sees only encrypted data), known plaintext (attacker sees ciphertext but he knows the plaintext), chosen plaintext (the attacker can ask our system to encrypt plaintext messages he chooses)!

Three basic types of cryptanalysis vs encryption

Ciphertext only attack

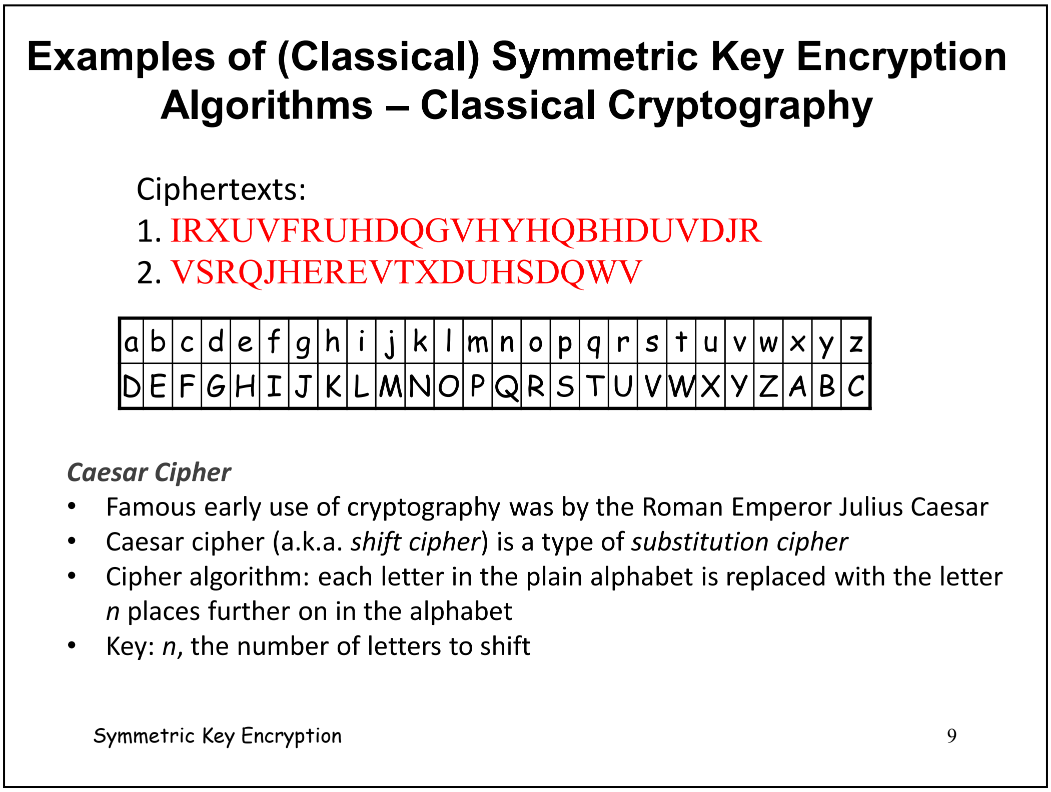
–The attacker only possesses some ciphertext

•Known plaintext attack

–The attackers possesses some corresponding pairs of plaintext and ciphertext

•Chosen plaintext attack

–The attacker has temporary access to the encryption process and hence can **choose** plaintexts and generate the corresponding ciphertexts



Examples of (Classical) Symmetric Key Encryption Algorithms – Classical Cryptography

Ciphertexts:

1. IRXUVFRUHDQGVHYHQBHDUVDJR
2. VSRQJHEREVTXDUHSDQWV

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

Caesar Cipher

- Famous early use of cryptography was by the Roman Emperor Julius Caesar
- Caesar cipher (a.k.a. *shift cipher*) is a type of *substitution cipher*
- Cipher algorithm: each letter in the plain alphabet is replaced with the letter *n* places further on in the alphabet
- Key: *n*, the number of letters to shift

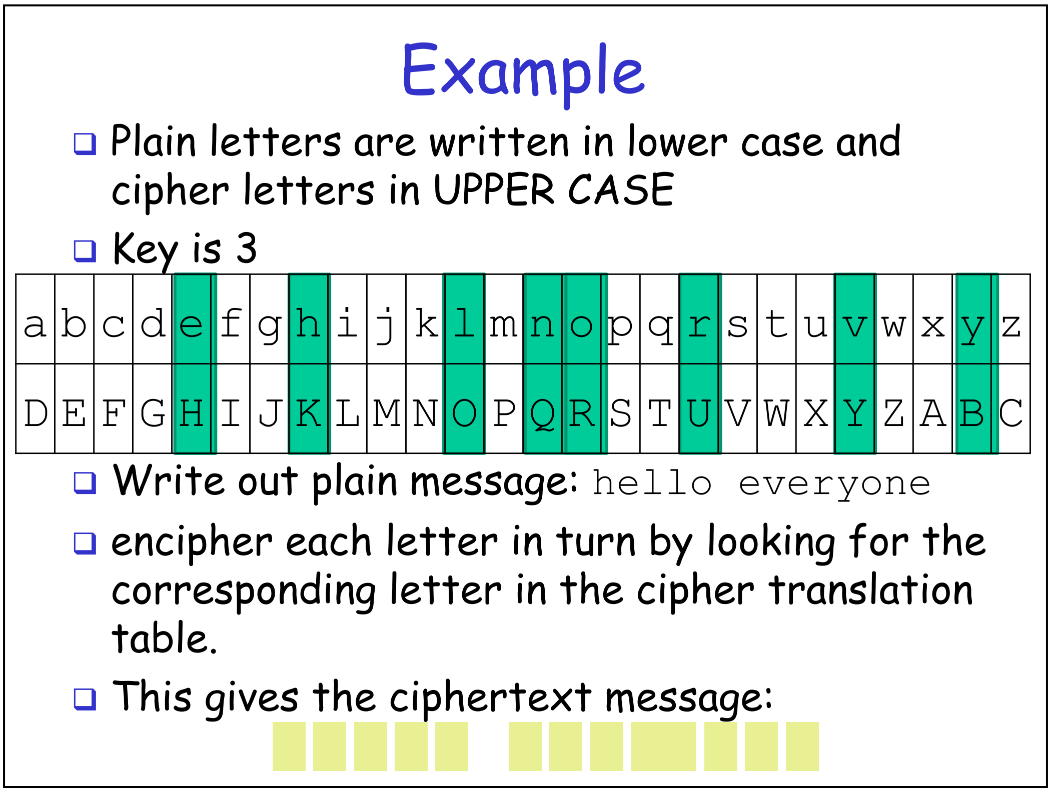
Look at next two slides then do examples.

I expect you to know what a substitution cipher is, and know the special case of Ceasar cipher type.

What is the key in this case? N=3 (A Caesar cipher is a shift cipher with N=3)

1. IRXUVFRUHDQGVHYHQBHDUVDJR
   FOURSCORE AND SEVEN YEARS AGO

2. VSRQJHEREVTXDUHSDQWV
Sponge Bob Square Pants



Example

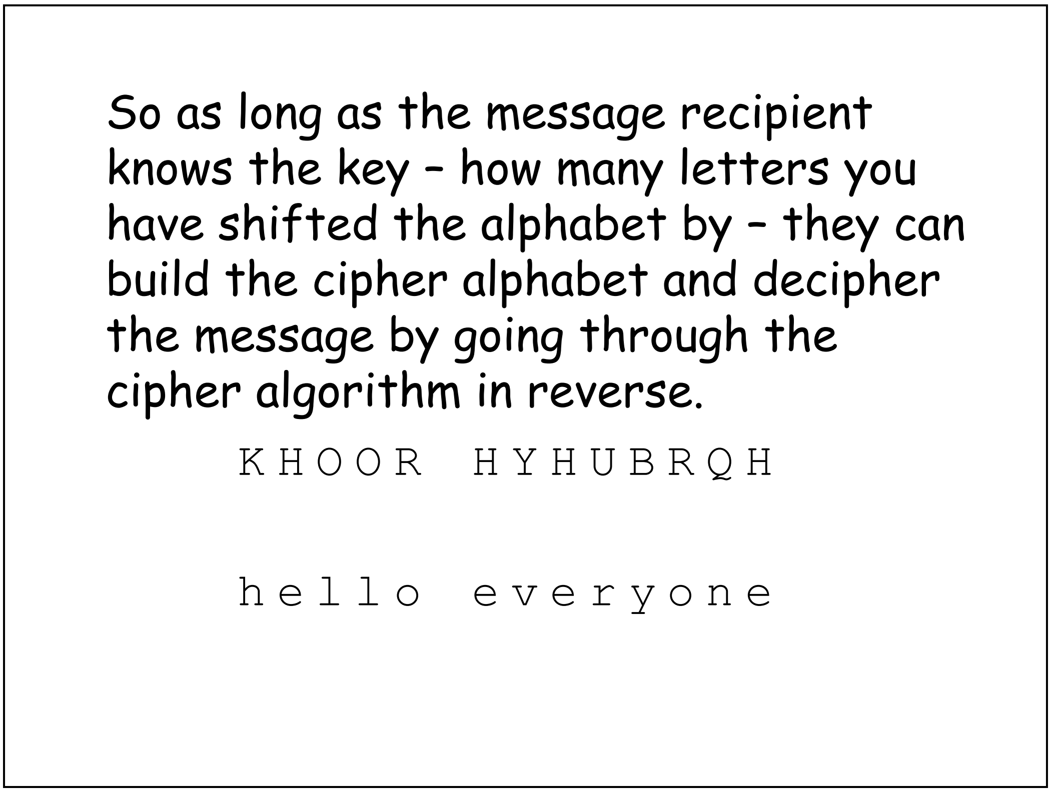
- Plain letters are written in lower case and cipher letters in UPPER CASE
- Key is 3

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

- Write out plain message: `hello everyone`
- encipher each letter in turn by looking for the corresponding letter in the cipher translation table.
- This gives the ciphertext message:

Hello everyone is:

KHOORHYHURQH



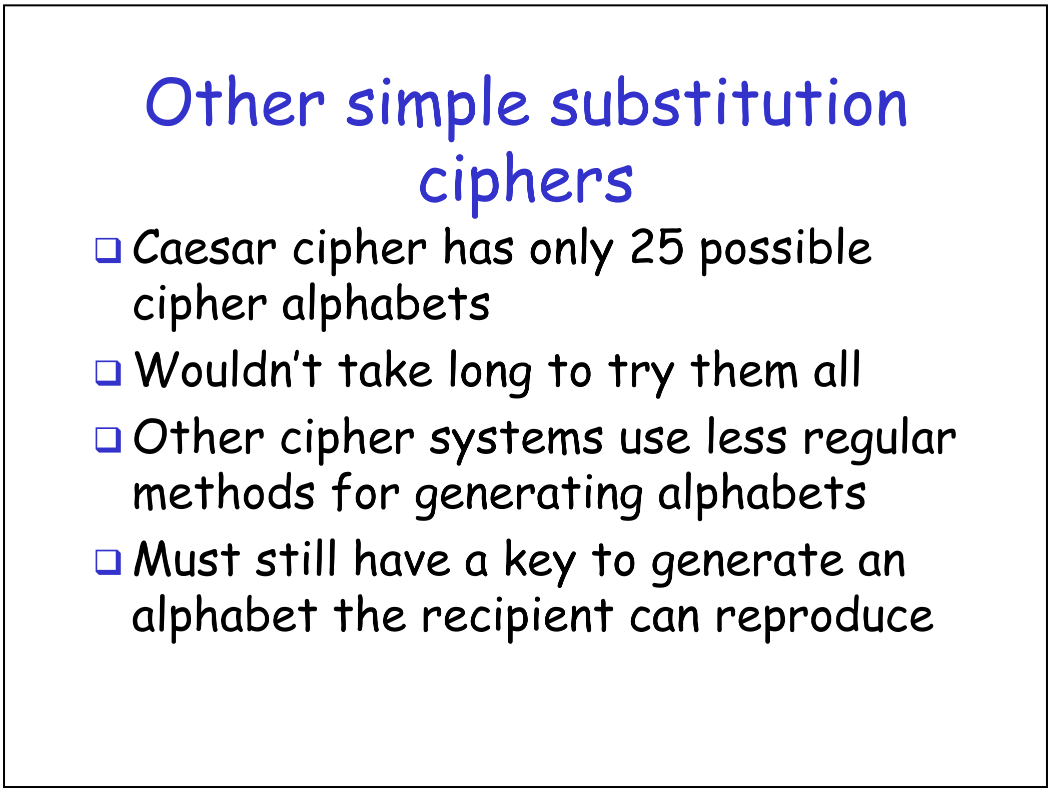
So as long as the message recipient knows the key – how many letters you have shifted the alphabet by – they can build the cipher alphabet and decipher the message by going through the cipher algorithm in reverse.

```
KHOOR HYHUBRQH

hello everyone
```

Go back to slide 9 and try the examples.

What is the key in this case?

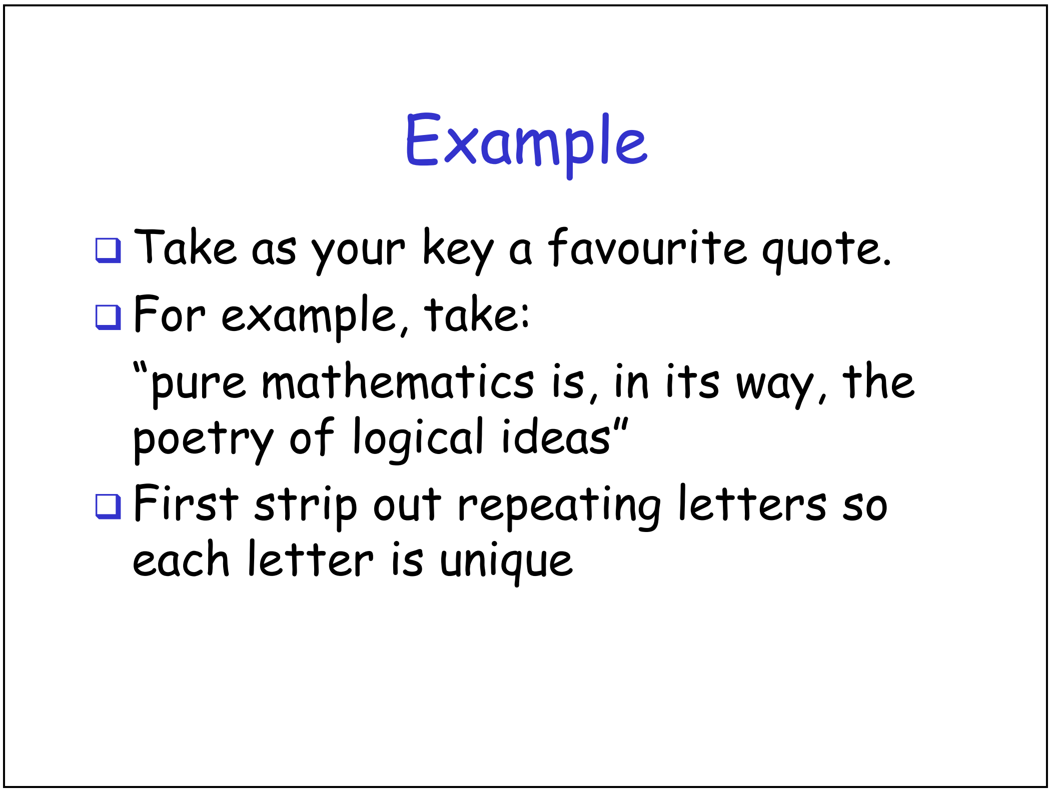


Other simple substitution ciphers

- Caesar cipher has only 25 possible cipher alphabets
- Wouldn't take long to try them all
- Other cipher systems use less regular methods for generating alphabets
- Must still have a key to generate an alphabet the recipient can reproduce

In the case of a shift cipher what is the secret key?

The offset of the shift.



Example

- Take as your key a favourite quote.
- For example, take:

  “pure mathematics is, in its way, the poetry of logical ideas”
- First strip out repeating letters so each letter is unique

Quote from Albert Einstein

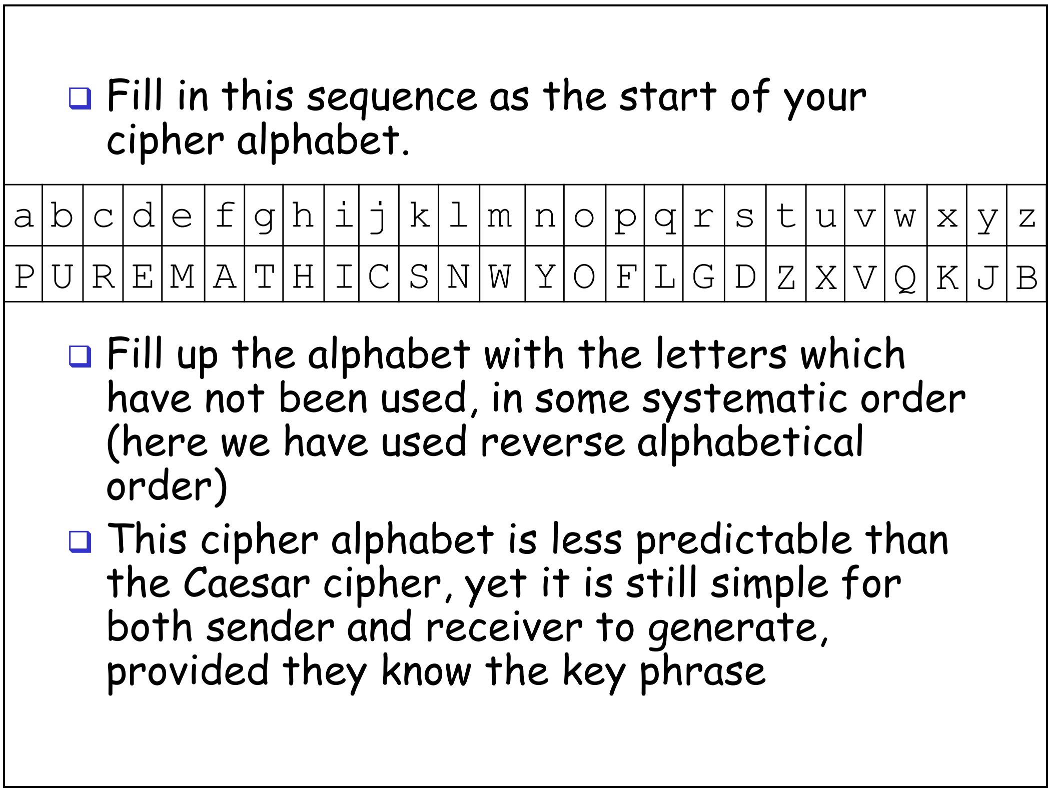
pure mathematics is, in its way,

pure*math****ics **, *n *** w*y,

the poetry of logical ideas

*** ****** of l*g**** *d***

puremathicsnwyoflgd



- Fill in this sequence as the start of your cipher alphabet.

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

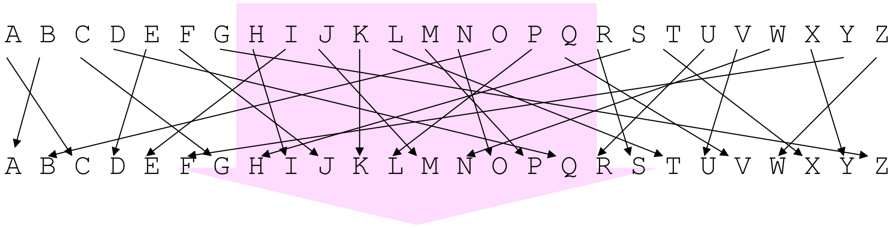
- Fill up the alphabet with the letters which have not been used, in some systematic order (here we have used reverse alphabetical order)
- This cipher alphabet is less predictable than the Caesar cipher, yet it is still simple for both sender and receiver to generate, provided they know the key phrase

What is the key in this case? What must both parties know?

You should know the substitution table but I would argue that the complete phrase is the key. This allows both parties to construct the table and encrypt/decrypt.

Simple Substitution: each plaintext letter is substituted by a distinct ciphertext letter

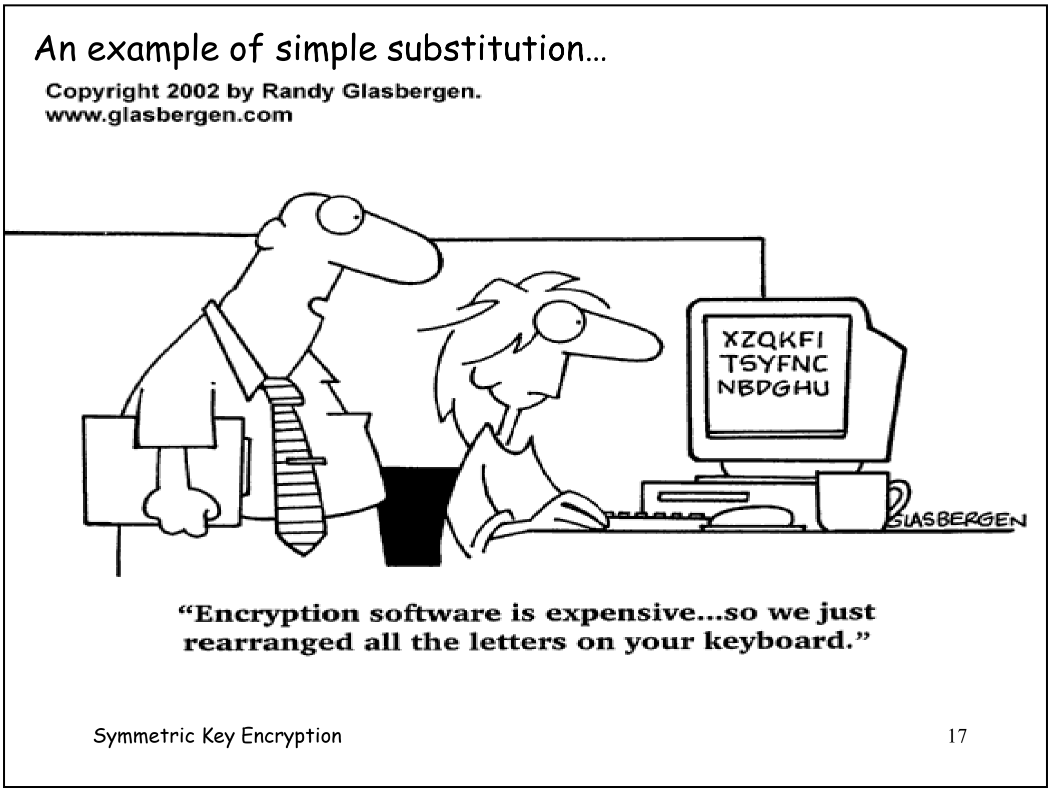
EIMBULJIWLNYANJMVLIURAHIWAI


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
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

DEPARTMENTOFCOMPUTERSCIENCE

You can choose any permutation you like as long as the recipient knows what you are using.

Shift is a special sub-category of substitution.



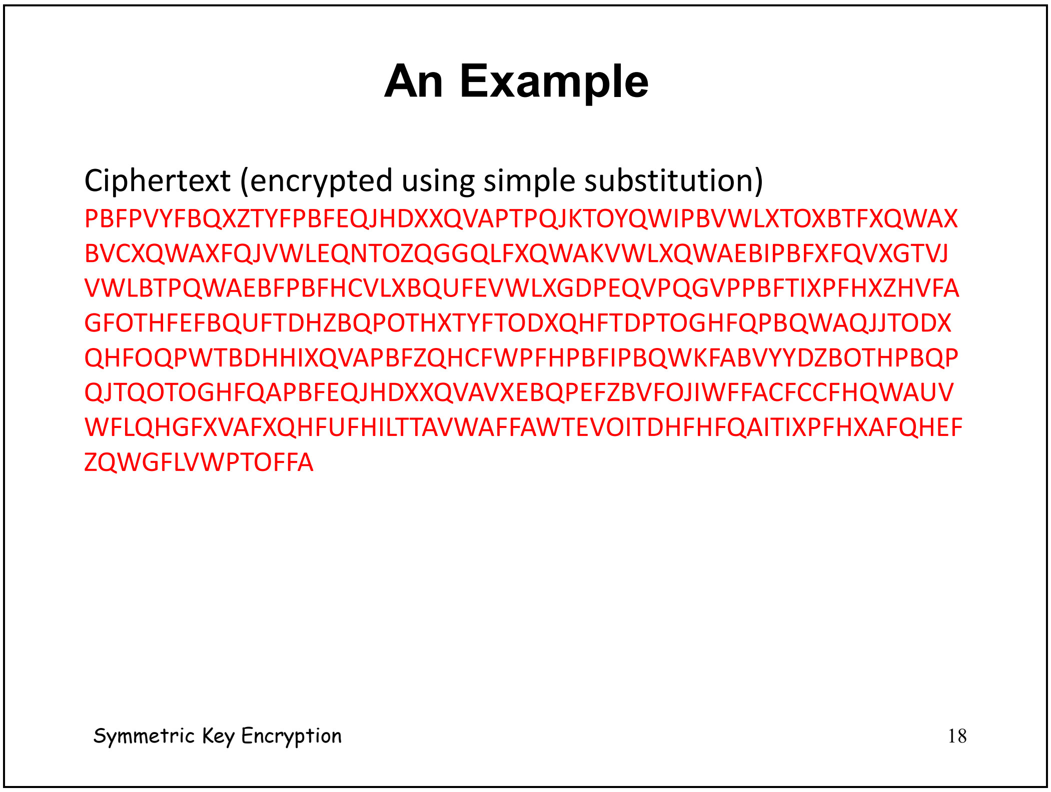
An example of simple substitution…

Copyright 2002 by Randy Glasbergen.
www.glasbergen.com

“Encryption software is expensive…so we just rearranged all the letters on your keyboard.”

Symmetric Key Encryption 17

Substitution ciphers are made for people to understand



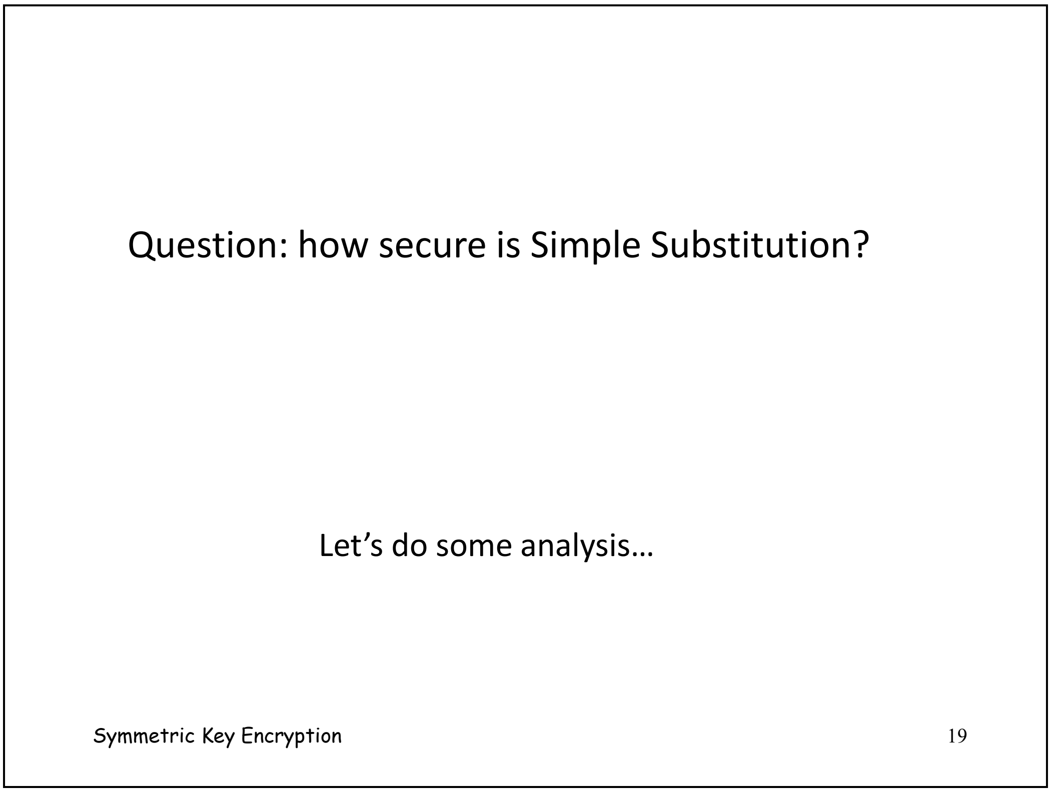
An Example

Ciphertext (encrypted using simple substitution)

PBFPVYFBQXZTYFPBFEQJHDXXQVAPTPQJKTOYQWIPBVWLXTOXBTFXQWAX
BVCXQWAXFQJVWLEQNTOZQGGQLFXQWAKVWLXQWAEBIPBFXFQVXGTVJ
VWLBTPQWAEBFPBFHCVLXBQUFEVWLXGDPEQVPQGVPPBFTIXPFHXZHVFA
GFOTHFEFBQUFTDHZBQPOTHXTYFTODXQHFTDPTOGHFQPBQWAQJJTODX
QHFOQPWTBDHHIXQVAPBFZQHCFWPFHPBFIPBQWKFABVYYDZBOTHPBQP
QJTQOTOGHFQAPBFEQJHDXXQVAVXEBQPEFZBVFOJIWFFACFCCFHQWAUV
WFLQHGFXVAFXQHFUFHILTTAVWAFFAWTEVOITDHFHFQAITIXPFHXAFQHEF
ZQWGFLVWPTOFFA

Where would you start if given a ciphertext to decode? What is your strategy?

What cipher? In this case substitution – so you know each letter maps onto another letter but you do not know the permutation for all 26 letters.



Question: how secure is Simple Substitution?

Let's do some analysis…

Symmetric Key Encryption 19

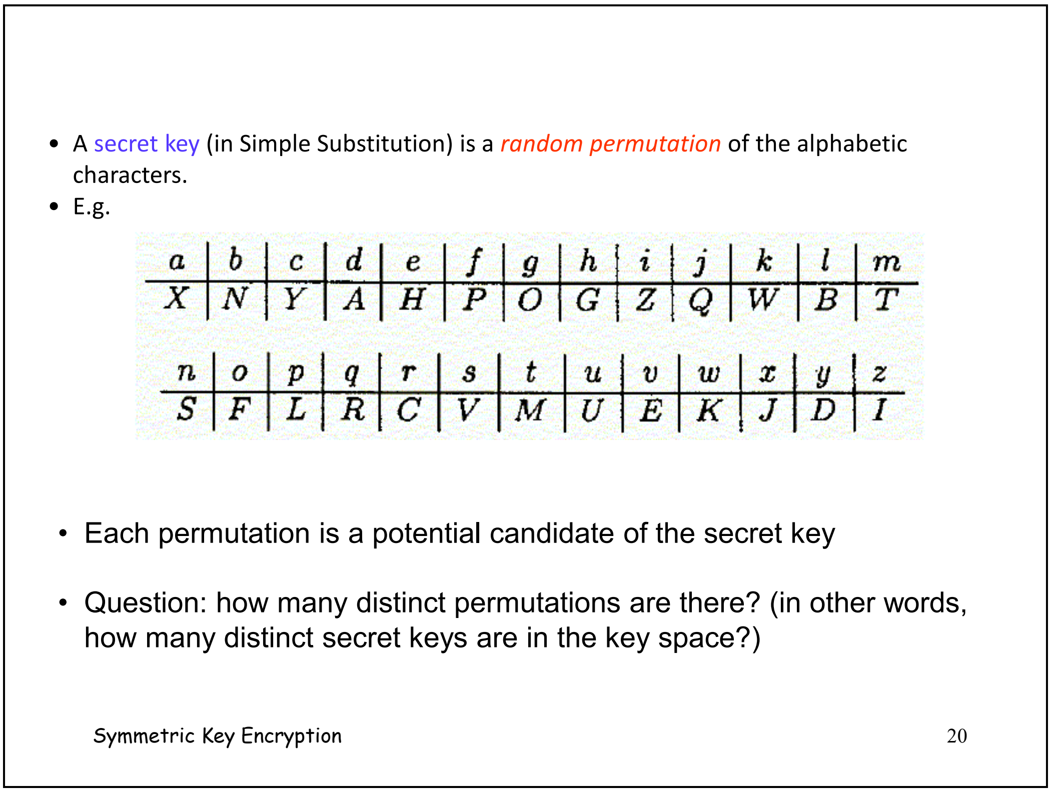
Is substitution a very weak approach in itself?

The problem is that it can only encrypt text really.

What is the key?

For Ceasar – the shift value
For a Ceasar cipher everything mapped using a uniform shift 1-25.

For substituion – the substitution alphabet



- A secret key (in Simple Substitution) is a *random permutation* of the alphabetic characters.
- E.g.

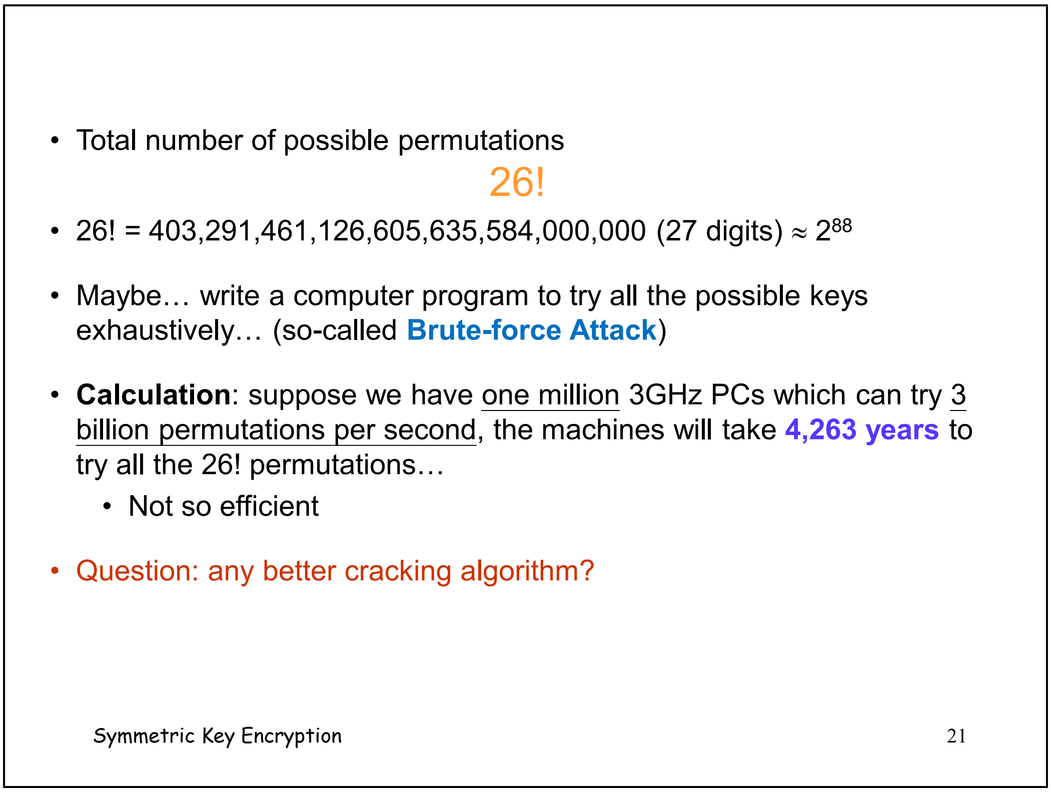
a	b	c	d	e	f	g	h	i	j	k	l	m
X	N	Y	A	H	P	O	G	Z	Q	W	B	T

n	o	p	q	r	s	t	u	v	w	x	y	z
S	F	L	R	C	V	M	U	E	K	J	D	I

- Each permutation is a potential candidate of the secret key
- Question: how many distinct permutations are there? (in other words, how many distinct secret keys are in the key space?)

Think why we said the number of keys in a Ceasar Cipher is 25 while here we say to total number of permutations is 26! (this is all possible combinations of mappings – A can map to 26 possible letters – choose 1; B can then map to 25 possible letters, choose 1; C can then map to 24 possible letters, etc….)

For a Ceasar cipher everything mapped using a uniform shift 1-25.



- Total number of possible permutations

  26!

- 26! = 403,291,461,126,605,635,584,000,000 (27 digits) ≈ 2^{88}
- Maybe… write a computer program to try all the possible keys exhaustively… (so-called **Brute-force Attack**)
- **Calculation**: suppose we have one million 3GHz PCs which can try 3 billion permutations per second, the machines will take **4,263 years** to try all the 26! permutations…
  - Not so efficient
- Question: any better cracking algorithm?

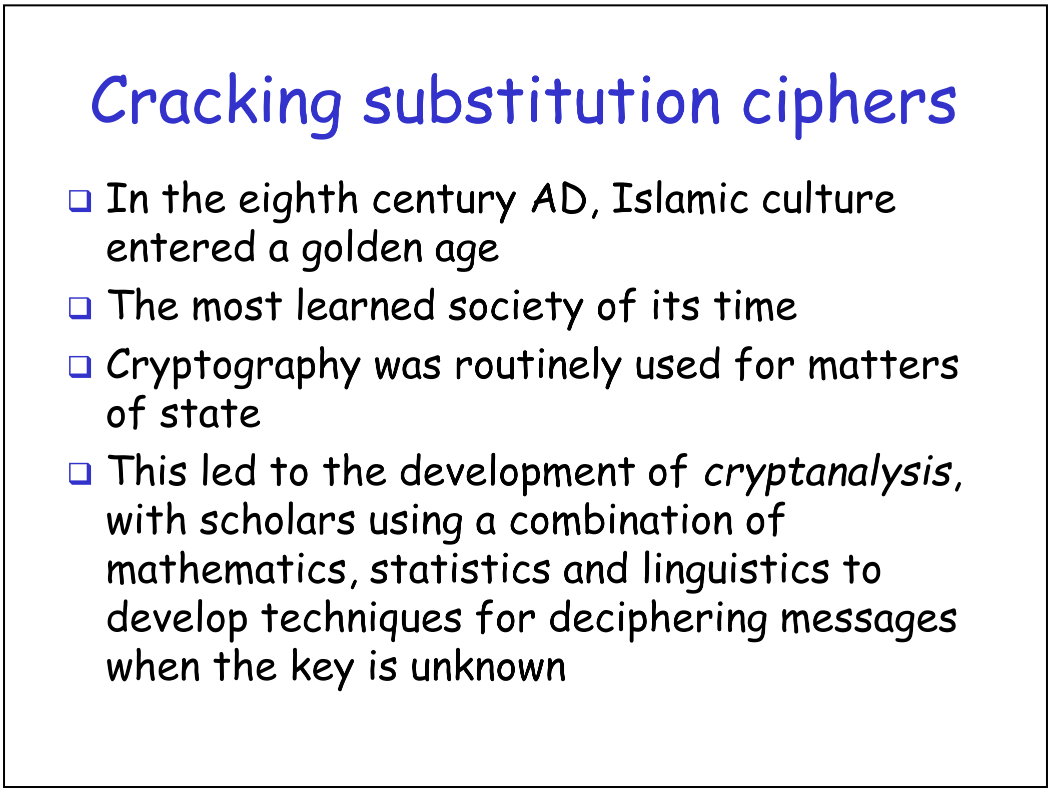
You must understand how we get to a keyspace of 26!

26! is Factorial of 26.

Brute force is not the best way of doing things – all algorithms can be brute forced! It is just that some algorithms require more effort to brute force than others.

Also think about the practical issues involved with brute force – how you do know when you found the right permutation? In substitution it is easier – the plaintext must be coherent text.

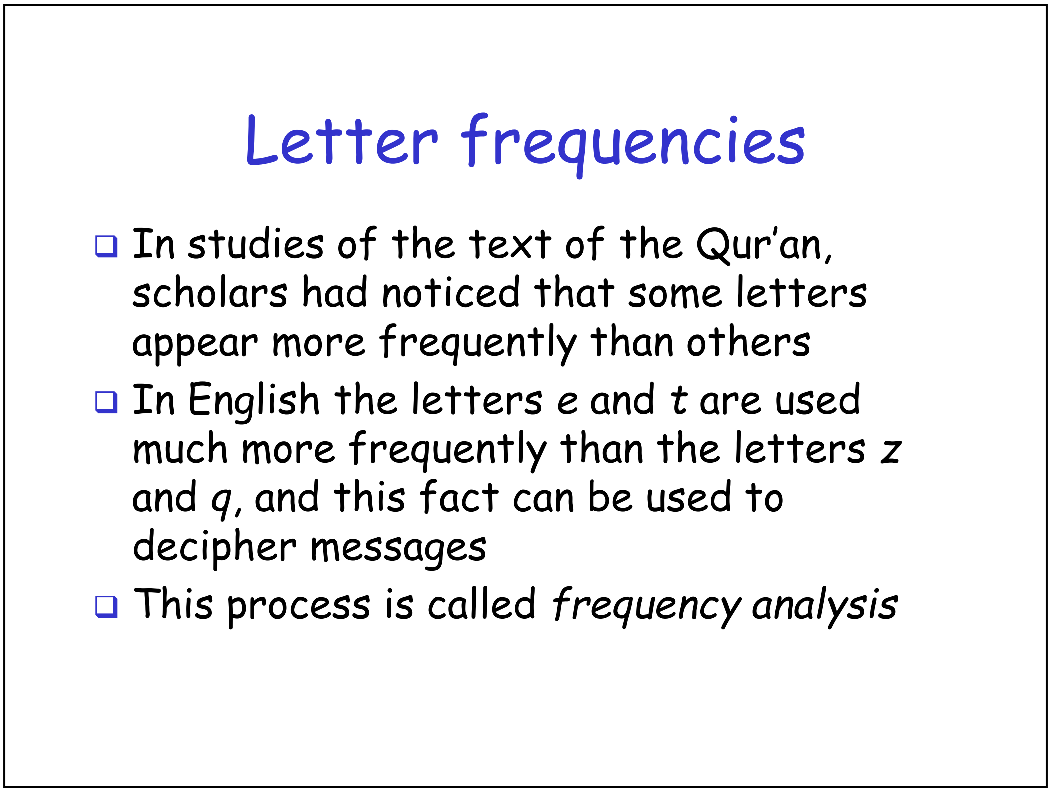
What is data represented as in computer systems? Binary data. Remember this when thinking about one time pad later…



Cracking substitution ciphers

- ❑ In the eighth century AD, Islamic culture entered a golden age
- ❑ The most learned society of its time
- ❑ Cryptography was routinely used for matters of state
- ❑ This led to the development of *cryptanalysis*, with scholars using a combination of mathematics, statistics and linguistics to develop techniques for deciphering messages when the key is unknown

This history is not important but you need to know what frequency analysis approach to cryptanalysis is, and how it works.

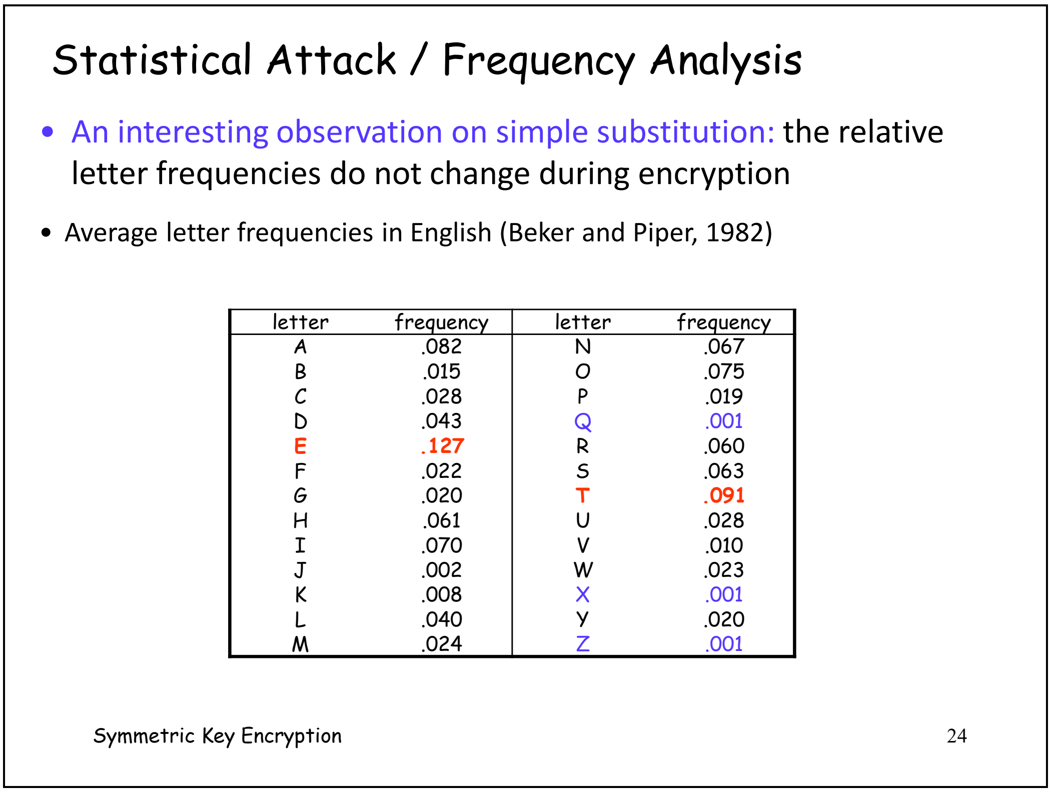


Letter frequencies

- In studies of the text of the Qur’an, scholars had noticed that some letters appear more frequently than others
- In English the letters *e* and *t* are used much more frequently than the letters *z* and *q*, and this fact can be used to decipher messages
- This process is called *frequency analysis*

Why is the statistical information of the plaintext important in substitution cipher?

Because there is a one to one match from plaintext character to ciphertext character, plaintext statistical information will therefore determine ciphertext statistical information.



Statistical Attack / Frequency Analysis

- An interesting observation on simple substitution: the relative letter frequencies do not change during encryption
- Average letter frequencies in English (Beker and Piper, 1982)

letter	frequency	letter	frequency
A	.082	N	.067
B	.015	O	.075
C	.028	P	.019
D	.043	Q	.001
E	.127	R	.060
F	.022	S	.063
G	.020	T	.091
H	.061	U	.028
I	.070	V	.010
J	.002	W	.023
K	.008	X	.001
L	.040	Y	.020
M	.024	Z	.001

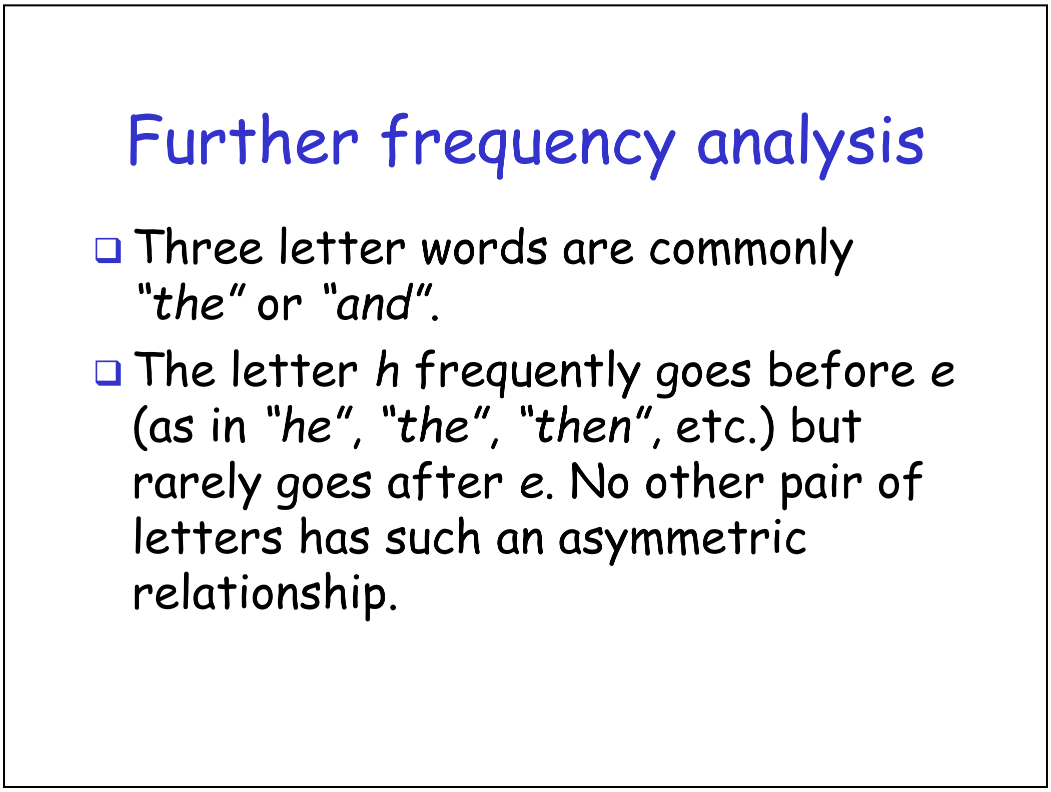
Substitution does not change the letter frequency of the plaintext

These are obviously average frequency – so you cannot expect these values to exactly appear in a ciphertext but it is useful to know what letters are more likely to occur. So for example, if you see a letter appearing lots then is probably not mapping to X or Z.

You should be able to solve basic short substitution problems using frequency analysis – so I suggest you study the most common letters and letter combination (you should not memorise every single frequency or all letter combination).

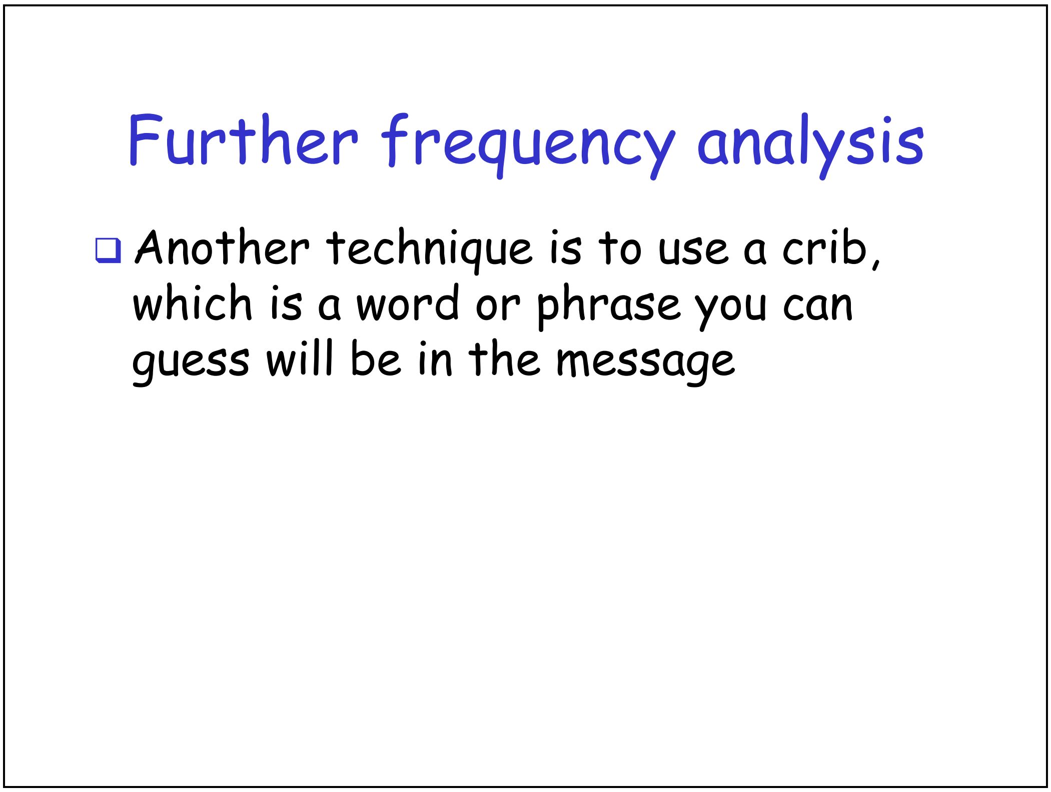
Further frequency analysis

- Pairs of letters in words are most likely to be: “*ss*”, “*ee*”, “*tt*”, “*ff*”, “*ll*”, “*mm*” or “*oo*”.
- *A* one letter word is either “*a*” or “*I*”.
- Two letter words are commonly: “*of*”, “*to*”, “*in*”, “*it*”, “*is*”, “*be*”, “*as*”, “*at*”, “*so*”, “*we*”, “*he*”, “*by*”, “*or*”, “*on*” or “*do*”, in that order.



Further frequency analysis

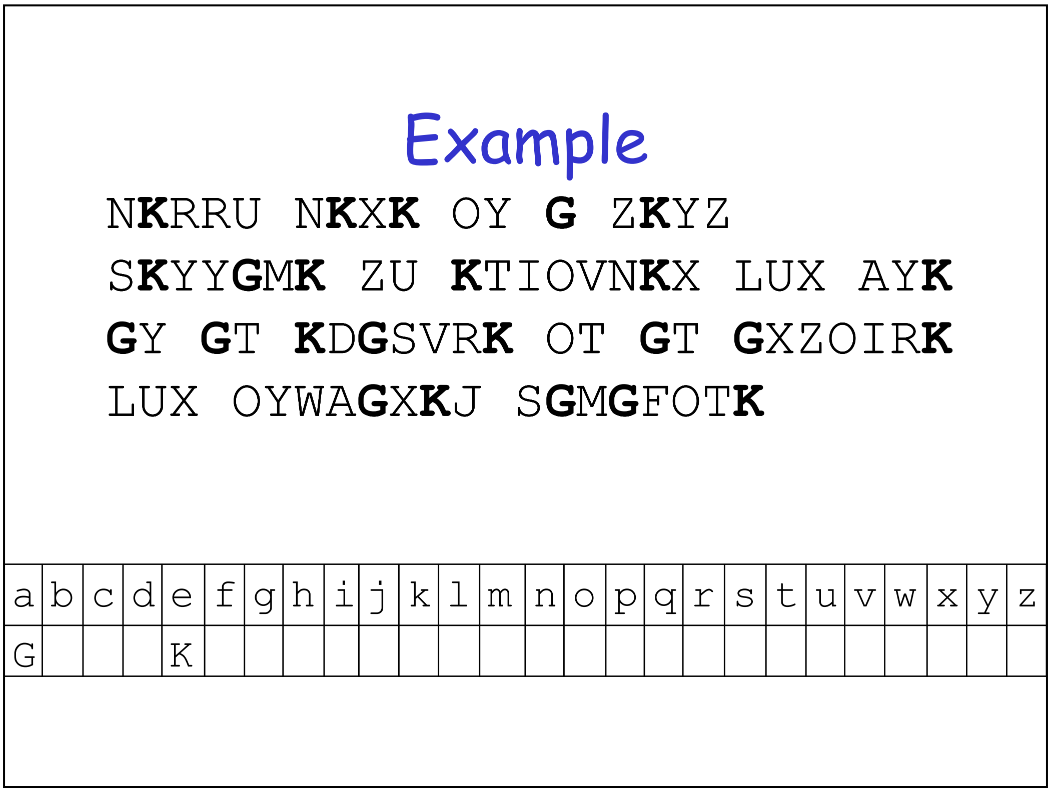
- Three letter words are commonly *"the"* or *"and"*.
- The letter *h* frequently goes before *e* (as in *"he"*, *"the"*, *"then"*, etc.) but rarely goes after *e*. No other pair of letters has such an asymmetric relationship.



Further frequency analysis

- Another technique is to use a crib, which is a word or phrase you can guess will be in the message

If you know there is a certain word and where it is you know the associated ciphertext (known plaintext approach) – this means you know the mapping for all the word’s letters.



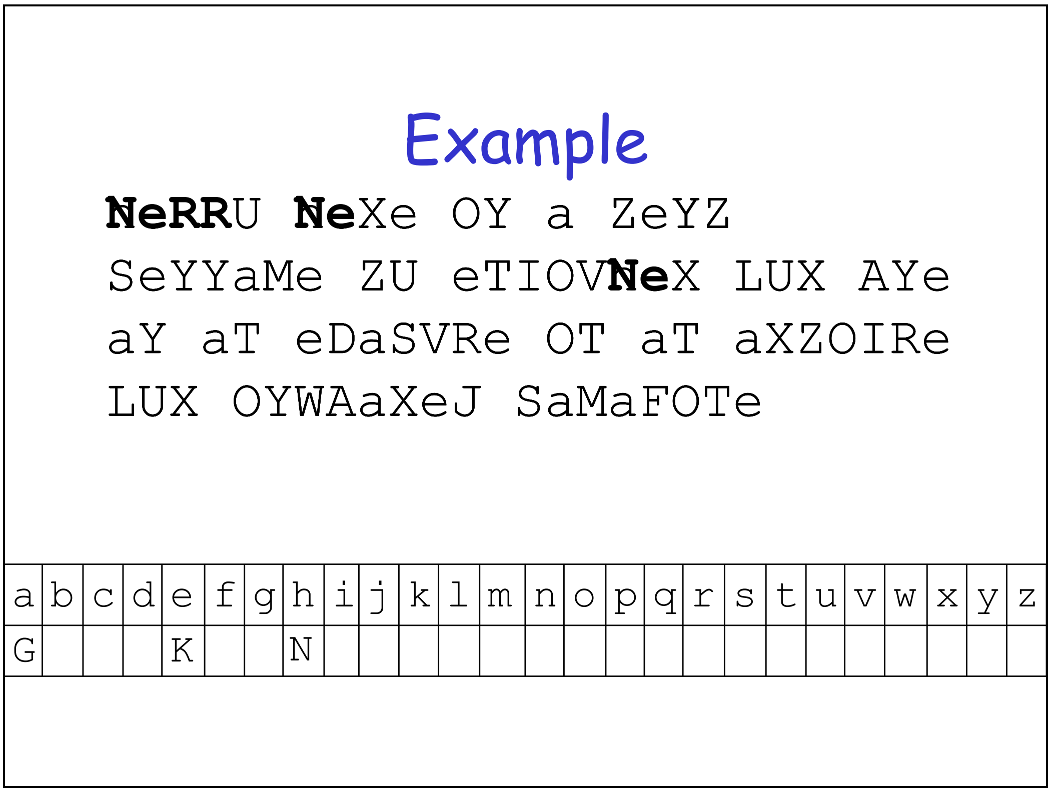
Example

N**K**RRU N**K**X**K** OY **G** Z**K**YZ
S**K**YY**G**M**K** ZU **K**TIOVN**K**X LUX AY**K**
GY **G**T **K**D**G**SVR**K** OT **G**T **G**XZOIR**K**
LUX OYWA**G**X**K**J S**G**M**G**FOT**K**

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
G				K																					

Any letter more than usual? K

Single letter? A?



Example

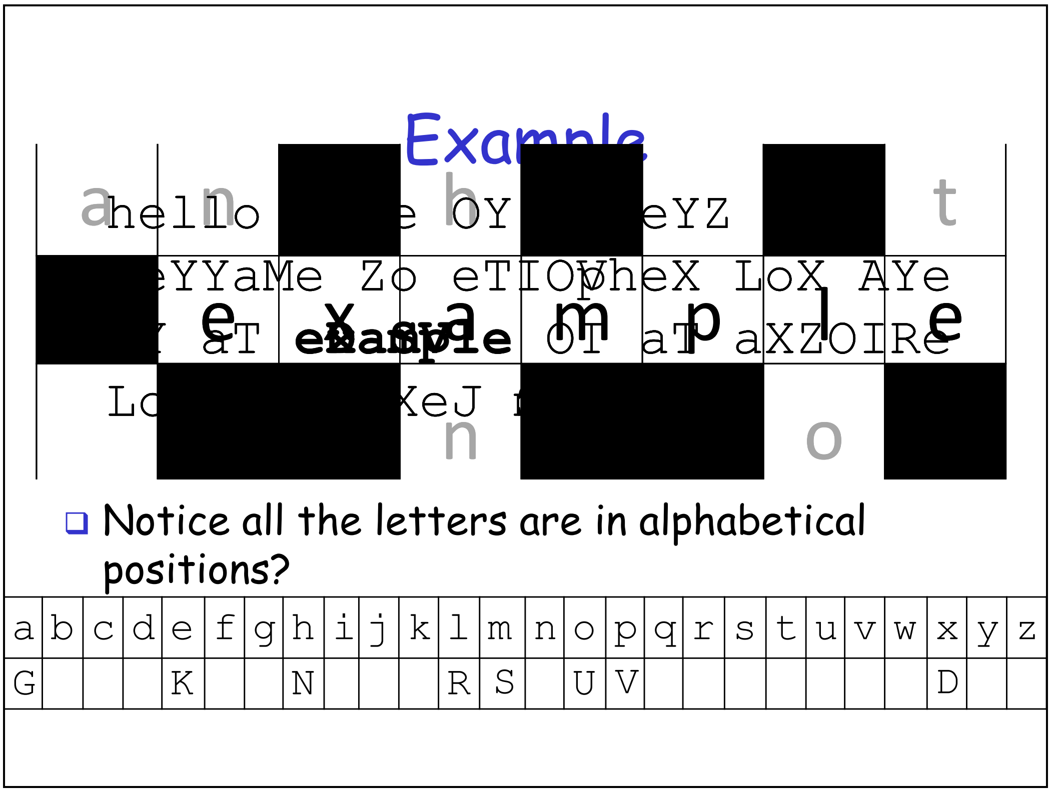
NeRRU **Ne**Xe OY a ZeYZ
SeYYaMe ZU eTIOV**Ne**X LUX AYe
aY aT eDaSVRe OT aT aXZOIRe
LUX OYWAaXeJ SaMaFOTe

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
G				K			N																		

Some letter always front of e? N >> h

First word

We can see that the first word has *"RR"* as a repeated character, so we might try to think of a word with a repeated letter in the middle. Also, noticing that the combination *"Ne"* keeps appearing in the message, you might guess this is *"he"*, which appears frequently in English. So perhaps *R* corresponds to *l* and *N* to *h*, so that the first word is *"hello"*?



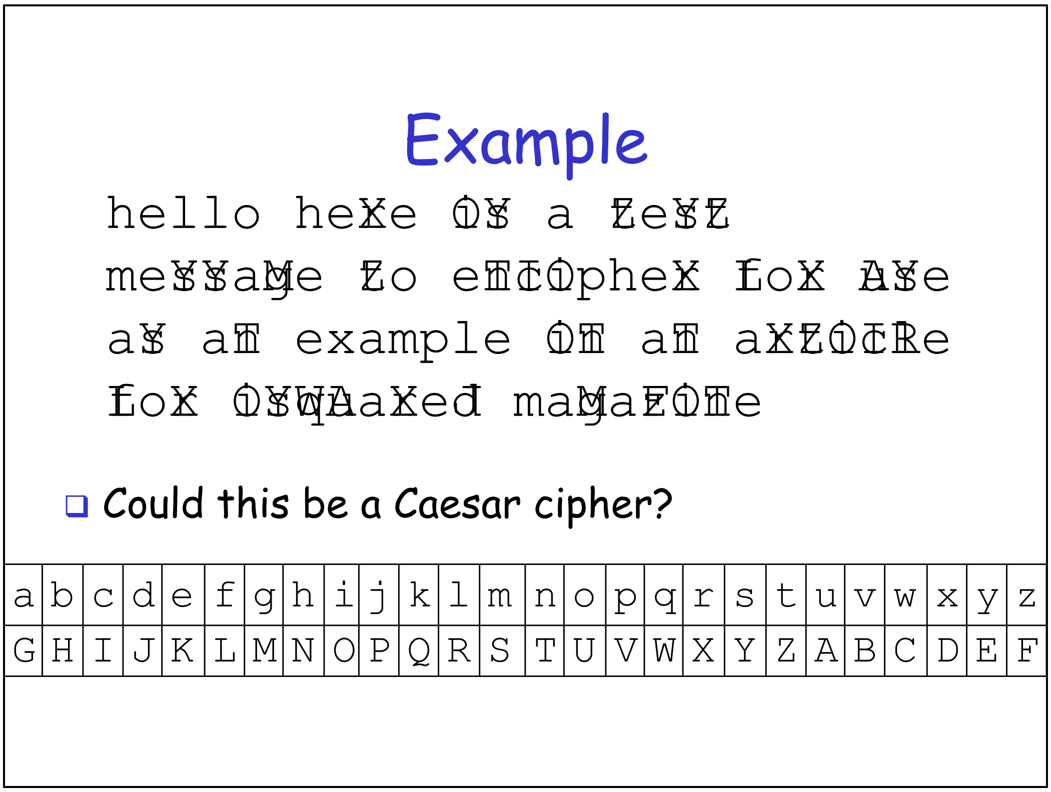
Example

- Notice all the letters are in alphabetical positions?

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
G				K			N				R	S		U	V								D		

We can see that the first word has *"RR"* as a repeated character, so we might try to think of a word with a repeated letter in the middle. Also, noticing that the combination *"Ne"* keeps appearing in the message, you might guess this is *"he"*, which appears frequently in English. So perhaps *R* corresponds to *l* and *N* to *h*, so that the first word is *"hello"*?

What do you notice down there? Alphabetical with a shift of 6



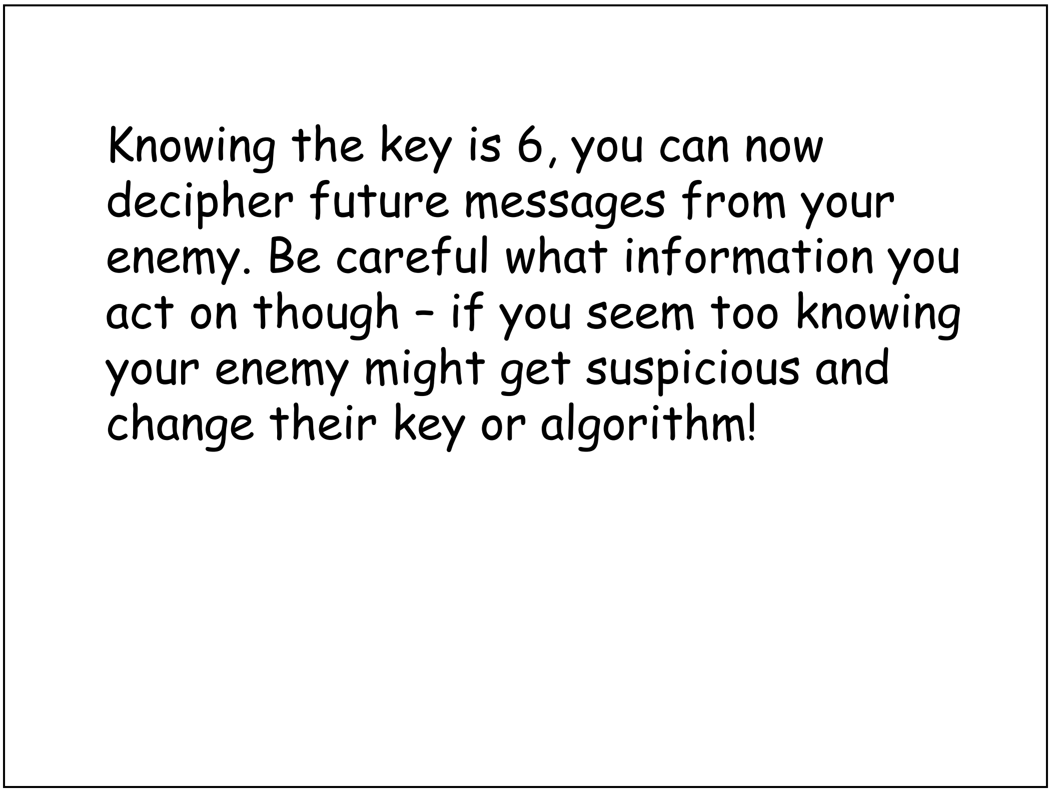
Example

```
hello here is a test
message to encipher for use
as an example in an article
for isquared magazine
```

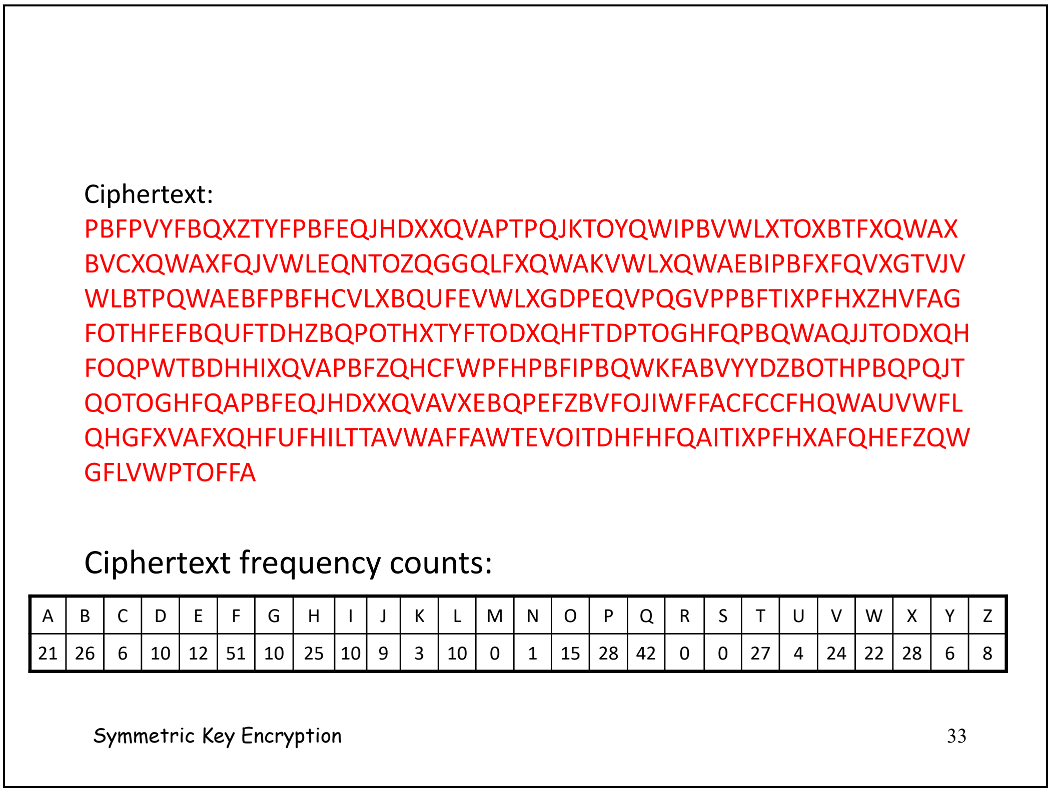
- ❑ Could this be a Caesar cipher?

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

Since we have obtained a sensible message, it is clear that we have found the correct cipher alphabet.



Knowing the key is 6, you can now decipher future messages from your enemy. Be careful what information you act on though – if you seem too knowing your enemy might get suspicious and change their key or algorithm!



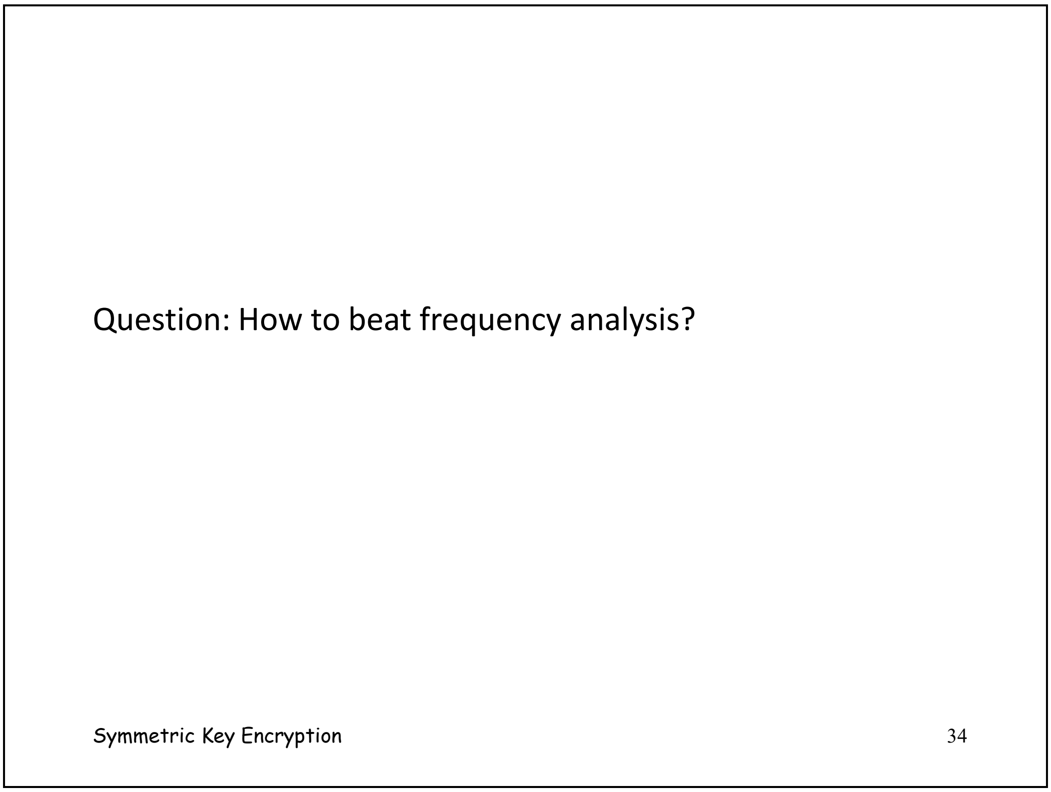
Ciphertext:

```
PBFPVYFBQXZTYFPBFEQJHDXXQVAPTPQJKTOYQWIPBVWLXTOXBTFXQWAX
BVCXQWAXFQJVWLEQNTOZQGGQLFXQWAKVWLXQWAEBIPBFXFQVXGTVJV
WLBTPQWAEBFPBFHCVLXBQUFEVWLXGDPEQVPQGVPPBFTIXPFHXZHVFAG
FOTHFEFBQUFTDHZBQPOTHXTYFTODXQHFTDPTOGHFQPBQWAQJJTODXQH
FOQPWTBDHHIXQVAPBFZQHCFWPFHPBFIPBQWKFABVYYDZBOTHPBQPQJT
QOTOGHFQAPBFEQJHDXXQVAVXEBQPEFZBVFOJIWFFACFCCFHQWAUVWFL
QHGFXVAFXQHFUFHILTTAVWAFFAWTEVOITDHFHFQAITIXPFHXAFQHEFZQW
GFLVWPTOFFA
```

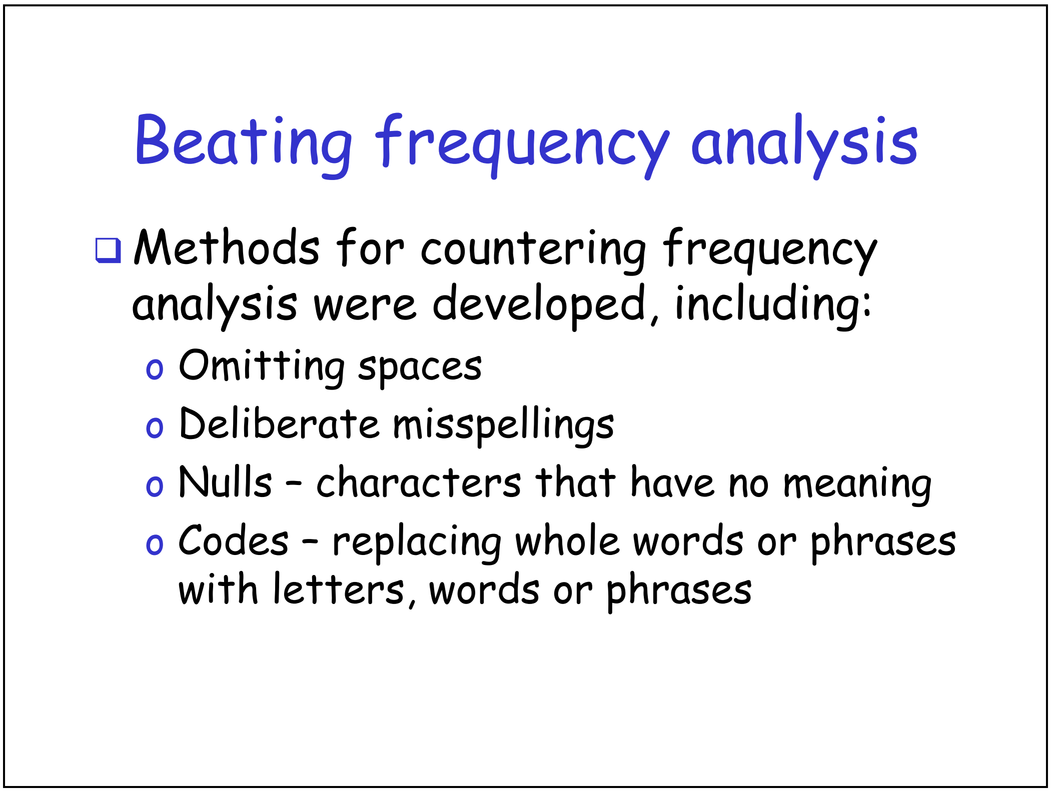
Ciphertext frequency counts:

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
21	26	6	10	12	51	10	25	10	9	3	10	0	1	15	28	42	0	0	27	4	24	22	28	6	8

Fun exercise – decipher the following plaintext….



Question: How to beat frequency analysis?



Beating frequency analysis

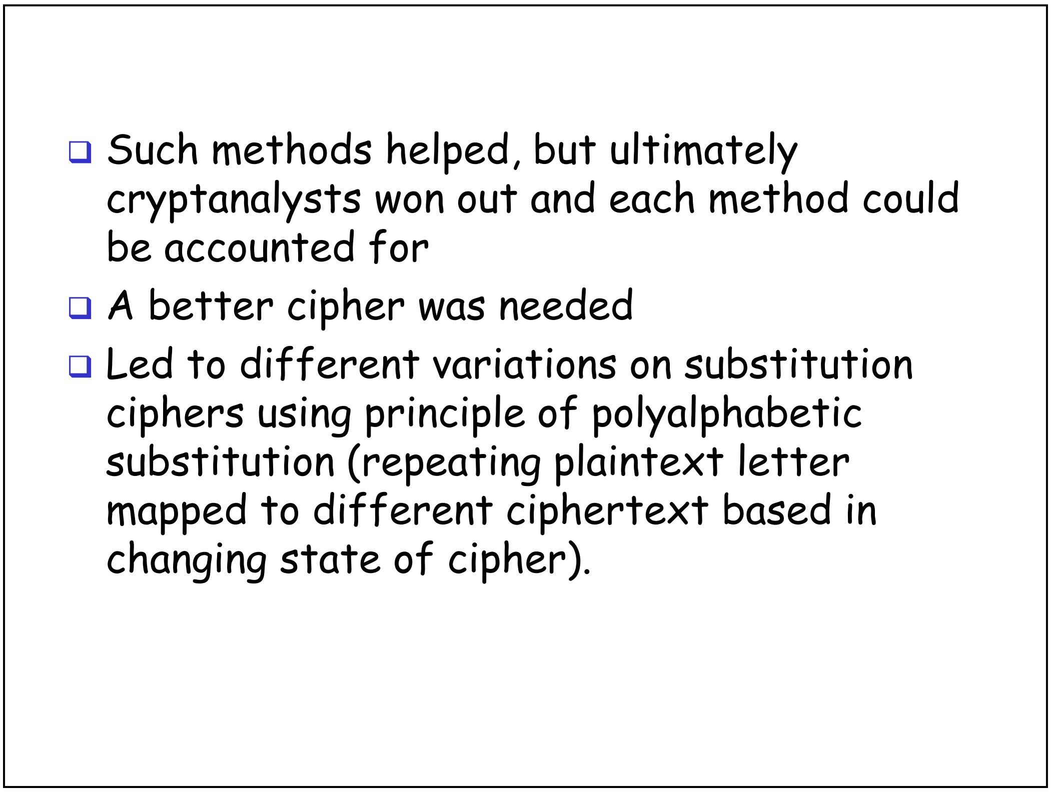
- Methods for countering frequency analysis were developed, including:
  - Omitting spaces
  - Deliberate misspellings
  - Nulls – characters that have no meaning
  - Codes – replacing whole words or phrases with letters, words or phrases

Slide 35 and 36 for references – just know that substitution ciphers are not statistically strong enough.

Remember the quotes about 'arms race' with codes….each times we make modification it makes things better for a while, then someone finds a way around it…

We cab out in nulls ('dummy' words), or first substitute and then substitute again….but at the end of the day simple substitution is not strong enough method against modern cryptanalysis and computing capability.

How to make better? Ideas?



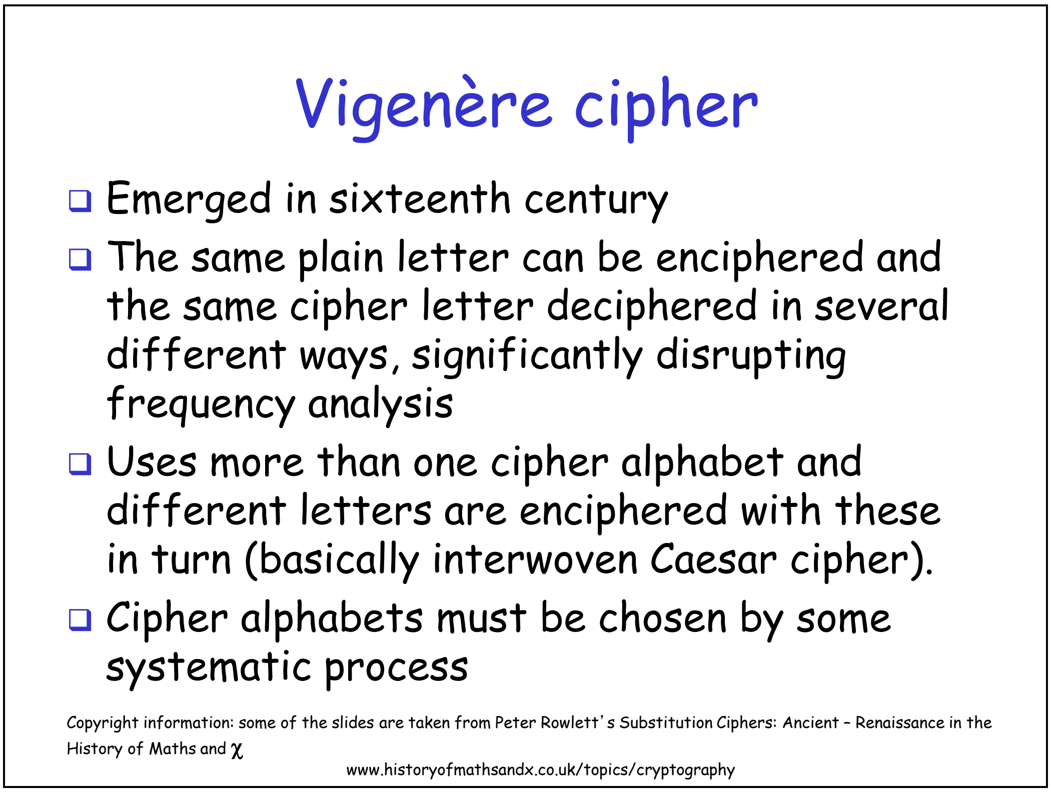
- Such methods helped, but ultimately cryptanalysts won out and each method could be accounted for
- A better cipher was needed
- Led to different variations on substitution ciphers using principle of polyalphabetic substitution (repeating plaintext letter mapped to different ciphertext based in changing state of cipher).

The next step was polyalphabetic ciphers – the substitution alphabet changes during the message.

Not necessarily in terms of principle (some substitution ciphers were historically quite tricky to try and figure out – e.g. read about the Enigma machine).

Was quite good – operational issues led to cryptanalysis (repeat use of keys, often sending the same message, start message same way –'To:', 'Nothing to report'. Having a user manual with some plaintext/ciphertext examples.

Vigenere Cipher is only for reference.



Vigenère cipher

- Emerged in sixteenth century
- The same plain letter can be enciphered and the same cipher letter deciphered in several different ways, significantly disrupting frequency analysis
- Uses more than one cipher alphabet and different letters are enciphered with these in turn (basically interwoven Caesar cipher).
- Cipher alphabets must be chosen by some systematic process

Copyright information: some of the slides are taken from Peter Rowlett's Substitution Ciphers: Ancient – Renaissance in the History of Maths and χ

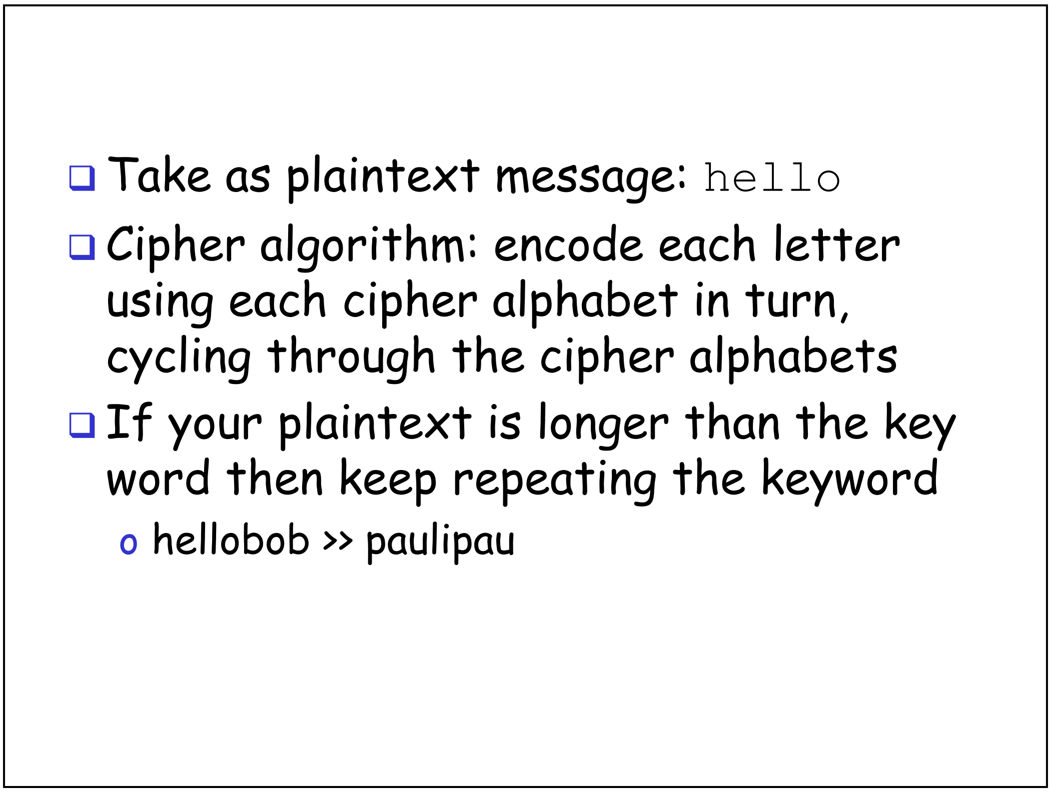
www.historyofmathsandx.co.uk/topics/cryptography

This is just for extra information, no need to study Vigenere

Example

- First, choose a word for your key
- Key: Choose "pauli"
- The Caesar cipher alphabets beginning with the letters of the keyword are then produced:

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



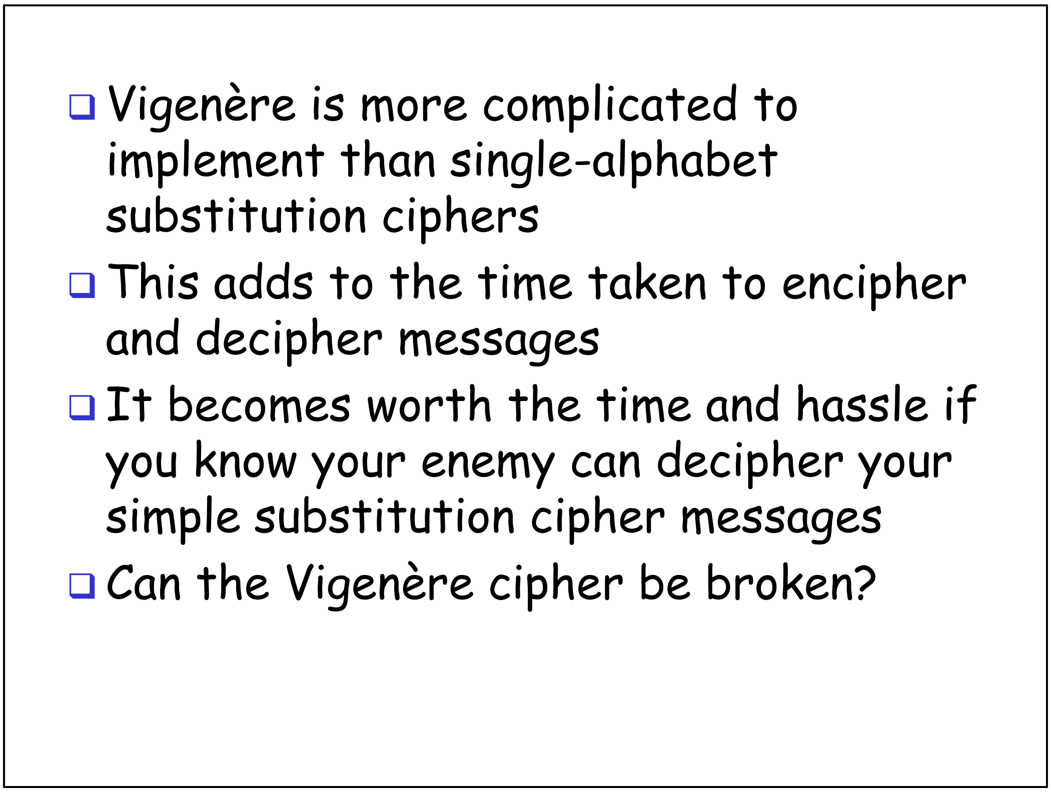
- Take as plaintext message: `hello`
- Cipher algorithm: encode each letter using each cipher alphabet in turn, cycling through the cipher alphabets
- If your plaintext is longer than the key word then keep repeating the keyword
  - hellobob >> paulipau

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

- *“h”* is enciphered using the *“P”* alphabet, giving *“W”*
- *“e”* is enciphered using the *“A”* alphabet, giving *“E”*
- *“l”* is enciphered using the *“U”* alphabet, giving *“F”*
- *“l”* is enciphered using the *“L”* alphabet, giving *“W”*
- *“o”* is enciphered using the *“I”* alphabet, giving *“W”*

- ciphertext message: `WEFWW`

- `hello` to ciphertext message: `WEFWW`
- Notice that, crucially, we have
  - (a) enciphered the two letters "*l*" to give different cipher letters "*F*" and "*W*";
  - and, (b) enciphered different plaintext letters "*h*", "*l*" and "*o*" to give the same ciphertext letter "*W*".
- Through use of multiple alphabets, the chart of letter frequencies is distorted, providing strong resistance to frequency analysis



- Vigenère is more complicated to implement than single-alphabet substitution ciphers
- This adds to the time taken to encipher and decipher messages
- It becomes worth the time and hassle if you know your enemy can decipher your simple substitution cipher messages
- Can the Vigenère cipher be broken?

Can we use frequency analysis? No since there are no discernible substation mapping for each plaintext word. It is a one to many mapping.

- Vigenère was for 300 years considered undecipherable (1553-1863)
- Primary weakness is that if the length of the codeword is known we can break each of the individual Caesar ciphers independently
- 1863 Friedrich Kasiski published his Kasiski Examination method
  - Estimates keyword length without plaintext knowledge or the keyword needing to be a recognisable word

Source: https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher

- Kasiski notices that repeated words are by chance encrypted using same key letters
- For keyword ABCD:

Key: ABCDABCDABCDABCDABCDABCDABCD
Plaintext: ***CRYPTO***ISSHORTFOR***CRYPTO***GRAPHY
Ciphertext: ***CSASTP***KVSIQUTGQU***CSASTP***IUAQJB

- Repetition distance is 16 - key size is 16,8,4,2,1
- If you find multiple repetitions then easier

Ciphertext: ***VHVS***SP***QUCE***MRVBVBBB***VHVS***URQGIBDUGRNICJ***QUCE***RVUAXSSR

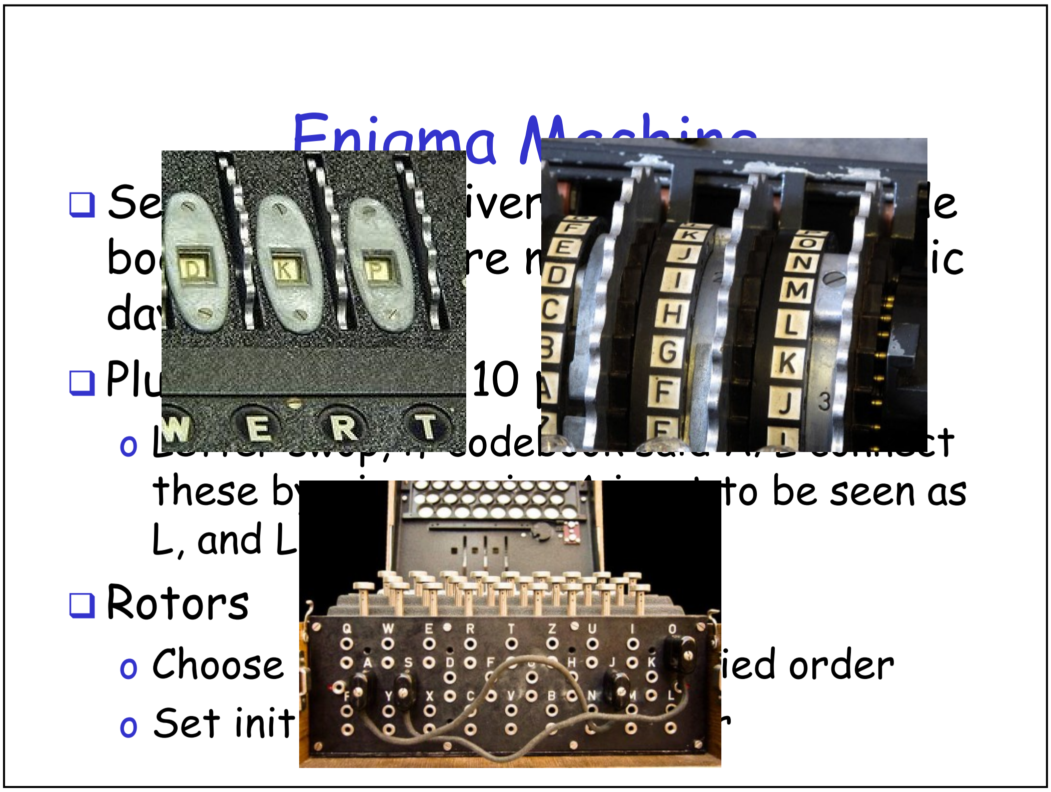
- VHVS is 18 (18,9,6,3,2,1) and QUCE is 30 (30,15,10,6,5,3,2,1)
- Key size 6,3,2,1 (most probably 6)

- Usec II
  - Las
- Polyc
  - To have ma state as the

Credit: https://brilliant.org/wiki/enigma-machine/

This is just for extra information, no need to study Ënigma

Configured rotors, and plug board on front. Pressing keys would light a lamp showing the corresponding plaintext (or ciphertext when decrypting).

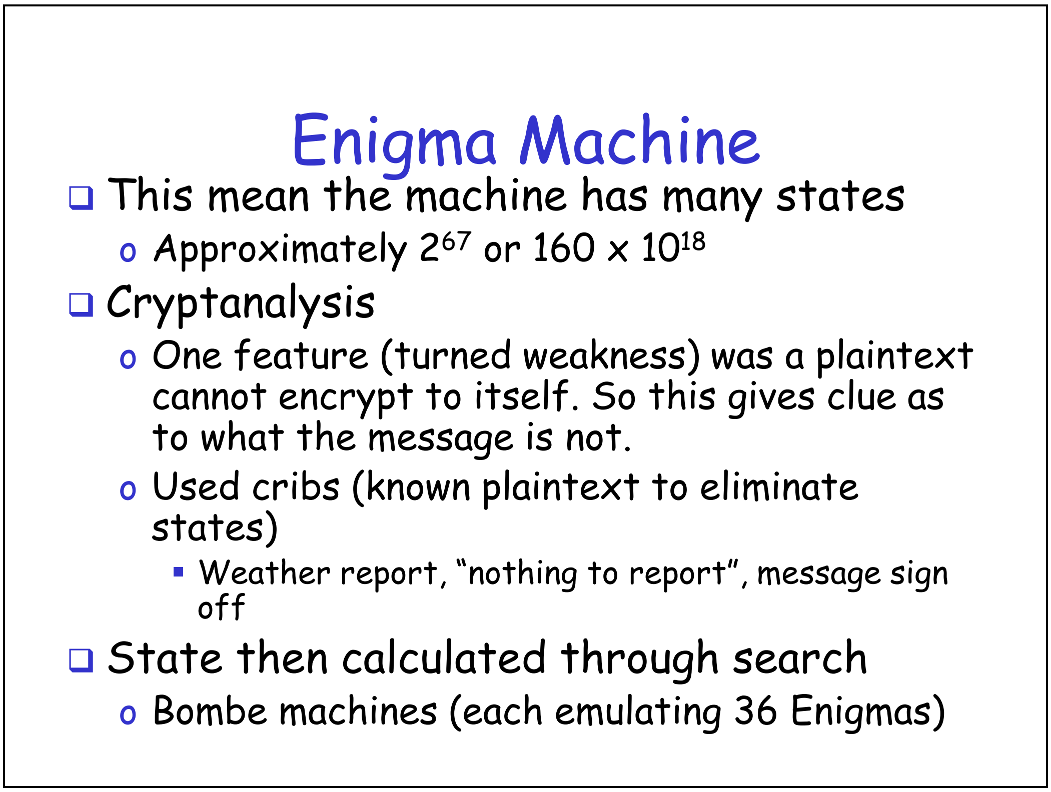


Enigma Machine

- Se iver le bo re n ic da
- Plu 10
  - L swap, codebook said A, L connect these by A to be seen as L, and L
- Rotors
  - Choose ied order
  - Set init r

This is just for extra information, no need to study Ënigma

Configured rotors, and plug board on front. Pressing keys would light a lamp showing the corresponding plaintext (or ciphertext when decrypting).



Enigma Machine

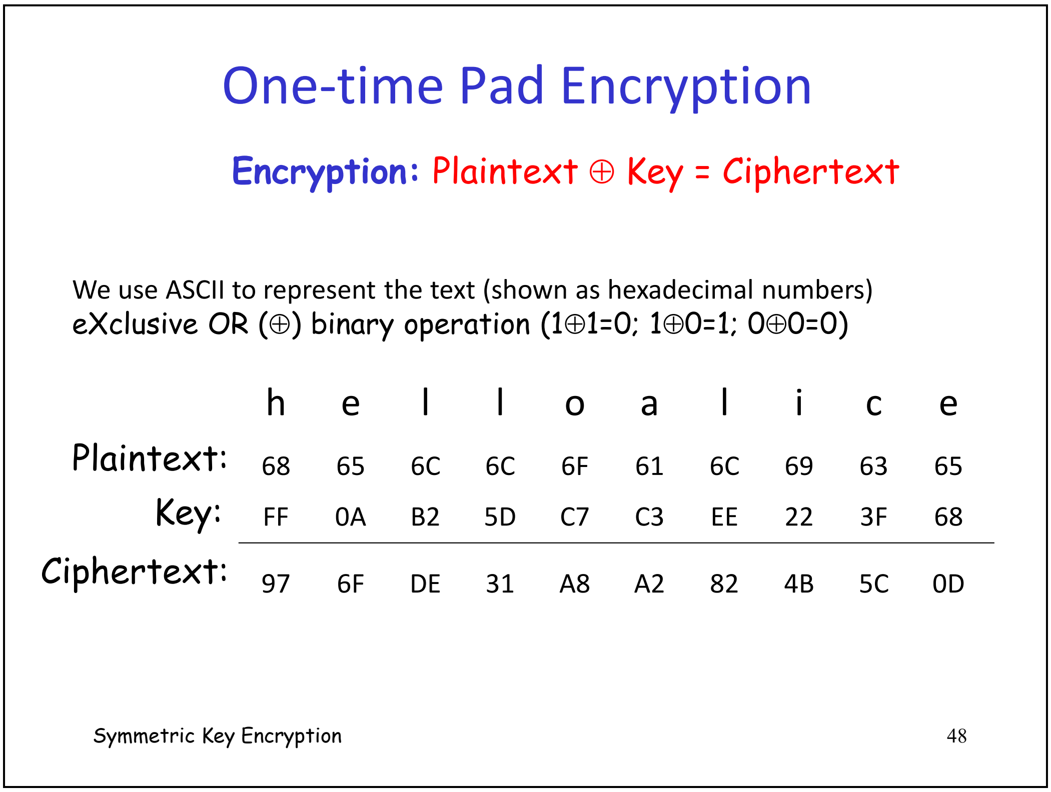
- ❑ This mean the machine has many states
  - o Approximately 2^{67} or 160 x 10^{18}
- ❑ Cryptanalysis
  - o One feature (turned weakness) was a plaintext cannot encrypt to itself. So this gives clue as to what the message is not.
  - o Used cribs (known plaintext to eliminate states)
    - ▪ Weather report, "nothing to report", message sign off
- ❑ State then calculated through search
  - o Bombe machines (each emulating 36 Enigmas)

Usually message started with a weather report, and message was signed Heil Hitler. Commonly message was "*Keine besonderen Ereignisse" which means nothing to report.*

Move one from substitution - In the end substitution it can also only encrypt text characters…

Efforts on Enigma drove lots of other computer science inventions (first effort at a computer) and involved famous mathematicians/scientists. (Alan Turing).

The Germans also used an early stream cipher called the Lorenz cipher (this led to development of early computer – the Colossus to aid in cryptanalysis)



One-time Pad Encryption

Encryption: Plaintext ⊕ Key = Ciphertext

We use ASCII to represent the text (shown as hexadecimal numbers)
eXclusive OR (⊕) binary operation (1⊕1=0; 1⊕0=1; 0⊕0=0)

	h	e	l	l	o	a	l	i	c	e
Plaintext:	68	65	6C	6C	6F	61	6C	69	63	65
Key:	FF	0A	B2	5D	C7	C3	EE	22	3F	68
Ciphertext:	97	6F	DE	31	A8	A2	82	4B	5C	0D

Symmetric Key Encryption 48

Times were changing - arguably due to changes in communication methods binary data rather than alphabet was considered. Early telegraph/radio used Baudot code (letters presented by 5 bits).

The idea for the one time pad was first mentioned by Frank Miller, but originally through to be invented in 1917 by Gilbert Vernam (called the Vernam cipher, a patent was issued for the XOR operation) for teleprinters.

In 1940s, Claude Shannon proved the significant of the one time pad - he studied security from information theory perspective and introduced the concept of perfect secrecy.

Go back to substitution – if we sends same message twice, and we see the same ciphertext – do we learn something?

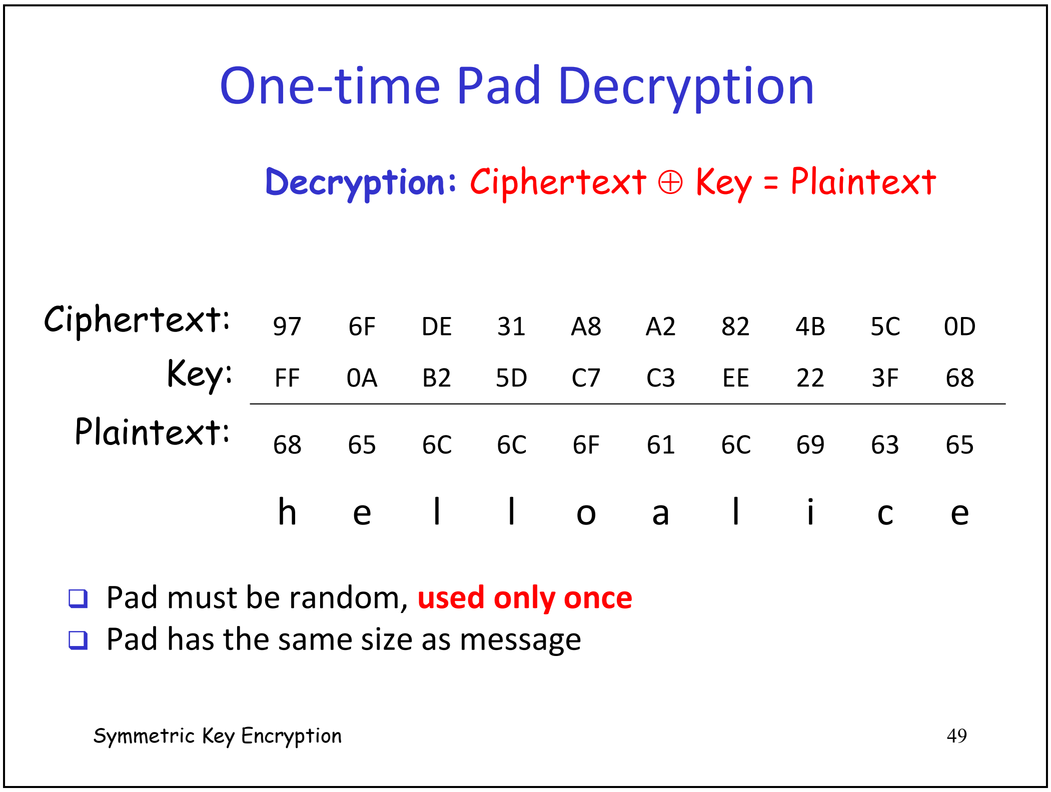
XOR is special binary function, if C = A XOR B then C XOR A = B, or C XOR B = A

True one time pad the entire message is XORed to a random binary value (string) of equal length.

Right so what does the perfect scheme look like?

You need to understand that perfect secrecy means that the attacker has now idea about any of the plaintext – this does not mean he does not know the plaintext, this means he knows *nothing*. For example, if we use a good cipher and we encrypt the same plaintext twice and it gives the same ciphertext then even the plaintext is not known we lose perfect secrecy as we know something – the message is the same.

Here the key is random value that has the same length as the plaintext.



One-time Pad Decryption

Decryption: Ciphertext ⊕ Key = Plaintext

Ciphertext:	97	6F	DE	31	A8	A2	82	4B	5C	0D
Key:	FF	0A	B2	5D	C7	C3	EE	22	3F	68
Plaintext:	68	65	6C	6C	6F	61	6C	69	63	65
	h	e	l	l	o	a	l	i	c	e

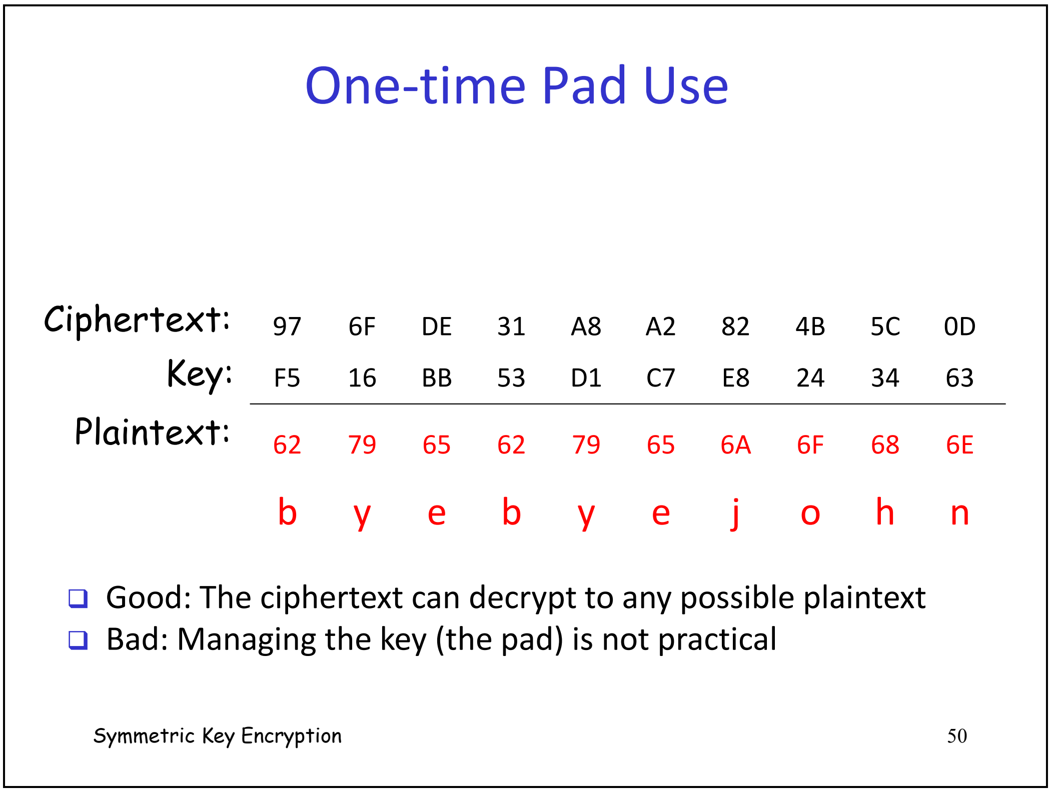
- ❑ Pad must be random, **used only once**
- ❑ Pad has the same size as message

Symmetric Key Encryption 49

A one time pad is very simple – yet it offers perfect secrecy – theoretically secure. Why?

If given the ciphertext there is no way we can know for sure the message. Any ciphertext can decrypt to any possible plaintext.

Any encrypted message can then decrypt to any other message.



One-time Pad Use

Ciphertext:	97	6F	DE	31	A8	A2	82	4B	5C	0D
Key:	F5	16	BB	53	D1	C7	E8	24	34	63
Plaintext:	62	79	65	62	79	65	6A	6F	68	6E
	b	y	e	b	y	e	j	o	h	n

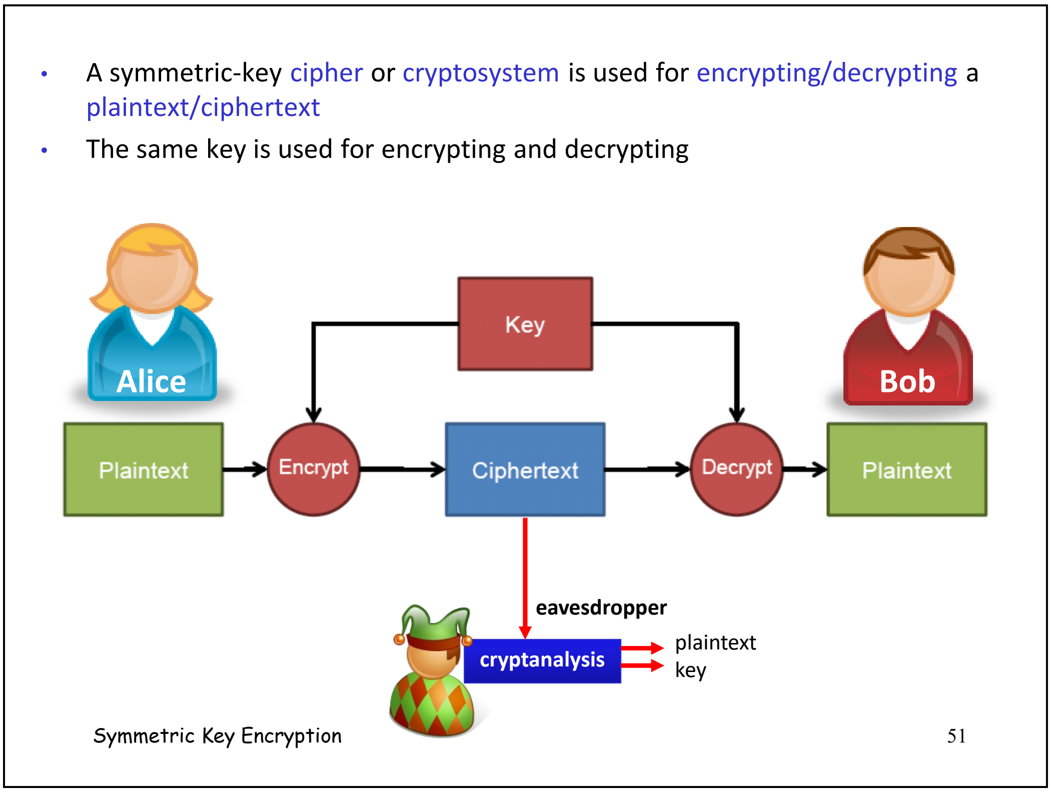
- ❑ Good: The ciphertext can decrypt to any possible plaintext
- ❑ Bad: Managing the key (the pad) is not practical

Why do we not use it? Key management…

1. How will we keep giving the pad to the recipient? Each time we send a message we need to send a new random 'pad' with it.

2. We need a key that is the same length as the message.

3. We also need a new key each and every time.



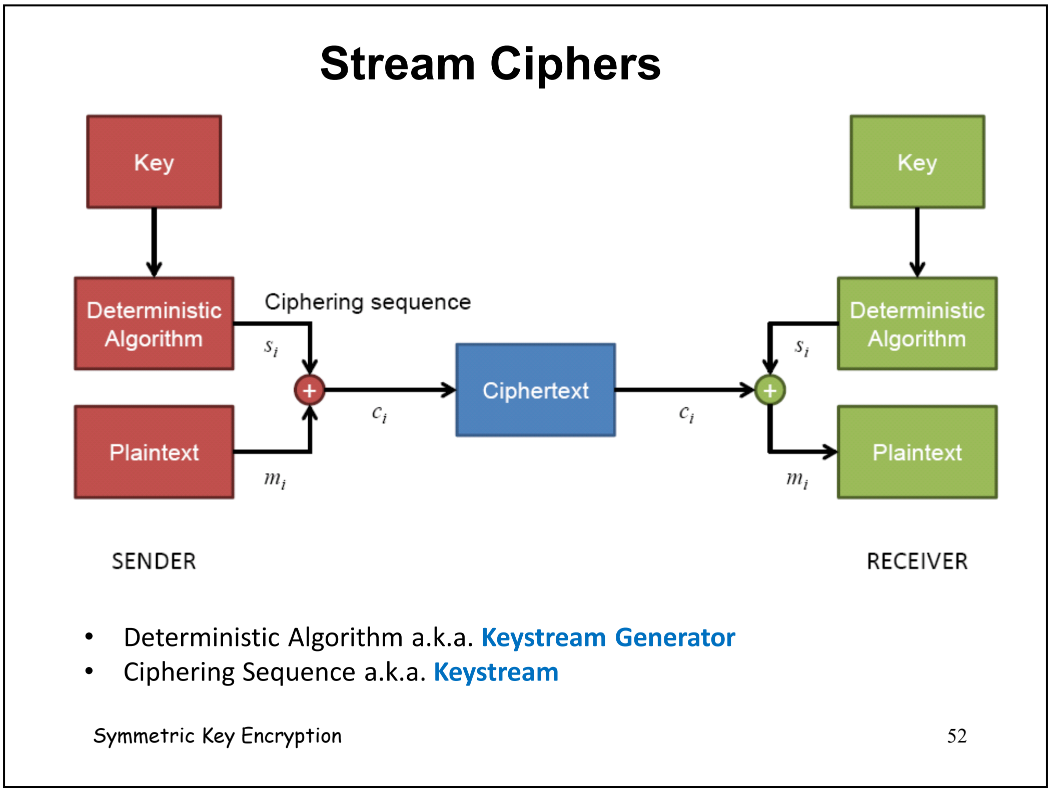
- A symmetric-key cipher or cryptosystem is used for encrypting/decrypting a plaintext/ciphertext
- The same key is used for encrypting and decrypting

Remember Alice and Bob

Same key.

Algorithm for encrypt and decrypt.

What two main types: Stream and block? Can you remember the difference?



Stream Ciphers

Key

Deterministic Algorithm

Ciphering sequence

s_i

Plaintext

m_i

c_i

Ciphertext

c_i

Key

Deterministic Algorithm

s_i

Plaintext

m_i

SENDER

RECEIVER

- Deterministic Algorithm a.k.a. **Keystream Generator**
- Ciphering Sequence a.k.a. **Keystream**

You need to study this slide.

XOR is special binary function, if C = A XOR B then C XOR A = B, or C XOR B = A

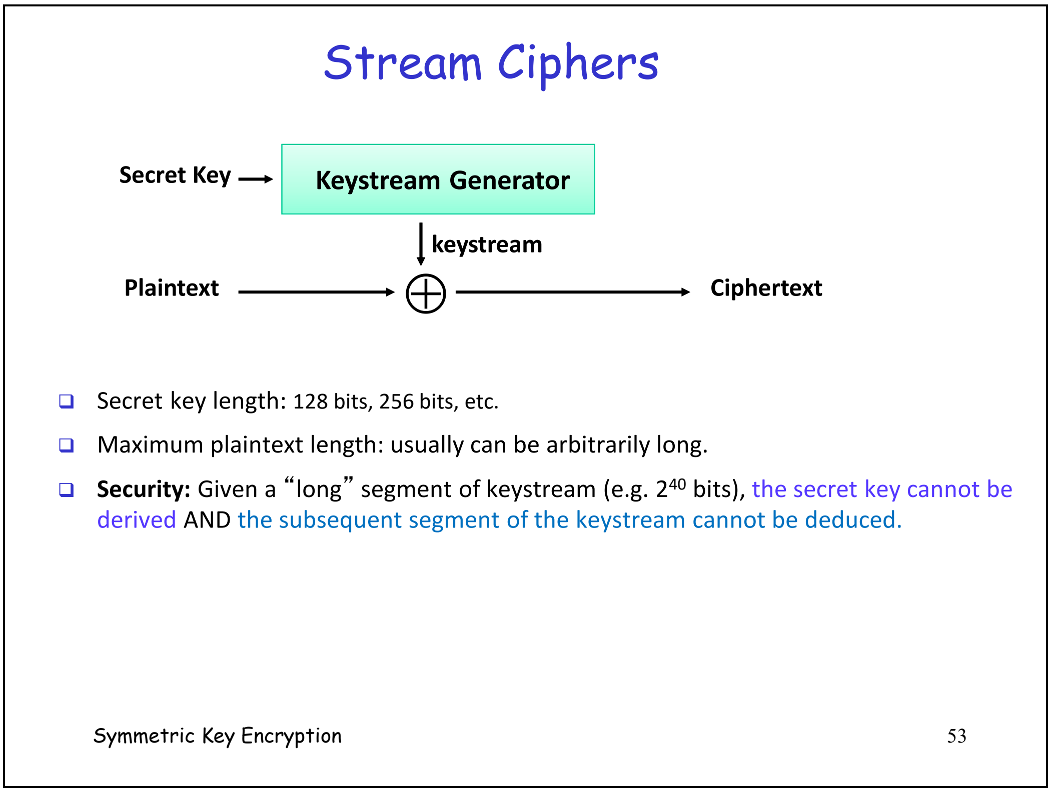
Remember our basic symmetric encryption model – in this case we have a shared key, and the 'encryption' and 'decryption' function is actually the same mathematical function (each side calculates the same keystream and does the XOR).

The key is fed into a deterministic algorithm (a function that has no randomness in future states – reproducable and predictable). This only holds for someone with the key – if you know the key the algorithm is deterministic.

If someone just observes the output, and he does not have the key, it appears like random output.

Later we will see that other schemes can have different encrypt and decrypt functions, i.e. encrypt and then reverse the encryption operation

What is important here? Sender and receiver synchronisation.



Stream Ciphers

Secret Key → Keystream Generator

keystream

Plaintext → ⊕ → Ciphertext

- ❑ Secret key length: 128 bits, 256 bits, etc.
- ❑ Maximum plaintext length: usually can be arbitrarily long.
- ❑ **Security:** Given a "long" segment of keystream (e.g. 2^{40} bits), the secret key cannot be derived AND the subsequent segment of the keystream cannot be deduced.

Symmetric Key Encryption 53

Study this slide.

We consider a stream cipher to be secure if – the attackers gets a long sequence of the keystream (not the ciphertext) then he cannot 1) determine the key 2) calculate future keystream.

Stream ciphers are considered to be fast, and you can make as much cipherstream as you have plaintext.

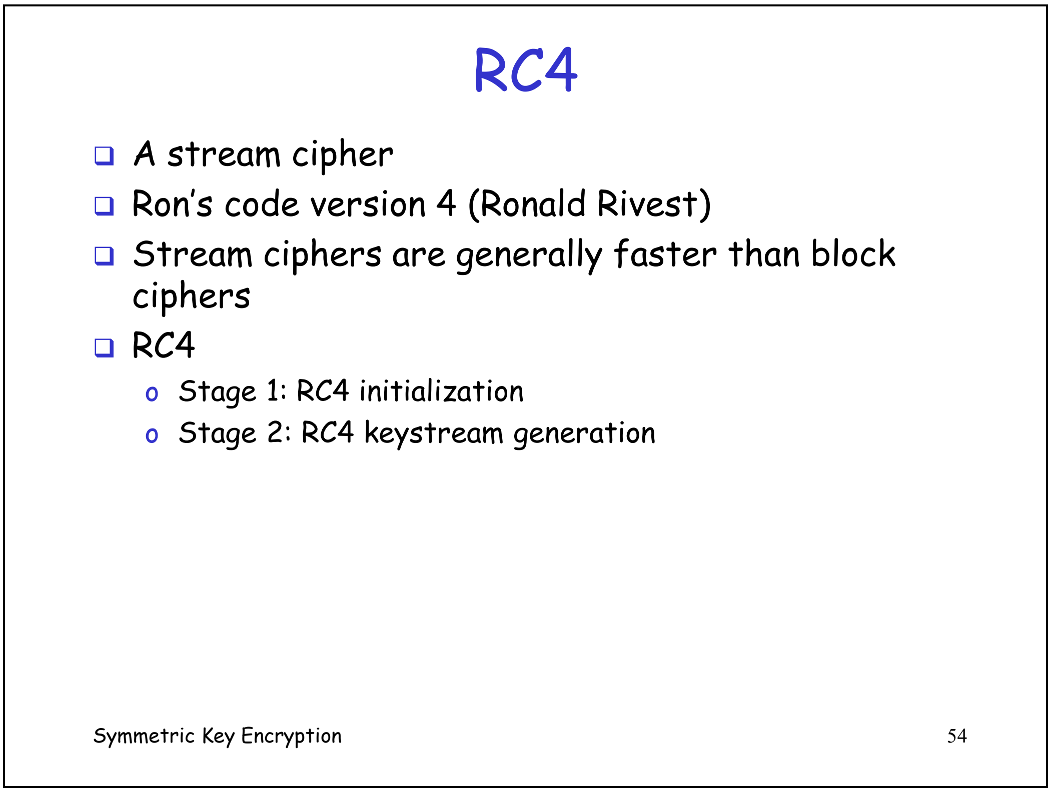
What about disadvantage ? What happens if we know the plaintext? We can XOR the plaintext with the ciphertext – then we have the keystream. Then we XOR it own message.

As we combine the plaintext and keystream using XOR – we should really also then provide integrity for the data.

Consider if we send a value 0010 encrypted with keystream 1010 (C= 0010 XOR 1010 = 1000) Without knowing the plaintext (maybe just the format) an attacker can modify the message by flipping a bit C'=0000, then means P = 0000 XOR 1010 = 1010 and the value is bigger.

Alternative if the attacker knows the plaintext he can go KS= C XOR P and then create C' = KS XOR M (his message). The reciever will then decrypt to M

If we are to use stream ciphers in practice we need additional security services – especially with regards to integrity/data origin authentication.



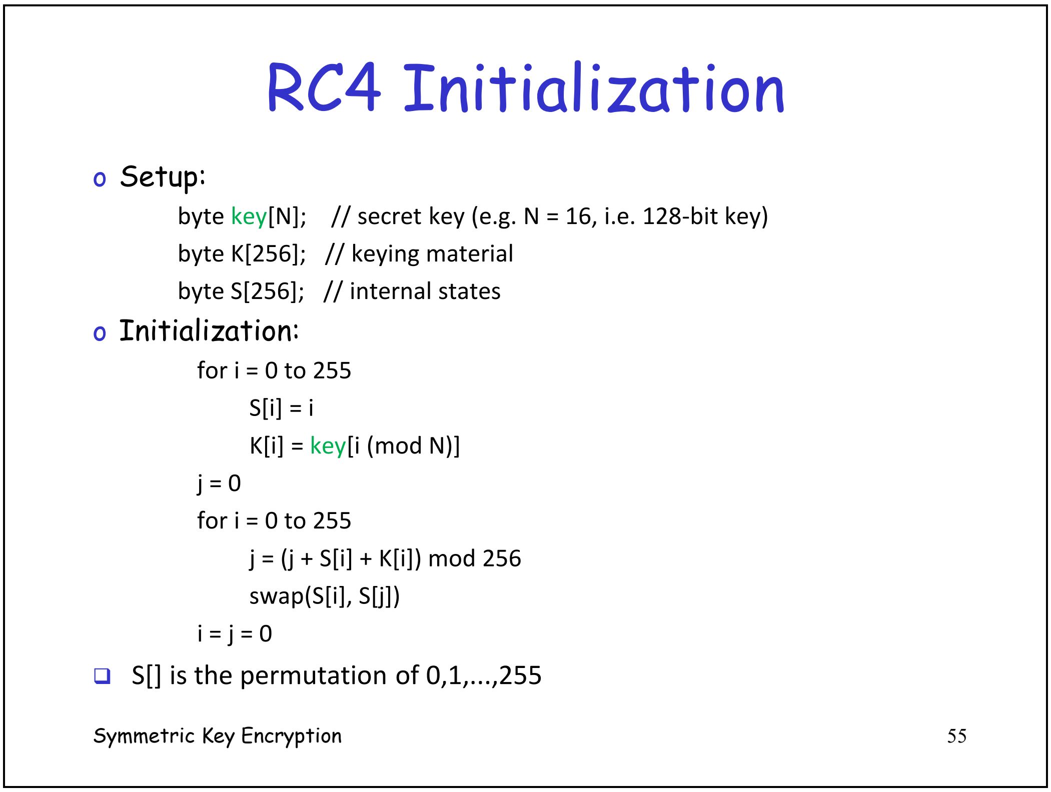
RC4

- A stream cipher
- Ron’s code version 4 (Ronald Rivest)
- Stream ciphers are generally faster than block ciphers
- RC4
  - Stage 1: RC4 initialization
  - Stage 2: RC4 keystream generation

You need to know that RC4 is example of a stream cipher but no need to study the inner details/working.

RC4 is a good (or famous anyway) example of a stream cipher. From late 1980s, not secure anymore.

In practice we rarely use special/dedicated stream ciphers – if we would like to generate keystream it is seen as more secure to generate keystream using block ciphers in a mode of operation that gives us keystream (Like OFB of CFB output feedback or cipher feedback mode). Stallings has more details about these modes.



RC4 Initialization

- Setup:

```
byte key[N];    // secret key (e.g. N = 16, i.e. 128-bit key)
byte K[256];  // keying material
byte S[256];  // internal states
```

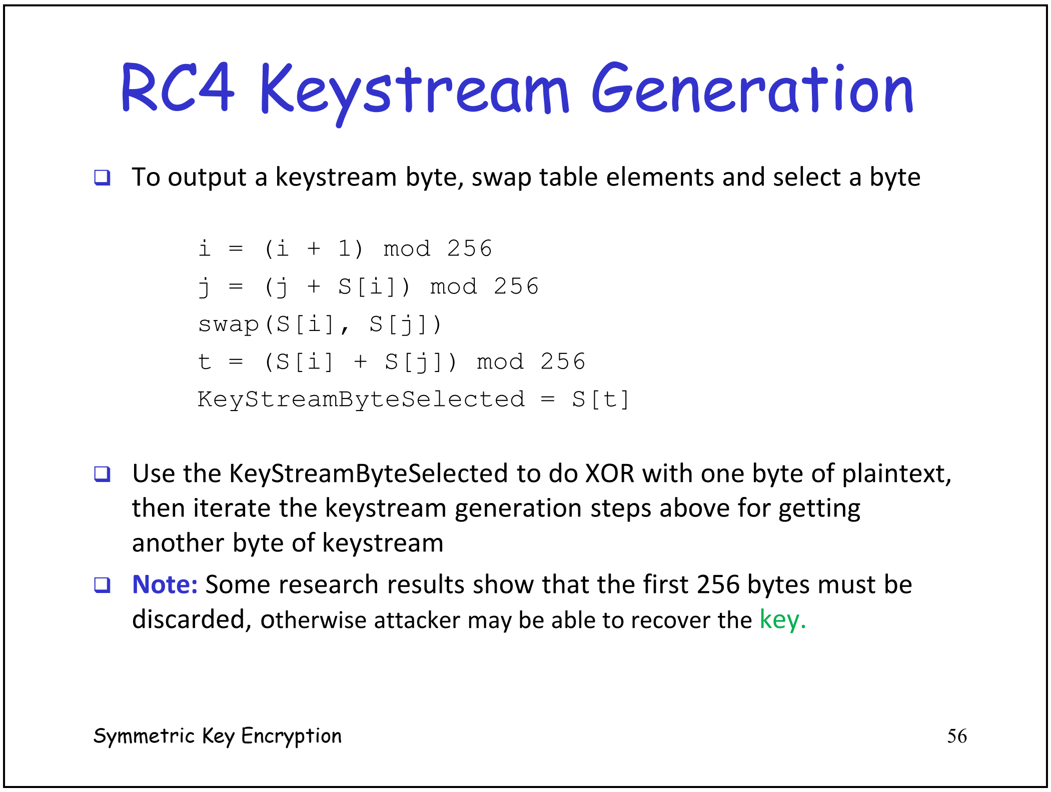
- Initialization:

```
for i = 0 to 255
    S[i] = i
    K[i] = key[i (mod N)]
j = 0
for i = 0 to 255
    j = (j + S[i] + K[i]) mod 256
    swap(S[i], S[j])
i = j = 0
```

- S[] is the permutation of 0,1,...,255

We are interested in the fact that after initialisation the internal state is now a permutation dependent on the key

Can see that in keeping with earlier ideas on permutation we are actually just dealing with bit level permutation/substitution.



RC4 Keystream Generation

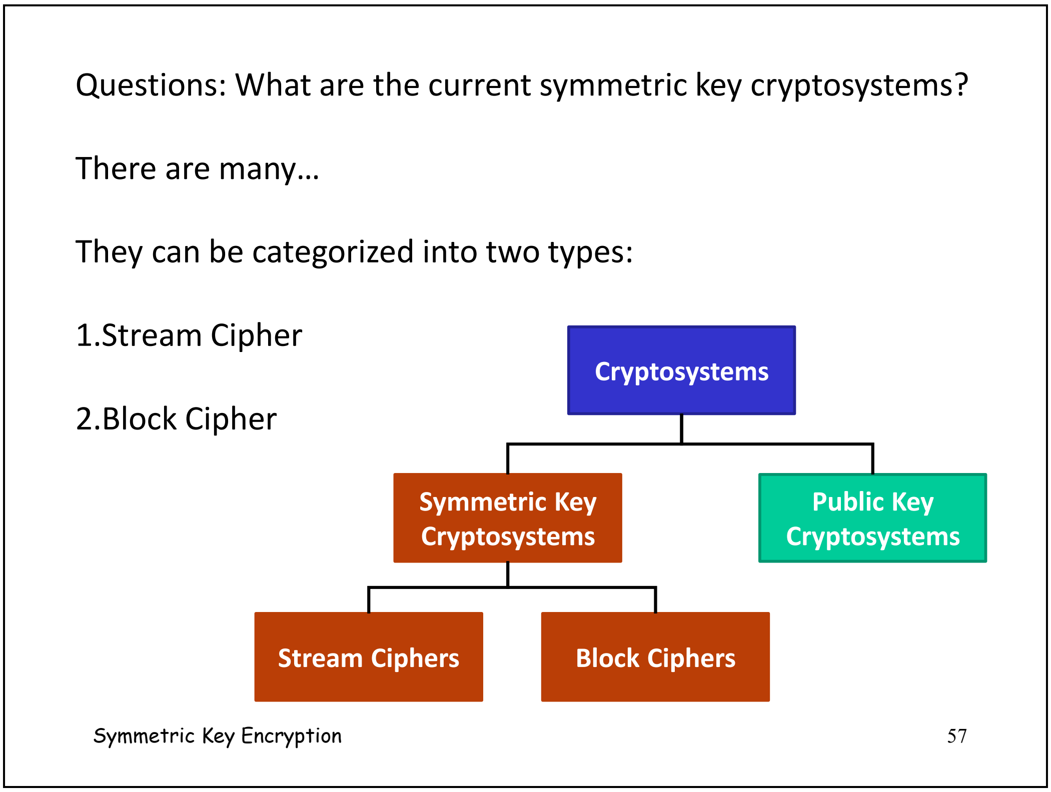
- ❑ To output a keystream byte, swap table elements and select a byte

```
i = (i + 1) mod 256
j = (j + S[i]) mod 256
swap(S[i], S[j])
t = (S[i] + S[j]) mod 256
KeyStreamByteSelected = S[t]
```

- ❑ Use the KeyStreamByteSelected to do XOR with one byte of plaintext, then iterate the keystream generation steps above for getting another byte of keystream
- ❑ **Note:** Some research results show that the first 256 bytes must be discarded, otherwise attacker may be able to recover the key.

This illustrates the advantage of a stream cipher – if we need a byte of keystream then we just iterate once more…

We can make as much keystream as we have plaintext….need to encrypt another byte, make another byte of keystream.



Questions: What are the current symmetric key cryptosystems?

There are many…

They can be categorized into two types:

1.Stream Cipher

2.Block Cipher

Cryptosystems

Symmetric Key Cryptosystems

Public Key Cryptosystems

Stream Ciphers

Block Ciphers

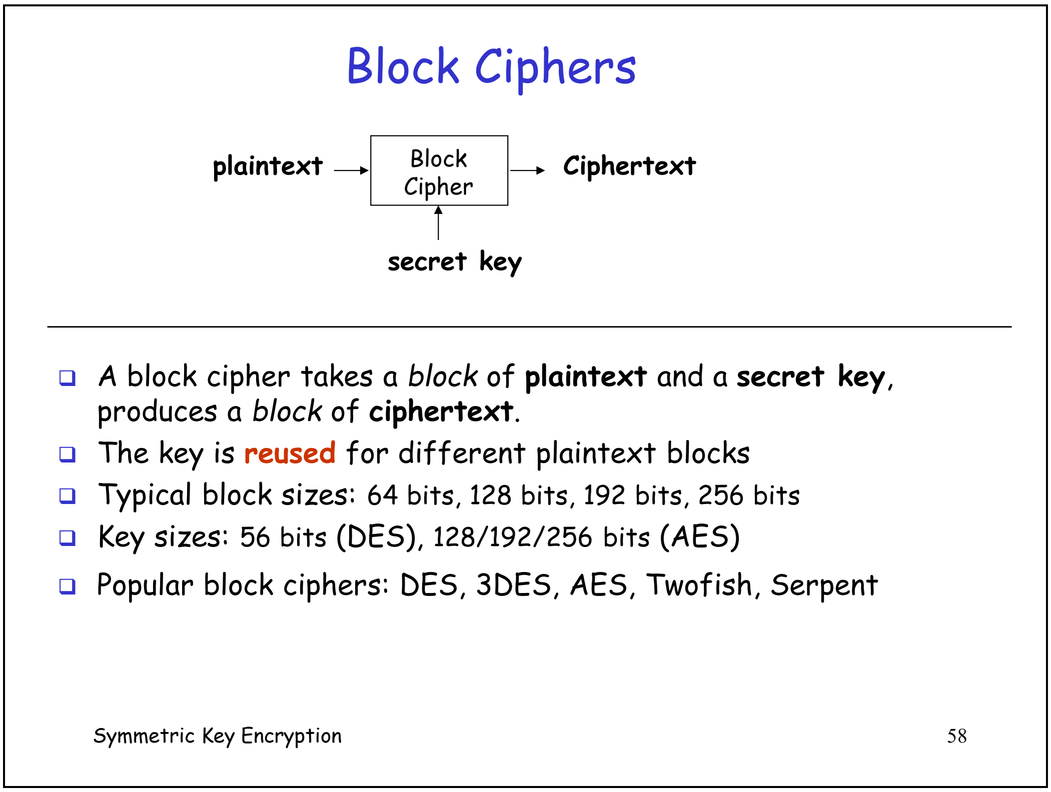
Symmetric Key Encryption 57

What is symmetric key cryptosystem?

Finished with looking at stream and block cipher.

You can use this picture to understand how cryptosystem we will look at relate to each other.

Symmetric encryption we can have either a stream cipher of block cipher.



Block Ciphers

plaintext → Block Cipher → Ciphertext

secret key

- A block cipher takes a *block* of **plaintext** and a **secret key**, produces a *block* of **ciphertext**.
- The key is **reused** for different plaintext blocks
- Typical block sizes: 64 bits, 128 bits, 192 bits, 256 bits
- Key sizes: 56 bits (DES), 128/192/256 bits (AES)
- Popular block ciphers: DES, 3DES, AES, Twofish, Serpent

Study this slides

You must at least know the most common block ciphers like DES, 3DES and AES.

What is the main difference between stream cipher and block cipher?

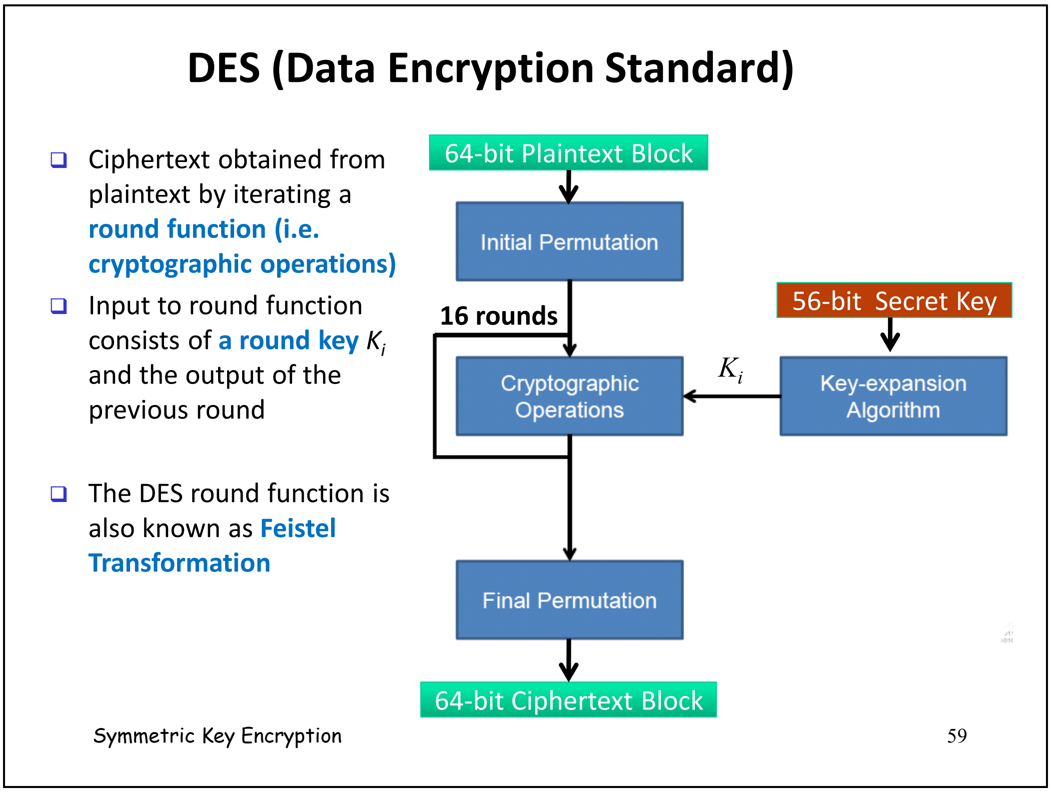
A Stream cipher used a cryptographic function and key to generate keystream the same length as the plaintext – it can be any length. The key stream is then XORed to the plaintext to form the ciphertext.

This means that the function does not need to be reversible it just needs to generate good unpredictable keystream. You do not need to reverse the function, the actual plaintext encryption and decryption combination is the XOR.

Block ciphers takes a plaintext input of fixed size (the block size, if you do not have enough plaintext you need to pad until you have a block of required size). It then generates ciphertext of a fixed size (the block size).

The block cipher cryptographic function actually works on the plaintext (for example, it does permutation/substitution,etc. on the plaintext) to come up with ciphertext. This means the block cipher function must be reversible/invertible (i.e. given ciphertext you must be able to work back to plaintext).

Do not confuse stream ciphers with block cipher modes of operation that try to approximate stream ciphers – in such mode the underlying function is still a block cipher the output of the deterministic function is still multiples of the block size.



DES (Data Encryption Standard)

- Ciphertext obtained from plaintext by iterating a **round function (i.e. cryptographic operations)**
- Input to round function consists of **a round key** K_i and the output of the previous round
- The DES round function is also known as **Feistel Transformation**

You should know the basic operation of DES, like block size, key size and basic architecture (initial, final permutation and 16 rounds each with own round key).

Text below is for reference only

1973: NBS publish a call for proposals for an encryption algorithm standard.

1974: IBM encouraged to submit an encryption algorithm.

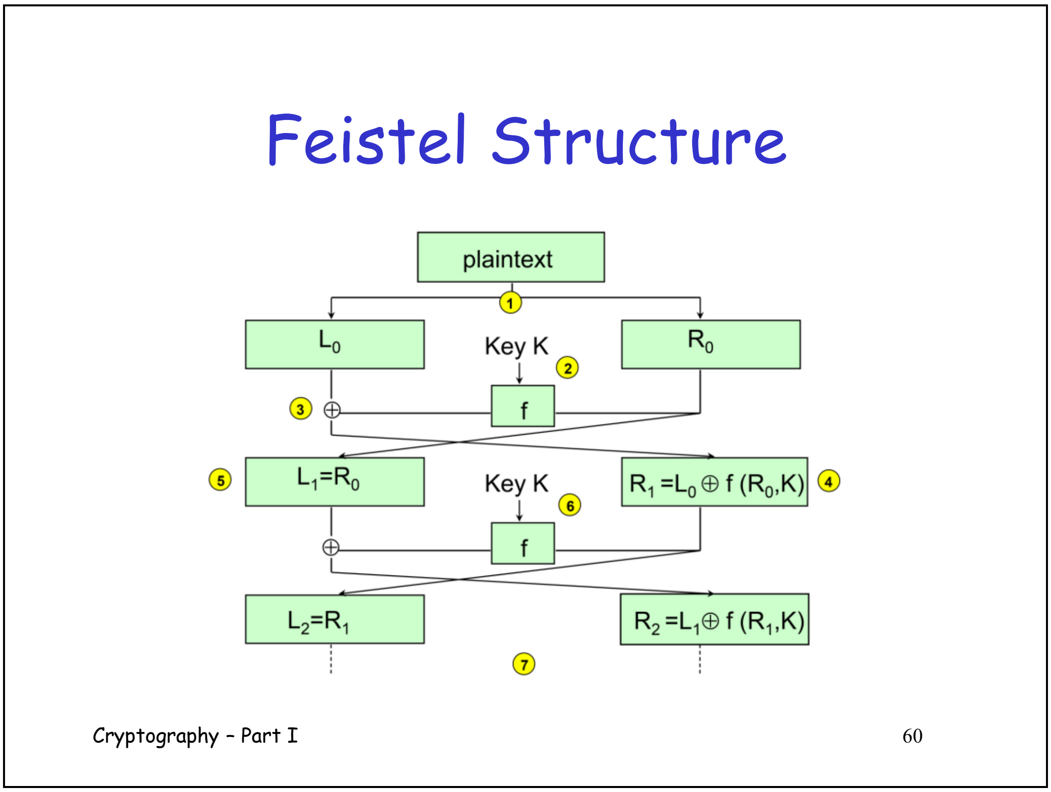
1976: After consultation this algorithm adopted as a federal standard and then published as DES the following year.

1977: DES mandatory for Federal Agencies and adopted as ANSI X3.92 and used throughout international financial industry.

1988: NSA removed its endorsement of DES (despite predicted 15-year lifespan).

1988: NBS reaffirmed use of DES to appease the financial industry.

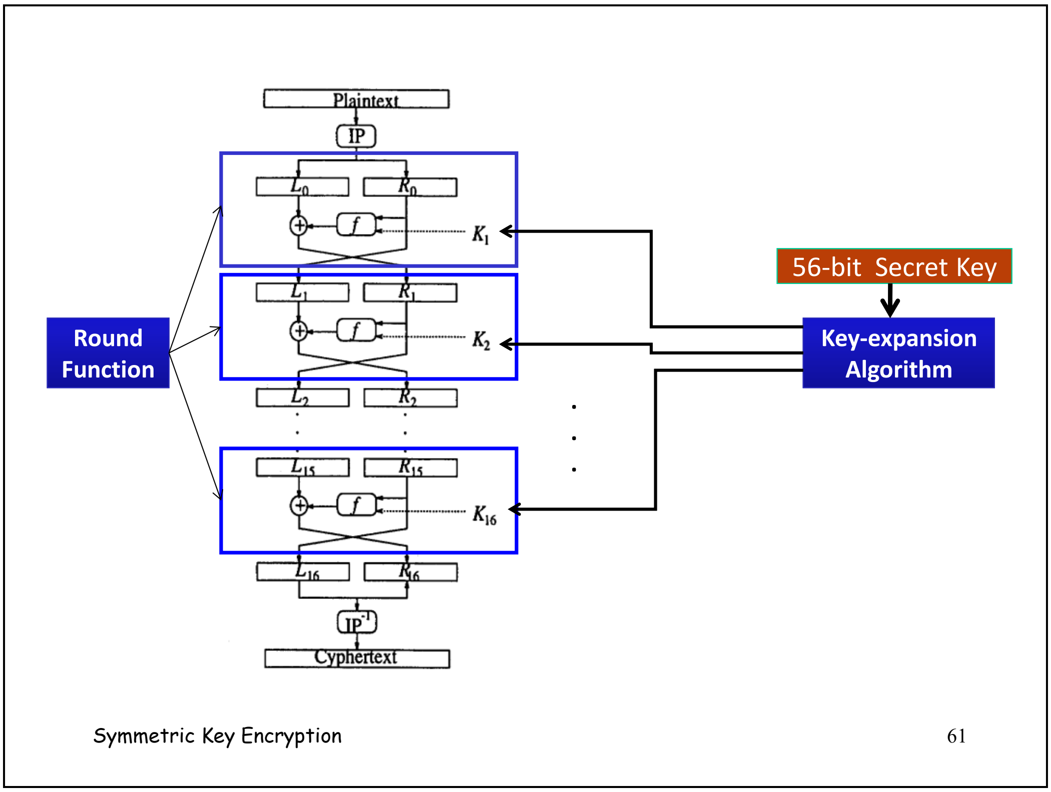
1998: NIST issue a new call for an algorithm in 1998.



Feistel Structure

Interest only

We only deal with half the data during one round.



Round Function

56-bit Secret Key

Key-expansion Algorithm

Symmetric Key Encryption 61

Interest only

A Feistel cipher is a basic architecture used in a number of ciphers).

This is the structure shown about where each round as a split of text into left and right 32 bits, Li+1=Ri and Ri+1=f(Ri) XOR Li

1. How do you choose the round function f ?

Feistel cipher – example – f does not need to be reversible. It can be one way, the reversibility comes from the XOR

2. How do you decrypt using a Feistel cipher?

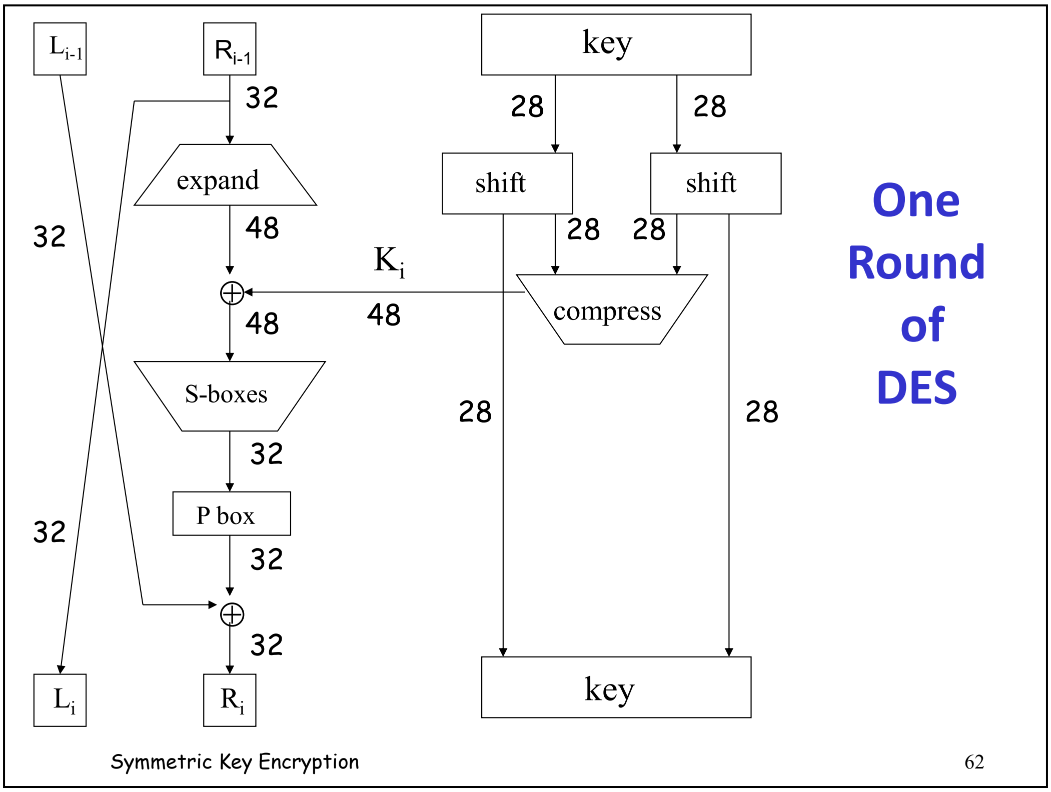
How to we decrypt flip when whe have a feistel architecture? order of rounds to decrypt (i.e. order you used the round keys) , otherwise exactly the same logic.

This saves resources, 1976 what is status of computers (Apple I released 1976 – 1 MHz clock, standard 4kB memory).

3. How many rounds should a Feistel cipher have?

Generally you need more rounds, because you are only working on half the block at a time.

Key expansion – need out one 56 bit key for 16 rounds



One Round of DES

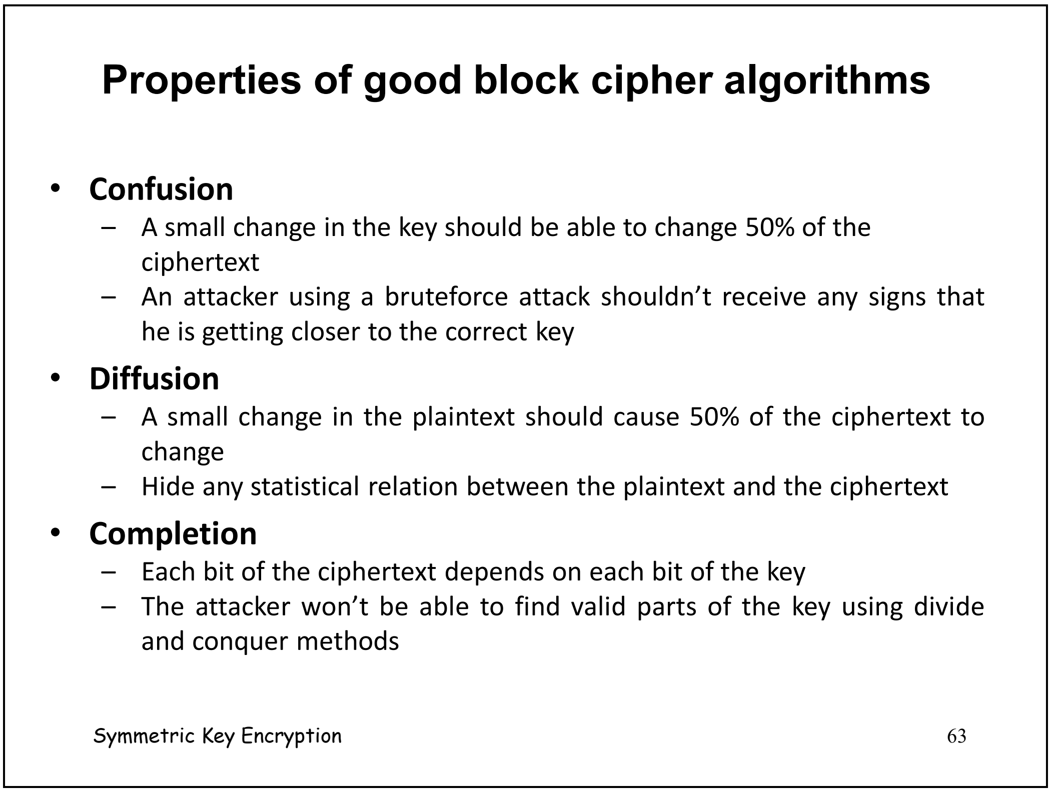
Symmetric Key Encryption 62

You should know that in DES the round function F is a expansion, substitution and permutation…

Substitution every 6 bits to 4 bits

P box – one fixed shuffle of each bit

The rest is for references only.

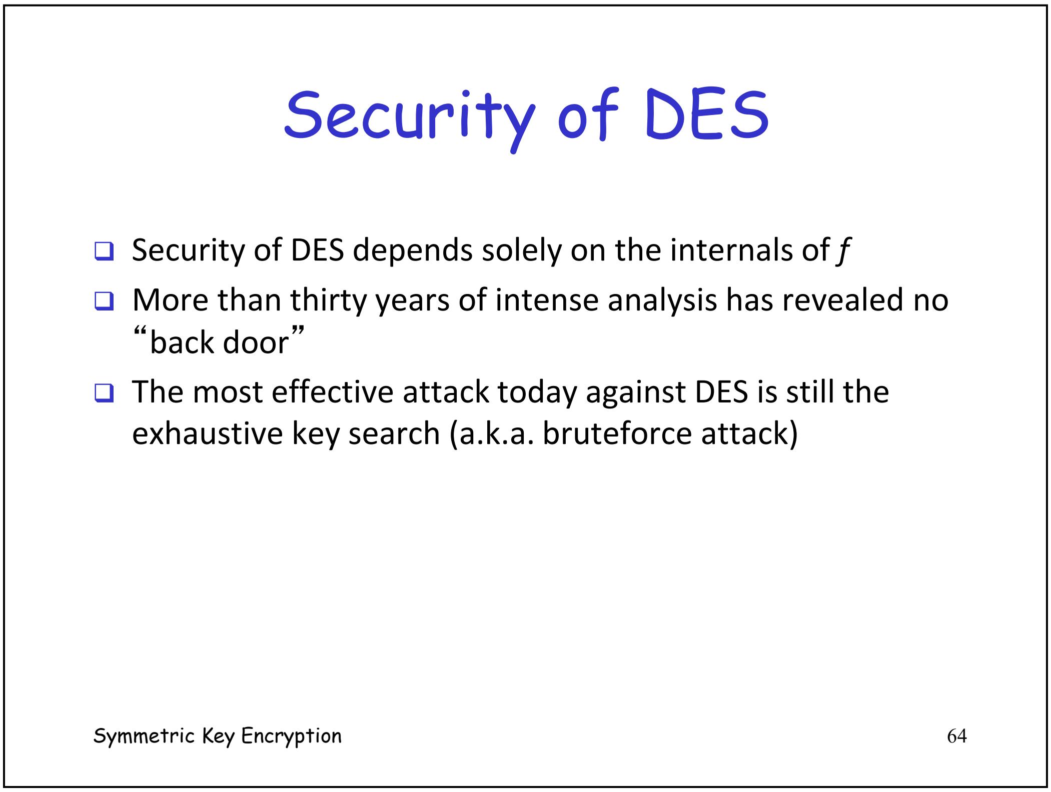


Properties of good block cipher algorithms

- **Confusion**
  - A small change in the key should be able to change 50% of the ciphertext
  - An attacker using a bruteforce attack shouldn’t receive any signs that he is getting closer to the correct key
- **Diffusion**
  - A small change in the plaintext should cause 50% of the ciphertext to change
  - Hide any statistical relation between the plaintext and the ciphertext
- **Completion**
  - Each bit of the ciphertext depends on each bit of the key
  - The attacker won’t be able to find valid parts of the key using divide and conquer methods

Study this slide

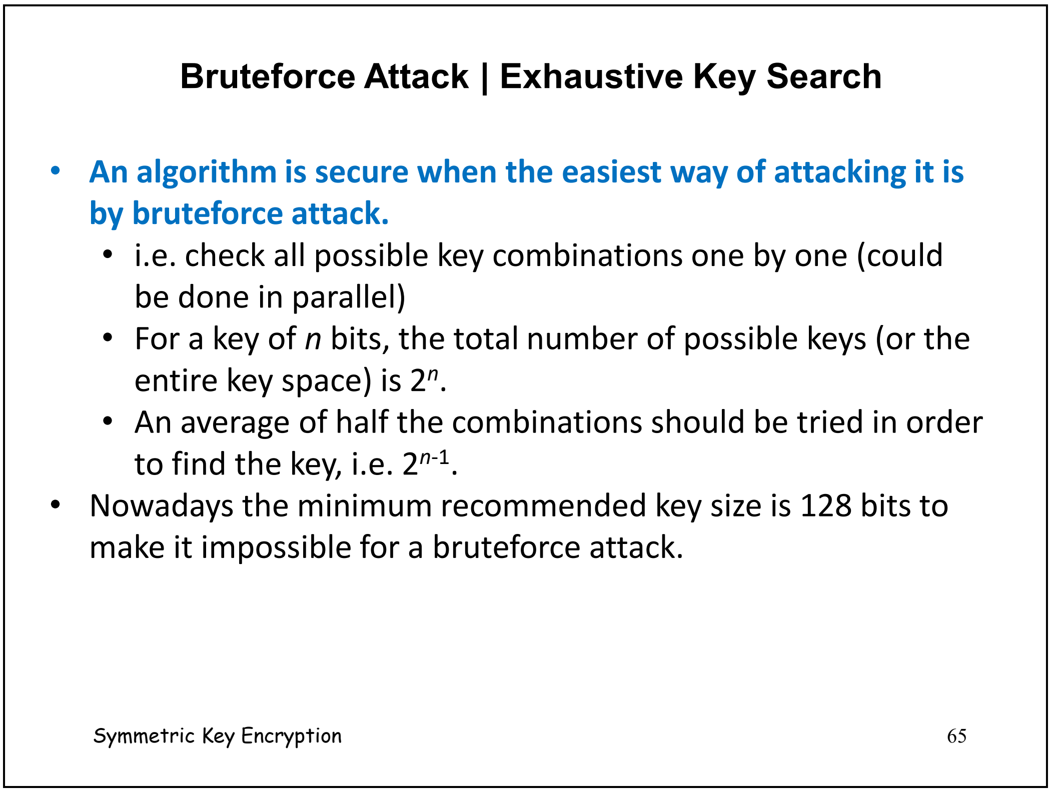
Completion means if we flip a bit if it always change the same 50% of bits not



Security of DES

- Security of DES depends solely on the internals of *f*
- More than thirty years of intense analysis has revealed no "back door"
- The most effective attack today against DES is still the exhaustive key search (a.k.a. bruteforce attack)

Best attack against DES remains exhaustive key search



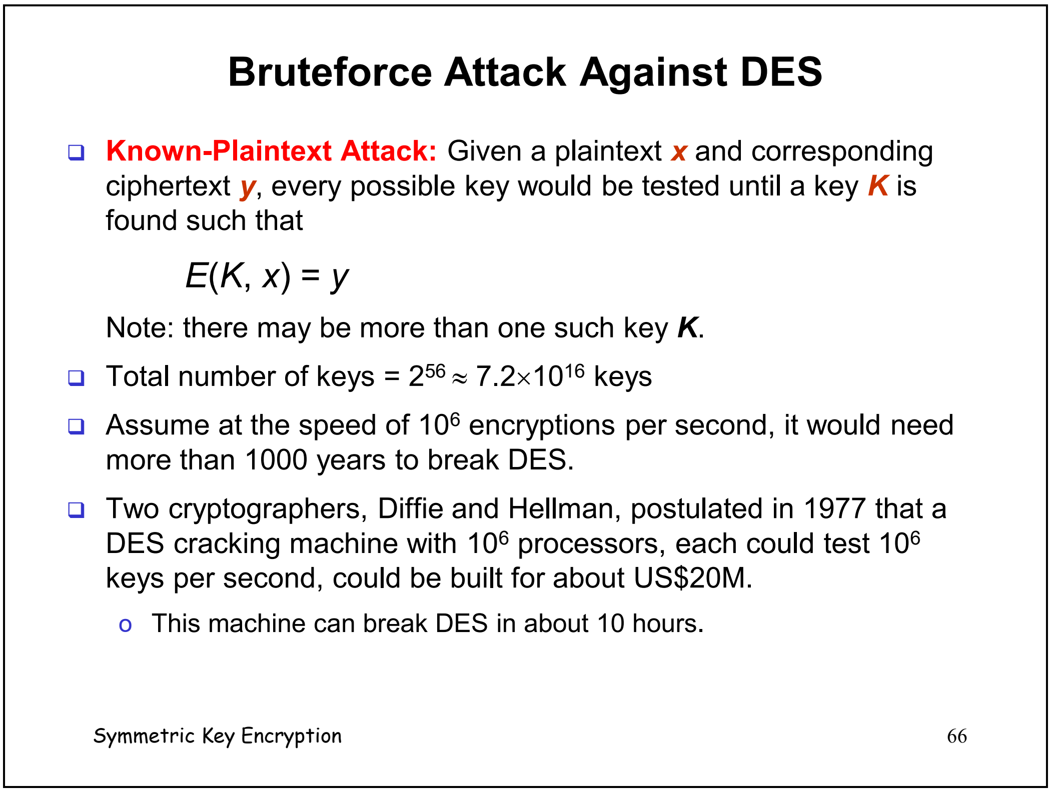
Bruteforce Attack | Exhaustive Key Search

- **An algorithm is secure when the easiest way of attacking it is by bruteforce attack.**
  - i.e. check all possible key combinations one by one (could be done in parallel)
  - For a key of *n* bits, the total number of possible keys (or the entire key space) is 2^n.
  - An average of half the combinations should be tried in order to find the key, i.e. 2^{n-1}.
- Nowadays the minimum recommended key size is 128 bits to make it impossible for a bruteforce attack.

Know the recommended key size, and the minimum key size.

128 bit symmetric keys recommended until 2030.

Also note that if doing brute force search on average you need to search half the key space (2^(n-1)).



Bruteforce Attack Against DES

- **Known-Plaintext Attack:** Given a plaintext ***x*** and corresponding ciphertext ***y***, every possible key would be tested until a key ***K*** is found such that

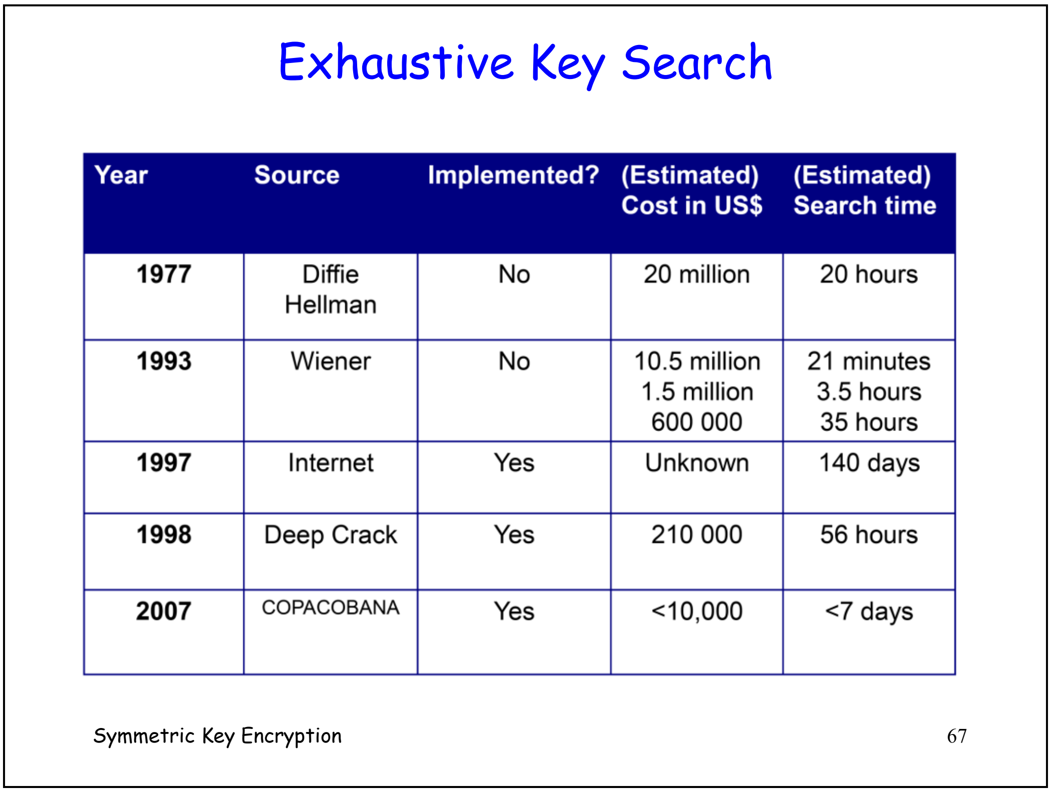
  $$E(K, x) = y$$

  Note: there may be more than one such key ***K***.
- Total number of keys = $2^{56} \approx 7.2\times10^{16}$ keys
- Assume at the speed of 10^6 encryptions per second, it would need more than 1000 years to break DES.
- Two cryptographers, Diffie and Hellman, postulated in 1977 that a DES cracking machine with 10^6 processors, each could test 10^6 keys per second, could be built for about US$20M.
  - This machine can break DES in about 10 hours.

Only know that brute force attack (finding key with know plaintext) is now viable against DES.

US$20 million in 1977 is about US$ 107 million today

Rest of Slide 66 and 67 for reference



Exhaustive Key Search

Year	Source	Implemented?	(Estimated) Cost in US$	(Estimated) Search time
1977	Diffie Hellman	No	20 million	20 hours
1993	Wiener	No	10.5 million 1.5 million 600 000	21 minutes 3.5 hours 35 hours
1997	Internet	Yes	Unknown	140 days
1998	Deep Crack	Yes	210 000	56 hours
2007	COPACOBANA	Yes	<10,000	<7 days

Rest for reference

- In 1993, Michael Wiener presented a pipelined chip which tests 5×10^7 DES keys per second.
  - Each chip could cost US$10 and a frame of 5760 chips would cost about $100K.

- In 1998, DES cracker (nicknamed "Deep Crack" http://en.wikipedia.org/wiki/EFF_DES_cracker) was built by the Electronic

Frontier Foundation (EFF).

- •It performs 2^{56} DES operations in 56 hours. 90 billion searches per second.
- •Cost: US$250K (first piece), US$50K - $75K (duplicates).

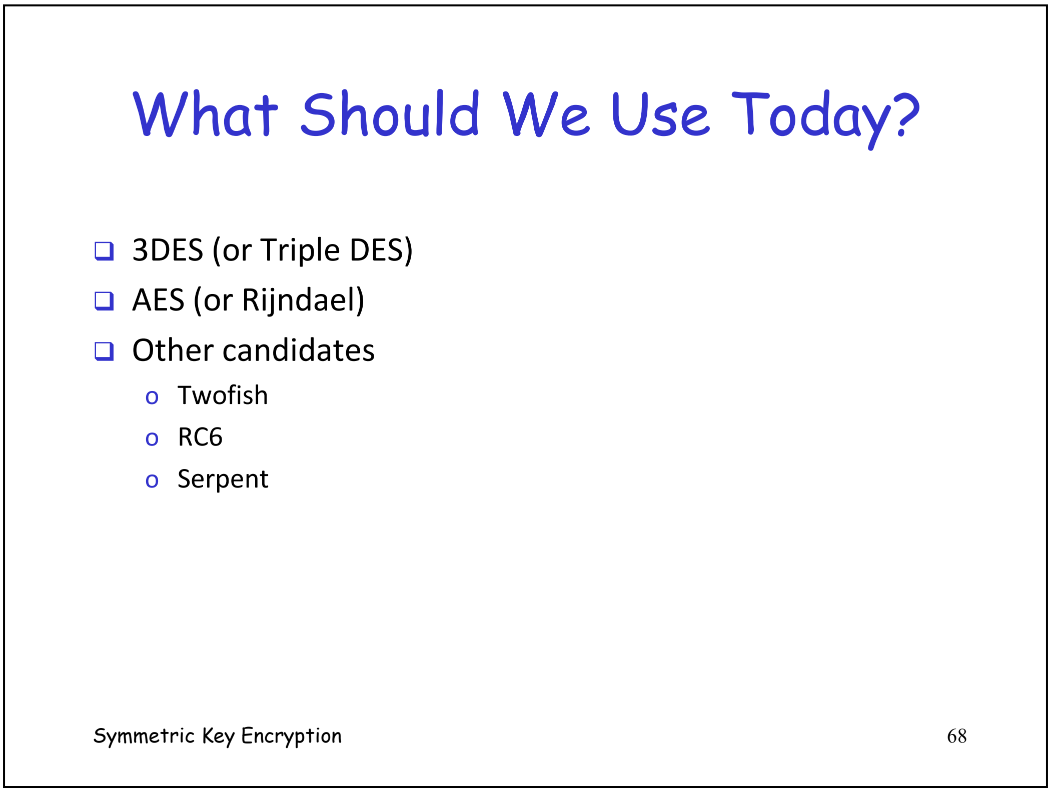
•Software version of DES cracking effort can be found at

http://www.distributed.net/des/

2007 we also had the COPACOBANA platforms – parallel FPGA platform (128 FPGAs)
10,000 USD dedicate hardware platform – took about a week for a key.

•Other examples: Record 22 hrs and 15 mins by distributed software cracking effort, various attempts with GPU/CUDA around similar to COPACOBANA days.

•Total bitcoin mining system 2^65 hash per second DES 1 millisecond…

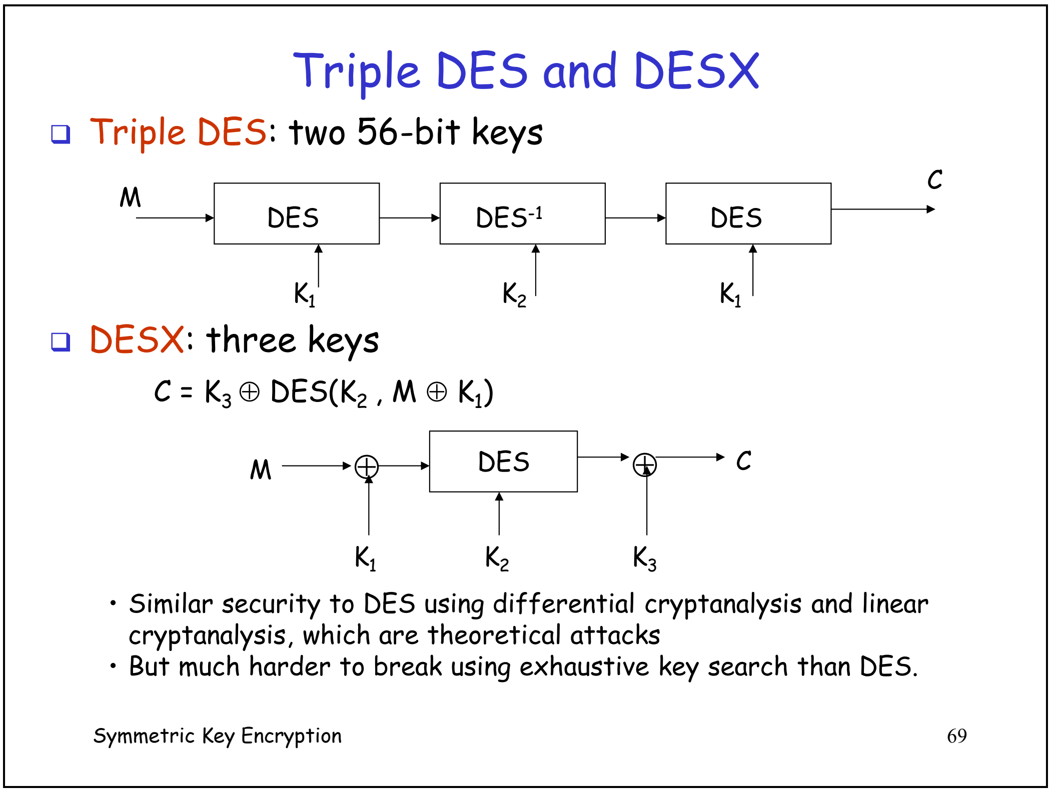


What Should We Use Today?

- 3DES (or Triple DES)
- AES (or Rijndael)
- Other candidates
  - Twofish
  - RC6
  - Serpent

We should use 3DES or ideally AES (other ciphers for reference).

What is the main problem with DES? How should we fix? Address key size



Triple DES and DESX

- Triple DES: two 56-bit keys

M
DES
DES^{-1}
DES
C
K_1
K_2
K_1

- DESX: three keys

$C = K_3 \oplus DES(K_2 , M \oplus K_1)$

M
DES
C
K_1
K_2
K_3

- Similar security to DES using differential cryptanalysis and linear cryptanalysis, which are theoretical attacks
- But much harder to break using exhaustive key search than DES.

Symmetric Key Encryption 69

Please study these variations of DES (you do not need to memorise the architecture)

Triple DES – Encrypt, decrypt then encrypt.

Two key (use K1 K2 and K1 again) and three key variety (use K1 K2 and K3)

In theory this gives us 112 and 168 bit keyspace but due to some improved cryptanalysis attacks the keyspace is actually reduced (in other words there are better attacks than Brute Force):

This reduced keyspace is called the 'effective key space'

For 2-key effective key space is 80 bits (what is the effort needed? 2^79

For 3-key effective key space is 112 bits

Better than DES but less secure than AES

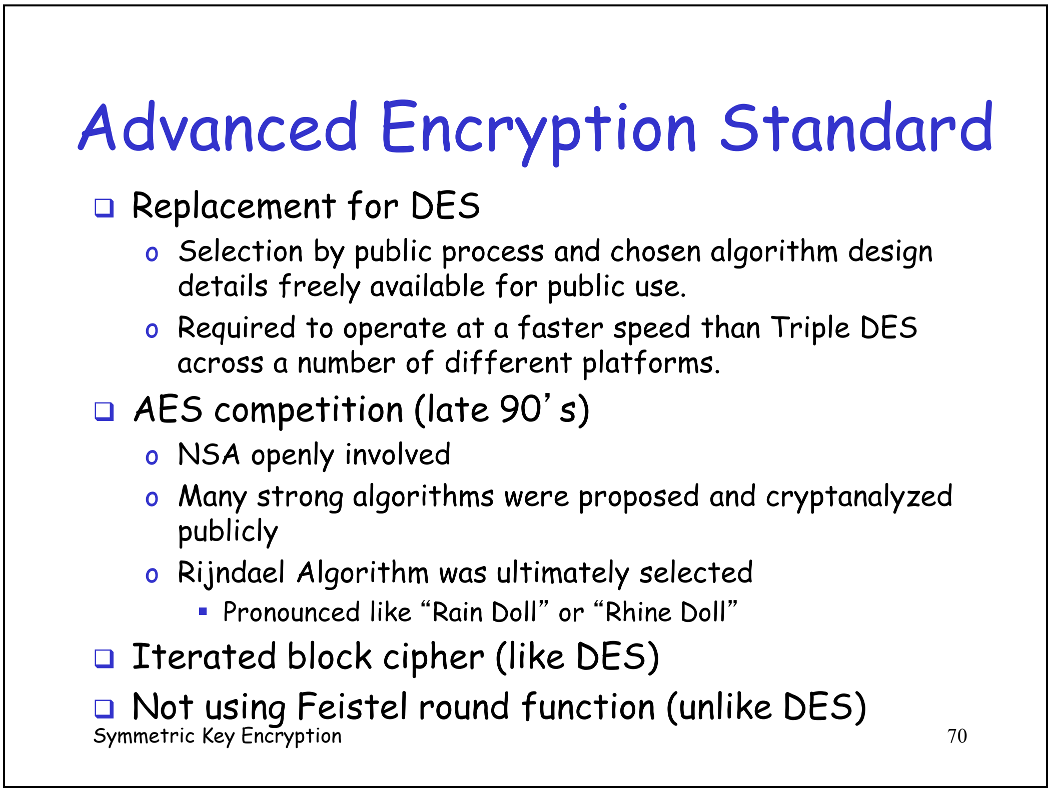
Why do we still have DES – legacy applications, most banking standards use DES.

Why do we have the encrypt/decrypt/encrypt design?

Backward compatibility? K1=K2 (and = K3 in three key model) – then it is just single DES.

DESX

What is the length of K1 K2 and K3? K1 and K3 = |M| while K2 is just a DES key (56-bits)



Advanced Encryption Standard

- ❑ Replacement for DES
  - o Selection by public process and chosen algorithm design details freely available for public use.
  - o Required to operate at a faster speed than Triple DES across a number of different platforms.
- ❑ AES competition (late 90’s)
  - o NSA openly involved
  - o Many strong algorithms were proposed and cryptanalyzed publicly
  - o Rijndael Algorithm was ultimately selected
    - ▪ Pronounced like “Rain Doll” or “Rhine Doll”
- ❑ Iterated block cipher (like DES)
- ❑ Not using Feistel round function (unlike DES)

You should know the basic operation of AES, like block size, key size and basic architecture (the basic idea of the permutation and it having multiple rounds each with own round key).

Designed by Vincent Rijmen and Joan Daemen (the name of the cipher is combination of their name RIJnDAEl)

Other secure algorithms: Serpent, TwoFish, RC6

Rijndael chosen because it had the best performance across all the proposals (especially 8-bit processors).

We will now look at AES in more detail – this is for your interest. We would like for you to know a little about the inner workings of a cipher – but I do not expect that you remember all the lower details but you must be able to give a basic description of the round function and overall structure. So for example, know the basic intention of subbyte, shiftrow, mixcolumn and add round key (but the actual operation, like finite field arithmetic for mixcolumn is not needed)

Know AES is not a Feistel design, and that AES is now recommended for new systems.

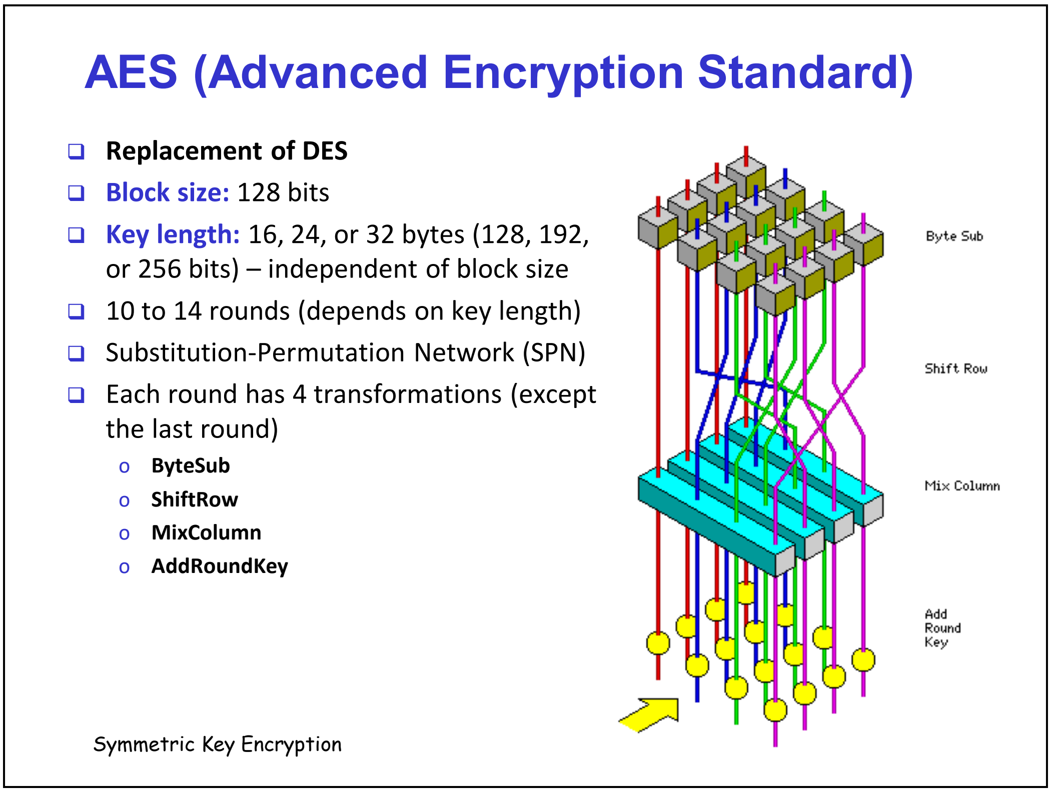
For reference only

Requirements:

128-bit block

128-256 bit key

Faster than 3DES in a number of specified platforms



AES (Advanced Encryption Standard)

- ❑ **Replacement of DES**
- ❑ **Block size:** 128 bits
- ❑ **Key length:** 16, 24, or 32 bytes (128, 192, or 256 bits) – independent of block size
- ❑ 10 to 14 rounds (depends on key length)
- ❑ Substitution-Permutation Network (SPN)
- ❑ Each round has 4 transformations (except the last round)
  - o **ByteSub**
  - o **ShiftRow**
  - o **MixColumn**
  - o **AddRoundKey**

Byte Sub
Shift Row
Mix Column
Add Round Key

Symmetric Key Encryption

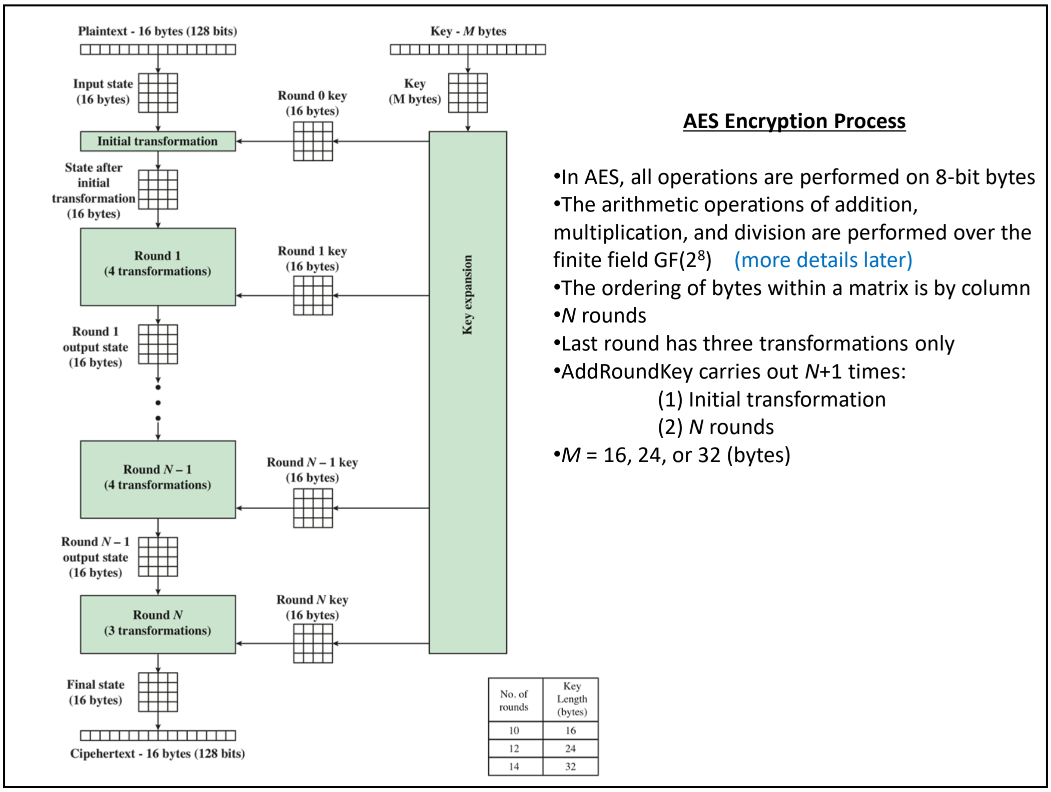
Study this slide

Think back - what is the block size? We can only input into a block cipher a fixed length piece of plaintext of size equal to the block. No less, no more.

What was key size and block size for DES? 56 and 64.

You must be able to describe AES – the detail on this slide is important.

You do not need to study slides 71-87 in detail but you need to look through it an be able to explain in a sentence or two each transformation, and also the general order of the transformations (and round keys) during an encryption and decryption of data.



Plaintext - 16 bytes (128 bits)

Input state (16 bytes)

Initial transformation

State after initial transformation (16 bytes)

Round 1 (4 transformations)

Round 1 output state (16 bytes)

Round *N* – 1 (4 transformations)

Round *N* – 1 output state (16 bytes)

Round *N* (3 transformations)

Final state (16 bytes)

Cipehertext - 16 bytes (128 bits)

Key - *M* bytes

Key (M bytes)

Key expansion

Round 0 key (16 bytes)

Round 1 key (16 bytes)

Round *N* – 1 key (16 bytes)

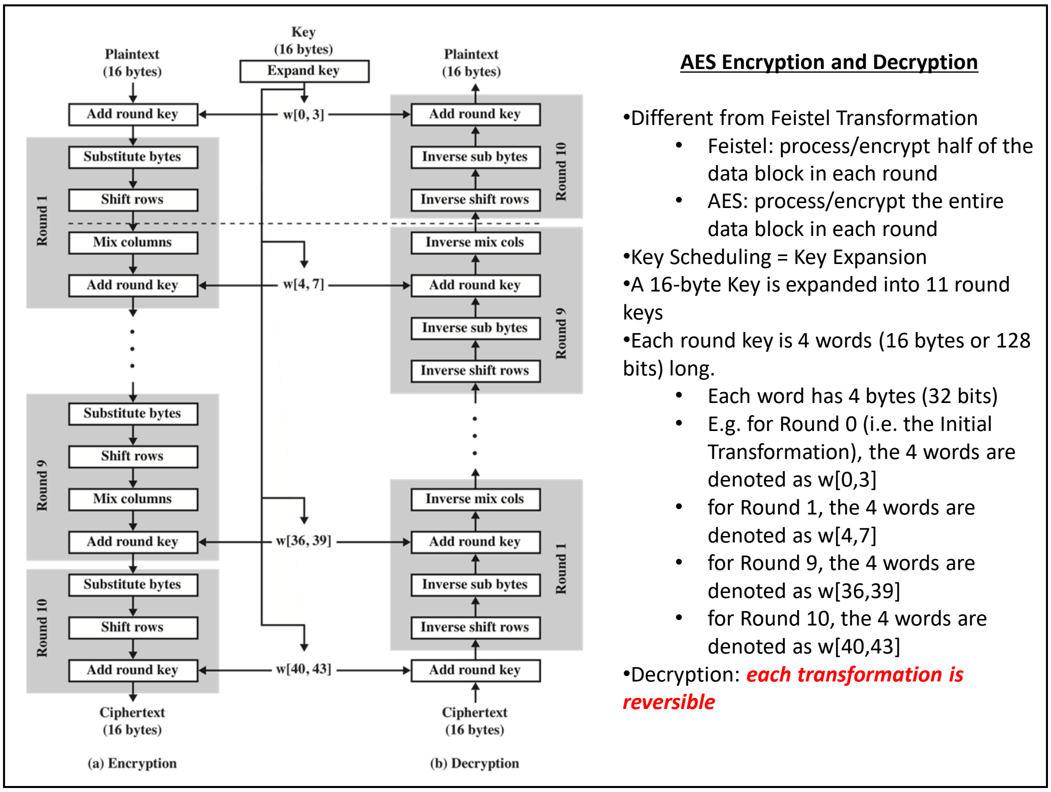
Round *N* key (16 bytes)

No. of rounds	Key Length (bytes)
10	16
12	24
14	32

AES Encryption Process

- In AES, all operations are performed on 8-bit bytes
- The arithmetic operations of addition, multiplication, and division are performed over the finite field GF(2^8) (more details later)
- The ordering of bytes within a matrix is by column
- *N* rounds
- Last round has three transformations only
- AddRoundKey carries out *N*+1 times:
  (1) Initial transformation
  (2) *N* rounds
- *M* = 16, 24, or 32 (bytes)

This slide shows the high level process of encryption – multiple rounds each with 4 transformation (except for initial permutation and final round).

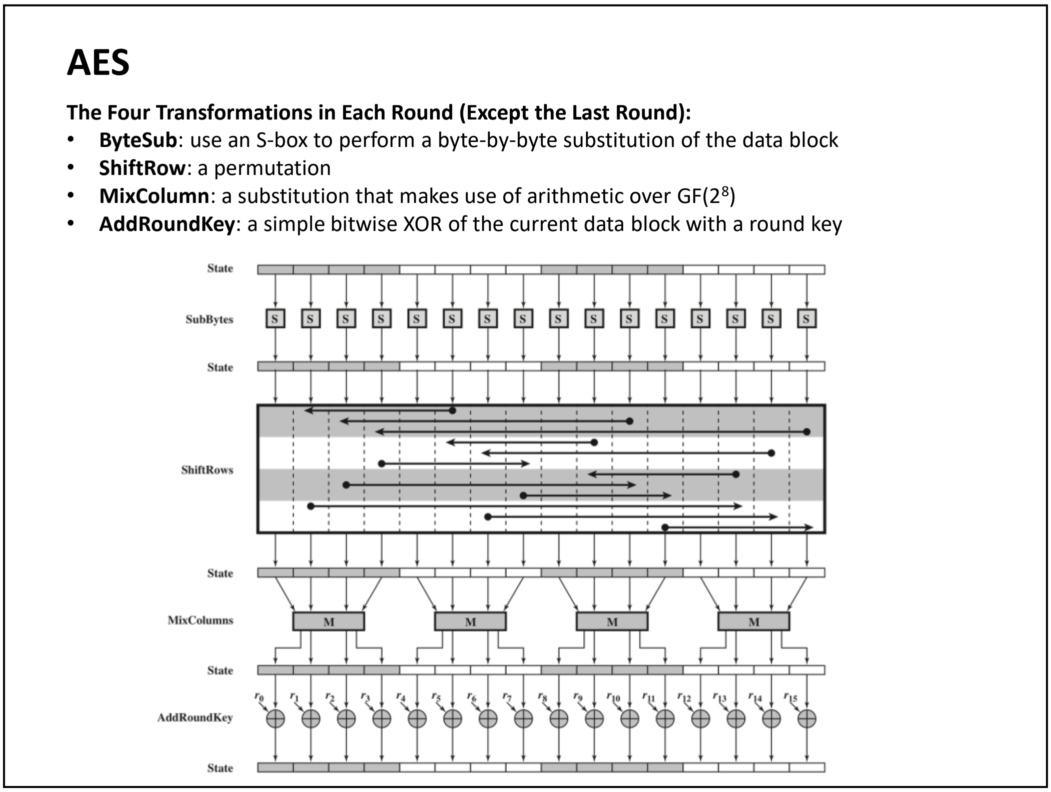


AES Encryption and Decryption

- Different from Feistel Transformation
  - Feistel: process/encrypt half of the data block in each round
  - AES: process/encrypt the entire data block in each round
- Key Scheduling = Key Expansion
- A 16-byte Key is expanded into 11 round keys
- Each round key is 4 words (16 bytes or 128 bits) long.
  - Each word has 4 bytes (32 bits)
  - E.g. for Round 0 (i.e. the Initial Transformation), the 4 words are denoted as w[0,3]
  - for Round 1, the 4 words are denoted as w[4,7]
  - for Round 9, the 4 words are denoted as w[36,39]
  - for Round 10, the 4 words are denoted as w[40,43]
- Decryption: ***each transformation is reversible***

This slide shows the order of transformations of encryption and decryption.

Understand that decryption is the reverse of encryption – all functions are reversible.



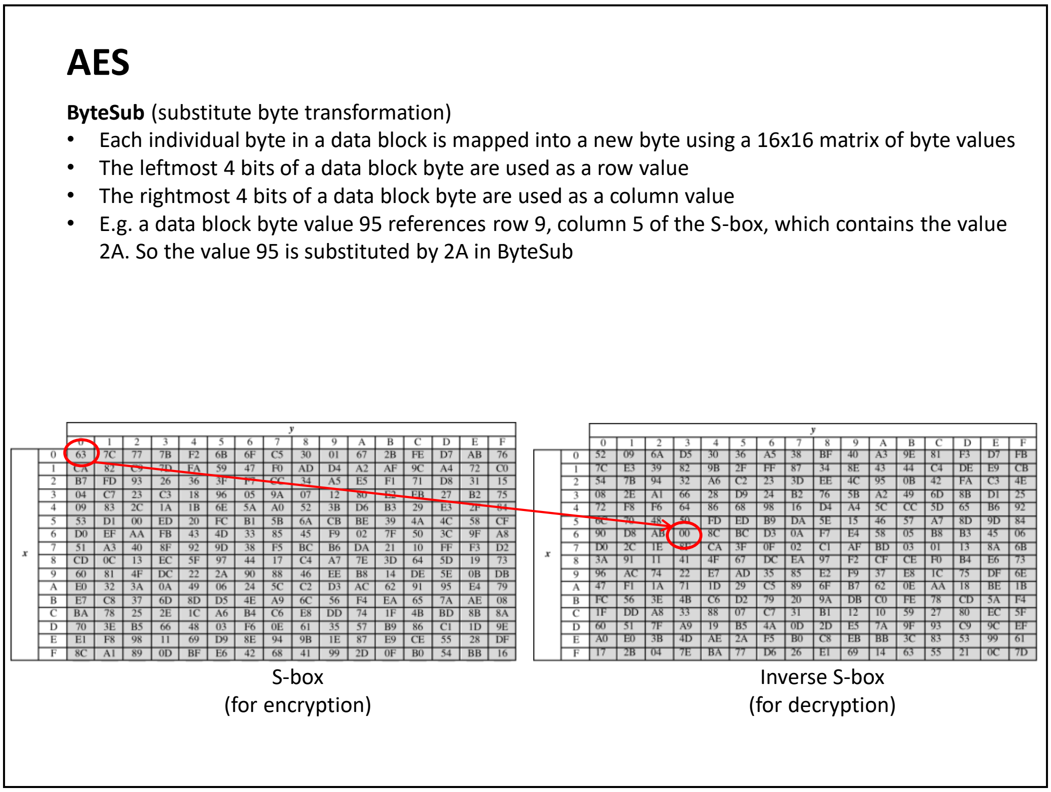
AES

The Four Transformations in Each Round (Except the Last Round):

- **ByteSub**: use an S-box to perform a byte-by-byte substitution of the data block
- **ShiftRow**: a permutation
- **MixColumn**: a substitution that makes use of arithmetic over GF(2^8)
- **AddRoundKey**: a simple bitwise XOR of the current data block with a round key

State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
r_0 r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9 r_{10} r_{11} r_{12} r_{13} r_{14} r_{15}
AddRoundKey
State

Overall visual summary of what happens to the data during changing from plaintext to ciphertext during ONE round.



AES

ByteSub (substitute byte transformation)

- Each individual byte in a data block is mapped into a new byte using a 16x16 matrix of byte values
- The leftmost 4 bits of a data block byte are used as a row value
- The rightmost 4 bits of a data block byte are used as a column value
- E.g. a data block byte value 95 references row 9, column 5 of the S-box, which contains the value 2A. So the value 95 is substituted by 2A in ByteSub

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

S-box
(for encryption)

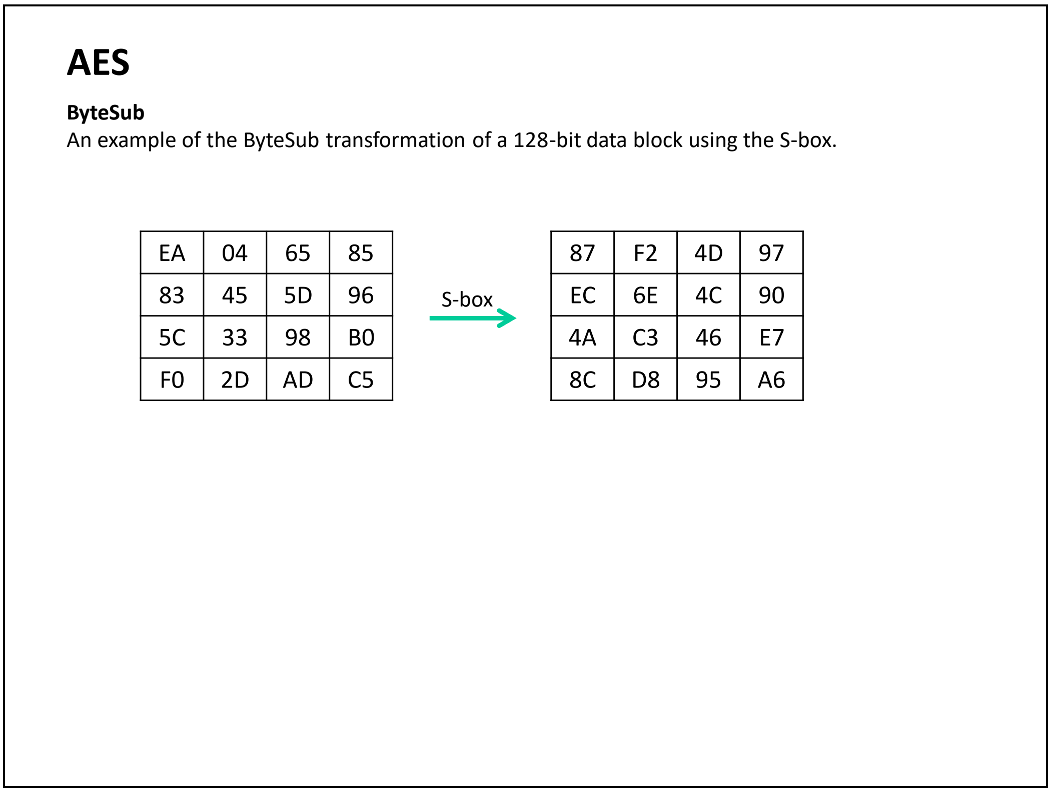
		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Inverse S-box
(for decryption)

More detail on Byte Substitution transform – just a lookup table.

There are two table – one for encrypt and one decrypt.

For the picture lets say we sub 00, then it is 63. For decryption if we look up 63 then 00. Reversible.



AES

ByteSub

An example of the ByteSub transformation of a 128-bit data block using the S-box.

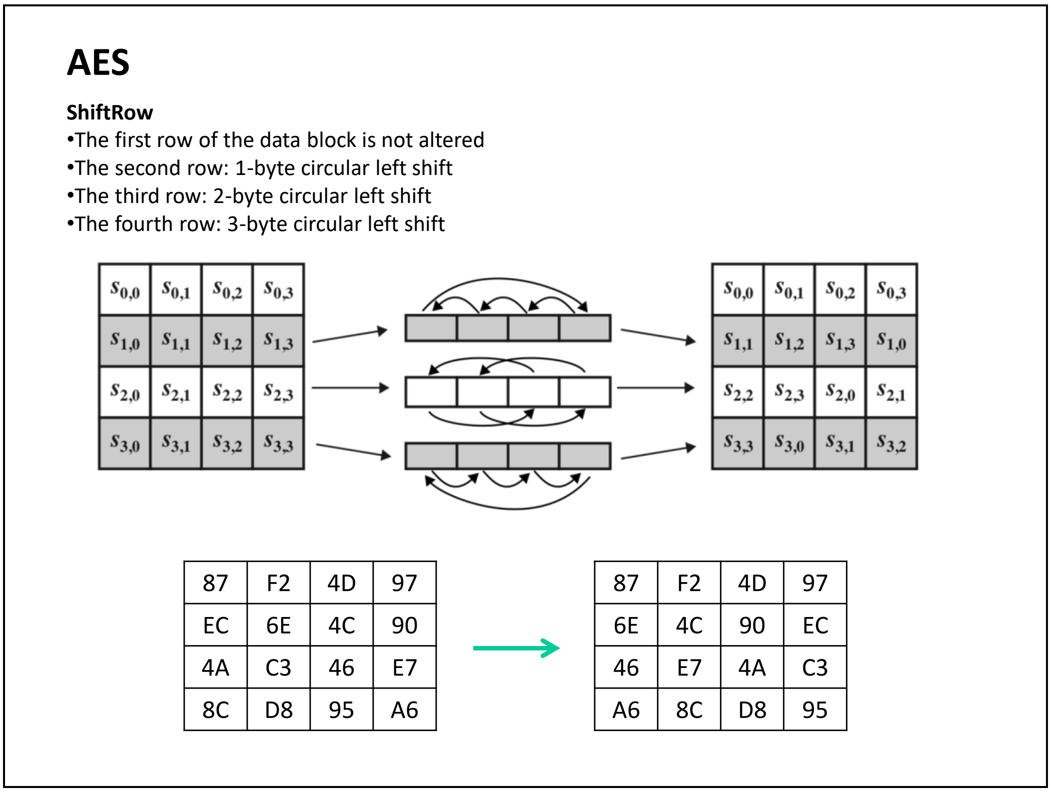
EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

S-box →

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

Check previous slide and try for yourself.

See if you can substitute and get same answers



AES

ShiftRow

- •The first row of the data block is not altered
- •The second row: 1-byte circular left shift
- •The third row: 2-byte circular left shift
- •The fourth row: 3-byte circular left shift

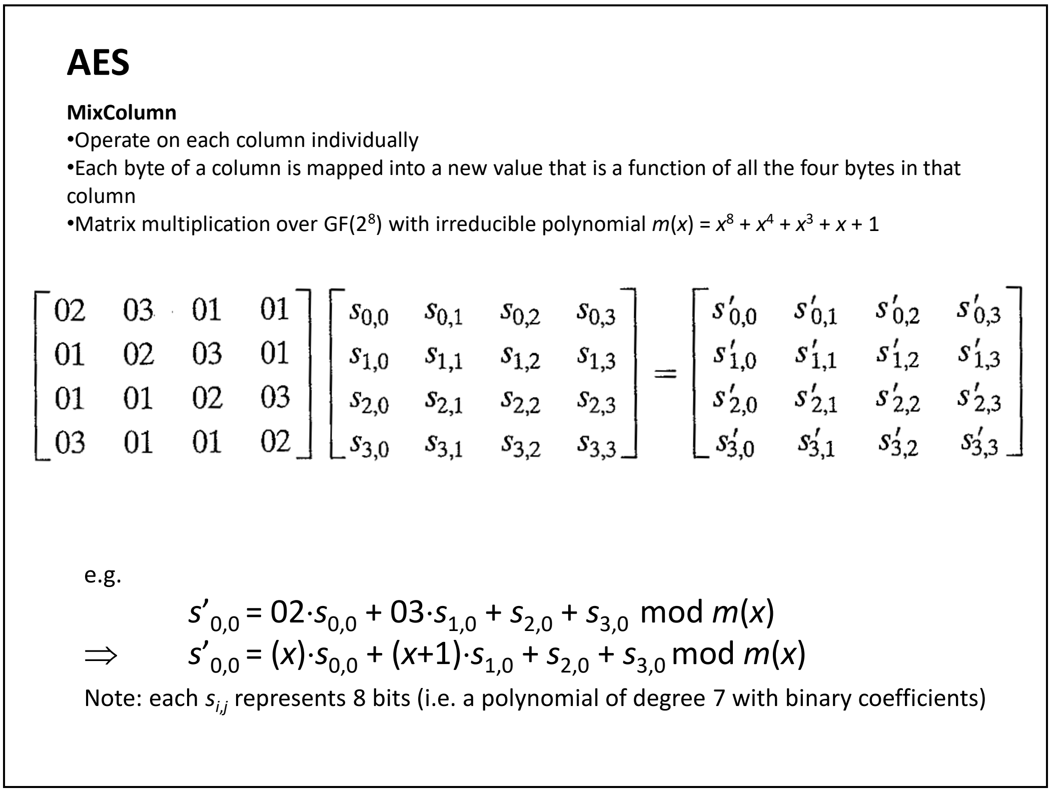
$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,1}$	$s_{1,2}$	$s_{1,3}$	$s_{1,0}$
$s_{2,2}$	$s_{2,3}$	$s_{2,0}$	$s_{2,1}$
$s_{3,3}$	$s_{3,0}$	$s_{3,1}$	$s_{3,2}$

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

Shift row just means we left rotate each row of the 4x4 data matrix



AES

MixColumn

- Operate on each column individually
- Each byte of a column is mapped into a new value that is a function of all the four bytes in that column
- Matrix multiplication over GF(2^8) with irreducible polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

e.g.

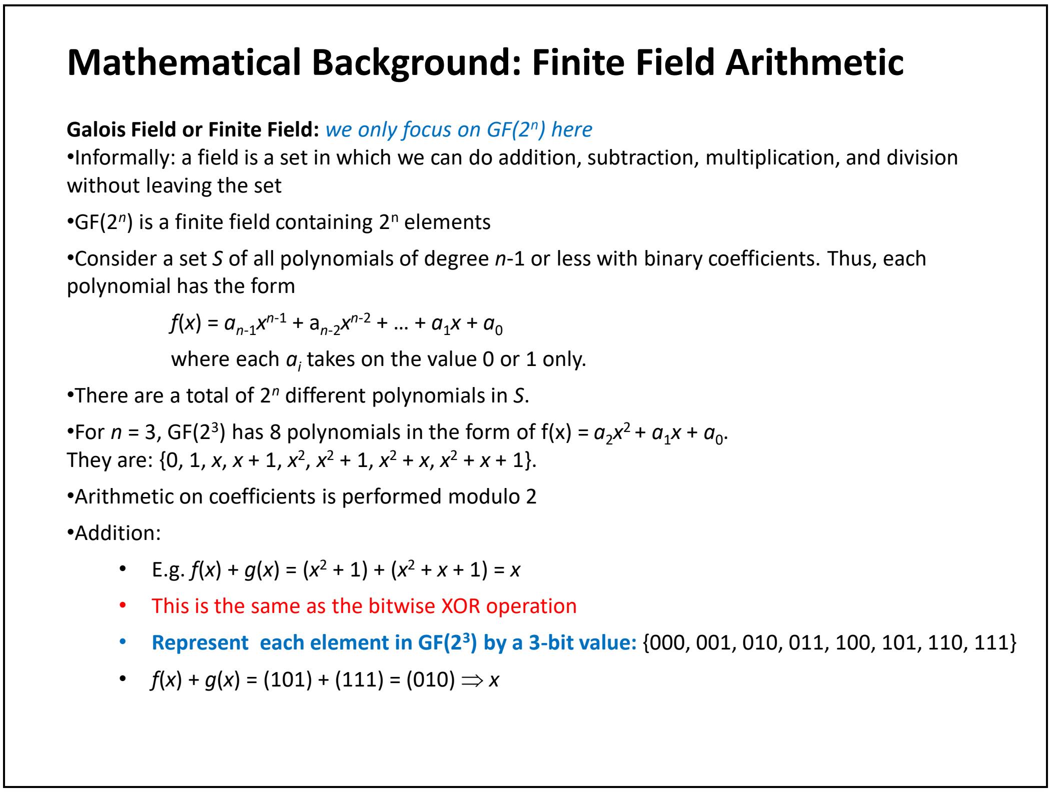
$$s'_{0,0} = 02 \cdot s_{0,0} + 03 \cdot s_{1,0} + s_{2,0} + s_{3,0} \bmod m(x)$$

$$\Rightarrow \quad s'_{0,0} = (x) \cdot s_{0,0} + (x+1) \cdot s_{1,0} + s_{2,0} + s_{3,0} \bmod m(x)$$

Note: each $s_{i,j}$ represents 8 bits (i.e. a polynomial of degree 7 with binary coefficients)

MixColumn does not mean moving around the column – in means we multiply the data matrix over GF(2^8) mod m(x) to create a new matrix.

Do decrypt we multiple by the inverse of the Matrix shown on the left here (no need to know the exact matrix)



Mathematical Background: Finite Field Arithmetic

Galois Field or Finite Field: *we only focus on GF(2^n) here*

•Informally: a field is a set in which we can do addition, subtraction, multiplication, and division without leaving the set

•GF(2^n) is a finite field containing 2^n elements

•Consider a set S of all polynomials of degree n-1 or less with binary coefficients. Thus, each polynomial has the form

$$f(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \dots + a_1x + a_0$$

where each a_i takes on the value 0 or 1 only.

•There are a total of 2^n different polynomials in S.

•For n = 3, GF(2^3) has 8 polynomials in the form of $f(x) = a_2x^2 + a_1x + a_0$.
They are: {0, 1, x, x + 1, x^2, x^2 + 1, x^2 + x, x^2 + x + 1}.

•Arithmetic on coefficients is performed modulo 2

•Addition:

- E.g. $f(x) + g(x) = (x^2 + 1) + (x^2 + x + 1) = x$
- This is the same as the bitwise XOR operation
- **Represent each element in GF(2^3) by a 3-bit value:** {000, 001, 010, 011, 100, 101, 110, 111}
- $f(x) + g(x)$ = (101) + (111) = (010) $\Rightarrow x$

Mathematical Background: Finite Field Arithmetic

- Multiplication:
  - Multiply two polynomials together. If the resulting polynomial has degree greater than n-1, then the polynomial is reduced modulo some *irreducible polynomial* $m(x)$ of degree n.
  - Irreducible polynomial $m(x)$: a polynomial cannot be expressed as a product of two polynomials, both with degree smaller than that of $m(x)$.
  - Irreducible polynomials of degree 3: $(x^3 + x^2 + 1)$ and $(x^3 + x + 1)$
  - $f(x) \cdot g(x) = (x^2 + 1) \cdot (x^2 + x + 1) \bmod m(x)$ $= (x^4 + x^3 + x^2) + (x^2 + x + 1) \bmod m(x)$

    $= x^4 + x^3 + x + 1 \bmod m(x)$

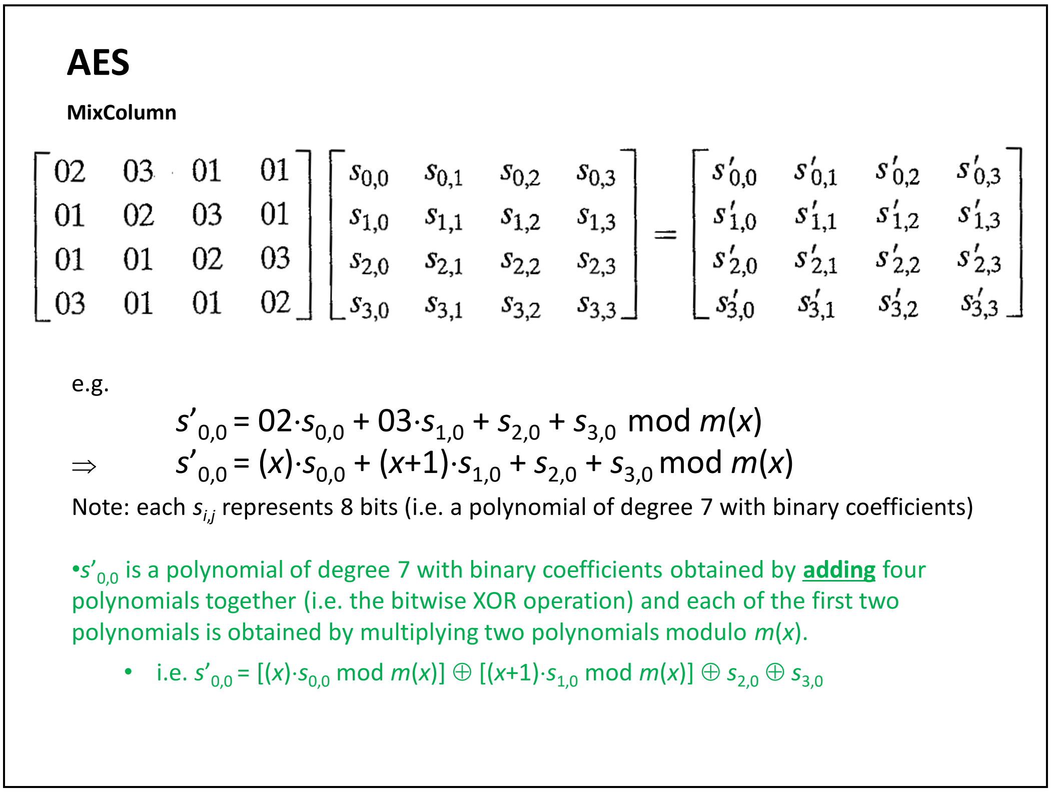
    take $m(x) = (x^3 + x + 1)$ as the irreducible polynomial, we have

    $f(x) \cdot g(x) = x^4 + x^3 + x + 1 \bmod (x^3 + x + 1)$ $= (x + 1)(x^3 + x + 1) + (x^2 + x) \bmod (x^3 + x + 1)$

    $= x^2 + x$
  - **Represent each element in GF(2^3) by a 3-bit value:** {000, 001, 010, 011, 100, 101, 110, 111}
  - $f(x) \cdot g(x) = (101) \cdot (111) = (110)$

- AES uses arithmetic in the finite field GF(2^8) with the irreducible polynomial

  $m(x) = x^8 + x^4 + x^3 + x + 1$



AES

MixColumn

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

e.g.

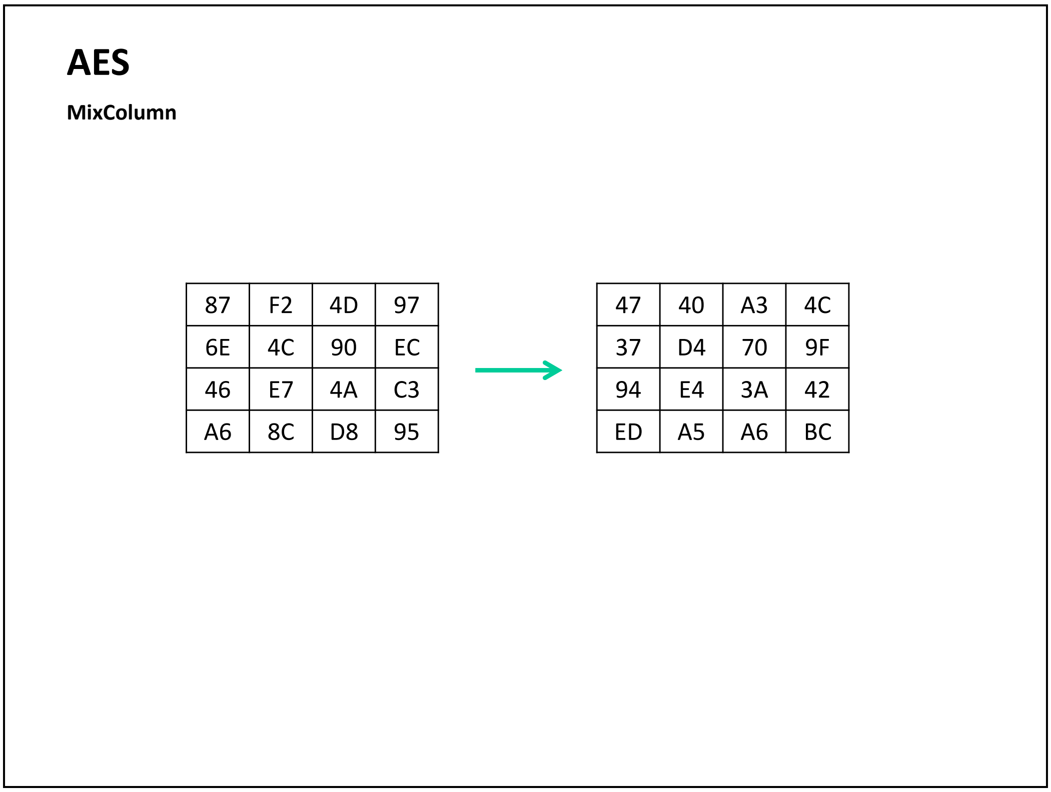
$$s'_{0,0} = 02 \cdot s_{0,0} + 03 \cdot s_{1,0} + s_{2,0} + s_{3,0} \bmod m(x)$$

$$\Rightarrow \quad s'_{0,0} = (x) \cdot s_{0,0} + (x+1) \cdot s_{1,0} + s_{2,0} + s_{3,0} \bmod m(x)$$

Note: each $s_{i,j}$ represents 8 bits (i.e. a polynomial of degree 7 with binary coefficients)

- $s'_{0,0}$ is a polynomial of degree 7 with binary coefficients obtained by **adding** four polynomials together (i.e. the bitwise XOR operation) and each of the first two polynomials is obtained by multiplying two polynomials modulo $m(x)$.
  - i.e. $s'_{0,0} = [(x) \cdot s_{0,0} \bmod m(x)] \oplus [(x+1) \cdot s_{1,0} \bmod m(x)] \oplus s_{2,0} \oplus s_{3,0}$

The mixcolumn value is either x+1 (11, 3) x (10,2), or 1 (01, 1)



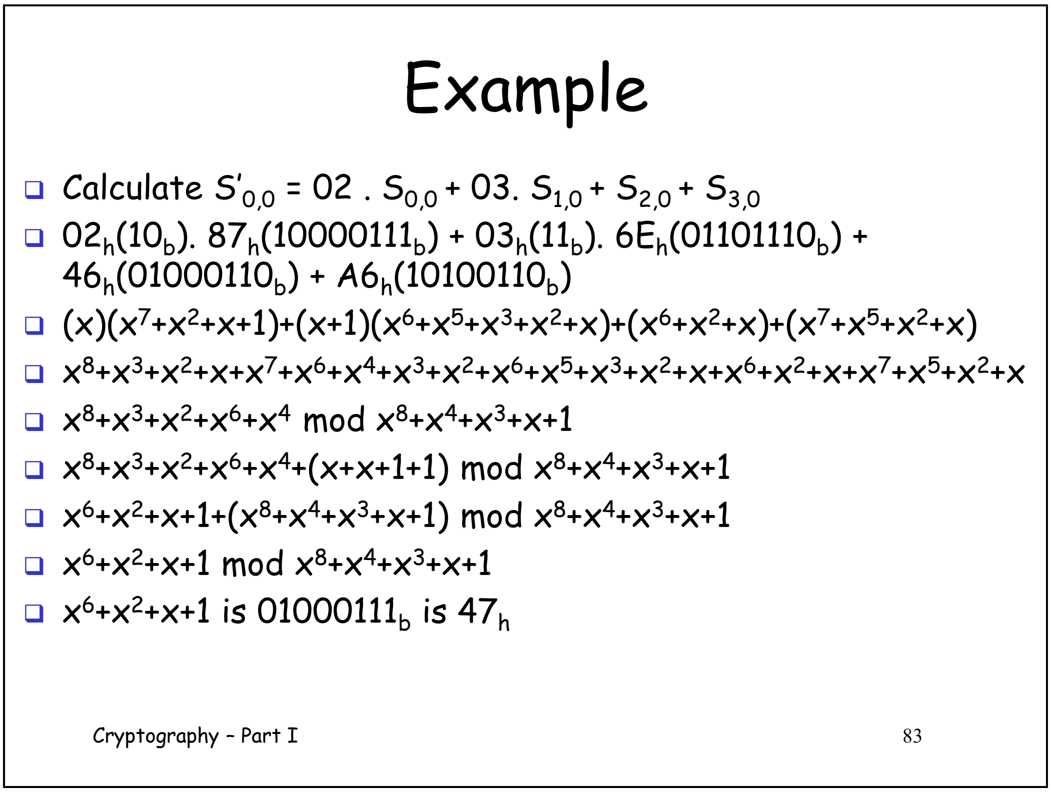
AES

MixColumn

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

→

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC



Example

- Calculate $S'_{0,0}$ = 02 . $S_{0,0}$ + 03. $S_{1,0}$ + $S_{2,0}$ + $S_{3,0}$
- $02_h(10_b)$. $87_h(10000111_b)$ + $03_h(11_b)$. $6E_h(01101110_b)$ + $46_h(01000110_b)$ + $A6_h(10100110_b)$
- $(x)(x^7+x^2+x+1)+(x+1)(x^6+x^5+x^3+x^2+x)+(x^6+x^2+x)+(x^7+x^5+x^2+x)$
- $x^8+x^3+x^2+x+x^7+x^6+x^4+x^3+x^2+x^6+x^5+x^3+x^2+x+x^6+x^2+x+x^7+x^5+x^2+x$
- $x^8+x^3+x^2+x^6+x^4$ mod $x^8+x^4+x^3+x+1$
- $x^8+x^3+x^2+x^6+x^4+(x+x+1+1)$ mod $x^8+x^4+x^3+x+1$
- $x^6+x^2+x+1+(x^8+x^4+x^3+x+1)$ mod $x^8+x^4+x^3+x+1$
- x^6+x^2+x+1 mod $x^8+x^4+x^3+x+1$
- x^6+x^2+x+1 is 01000111_b is 47_h

A different example (from a different matrix state)

S0,0=3E

S1,0=1C

S2,0=22

S3,0=C0

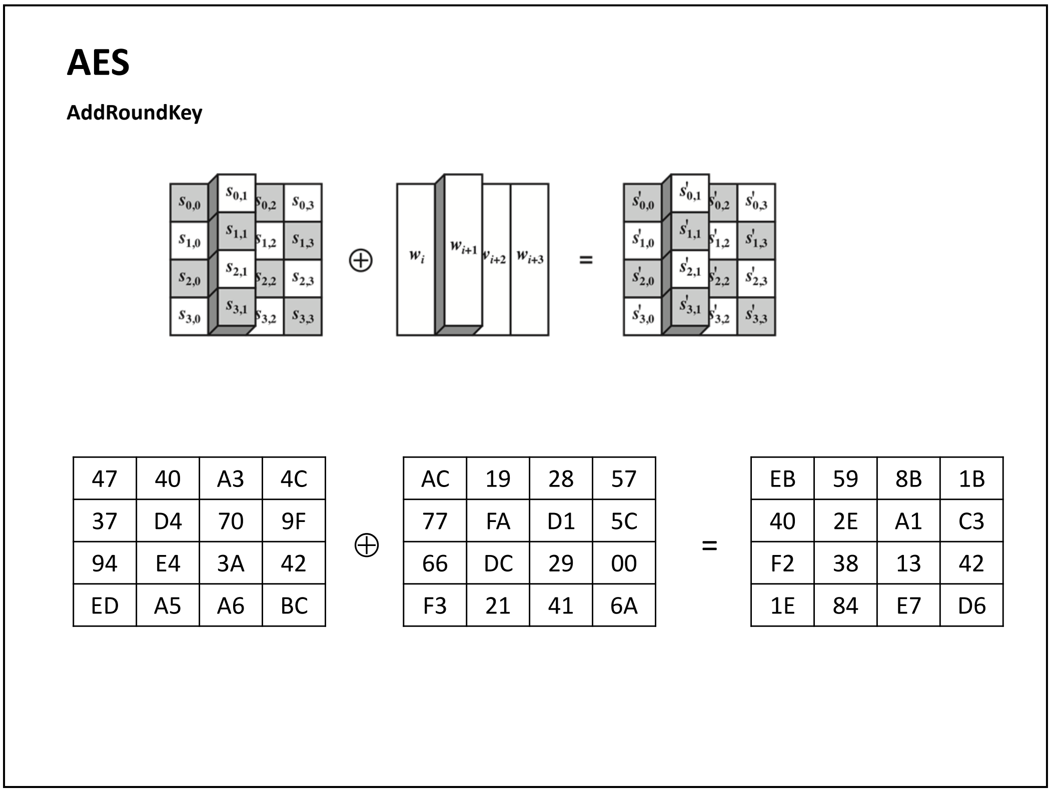
Calculate S3',0= 03.3E(00111110)+01.1C(00011100)+01.22(00100010)+02.C0(11000000)

= (x+1)(x5+x4+x3+x2+x)+(x4+x3+x2)+(x5+x)+x.(x7+x6) = x6+x5+x4+x3+x2+x5+x4+x3+x2+x +x4+x3+x2+x5+x+x8+x7

= x6+x4+x3+x2+x5+x8+x7 mod x8 + x4 + x3 + x + 1

= x8+x7+x6+x5+x4+x3+x2 +1+1+x+x mod x8 + x4 + x3 + x + 1

= (x8+x4+x3+x+1)+x7+x6+x5+x2 +x+1 mod x8 + x4 + x3 + x + 1 = x7+x6+x5+x2 +x+1 =11100111 = E7



AES

AddRoundKey

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

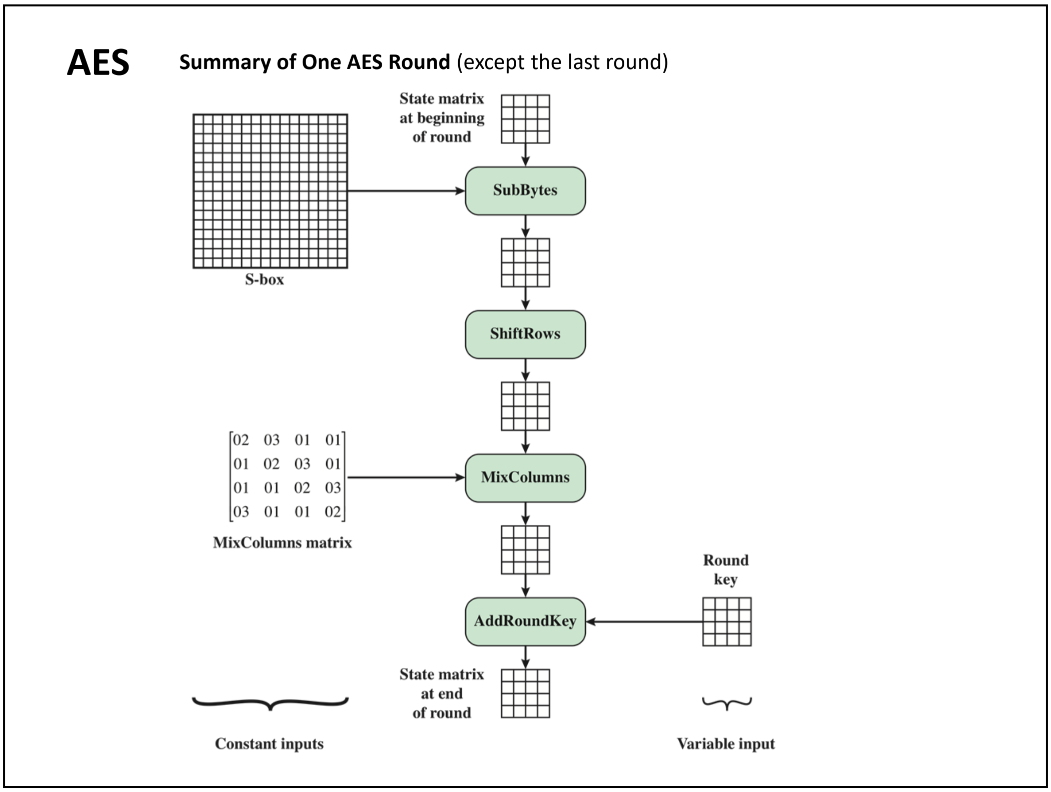
$\oplus$

AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A

=

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D6

Finally we just XOR the round key (which we also divided into a matrix) with the data matrix.



AES

Summary of One AES Round (except the last round)

State matrix at beginning of round

S-box

SubBytes

ShiftRows

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}$$

MixColumns matrix

MixColumns

Round key

AddRoundKey

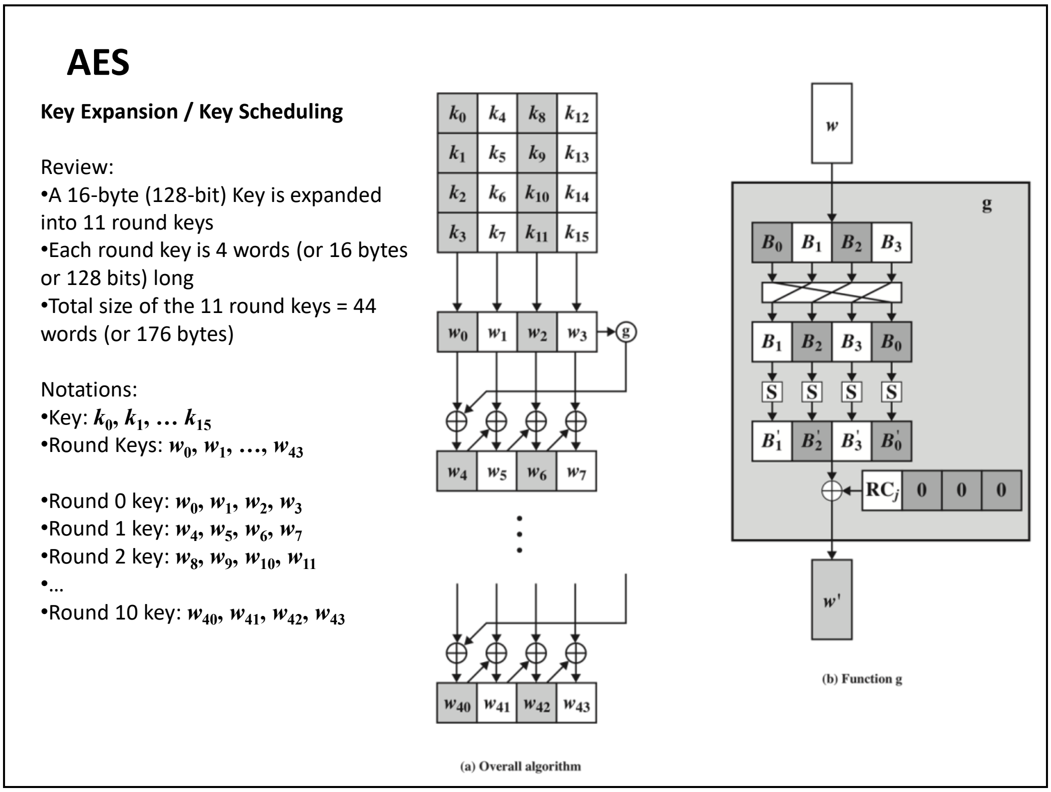
State matrix at end of round

Constant inputs

Variable input

Summary of what happens during encryption round.

Implementation is easy – simple binary operations (why it works, or design not so easy).



AES

Key Expansion / Key Scheduling

Review:
- A 16-byte (128-bit) Key is expanded into 11 round keys
- Each round key is 4 words (or 16 bytes or 128 bits) long
- Total size of the 11 round keys = 44 words (or 176 bytes)

Notations:
- Key: $k_0, k_1, \ldots k_{15}$
- Round Keys: $w_0, w_1, \ldots, w_{43}$

- Round 0 key: w_0, w_1, w_2, w_3
- Round 1 key: w_4, w_5, w_6, w_7
- Round 2 key: w_8, w_9, w_{10}, w_{11}
- …
- Round 10 key: $w_{40}, w_{41}, w_{42}, w_{43}$

k_0 k_4 k_8 k_{12}
k_1 k_5 k_9 k_{13}
k_2 k_6 k_{10} k_{14}
k_3 k_7 k_{11} k_{15}
w_0 w_1 w_2 w_3
g
w_4 w_5 w_6 w_7
w_{40} w_{41} w_{42} w_{43}
w
g
B_0 B_1 B_2 B_3
B_1 B_2 B_3 B_0
S S S S
B'_1 B'_2 B'_3 B'_0
RC_j 0 0 0
w'

(a) Overall algorithm

(b) Function g

From slides 86 and 87 you only need to know that we generate a round key from the single AES key.

g next slide

Details for reference only.

AES

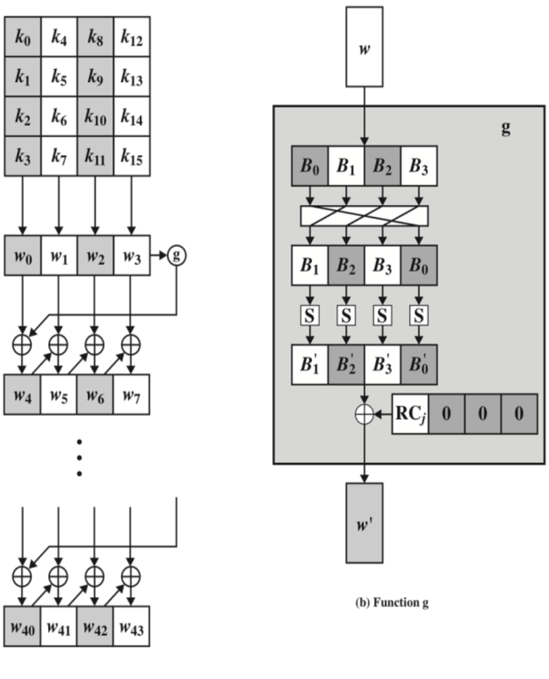
Key Expansion / Key Scheduling

Summary:

- The 16-byte key is copied into the first four words for Round 0 key
- i.e. the key is used **directly** to do the AddRoundKey at the initial transformation

- Each subsequent word w[i] in a round key depends on the immediately preceding word w[i-1], and the word four positions back, w[i-4]
  - in three out of the four cases a simple XOR is used, e.g. w_5, w_6, and w_7
  - for a word whose position in the ***w*** array is a multiple of 4, a more complex function ***g*** is used

- RC stands for Round Constant

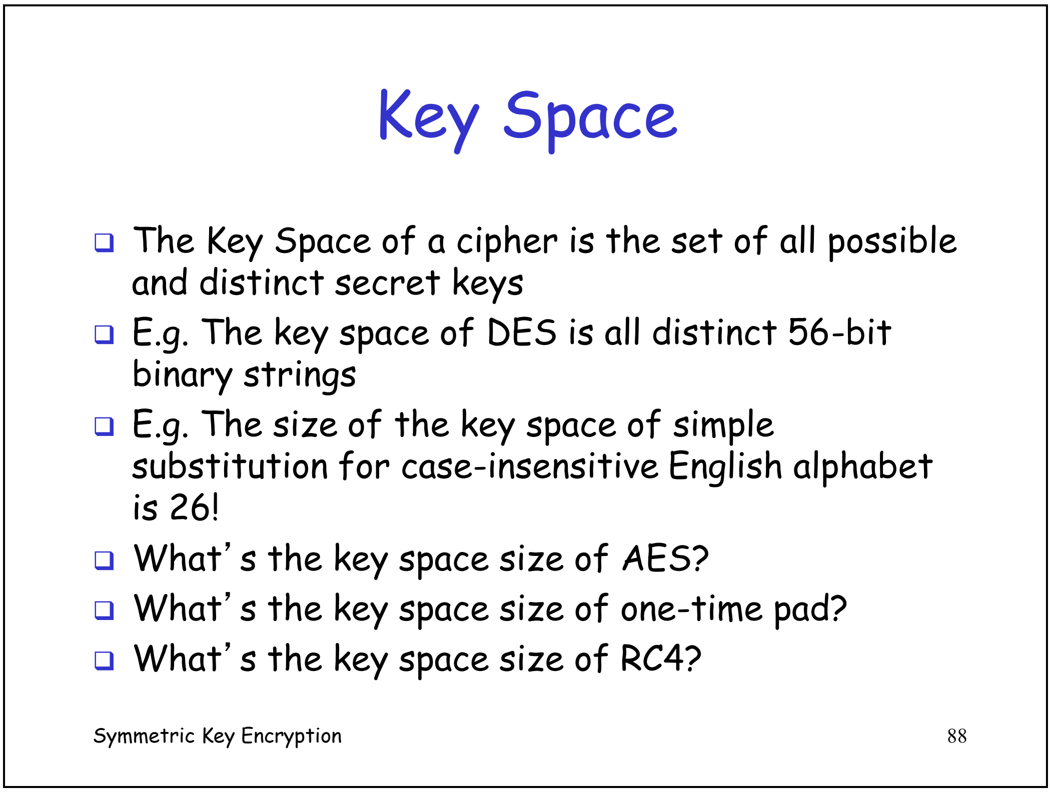


k_0 k_4 k_8 k_{12}
k_1 k_5 k_9 k_{13}
k_2 k_6 k_{10} k_{14}
k_3 k_7 k_{11} k_{15}
w_0 w_1 w_2 w_3
g
w_4 w_5 w_6 w_7
w_{40} w_{41} w_{42} w_{43}
w
g
B_0 B_1 B_2 B_3
B_1 B_2 B_3 B_0
S S S S
B'_1 B'_2 B'_3 B'_0
RC_j 0 0 0
w'

(b) Function g

(a) Overall algorithm

RCj is a constant.



Key Space

- The Key Space of a cipher is the set of all possible and distinct secret keys
- E.g. The key space of DES is all distinct 56-bit binary strings
- E.g. The size of the key space of simple substitution for case-insensitive English alphabet is 26!
- What's the key space size of AES?
- What's the key space size of one-time pad?
- What's the key space size of RC4?

Study this slide – you need to know what the keyspace is, how to calculate it, and how much exhaustive search attempts are required for a given keyspace (see slide 52)

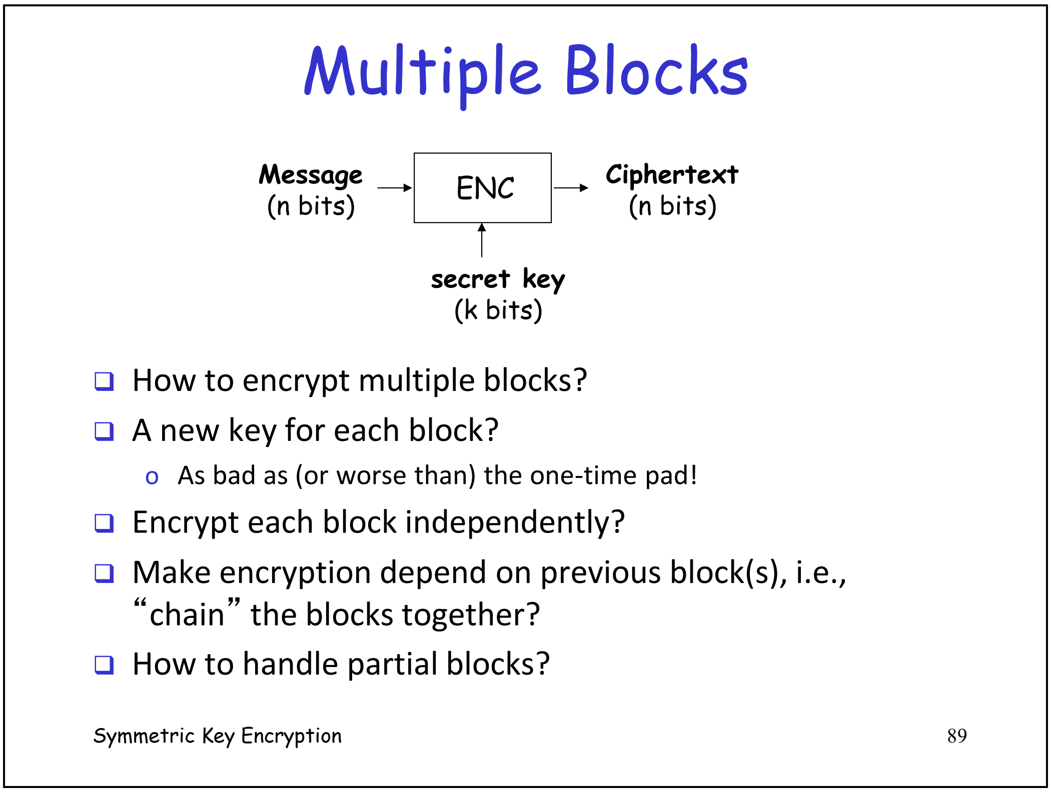
AES 128-256 bits

So searching for DES? 2^55

So searching for AES (128-bit key version)? 2^127

One time pad – key is size of the message – keyspace is 2^keysize

RC4 40-2048 bits



Multiple Blocks

Message
(n bits)

ENC

Ciphertext
(n bits)

secret key
(k bits)

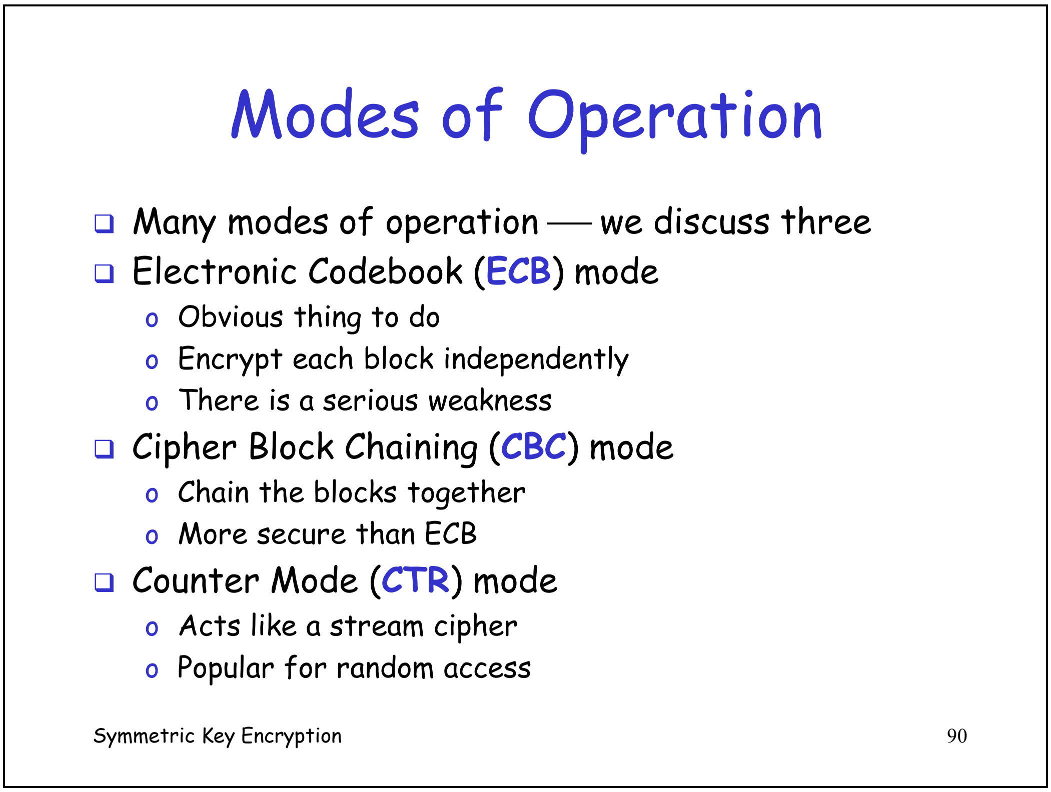
- ❑ How to encrypt multiple blocks?
- ❑ A new key for each block?
  - o As bad as (or worse than) the one-time pad!
- ❑ Encrypt each block independently?
- ❑ Make encryption depend on previous block(s), i.e., “chain” the blocks together?
- ❑ How to handle partial blocks?

Study slide

A block cipher only encrypts a fixed block – not like a steam cipher that just keeps generating keystream.

So we need ways to encrypt multiple block if the message is larger than the block size.

We also need a padding method if out message is not a multiple of the block size.

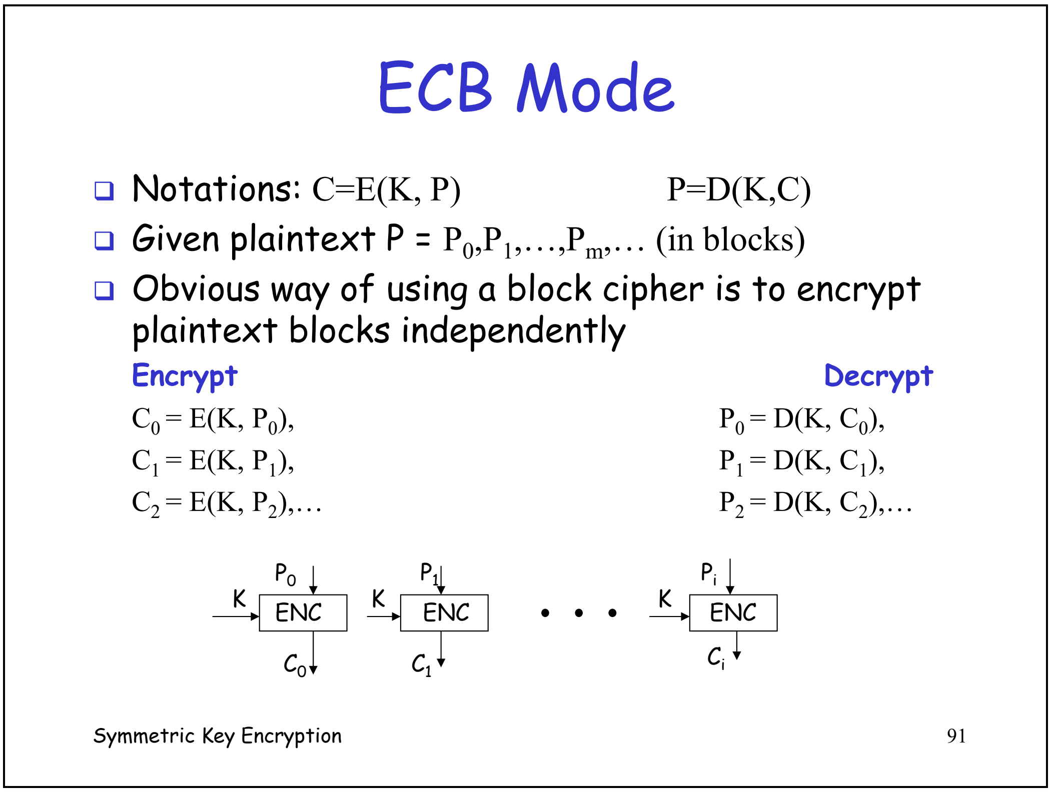


Modes of Operation

- Many modes of operation — we discuss three
- Electronic Codebook (**ECB**) mode
  - Obvious thing to do
  - Encrypt each block independently
  - There is a serious weakness
- Cipher Block Chaining (**CBC**) mode
  - Chain the blocks together
  - More secure than ECB
- Counter Mode (**CTR**) mode
  - Acts like a stream cipher
  - Popular for random access

You need to know the three modes of operation well (plus CFB) (understand what is going on – but you do not need to memorise what is going on)

Be comfortable with how data is encrypted and decrypted.



ECB Mode

- Notations: C=E(K, P) P=D(K,C)
- Given plaintext P = $P_0, P_1, \ldots, P_m, \ldots$ (in blocks)
- Obvious way of using a block cipher is to encrypt plaintext blocks independently

Encrypt

$C_0 = E(K, P_0)$,
$C_1 = E(K, P_1)$,
$C_2 = E(K, P_2), \ldots$

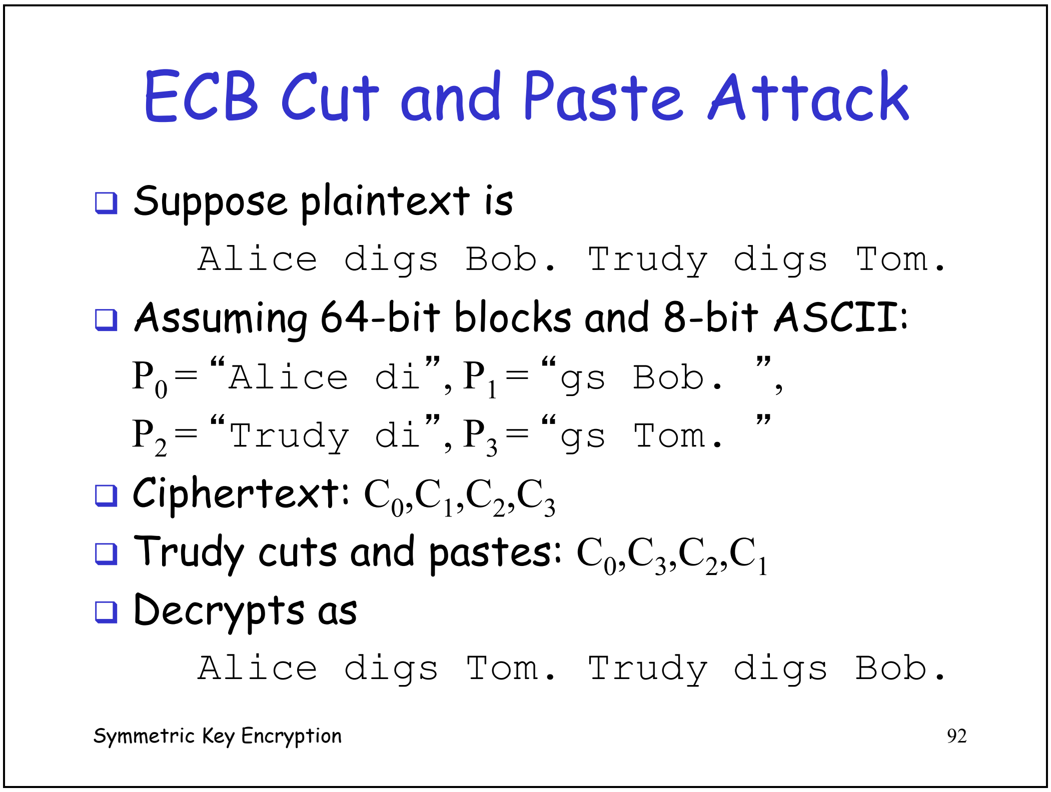
Decrypt

$P_0 = D(K, C_0)$,
$P_1 = D(K, C_1)$,
$P_2 = D(K, C_2), \ldots$

P_0
K
ENC
C_0
P_1
K
ENC
C_1
• • •
P_i
K
ENC
C_i

What are the problems?

Cut and paste
Can see if Pi=Pj



ECB Cut and Paste Attack

- Suppose plaintext is

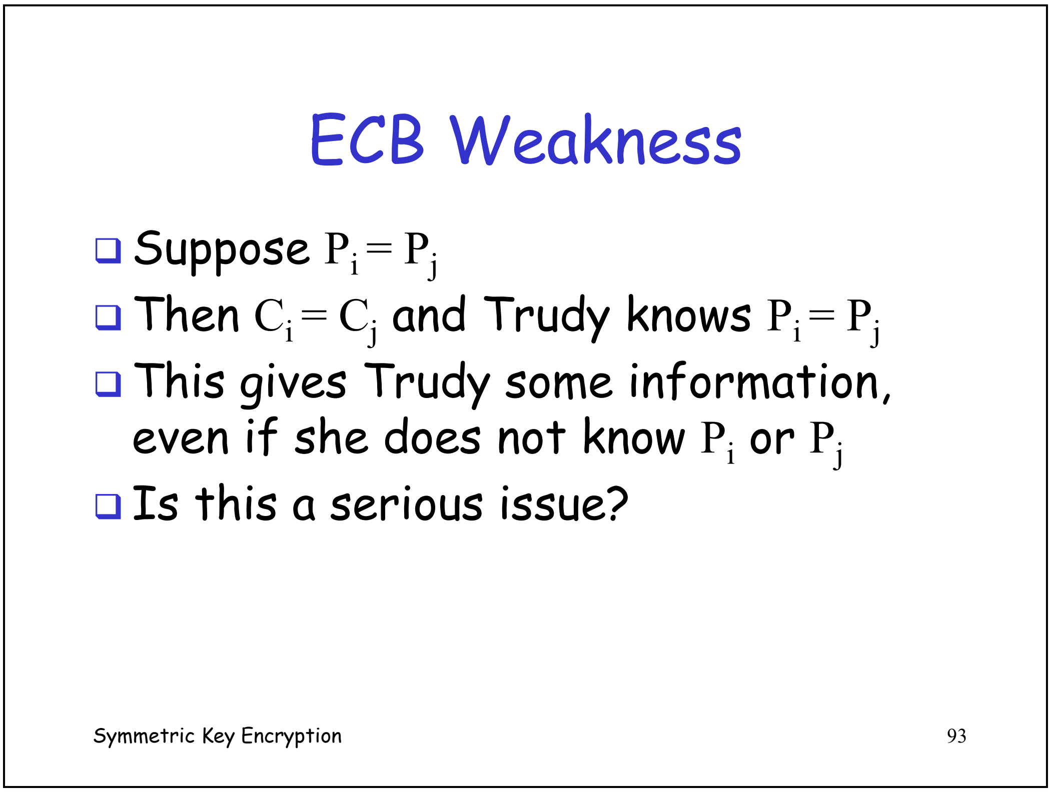
  `Alice digs Bob. Trudy digs Tom.`
- Assuming 64-bit blocks and 8-bit ASCII:

  $P_0 =$ “`Alice di`”, $P_1 =$ “`gs Bob.  `”,

  $P_2 =$ “`Trudy di`”, $P_3 =$ “`gs Tom.  `”
- Ciphertext: C_0,C_1,C_2,C_3
- Trudy cuts and pastes: C_0,C_3,C_2,C_1
- Decrypts as

  `Alice digs Tom. Trudy digs Bob.`

In ECB all the plaintext blocks are encrypted independently – this means you can in theory delete some of the ciphertext blocks, or move some of the ciphertext blocks around.



ECB Weakness

- Suppose $P_i = P_j$
- Then $C_i = C_j$ and Trudy knows $P_i = P_j$
- This gives Trudy some information, even if she does not know P_i or P_j
- Is this a serious issue?

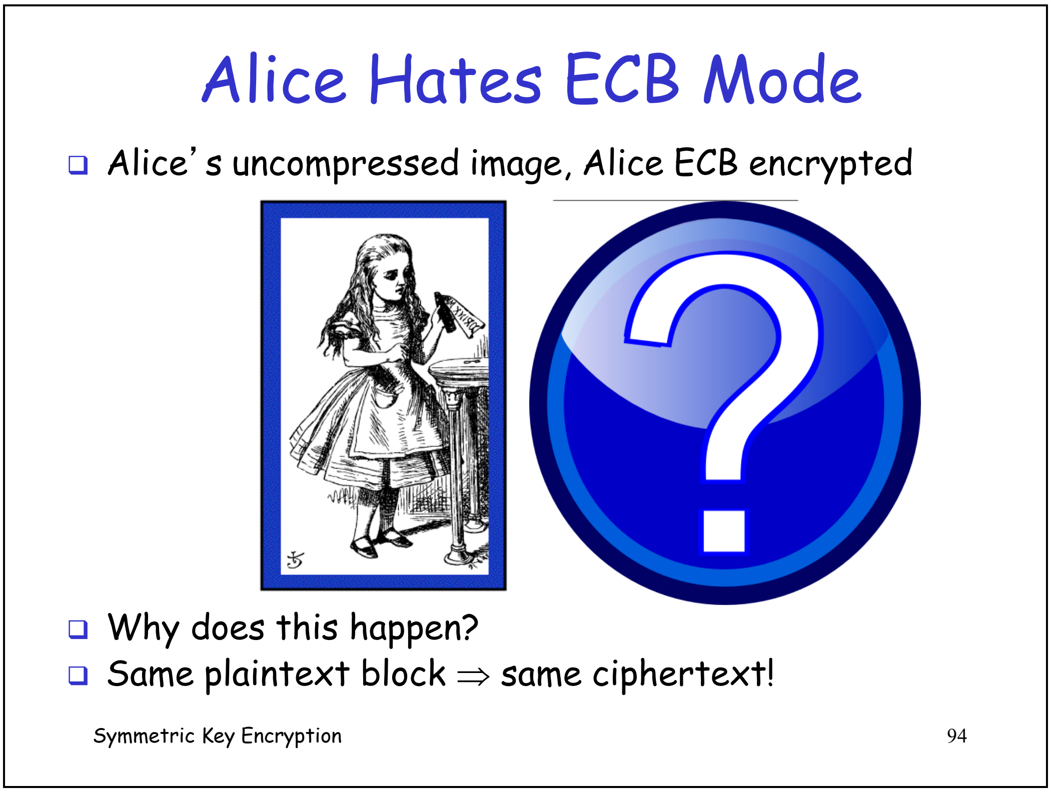
Symmetric Key Encryption 93

The second weakness is that even if the rest of the message is different each identical plaintext block encrypts to same ciphertext block.

This means if we see same ciphertext twice the same plaintext got sent twice…

This is now like a very very large substitution cipher! (with 2^n possible permutations where n is blocksize)….

Traffic analysis – can see the same messages being sent. What if message repeat?



Alice Hates ECB Mode

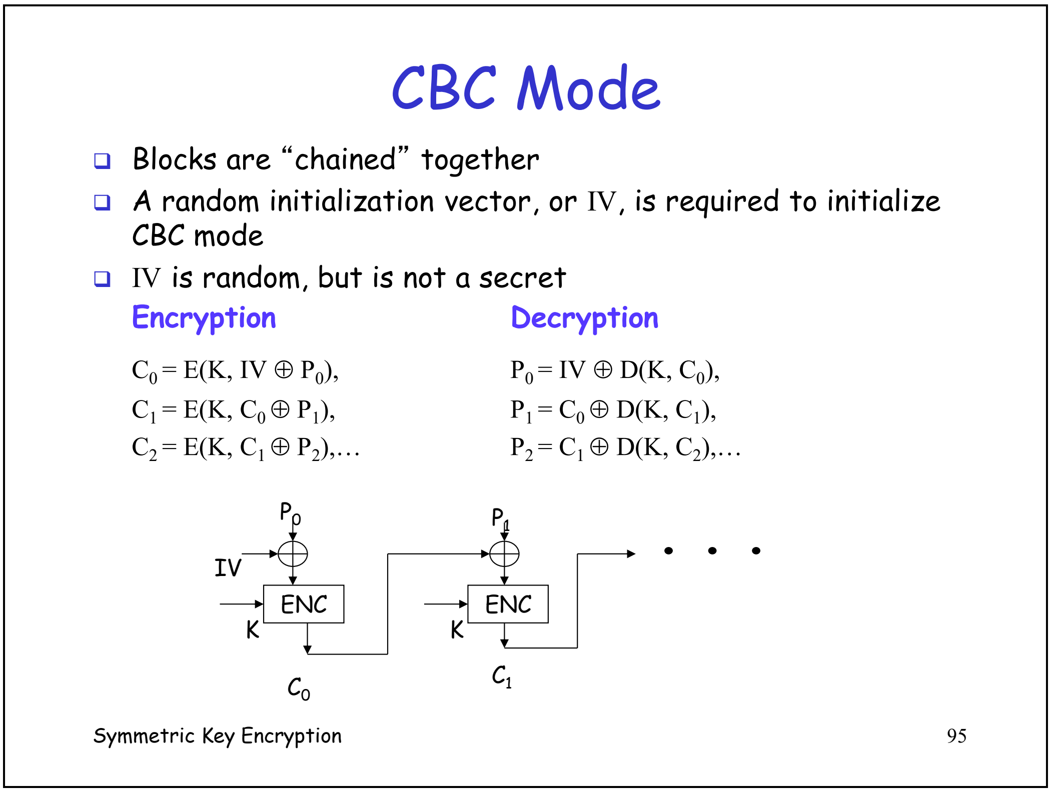
- Alice's uncompressed image, Alice ECB encrypted
- Why does this happen?
- Same plaintext block ⇒ same ciphertext!

Symmetric Key Encryption 94

This is an examples of that to a practical scenario – a picture encrypted in ECB mode appears to still expose most of the data.

This is a black and white picture – the same pixel combinations encrypt to same ciphertext….

ECB is essentially like a fancy substitution cipher with a much larger binary alphabet



CBC Mode

- Blocks are “chained” together
- A random initialization vector, or IV, is required to initialize CBC mode
- IV is random, but is not a secret

Encryption

$C_0 = E(K, IV \oplus P_0)$,
$C_1 = E(K, C_0 \oplus P_1)$,
$C_2 = E(K, C_1 \oplus P_2), \ldots$

Decryption

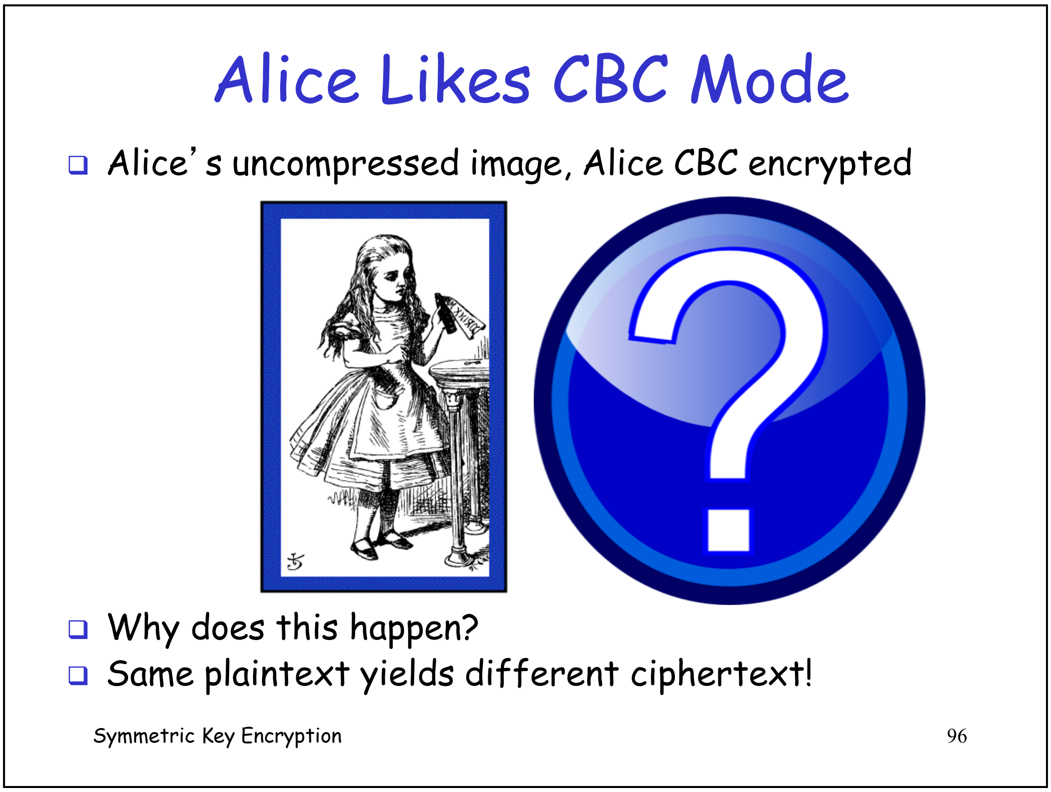
$P_0 = IV \oplus D(K, C_0)$,
$P_1 = C_0 \oplus D(K, C_1)$,
$P_2 = C_1 \oplus D(K, C_2), \ldots$

Cipher Block Chaining tries to address this. Here the ciphertext depends on all previous ciphertext. So even if we send the same plaintext block twice the cipher text block differs (as the set of ciphertext sent before since the beginning of the message differs).

We must still be careful if sending the same entire plaintext message, in such a case we must use a different IV for each message otherwise all the ciphertext will be the same.

If P0 changes in the messages what happens to all C blocks? All change.

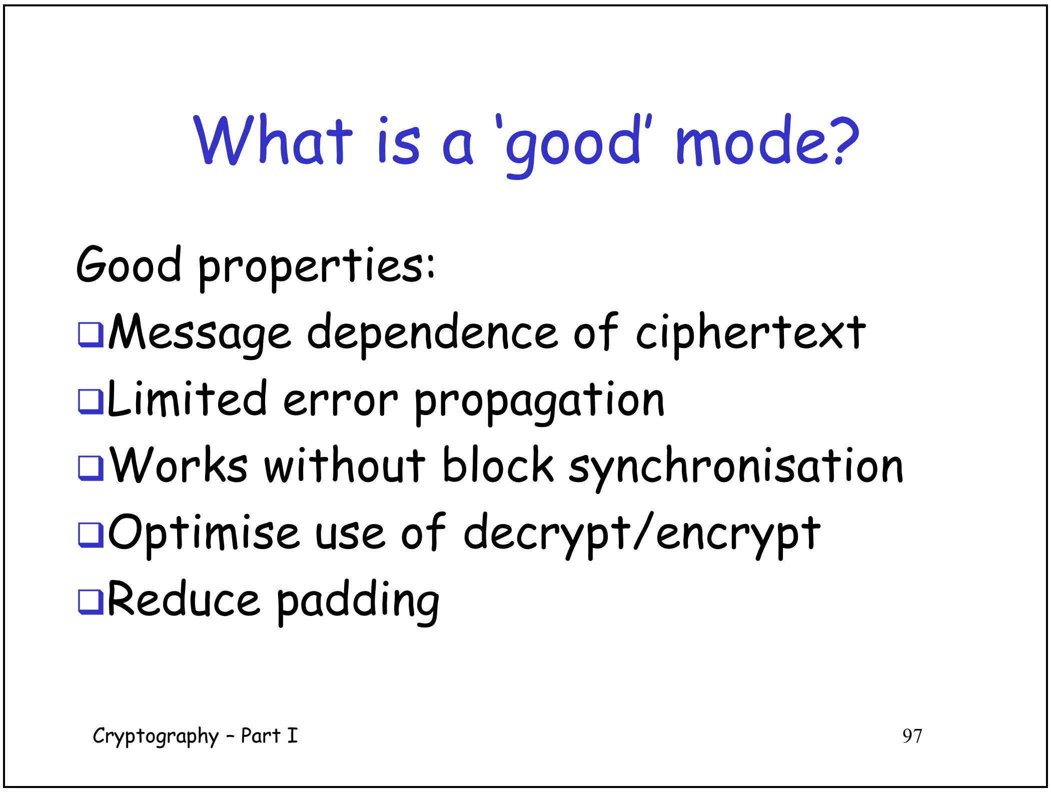
The implication is all C blocks are dependent on all previous P blocks.



Alice Likes CBC Mode

- Alice's uncompressed image, Alice CBC encrypted
- Why does this happen?
- Same plaintext yields different ciphertext!

As seen with the picture, the same plaintext now delivers much different ciphertext – and you can now longer see Alice



What is a 'good' mode?

Good properties:

- ❑Message dependence of ciphertext
- ❑Limited error propagation
- ❑Works without block synchronisation
- ❑Optimise use of decrypt/encrypt
- ❑Reduce padding

Study this slide…

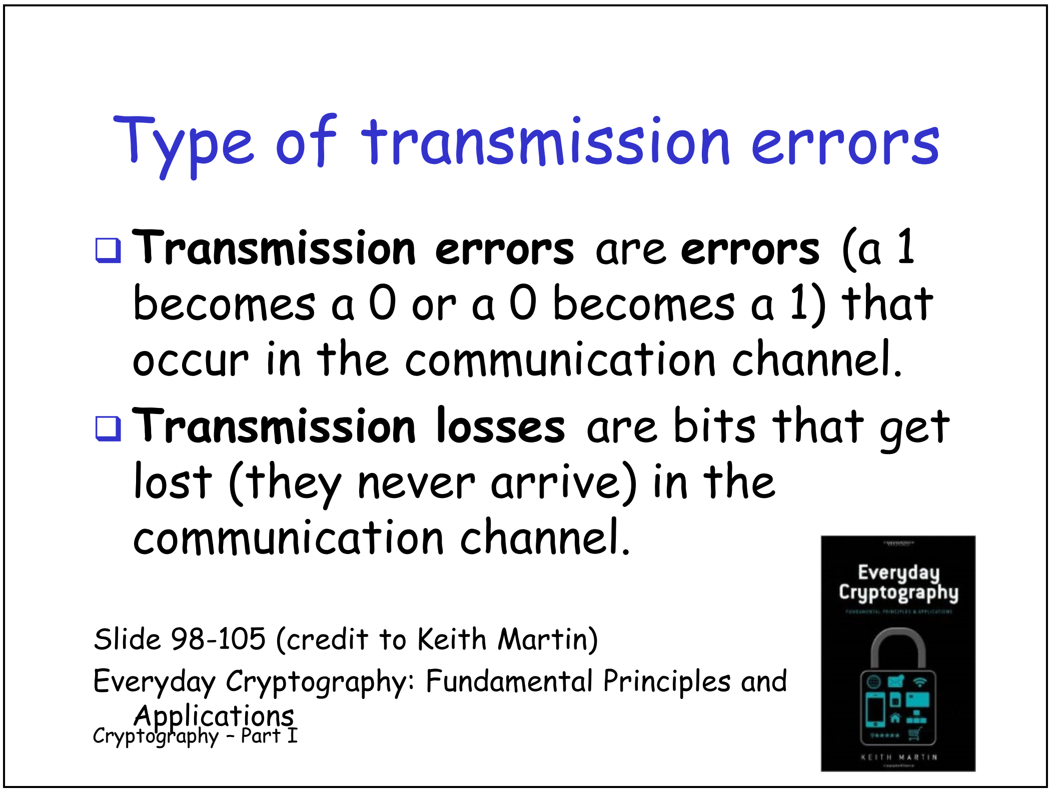
Message dependence means the ciphertext also depends on previous blocks of the message rather that only the input plaintext block (For example, ECB mode does not have good message dependence).

A mode of operation should be able to recover from errors (the error should not continue until the end of the message).

Does not need block synchronization between sender and receiver (if a block of the message goes missing, the impact on subsequent messages should be limited)

Optimise use of decrypt/encrypt (in some block ciphers implementing both decrypt a block and encrypt a block functionality takes more resources) – some block modes only require one of these to be implemented to encrypt or decrypt and entire message.

Ideally we do not want to generate too much ciphertext in comparison to our plaintext if possible (this is not always practical, especially with block ciphers – you need to enter a full block even if your message (or what is left of your message is smaller than the block).



Type of transmission errors

- **Transmission errors** are **errors** (a 1 becomes a 0 or a 0 becomes a 1) that occur in the communication channel.
- **Transmission losses** are bits that get lost (they never arrive) in the communication channel.

Slide 98-105 (credit to Keith Martin)

Everyday Cryptography: Fundamental Principles and Applications

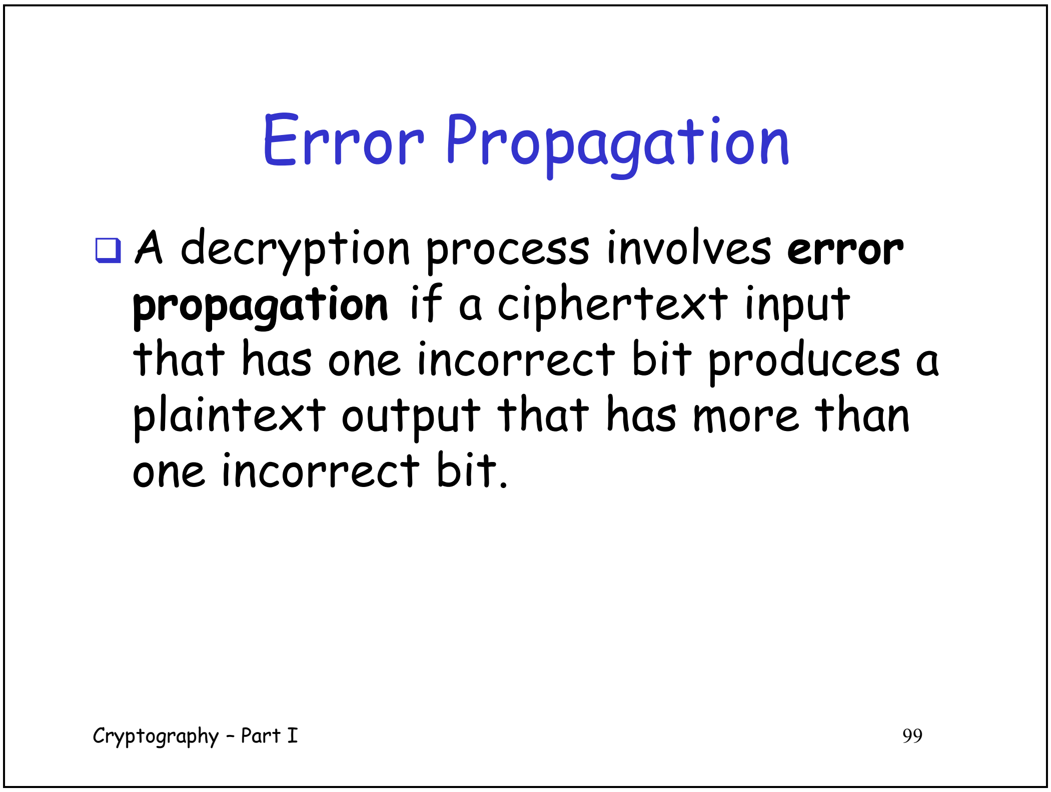
Cryptography – Part I

Everyday
Cryptography
KEITH MARTIN

Study this slide…

Transmission error means we receive the block but one or more bits are wrong.

Transmission losses basically means the block goes missing.



Error Propagation

- A decryption process involves **error propagation** if a ciphertext input that has one incorrect bit produces a plaintext output that has more than one incorrect bit.

Cryptography – Part I 99

Study this slide…

1.To what extent does error propagation occur in basic stream and block ciphers?

In a basic stream ciphers – errors in ciphertext map directly to errors in plaintext. This is due to the XOR used. Think – one bit of C XOR one bit of corresponding KS is P, then C' XOR the same KS is P' (where if X=1 then X'=0, if X=0 then X'=1)

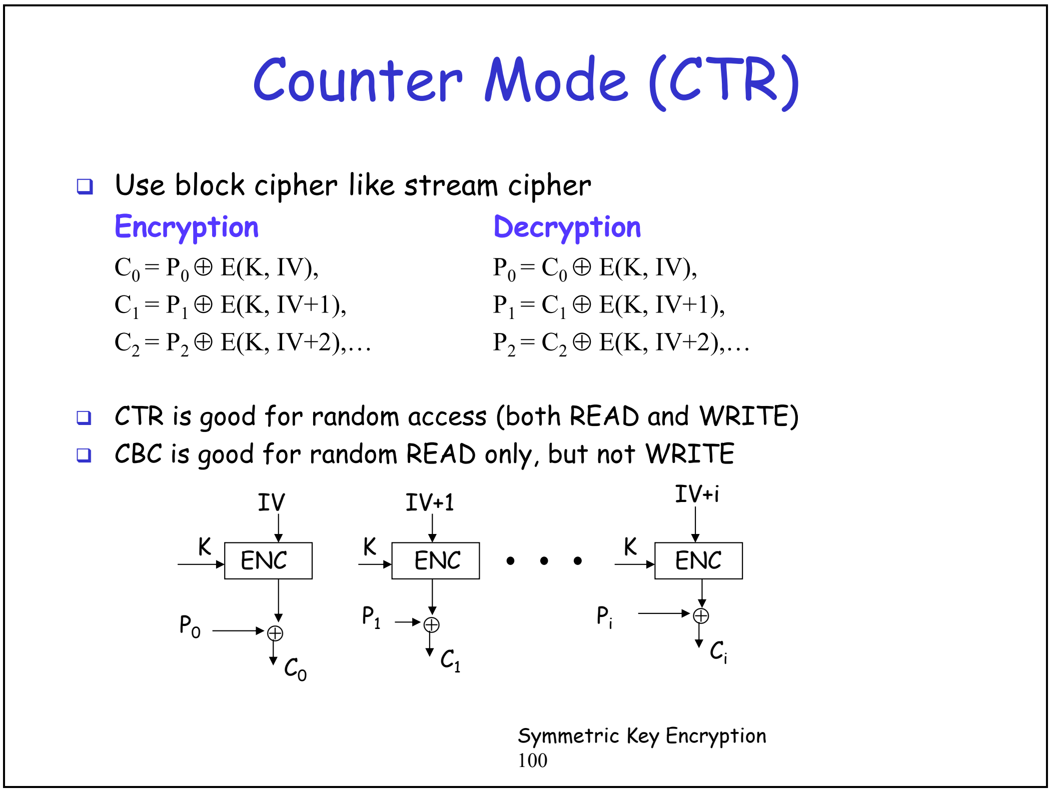
2. Does error propagation have anything to with error prevention or error correction?

No, we cannot prevent errors as such. We can however limit the effect is has on the received message.

A little bit connected – if we minimise the errors is might be possible to correct the received message. It is nearly impossible to recover from a major amount of errors.

3. Is error propagation a good thing?

No, if an error occurs early in the message that means the entire message is wrong.



Counter Mode (CTR)

- Use block cipher like stream cipher

Encryption	**Decryption**
$C_0 = P_0 \oplus E(K, IV)$,	$P_0 = C_0 \oplus E(K, IV)$,
$C_1 = P_1 \oplus E(K, IV+1)$,	$P_1 = C_1 \oplus E(K, IV+1)$,
$C_2 = P_2 \oplus E(K, IV+2),\ldots$	$P_2 = C_2 \oplus E(K, IV+2),\ldots$

- CTR is good for random access (both READ and WRITE)
- CBC is good for random READ only, but not WRITE

IV
IV+1
IV+i
K
ENC
K
ENC
• • •
K
ENC
P_0
P_1
P_i
C_0
C_1
C_i

Error? 1 bit

What do we mean by synchronisation? Receiver needs to know what block we have incoming.

CTR mode requires synchronisation between readers in terms of the counter. It overcomes ECB weakness as even with same plaintext block it will have different ciphertext as value of IV is different.

Does CBC need synchronisation? No, you just feed to received blocks into the decryption function there is no need to keep track of how many your received.

For error propagation – how would CTR perform if

1 bit is incorrect during transmission? There will be a one bit mistake in the decrypted plaintext.

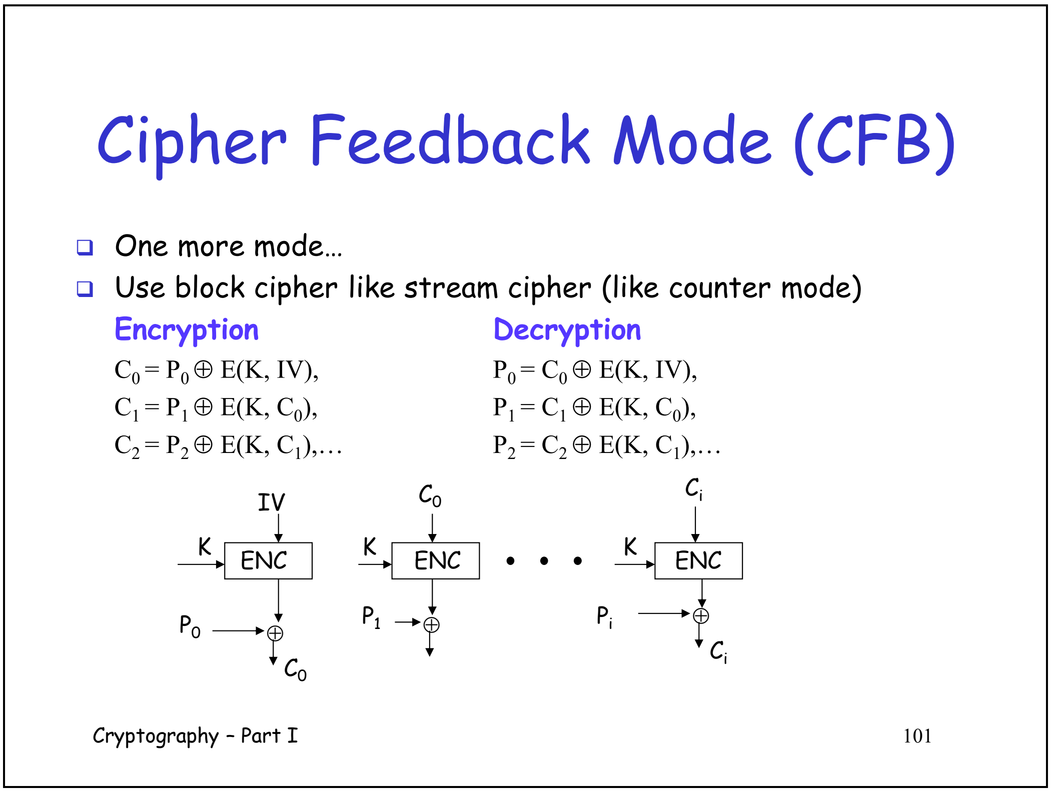
1 block is lost? The receiver IV counter would now be one behind and all data will decrypt incorrectly.

CTR mode is good if you store encrypted data. Consider that the IV is also the index of the stored ciphertext block. To read the block's plaintext you do E(Ivi, K) and XOR it to stored C. If you edit the block you can then simple XOR the new plaintext the E(Ivi, K) and store it.

Can you do the same with CBC?

Reading is easy, you take cipher block Ci-1 and use it to decrypt Ci, now you can read the data. However, what happens now if you change the plaintext? You can encrypt Ci again using Ci-1. Is that good enough?

No! Now you need to decrypt and encrypt all the remaining block as the change in Ci means all the rest of the blocks will change.



Cipher Feedback Mode (CFB)

- One more mode…
- Use block cipher like stream cipher (like counter mode)

Encryption

$C_0 = P_0 \oplus E(K, IV)$,
$C_1 = P_1 \oplus E(K, C_0)$,
$C_2 = P_2 \oplus E(K, C_1)$,…

Decryption

$P_0 = C_0 \oplus E(K, IV)$,
$P_1 = C_1 \oplus E(K, C_0)$,
$P_2 = C_2 \oplus E(K, C_1)$,…

Cryptography – Part I 101

Study this slide…

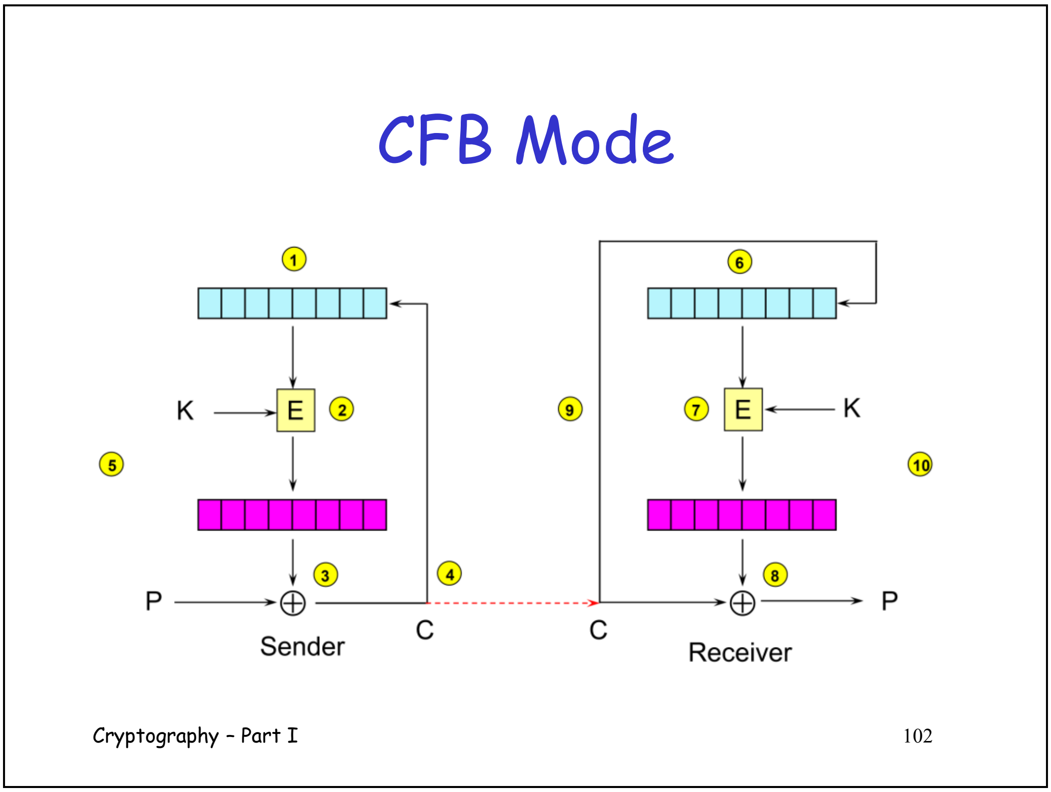
How is this similar to CBC?

There is an XOR operation, and the ciphertext Ci is used to calculate ciphertext Ci+1.

How is this different?

In CBC the plaintext is XORed with Ci and then encrypted to form Ci+1, this means means the receiver has to implemented a decrypt (decrypt received ciphertext Ci+1 and then XOR to previous Ci)

In CFB we use the block cipher to create keystream (similar to a stream cipher approach), this means both encryption and decryption of message requires only E() encrypt function.



CFB Mode

Cryptography – Part I 102

Study this slide…

Explain then ask about error – what happens if a 1 bit error occurs in C?

Ok lets see how CFB works – follow the numbers.

1.At the start Ci has been stored by the sender, at the same time Ci is also stored be the receiver (from previous round) in blue registers.

2.Sender encrypts (with block cipher) and key K to get keystream (purple block)

3.To generate Ci+1 the plaintext is XOR to the plaintext

4.Ci+1 is send, and also fed back and stored by the sender (blue register)

5. The sender is now finished and ready for next block to be sent.

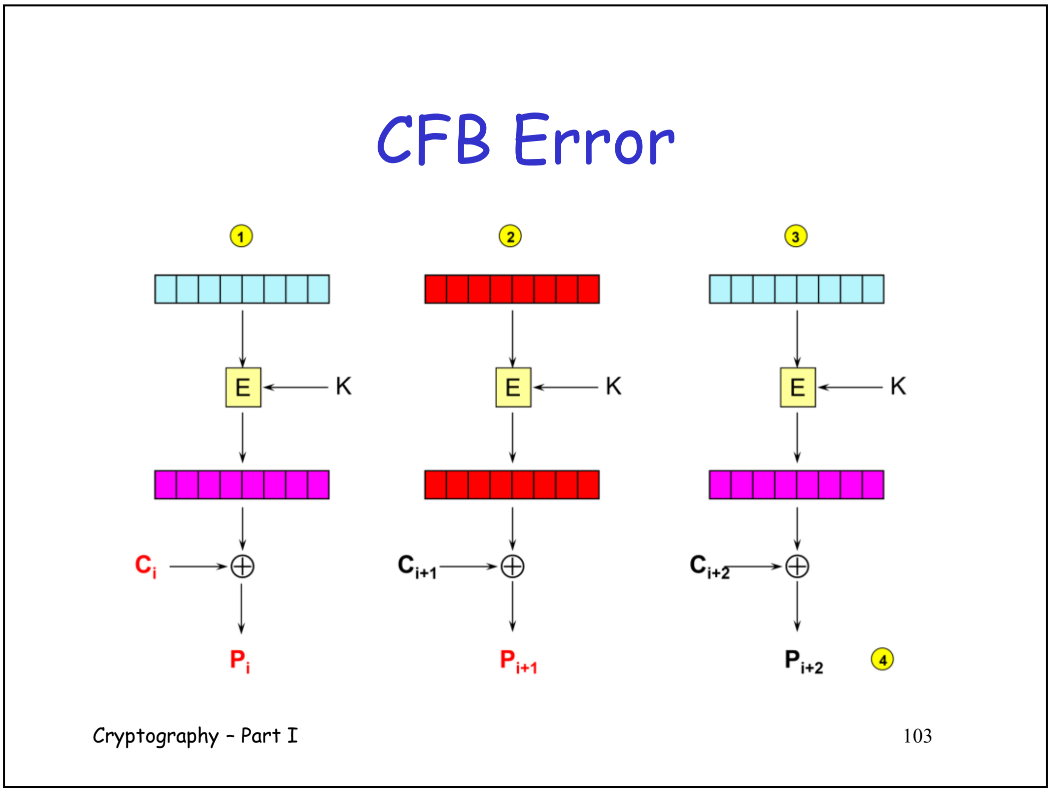
6.The receivers has Ci

7.He encrypts it with K to get keystream (purple). This keystream is the same as that used by sender (as both used Ci).

8.He XOR keystream to Ci+1 to get Pi+1.

9.Finally het stores Ci+1 for the next block.

10.Repeat the process….



CFB Error

Study this slide…

This picture is drawn from the receiver's perspective…

Ci has been sent (sender has use Ci-1 to generate it Ci has been stored for next block by the sender). There is a one bit error in Ci > lets call error block C'i.

The receiver receives C'i, and XOR E(Ci-1, K) to get Pi. As there is one bit error in C'i and we XOR with same KS that sender used we get one bit error in Pi.

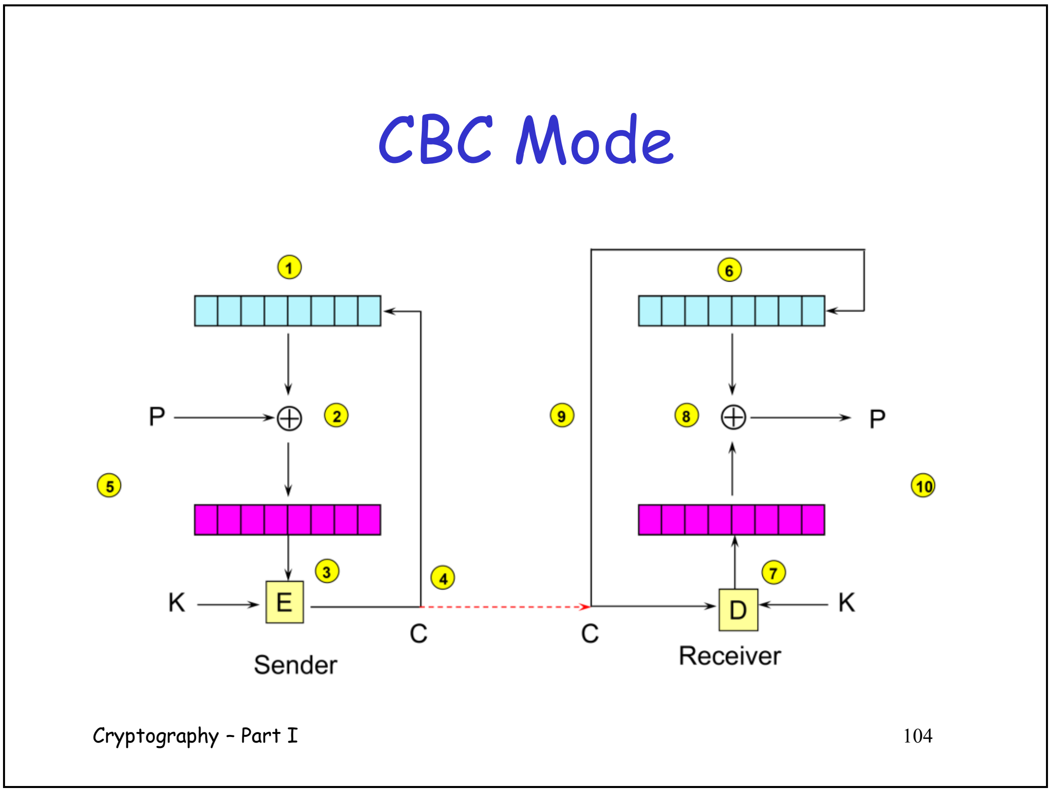
C'i is now stored for the next block received.

We now received Ci+1 (sender used Ci to generate) but we try to decrypt is with keystream KS=E(C'I,K). Remember what we said about a good block cipher – one bit in plaintext change should change at least 50% of ciphertext. So if input to E() is one bit different at receiver than sender then at least 50% of keystream is different. We XOR this to the Ci+1 and thus Pi+1 has lots of bit errors.

The receiver no stores Ci+1 for next block – it arrived without error, and is the same data the sender will use to generate Ci+2, therefore the error will not propagate any further and Pi+2 would be fine.

NOTE! For error propogation do not try to memorise the error propogation, you need to be able to understand and work things out from what you know if CFB (and other modes of operation). What if you get a question that does not ask about a 1-bit error?

When working out the error do what is comfortable for you – Prefer mathematic representation? Prefer drawing a picture?



CBC Mode

Study this slide…

Explain and ask about error.

1.At the start both sender and receiver has Ci-1 stored (in blue)

2.Sender calculates Pi XOR Ci-1

3.Then Ci= E(Pi XOR Ci-1)

4.Ci is sent (and also stored for the next block by sender)

5.Sender is now done with this block.

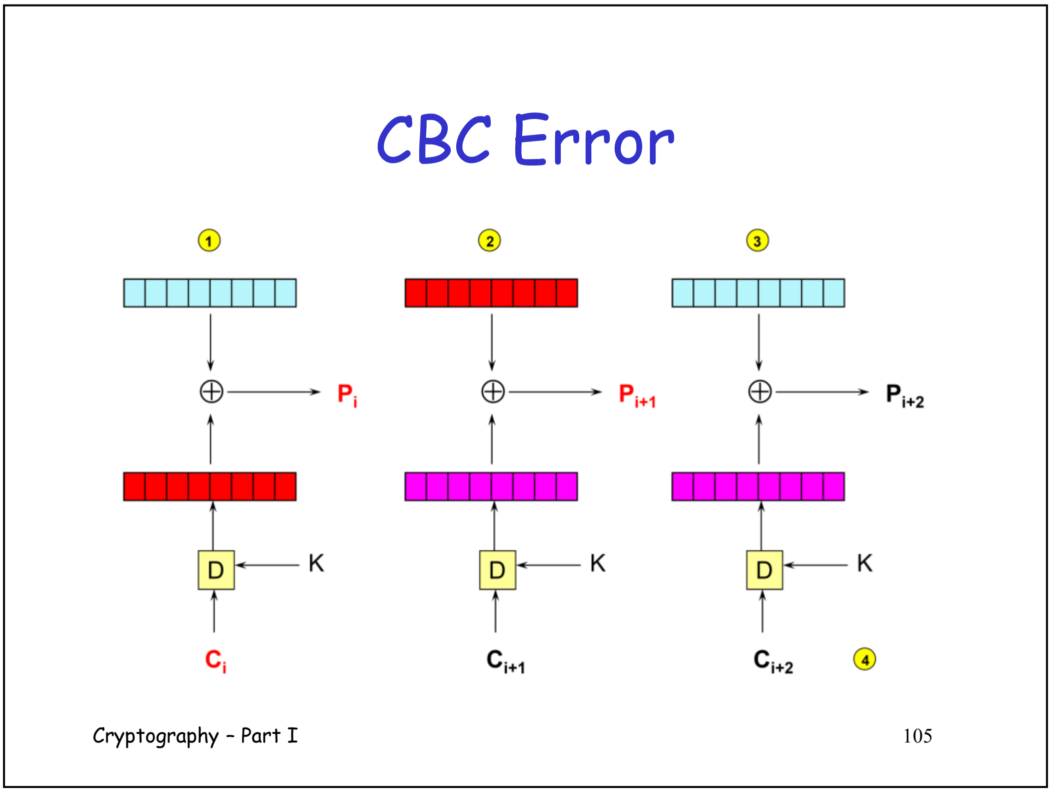
6.The receiver (remember) has stored Ci-1…

7.The receiver decrypts Ci (result is D(Ci) = Ci-1 XOR Pi)

8. D(Ci) XOR Ci-1 is therefore Pi

9.Receiver stores Ci for next round.

10.We repeat for next block.



CBC Error

Cryptography – Part I 105

Study this slide…

Once again from the receiver's perspective, and assuming a 1-bit error.

Give the quick notes supplementary material!!

The receiver receives C'i instead of Ci (one bit error).

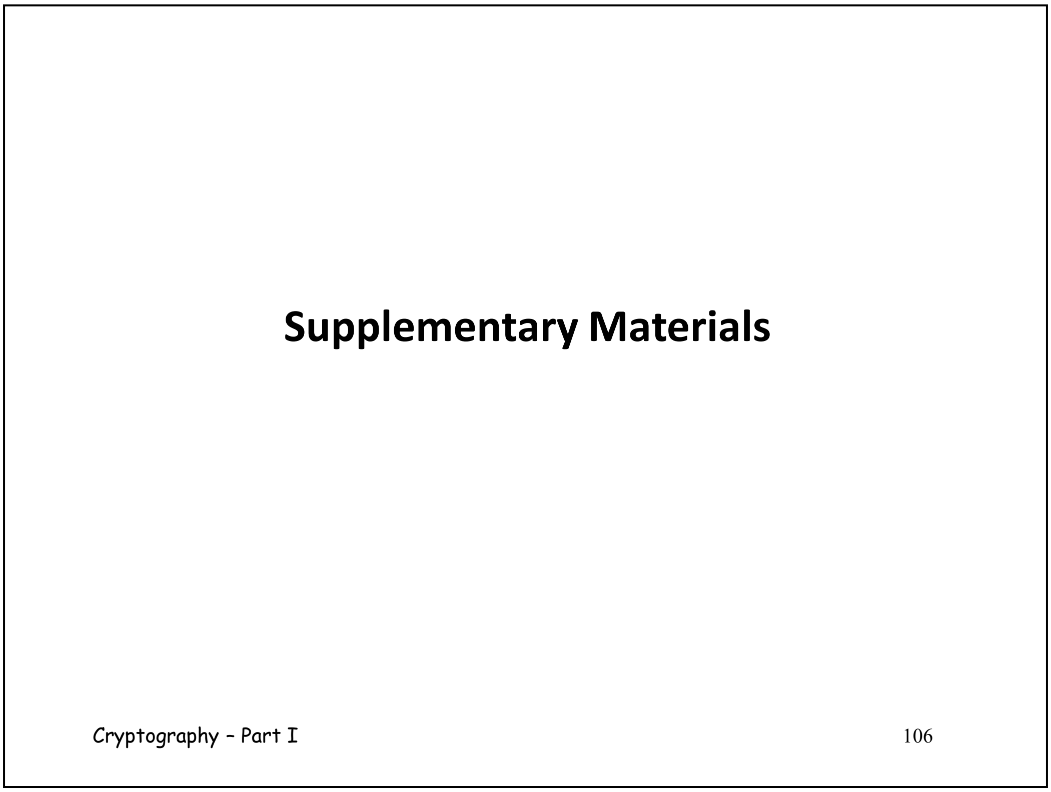
D(C'i,K) not equal Ci-1 XOR Pi (many bit errors as input to decrypt is one bit different than sender), so D(C'i,K) XOR Ci-1 not equal to Pi (Pi has lots of errors, due to decrypt function).

Receiver now stores C'i.

Receiver receives Ci+1 = E(Ci XOR Pi+1,K). D(Ci+1,K) = Ci XOR Pi+1

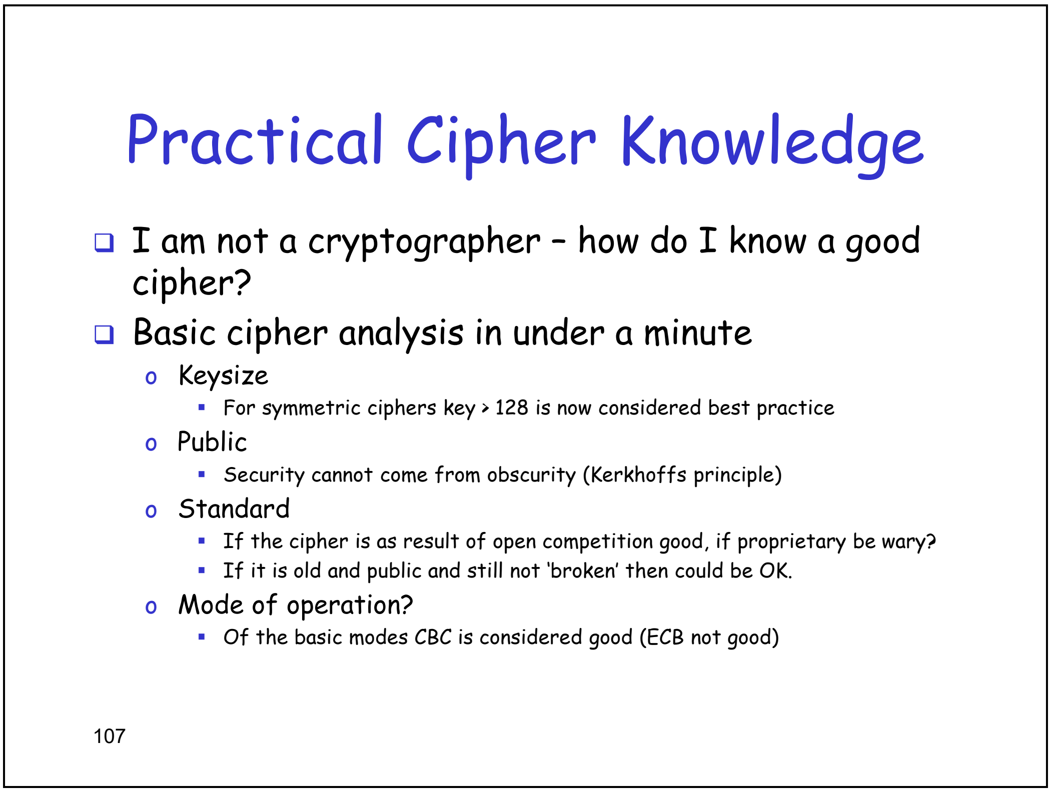
Ci XOR Pi+1 XOR C'I = P'i+1 (one bit error in Pi+1).

The next block is fine as both parties use correct Ci+1



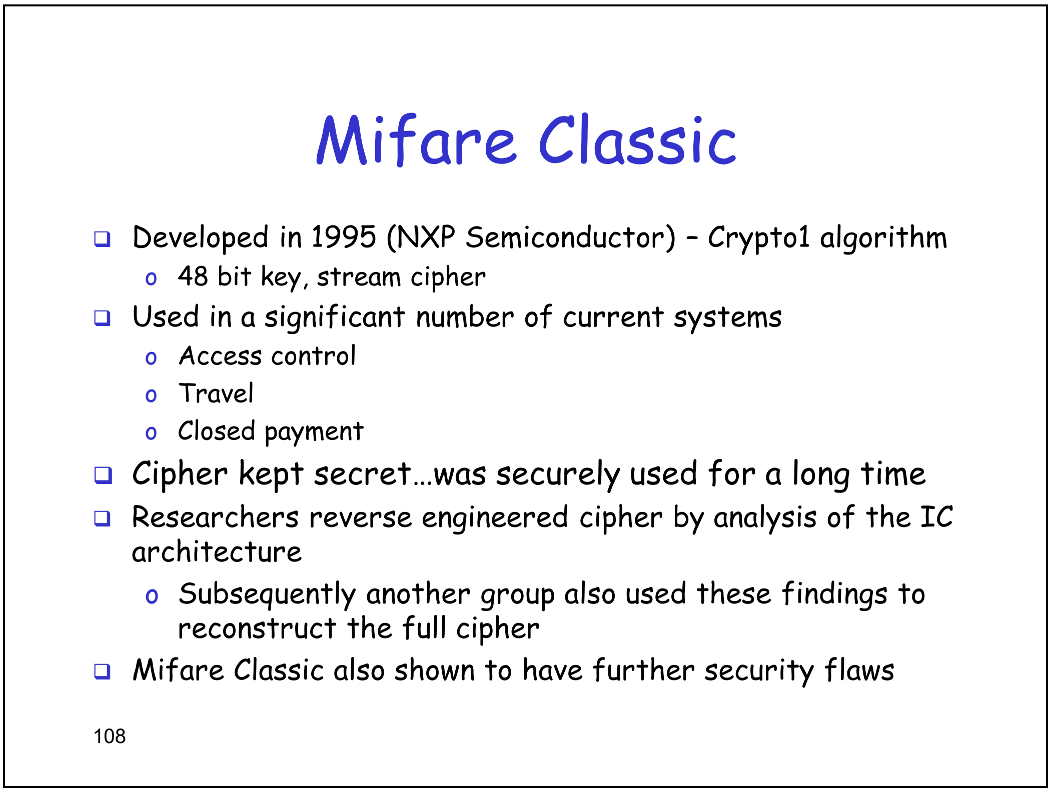
Supplementary Materials

Rest of slides only for reference



Practical Cipher Knowledge

- I am not a cryptographer – how do I know a good cipher?
- Basic cipher analysis in under a minute
  - Keysize
    - For symmetric ciphers key > 128 is now considered best practice
  - Public
    - Security cannot come from obscurity (Kerkhoffs principle)
  - Standard
    - If the cipher is as result of open competition good, if proprietary be wary?
    - If it is old and public and still not 'broken' then could be OK.
  - Mode of operation?
    - Of the basic modes CBC is considered good (ECB not good)



Mifare Classic

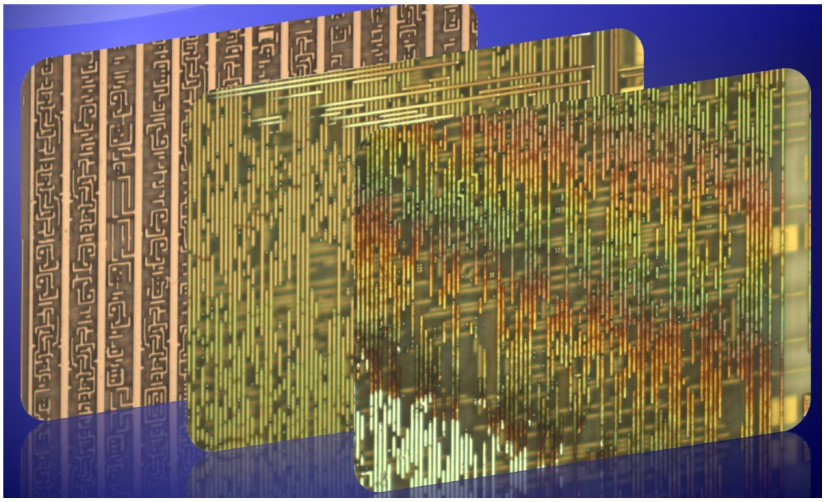
- Developed in 1995 (NXP Semiconductor) – Crypto1 algorithm
  - 48 bit key, stream cipher
- Used in a significant number of current systems
  - Access control
  - Travel
  - Closed payment
- Cipher kept secret…was securely used for a long time
- Researchers reverse engineered cipher by analysis of the IC architecture
  - Subsequently another group also used these findings to reconstruct the full cipher
- Mifare Classic also shown to have further security flaws

What do you think about this cipher?

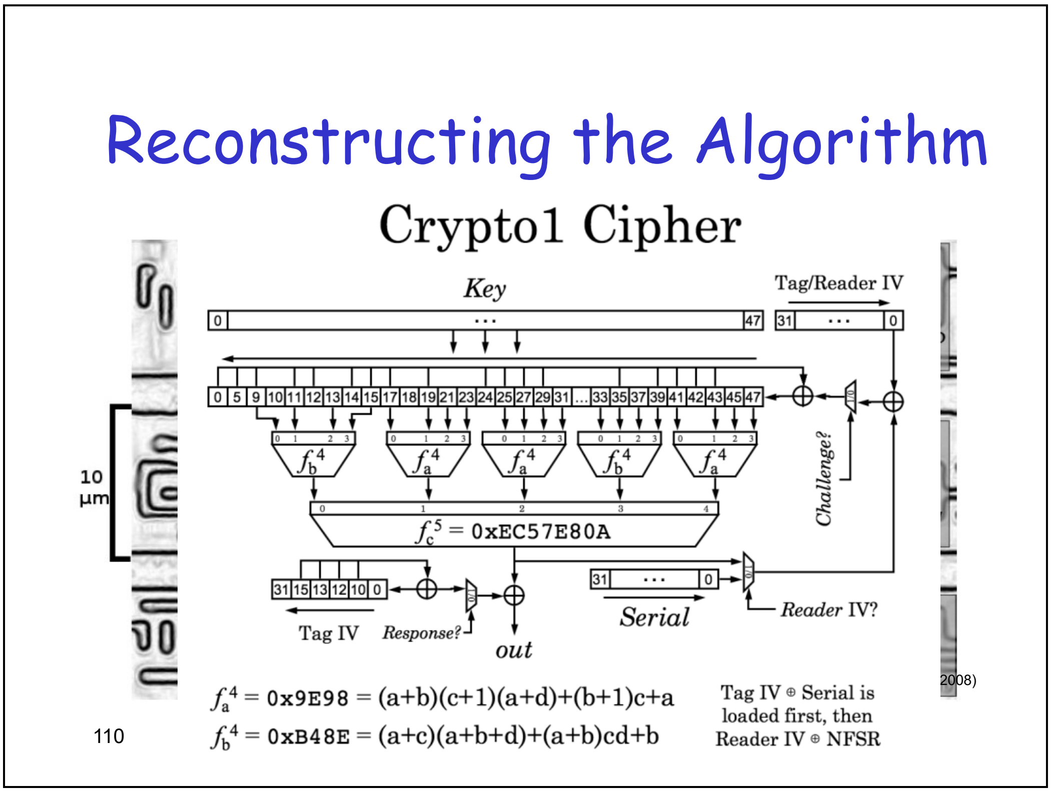
Keysize? Poor

Secrecy? Kerkhoffs principle? Bad, kept secret.

Mifare Classic Metal Layers



Nohl, et al (2008)



Reconstructing the Algorithm

$$f_a^4 = \mathtt{0x9E98} = (a+b)(c+1)(a+d)+(b+1)c+a$$

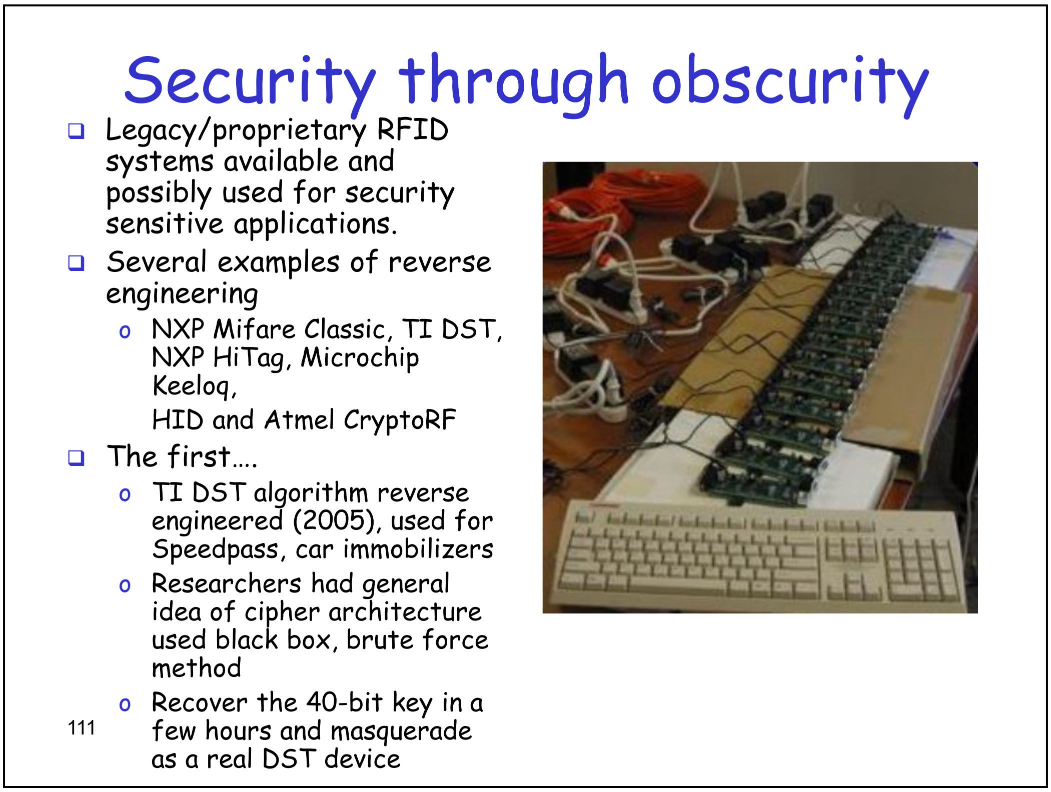
$$f_b^4 = \mathtt{0xB48E} = (a+c)(a+b+d)+(a+b)cd+b$$

Tag IV ⊕ Serial is
loaded first, then
Reader IV ⊕ NFSR

Would sufficient tamper resistance have prevented this attack? Not really. Tamper resistance usually protects the inner circuitry from being read (e.g. probe data lines, read memory content – if chip is opened the memory is deleted), but here we are interested in the logic only by observing metal layers so unless there is some countermeasure in terms of obfuscation of fake gates it is not likely to help.

And or Invert (AOI)

Initial state of the register is the key, possible to reverse LFSR to initial state and get the key.



Security through obscurity

- Legacy/proprietary RFID systems available and possibly used for security sensitive applications.
- Several examples of reverse engineering
  - NXP Mifare Classic, TI DST, NXP HiTag, Microchip Keeloq,
    HID and Atmel CryptoRF
- The first….
  - TI DST algorithm reverse engineered (2005), used for Speedpass, car immobilizers
  - Researchers had general idea of cipher architecture used black box, brute force method
  - Recover the 40-bit key in a few hours and masquerade as a real DST device

Illustrate skill and devotion of attackers…

The end!

?

Any questions…