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

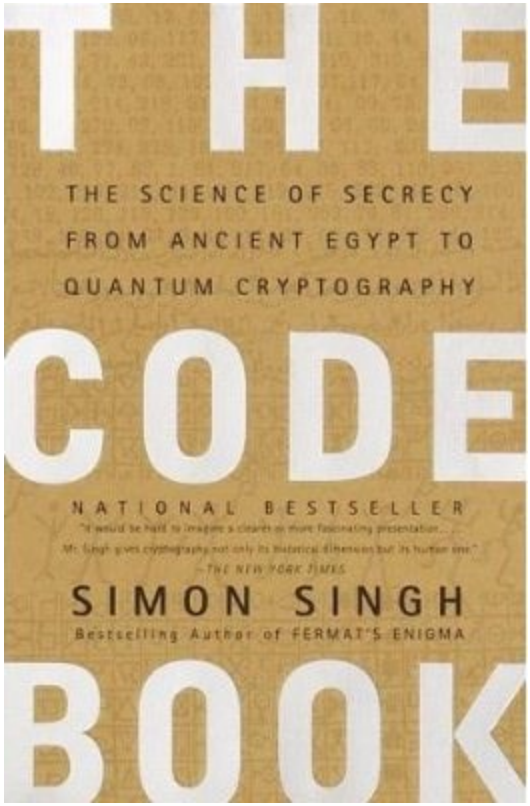
  (technology that impact systems, and security mechanisms)

Cryptographic Tools:

Symmetric Key Encryption

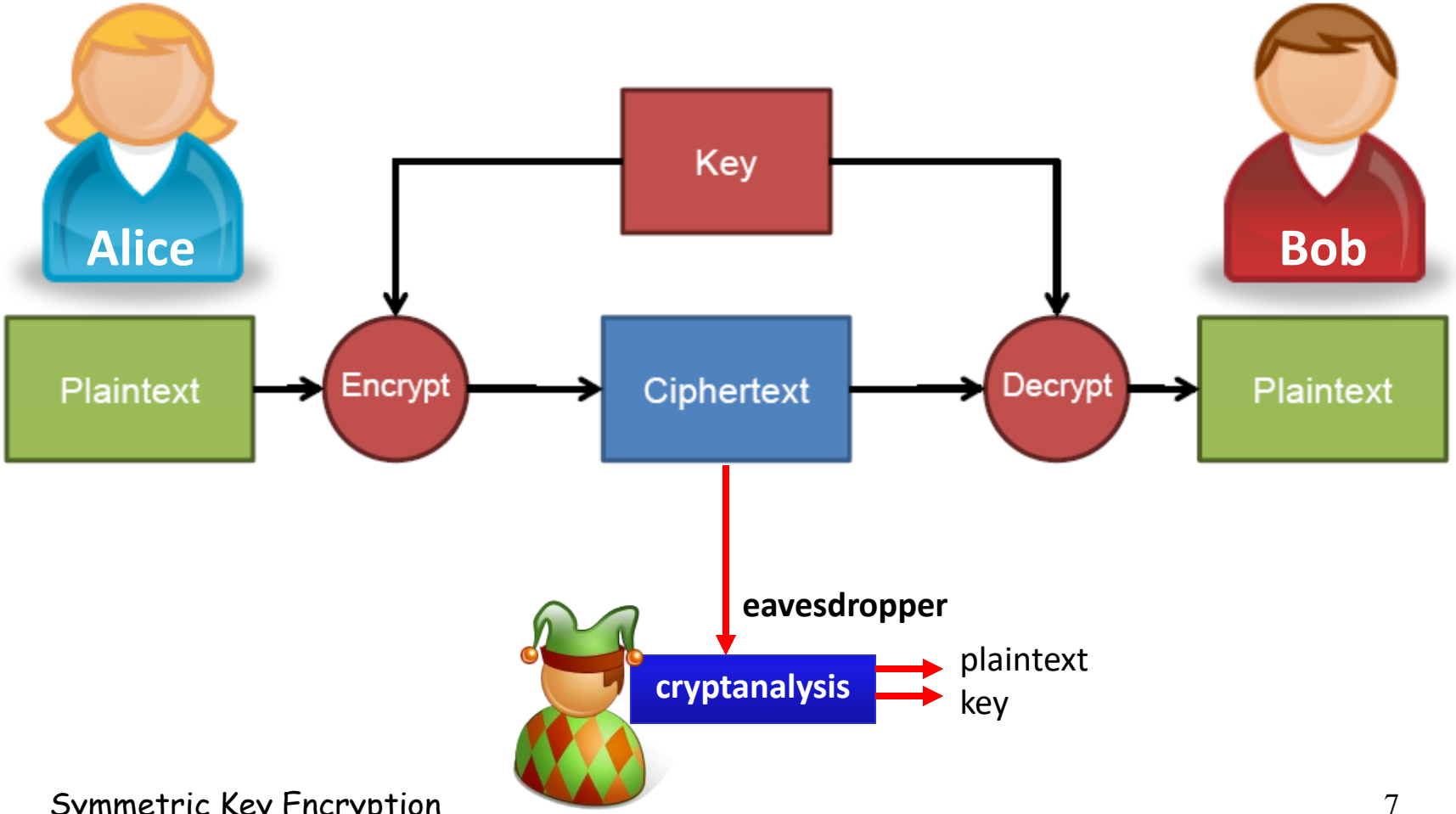
Crypto – a brief introduction

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



"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

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



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

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

Ciphertexts:

1. IRXUVFRUHDQGVHYHQBHDUVDJR
2. VSRQJHREVTXDUHSDQWV

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

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:

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.

K H O O R H Y H U B R Q H

h e l l o e v e r y o n e

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

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

pure mathematics is, in its way,

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

the poetry of logical ideas

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

puremathicsnwyoflgd

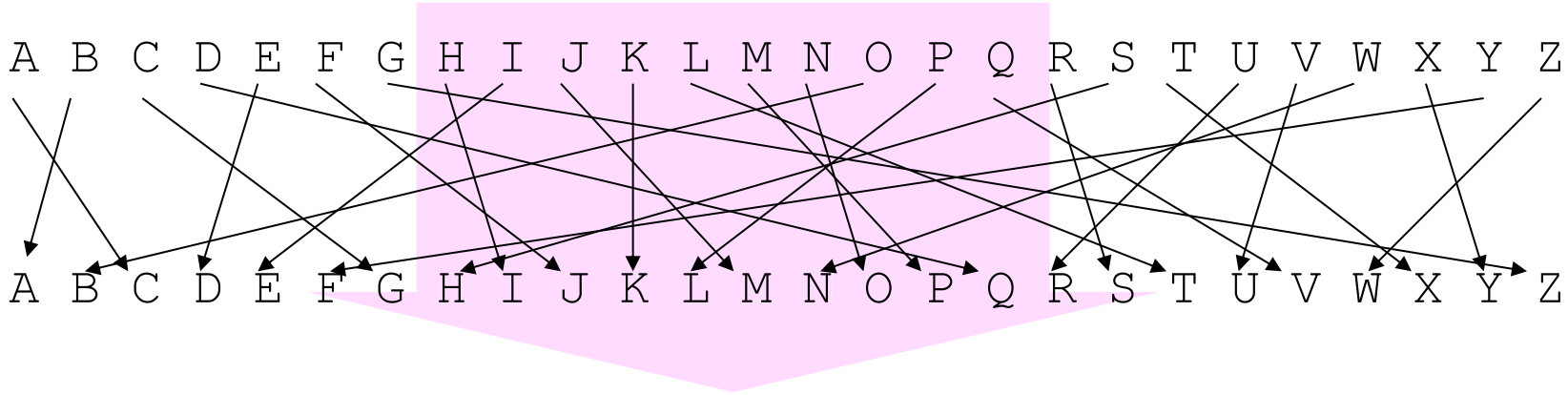
- Fill in this sequence as the start of your cipher alphabet.

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

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

EIMBULJIWLNYANJMVLIURAHIWAI



DEPARTMENTOFCOMPUTERSCIENCE

An example of simple substitution…

"Encryption software is expensive...so we just rearranged all the letters on your keyboard."

An Example

Ciphertext (encrypted using simple substitution)

PBFPVYFBQXZTYFPBFEQJHDXXQVAPTPQJKTOYQWIPBVWLXTOXBTFXQWAX
BVCXQWAXFQJVWLEQNTOZQGGQLFXQWAKVWLXQWAEBIPBFXFQVXGTVJ
VWLBTPQWAEBFPBFHCVLXBQUFEVWLXGDPEQVPQGVPPBFTIXPFHXZHVFA
GFOTHFEFBQUFTDHZBQPOTHXTYFTODXQHFTDPTOGHFQPBQWAQJJTODX
QHFOQPWTBDHHIXQVAPBFZQHCFWPFHPBFIPBQWKFABVYYDZBOTHPBQP
QJTQOTOGHFQAPBFEQJHDXXQVAVXEBQPEFZBVFOJIWFFACFCCFHQWAUV
WFLQHGFXVAFXQHFUFHILTTAVWAFFAWTEVOITDHFHFQAITIXPFHXAFQHEF
ZQWGFLVWPTOFFA

Question: how secure is Simple Substitution?

Let’s do some analysis…

- 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?)

- Total number of possible permutations

$$26!$$

- $26! = 403,291,461,126,605,635,584,000,000$ (27 digits) $\approx 2^{88}$
- 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?

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

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*

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

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

Example

NKRRU NKXK OY G ZKYZ

SKYYGMK ZU KTIOVNKX LUX AYK

GY GT KDGSVRK OT GT GXZOIRK

LUX OYWAGXKJ SGMGFOTK

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																					

Example

NeRRU **Ne**Xe OY a ZeYZ

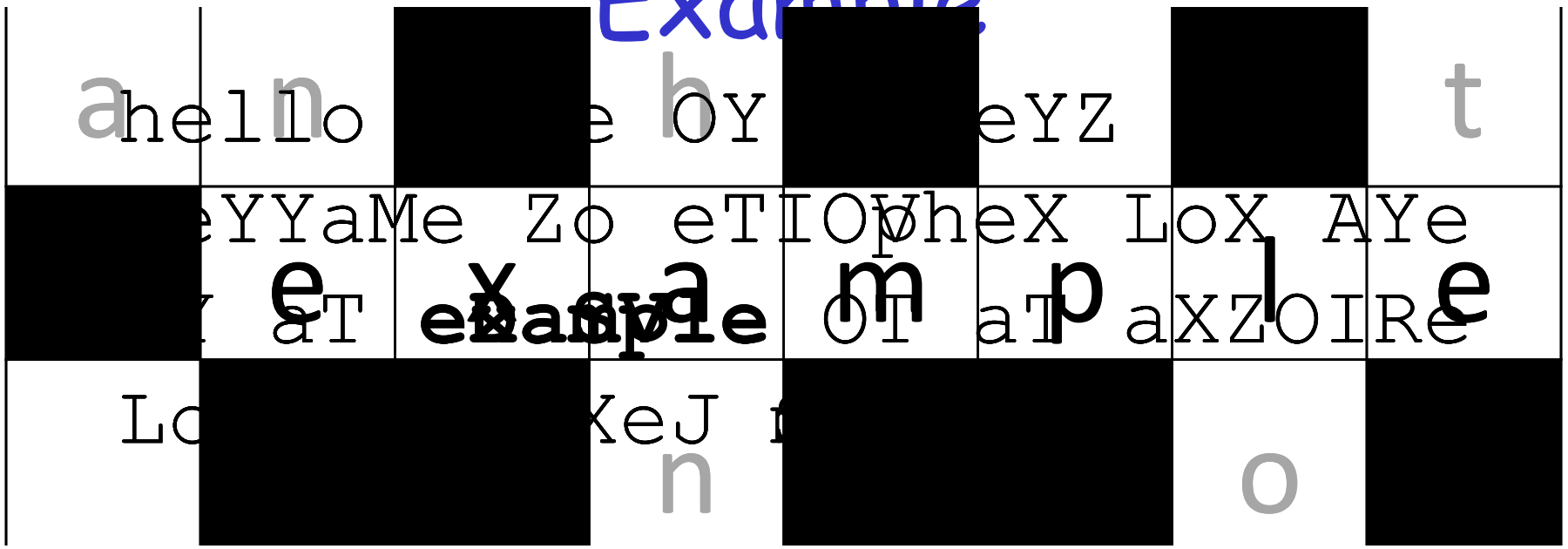
SeYYaMe ZU eTIOV**Ne**X LUX AYe

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																		

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		

Example

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

```
hello heXe OY a ZeYZ
meYYaMe Zo eTIOpheX LoX AYe
aY aT example OT aT aXZOIRe
LoX OYWAaXeJ maMaFOTe
```

- 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

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

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

- 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).

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

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?

- 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: CRYPTOISSHORTFORCRYPTOGRAPHY
Ciphertext: CSASTPKVSIQUTGQUCSASTPIUAQJB
```

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

```
Ciphertext: VHVSSPQUCEMRVBVBBBVHVSURQGIBDUGRNICJQUCERVUAXSSR
```

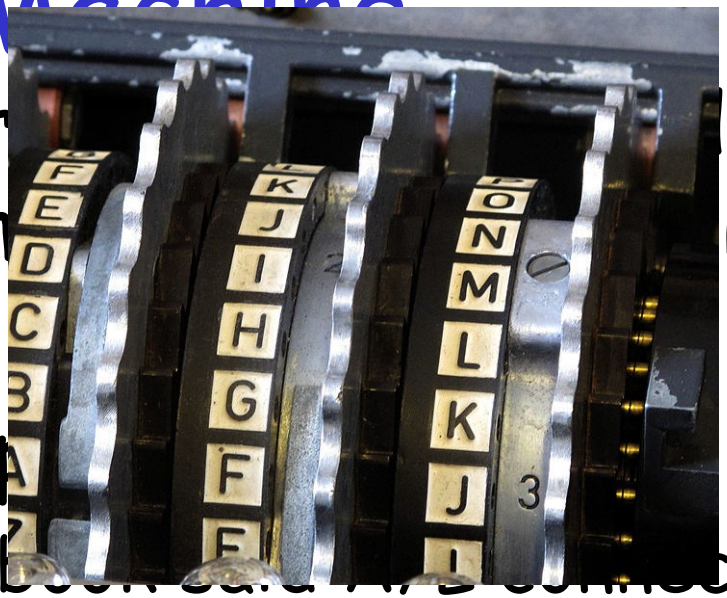
- VHVS is 18 (18,9,6,3,2,1) and QUCE is 30 (30,15,10,6,5,3,2,1)
- Key size 6,3,2,1 (most probably 6)

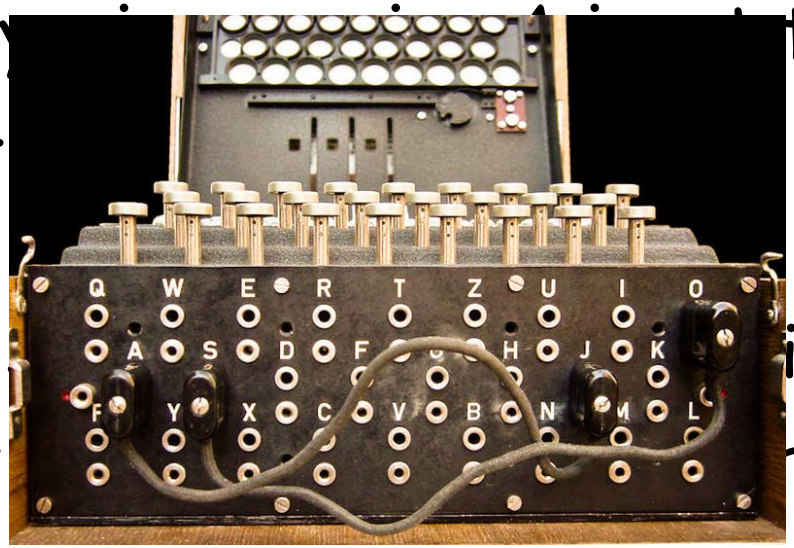
- Usec II
  - Las
- Polyc
  - To have ma state as the

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

Enigma Machine

- Se… iver… …e bo… re m… …ic da…
- Plu… 10 …
  - L… …odeb… …connect these b… …to be seen as L, and L…
- Rotors
  - Choose … …ied order
  - Set init… …r





Enigma Machine

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

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

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

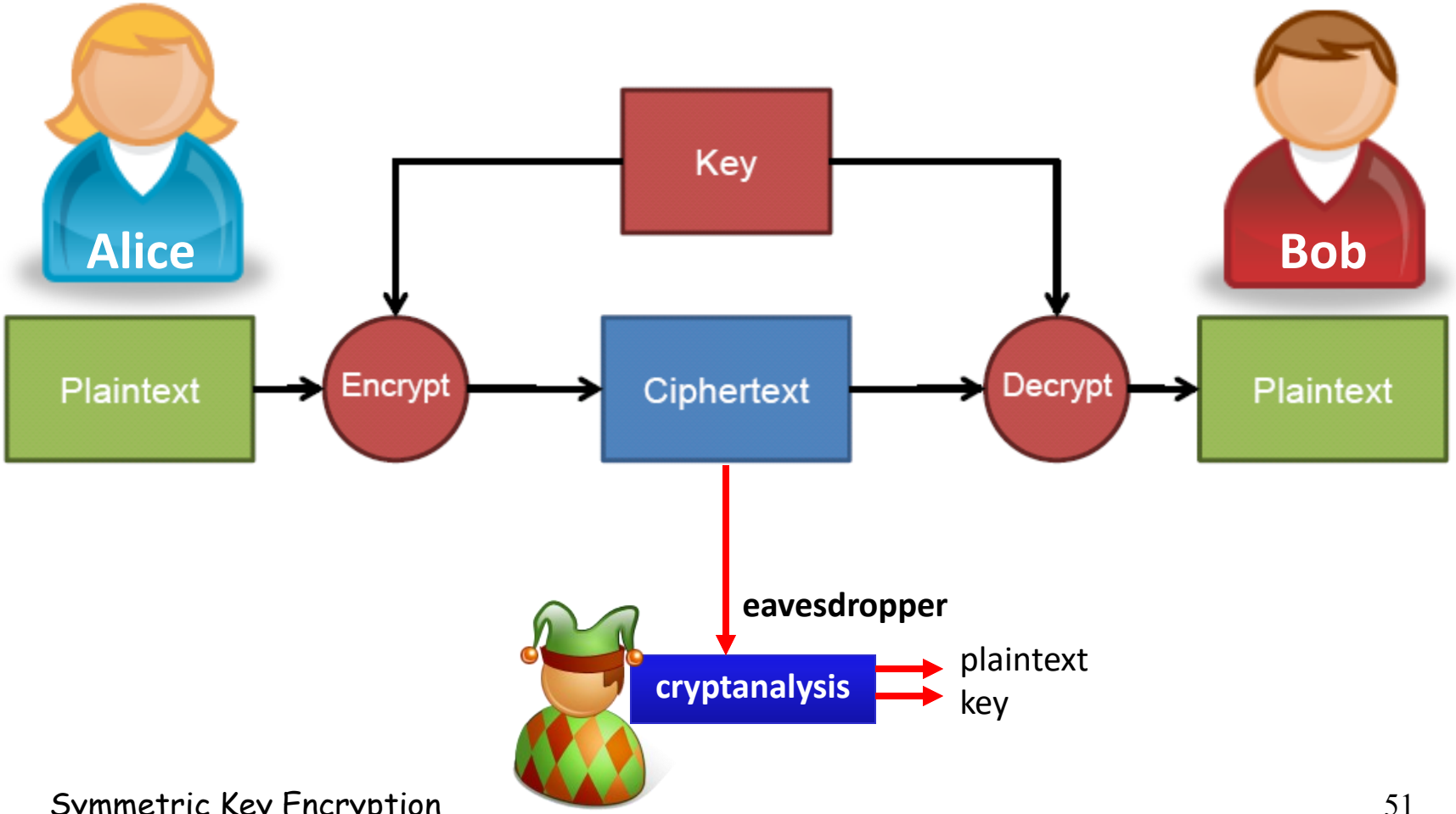
- Pad must be random, **used only once**
- Pad has the same size as message

One-time Pad Use

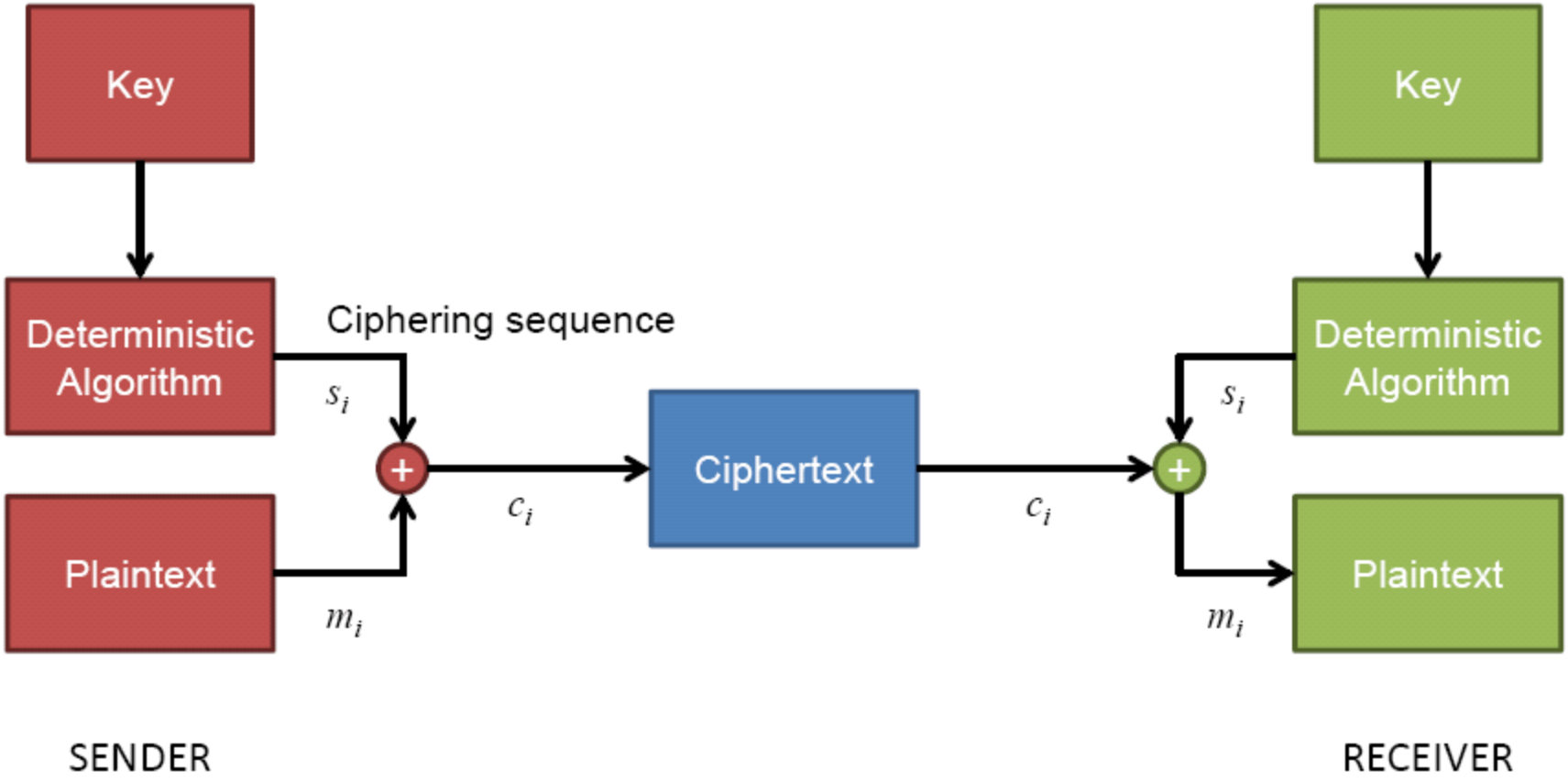
Ciphertext:	97	6F	DE	31	A8	A2	82	4B	5C	0D
Key:	F5	16	BB	53	D1	C7	E8	24	34	63
Plaintext:	62	79	65	62	79	65	6A	6F	68	6E
	b	y	e	b	y	e	j	o	h	n

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

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

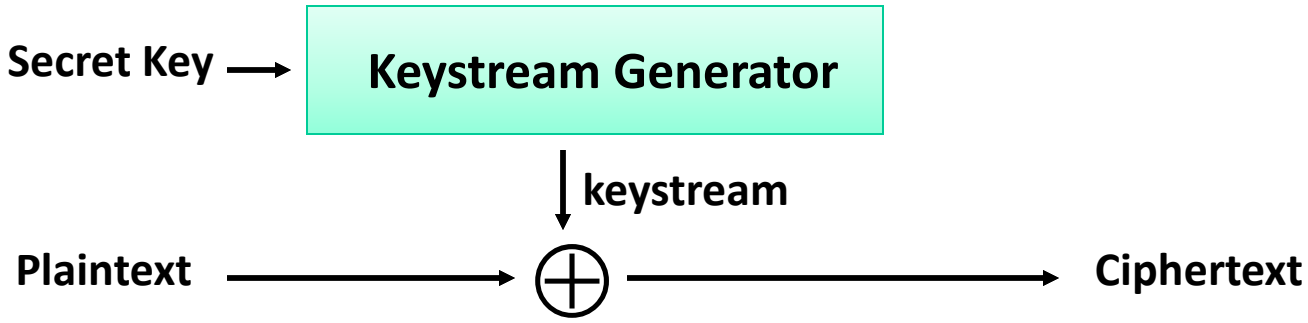


Stream Ciphers



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

Stream Ciphers



- 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.

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

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

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.

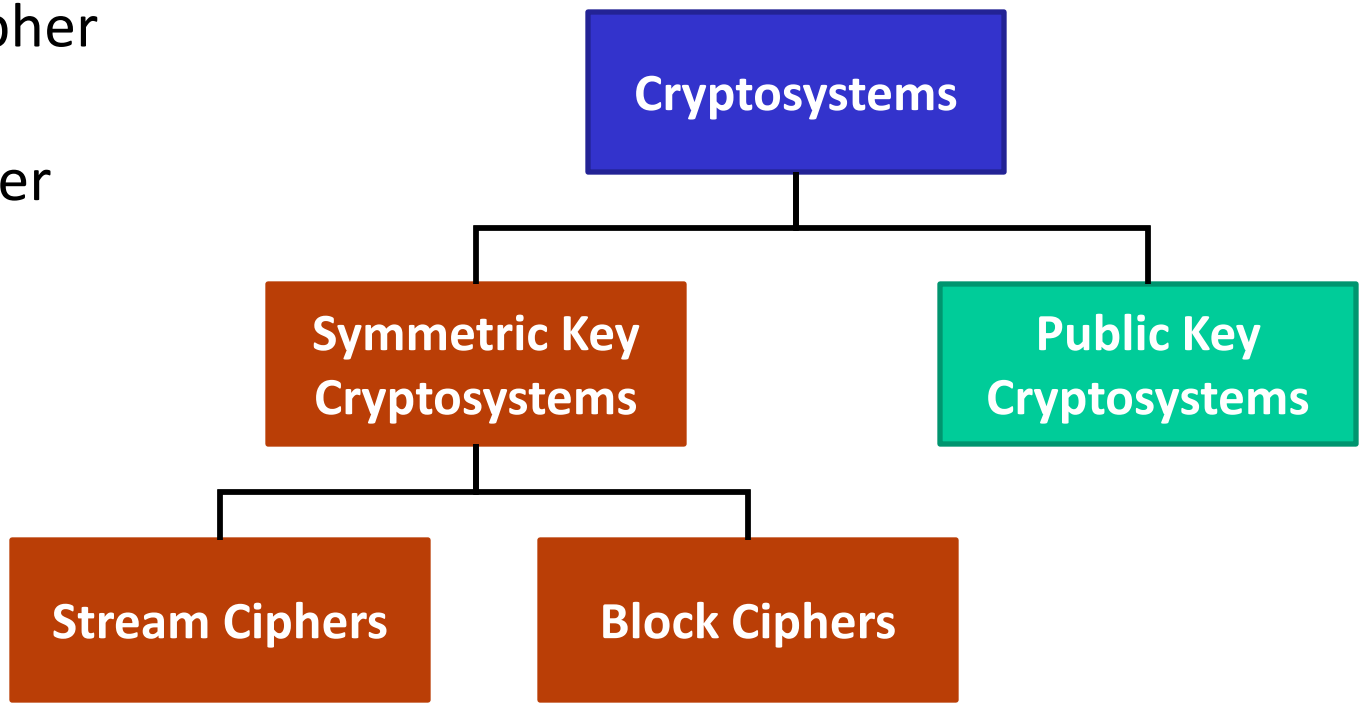
Questions: What are the current symmetric key cryptosystems?

There are many…

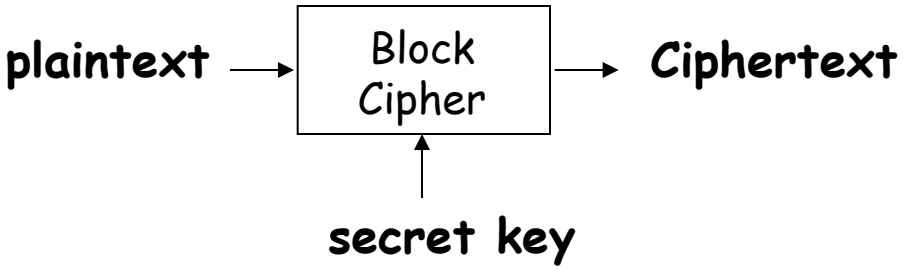
They can be categorized into two types:

1.Stream Cipher

2.Block Cipher



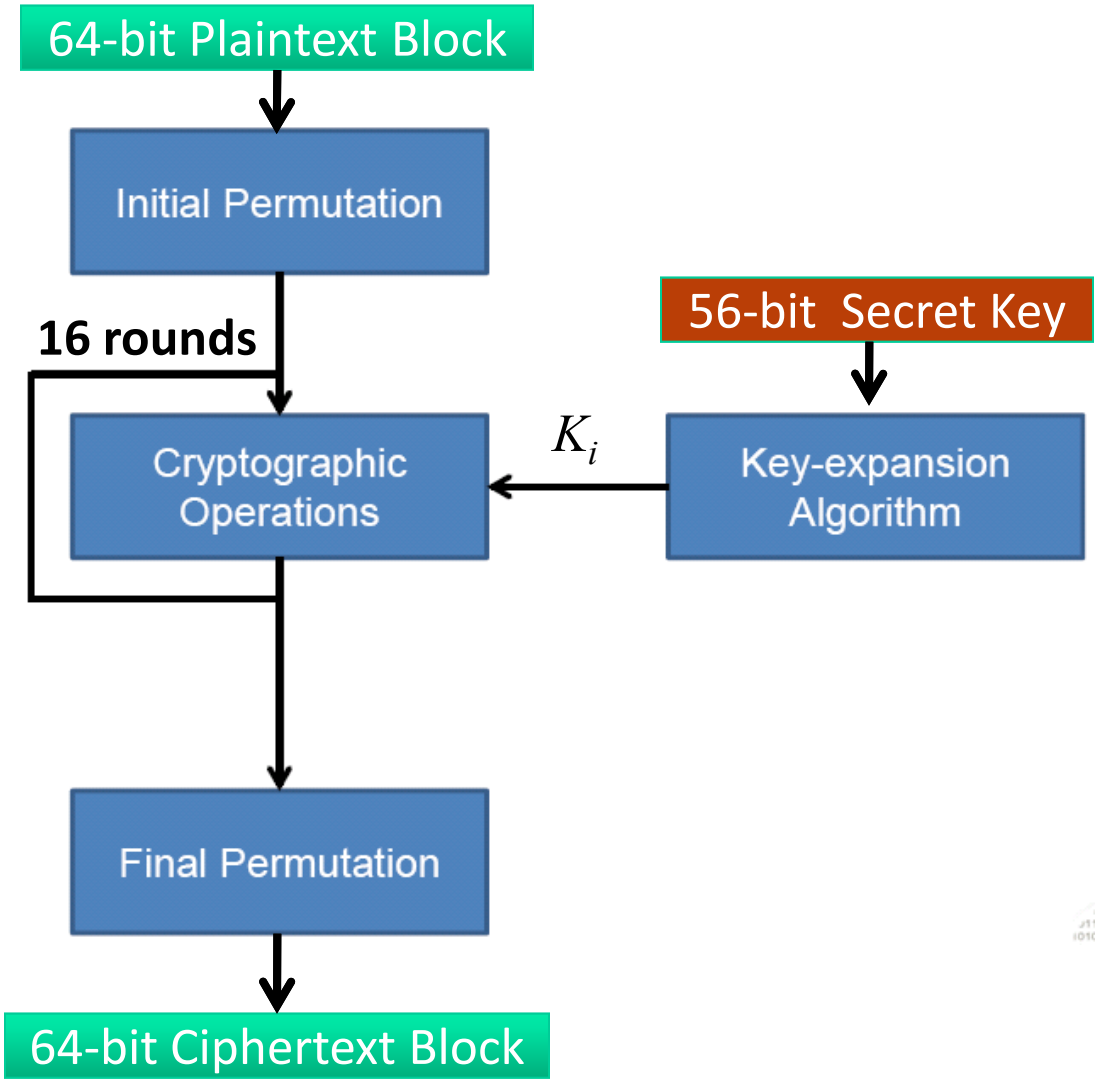
Block Ciphers



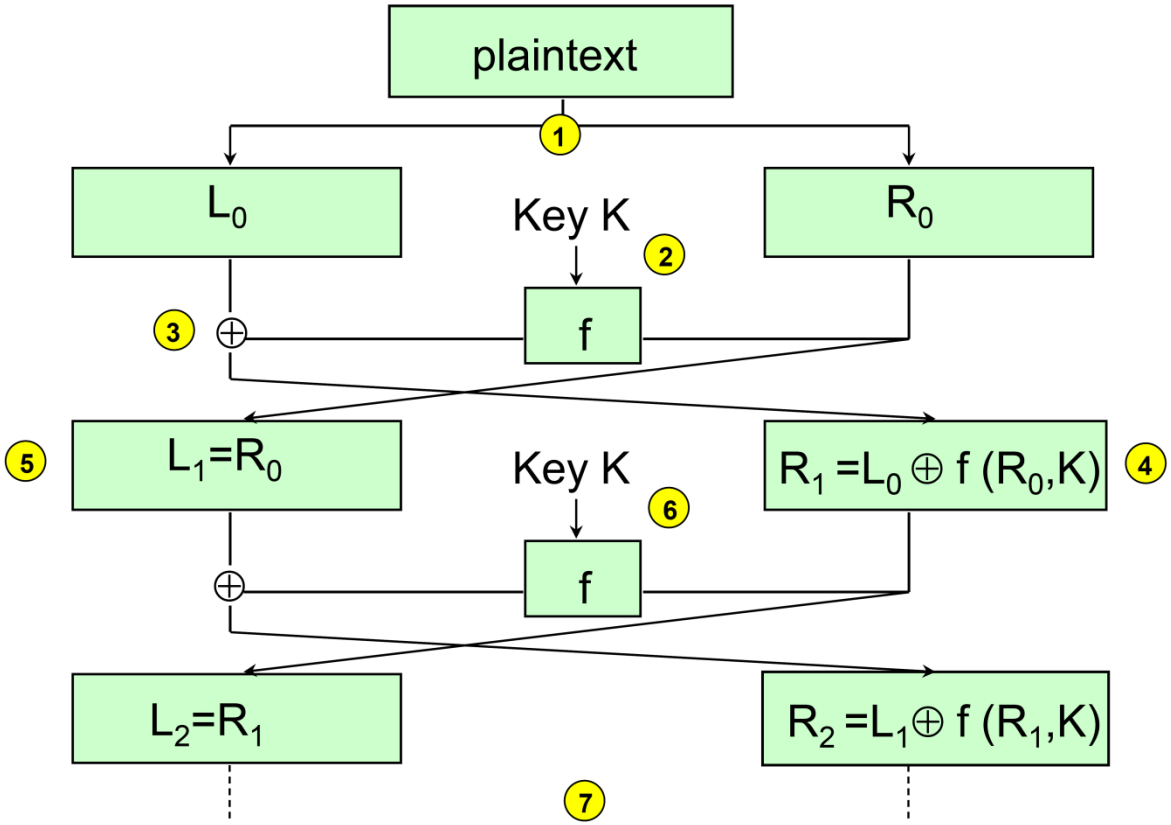
- A block cipher takes a *block* of **plaintext** and a **secret key**, produces a *block* of **ciphertext**.
- The key is **reused** for different plaintext blocks
- Typical block sizes: 64 bits, 128 bits, 192 bits, 256 bits
- Key sizes: 56 bits (DES), 128/192/256 bits (AES)
- Popular block ciphers: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

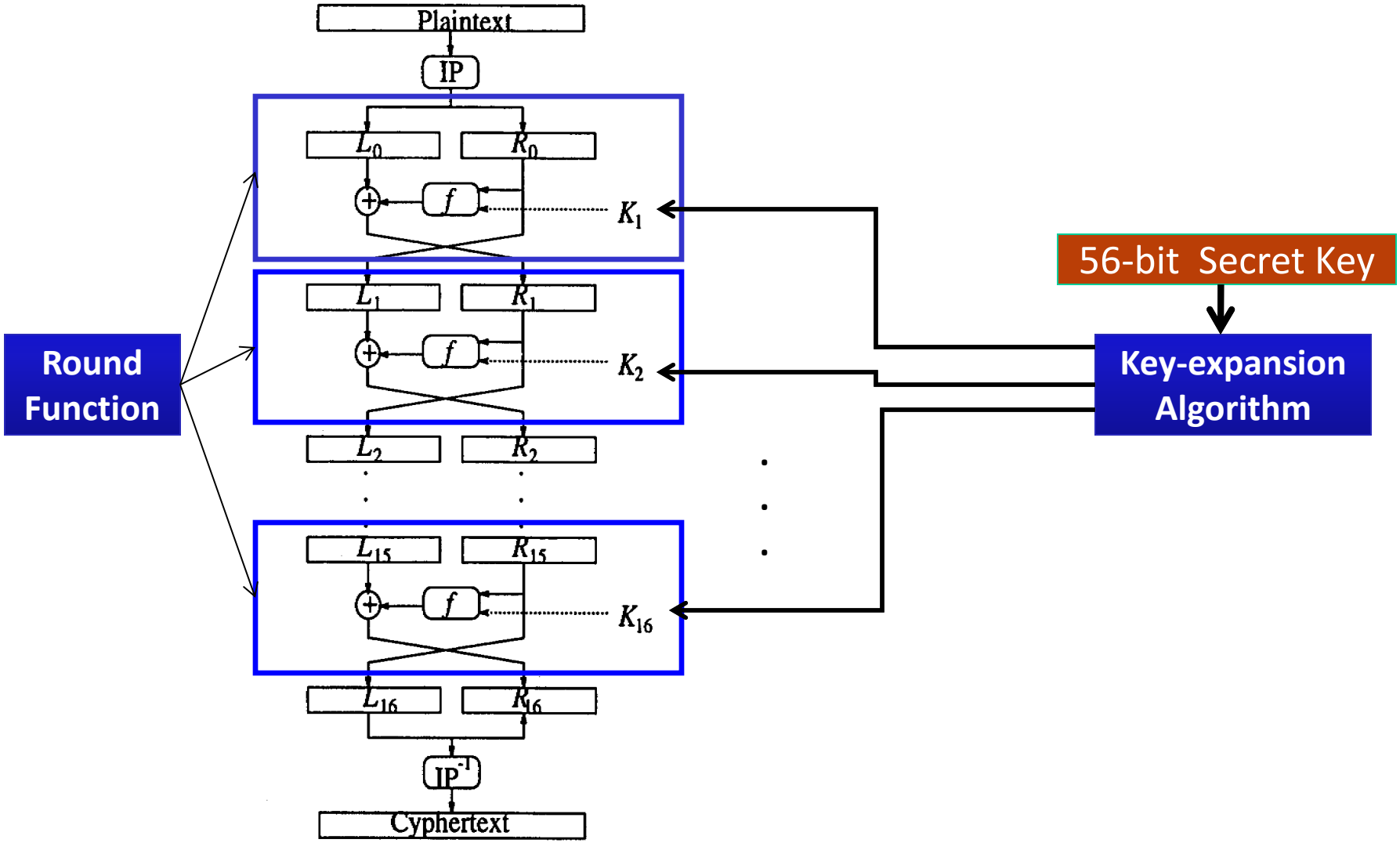
DES (Data Encryption Standard)

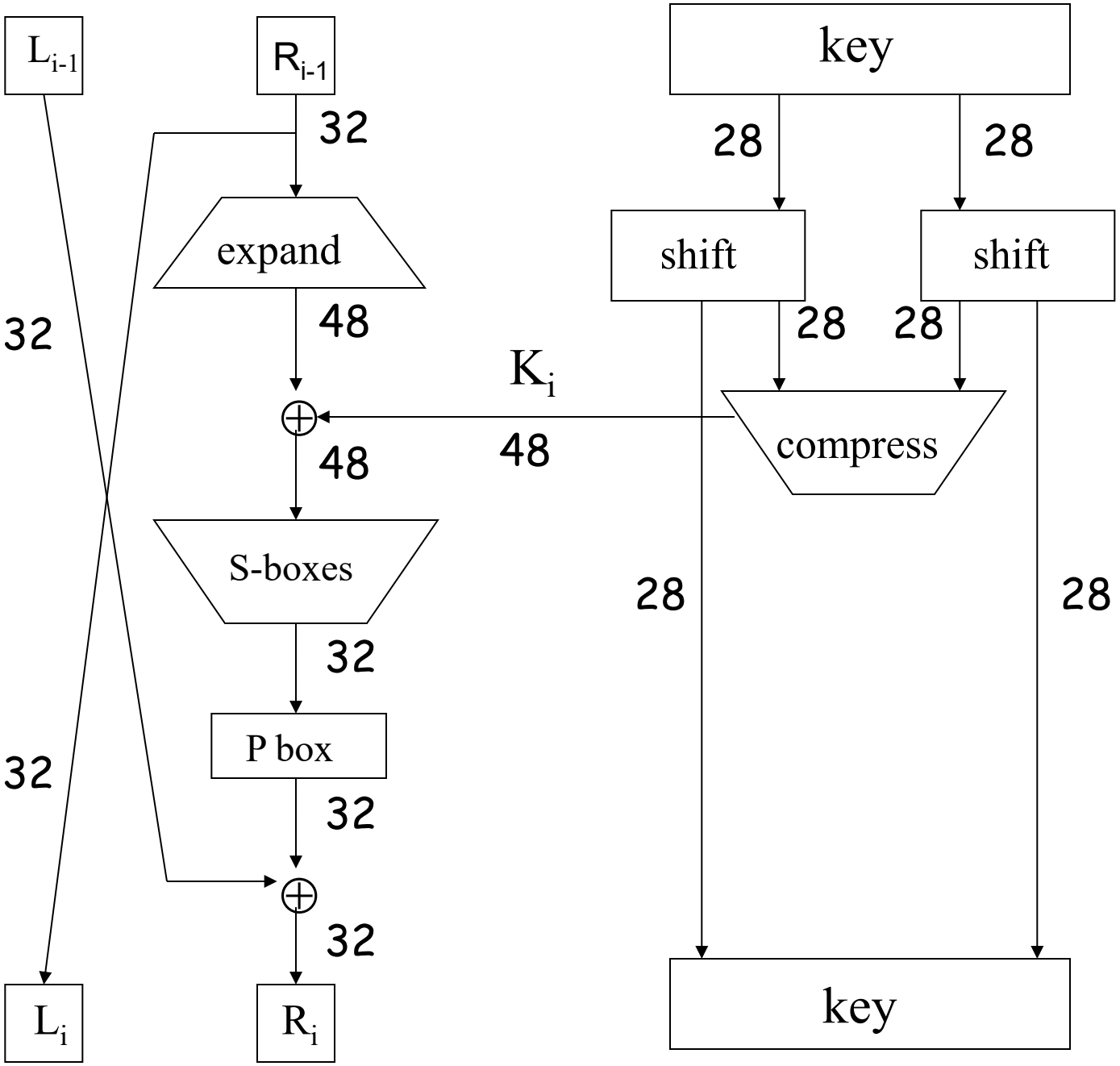
- Ciphertext obtained from plaintext by iterating a **round function (i.e. cryptographic operations)**
- Input to round function consists of **a round key** K_i and the output of the previous round
- The DES round function is also known as **Feistel Transformation**



Feistel Structure




Plaintext
IP
L_0
R_0
K_1
L_1
R_1
K_2
L_2
R_2
L_{15}
R_{15}
K_{16}
L_{16}
R_{16}
IP^{-1}
Cyphertext
Round Function
56-bit Secret Key
Key-expansion Algorithm



One Round of DES

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

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)

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.

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 = $2^{56} \approx 7.2 \times 10^{16}$ 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.

Exhaustive Key Search

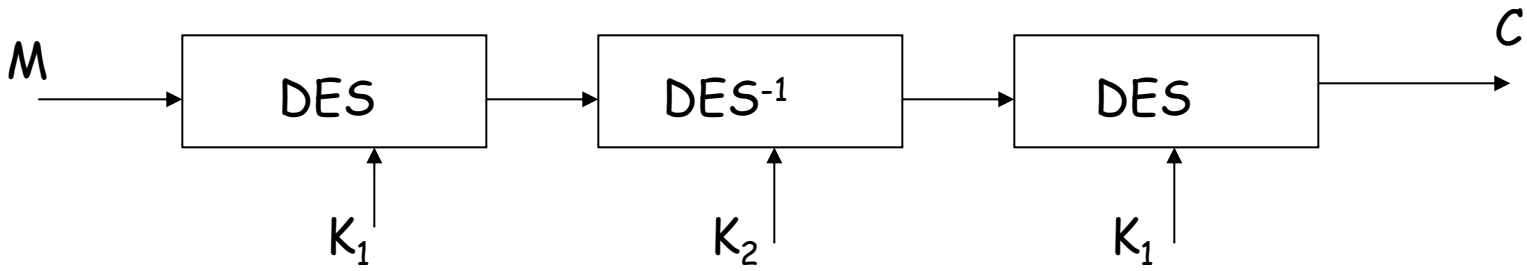
Year	Source	Implemented?	(Estimated) Cost in US$	(Estimated) Search time
1977	Diffie Hellman	No	20 million	20 hours
1993	Wiener	No	10.5 million 1.5 million 600 000	21 minutes 3.5 hours 35 hours
1997	Internet	Yes	Unknown	140 days
1998	Deep Crack	Yes	210 000	56 hours
2007	COPACOBANA	Yes	<10,000	<7 days

What Should We Use Today?

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

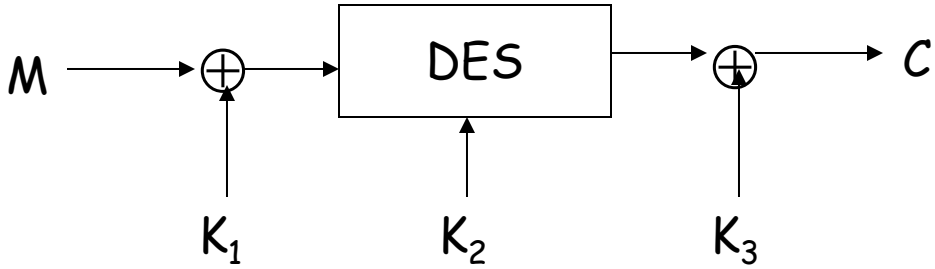
Triple DES and DESX

- Triple DES: two 56-bit keys



- DESX: three keys

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



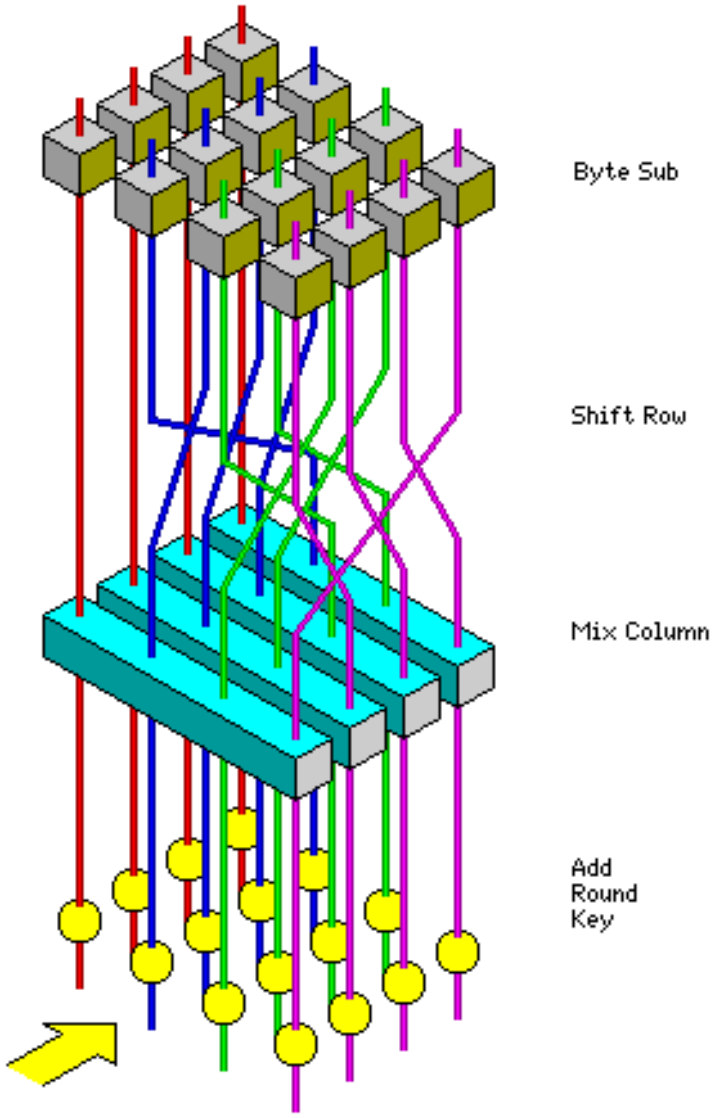
- Similar security to DES using differential cryptanalysis and linear cryptanalysis, which are theoretical attacks
- But much harder to break using exhaustive key search than DES.

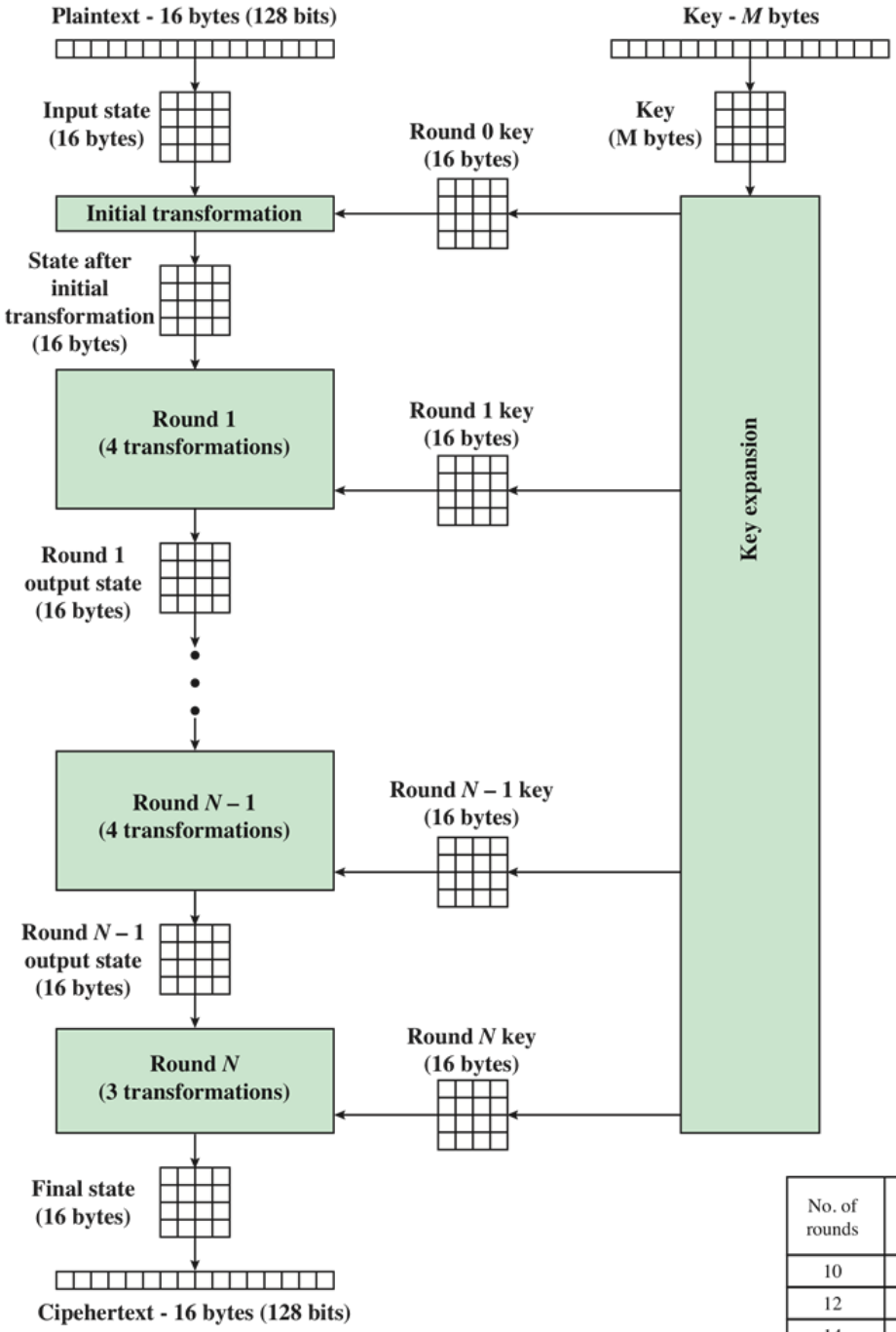
Advanced Encryption Standard

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

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)
  - **ByteSub**
  - **ShiftRow**
  - **MixColumn**
  - **AddRoundKey**

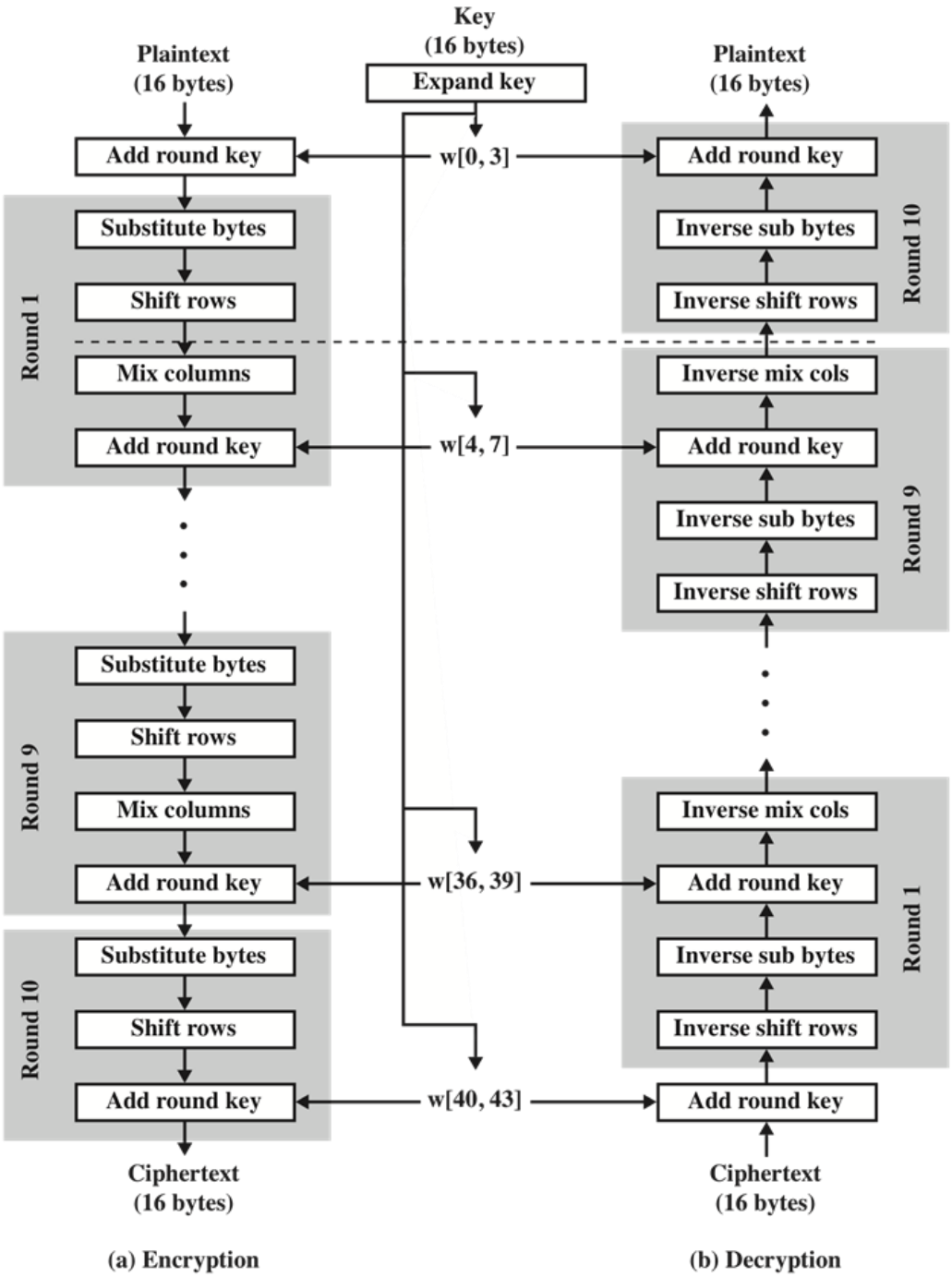




No. of rounds	Key Length (bytes)
10	16
12	24
14	32

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)



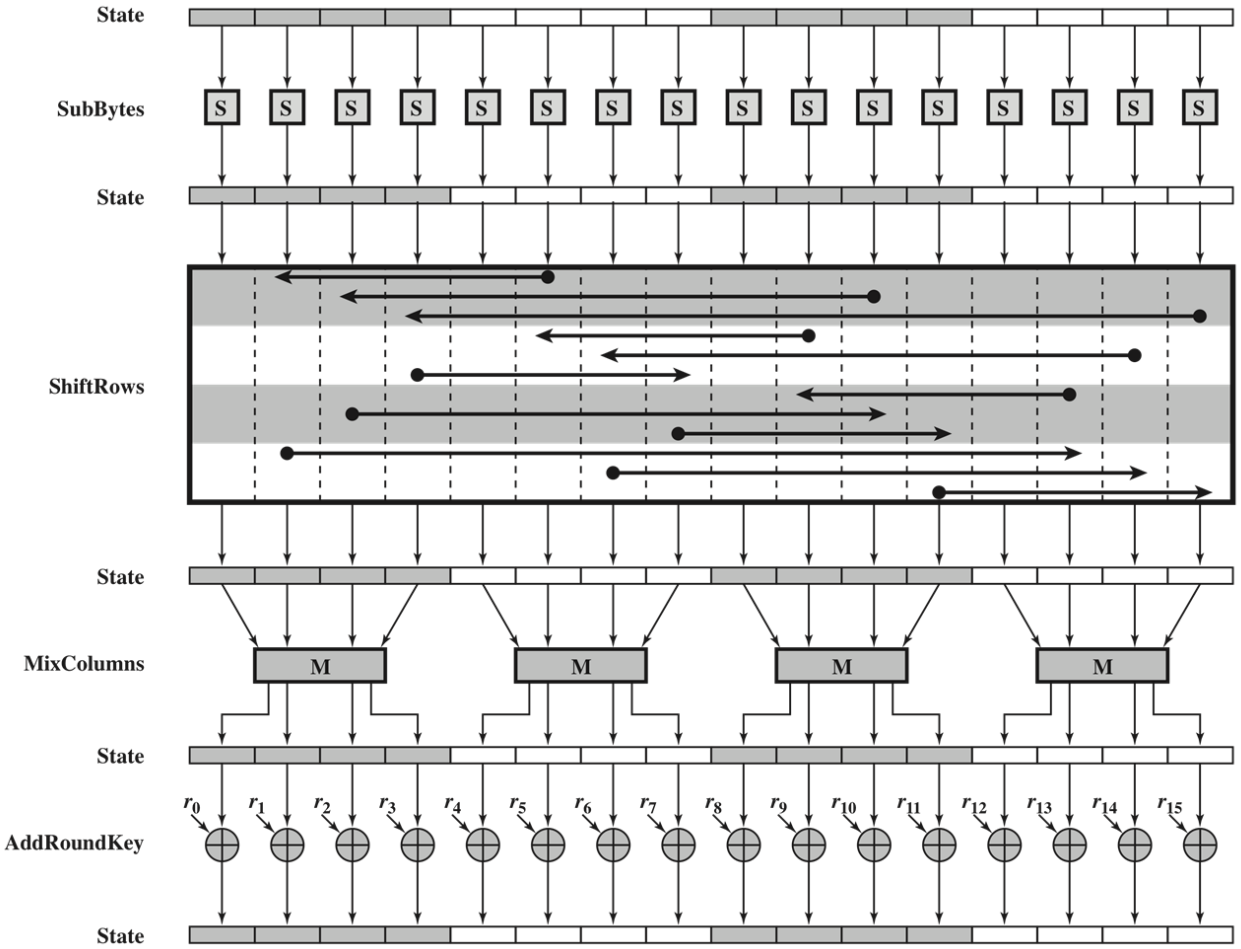
AES Encryption and Decryption

- 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***

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



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

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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)

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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)

AES

ByteSub

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

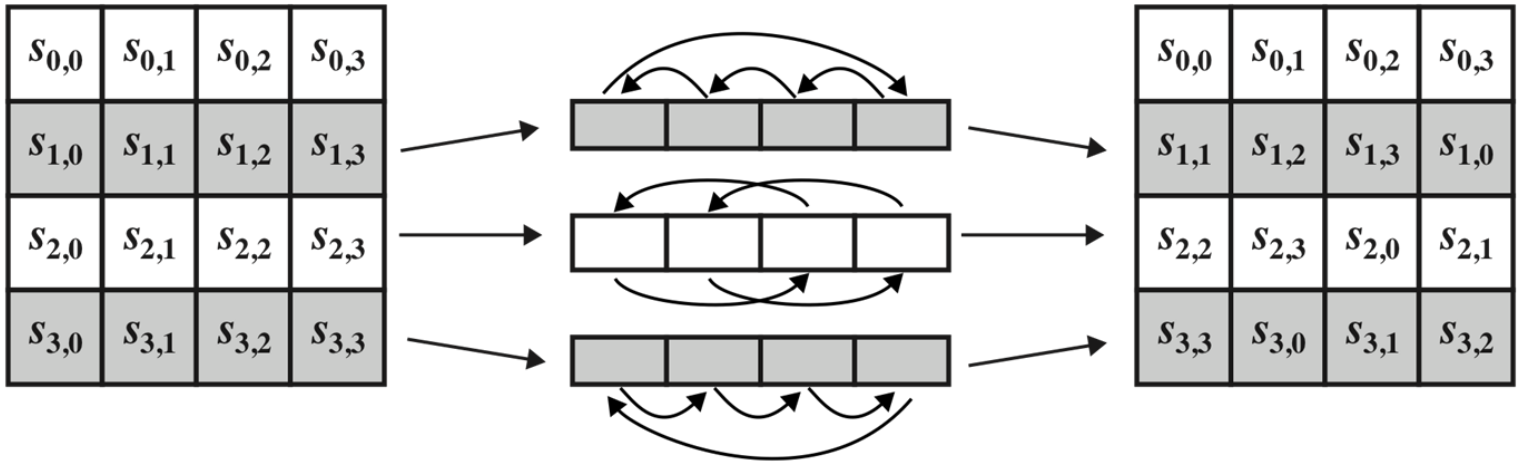
EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

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

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



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

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)

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

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

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

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

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

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

    $= x^2 + x$
  - **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. $s'_{0,0} = [(x) \cdot s_{0,0} \bmod m(x)] \oplus [(x+1) \cdot s_{1,0} \bmod m(x)] \oplus s_{2,0} \oplus s_{3,0}$

AES

MixColumn

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

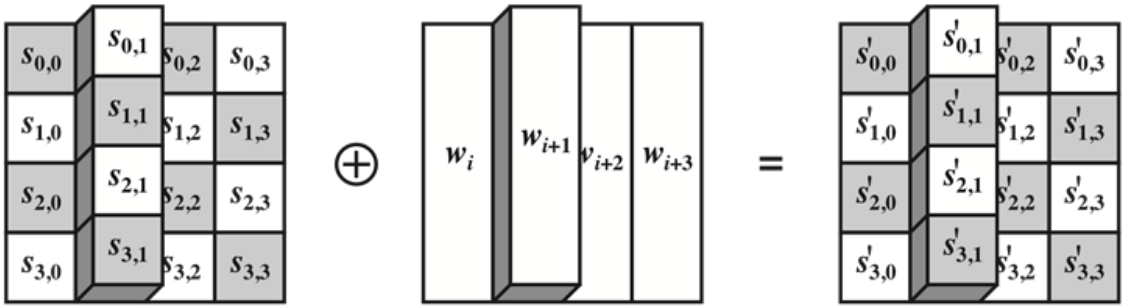
47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

Example

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

AES

AddRoundKey



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

⊕

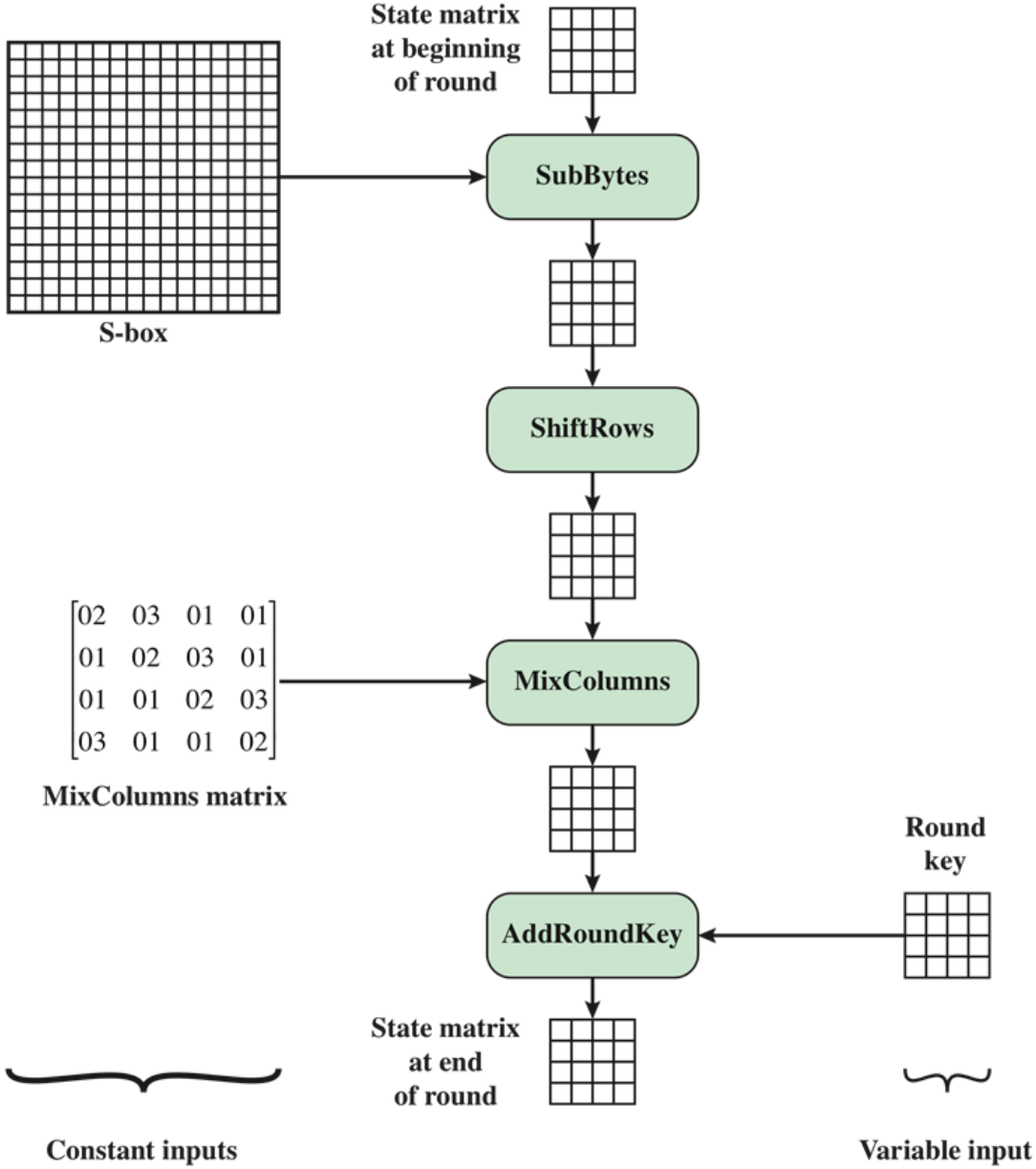
AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A

=

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

AES

Summary of One AES Round (except the last round)



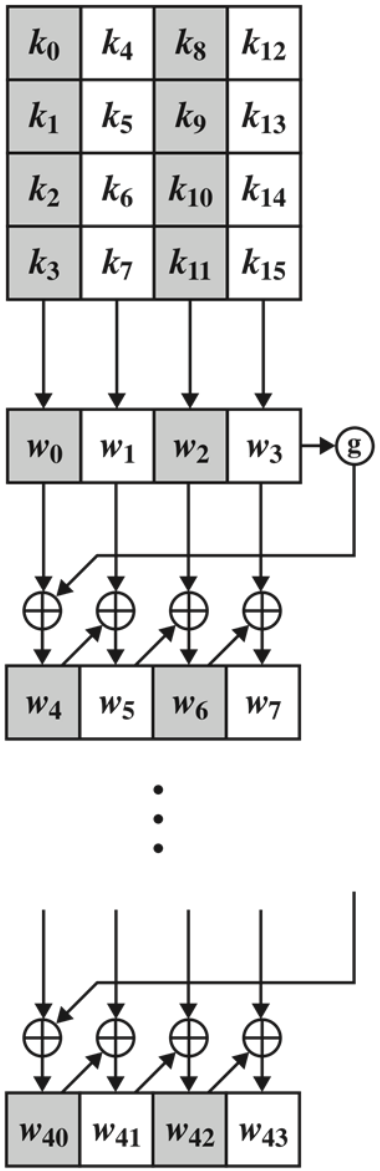
AES

Key Expansion / Key Scheduling

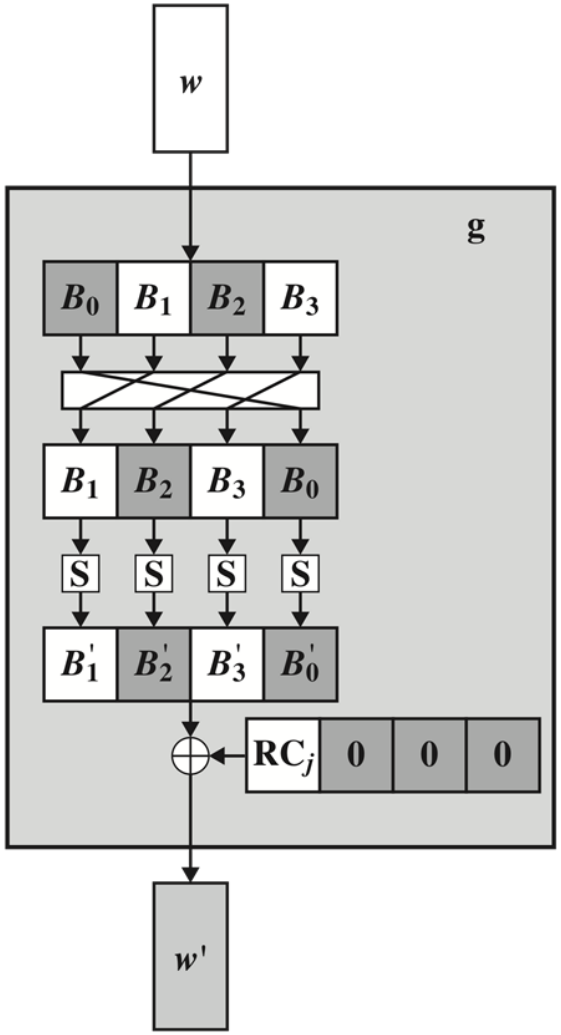
Review:
- A 16-byte (128-bit) Key is expanded into 11 round keys
- Each round key is 4 words (or 16 bytes or 128 bits) long
- Total size of the 11 round keys = 44 words (or 176 bytes)

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: $w_{40}, w_{41}, w_{42}, w_{43}$



(a) Overall algorithm



(b) Function g

AES

Key Expansion / Key Scheduling

Summary:

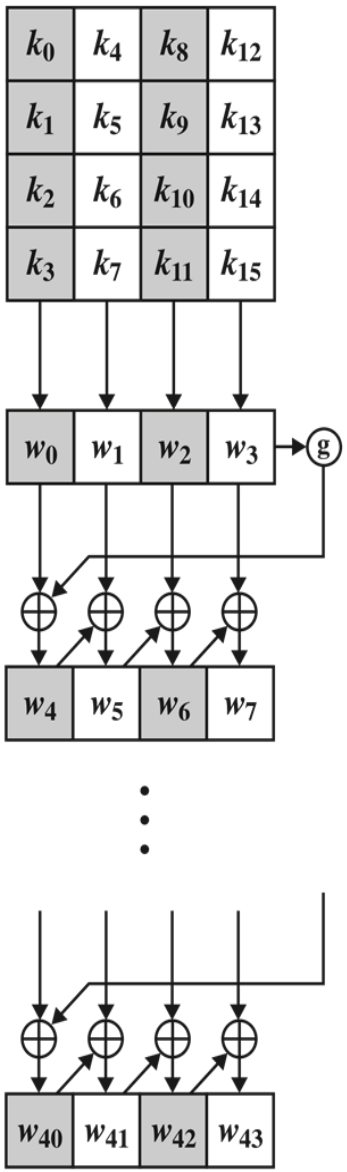
•The 16-byte key is copied into the first four words for Round 0 key

•i.e. the key is used **directly** to do the AddRoundKey at the initial transformation

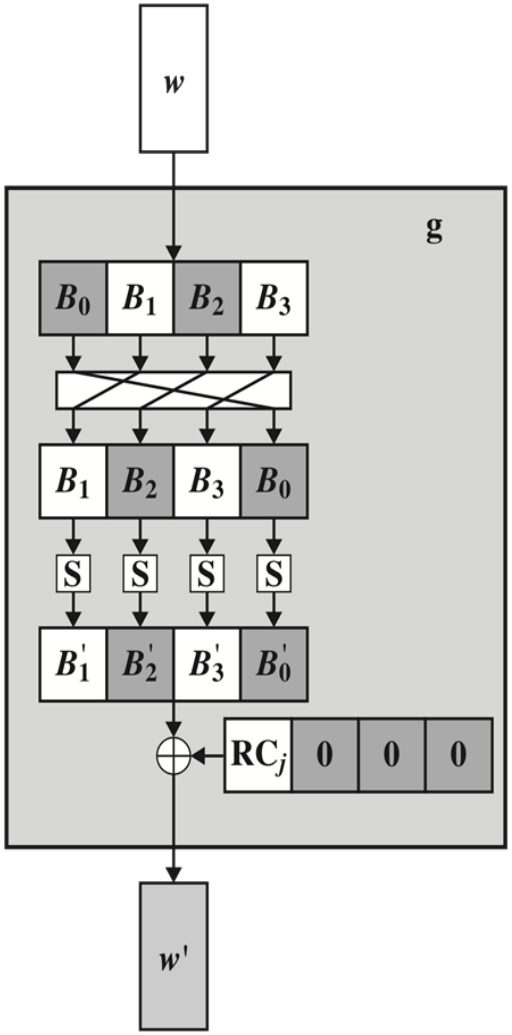
•Each subsequent word w[i] in a round key depends on the immediately preceding word w[i-1], and the word four positions back, w[i-4]

- 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



(a) Overall algorithm

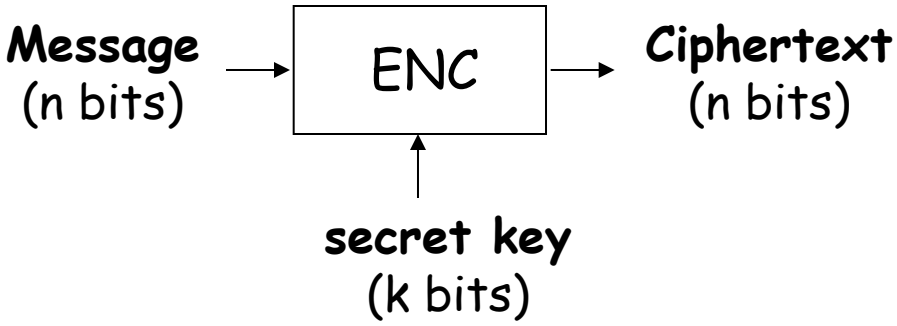


(b) Function g

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?

Multiple Blocks



- How to encrypt multiple blocks?
- A new key for each block?
  - 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?

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

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

$C_0 = E(K, P_0)$,

$C_1 = E(K, P_1)$,

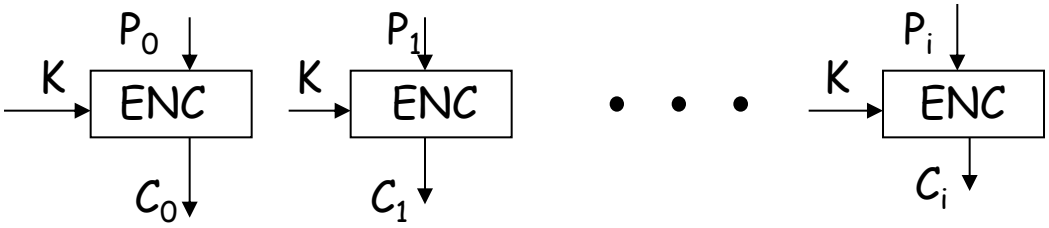
$C_2 = E(K, P_2),\ldots$

Decrypt

$P_0 = D(K, C_0)$,

$P_1 = D(K, C_1)$,

$P_2 = D(K, C_2),\ldots$



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.`

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?

Alice Hates ECB Mode

- Alice's uncompressed image, Alice ECB encrypted

- Why does this happen?
- Same plaintext block ⇒ same ciphertext!

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

$C_0 = E(K, IV \oplus P_0)$,

$C_1 = E(K, C_0 \oplus P_1)$,

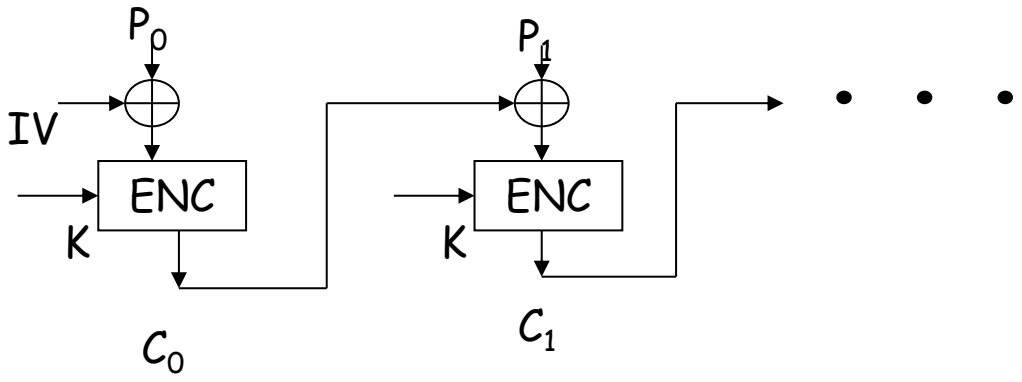
$C_2 = E(K, C_1 \oplus P_2)$,…

Decryption

$P_0 = IV \oplus D(K, C_0)$,

$P_1 = C_0 \oplus D(K, C_1)$,

$P_2 = C_1 \oplus D(K, C_2)$,…



Alice Likes CBC Mode

- Alice's uncompressed image, Alice CBC encrypted

- Why does this happen?
- Same plaintext yields different ciphertext!

What is a 'good' mode?

Good properties:

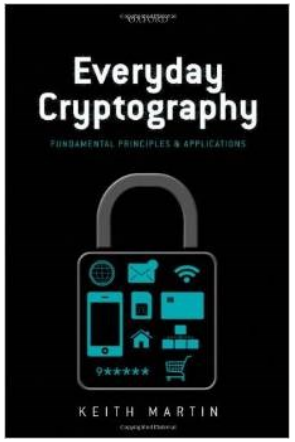
- Message dependence of ciphertext
- Limited error propagation
- Works without block synchronisation
- Optimise use of decrypt/encrypt
- Reduce padding

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



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.

Counter Mode (CTR)

- Use block cipher like stream cipher

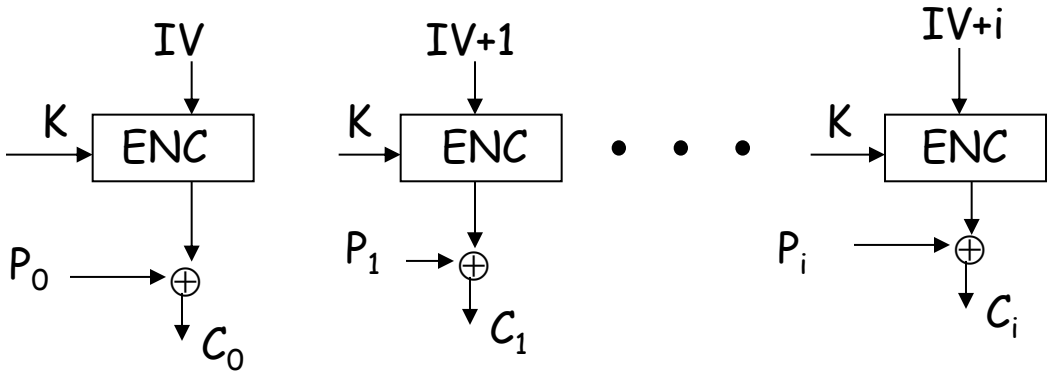
Encryption

$C_0 = P_0 \oplus E(K, IV)$,
$C_1 = P_1 \oplus E(K, IV+1)$,
$C_2 = P_2 \oplus E(K, IV+2)$,…

Decryption

$P_0 = C_0 \oplus E(K, IV)$,
$P_1 = C_1 \oplus E(K, IV+1)$,
$P_2 = C_2 \oplus E(K, IV+2)$,…

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



Cipher Feedback Mode (CFB)

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

Encryption

$C_0 = P_0 \oplus E(K, IV)$,

$C_1 = P_1 \oplus E(K, C_0)$,

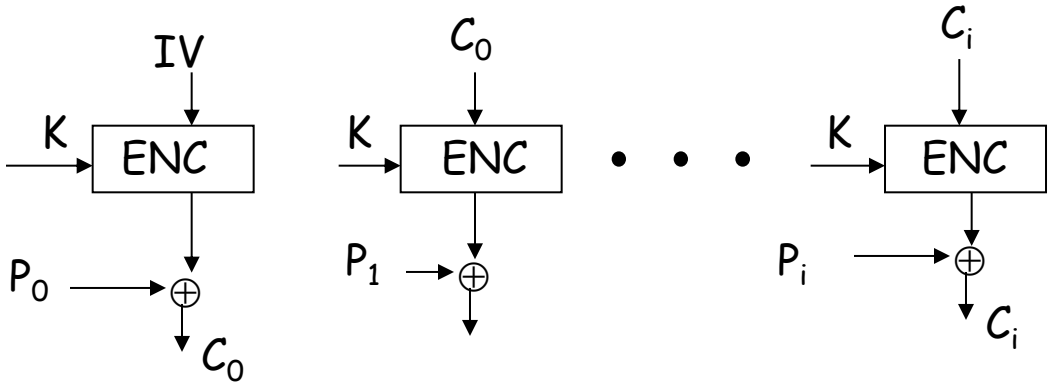
$C_2 = P_2 \oplus E(K, C_1)$,…

Decryption

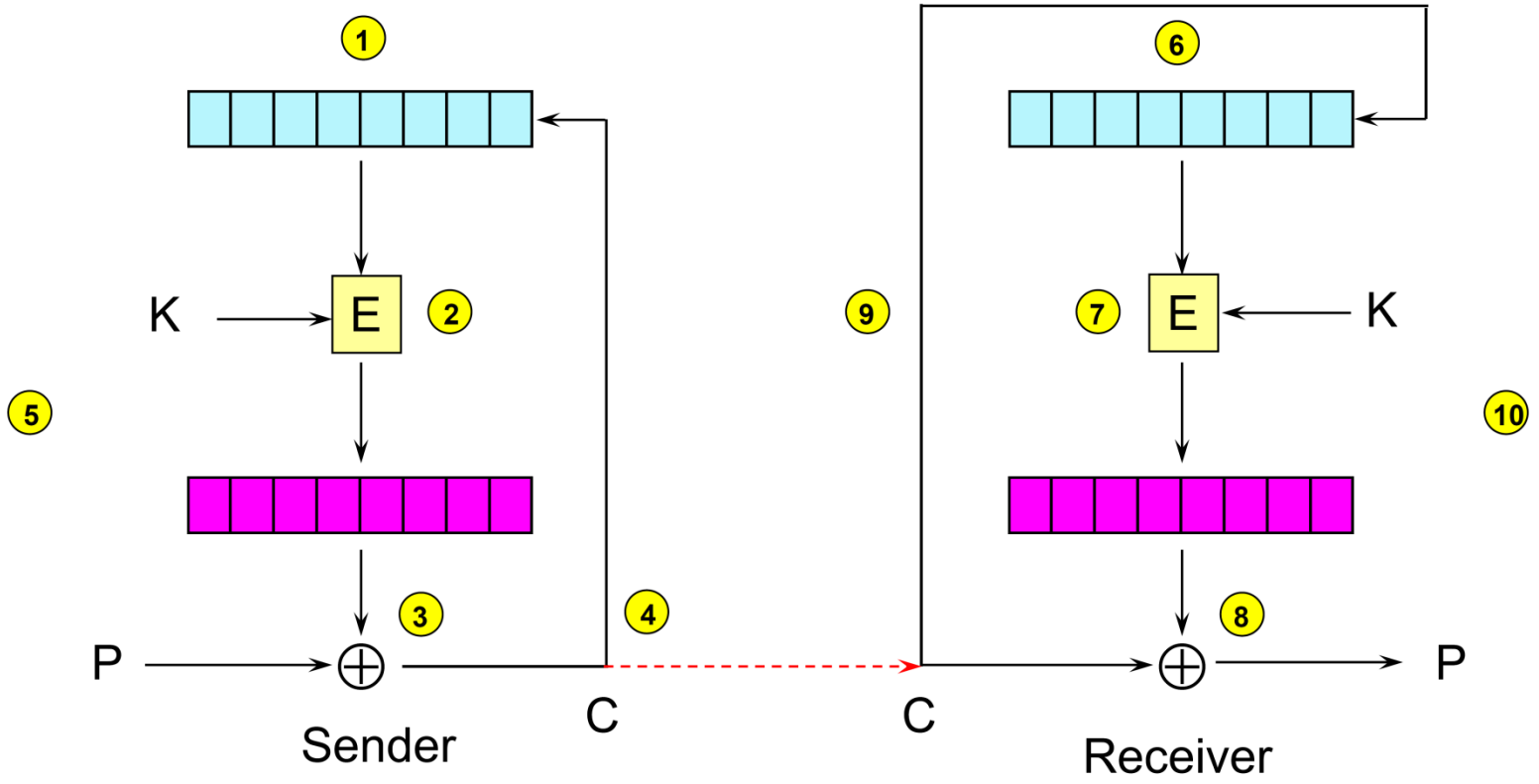
$P_0 = C_0 \oplus E(K, IV)$,

$P_1 = C_1 \oplus E(K, C_0)$,

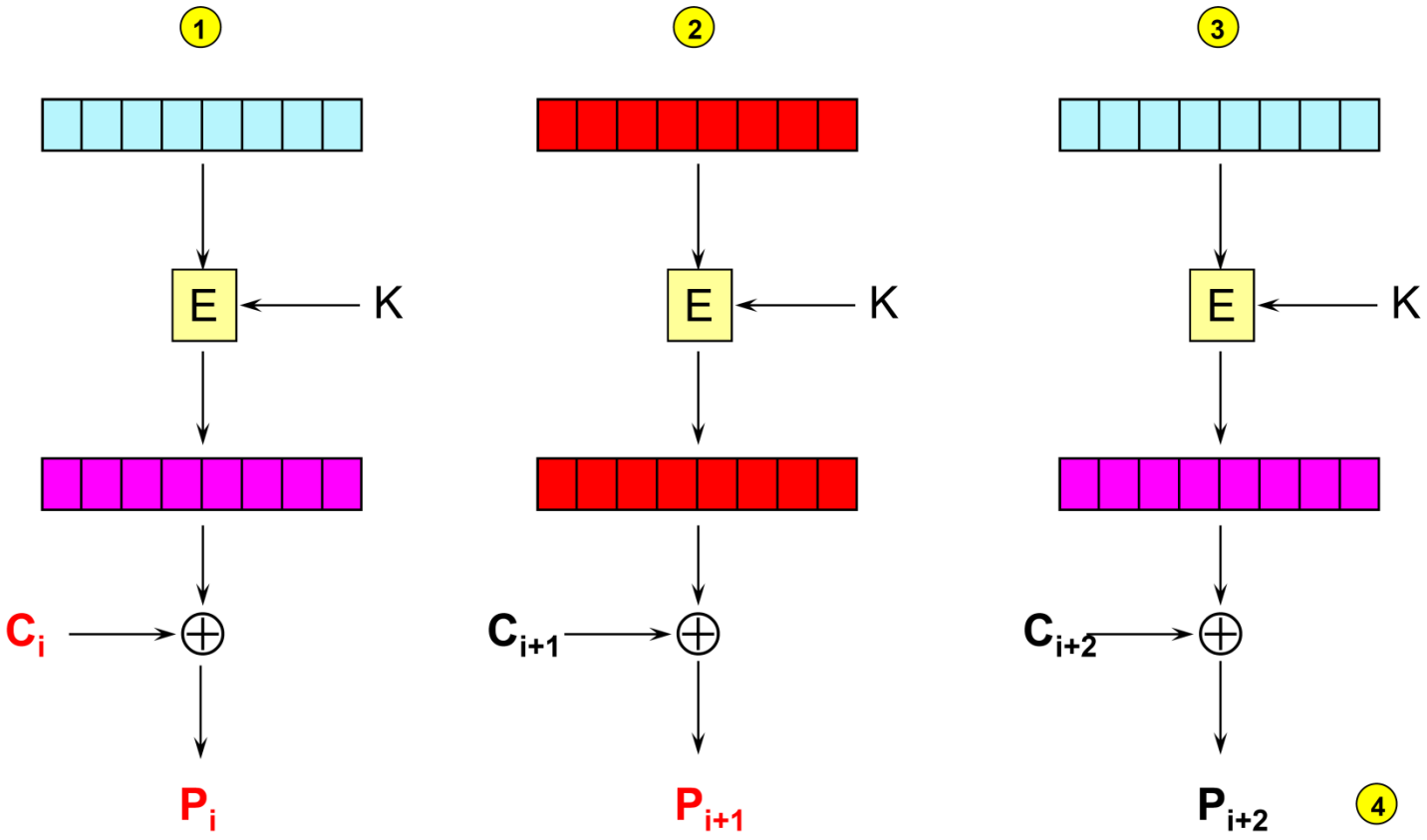
$P_2 = C_2 \oplus E(K, C_1)$,…



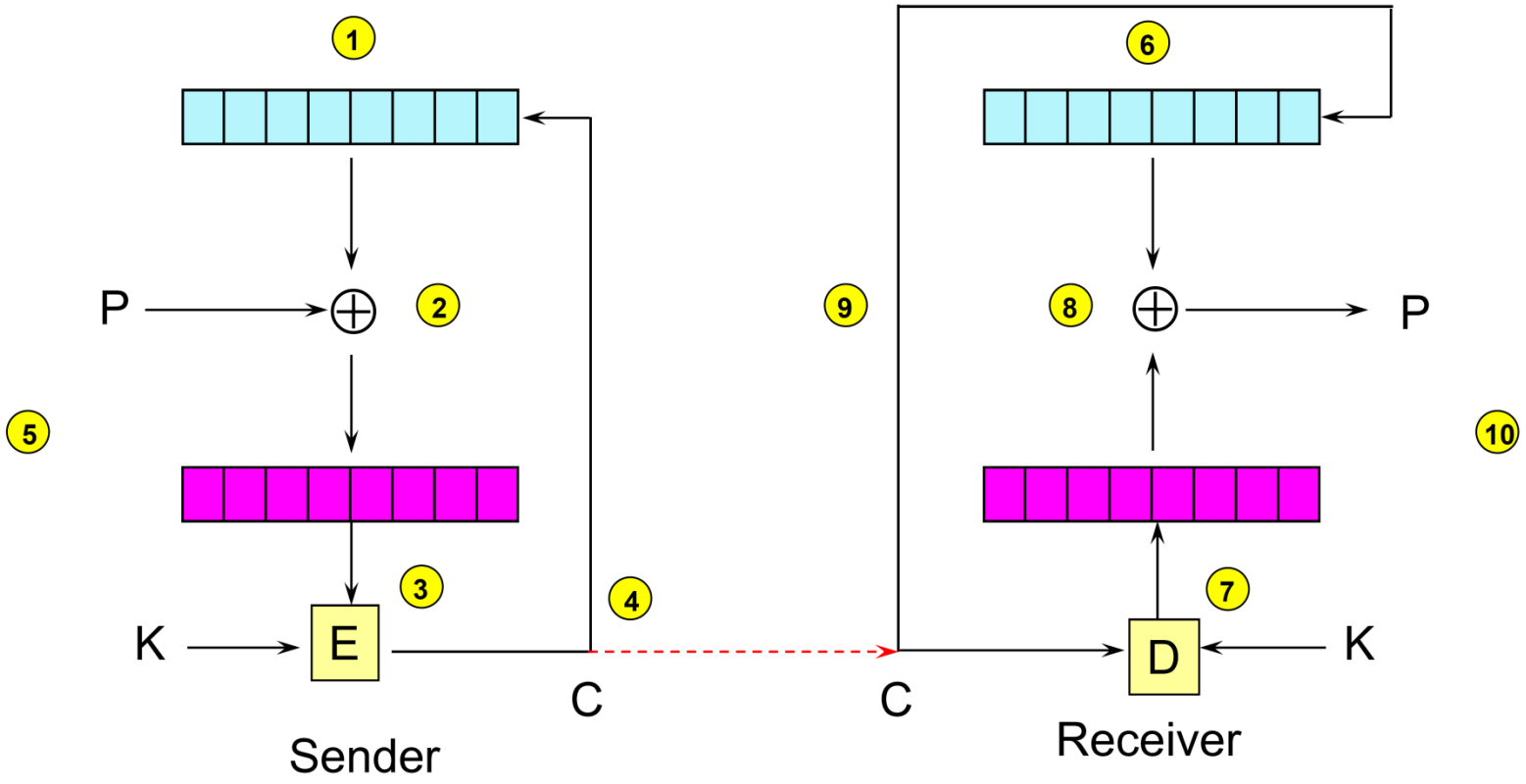
CFB Mode



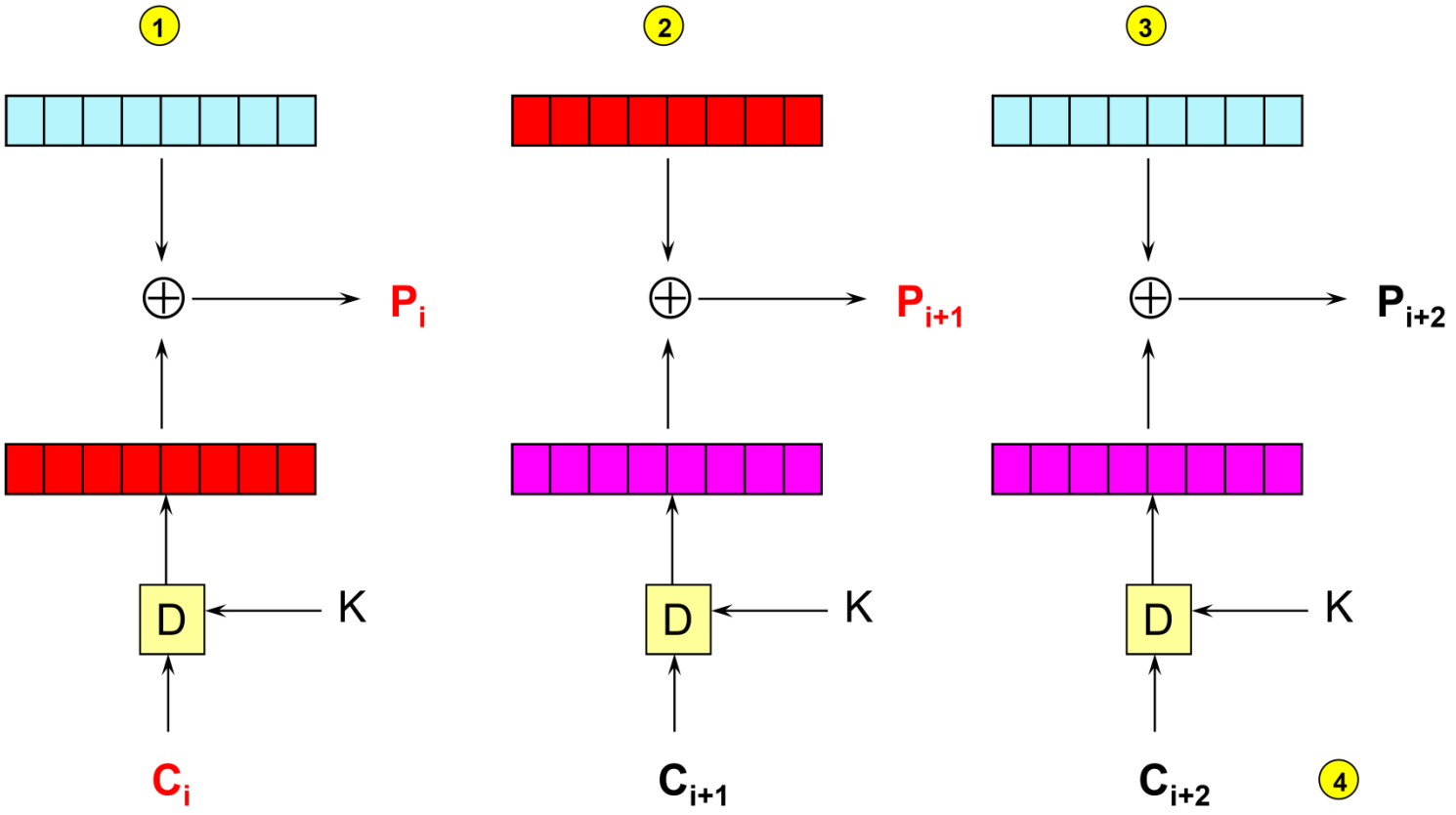
CFB Error



CBC Mode



CBC Error



Supplementary Materials

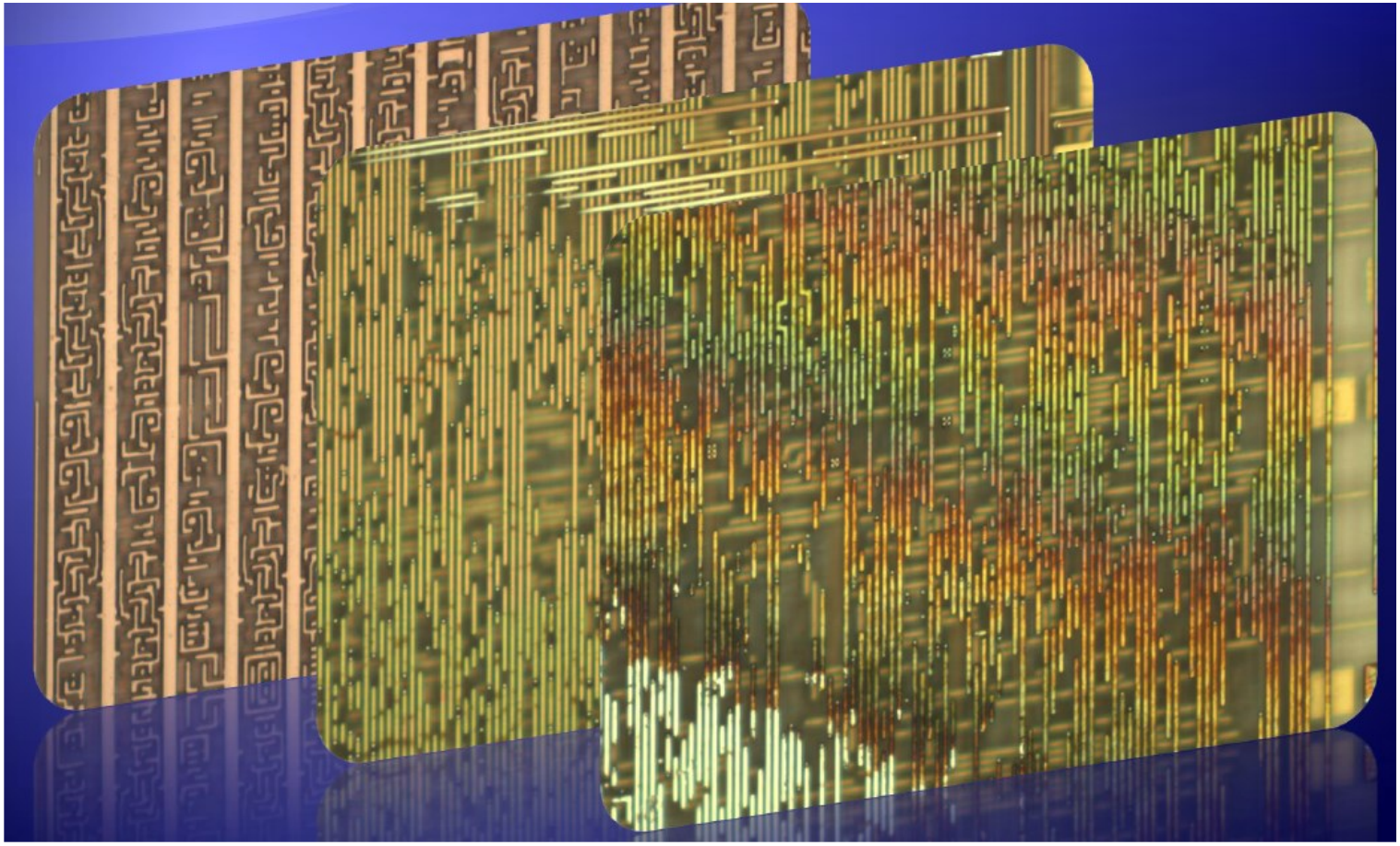
Practical Cipher Knowledge

- 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

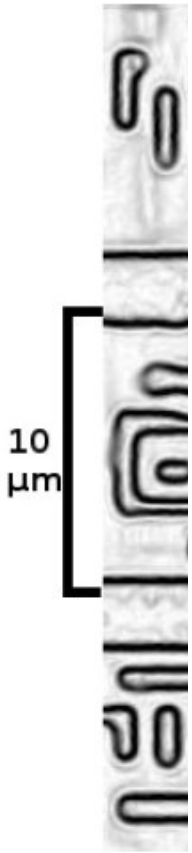
- Developed in 1995 (NXP Semiconductor) – Crypto1 algorithm
  - 48 bit key, stream cipher
- Used in a significant number of current systems
  - Access control
  - Travel
  - Closed payment
- Cipher kept secret…was securely used for a long time
- Researchers reverse engineered cipher by analysis of the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

Mifare Classic Metal Layers

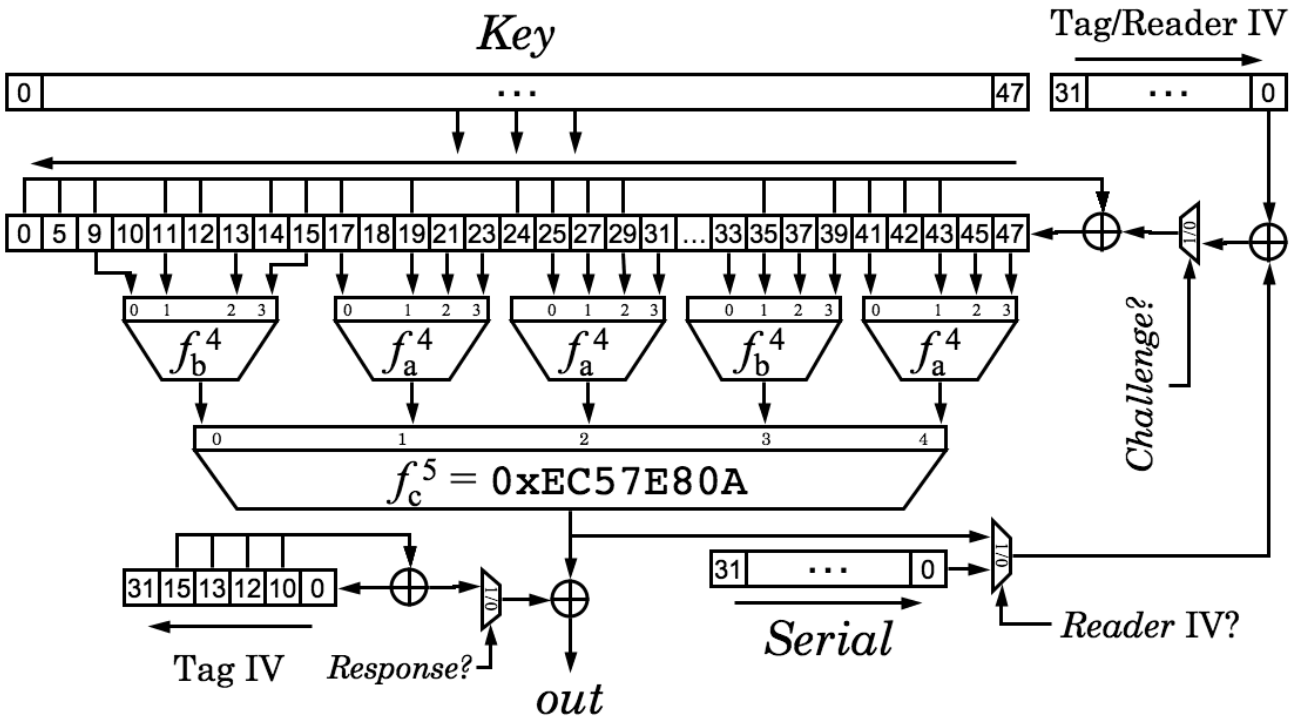


Nohl, et al (2008)

Reconstructing the Algorithm



Crypto1 Cipher



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

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

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

2008)

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

The end!

Any questions…