Hardware Security Week 2 - Cryptography in Hardware

26T1

Hammond Pearce

Slide material acknowledgements to
Ramesh Karri, Benjamin Tan,
JV Rajendran, Jason Blocklove
Christian Pilato, Luca Collini

RECAP

What is the “C I A” triad?

Hardware Security

- Hardware security and trust implies support for:
  - Data confidentiality
  - Data integrity
  - Authentication
  - Non-repudiation
- Symmetric key cryptography is the building block

Part 1: History

Cryptography - History

- Has been around for 2000+ years
- In 513 B.C., Histiaeus of Miletus, shaved a slave's head and tattooed a message there, then let the hair grow

Pencil & Paper Era

- What does this say?

GO CKI KQKSX NOVSLOBKDOVI DRKD REWKX

SXQOXESDI MKXXYD MYXMYMD K MIZROB

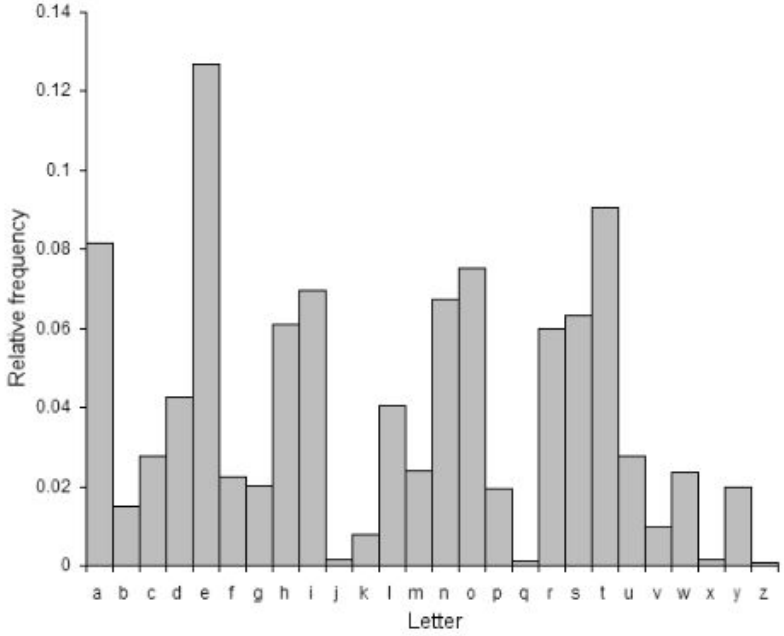
GRSMR REWKX SXQOXESDI MKXXYD BOCYVFO

It’s a caesar cipher

- How does it work?
- Simple monoalphabetic substitution (simple substitution)
- $c_i = E(p_i) = p_i + 3 \bmod 26$ (26 letters in the English alphabet)
  - E.g. change each letter to the third letter following it (circularly)
  - A → D, B → E, …. X → A, Y → B, Z → C.
  - Here, the key would be “3” or “D” (because D represents 3)

Breaking a substitution cipher

- Letters in english have different frequencies, “e” most common



Breaking a substitution cipher

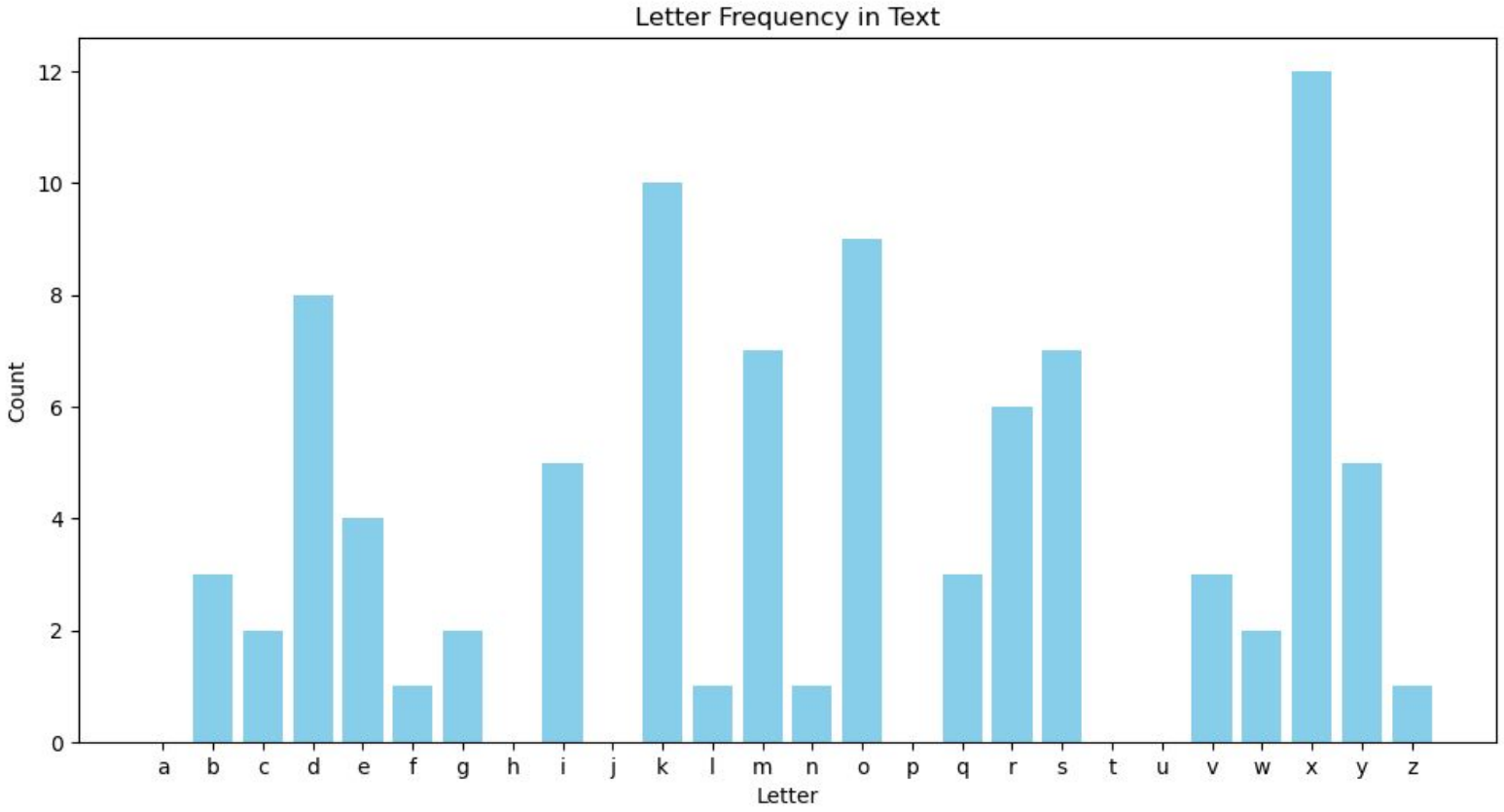
Secret text:

GO CKI KQKSX NOVSLOBKDOVI DRKD REWKX

SXQOXESDI MKXXYD MYXMYMD K MIZROB

GRSMR REWKX SXQOXESDI MKXXYD BOCYVFO

Our letter frequencies:



Breaking a substitution cipher

- “X” most common in ciphertext, “E” most common in english
- Difference of 7

ZH VDB DJDLQ GHOLEHUDWHOB WKDW KXPDQ

LQJHQXLWB FDQQRW FRQFRFW D FBSKHU

ZKLFK KXPDQ LQJHQXLWB FDQQRW UHVROYH

- nope!

Breaking a substitution cipher

“K” next most common:

MU IQO QWQYD TUBYRUHQJUBO JXQJ XKCQD

YDWUDKYJO SQDDEJ SEDSESJ Q SOFXUH

MXYSX XKCQD YDWUDKYJO SQDDEJ HUIEBLU

- nope!

Breaking a substitution cipher

“O” next most common:

WE SAY AGAIN DELIBERATELY THAT HUMAN

INGENUITY CANNOT CONCOCT A CYPHER

WHICH HUMAN INGENUITY CANNOT RESOLVE

- Broke it with 3 attempts

Class exercise:

Break the following text (words re-grouped to make it harder!)

UOMYQ FAFTQ YQQFU ZSBAU ZFNGF KAGIQ DQZAF FTQDQ

UOMZA ZXKTA BQKAG IQDQZ AFUZF QDOQB FQP

Class exercise:

Break the following text (words re-grouped to make it harder!)

UOMYQ FAFTQ YQQFU ZSBAU ZFNGF KAGIQ DQZAF FTQDQ

UOMZA ZXKTA BQKAG IQDQZ AFUZF QDOQB FQP

Lazy mode: https://www.boxentriq.com/code-breaking/vigenere-cipher

Caesar's problem

- Key is too short
  - 1-char key: monoalphabetic substitution
  - Can be found by exhaustive search or by frequency analysis
  - Statistical frequencies not concealed well by short keys
- Solution:
  - Make the key longer
  - N-char key (n>=2) - polyalphabetic substitution
  - Makes exhaustive search much more difficult
  - Statistical frequencies concealed better, making cryptanalysis harder!

Example: Vigenère Table

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

Vigenère Table

- The first row has the 26 english letters
- The next row has the letters shifted to the left one position
- This continues until every rotation is done

Vigenère Table: Encryption

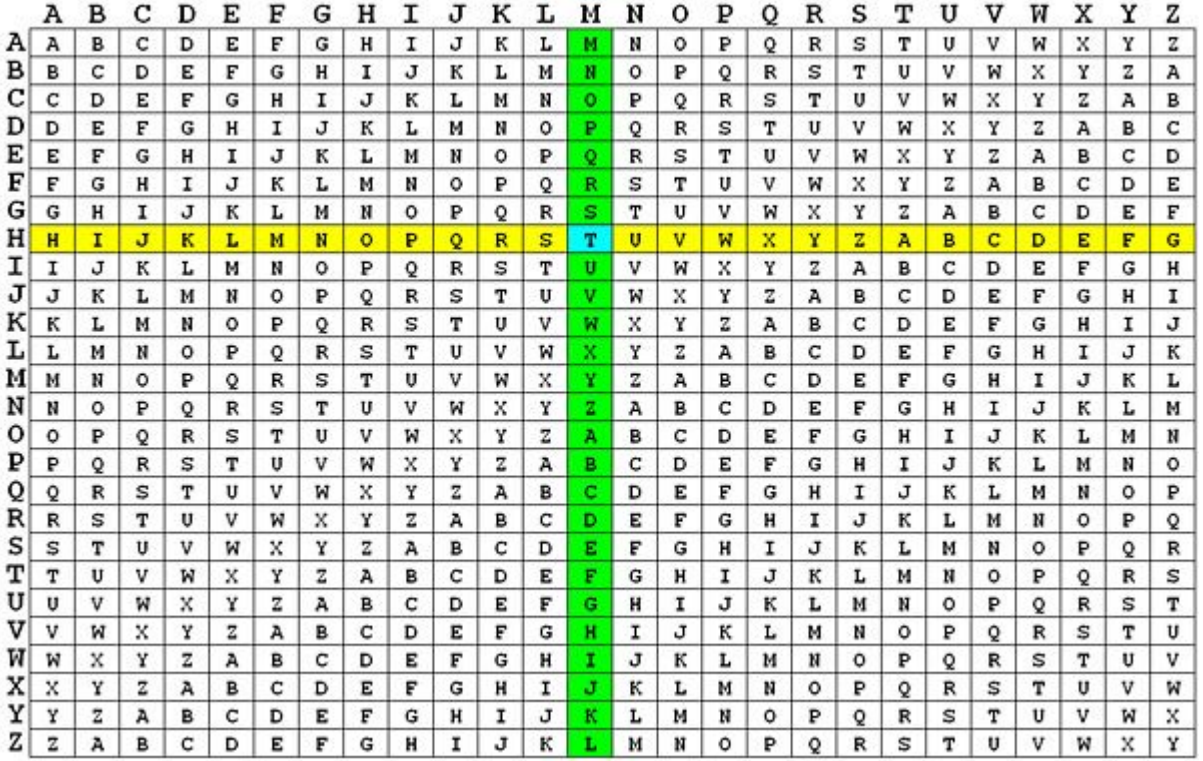
- Decide a plaintext: “MOST HARDWARE IS PRETTY COOL”
- Decide a key: “HAMMOND”
- Repeat the key under the plaintext as follows:

M	O	S	T	H	A	R	D	W	A	R	E	I	S	P	R	E	T	T	Y	C	O	O	L
H	A	M	M	O	N	D	H	A	M	M	O	N	D	H	A	M	M	O	N	D	H	A	M

Vigenère Table: Encryption

- To encrypt, pick plaintext letter and its corresponding key

- Look up in table and replace:



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

Vigenère Table: Encryption example

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

- Decryption is the reverse
  - (take key row, find cipher letter - column is plaintext letter)

Caesar and Vigenère: autosolvers

Break Caesar cipher:

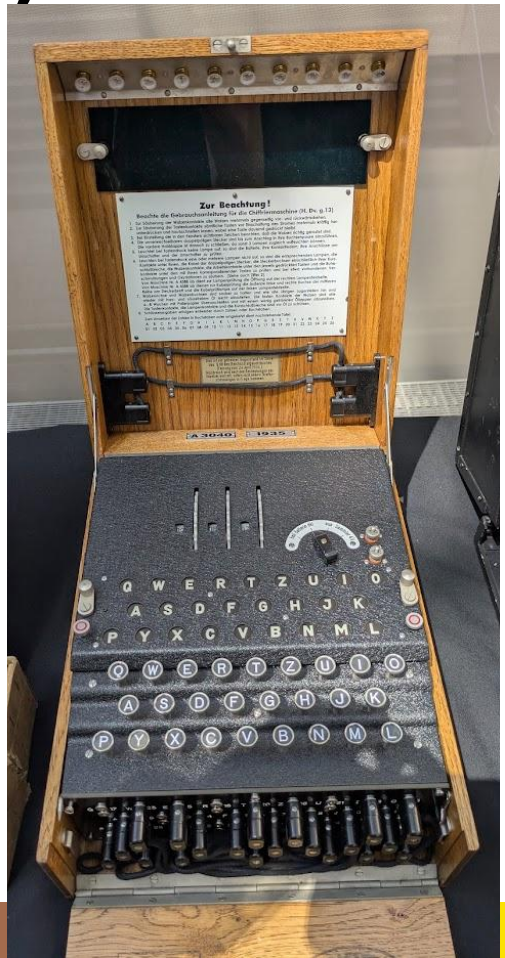
- https://www.dcode.fr/caesar-cipher

Break Vigenère cipher:

- https://www.boxentriq.com/code-breaking/vigenere-cipher
  - Not that good, helps to set the right length for the key
  - Better solvers exist

Crypto - Mechanical Era (Enigma)

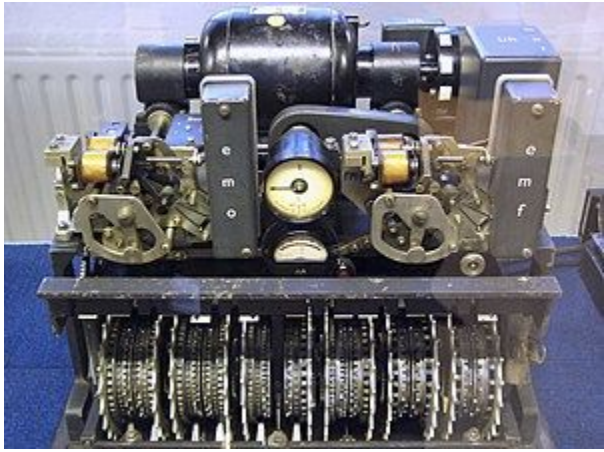
- ~1900, realisation: cryptography has math and statistics roots
- ~1917: Germany creates mechanical device to encrypt text
  - **Enigma** machine → supposedly unbreakable
  - A few polish mathematicians got a working copy, later sold to Britain
- ~1932: Polish cryptographers break first Enigma codes
  - Not all codes vulnerable
- ~1939: Polish cryptographers assist with UK to break code
  - Hut 6 at Bletchley Park
- By end of WW2, Enigma cracked: German messages visible
  - Estimated that cut the war length by about a year
  - British kept it secret until the last working Enigma!



A pair of enigma came to UNSW last year!

Rotor settings

Crypto - Mechanical Era (Tunny/Lorenz)



- Also a German development during WW2
- Tunny/Lorenz machines were in-line teleprinter attachments
  - Enigma used by German military,
  - Tunny by German High Command
- Tunny implemented a stream cipher
- Key XOR'd with plaintext to produce cipher
  - $C = P \oplus K_s$
- Tunny broken at Bletchley Park in Jan 1942…

XOR

Symbol	A	B	Q
	0	0	0
	1	0	1
	0	1	1
	1	1	0
$Q = A \oplus B$			

Breaking Tunny

- On 30 August 1941, a 4,000 character message was transmitted with errors
- Recipient asked for retransmission
- Message retransmitted with *same key* but *differing contents*
  - Start of message same
  - Varied wildly after operator began using abbreviations
- First message → $C = P \oplus K_s$, second message → $C' = (P<<1) \oplus K_s$
- Thus → $C \oplus C' = P \oplus (P<<1)$
- Once P is known, K_s revealed
- This keystream led to understanding how Tunny worked → codes broken

Cryptography - Modern Era

- Shannon's Information Theory
  - First major development in crypto theory after WWII
- Shannon introduced the one-time pad and presented theoretical analysis of the code
- OTP are (and continue to be) **unbreakable** if used correctly
  - Considered the "perfect cipher"
  - Used extensively during the cold war

One-Time Pads (OTP)

- OTP is a variant of Vigenère tables
  - Fixes the main problem with VT: the key might be too short
- OTP has large, non repeating sets of long keys on pads/sheets
- Sender/receiver have identical pads
- Example:
  - 300-char message to send, 20 character key per sheet
  - Use and tear off 300/20 = 15 pages from the pad

One-Time Pads (OTP) - Hardware

- Encryption
  - Sender writes letters of consecutive character keys above plaintext P
  - Sender enciphers P using Vigenère table (or other pre-arranged chart)
  - Sender destroys used/keys sheets
  - Sender transmits (e.g. using radio)

- Decryption
  - Receiver receives ciphertext (e.g. using radio)
  - Receiver uses Vigenère table (or other pre-arranged chart)
  - Receiver uses same set of consecutive keys from same pages
  - Receiver destroys used keys/sheets

OTP from UNSW visit

OTP Example

https://www.cryptomuseum.com/crypto/img/301277/005/full.jpg

OTP Example: R-353 captured from East German Spy in the Netherlands

https://www.cryptomuseum.com/crypto/otp/index.htm

OTP weaknesses

- Need a key as long as the message
- Key must always change (and destroyed after use)
- Perfect synchronization between sender and receiver
  - Intercepted/destroyed messages can cause de-sync
- Need lots of keys
- Need to distribute pads securely
  - Printing, distribution, storing, accounting
- Frequency distribution needs to be perfectly flat
  - Non-flat distribution facilitates breaking

OTP weaknesses (more!)

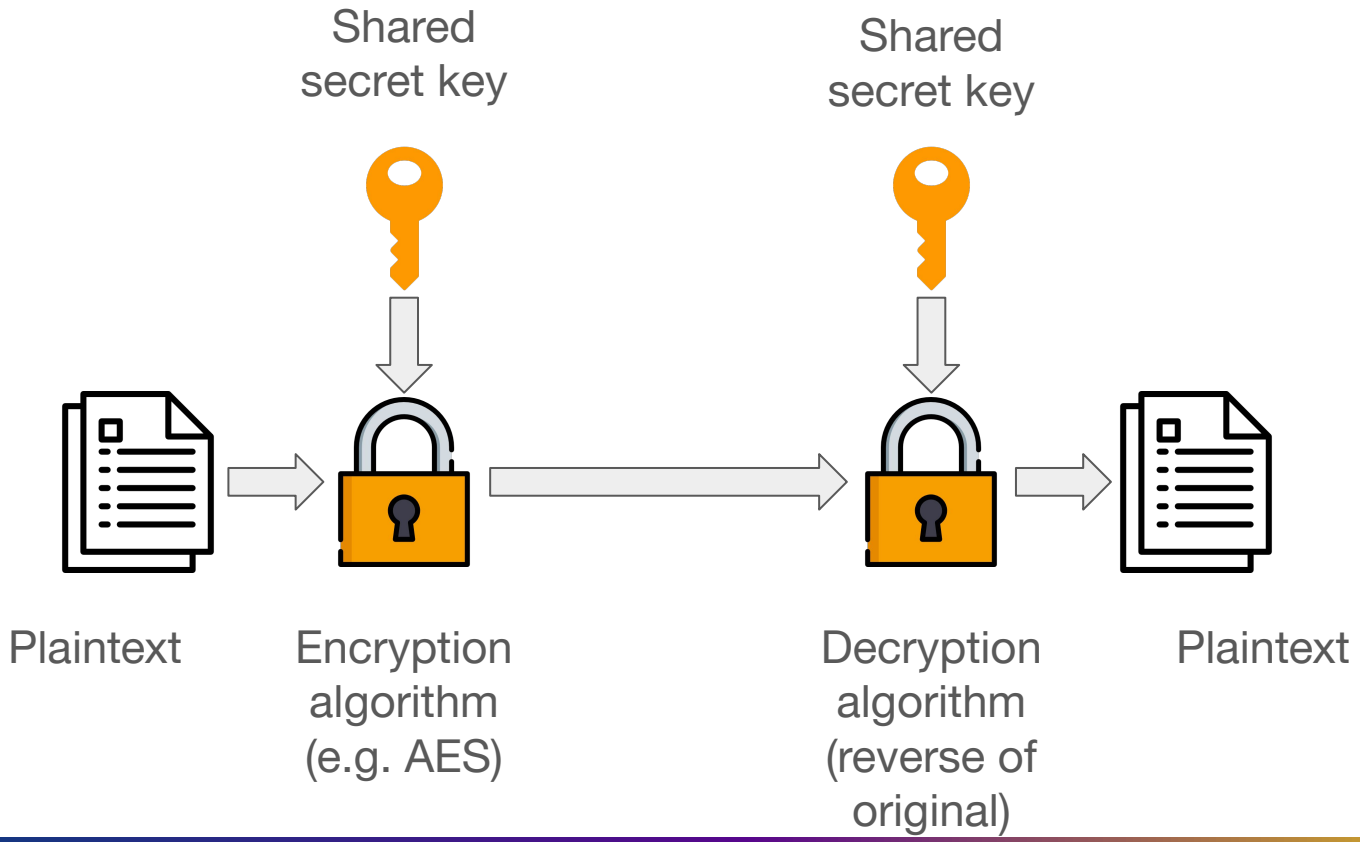
- Provenance
  - No guarantee that sender/receiver are the same person
- Key distribution
  - No method to know if keys have been compromised
- Potential for errors

But OTPs are still in use! Why?

Part 2: Computerized Crypto

Symmetric vs Asymmetric Keys

Symmetric key



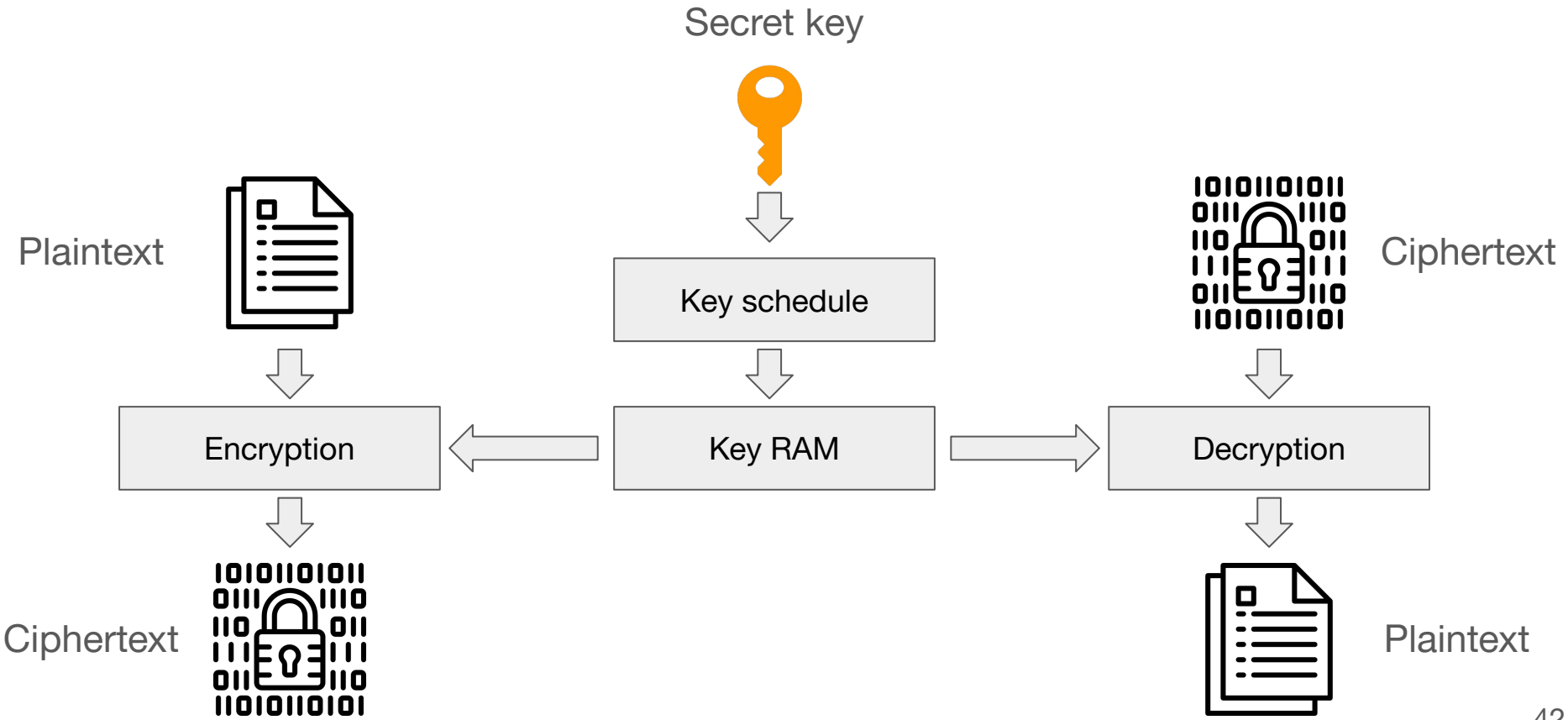
Symmetric key

- Oldest and most intuitive form of cryptography
    - OTP, Enigma, etc - Symmetric key
- Sender and receiver share a secret key
- Sender uses key to encrypt plaintext to ciphertext
- Receiver uses key to decrypt ciphertext to plaintext

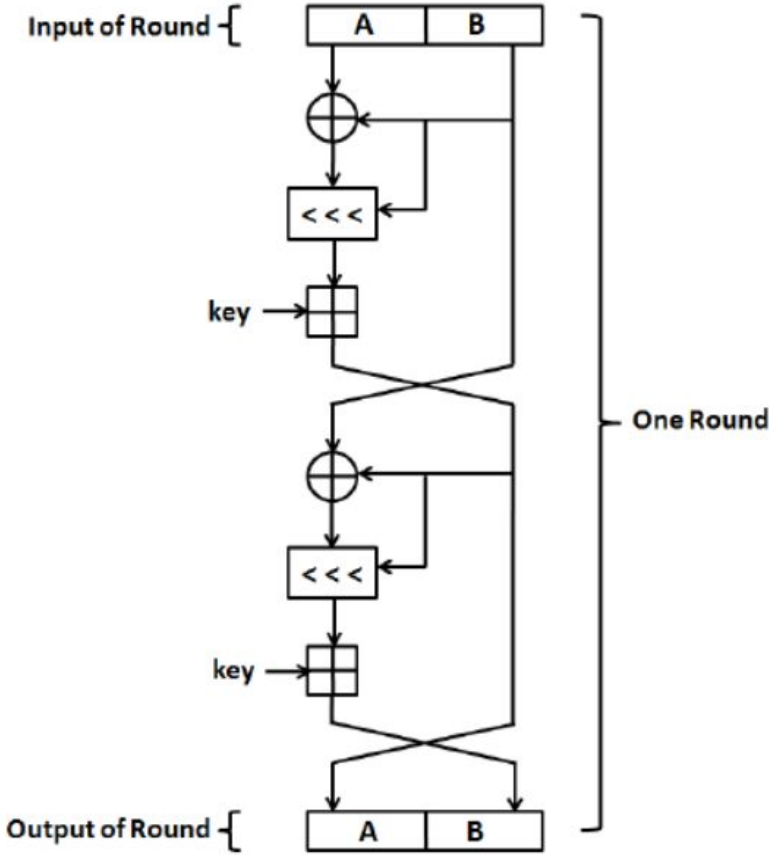
Examples: Block ciphers

- Symmetric block cipher (XOR?)
  - Encrypt (plaintext block, key) → ciphertext block
  - Decrypt (ciphertext block, key) → plaintext block
  - Encryption key = Decryption Key
- E.g. Data Encryption Standard (DES)
  - 64-bit plaintext block, 56-bit secret key
- E.g. Advanced Encryption Standard (AES)
  - 128-bit plaintext block, 128-bit secret key (usually)

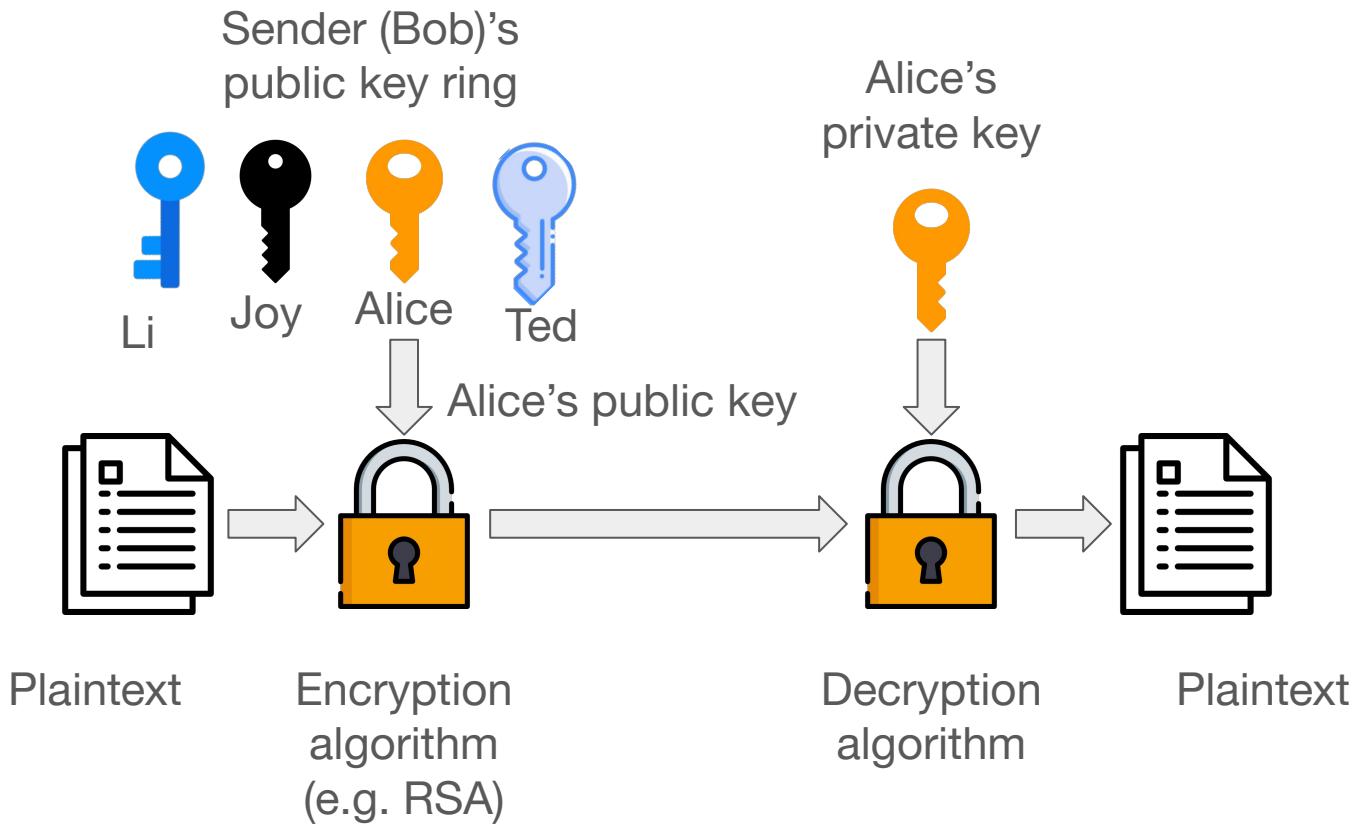
Symmetric Block Ciphers



Example block cipher (RC5):



Asymmetric key



Asymmetric key

- Two keys: one for encryption, one for decryption
- Rely on “one-way functions”
  - Mathematical operations that are easy to do forwards,
  - But difficult to do in reverse
- First public proposal: Whitford Diffie / Martin Hellman
  - 1976/1977 paper, “New Directions in Cryptography”
  - One-way function: modulo operations
  - Diffie-Helman Key Exchange
- 1969 “original” discovery by GCHQ researchers

Diffie-Helman Key Exchange

- Construct a secret (symmetric key)
- Do this over insecure channel without sending secrets

$$(g^a \bmod p)^b \bmod p = (g^b \bmod p)^a \bmod p$$

g, p: public values
a: Alice's secret, Alice sends:
b: Bob's secret, Bob sends:

Key derived from total function

RSA

- 1977: Ron Rivest, Adi Shamir, Leonard Adleman created one-way function to extend DH
  - GCHQ: Equivalent system in 1973
- Based on difficulty to factorise large numbers
- Based on observation that it is practical to find three large integers *e*, *d*, *n*, such that for all integers *(0 ≤ m < n)*:

$$(m^e)^d \equiv m \pmod{n}$$

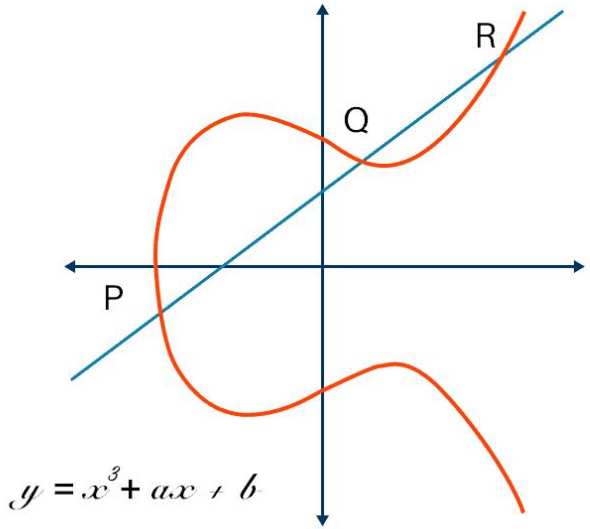
- Knowing *e* and *n*, and often even *m*, it is still difficult to find *d*

Elliptic Curve Cryptography

- Proposed independently in 1985 by N. Koblitz and V. Miller
- Uses curves instead of prime factorisation
- Shorter keys, harder math:

Key Comparison

ECC Key size	RSA Key size	Key Ratio
163	1024	1:6
256	3072	1:12
384	7680	1:20
512	15360	1:30



- The original paper was rejected!
- Slow adoption: not widely in use till late 1990s and 2000s

Why not Asymmetric Encryption?

- Maths is extremely difficult, computationally expensive
  - Logarithms, Curves - hard!
- In contrast, symmetric encryption:
  - Bitwise operators, XOR, substitutions and permutations
- Keys tend to be much larger:
  - E.g. RSA-2048 provides same security as AES-256
  - But key is 8 times larger, and the computation is much more difficult!

So…

Asymmetric key crypto typically used to protect symmetric keys

This course will focus on implementations of symmetric key cryptography

Other courses cover the math and theory of both asymmetric and symmetric key cryptography!

Break (1/2): 3 minutes

Break activity: Inspect the HTTPS certificate used by UNSW

What types of crypto does it use?

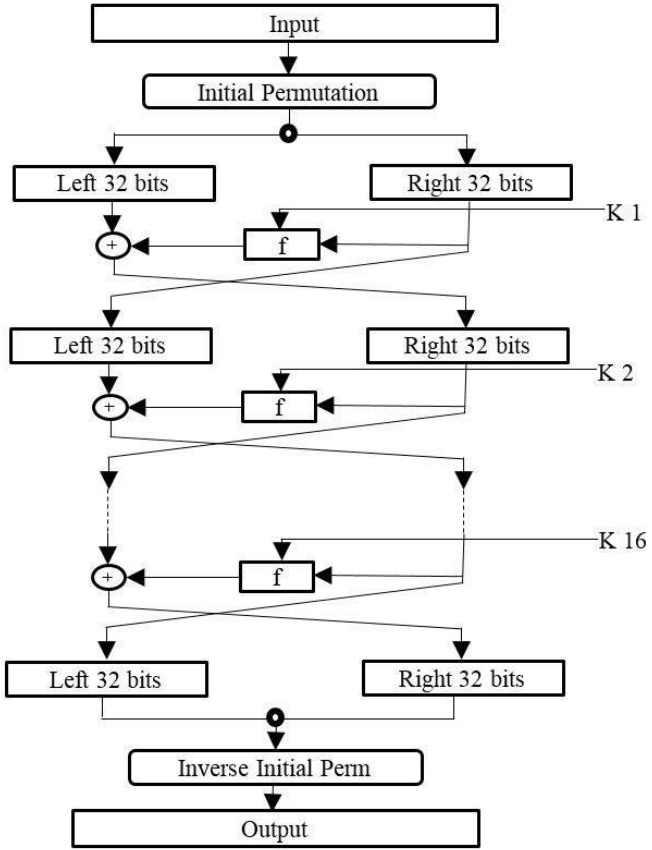
Data Encryption Standard (DES)

- Symmetric key algorithm
- Published standard 1977
- Somewhat controversial, why?
- 1999 - first successful attack by EFF
- 2002 - replaced by AES
- 2018 - finally retired Triple DES / 3DES

DES Encryption

1. Permute input (IP)
2. Split into L/R
3. New L = old R
4. New R = old L XOR F(old R, RK_n)
5. In 16th round, take pre-out and
   Permute through IP^{-1}

6. Round keys for encryption: K1-K16
7. Round keys for decryption: K16-K1



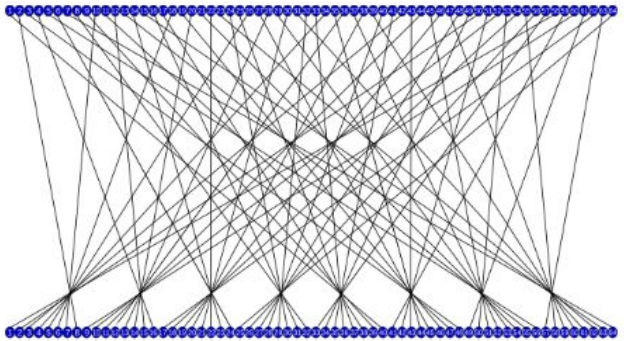
Initial Permutation (IP)

- Don’t have much (any?) cryptographic benefit

- But: Hardware benefit?
- Supposedly arranged bits in a way that made it easier to parallelize and optimize for IBM mainframes

IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7



Key generation K

- Key → 56 bits (drop parity bits)
- Permutation
- Rotation (each 28-bit half)
- Compression → 48 bits

Permutation Table

	1	2	3	4	5	6	7	8
0	57	49	41	33	25	17	9	1
1	58	50	42	34	26	18	10	2
2	59	51	43	35	27	19	11	3
3	60	52	44	36	63	55	47	39
4	31	23	15	7	62	54	46	38
5	30	22	14	6	61	53	45	37
6	29	21	13	5	28	20	12	4

Key Compression Table

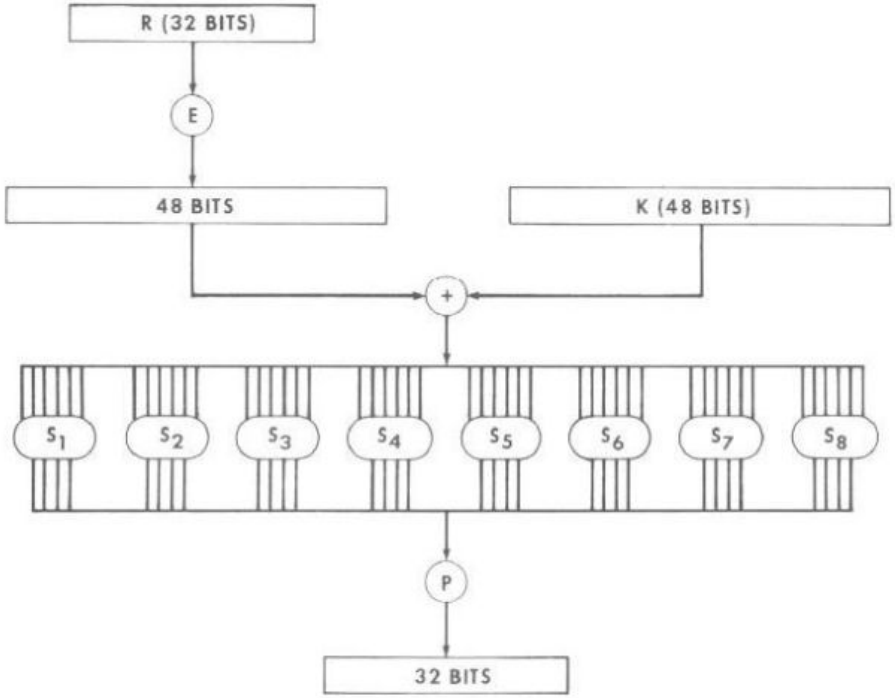
	1	2	3	4	5	6	7	8
1	14	17	11	24	01	05	03	28
2	15	06	21	10	23	19	12	04
3	26	08	16	07	27	20	13	02
4	41	52	31	37	47	55	30	40
5	51	45	33	48	44	49	39	56
6	34	53	46	42	50	36	29	32

Bits Rotation Table

Number of Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number of Left rotations	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

F Function

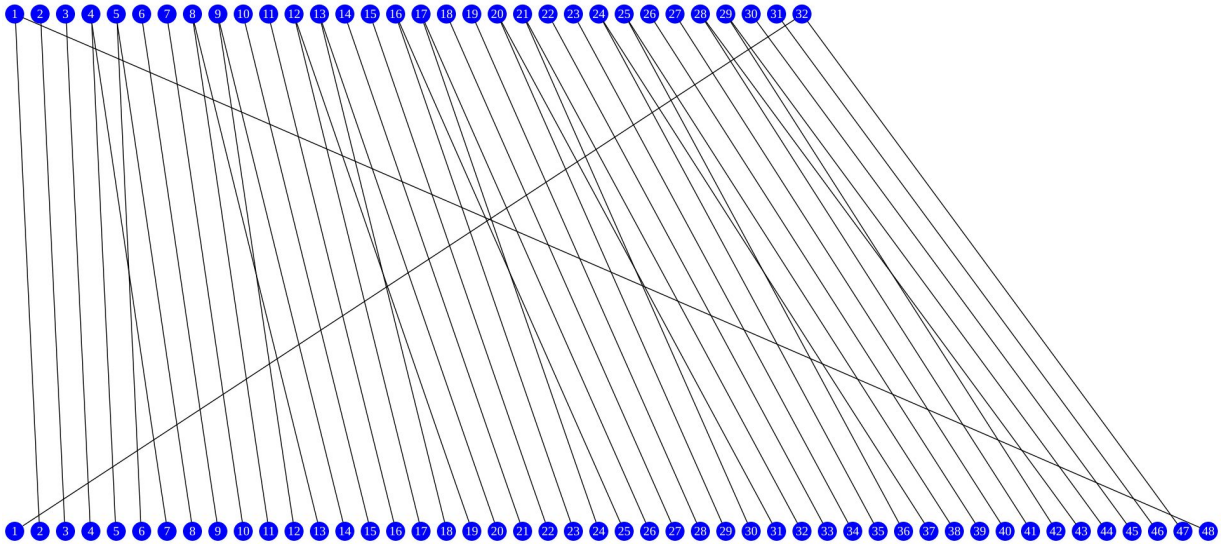
1. Expand R
2. XOR with RK_n
3. Split into 8 sets of 6-bits
4. Look up 6-bit values in S-Box
5. Concatenate
6. Permute through P



Expansion (E)

E

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1



S-Box

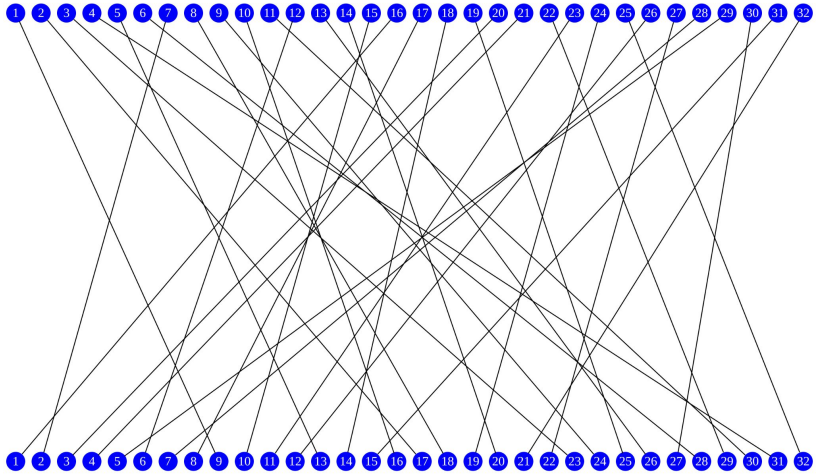
- Six-bit input: A0, A1, A2, A3, A4, A5
- Four-bit output
- Use A0,A5 as row address; A1,A2,A3,A4 as column
- Return value at row/column
- All 8 S-Boxes unique
- Example for S_1: 010011 → 0110 (6)

S_1	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0yyyy1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
1yyyy0	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
1yyyy1	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

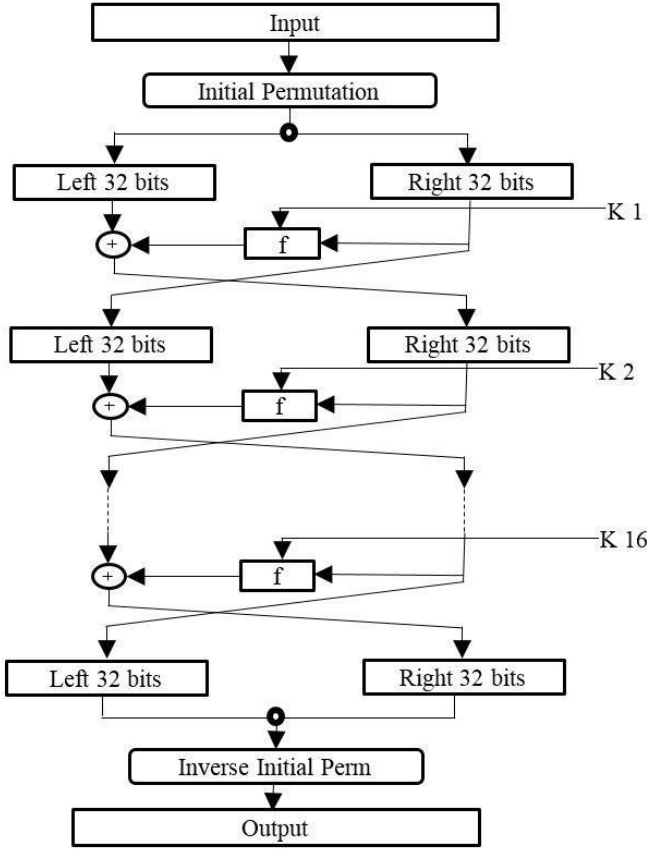
Permutation (P)

P

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25



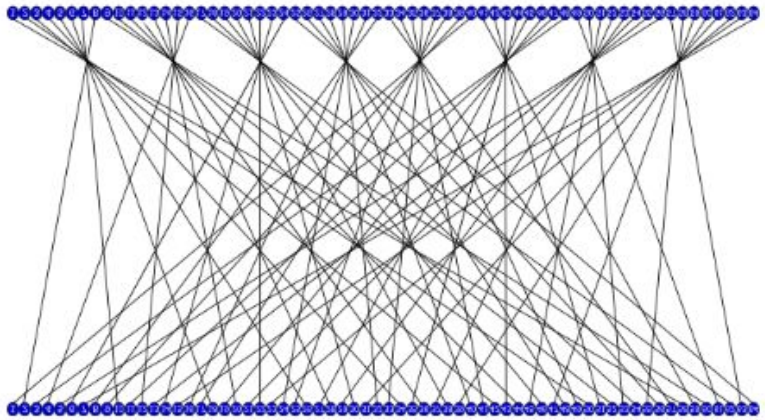
DES (again)


Input
Initial Permutation
Left 32 bits
Right 32 bits
K 1
f
Left 32 bits
Right 32 bits
K 2
f
K 16
f
Left 32 bits
Right 32 bits
Inverse Initial Perm
Output

Final Permutation: Inverse of Initial

IP^{-1}

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25



Inverting DES: Decryption

- $L_n = R_{n-1}$
- $R_n = L_{n-1}$ XOR F (R_{n-1}, RK_n)

Becomes….

- $R_{n-1} = L_n$
- $L_{n-1} = R_n$ XOR F(L_n, RK_n)

Decryption is the inverse of encryption

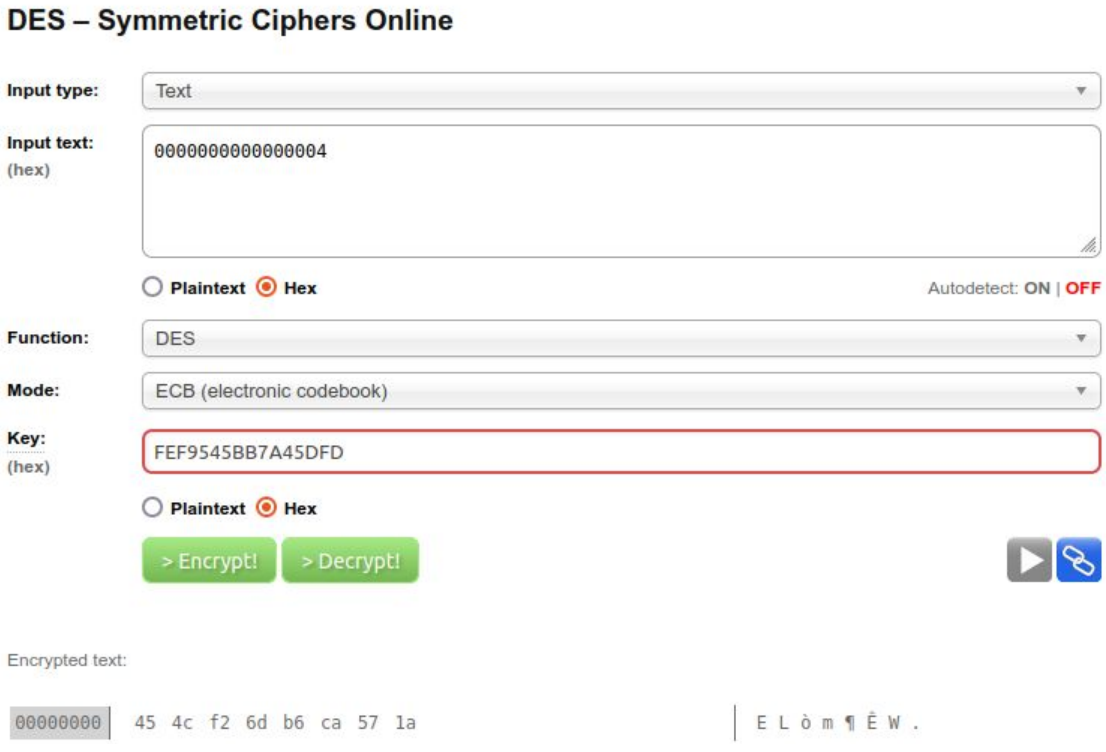
DES Encryption	DES Decryption
Initial Permutation	Inverse Initial Permutation
Ln = Rn-1	Rn-1 = Ln
Rn = Ln-1 XOR F(Rn-1, Kn)	Ln-1 = Rn XOR F(Ln, Kn)
Inverse Initial Permutation	Initial Permutation

- Initial and Inverse Initial Permutations don't matter
  - (They're undone at the end of encryption and decryption)

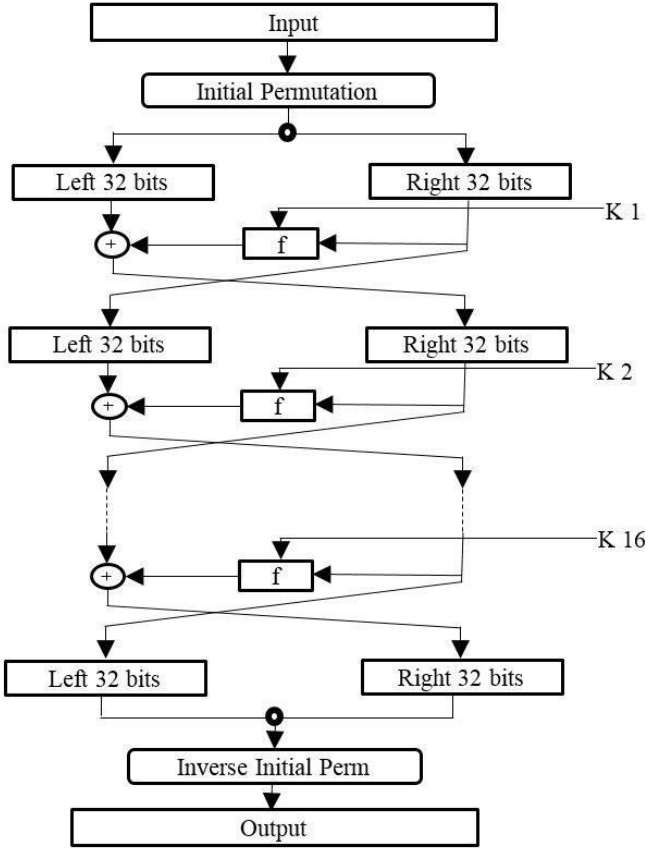
- Property of XOR: If X = P XOR K, then P = C XOR K
  - Nice design feature: less hardware required

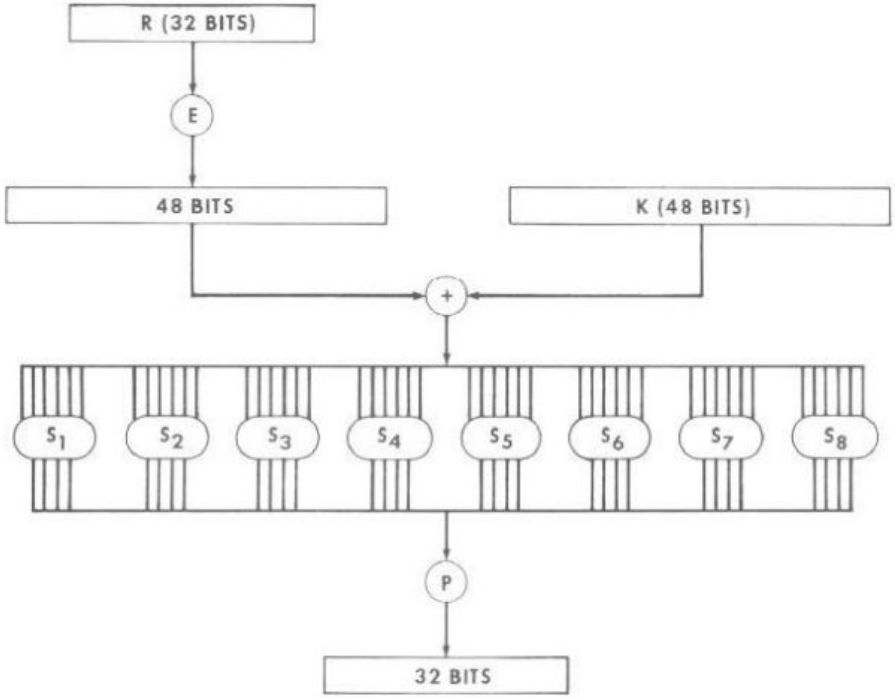
DES Example

http://des.online-domain-tools.com/



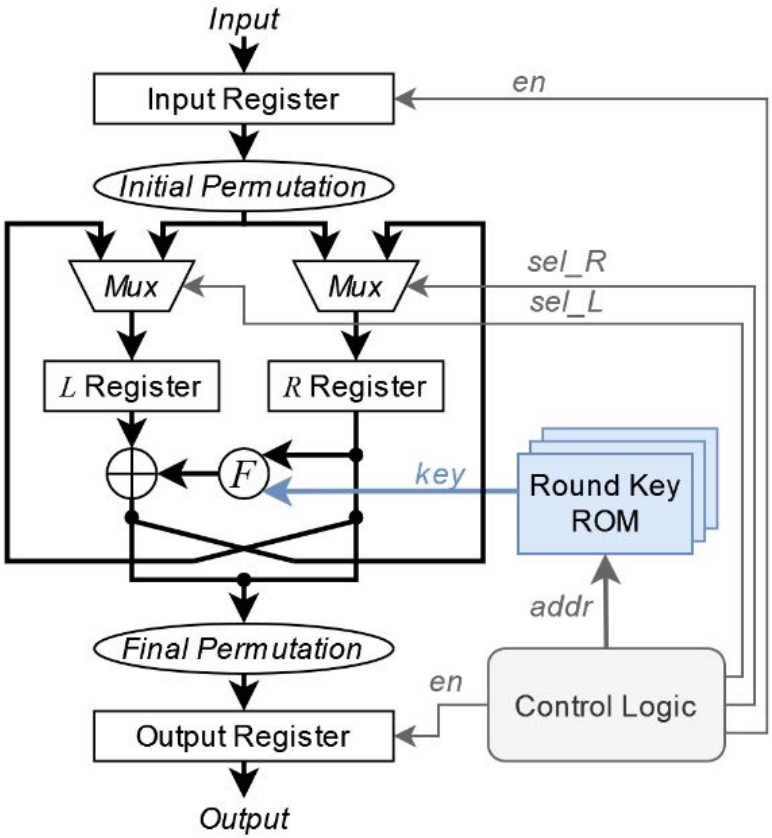
DES in Hardware (Whiteboarding)





DES in Hardware (Whiteboarding)

DES in Hardware (Whiteboarding)



DES in Hardware (Example)

`make run_sim` in `examples/spi_des`

Attacking DES

- Only practical attack: brute force?
- What did the NSA do???

Attacking DES

- Only practical attack: brute force?
- What did the NSA do???
  - They assisted with the development
  - Provided the S-box values
  - Insisted on 56-bit key length

Attacking DES

- Only practical attack: brute force?
- What did the NSA do???
  - They assisted with the development
  - Provided the S-box values (Good!)
  - Insisted on 56-bit key length (Bad!)
- S-box values prevented Differential Cryptanalysis
  - Classified knowledge at the time
- Other types of attack
  - Linear Cryptanalysis
  - Davies attack
- These attacks work better on other block ciphers

Extending DES

- Double DES:
  - C = ENC(k2, ENC(k1, P))
  - Doesn't increase security as much as you'd think
  - "Meet in the middle attack"
  - Given ciphertext, plaintext, perform DEC(k1, C) = END(k2, P)
  - Doubles key space, attack space 2^56 → 2^57
- Triple DES
  - C = ENC(k3, DEC(k2, ENC(k1, P)))
  - Option 3: k3, k2, k1 unique
    - Attack: 2^(2x56) = 2^112
  - Option 2: k3 == k1
    - Attack: 2^80
  - Option 1: k3 == k2 == k1 (backwards compatible)

Break (2/2): 3 minutes

Break activity: Talk to your neighbour -

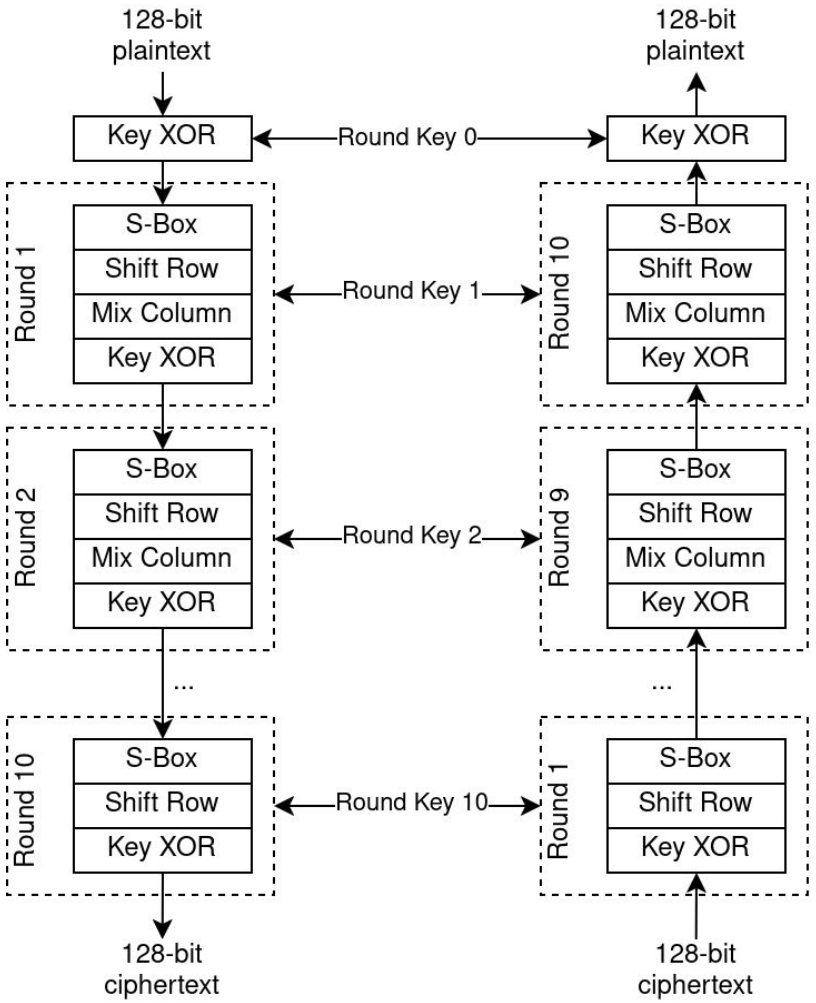
hi
how long would your laptop take to try 2^56 keys?

Advanced Encryption Standard (AES)

- Symmetric key algorithm
- First published 1998 by Joan Daemen and Vincent Rijmen
- Established as standard in 2001
- Three versions:
  - AES-128: 128-bit data, 128-bit key
  - AES-192: 128-bit data, 192-bit key
  - AES-256: 128-bit data, 256-bit key
  - (We'll focus on AES-128 - the differences are slight)

AES Overview

1. Initial round key:
    a. XOR with key
2. 9 rounds:
    a. S-Box (Simple substitution)
    b. Shift Rows (Transposition)
    c. Mix Column (Linear mixing step)
    d. XOR with key
3. Final round
    a. No Mix Column



AES operations

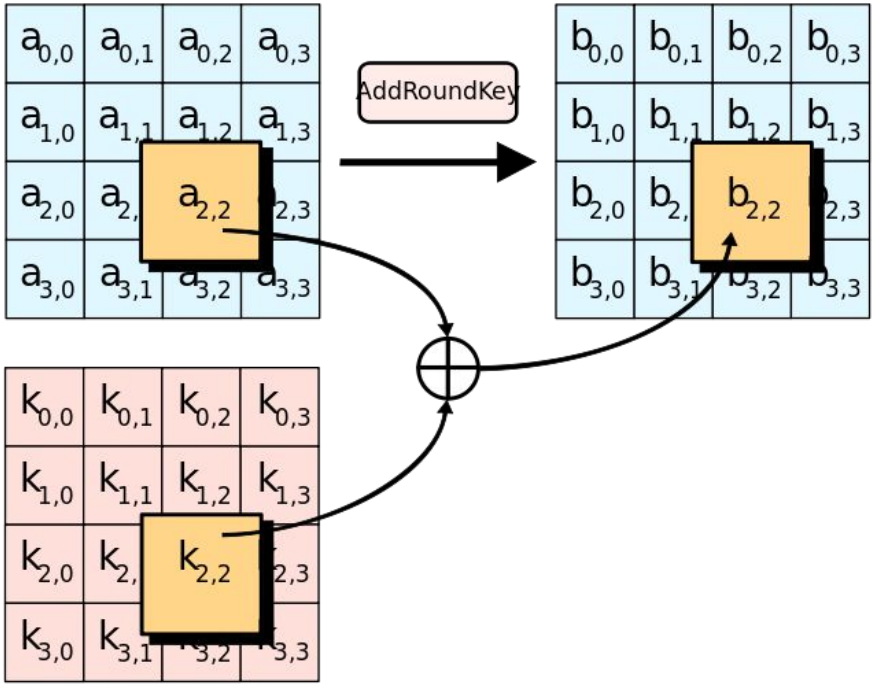
- AES operates on a 4 x 4 column-major array of bytes (“state”)
- Starts with 16 bytes of plaintext

$$\begin{bmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \end{bmatrix}$$

Nomenclature: Each step goes a → b

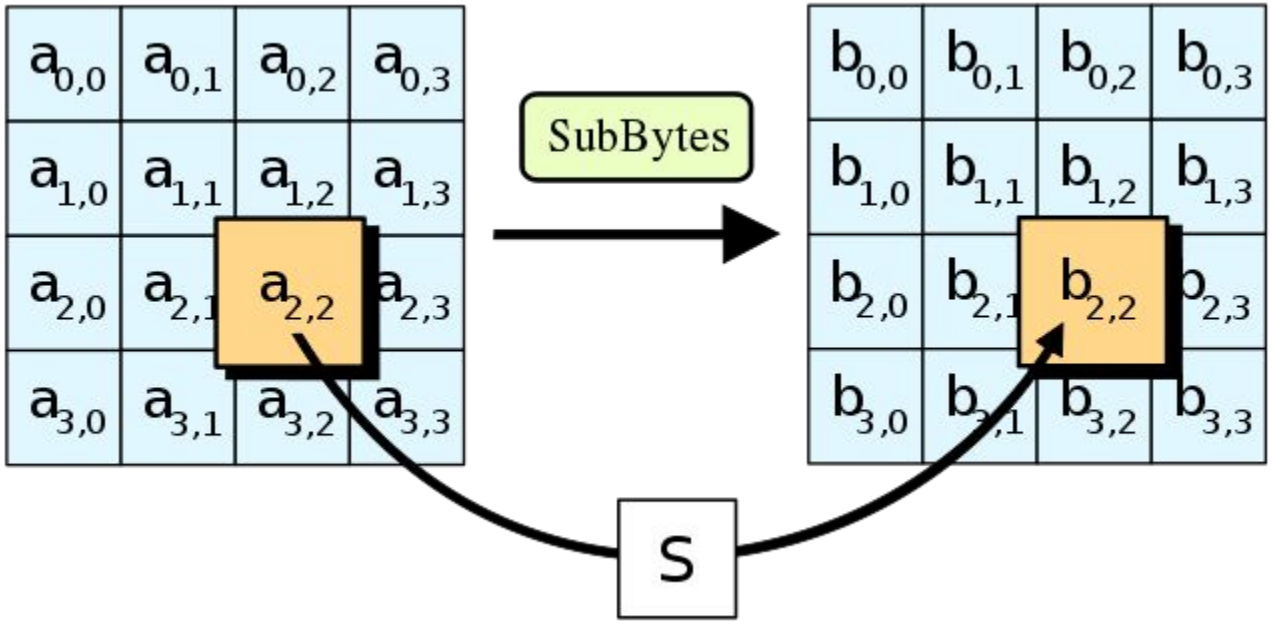
1 a) “AddRoundKey” - Key XOR

- Each byte $a_{i,j}$ of the state is XOR with the round key bytes $k_{i,j}$:



2 a) “SubBytes” - S-Box

- Each byte $a_{i,j}$ is replaced with a SubByte $S(a_{i,j})$ using 8-bit S-Box



2 a) “SubBytes” - The Rijndael S-Box

AES S-box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

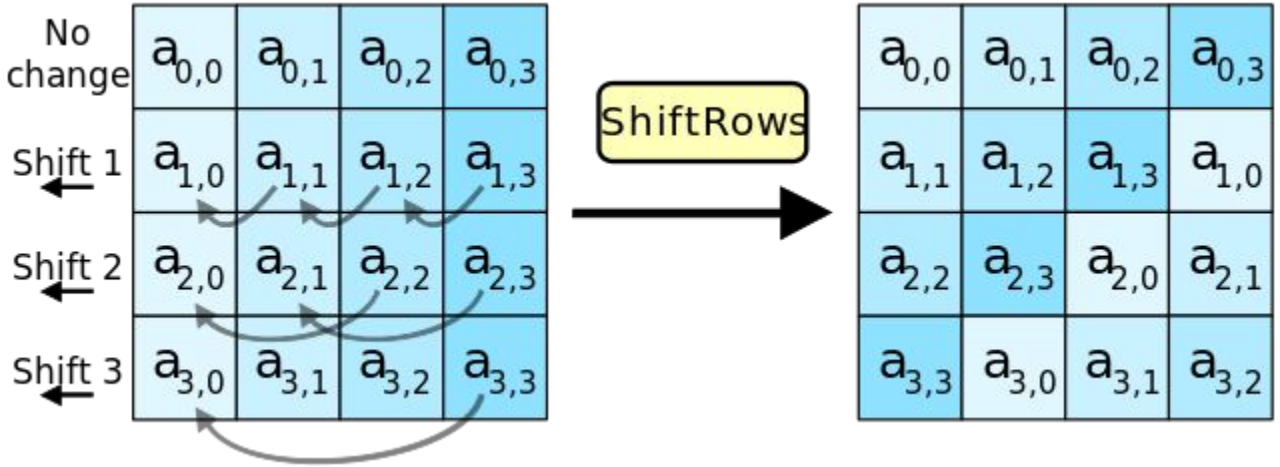
The column is determined by the least significant nibble, and the row by the most significant nibble. For example, the value $9a_{16}$ is converted into $b8_{16}$.

Inverse S-box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
10	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
20	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
30	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
40	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
50	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
60	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
70	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
80	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
90	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a0	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b0	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c0	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d0	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e0	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f0	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

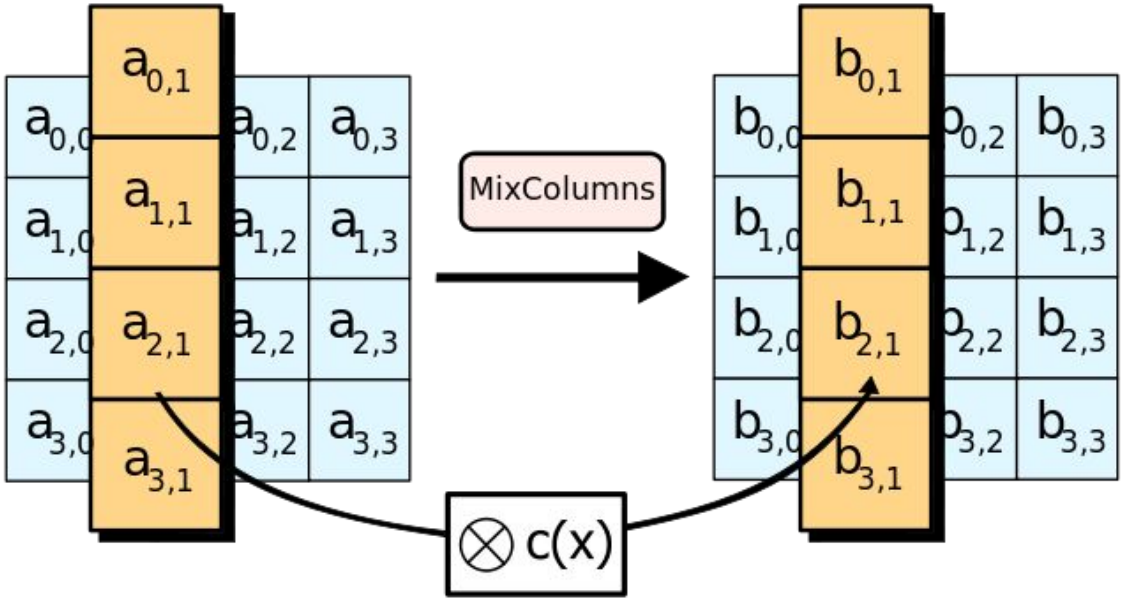
2 b) Shift Rows

- Cyclically shift the bytes in each row by fixed offset



2 c) MixColumns

- Four bytes of each column of “state” are mixed linearly



2 c) MixColumns

- Each column transformed by fixed matrix:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

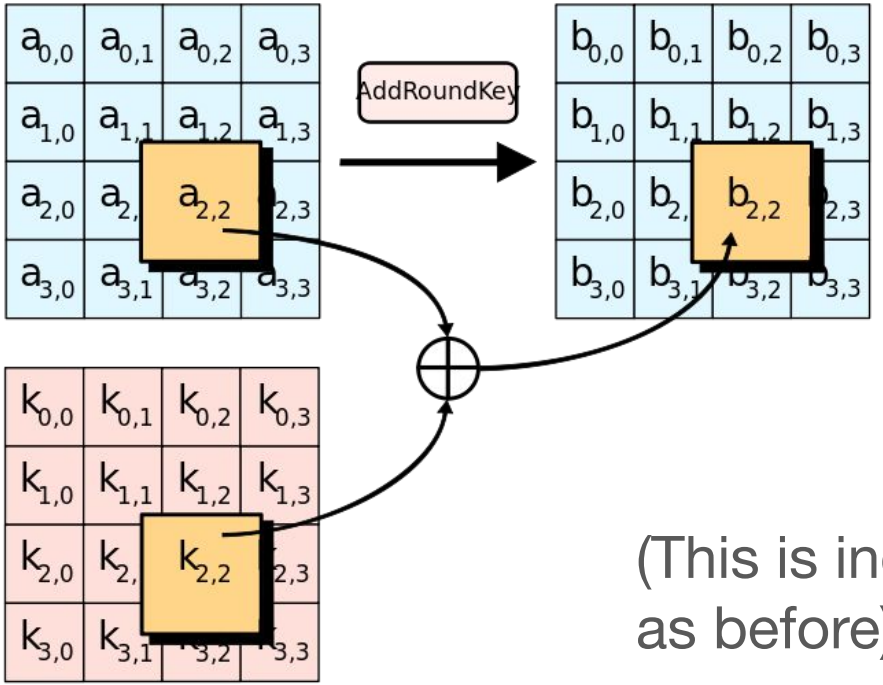
- This matrix multiplication is over a Galois Field GF(2^8)

$$\begin{aligned} d_0 &= 2 \bullet b_0 \oplus 3 \bullet b_1 \oplus 1 \bullet b_2 \oplus 1 \bullet b_3 \\ d_1 &= 1 \bullet b_0 \oplus 2 \bullet b_1 \oplus 3 \bullet b_2 \oplus 1 \bullet b_3 \\ d_2 &= 1 \bullet b_0 \oplus 1 \bullet b_1 \oplus 2 \bullet b_2 \oplus 3 \bullet b_3 \\ d_3 &= 3 \bullet b_0 \oplus 1 \bullet b_1 \oplus 1 \bullet b_2 \oplus 2 \bullet b_3 \end{aligned}$$

We can implement this with add, shift, and XOR

2 d) “AddRoundKey” - Key XOR

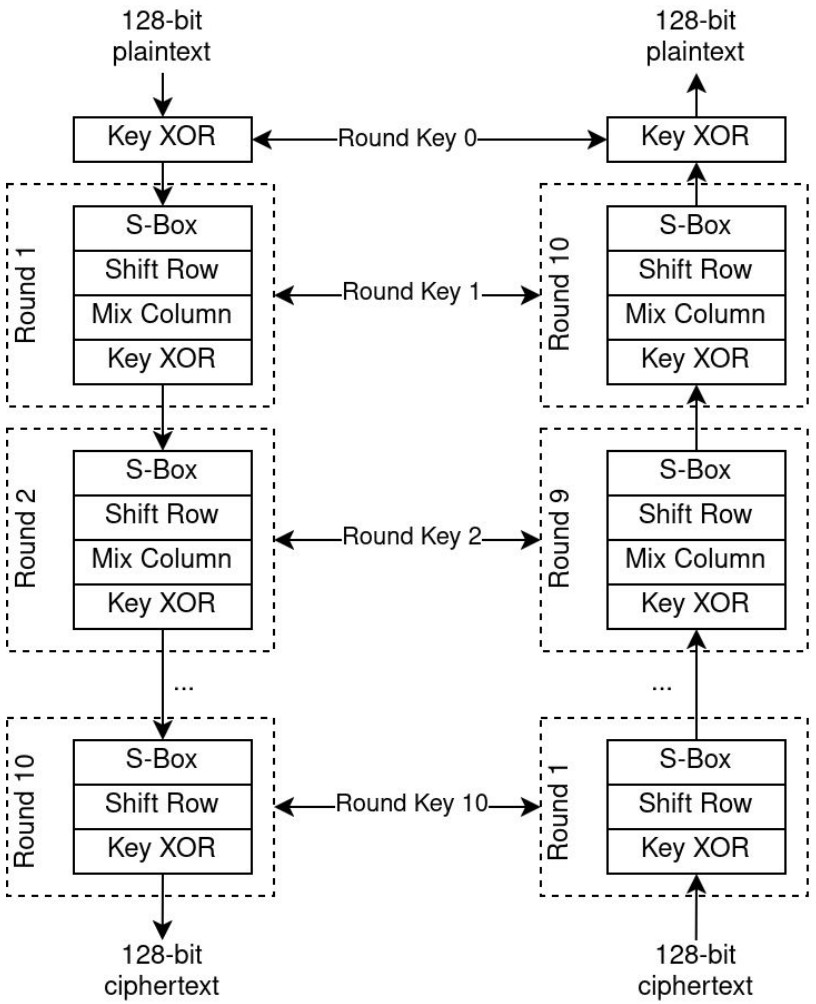
- Each byte $a_{i,j}$ of the state is XOR with the round key bytes $k_{i,j}$:



(This is indeed the same as before)

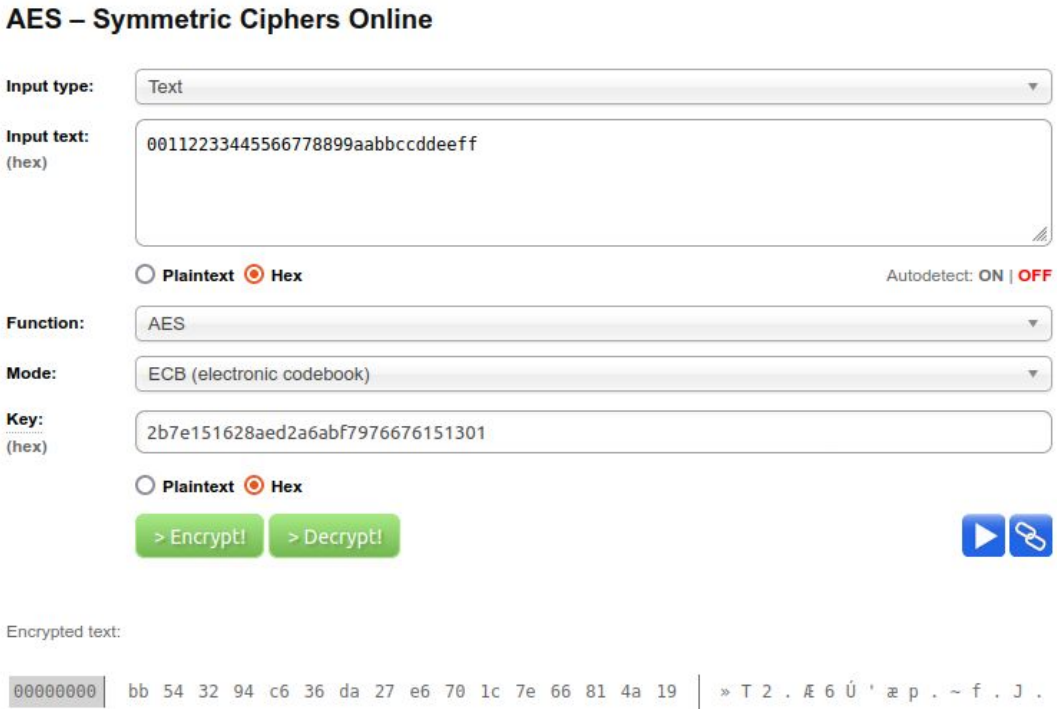
AES Overview (again)

1. Initial round key:
    a. XOR with key
2. 9 rounds:
    a. S-Box (Simple substitution)
    b. Shift Rows (Transposition)
    c. Mix Column (Linear mixing step)
    d. XOR with key
3. Final round
    a. No Mix Column



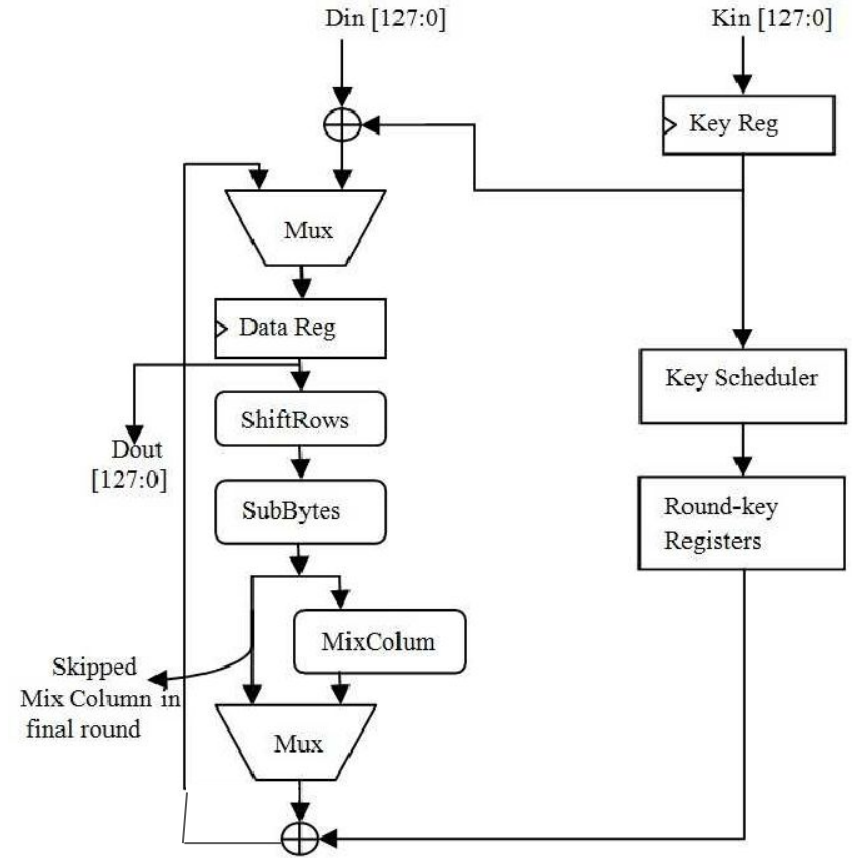
AES example

http://aes.online-domain-tools.com/



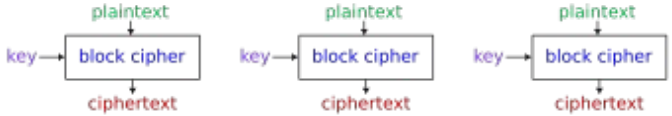
AES encryption in HW - Whiteboarding

AES encryption in HW - Whiteboarding

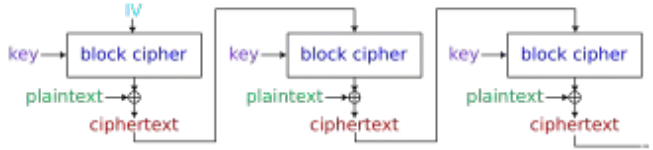


Different Block Cipher Modes

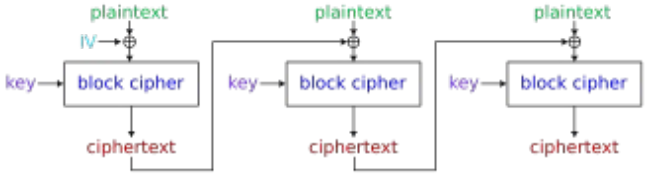
Electronic codebook (ECB)



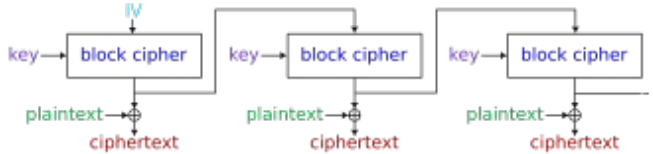
Cipher feedback (CFB)



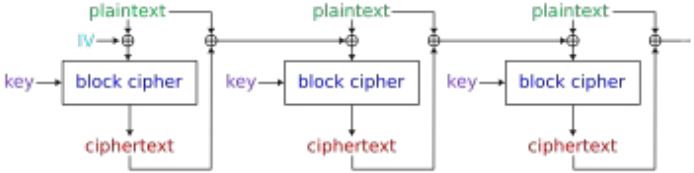
Cipher block chaining (CBC)



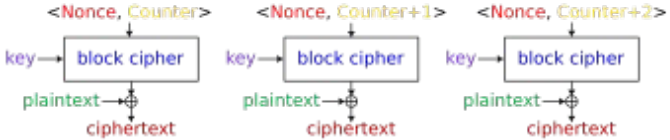
Output feedback (OFB)



Propagating cipher block chaining (PCBC)



Counter (CTR)

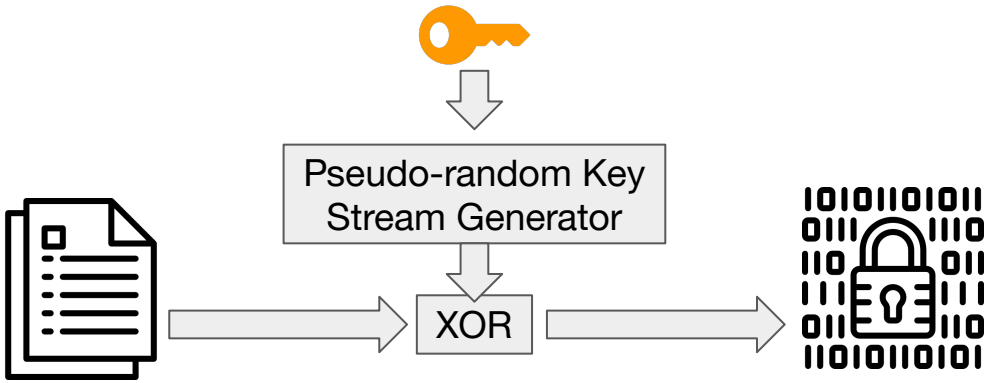


Encrypting larger data: Stream Ciphers

- (Pseudo) random key stream generator
- Encryption a simple XOR operation
- Small number of gates
- Low power consumption
- High speed operation
- No error propagation

Stream ciphers

- Encrypt a message one bit/byte/block at a time
- Do not (necessarily) derive security from complexity of transformation, but from unpredictable way in which transmission depends on position in the plaintext stream
- Applications: RFID, Wireless Telecoms



Assessments: Week 2

Report 2: Hardware Crypto “Whiteboard”

- I explained one possible implementation of AES hardware
  - More are possible!
- `examples/spi_aes` has another, different implementation
- Tell me how it works!
- Record a 5 min video walkthrough of the AES HW (not the SPI)
- Full instruction on Moodle, but:
  - Make the video accurate, complete, and enjoyable to watch!
  - Save to YouTube or your UNSW sharepoint (or to Moodle if <200MB)

Report 2: Due Friday Week 3

That’s it! Questions?