PRESIDENCY UNIVERSITY

GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

OVER 40 YEARS OF ACADEMIC WISDOM

CSE3078 – Cryptography and Network Security

School of Computer Science and Engineering

Module 2

Private Key Cryptography and Number Theory

PRESIDENCY UNIVERSITY
GAIN MORE KNOWLEDGE REACH GREATER HEIGHTS
Private University Estd. in Karnataka State by Act No. 41 of 2013

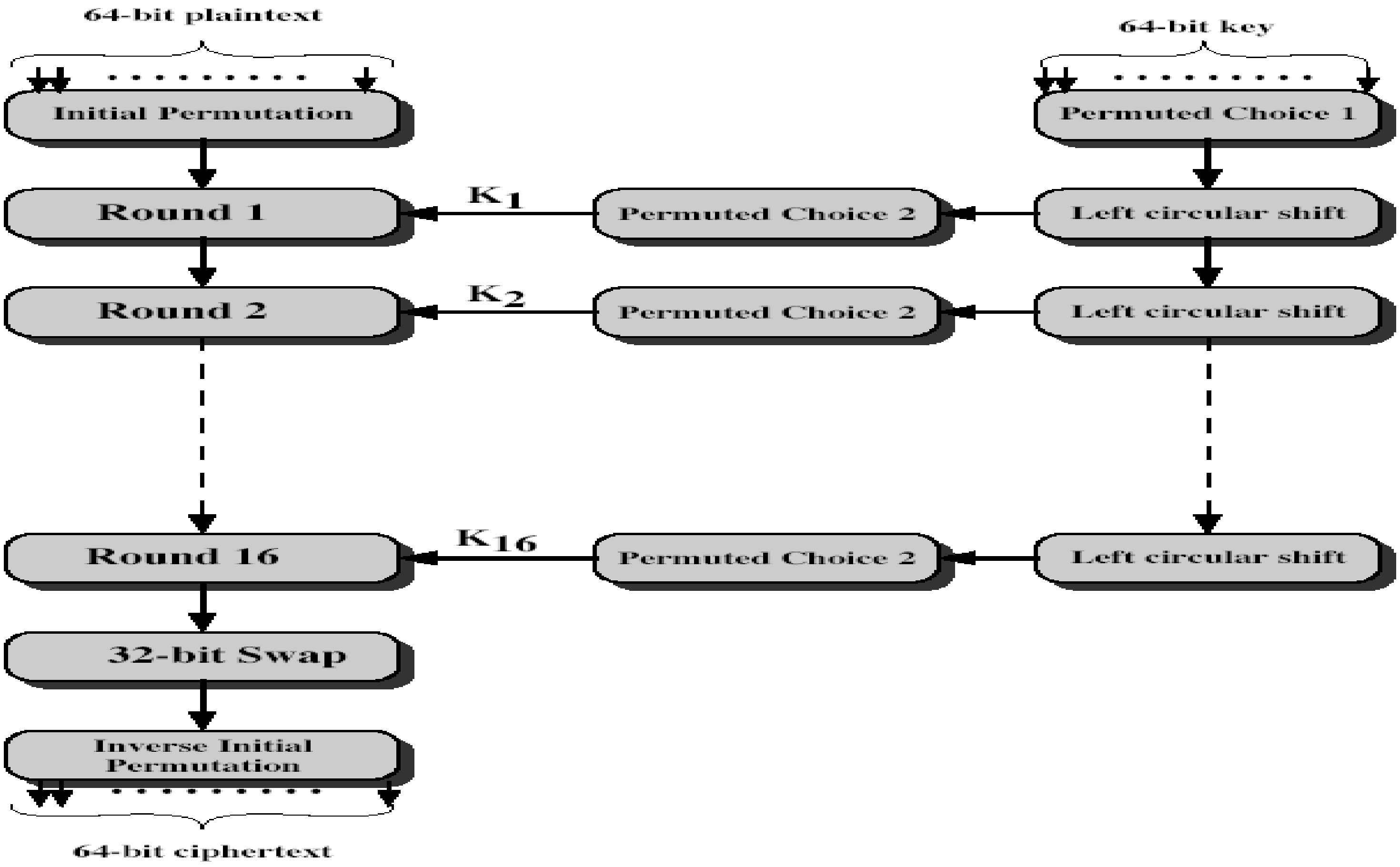
Data Encryption Standard (DES)

- most widely used block cipher in world
- adopted in 1977 by NBS-National Bureau of Standards (now NISTNational Institute of Standards and Technology)
  - as FIPS (FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION) PUB 46
- encrypts 64-bit data using 56-bit key
- has widespread use
- has been considerable controversy over its security

DES History

- IBM developed Lucifer cipher
  - by team led by Feistel
  - used 64-bit data blocks with 128-bit key
- then redeveloped as a commercial cipher with input from NSA and others
- in 1973 NBS issued request for proposals for a national cipher standard
- IBM submitted their revised Lucifer which was eventually accepted as the DES

DES Encryption



Initial Permutation IP

- first step of the data computation
- IP reorders the input data bits
- even bits to LH half, odd bits to RH half
- quite regular in structure (easy in h/w)
- example:

```
IP(675a6967 5e5a6b5a) = (ffb2194d 004df6fb)
```

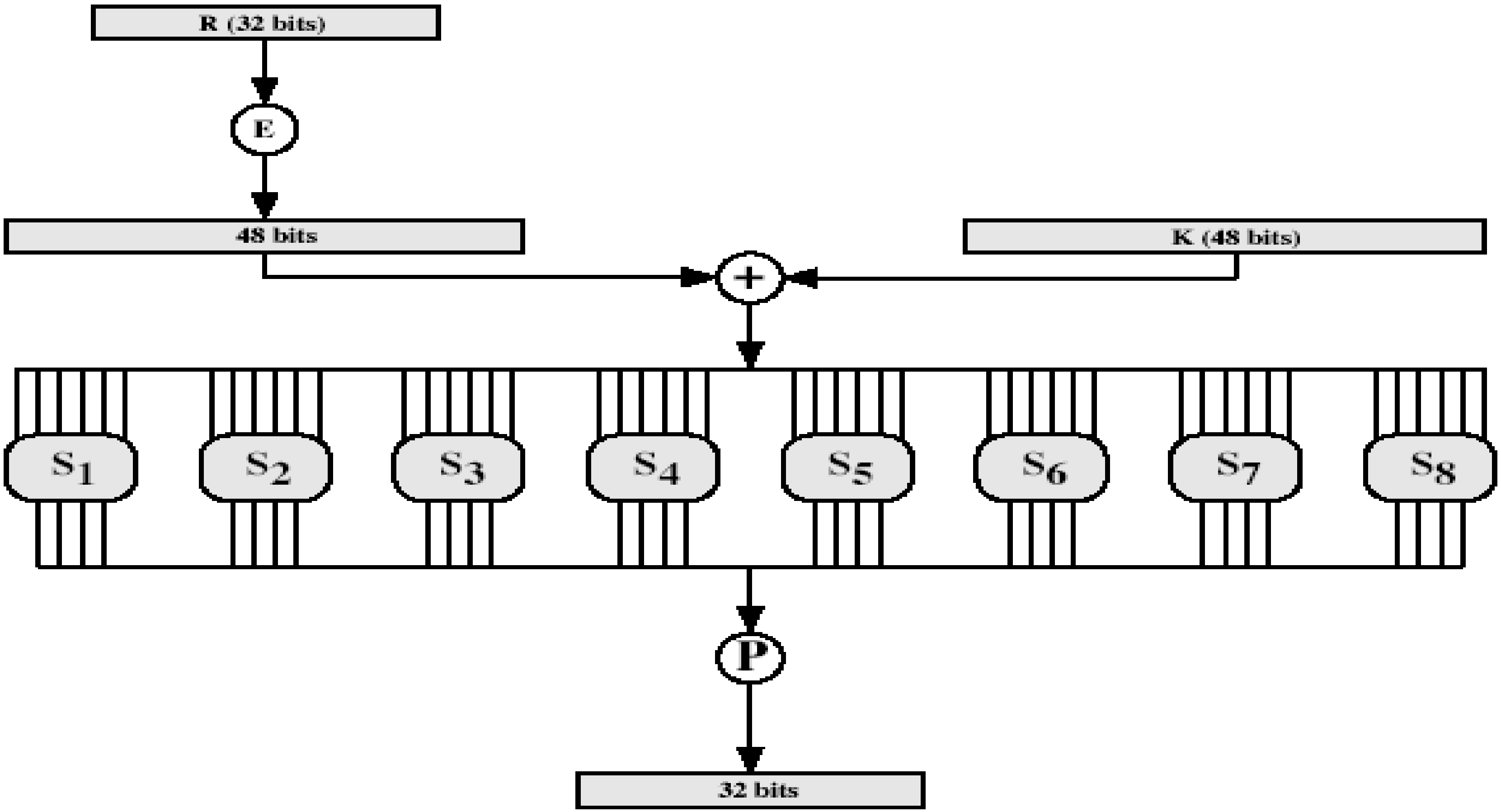
DES Round Structure

- uses two 32-bit L & R halves
- as for any Feistel cipher can describe as:

  $L_i = R_{i-1}$

  $R_i = L_{i-1} \text{ xor } F(R_{i-1}, K_i)$
- takes 32-bit R half and 48-bit subkey and:
  - expands R to 48-bits using perm E
  - adds to subkey
  - passes through 8 S-boxes to get 32-bit result
  - finally permutes this using 32-bit perm P

DES Round Structure



Substitution Boxes S

- have eight S-boxes which map 6 to 4 bits
- each S-box is actually 4 little 4 bit boxes
  - outer bits 1 & 6 (**row** bits) select one rows
  - inner bits 2-5 (**col** bits) are substituted
  - result is 8 lots of 4 bits, or 32 bits
- row selection depends on both data & key
  - feature known as autoclaving (autokeying)
- example:

```
S(18 09 12 3d 11 17 38 39) = 5fd25e03
```

DES Key Schedule

- forms subkeys used in each round
- consists of:
  - initial permutation of the key (PC1) which selects 56-bits in two 28-bit halves
  - 16 stages consisting of:
    - selecting 24-bits from each half
    - permuting them by PC2 for use in function f,
    - rotating **each half** separately either 1 or 2 places depending on the **key rotation schedule** K

DES Decryption

- decrypt must unwind steps of data computation
- with Feistel design, do encryption steps again
- using subkeys in reverse order (SK16 … SK1)
- note that IP undoes final FP step of encryption
- 1st round with SK16 undoes 16th encrypt round
- ….
- 16th round with SK1 undoes 1st encrypt round
- then final FP undoes initial encryption IP
- thus recovering original data value

Strength of DES – Key Size

- 56-bit keys have $2^{56} = 7.2 \times 10^{16}$ values
- brute force search looks hard
- recent advances have shown is possible
  - in 1997 on Internet in a few months
  - in 1998 on dedicated h/w (EFF) in a few days
  - in 1999 above combined in 22hrs!
- still must be able to recognize plaintext
- now considering alternatives to DES

Strength of DES – Timing Attacks

- attacks actual implementation of cipher
- use knowledge of consequences of implementation to derive knowledge of some/all subkey bits
- specifically use fact that calculations can take varying times depending on the value of the inputs to it
- particularly problematic on smartcards

Introduction to Finite Fields – Number Theory

- Will now introduce finite fields
- Of increasing importance in cryptography
  - AES, Elliptic Curve, IDEA, Public Key
- Concern operations on “numbers”
  - where what constitutes a “number” and the type of operations varies considerably
- Start with concepts of groups, rings, fields from abstract algebra

Group

- a set of elements or “numbers”
- with some operation whose result is also in the set (closure)
- obeys:
  - associative law: (a.b).c = a.(b.c)
  - has identity e: e.a = a.e = a
  - has inverses a^{-1}: $a.a^{-1} = e$
- if commutative a.b = b.a
  - then forms an **abelian group**

Cyclic Group

- define **exponentiation** as repeated application of operator
  - example: $a^{-3} = a.a.a$
- and let identity be: $e=a^0$
- a group is cyclic if every element is a power of some fixed element
  - ie $b = a^k$ for some a and every b in group
- a is said to be a generator of the group

PRESIDENCY UNIVERSITY
Private University Estd. in Karnataka State by Act No. 41 of 2013

Ring

- a set of “numbers”
- with two operations (addition and multiplication) which form:
- an abelian group with addition operation
- and multiplication:
  - has closure
  - is associative
  - distributive over addition: `a(b+c) = ab + ac`
- if multiplication operation is commutative, it forms a **commutative ring**
- if multiplication operation has an identity and no zero divisors, it forms an **integral domain**

Field

- a set of numbers
- with two operations which form:
  - abelian group for addition
  - abelian group for multiplication (ignoring 0)
  - ring
- have hierarchy with more axioms/laws
  - group -> ring -> field

Modular Arithmetic

- define **modulo operator** "a mod n" to be remainder when a is divided by n
- use the term **congruence** for: a = b mod n
  - when divided by *n*, a & b have same remainder
  - eg. 100 = 34 mod 11
- b is called a **residue** of a mod n
  - since with integers can always write: a = qn + b
  - usually chose smallest positive remainder as residue
    - ie. 0 <= b <= n-1
  - process is known as **modulo reduction**
    - eg. -12 mod 7 = -5 mod 7 = 2 mod 7 = 9 mod 7

PRESIDENCY UNIVERSITY
Private University Estd. in Karnataka State by Act No. 41 of 2013

Modular Arithmetic Operations

- is 'clock arithmetic'
- uses a finite number of values, and loops back from either end
- modular arithmetic is when do addition & multiplication and modulo reduce answer
- can do reduction at any point, ie
  - a+b mod n = [a mod n + b mod n] mod n

Modular Arithmetic

- can do modular arithmetic with any group of integers: $Z_n = \{0, 1, \ldots, n-1\}$
- form a commutative ring for addition
- with a multiplicative identity
- note some peculiarities
  - if $(a+b)=(a+c) \bmod n$
    then $b=c \bmod n$
  - but if $(a.b)=(a.c) \bmod n$
    then $b=c \bmod n$ only if a is relatively prime to n

Modulo 8 Addition Example

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

Galois Fields

- Named in honor of Évariste Galois- is a field that contains a finite number of elements
- finite fields play a key role in cryptography
- can show number of elements in a finite field **must** be a power of a prime p^n
- known as Galois fields
- denoted $GF(p^n)$
- in particular often use the fields:
  - $GF(p)$
  - $GF(2^n)$

Galois Fields GF(p)

- GF(p) is the set of integers {0,1, … , p-1} with arithmetic operations modulo prime p
- these form a finite field
  - since have multiplicative inverses
- hence arithmetic is "well-behaved" and can do addition, subtraction, multiplication, and division without leaving the field GF(p)

GF(7) Multiplication Example

×	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

Origins

- clear a replacement for DES was needed
  - have theoretical attacks that can break it
  - have demonstrated exhaustive key search attacks
- can use Triple-DES – but slow, has small blocks
- US NIST issued call for ciphers in 1997
- 15 candidates accepted in Jun 98
- 5 were shortlisted in Aug-99
- Rijndael was selected as the AES in Oct-2000
- issued as FIPS PUB 197 standard in Nov-2001

AES Requirements

- private key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- stronger & faster than Triple-DES
- active life of 20-30 years (+ archival use)
- provide full specification & design details
- both C & Java implementations
- NIST have released all submissions & unclassified analyses

AES Evaluation Criteria

- initial criteria:
  - security – effort for practical cryptanalysis
  - cost – in terms of computational efficiency
  - algorithm & implementation characteristics
- final criteria
  - general security
  - ease of software & hardware implementation
  - implementation attacks
  - flexibility (in en/decrypt, keying, other factors)

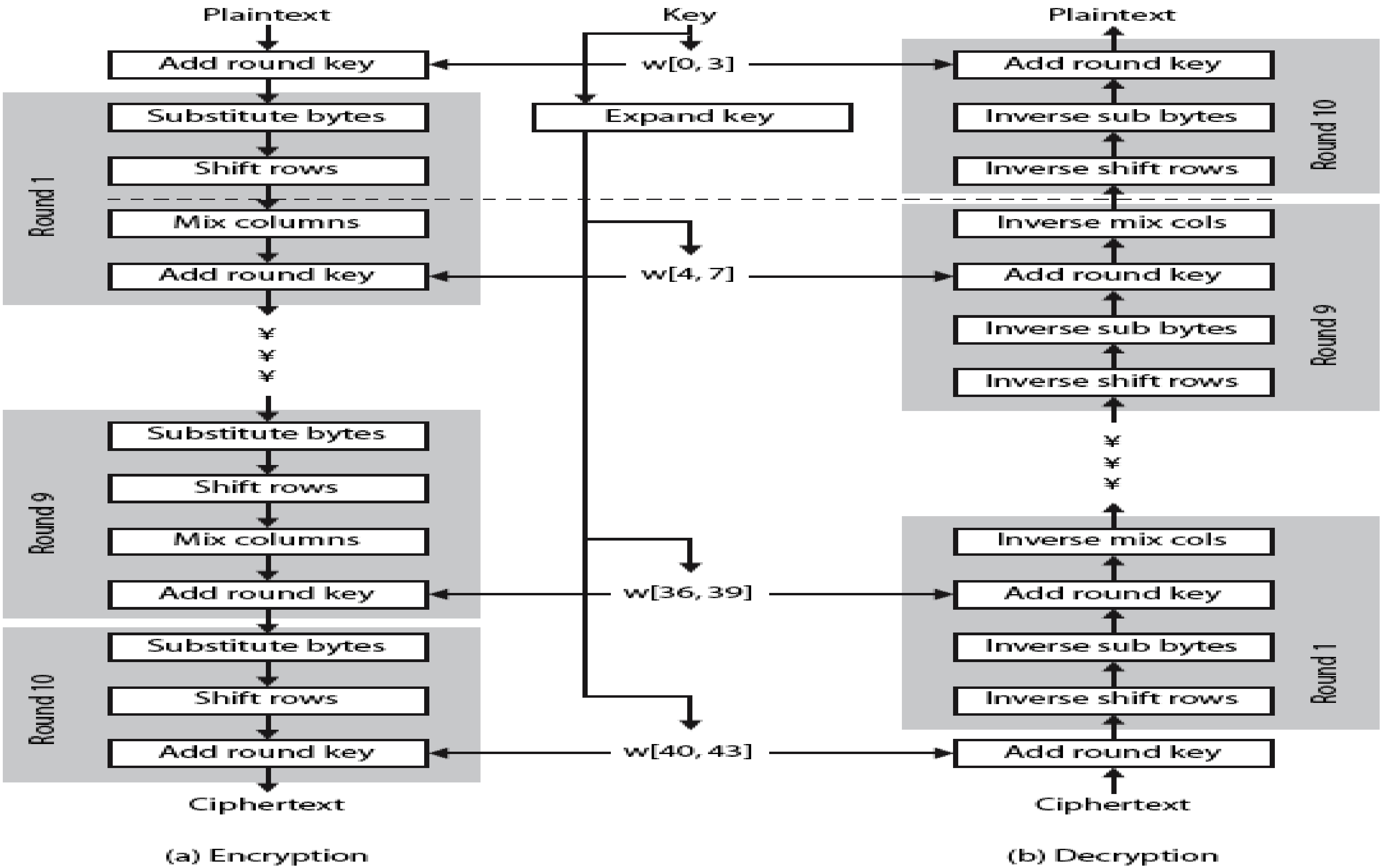
The AES Cipher - Rijndael

- designed by Rijmen-Daemen in Belgium
- has 128/192/256 bit keys, 128 bit data
- an **iterative** rather than **feistel** cipher
  - processes data as block of 4 columns of 4 bytes
  - operates on entire data block in every round
- designed to be:
  - resistant against known attacks
  - speed and code compactness on many CPUs
  - design simplicity

Rijndael

- data block of 4 columns of 4 bytes is state
- key is expanded to array of words
- has 9/11/13 rounds in which state undergoes:
  - byte substitution (1 S-box used on every byte)
  - shift rows (permute bytes between groups/columns)
  - mix columns (subs using matrix multipy of groups)
  - add round key (XOR state with key material)
  - view as alternating XOR key & scramble data bytes
- initial XOR key material & incomplete last round
- with fast XOR & table lookup implementation

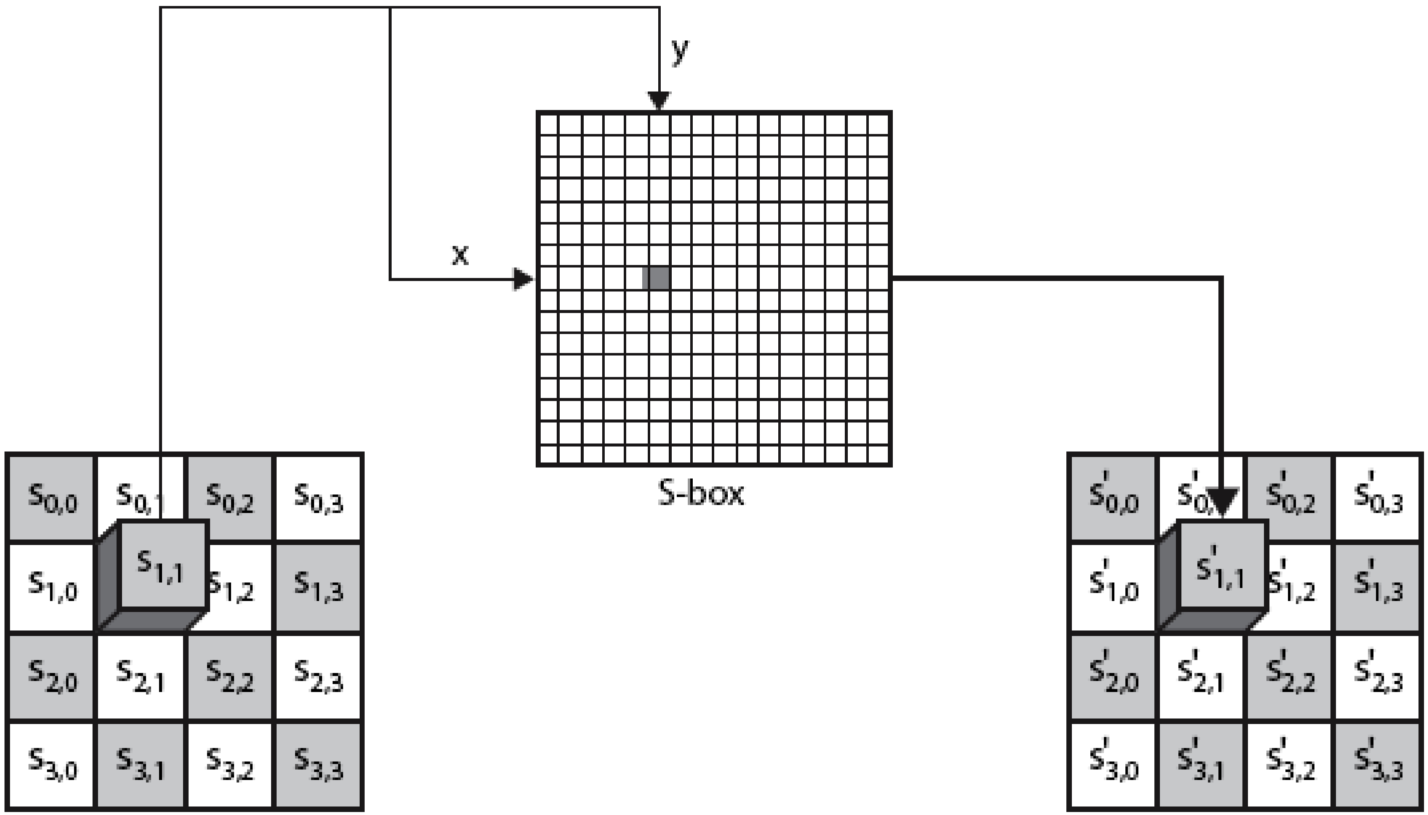
Rijndael


Plaintext
Key
Plaintext
Add round key
w[0, 3]
Add round key
Substitute bytes
Expand key
Inverse sub bytes
Shift rows
Inverse shift rows
Round 10
Round 1
Mix columns
Inverse mix cols
Add round key
w[4, 7]
Add round key
Inverse sub bytes
Inverse shift rows
Round 9
¥
¥
¥
Substitute bytes
Shift rows
Round 9
Mix columns
Inverse mix cols
Add round key
w[36, 39]
Add round key
Substitute bytes
Inverse sub bytes
Shift rows
Inverse shift rows
Round 1
Round 10
Add round key
w[40, 43]
Add round key
Ciphertext
Ciphertext
(a) Encryption
(b) Decryption

Byte Substitution

- a simple substitution of each byte
- uses one table of 16x16 bytes containing a permutation of all 256 8-bit values
- each byte of state is replaced by byte indexed by row (left 4-bits) & column (right 4-bits)
  - eg. byte {95} is replaced by byte in row 9 column 5
  - which has value {2A}
- S-box constructed using defined transformation of values in GF(2^8)
- designed to be resistant to all known attacks

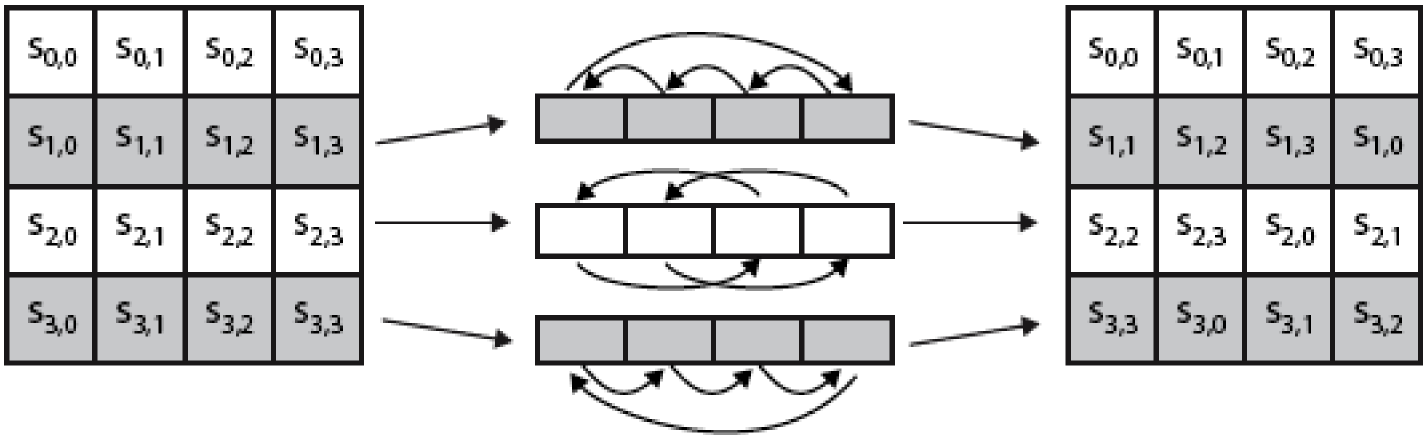
Byte Substitution



Shift Rows

- a circular byte shift in each each
  - 1^{st} row is unchanged
  - 2^{nd} row does 1 byte circular shift to left
  - 3rd row does 2 byte circular shift to left
  - 4th row does 3 byte circular shift to left
- decrypt inverts using shifts to right
- since state is processed by columns, this step permutes bytes between the columns

Shift Rows



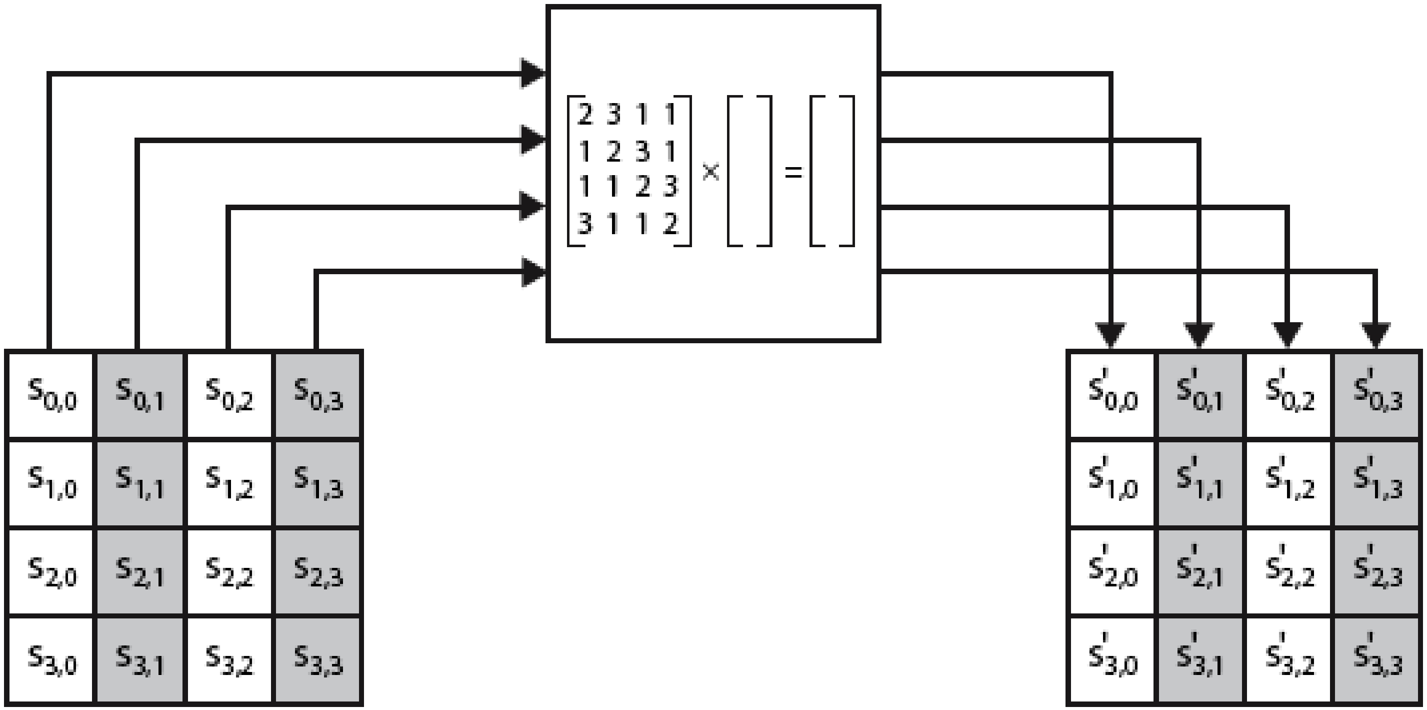
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $s_{1,0}$
$s_{2,2}$ $s_{2,3}$ $s_{2,0}$ $s_{2,1}$
$s_{3,3}$ $s_{3,0}$ $s_{3,1}$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

Mix Columns

- each column is processed separately
- each byte is replaced by a value dependent on all 4 bytes in the column
- effectively a matrix multiplication in $GF(2^8)$ using prime poly $m(x) = x^8+x^4+x^3+x+1$

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

Mix Columns



2 3 1 1
1 2 3 1
1 1 2 3
3 1 1 2
×
=
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

Mix Columns

- can express each col as 4 equations
  - to derive each new byte in col
- decryption requires use of inverse matrix
  - with larger coefficients, hence a little harder
- have an alternate characterisation
  - each column a 4-term polynomial
  - with coefficients in GF(2^8)
  - and polynomials multiplied modulo (x^4+1)

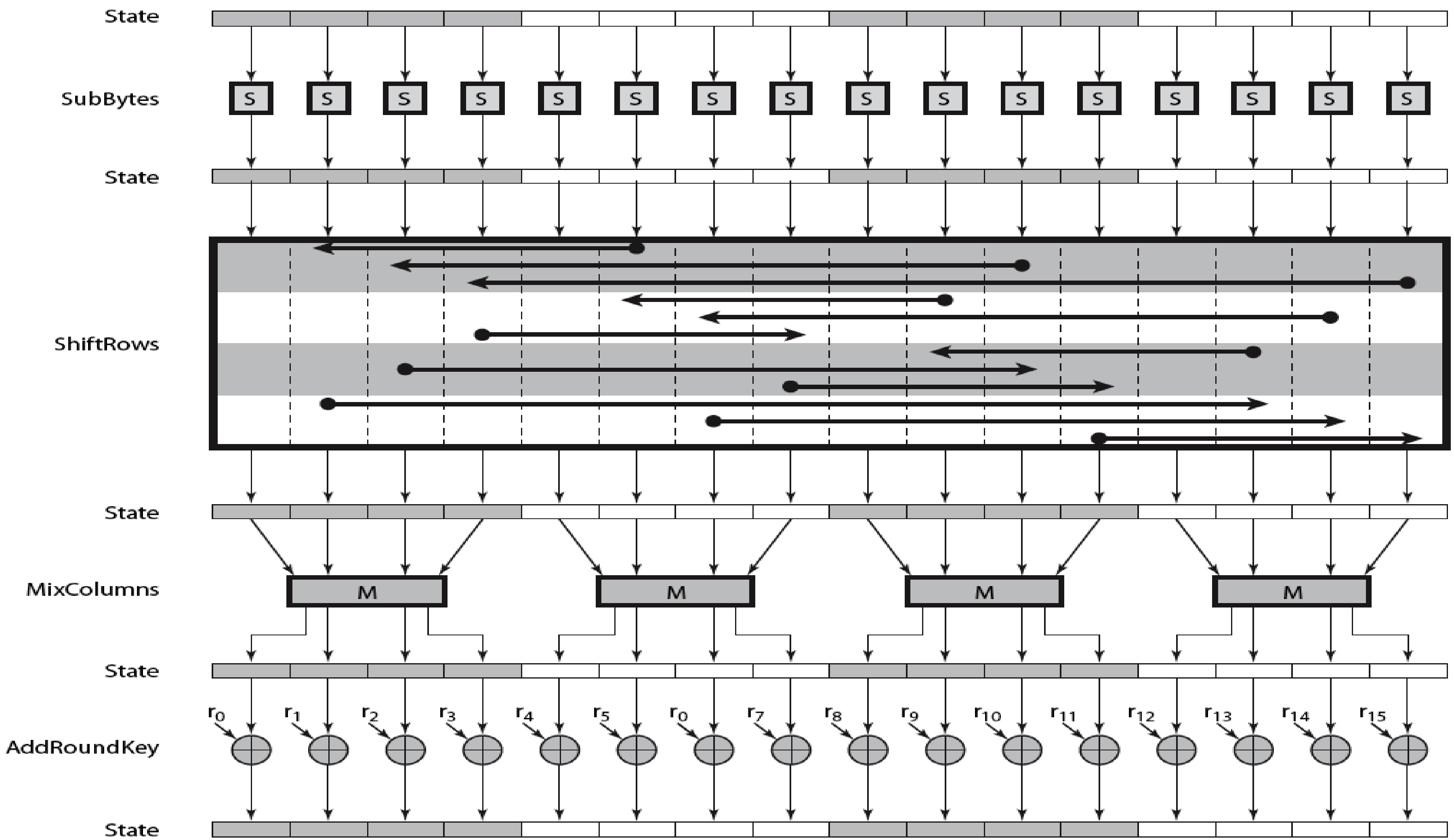
Add Round Key

- XOR state with 128-bits of the round key
- again processed by column (though effectively a series of byte operations)
- inverse for decryption identical
  - since XOR own inverse, with reversed keys
- designed to be as simple as possible
  - a form of Vernam cipher on expanded key
  - requires other stages for complexity / security

Add Round Key

$$\begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} \oplus \begin{bmatrix} w_i & w_{i+1} & w_{i+2} & w_{i+3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

AES Round



State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r0 r1 r2 r3 r4 r5 r0 r7 r8 r9 r10 r11 r12 r13 r14 r15
State

EXAMPLE

Plaintext – Two One Nine Two

T	w	o		O	n	e		N	i	n	e		T	w	o
54	77	6F	20	4F	6E	65	20	43	69	6E	25	20	54	77	6F

Plaintext in Hex Format

54 77 6F 20 4F 6E 65 20 43 69 6E 25 20 54 77 6F

Encryption Key – Thats my Kung Fu

T	h	a	t	s		m	y		K	u	n	g		F	u
54	68	61	74	73	20	6D	79	20	4B	75	6E	67	20	46	75

Encryption Key in Hex Format

54 68 61 74 73 20 6D 79 20 4B 75 6E 67 20 46 75

Key Generation For First Round

- Key in Hex (128 bits): 54 68 61 74 73 20 6D 79 20 4B 75 6E 67 20 46 75
- $w[0] = (54, 68, 61, 74), w[1] = (73, 20, 6D, 79), w[2] = (20, 4B, 75, 6E), w[3] = (67, 20, 46, 75)$
- $g(w[3])$:
  - circular byte left shift of $w[3]$: $(20, 46, 75, 67)$
  - Byte Substitution (S-Box): $(B7, 5A, 9D, 85)$
  - Adding round constant $(01, 00, 00, 00)$ gives: $g(w[3]) = (B6, 5A, 9D, 85)$
- $w[4] = w[0] \oplus g(w[3]) = (E2, 32, FC, F1)$:

0101 0100	0110 1000	0110 0001	0111 0100
1011 0110	0101 1010	1001 1101	1000 0101
1110 0010	0011 0010	1111 1100	1111 0001
E2	32	FC	F1

- $w[5] = w[4] \oplus w[1] = (91, 12, 91, 88)$, $w[6] = w[5] \oplus w[2] = (B1, 59, E4, E6)$,
  $w[7] = w[6] \oplus w[3] = (D6, 79, A2, 93)$
- first roundkey: E2 32 FC F1 91 12 91 88 B1 59 E4 E6 D6 79 A2 93

Key Generation For All Rounds

- Round 0: 54 68 61 74 73 20 6D 79 20 4B 75 6E 67 20 46 75
- Round 1: E2 32 FC F1 91 12 91 88 B1 59 E4 E6 D6 79 A2 93
- Round 2: 56 08 20 07 C7 1A B1 8F 76 43 55 69 A0 3A F7 FA
- Round 3: D2 60 0D E7 15 7A BC 68 63 39 E9 01 C3 03 1E FB
- Round 4: A1 12 02 C9 B4 68 BE A1 D7 51 57 A0 14 52 49 5B
- Round 5: B1 29 3B 33 05 41 85 92 D2 10 D2 32 C6 42 9B 69
- Round 6: BD 3D C2 B7 B8 7C 47 15 6A 6C 95 27 AC 2E 0E 4E
- Round 7: CC 96 ED 16 74 EA AA 03 1E 86 3F 24 B2 A8 31 6A
- Round 8: 8E 51 EF 21 FA BB 45 22 E4 3D 7A 06 56 95 4B 6C
- Round 9: BF E2 BF 90 45 59 FA B2 A1 64 80 B4 F7 F1 CB D8
- Round 10: 28 FD DE F8 6D A4 24 4A CC C0 A4 FE 3B 31 6F 26

Add Round Key

- State Matrix and Roundkey No.0 Matrix:

$$\begin{pmatrix} 54 & 4F & 4E & 20 \\ 77 & 6E & 69 & 54 \\ 6F & 65 & 6E & 77 \\ 20 & 20 & 65 & 6F \end{pmatrix} \qquad \begin{pmatrix} 54 & 73 & 20 & 67 \\ 68 & 20 & 4B & 20 \\ 61 & 6D & 75 & 46 \\ 74 & 79 & 6E & 75 \end{pmatrix}$$

- XOR the corresponding entries, e.g., $69 \oplus 4B = 22$

$$\begin{array}{r} 0110\ 1001 \\ 0100\ 1011 \\ \hline 0010\ 0010 \end{array}$$

- the new State Matrix is

$$\begin{pmatrix} 00 & 3C & 6E & 47 \\ 1F & 4E & 22 & 74 \\ 0E & 08 & 1B & 31 \\ 54 & 59 & 0B & 1A \end{pmatrix}$$

Add Round Key

54	4F	4E	20
77	6E	69	54
6F	65	6E	77
20	20	65	6F

Plaintext

$\oplus$

XOR

54	73	20	67
68	20	4B	20
61	6D	75	46
74	79	6E	75

Round 0 Key

00	3C	63	47
1F	4E	22	74
0E	08	1B	31
54	59	0B	1A

New State Array

Substitute Bytes

- current State Matrix is

$$\begin{pmatrix} 00 & 3C & 6E & 47 \\ 1F & 4E & 22 & 74 \\ 0E & 08 & 1B & 31 \\ 54 & 59 & 0B & 1A \end{pmatrix}$$

- substitute each entry (byte) of current state matrix by corresponding entry in AES S-Box
- for instance: byte 6E is substituted by entry of S-Box in row 6 and column E, i.e., by 9F
- this leads to new State Matrix

$$\begin{pmatrix} 63 & EB & 9F & A0 \\ C0 & 2F & 93 & 92 \\ AB & 30 & AF & C7 \\ 20 & CB & 2B & A2 \end{pmatrix}$$

- this non-linear layer is for resistance to differential and linear cryptanalysis attacks

Substitute Bytes

63	EB	9F	A0
C0	2F	93	92
AB	30	AF	C7
20	CB	2B	A2

New State Array

Shift Rows

63	EB	9F	A0
C0	2F	93	92
AB	30	AF	C7
20	CB	2B	A2

Old State Array

→

63	EB	9F	A0
2F	93	92	C0
AF	C7	AB	30
A2	20	CB	2B

New State Array

Mix Columns

- Mix Column multiplies fixed matrix against current State Matrix:

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} 63 & EB & 9F & A0 \\ 2F & 93 & 92 & C0 \\ AF & C7 & AB & 30 \\ A2 & 20 & CB & 2B \end{pmatrix} = \begin{pmatrix} BA & 84 & E8 & 1B \\ 75 & A4 & 8D & 40 \\ F4 & 8D & 06 & 7D \\ 7A & 32 & 0E & 5D \end{pmatrix}$$

- entry BA is result of $(02 \bullet 63) \oplus (03 \bullet 2F) \oplus (01 \bullet AF) \oplus (01 \bullet A2)$:
  - $02 \bullet 63 = 00000010 \bullet 01100011 = 11000110$
  - $03 \bullet 2F = (02 \bullet 2F) \oplus 2F = (00000010 \bullet 00101111) \oplus 00101111 = 01110001$
  - $01 \bullet AF = AF = 10101111$ and $01 \bullet A2 = A2 = 10100010$
  - hence

$$\begin{array}{c} 11000110 \\ 01110001 \\ 10101111 \\ 10100010 \\ \hline 10111010 \end{array}$$

Mix Columns

$$
\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}
\times
\begin{bmatrix} 63 & EB & 9F & A0 \\ 2F & 93 & 92 & C0 \\ AF & C7 & AB & 30 \\ A2 & 20 & CB & 2B \end{bmatrix}
\rightarrow
\begin{bmatrix} BA & 84 & E8 & 1B \\ 75 & A4 & 8D & 40 \\ F4 & 8D & 06 & 7D \\ 7A & 32 & 0E & 5D \end{bmatrix}
$$

Constant Matrix   Old State Array   New State Array

Add Round Key

BA	84	E8	1B
75	A4	8D	40
F4	8D	06	7D
7A	32	0E	5D

Old State Array

⊕

XOR

E2	91	B1	D6
32	12	59	79
FC	91	E4	A2
F1	88	E6	93

Round 1 Key

58	15	59	CD
47	B6	D4	39
08	1C	E2	DF
8B	BA	E8	CE

New State Array

Final State

Final State Array after Round 10

29	57	40	1A
C3	14	22	02
50	20	99	D7
5F	F6	B3	3A

AES Final Output

29 C3 50 5F 57 14 20 F6 40 22 99 B3 1A 02 D7 3A

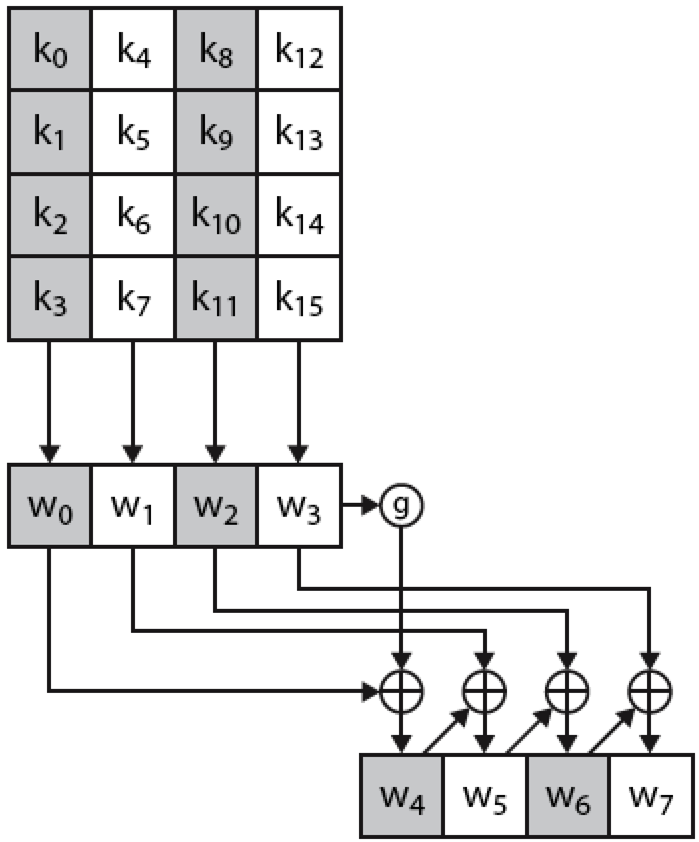
↓

Ciphertext

AES Key Expansion

- takes 128-bit (16-byte) key and expands into array of 44/52/60 32-bit words
- start by copying key into first 4 words
- then loop creating words that depend on values in previous & 4 places back
  - in 3 of 4 cases just XOR these together
  - 1st word in 4 has rotate + S-box + XOR round constant on previous, before XOR 4th back

AES Key Expansion



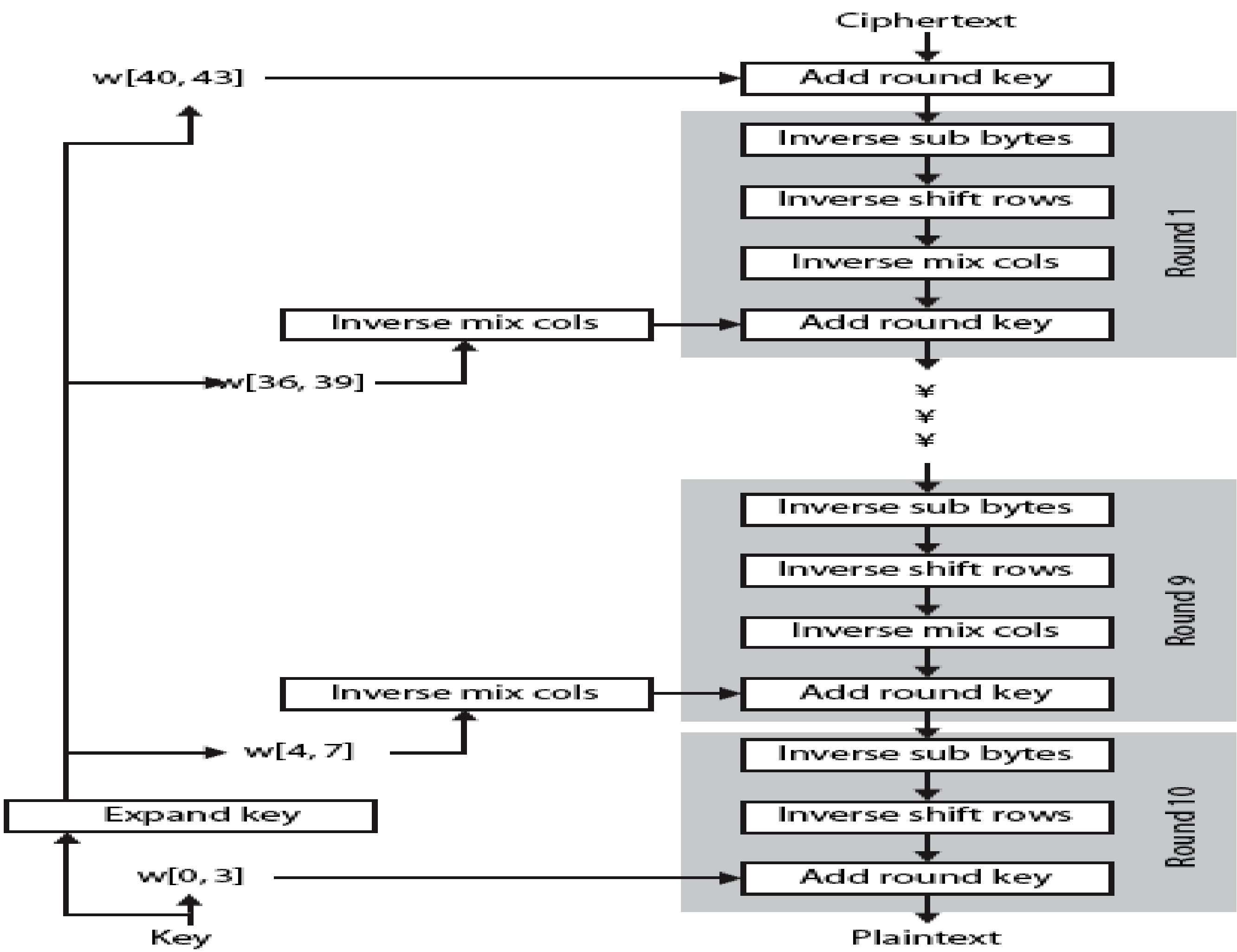
Key Expansion Rationale

- designed to resist known attacks
- design criteria included
  - knowing part key insufficient to find many more
  - invertible transformation
  - fast on wide range of CPU’s
  - use round constants to break symmetry
  - diffuse key bits into round keys
  - enough non-linearity to hinder analysis
  - simplicity of description

AES Decryption

- AES decryption is not identical to encryption since steps done in reverse
- but can define an equivalent inverse cipher with steps as for encryption
  - but using inverses of each step
  - with a different key schedule
- works since result is unchanged when
  - swap byte substitution & shift rows
  - swap mix columns & add (tweaked) round key

AES Decryption



Implementation Aspects

- can efficiently implement on 8-bit CPU
  - byte substitution works on bytes using a table of 256 entries
  - shift rows is simple byte shift
  - add round key works on byte XOR's
  - mix columns requires matrix multiply in $GF(2^8)$ which works on byte values, can be simplified to use table lookups & byte XOR's

Implementation Aspects

- can efficiently implement on 32-bit CPU
  - redefine steps to use 32-bit words
  - can precompute 4 tables of 256-words
  - then each column in each round can be computed using 4 table lookups + 4 XORs
  - at a cost of 4Kb to store tables
- designers believe this very efficient implementation was a key factor in its selection as the AES cipher

Greatest Common Divisor (GCD)

- a common problem in number theory
- GCD (a,b) of a and b is the largest number that divides evenly into both a and b
  - eg GCD(60,24) = 12
- often want **no common factors** (except 1) and hence numbers are **relatively prime**
  - eg GCD(8,15) = 1
  - hence 8 & 15 are relatively prime

Euclid's GCD Algorithm

- an efficient way to find the GCD(a,b)
- uses theorem that:
  - `GCD(a,b) = GCD(b, a mod b)`
  - With any integers a,b and a>=b>=0
  - GCD (55,22)=GCD(22,55 mod 22)=gcd(22,11)=11.

  Where a=kb+r = r(mod b) or a mod b = r.
- Euclid's Algorithm to compute GCD(a,b):

```
– Recursive fn. Euclid(a,b)
– If b=0 then
    return a;
Else returnEuclid (b,a mod b)
```

Extended Euclid's GCD Algorithm

- ✓ Imp. Finitefield in Encryption algorithm such as RSA.
- ✓ For Given integers a,b

The Extended Euclidean algorithm not only calculate the greatest common divisor d but also two additional integers x and y.

i.e., $ax+by=d=gcd(a,b)$

So, E.E A ends with remainder are 0 but the GCD of a,b is $d= gcd\ (a,b)= r_n$

Therefore $x=x_n$; $y = y_n$.

Finding Inverses

```
EXTENDED EUCLID(m, b)
1. (A1, A2, A3)=(1, 0, m);
   (B1, B2, B3)=(0, 1, b)
2. if B3 = 0
   return A3 = gcd(m, b); no inverse
3. if B3 = 1
   return B3 = gcd(m, b); B2 = b^-1 mod m
4. Q = A3 div B3
5. (T1, T2, T3)=(A1 – Q B1, A2 – Q B2, A3 – Q B3)
6. (A1, A2, A3)=(B1, B2, B3)
7. (B1, B2, B3)=(T1, T2, T3)
8. goto 2
```

Inverse of 550 in GF(1759)

Q	A1	A2	A3	B1	B2	B3
—	1	0	1759	0	1	550
3	0	1	550	1	–3	109
5	1	–3	109	–5	16	5
21	–5	16	5	106	–339	4
1	106	–339	4	–111	355	1

Prime Numbers

- Prime numbers only have divisors of 1 and self
  - they cannot be written as a product of other numbers
  - note: 1 is prime, but is generally not of interest
- eg. 2,3,5,7 are prime, 4,6,8,9,10 are not
- prime numbers are central to number theory
- list of prime number less than 200 is:

  2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199

Prime Factorisation

- to **factor** a number n is to write it as a product of other numbers: **n=a × b × c**
- note that factoring a number is relatively hard compared to multiplying the factors together to generate the number
- the **prime factorisation** of a number n is when its written as a product of primes
  - eg. $91=7\times13$; $3600=2^4\times3^2\times5^2$

$$a = \prod_{p \in P} p^{a_p}$$

Relatively Prime Numbers & GCD

- two numbers a, b are **relatively prime** if have **no common divisors** apart from 1
  - eg. 8 & 15 are relatively prime since factors of 8 are 1,2,4,8 and of 15 are 1,3,5,15 and 1 is the only common factor
- conversely can determine the greatest common divisor by comparing their prime factorizations and using least powers
  - eg. $300=2^1\times3^1\times5^2$ $18=2^1\times3^2$
  - hence $GCD(18,300)=2^1\times3^1\times5^0=6$

Fermat's Theorem

If **p is prime** and **a is a positive integer** not divisible by p, then

$$a^{p-1} \bmod p = 1$$

While gcd(a,p)=1

- also known as Fermat's Little Theorem
- Alternatively We can say
- If p is a prime and a is a positive integer, then a power n **= a (mod p)**

Two no.are relatively prime if they have no prime factors in common.i.e., only one divisor and GCD is also 1.

- useful in public key and primality testing

Euler Totient Function ø(n)

- when doing arithmetic modulo n
- **complete set of residues** is: `0..n-1`
- **reduced set of residues** is those numbers (residues) which are relatively prime to n
  - eg for n=10,
  - complete set of residues is {0,1,2,3,4,5,6,7,8,9}
  - reduced set of residues is {1,3,7,9}
- number of elements in reduced set of residues is called the **Euler Totient Function ø(n)**

Euler Totient Function $\varnothing(n)$

- to compute ø(n) need to count number of elements to be excluded
- in general need prime factorization, but
  - for p (p prime) $\varnothing(p) = p-1$
  - for p.q (p,q prime) $\varnothing(p.q) = (p-1)(q-1)$
- eg.
  - $\varnothing(37) = 36$
  - $\varnothing(21) = (3-1)\times(7-1) = 2\times6 = 12$

Euler's Theorem

- a generalisation of Fermat's Theorem
- **$a^{ø(n)} \bmod N = 1$**
  - where gcd(a,N)=1 for every a,n are relatively prime.
- eg.
  - $a=3; n=10;\ ø(10)=4;$
  - hence $3^4 = 81 = 1 \bmod 10$
  - $a=2; n=11;\ ø(11)=10;$
  - hence $2^{10} = 1024 = 1 \bmod 11$

Primality Testing

- often need to find large prime numbers
- traditionally done using **trial division**
  - ie. divide by all numbers (primes) in turn less than the square root of the number
  - only works for small numbers
- alternatively can use statistical primality tests based on properties of primes
  - for which all primes numbers satisfy property
  - but some composite numbers, called pseudo-primes, also satisfy the property

Miller Rabin Algorithm

- a test based on Fermat's Theorem
- algorithm is:

TEST (n) is:

1. Find integers k, q, $k > 0$, q odd, so that $(n-1)=2^k q$
2. Select a random integer a, $1<a<n-1$
3. **if** $a^q \bmod n = 1$ **then** return ("maybe prime");
4. **for** $j = 0$ **to** $k - 1$ **do**
   5. **if** ($a^{2^j q} \bmod n = n-1$)
      then return(" maybe prime ")
6. return ("composite")

Probabilistic Considerations

- if Miller-Rabin returns "composite" the number is definitely not prime
- otherwise is a prime or a pseudo-prime
- chance it detects a pseudo-prime is < ¼
- hence if repeat test with different random a then chance n is prime after t tests is:
  - Pr(n prime after t tests) = $1-4^{-t}$
  - eg. for t=10 this probability is > 0.99999

Prime Distribution

- prime number theorem states that primes occur roughly every (`ln n`) integers
- since can immediately ignore evens and multiples of 5, in practice only need test `0.4 ln(n)` numbers of size n before locate a prime
  - note this is only the "average" sometimes primes are close together, at other times are quite far apart

Chinese Remainder Theorem

- used to speed up modulo computations
- working modulo a product of numbers
  - eg. mod $M = m_1m_2..m_k$
- Chinese Remainder theorem lets us work in each moduli m_i separately
- since computational cost is proportional to size, this is faster than working in the full modulus M

Chinese Remainder Theorem

- can implement CRT in several ways
- It is possible to reconstruct the integers in a certain range from their resudues modulo to attain set of pairwise relatively prime moduli.
- to compute (A mod M) we can first compute all (a_i mod m_i) separately and then combine results to get answer using:

$$A = \left(\sum_{i=1}^{k} a_i c_i \right) \bmod M$$

$$c_i = M_i \times \left(M_i^{-1} \bmod m_i \right) \quad \text{for } 1 \le i \le k$$

Primitive Roots

- from Euler's theorem have $a^{ø(n)} \bmod n=1$
- consider $a^m \bmod n=1$, $GCD(a,n)=1$
  - must exist for m= ø(n) but may be smaller
  - once powers reach m, cycle will repeat
- if smallest is m= ø(n) then a is called a **primitive root**
- if p is prime, then successive powers of a "generate" the group mod p
- these are useful but relatively hard to find

Discrete Logarithms or Indices

- the inverse problem to exponentiation is to find the **discrete logarithm** of a number modulo p
- that is to find x where $a^x = b \bmod p$
- written as $x=\log_a b \bmod p$ or $x=ind_{a,p}(b)$
- if a is a primitive root then always exists, otherwise may not
  - $x = \log_3 4 \bmod 13$ (x st $3^x = 4 \bmod 13$) has no answer
  - $x = \log_2 3 \bmod 13 = 4$ by trying successive powers
- whilst exponentiation is relatively easy, finding discrete logarithms is generally a **hard** problem

THANK YOU