Elliptic Curve Key Exchange and AES

By Tommy He, mentor Ileana Vasu

ECIES (Elliptic Curve Integrated Encryption Scheme)

- Symmetric Key Encryption System such as AES
- Method of public-key cryptography ECC for safe Diffie-Hellman Key Exchange
- Good balance between speed/security

$$y^2 = x^3 + ax + b$$

Symmetric Key Encryption: able to encrypt and decrypt a message using the same cipher

ECC vs. RSA

ECC and RSA bit size equivalence to obtain similar security:

160	1024
256	3072
384	7680

BUT, ECC is harder to implement and program slower http://royalforkblog.github.io/2014/09/04/ecc/

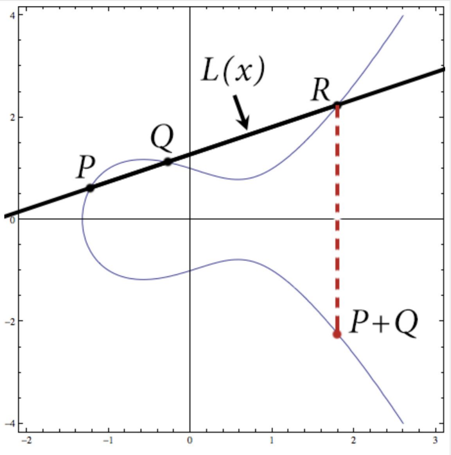
Sample Curve

Point Addition/Doubling
in the curve

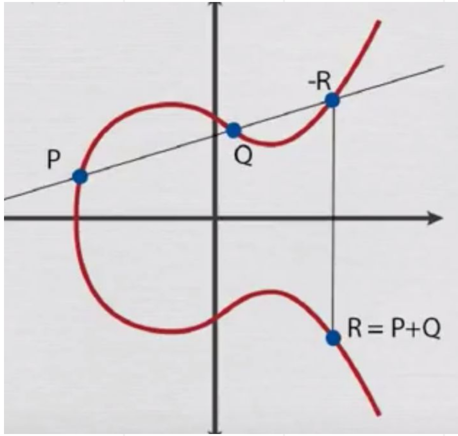
$$y^2 = x^3 - 3x + 4$$

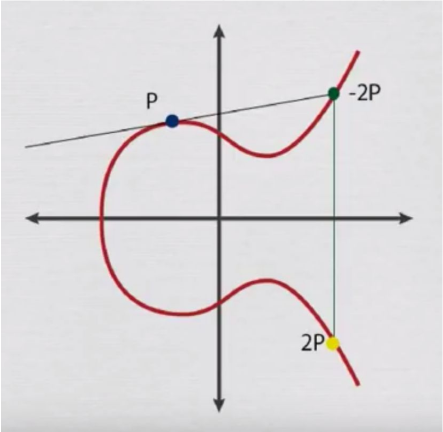
Point Addition, Doubling

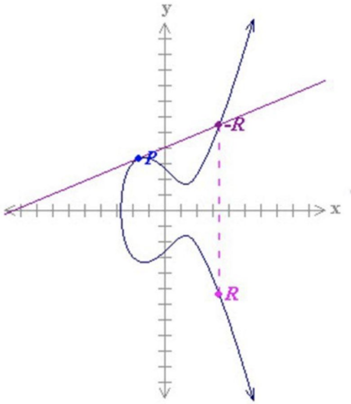
$$y^2 = x^3 + ax + b$$



Adding P and Q on an elliptic curve.





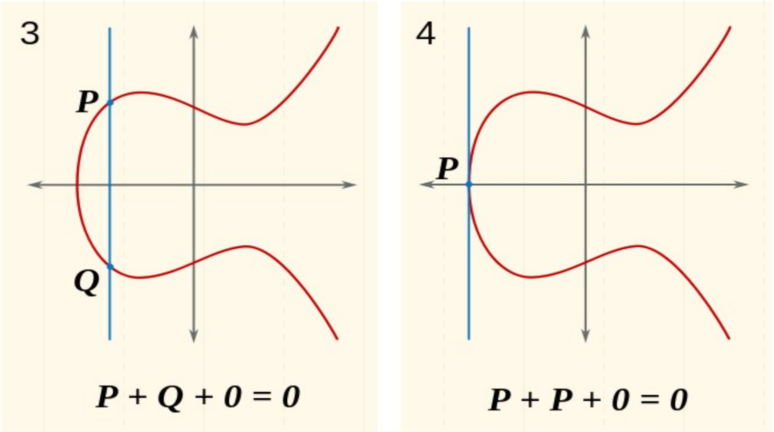


Point Addition

Point Doubling

-P is P reflected across x-axis
3P = P + 2P
4P = P + 3P...

Point at Infinity



P + (-P) gives $\mathcal{O}$
If derivative at P = 0, also $\mathcal{O}$

The Graph, why it works

- The graph of $y^2 = x^3 + ax + b$
- The graph will become a singularity if discriminant equals 0:

$$\Delta = -16(4a^3 + 27b^2) = 0$$

- All (x,y) on the curve restricted over $\mathbb{Z}_p$ (all integers taken mod prime p)
- Initial Generator point G
- Order of G is least n such that nG becomes $\mathcal{O}$
- Properties:
    - a(bP) = b(aP)
    - P + Q = Q + P

A “Point at Infinity” called $\mathcal{O}$ (the vertical line “point”) from P + (-P)

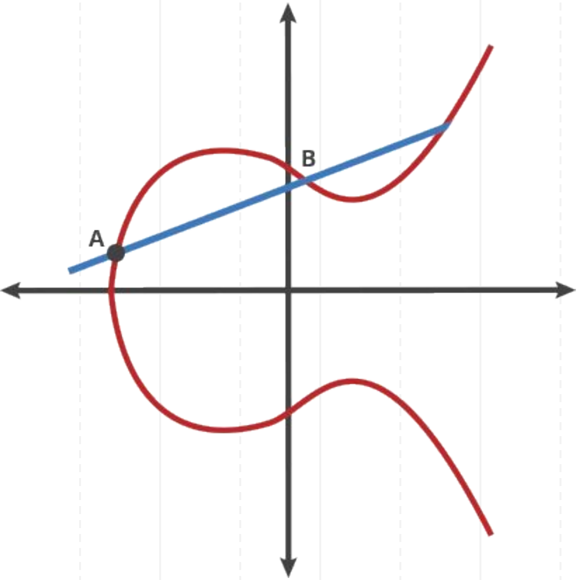
The curve $y^2 = x^3 + 7$ is used in Bitcoin

Valid Curves $b = 1, a \in [-2, 3]$

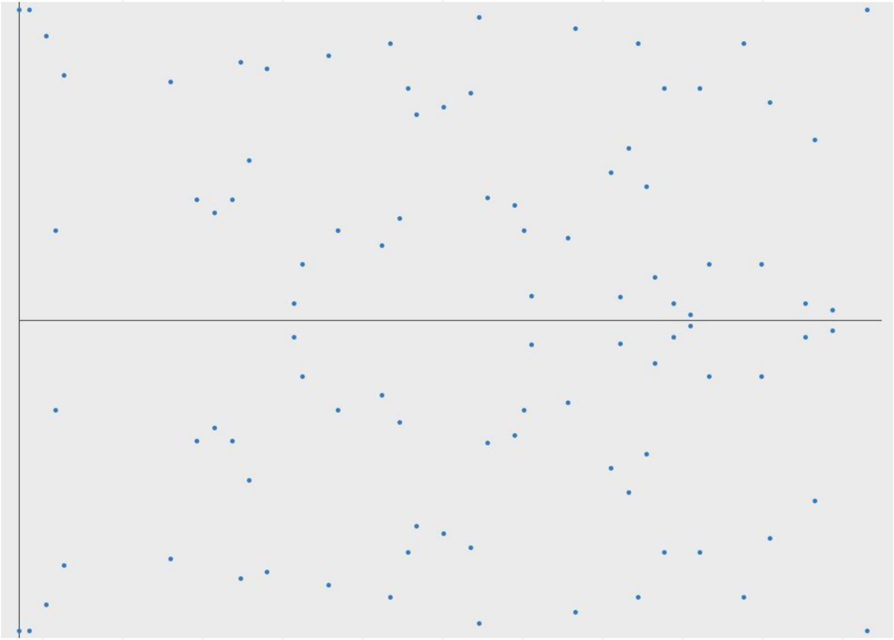
Singularities

$$\Delta = -16(4a^3 + 27b^2) = 0$$

Starting Point A
Not restricted to $\mathbb{Z}_p$



$$y^2 = x^3 - x + 1 \pmod{97}$$



Parameters

- We now have {p, a, b, G, n}
- p to determine our field for the curve $\mathbb{Z}_p$
- a, b are parameters for the curve itself $y^2 = x^3 + ax + b$
- Known to everyone
- n is ord(G)

Elliptic Curve Diffie-Hellman Key Exchange Goal

- Bob wants to send message to Alice
- Create a shared private key for Alice/Bob, Eve no access
- With this shared key, Alice/Bob can use a symmetric encryption system (like AES)

Elliptic Curve Diffie Hellman Key Exchange

Elliptic Curve Discrete Logarithm Problem acts as the Trapdoor function

Alice:

- Private Key: $\alpha \in [1, n-1]$
- Public: $A = \alpha G$
- A released to everyone
- Finds $\alpha B = \alpha(\beta G) = P$

Bob:

- Private Key: $\beta \in [1, n-1]$
- Public: $B = \beta G$
- B released to everyone
- Finds $\beta A = \beta(\alpha G) = P$

Eve (Outsider):

- Has A B
- ECDLP prevents Eve from easily finding α β
- No access to P

ECDLP:

- Hard to find n given initial point nG and generator point G

Elliptic Curve Diffie-Hellman

Bob

Bob picks private key β

$$1 \le \beta \le n-1$$

Computes

$$B = \beta G$$

Receives

$$A = (x_A, y_A)$$

Computes

$$P = \beta\alpha G$$

Eve

$$y^2 = x^3 + ax + b$$

p

a

b

G

n

h

A

B

Alice

Alice picks private key α

$$1 \le \alpha \le n-1$$

Computes

$$A = \alpha G$$

Receives

$$B = (x_B, y_B)$$

Computes

$$P = \alpha\beta G$$

Symmetric Encryption

- Let mk (masterkey) be our x-coord of $\alpha\beta G$
- Encrypt/Decrypt with symmetric encryption system
- Ex. AES

- Put mk through a Key Derivation Function (stretch/strengthen)
- Alice/Bob have access to mk and will produce the same cipher key from the KDF, Eve can not

Other Uses of Elliptic Curves

- Elliptic Curve Digital Signature Algorithm (ECDSA)
- Edwards-curve Digital Signature Algorithm (EcDSA)
- Elliptic Curve MQV
- ElGamal (uses curve for the encryption)

Thank You!

AES-128

32	88	31	e0
43	5a	31	37
f6	30	98	07
a8	8d	a2	34

- 1 bit is a binary digit (either 1 or 0)
- 1 byte is essentially 8 bits
- AES-128 uses 128 bit (16 byte) keys and states
- We first want to convert our input string into a state (4 by 4 block of bytes). We may use the ASCII conversion table, which works perfectly with hexadecimal bytes since it has 128 conversions
- Every 2 digit hexadecimal is 1 byte

ASCII/Unicode

Used to convert strings of text into numbers, which can then go through the cryptological system

ASCII:

- Works for English language only
- Converts 2^7 (128) of the symbols into integers 0 to 127.
- Smaller memory required than unicode

Unicode:

- Works for most characters in the world
- Has 2^16 conversions
- More universal
- Because of its large size, unicode might require the message to be broken up into smaller parts, which would each separately go through the cryptography

Dec	Bin	Hex	Char	Dec	Bin	Hex	Char	Dec	Bin	Hex	Char	Dec	Bin	Hex	Char
0	0000 0000	00	[NUL]	32	0010 0000	20	space	64	0100 0000	40	@	96	0110 0000	60	`
1	0000 0001	01	[SOH]	33	0010 0001	21	!	65	0100 0001	41	A	97	0110 0001	61	a
2	0000 0010	02	[STX]	34	0010 0010	22	"	66	0100 0010	42	B	98	0110 0010	62	b
3	0000 0011	03	[ETX]	35	0010 0011	23	#	67	0100 0011	43	C	99	0110 0011	63	c
4	0000 0100	04	[EOT]	36	0010 0100	24	$	68	0100 0100	44	D	100	0110 0100	64	d
5	0000 0101	05	[ENQ]	37	0010 0101	25	%	69	0100 0101	45	E	101	0110 0101	65	e
6	0000 0110	06	[ACK]	38	0010 0110	26	&	70	0100 0110	46	F	102	0110 0110	66	f
7	0000 0111	07	[BEL]	39	0010 0111	27	'	71	0100 0111	47	G	103	0110 0111	67	g
8	0000 1000	08	[BS]	40	0010 1000	28	(	72	0100 1000	48	H	104	0110 1000	68	h
9	0000 1001	09	[TAB]	41	0010 1001	29	)	73	0100 1001	49	I	105	0110 1001	69	i
10	0000 1010	0A	[LF]	42	0010 1010	2A	*	74	0100 1010	4A	J	106	0110 1010	6A	j
11	0000 1011	0B	[VT]	43	0010 1011	2B	+	75	0100 1011	4B	K	107	0110 1011	6B	k
12	0000 1100	0C	[FF]	44	0010 1100	2C	,	76	0100 1100	4C	L	108	0110 1100	6C	l
13	0000 1101	0D	[CR]	45	0010 1101	2D	-	77	0100 1101	4D	M	109	0110 1101	6D	m
14	0000 1110	0E	[SO]	46	0010 1110	2E	.	78	0100 1110	4E	N	110	0110 1110	6E	n
15	0000 1111	0F	[SI]	47	0010 1111	2F	/	79	0100 1111	4F	O	111	0110 1111	6F	o
16	0001 0000	10	[DLE]	48	0011 0000	30	0	80	0101 0000	50	P	112	0111 0000	70	p
17	0001 0001	11	[DC1]	49	0011 0001	31	1	81	0101 0001	51	Q	113	0111 0001	71	q
18	0001 0010	12	[DC2]	50	0011 0010	32	2	82	0101 0010	52	R	114	0111 0010	72	r
19	0001 0011	13	[DC3]	51	0011 0011	33	3	83	0101 0011	53	S	115	0111 0011	73	s
20	0001 0100	14	[DC4]	52	0011 0100	34	4	84	0101 0100	54	T	116	0111 0100	74	t
21	0001 0101	15	[NAK]	53	0011 0101	35	5	85	0101 0101	55	U	117	0111 0101	75	u
22	0001 0110	16	[SYN]	54	0011 0110	36	6	86	0101 0110	56	V	118	0111 0110	76	v
23	0001 0111	17	[ETB]	55	0011 0111	37	7	87	0101 0111	57	W	119	0111 0111	77	w
24	0001 1000	18	[CAN]	56	0011 1000	38	8	88	0101 1000	58	X	120	0111 1000	78	x
25	0001 1001	19	[EM]	57	0011 1001	39	9	89	0101 1001	59	Y	121	0111 1001	79	y
26	0001 1010	1A	[SUB]	58	0011 1010	3A	:	90	0101 1010	5A	Z	122	0111 1010	7A	z
27	0001 1011	1B	[ESC]	59	0011 1011	3B	;	91	0101 1011	5B	[	123	0111 1011	7B	{
28	0001 1100	1C	[FS]	60	0011 1100	3C	<	92	0101 1100	5C	\	124	0111 1100	7C	|
29	0001 1101	1D	[GS]	61	0011 1101	3D	=	93	0101 1101	5D	]	125	0111 1101	7D	}
30	0001 1110	1E	[RS]	62	0011 1110	3E	>	94	0101 1110	5E	^	126	0111 1110	7E	~
31	0001 1111	1F	[US]	63	0011 1111	3F	?	95	0101 1111	5F	_	127	0111 1111	7F	[DEL]

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

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

Plaintext string to hexadecimal state

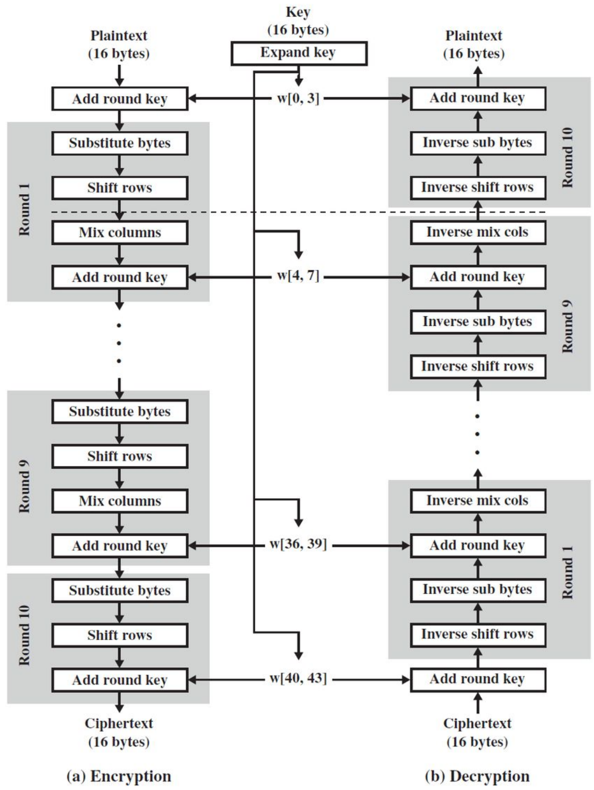
AES Encryption

right (low-order) nibble; left (high-order) nibble

	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

S-box (Substitution Box) for AES

Rcon Constants (Base 16)			
Round	Constant(Rcon)	Round	Constant(Rcon)
1	01 00 00 00	6	20 00 00 00
2	02 00 00 00	7	40 00 00 00
3	04 00 00 00	8	80 00 00 00
4	08 00 00 00	9	1B 00 00 00
5	10 00 00 00	10	36 00 00 00



Key Schedule- recursively created with the resulting cipher key and contains every round key.

1. Last column of every key is first SubByted, then RotWorded (shifted one by up), Xored with the first column and Xored with respective column in the Rcon
2. The newly generated column i is Xored with the column i-3 to get the next column

AES Process

Cipher Key				Round key 1				Round key 2				Round key 3				…	Round key 10			
2b	28	ab	09	a0	88	23	2a	f2	7a	23	73	3d	47	1e	6d		d0	c9	e1	b6
7e	ae	f7	cf	fa	54	a3	6c	c2	96	a3	59	80	16	23	7a		14	ee	3f	63
15	d2	15	4f	fe	2c	39	76	95	b9	39	f6	47	fe	7e	88		f9	25	0c	0c
16	a6	88	3c	17	b1	39	05	f2	43	39	7f	7d	3e	44	3b		a8	89	c8	a6

Add Initial round key (input cipher)
Xor each column with each other

32	88	31	e0
43	5a	31	37
f6	30	98	07
a8	8d	a2	34

$\oplus$

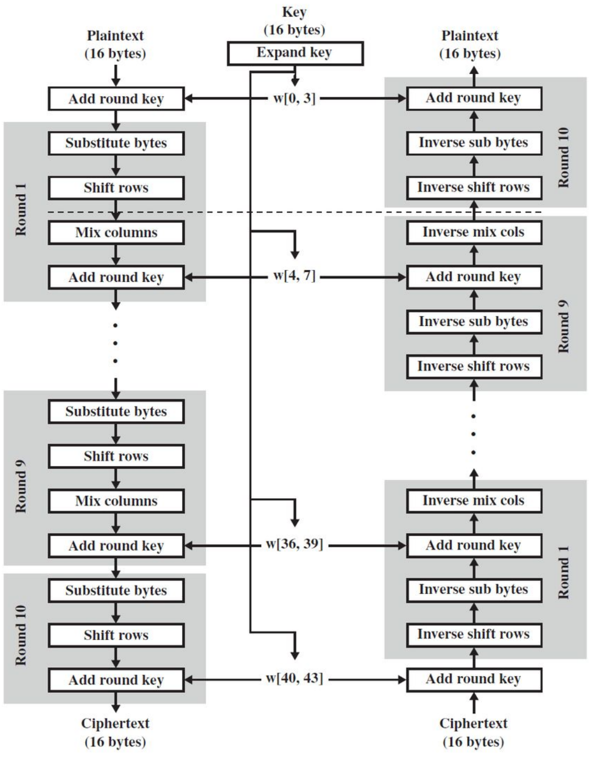
2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

State (to be encrypted)

Cipher key

19	a0	9a	e9
3d	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08

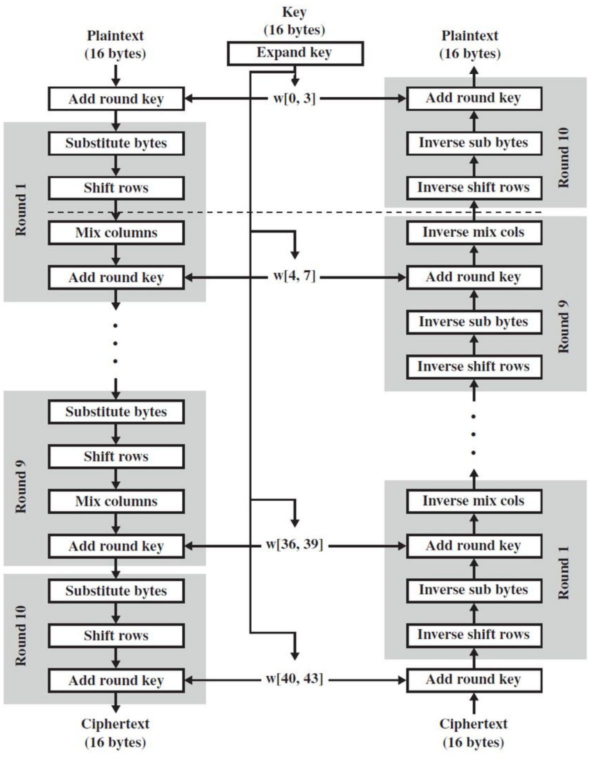
Number of rounds used is based on the key size:
128 bit: 10
192 bit: 12
256 bit: 14



(a) Encryption
(b) Decryption

right (low-order) nibble

left (high-order) nibble	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



(a) Encryption

(b) Decryption

SubByted (sub every byte according to s-box)

19	a0	9a	e9
3d	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

ShiftRows (row i moves to the left by i)

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

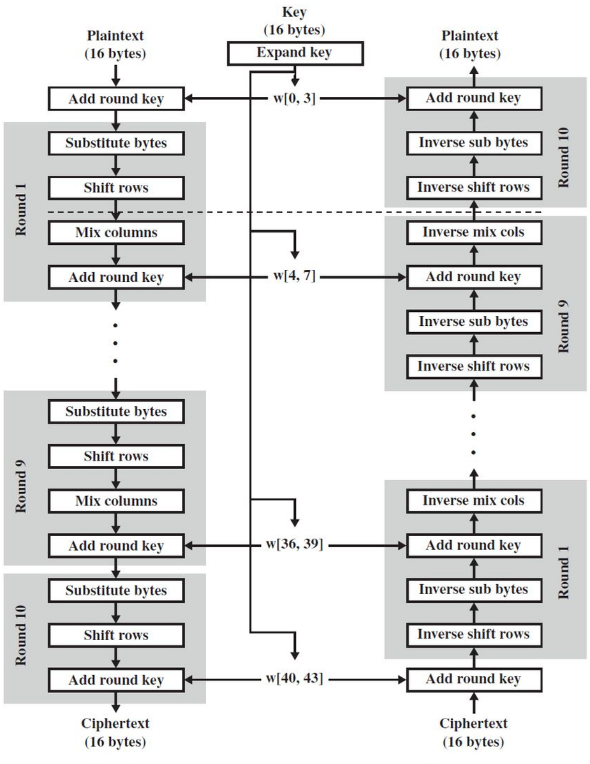
d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

MixColumns (mult each column by Matrix)

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

04	e0	48	28
66	cb	f8	06
81	19	d3	26
e5	9a	7a	4c



(a) Encryption

(b) Decryption

Add round specific keys (Xor each column again)

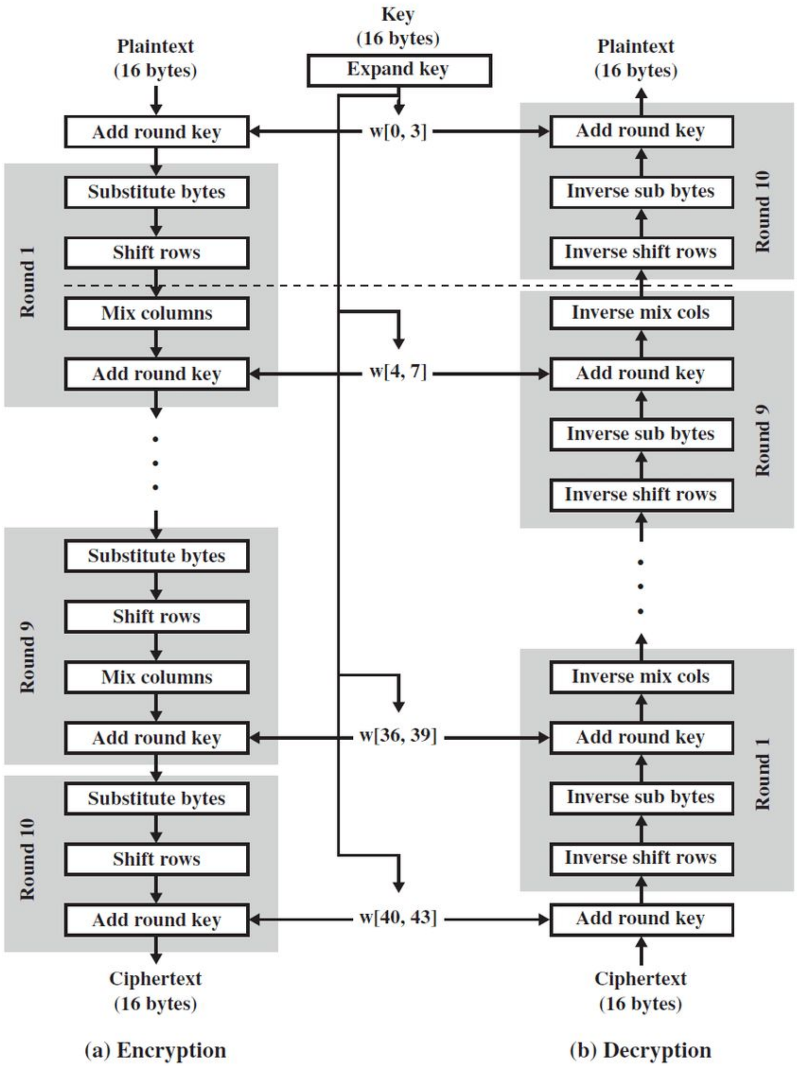
04	e0	48	28
66	cb	f8	06
81	19	d3	26
e5	9a	7a	4c

$\oplus$

a0	88	23	2a
fa	54	a3	6c
fe	2c	39	76
17	b1	39	05

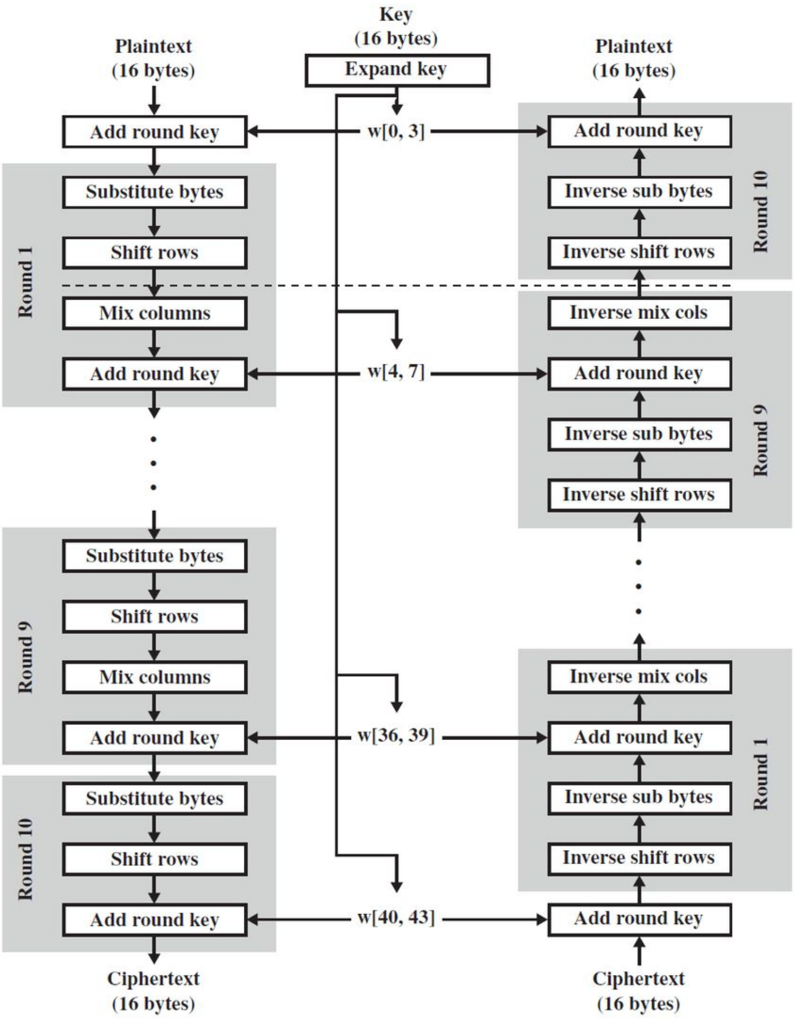
Round key 1

a4	68	6b	02
9c	9f	5b	6a
7f	35	ea	50
f2	2b	43	49



AES Decryption

- Same key is still used
- ^^ Allows Bob to encrypt the message and Alice to decrypt it (both have the right cipher key)
- The order is simply reversed and steps inversed



(a) Encryption

(b) Decryption