SKJ 4143

CRYPTOGRAPHY & APPLICATION

W2

Modular Arithmetic

(a system of arithmetic for integers where numbers "wrap around" when reaching a certain value, called "modulus")

① $12^{42} \bmod 25$

Fixidy exponentiation rule

$(12^2)^{21} \bmod 25 = (25 \cdot 731 + 287)^{21} \bmod 25$

$= 3^{21} \bmod 25$

$= 729 \cdot 9 \bmod 25$

$= 4$

$12^{42} \equiv 4 \bmod 25$

② $3 \cdot 2 / 5 \bmod 7$

finding the equivalence of 5^{-1} in mod 7 by using the concept of multiplicative inverse

$a \cdot a^{-1} \equiv 1 \bmod m$

$3 \cdot 2 \cdot 15 \bmod 7 = 3 \cdot 2 \cdot 5^{-1} \bmod 7$

$= 3 \cdot 2 \cdot 3 \bmod 7$

$= 18 \bmod 7$

$= 4$

$5 \cdot x \equiv 1 \bmod 7$

$5 \cdot 1 \not\equiv 1 \bmod 7$

$5 \cdot 2 \not\equiv 1 \bmod 7$

$5 \cdot 3 \equiv 1 \bmod 7$

in modulo 7, $3 = 5^{-1}$ because $5 \cdot 5^{-1} = 1 \bmod 7$

Affine cipher : $y = a \cdot x + b \bmod 26$

3. This problem deals with the affine cipher with the key parameters $a = 7$, $b = 22$.

Encrypt the text below:

I LOVE USIM

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

$a = 7$, $b = 22$, $y = a \cdot x + b \bmod 26$

I = 7·8 + 22 mod 26 = 0 (A)

L = 7·11 + 22 mod 26 = 21 (V)

O = 7·14 + 22 mod 26 = 16 (G)

V = 7·21 + 22 mod 26 = 15 (P)

E = 7·4 + 22 mod 26 = 24 (Y)

U = 7·20 + 22 mod 26 = 6 (G)

S = 7·18 + 22 mod 26 = 12 (M)

I = 7·8 + 22 mod 26 = 0 (A)

M = 7·12 + 22 mod 26 = 2 (C)

A YOUNG GEAR

Playfair matrix

6. Using this Playfair matrix:

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

Encrypt this message: ***leader in the integration of naqli and aqli knowledge.***

LE = GO

AD = BA

ER = KD

IN = GQ

TH = VF

EI = WG

NT = MV

EG = DN

RA = DB

TI = XE

ON = ES

OF = KL

NA = PB

GL = SJ

IA = HB

ND = GU

AQ = BP

LI = FL

NY = EM

OW = NO

LE = GO

DG = GN

EZ = OU

① if it's in the same column, take 1 to down

② if it's in the same row, take 1 to right

③ if 2 same words, add padding, eg: EE : EZ ZE

④ if the last word is single, add padding, eg E : EZ

Vigenere cipher

4. This problem explores the use of a one-time pad version of the Vigenere cipher. In this scheme, the key is a stream of random numbers between 0 and 25. For example, if the key is 3 19 5 ....., then the first letter of plaintext is encrypted with a shift of 3 letters, the second with a shift of 19 letters, the third with a shift of 5 letters, and so on.

a. Encrypt the plaintext ***sendmoremoney*** with the key stream

9 5 1 7 25 15 21 14 11 11 2 8 9

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

a) S = (18 + 9) mod 26 = 1 (B)

E = (4 + 5) mod 26 = 9 (E)

N = (13 + 1) mod 26 = 14 (O)

D = (3 + 7) mod 26 = 10 (L)

M = (12 + 25) mod 26 = 11 (M)

O = (14 + 15) mod 26 = 3 (D)

R = (17 + 21) mod 26 = 12 (M)

E = (4 + 14) mod 26 = 18 (R)

M = (12 + 11) mod 26 = 23 (X)

O = (14 + 11) mod 26 = 25 (Z)

N = (13 + 2) mod 26 = 15 (P)

E = (4 + 7) mod 26 = 12 (M)

Y = (24 + 9) mod 26 = 7 (H)

= BEOLTDMSXZPMH

W4: DES

Plaintext, M : ABCDEF0123456789
Key, K : C333344445555672

Step 1: Convert to binary

M = 1010 1011 1100 1101 1110 1111 0000 0001
0010 0011 0100 0101 0110 0111 1000 1001

K = 1100 0011 0011 0011 0011 0100 0100 0100
0100 0101 0101 0101 0101 0110 0111 0010

Step 2: Permutate the key through the PC-1 table

PC-1:

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

k = 1000 0011 0011 0011 0011 0100 0100 0100
0100 0101 0101 0101 0101 0110 0111 0010

↓ PC-1

K_0 = 0000000 0111110 0010000 1101110
1100001 1011111 0000000 0000110

Step 3: Rotating each half (i = 1, 2, 9, 16 rotate 1 left the rest rotate 2 left)

0000000 1111110 0110000 1101110
1100001 1011111 0000000 0000110

↓ i = 1

K_1 = 0000001 1111100 1100001 1011100
1000011 0111110 0000000 0001101

↓ i = 2

K_2 = 0000011 1111001 1000011 0111000
0000110 1111100 0000000 0011011

↓ i = 3

K_3 = 0001111 1100110 0001101 1100000
0011011 1110000 0000000 1101100

↓ i = 4

K_4 = 0111111 0011000 0110111 0000000
1101111 1000000 0000011 0110000

↓ i = 5

K_5 = 1111100 1100001 1011100 0000001
0111110 0000000 0001101 1000011

↓ i = 6

K_6 = 1110011 0000110 1110000 0000111
1111000 0000000 0110110 0001101

↓ i = 7

K_7 = 1001100 0011011 1000000 0011111
1100000 0000001 1011000 0110111

↓ i = 8

K_8 = 0110000 1101110 0000000 1111110
0000000 0000110 1100001 1011111

K_8 = 0110000 1101110 0000000 1111110
0000000 0000110 1100001 1011111

↓ i = 9

K_9 = 1100001 1011100 0000001 1111100
0000000 0001101 1000011 0111110

↓ i = 10

K_{10} = 0000110 1110000 0000111 1110011
0000000 0110110 0001101 1111000

↓ i = 11

K_{11} = 0011011 1000000 0011111 1001100
0000001 1011000 0110111 1100000

↓ i = 12

K_{12} = 1101110 0000000 1111110 0110000
0000110 1100001 1011111 0000000

↓ i = 13

K_{13} = 0111000 0000011 1111001 1000011
0011011 0000110 1111100 0000000

↓ i = 14

K_{14} = 1100000 0000111 1100110 0001101
1101100 0011011 1110000 0000000

↓ i = 15

K_{15} = 0000000 0011111 0011000 0110111
0110000 1101111 1000000 0000011

↓ i = 16

K_{16} = 0000000 0111110 0110000 1101110
1100001 1011111 0000000 0000110

Step 4: Concatenate

Step 5: Permutate the key through PC-2 table

PC-2:

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

K_1 = 001100 001011 001011 110000 100100 010000 001111 000010
K_2 = 101100 001111 100001 010100 010101 010010 001111 000100
K_3 = 000001 000110 111101 010010 001100 001000 000011 001101
K_4 = 011001 100111 010100 110101 010000 101011 010010 000111
K_5 = 110011 111000 010101 000001 001011 100010 010110 101001
K_6 = 010010 111100 001010 111011 001010 100101 100101 000011
K_7 = 101111 011001 000110 001010 010001 101100 000100 110010
K_8 = 001100 100000 101011 001011 110001 010000 110101 00100
K_9 = 001110 000011 101011 010001 010001 011001 011001 010000
K_{10} = 000101 010111 110001 001100 110110 011000 010001 101000
K_{11} = 010000 100110 010111 010100 000010 001101 111000 001100
K_{12} = 010111 001100 110100 100101 000110 000111 010010 110100
K_{13} = 110000 111010 000100 101011 101010 010100 100010 100001
K_{14} = 101010 011000 011010 100111 100000 100110 101000 010011
K_{15} = 111100 010001 101010 001010 101101 110000 001100 010100
K_{16} = 011000 000001 101011 101010 100001 110000 001000 010110

Step 6: Permute M through 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

M = 1010 1011 1100 1101 1110 1111 0000 0001
0010 0011 0100 0101 0110 0111 1000 1001

↓ IP

0110 0110 0000 0000 0110 0110 1111 1111
1000 0111 0101 0101 1000 0111 0101 0101

Step 7: Encode the data

$$L_n = R_{n-1}$$
$$R_n = L_{n-1} \oplus P(S(K_n \oplus E(R_{n-1})))$$
$1 \le n \le 16$

L_0 = 0110 0110 0000 0000 0110 0110 1111 1111
R_0 = 1000 0111 0101 0101 1000 0111 0101 0101

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

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

S_1:

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2:

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3:

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4:

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5:

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6:

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7:

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8:

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

L_1 = 1000 0111 0101 0101 1000 0111 0101 0101
R_1 = 0101 0000 0111 0000 1100 0110 0000 1100
L_2 = 0101 0000 0111 0000 1100 0110 0000 1100
R_2 = 0111 1000 1110 0000 1011 0111 0111 0000
L_3 = 0111 1000 1110 0000 1011 0111 0111 0000
R_3 = 1101 1100 1011 1000 1111 0010 0001 0101
L_4 = 1101 1100 1011 1000 1111 0010 0001 0101
R_4 = 0011 0110 1010 1111 1000 0011 1001 1110
L_5 = 0011 0110 1010 1111 1000 0011 1001 1110
R_5 = 0101 1110 1100 1110 1010 0001 0111 0010
L_6 = 0101 1110 1100 1110 1010 0001 0111 0010
R_6 = 0100 0101 0010 1101 1100 1010 1011 0101

L_7 = 0100 0101 0010 1101 1100 1010 1011 0101
R_7 = 0101 1111 1111 1110 0011 1000 1101 1111
L_8 = 0101 1111 1111 1110 0011 1000 1101 1111
R_8 = 1110 1101 01111 1000 1000 0101 1100 1100
L_9 = 1110 1101 0111 1000 1000 0101 1100 1100
R_9 = 0000 0000 1011 1000 1001 1011 0000 0111
L_{10} = 0000 0000 1011 1000 1001 1001 0000 0111
R_{10} = 1110 1010 1001 0100 1110 1100 1101 1101
L_{11} = 1110 1010 1001 0100 1110 1100 1101 1101
R_{11} = 0100 0010 1101 1100 1011 1111 1001 0100
L_{12} = 0100 0010 1101 1100 1011 1111 1001 0100
R_{12} = 1000 0100 0001 1001 1001 1100 1110 1001

L_{13} = 1000 0100 0001 1001 1001 1100 1110 1001
R_{13} = 1111 0110 0010 0110 0001 1100 0111 1101
L_{14} = 1111 0110 0010 0110 0001 1100 0111 1101
R_{14} = 0001 1101 0011 0001 0011 0111 1111 1000
L_{15} = 0001 1101 0011 0001 0011 0111 1111 1000
R_{15} = 1000 0000 1011 1011 1100 1101 1000 1011
L_{16} = 1000 0000 1011 1011 1100 1101 1000 1011
R_{16} = 0111 1100 1110 1100 1100 1010 1010 0111

Step 8: Permutate the encoded data through IP^{-1} table

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

$R_{16}L_{16}$ = 0111 1100 1110 1100 1100 1010 1010 0111 1000 0000 1011 1011 1100 1101 1000 1011

↓ IP^{-1}

C_2 = 0010 1011 0010 0111 0101 1001 0111 1110 0110 0000 0111 0001 0101 1100 1011 1111

Step 9: Convert to hexa

C_2 : 0010 1011 0010 0111 0101 1001 0111 1110 0110 0000 0111 0001 0101 1100 1011 1111

↓ $C_2 \to C_{16}$

C_{16} = 2B27 597E 6071 5CBF #

K_1 = 001100 001011 001011 110000 100100 010000 001111 000010
$L_1 = R_0$ = 1000 0111 0101 0101 1000 0111 0101 0101
$R_1 = L_0 \oplus P(S(K_1 \oplus E(R_0)))$

$E(R_0)$ = 110000 001110 101010 101011 110000 001110 101010 101011

① $K_1 \oplus E(R_0)$ = 001100 001011 001011 110000 100100 010000 001111 000010
⊕ 110000 001110 101010 101011 110000 001110 101010 101011

111100 000101 100001 011011 010100 011110 100101 101001
s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8

S_1:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

② 111100 : 10b = 2d = 2nd row, 1110b = 14d = 14th column = 5d = 0101b

⋮

S_8:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	16
	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

101001 : 11b = 3d = 3rd row, 0100b = 4d = 4th column = 4d = 0100b

= 0101 0100 0001 1010
0011 1011 1101 0100

③ 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

↓ $P(S(K_1 \oplus E(R_0)))$

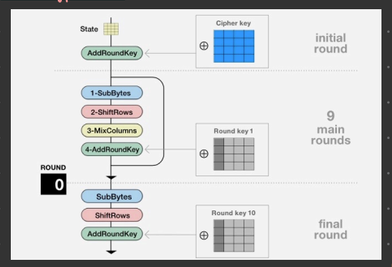
0110 0110 0000 0000 0110 0110 1111 1111

④ $R_1 = L_0 \oplus f(R_0, K_1)$ = 0110 0110 0000 0000 0110 0110 1111 1111
⊕ 0011 0110 0111 0000 1010 0000 1111 0011

0101 0000 0111 0000 1100 0110 0000 1100

W5: AES

Encryption:



① Add RoundKey

there'll be 16 sets of round key, by doing key scheduling

Previous block	4F	4E	4C	49	4E	45	20	4C	45	41	52	4E	49	4F	47	20
Round key		W0			W1				W2				W3			
	4E	45	57	20	4E	4F	52	4D	41	4C	20	43	4F	56	49	44
Result ⊕	01	0B	1B	69	00	0A	72	01	04	0D	72	0D	06	19	0E	64

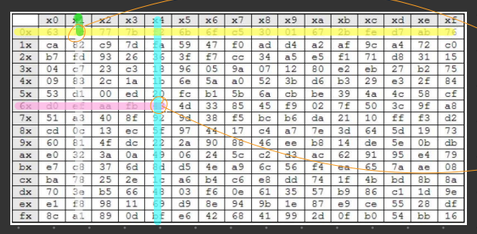
4F = 0100 1111
⊕ 4E ⊕ 0100 1110
0000 0001

State after roundkey:

01	00	04	06
0B	0A	0D	19
1B	72	72	0F
69	01	0D	64

② SubBytes

S-Box



	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xa	xb	xc	xd	xe	xf
0x	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1x	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2x	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3x	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4x	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5x	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6x	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7x	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8x	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9x	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
ax	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
bx	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
cx	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
dx	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
ex	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
fx	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Mark:

01	0B	1B	69
00	0A	72	01
04	0D	72	0D
06	19	0F	64

7C	63	F2	6F
2B	67	D7	AD
AF	40	40	AB
F9	7C	D7	43

③ Shift row

7C	63	F2	6F	(unchanged)
2B	67	D7	AD	(shift 1 left)
AF	40	40	AB	(shift 2 left)
F9	7C	D7	43	(shift 3 left)

shift rows →

7C	63	F2	6F
67	D7	AD	2B
40	AB	AF	40
43	F9	7C	D7

④ MixColumns

02	03	01	01		7C	63	F2	6F		52	C0	34
01	02	03	01		67	D7	AD	2B	=	31	19	25	2E
01	01	02	03		40	AB	AF	40		5E	E9	9E	A6
03	01	01	02		43	F9	7C	D7		25	30	F7	6F

C0 C1 C2 C3

Steps to find C0 (52)

02 03 01 01 · 7C 67 40 43

= (02 · 7C) ⊕ (03 · 67) ⊕ (01 · 40) ⊕ (01 · 43)

= 11111000 ⊕ 10101001 ⊕ 01000000 ⊕ 01000011

= 01010010 = 52

11111000
⊕ 10101001
⊕ 01000000
⊕ 01000011
01010010 → 52

7C · 02 = 01111100 × 10
= $(x^6+x^5+x^4+x^3+x^2)(x)$
= $x^7+x^6+x^5+x^4+x^3$ = 1111 1000

67 · 03 = 0110 0111 × 11
= $(x^6+x^5+x^2+x+1)(x+1)$
= $x^7+x^6+x^5+x^3+x^2+x^6+x^5+x^3+x^2+x+1$
= $x^7+x^5+x^3+x+1$ = 1010 1011

40 · 01 = 0100 0000 × 01
= $x^6 \times x^0$
= x^6 = 0100 0000

43 · 01 = 0100 0011 × 01
= $(x^6+x+1)(x^0)$
= x^6+x+1 = 0100 0011

* if result is $\geq x^8$, reduce using reduction Mod: $x^8+x^4+x^3+x+1$

Key scheduling

key (W = words = 4 bytes = 32 bits)

W_0	W_1	W_2	W_3
4E	4E	41	45
45	4F	4C	56
57	52	20	49
20	4D	43	44

k_2

$W_4 = W_0 \oplus g(W_3)$
= 4E 45 57 20 ⊕ B0 3B 1B 84

0100 1110 0100 0101 0101 0111 0010 0000
⊕ 1011 0000 0011 1011 0001 1011 1000 0100
1111 1110 0111 1110 0100 1100 1010 0100
F E 7 E 4 C A 4
W_4 = FE 7E 4C A4

$W_5 = W_1 \oplus W_4$
= 4E 4F 52 4D ⊕ FE 7E 4C A4
= B0 31 1E E9

$W_6 = W_2 \oplus W_5$
= 41 4C 20 43 ⊕ B0 31 1E E9
= F1 7D 3E AA

$W_7 = W_3 \oplus W_6$
= 4F 56 49 44 ⊕ F1 7D 3E AA
= BE 2B 77 EE

supposedly, we need to continue until k_{11}

g(W_3):
① rot (shift 1 to left)
4F 56 49 44
↓ shift 1 to left
56 49 44 4F

② Byte substitution (S-Box)
56 49 44 4F
↓

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xa	xb	xc	xd	xe	xf
0x	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1x	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2x	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3x	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4x	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5x	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6x	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7x	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8x	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9x	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
ax	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
bx	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
cx	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
dx	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
ex	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
fx	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

B1 3B 1B 84

③ Xor with round constant (for R_1 = 01 00 00 00)
B1 3B 1B 84
⊕ 01 00 00 00
B0 3B 1B 84

W6: Basic concept in number theory for public key cryptography

Euclidean algorithm, extended euclidean algorithm

5. Find the multiplicative inverse of 19 modulo 26.

$19x \equiv 1 \bmod 26$

① Euclidean algorithm

$26 = 19(1) + 7$ (divisor, quotient, remainder)

$19 = 7(2) + 5$

$7 = 5(1) + 2$

$5 = 2(2) + 1$

$2 = 1(2) + 0$

- step before remainder = 0
- $\gcd(19, 26) = 1$

② Extended euclidean

$1 = 5 + 2(-2)$

$1 = 5 + (7 + 5(-1))(-2)$

$1 = 5(3) + 7(-2)$

$1 = (19 + 7(-2))(3) + 7(-2)$

$1 = 19(3) + 7(-8)$

$1 = 19(3) + (26 + 19(-1))(-8)$

$1 = 19(11) + 26[-8]$ → $26x \bmod 26 = 0$

$19(11) \equiv 1 \bmod 26$

multiplicative inverse = 11 #

Fermat Theorem

▷ if p is prime, a is an integer not divisible by p, then

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

Euler Theorem

$a^{\phi(m)} \equiv 1 \bmod m$

Euler's phi Function

$p = m = 6 = \{0, 1, 2, 3, 4, 5\}$

euler's phi is to find how many of these are relative prime (numerically)

$\gcd(0,6) = 1$

$(0,6) = 0$

$(1,6) = 1$

$(2,6) = 2$

$(3,6) = 3$

$(4,6) = 2$

$(5,6) = 1$

in this case, there're 2

Totient

① $\phi(1) = 0$

② $\phi(p) = p - 1$ if p is prime

③ $\phi(m \times n) = \phi(m) \times \phi(n)$, m and n are relative prime

④ $\phi(p^e) = p^e - p^{e-1}$ if p is prime

a) $4^{-1} \bmod 7$

$a \cdot a^{-1} \equiv 1 \bmod n$

$4(x) \equiv 1 \bmod 7$

① Finding x (inverse of 4) using euler's theorem: $a^{\phi(m)} = 1 \bmod m$

$\phi(7) = 7 - 1 = 6$

$4^{(6)} \equiv 1 \bmod 7$

$4 \cdot 4^{(5)} \equiv 1 \bmod 7$

$4 \cdot 1024 \equiv 1 \bmod 7$

$4 \cdot 2 \equiv 1 \bmod 7$

$4^{-1} \bmod 7 \equiv 2$ #

b) $5^{-1} \bmod 12$

$5x \equiv 1 \bmod 12$

① Finding multiplications of prime numbers

$12 = 2 \cdot 6$, $6 = 2 \cdot 3$

$\phi(12) = \phi(2^2 \cdot 3) = \phi(2^2) \times \phi(3)$

$= (2^2 - 2^1) \cdot (3 - 3^0)$

$= 2 \cdot 3$

$= 6$

② Using eulers theorem, solve $5x \equiv 1 \bmod 12$

$5^6 \equiv 1 \bmod 12$

$5 \cdot 5^5 \equiv 1 \bmod 12$

$5 \cdot 3125 \equiv 1 \bmod 12$

$5 \cdot 5 \equiv 1 \bmod 12$

$5^{-1} \bmod 12 \equiv 5$ #

Chapter 7, 8: Public Key Algorithm; RSA, DHKE, El-Gamal

RSA

↳ Rivest Shamir Adleman

Alice

message, $x = 4$

Encryption

$y = x^e \equiv 4^3 \equiv 31 \bmod 33$

$y = 31$ →

Bob

key generation

1. $p = 3$, $q = 11$
2. $n = p \cdot q = 33$
3. $\varphi(n) = (3-1) \cdot (11-1) = 20$
4. K_{pub_B}, $e = 3$
5. K_{priv_B}, $d \equiv e^{-1} \equiv 7 \bmod 20$

$(K_{pub_B}, n) = (3, 33)$ ←

Decryption

$x \equiv y^d \equiv 31^7 \equiv 4 \bmod 33$

El-Gamal

↳ an encryption algorithm that is an extension of DHKE,

Alice

message, $x = 26$

key generation:

1. $i \in \{1, \dots, p-2\} = 5$ — equal to K_{prA}
2. ephemeral key, $K_E = \alpha^i \equiv 2^5 \equiv 3 \bmod 29$ — equal to K_{pubA}
3. masking key, $K_M = \beta^i \equiv 7^5 \equiv 16 \bmod 29$

Encryption

$y \equiv x \cdot K_M \equiv 26 \cdot 16 \equiv 10 \bmod 29$

$(p, \alpha, \beta) = (24, 2, 7)$ ←

$(y, K_E) = (10, 3)$ →

Bob

key generation

1. large prime, $p = 29$
2. $\alpha \in \mathbb{Z}_p^* = 2$
3. $K_{prB} = d \in \{2, \dots, p-2\} = 12$
4. $K_{pubB} = \beta = \alpha^d \equiv 2^{12} \equiv 7 \bmod 29$

Decryption

1. $K_M = K_E^d \equiv 3^{12} \equiv 16 \bmod 29$
2. $x \equiv y \cdot K_M^{-1} \equiv 10 \cdot 20 \equiv 26 \bmod 29$

DHKE

↳ a key exchange algorithm, not an encryption algorithm.

allow for secure key exchange for symmetric cryptography

Set up

1. Choose large prime, $p = 29$
2. Choose $\alpha \in \{1, 2, 3, \dots, p-2\}$
3. Publish p, α

Alice

key generation

1. $K_{prA} = a \in \{2, \dots, p-2\} = 5$
2. $K_{pubA} = A \equiv \alpha^a \equiv 2^5 \equiv 3 \bmod 29$

$A = 3$ →

← $B = 7$

Shared-key generation

$K_{AB} = B^a \equiv 7^5 \equiv 16 \bmod 29$

Bob

key generation

1. $K_{prB} = b \in \{2, \dots, p-2\} = 12$
2. $K_{pubB} = B \equiv \alpha^b \equiv 2^{12} \equiv 7 \bmod 29$

Shared-key generation

$K_{AB} = A^b \equiv 3^{12} \equiv 16 \bmod 29$

Chapter 7: Elliptic Curve Cryptosystem

the curve is non-singular → no self-intersections

* it has to satisfy $4a^3 + 27b^2 \bmod p \neq 0$

ECDH

↳ Elliptic-Curve Diffie-Hellman

Establish:
1. eliptic curve function, $y^2 = x^3 + (2)x + (2) \bmod 17$ (a = 2, b = 2)
2. base point, $P = (5,1)$

Alice

key generation

1. $K_{prA} = a \in \{2,3,\dots,\#E-1\} = 3$
2. $K_{pubA} = A = aP = 3P = (10,6)$

$A = (10,6)$ →

← $B = (7,11)$

session key computation

$T_{AB} = aB = 3(7,11) = (13,10)$

Bob

key generation

1. $K_{prB} = b \in \{2,3,\dots,\#E-1\} = 10$
2. $K_{pubB} = B = bP = 10P = (7,11)$

session key computation

$T_{AB} = bA = 10(10,6) = (13,10)$

calculating 3P:

$y^2 = x^3 + (2)x + (2) \bmod 17$ (a = 2, b = 2)

$P = (5,1)$ — $P = Q$, use point doubling

$2P = P + P = (5,1) + (5,1) = (6,3)$

$3P = 2P + P = (6,3) + (5,1) = (10,6)$

$P \neq Q$, use point addition

Finding $2P = P + P = (5,1) + (5,1)$

$P = Q$, use point doubling

Point doubling

$$s = \frac{3x_1^2 + a}{2y_1} \bmod p = \frac{3(5)^2 + 2}{2(1)} \bmod 17$$

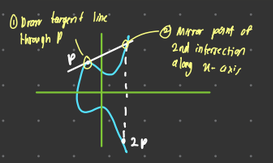
$$= 77 \cdot 2^{-1} \bmod 17$$

$$s = 77 \cdot 9 \equiv 13 \bmod 17$$

$$x_i = s^2 - x_1 - x_2 = 13^2 - 5 - 5 \equiv 6 \bmod 17$$

$$y_i = s(x_1 - x_i) - y_1 = 13(5-6) - 1 \bmod 17$$

$$y_3 = 3 \bmod 17$$



Finding $3P = 2P + P = (6,3) + (5,1)$

$P \neq Q$, use point addition

Point addition

$$s = \frac{y_2 - y_1}{x_2 - x_1} \bmod p = \frac{1-3}{5-6} \bmod 17$$

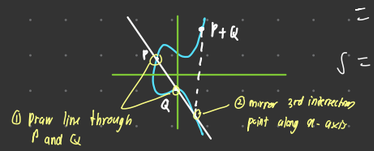
$$= \frac{-2}{-1} \bmod 17$$

$$s = 2 \bmod 17$$

$$x_i = s^2 - x_1 - x_2 = 2^2 - 6 - 5 \equiv 10 \bmod 17$$

$$y_i = s(x_1 - x_i) - y_1 = 2(6-10) - 3 \bmod 17$$

$$y_3 = -11 \equiv 6 \bmod 17$$



Finding all valid points on an elliptic curve

eg: given $y^2 = x^3 + x + 6 \bmod 11$

LHS RHS

$y^2 = x^3 + x + 6$

① Finding RHS

x	$x^3 + x + 6 \bmod 11$
0	6
1	8
2	(5)
3	(3)
4	8
5	(4)
6	8
7	(4)
8	(9)
9	7
10	(4)

② Finding LHS

y	$y^2 \bmod 11$
0	0
1	1
2	(4)
3	(9)
4	(5)
5	(3)
6	(3)
7	(5)
8	(9)
9	(4)
10	1

③ List down valid points (match ① and ②)

(2,4), (2,7)
(3,5), (3,6)
(5,2), (5,9)
(7,2), (7,9)
(8,3), (8,8)
(10,2), (10,9)

Chapter 9: Digital signature

↳ a mechanism to provide
- integrity
- authentication
- nonrepudiation: The sender of a message can't deny the creation of the message

General Principle:

Alice | Bob

Bob: generate K_{prB}, K_{pubB}

K_{pubB} →

← (x, s)

Bob: sign message, $s = sig_{K_{prB}}(x)$

Verify signature: $ver_{K_{pubB}}(x', s)$

if $x' = x$, true

RSA signature scheme

Alice | Bob

Key generation

1. Choose $p = 3,\ q = 11$
2. $n = p \cdot q = 33$
3. Choose $K_{pubB}: e = 3$
4. $K_{prB} = d = e^{-1} \equiv 7 \bmod 20$

$(n, e) = (33, 3)$ →

signature generation

$x = 4$

$s = x^d \bmod p = 4^7 \equiv 16 \bmod 33$

$(x, s) = (4, 16)$ →

Signature verification

$x' = s^e \bmod n = 16^3 \equiv 4 \bmod 33$

$\therefore$ since $x' = x = 16$, signature is valid

Elgamal signature scheme

Alice | Bob

Key generation:

1. Choose large prime, $p = 29$
2. Choose a primitive element of Z_p^*, $\alpha = 2$
3. Choose $d \in \{2, 3, \dots, p-2\}$, $d = 12$
4. Compute $\beta = \alpha^d \bmod p = 2^{12} \equiv 7 \bmod 29$

$(p, \alpha, \beta) = (29, 2, 7)$ →

Signature generation:

Choose $x = 26$

1. Choose ephemeral key K_E, where $K_E \in \{0, 1, 2, \dots, p-2\}$, $\gcd(K_E, p-1) = 1$ | $K_E = 5$
2. compute signature for message:

$r = \alpha^{K_E} \bmod p = 2^5 \equiv 3 \bmod 29$

$s = (x - d \cdot r) K_E^{-1} \bmod p-1 = (26 - 26 \cdot 3) \cdot 17 \equiv 26 \bmod 28$

$(x, (r, s)) = (26, (3, 26))$ →

Signature verification

1. $t \equiv \beta^r \cdot r^s \bmod p = 7^3 \cdot 3^{26} \equiv 22 \bmod 29$
2. $\alpha^x \bmod p = 2^{26} \equiv 22 \bmod 29$

$\therefore$ since $t = \alpha^x = 22$, signature is valid

Digital Signature Algorithm (DSA)

Alice | Bob

key generation

1. Generate a prime, $2^{1023} < p < 2^{1024}$, $p = 59$
2. Find a prime divisor q of $p-1$, $2^{159} < q < 2^{160}$, $q = 29$
3. Find α with $ord(\alpha) = q$, $\alpha = 3$ (α generates the subgroup with q elements)
4. Choose K_{prB}, $0 < d < q$, $d = 7$
5. $\beta = \alpha^d \bmod p = 3^7 \equiv 4 \bmod 59$

$(p, q, \alpha, \beta) = (59, 29, 3, 4)$ →

Signature generation

$h(x) = 26$

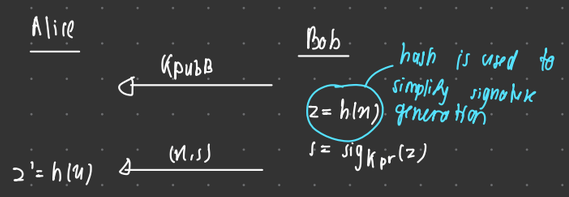
1. Choose $0 < K_E < q$, $K_E = 10$
2. $r \equiv (\alpha^{K_E} \bmod p) \bmod q = (3^{10} \bmod 59) = 20 \bmod 29$
3. $s \equiv (SHA(x) + d \cdot r) K_E^{-1} \bmod q = (26 + 7 \cdot 20) \cdot 3 \equiv 5 \bmod 29$

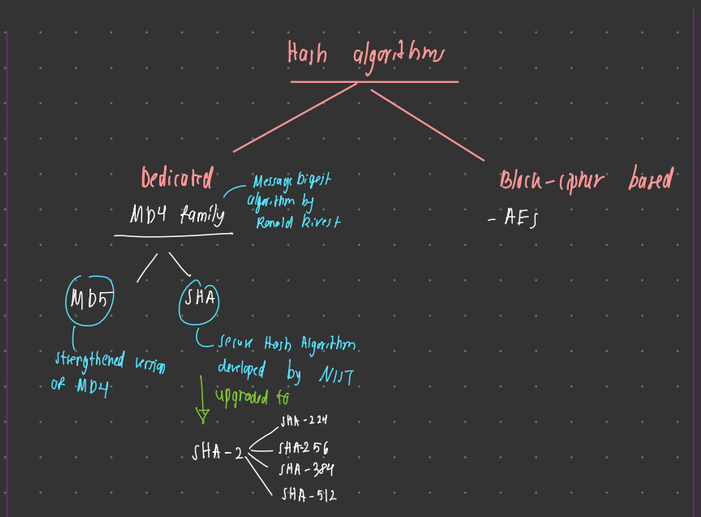
$(x, (r, s)) = (x, (20, 5))$ →

Signature Verification

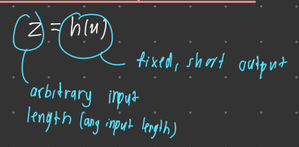
1. $w \equiv s^{-1} \bmod q = 5^{-1} \equiv 6 \bmod 29$
2. $u_1 \equiv w \cdot SHA(x) \bmod q = 6 \cdot 26 \equiv 11 \bmod 29$
3. $u_2 \equiv w \cdot r \bmod q = 6 \cdot 20 \equiv 4 \bmod 29$
4. $v \equiv (\alpha^{u_1} \cdot \beta^{u_2} \bmod p) \bmod q = (3^{11} \cdot 4^4 \bmod 59) \bmod 29 = 20$
5. $\begin{cases} \equiv r \bmod q & \rightarrow \text{valid signature} \\ \not\equiv r \bmod q & \rightarrow \text{invalid signature} \end{cases}$

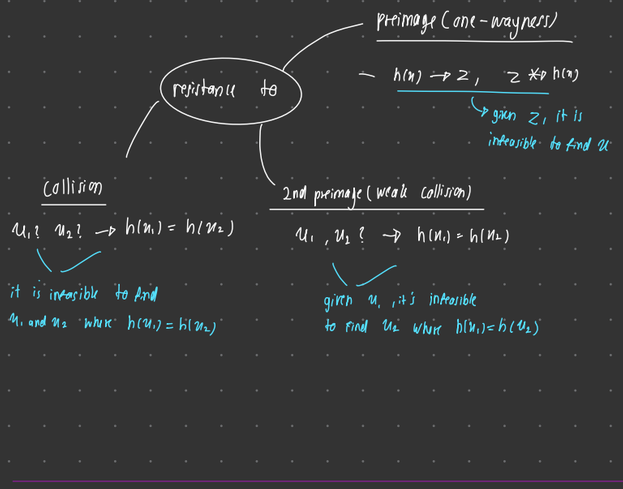
Chapter 10: Hash Function



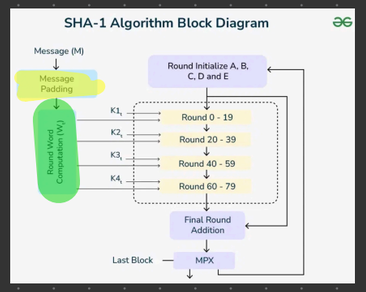


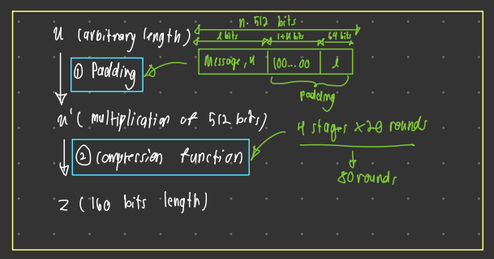
General requirment





SHA-1





$u = abc$, $h(u) = ?$

① Message padding

1 to indicate end of message and start of padding

01100001 01100010 01100011 | 1 00...0 | 0...011000

$l = 24$ = length of message

l represented in 64 bits

k number of Zeros,

to find how many no. of zeros to be added so its total length is multiplication of 512

length of u | 1 bit added before k | l represented in 64 bits

$k \equiv 512 - (l + 1 + 64) \bmod 512$ ← original formula

↓ simplified into

$k \equiv 448 - (l+1) \bmod 512$

② Message scheduling / Round word computation

① Divide into blocks of 512 bits $(u_1, u_2, \dots, u_i)$ with 16 words each block $(u_i^0, u_i^1, \dots, u_i^{15})$

$u_1^{(0)}$ ~ 1 word = 32 bits, $u_1^{(1)} \dots u_1^{(15)}$

$u_1 =$ 01100001 01100010 01100011 10000000 0...0

$u_2 =$

$u_i =$

since ours only produce 512 bits, we only have u_1

② For each blocks $(u_0 \dots u_i)$, extends from 16 words → 80 words $(W_0, W_1, \dots, W_{79})$

$$W_j = \begin{cases} u_i^{(j)} & 0 \le j \le 15 \\ (W_{j-16} \oplus W_{j-14} \oplus W_{j-8} \oplus W_{j-3}) \lll 1 & 16 \le j \le 79 \end{cases}$$

shift left 1 bit

for u_1,

$W_0 =$ 01100001 01100010 01100011 10000000

$W_1 = u_1^{(1)}$

$W_2 = u_1^{(2)}$

$\vdots$

$W_{15} = u_1^{(15)}$

$W_{16} = (W_{(16-16=0)} \oplus W_{(16-14=12)} \oplus W_{(16-8=8)} \oplus W_{(16-3=13)}) \lll 1$

$W_{17} = (W_{(17-16=1)} \oplus W_{(17-14=13)} \oplus W_{(17-8=9)} \oplus W_{(17-3=14)}) \lll 1$

$\vdots$

$W_{79} = (W_{(79-16=63)} \oplus W_{(79-14=65)} \oplus W_{(79-8=71)} \oplus W_{(79-3=76)}) \lll 1$

③ Hash computation

operation on each round

$A = (E + f_t(B,C,D) + (A) \lll 5 + W_j + K_t)$

$B = A$

$C = (B) \lll 30$

$D = C$

$E = D$

round 0

initial value = $h_{i-1} = h_0$

$A = H_0^{(0)} = 67452301$

$B = H_0^{(1)} = EFCDAB89$

$C = H_0^{(2)} = 98BADCFE$

$D = H_0^{(3)} = 10325476$

$E = H_0^{(4)} = C3D2E1F0$

$A = (H_0^{(4)} + f_t(H_0^{(1)}, H_0^{(2)}, H_0^{(3)}) + H_0^{(0)} \lll 5 + W_0 + K_t)$

$B = A = H_0^{(0)}$

$C = (B) \lll 30 = H_0^{(1)} \lll 30$

$D = C = H_0^{(2)}$

$E = D = H_0^{(3)}$

New value of A, B, C, D, E is passed to the next rounds

⋮

round 19

Stage t	Round j	Constant K_t	Function f_t
1	0…19	$K_1 =$ 5A827999	$f_1(B,C,D) = (B \wedge C) \vee (\bar{B} \wedge D)$
2	20…39	$K_2 =$ 6ED9EBA1	$f_2(B,C,D) = B \oplus C \oplus D$
3	40…59	$K_3 =$ 8F1BBCDC	$f_3(B,C,D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
4	60…79	$K_4 =$ CA62C1D6	$f_4(B,C,D) = B \oplus C \oplus D$

round 20 … round 39 ↓ pass new A B C D E

Stage t	Round j	Constant K_t	Function f_t
1	0…19	$K_1 =$ 5A827999	$f_1(B,C,D) = (B \wedge C) \vee (\bar{B} \wedge D)$
2	20…39	$K_2 =$ 6ED9EBA1	$f_2(B,C,D) = B \oplus C \oplus D$
3	40…59	$K_3 =$ 8F1BBCDC	$f_3(B,C,D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
4	60…79	$K_4 =$ CA62C1D6	$f_4(B,C,D) = B \oplus C \oplus D$

round 40 … round 59 ↓ pass new A B C D E

Stage t	Round j	Constant K_t	Function f_t
1	0…19	$K_1 =$ 5A827999	$f_1(B,C,D) = (B \wedge C) \vee (\bar{B} \wedge D)$
2	20…39	$K_2 =$ 6ED9EBA1	$f_2(B,C,D) = B \oplus C \oplus D$
3	40…59	$K_3 =$ 8F1BBCDC	$f_3(B,C,D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
4	60…79	$K_4 =$ CA62C1D6	$f_4(B,C,D) = B \oplus C \oplus D$

round 60 … round 79

Stage t	Round j	Constant K_t	Function f_t
1	0…19	$K_1 =$ 5A827999	$f_1(B,C,D) = (B \wedge C) \vee (\bar{B} \wedge D)$
2	20…39	$K_2 =$ 6ED9EBA1	$f_2(B,C,D) = B \oplus C \oplus D$
3	40…59	$K_3 =$ 8F1BBCDC	$f_3(B,C,D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
4	60…79	$K_4 =$ CA62C1D6	$f_4(B,C,D) = B \oplus C \oplus D$

↓

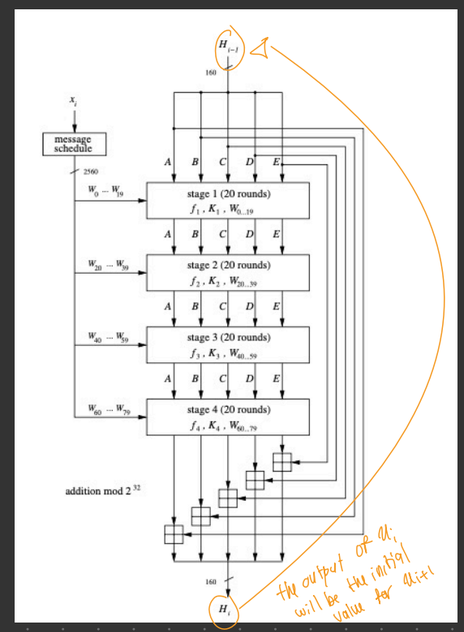
calculate $h_{i=1} = (h_{i-1=0} + ABCDE) \bmod 2^{32}$

use this as the initial value for u_2 ($h_{2-1} = h_1$) if exist

Stage t	Round j	Constant K_t	Function f_t
1	0…19	$K_1 = 5A827999$	$f_1(B,C,D) = (B \wedge C) \vee (\bar{B} \wedge D)$
2	20…39	$K_2 = 6ED9EBA1$	$f_2(B,C,D) = B \oplus C \oplus D$
3	40…59	$K_3 = 8F1BBCDC$	$f_3(B,C,D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
4	60…79	$K_4 = CA62C1D6$	$f_4(B,C,D) = B \oplus C \oplus D$

→ repeat for $u_2 \dots u_i$

result of this is the hash of u, $h(u) = z$

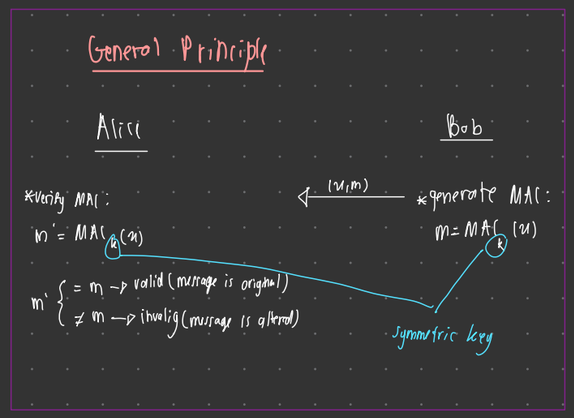


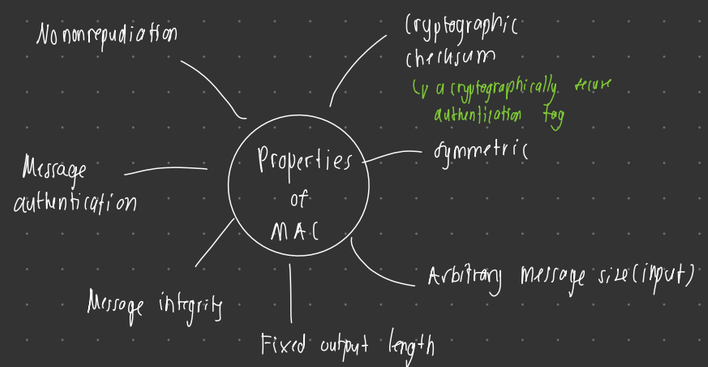
graphical representation of the 80 round compression function of SHA-1

Chapter 11: Message Authentication Code (MAC) aka Cryptographic Checksum

↳ a mechanism that append authentication tag to a message to provide → - integrity ✓
- authentication ✓
- nonrepudiation ✗ (since symmetric key is used)

General Principle

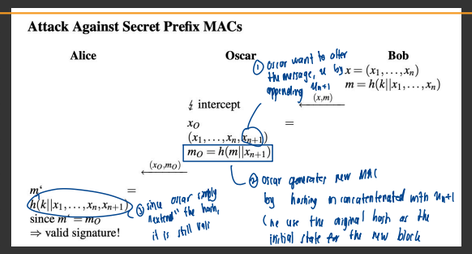




HMAC

Up using hash function to build a MAC

① Basic construction (weak)



can be attacked via

Bob

*generate MAC

$m = MAC_k(x)$

secret prefix, $m = h(k||x)$ — key concatenated with message

secret suffix, $m = h(x||k)$ — message concatenated with key

2 possible constructions

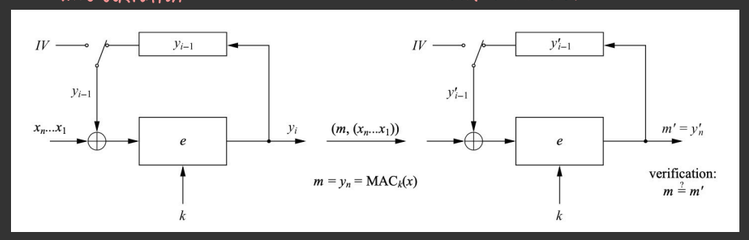
can be attacked by constructing a collision, where $h(x) = h(x_0)$

hence $h(x||k) = h(x_0||k)$

③ MAC from Block Ciphers: AES in Cipher Block Chaining (CBC) mode

MAC Generation

MAC Verification



Mac generation / verification

① Divide the message, x into blocks x_i, $i = 1, \ldots, n$

② Compute first iteration, $y_1 = e_k(x_1 \oplus IV)$ — use key; initial value is a public randomized value

③ For subsequent blocks, $y_i = e_k(x_i \oplus y_{i-1})$

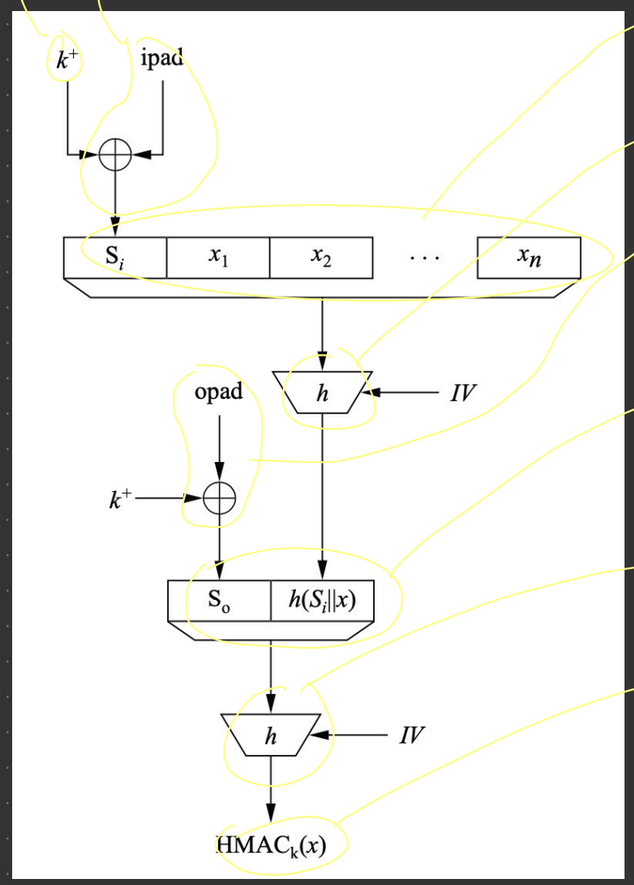
MAC is the output of the last round:

$m = MAC_k(x) = y_n = e_k(x_n \oplus y_{n-1})$

② HMAC prepared by Mihir, Ran, Hugo, $HMAC_k(x) = h[(k^+ \oplus opad)||h[(k^+ \oplus ipad)||x]]$

① expand key, $k^+ = \underbrace{000 \ldots 000}_{0s \text{ padding}} \underbrace{110110}_{k}$ — total length is b bits long; input block width of the hash function, eg: 512 for SHA1

② k^+ is XORed with ipad (00110110, 00110110, ..., 00110110) — sequence of this



③ The output of ②, S_i is concatenated with message $(x_1, x_2, \ldots, x_n)$

④ Output of ③ is hashed

⑤ k^+ is XORed with opad, (01011100, 01011100, ..., 01011100)

⑥ Output of ⑤, S_o is concatenated with output of ④

⑦ Output of ⑥ is hashed

$HMAC_k(x)$ is achieved

Chapter 12: Key Management

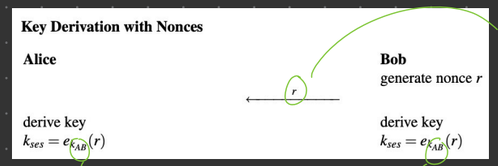
Key establishment

Is deals with establishing a shared key between 2 or more parties

- **Key transport**: One party generates and distributes a secret key
- **Key agreement**: Parties jointly generate a secret key

Key freshness

Is cryptographic key that is only valid for a limited time
- session key
- ephemeral key



new r value will be transported each time a new session key is to be derived

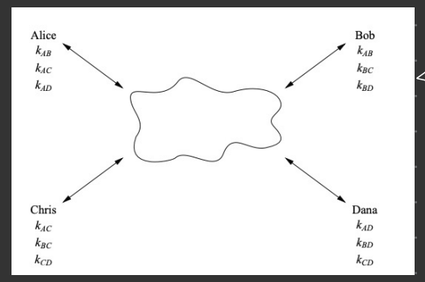
- the idea is to use an already established joint secret key to derive a session key
- k_{AB} is processed with r to derive a session key

Alternative ways to derive k_{ses}

1. $k_{ses} = HMAC_{k_{AB}}(r)$
2. $k_{ses} = e_{k_{AB}}((r+t))$
3. $k_{ses} = HMAC((r+t))$

a counter, will increment by a fixed amount

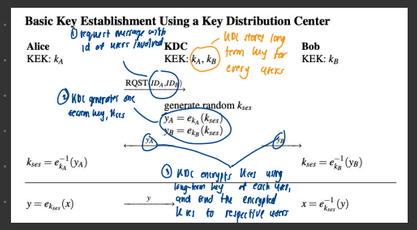
The N^2 key distribution problem



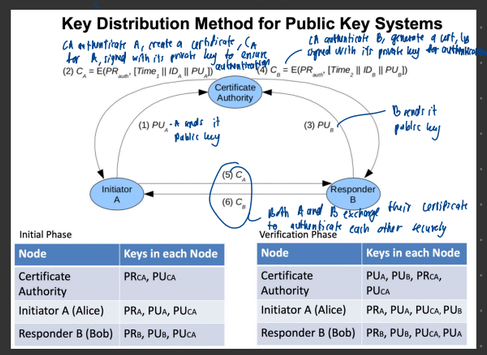
← if each user stores each other's public key,
- total keys in the system = $n(n-1) \approx n^2$ ← say $n = 750$ employees, $750 \times 749 \approx 561,750$ keys must be distributed
- key pairs = $n(n-1)/2$

To solve the issue, 2 methods can be employed for key establishment

Symmetric key distribution



real world implementation in Public key system in authentication process



Node	Keys in each Node
Certificate Authority	PR_{CA}, PU_{CA}
Initiator A (Alice)	PR_A, PU_A, PU_{CA}
Responder B (Bob)	PR_B, PU_B, PU_{CA}

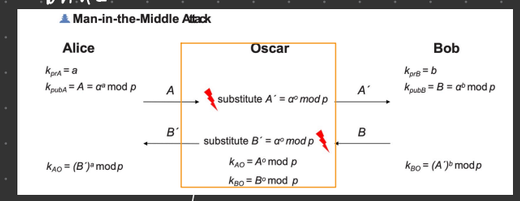
Node	Keys in each Node
Certificate Authority	PU_A, PU_B, PR_{CA}, PU_{CA}
Initiator A (Alice)	PR_A, PU_A, PU_{CA}, PU_B
Responder B (Bob)	PR_B, PU_B, PU_{CA}, PU_A

communication bottleneck

Weaknesses of KDC
- No perfect forward secrecy
- single point of failure

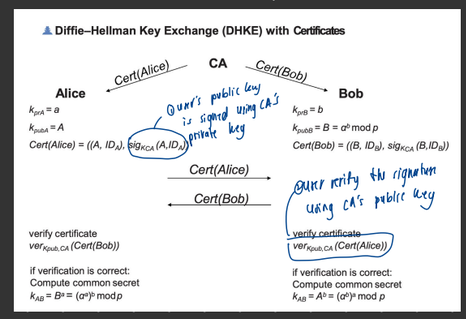
Asymmetric key distribution

DHKE:



- all public key cryptography is vulnerable to MITM attack
- because public key is not authenticated

can be solved by certificate authority (CA)



structure of X.509 certificate

Serial Number
Certificate Algorithm: - Algorithm - Parameters
Issuer
Period of Validity: - Not Before Date - Not After Date
Subject
Subject's Public Key: - Algorithm - Parameters - Public Key
Signature

Public-key infrastructure

↳ The entire system that is formed by CAs with necessary support mechanisms

some issues with PKIs
- User communication with user verified by different CAs
- Certification Revocation Lists (CRLs)

RSA :

Bob

① key generation :

1. p, q
2. $n = p \cdot q$
3. $\varnothing n = (p-1)(q-1)$
4. $k_{pub_B} = e$
5. $k_{pr_B} = e^{-1} \equiv d \bmod \varnothing n$

Alice

② encryption :

$y = x^e \bmod n$

③ Decryption

1. $x = y^d \bmod n$

DHKE :

① setup

α, p

Alice

② key generation

1. $k_{prA} = a$
2. $k_{pubA}, A = \alpha^a$

Bob

② key generation

1. $k_{prB} = b$
2. $k_{pubB}, B = \alpha^b$

Alice

③ shared key generation

1. $k_{AB} = B^a$

Bob

③ shared key generation

1. $k_{AB} = A^b$

Formulas

Elgamal

Alice

② key generation

1. $k_{prA} = i$
2. $k_{pubA} = k_E = \alpha^i$
3. $k_{AB} = k_m = \beta^i$

③ Encryption

1. $y = x \cdot k_m$

Bob

① key generation

1. α, p, $k_{prB} = d$
2. $k_{pubB}, \beta = \alpha^a$

④ calculate Masking key, k_M

1. $k_{AB} \cdot k_m = k_E^d$

⑤ Decryption

1. $x = y \cdot k_m^{-1}$

extended

Elgamal signature scheme

Alice

③ signature verification

1. $t = \beta^r \cdot r^s$
2. $\alpha^x \bmod p$

$t \begin{cases} = \alpha^x \bmod p & \text{- valid} \\ \neq \alpha^x \bmod p & \text{- invalid} \end{cases}$

Bob

① key generation

1. α, p, $k_{prB} = d$
2. $k_{pubB}, \beta = \alpha^a$

② signature generation

1. x, k_E
2. $r = \alpha^{k_E} \bmod p$
3. $s = (x - d \cdot r) \, k_E^{-1} \bmod p-1$

ECC

must satisfy $4a^3 + 27b^2 \bmod p \neq 0$

point doubling :

$s = \frac{3x_1^2 + a}{2y_1}$

point addition :

$s = \frac{y_2 - y_1}{x_2 - x_1}$

$x_3 = s^2 - x_1 - x_2$

$y_3 = s(x_1 - x_3) - y_1$

Digital signature algorithm (DSA)

Alice

③ signature verification

1. $w \equiv s^{-1} \bmod q$
2. $u_1 \equiv w \cdot h(x) \bmod q$
3. $u_2 \equiv w \cdot r \bmod q$
4. $v \equiv (\alpha^{u_1} \cdot \beta^{u_2} \bmod p) \bmod q$

$v \begin{cases} \equiv r \bmod q & \text{- valid} \\ \not\equiv r \bmod q & \text{- invalid} \end{cases}$

Bob

① key generation

1. p, q, κ, $k_{prB} = d$
2. $k_{pubB}, \beta = \alpha^d \bmod p$

② signature generation

1. $h(x)$, k_E
2. $r \equiv (\alpha^{k_E} \bmod p) \bmod q$
3. $s \equiv (h(x) + d \cdot r) \, k_E^{-1} \bmod q$

SHA-1

Stage t	Round j	Constant K_t	Function f_t
1	0…19	$K_1 = 5A827999$	$f_1(B,C,D) = (B \wedge C) \vee (\bar{B} \wedge D)$
2	20…39	$K_2 = 6ED9EBA1$	$f_2(B,C,D) = B \oplus C \oplus D$
3	40…59	$K_3 = 8F1BBCDC$	$f_3(B,C,D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
4	60…79	$K_4 = CA62C1D6$	$f_4(B,C,D) = B \oplus C \oplus D$

Operations on each round

$$A, B, C, D, E = (E + f_t(B,C,D) + (A)_{\lll 5} + W_j + K_t), A, (B)_{\lll 30}, C, D$$