Week 5

Topics: Advanced Encryption Standard (AES)

Introduction to publickey cryptosystem

Diffie-Hellman key Exchange.

Knapsack Cryptosystem

RSA cryptosystem.

The Advanced Encryption Standard (AES):

- Like DES, AES is a symmetric block cipher. However AES is quite different from DES in a number of ways.
- The block and key can be chosen independently from 128, 160, 192, 224, 256 bits and need not be the same.
- However, the AES standard states that the algorithm can only accept a block size of 128 bits and a choice of three keys - 128, 192, 256 bits.

• Depending on which version is used, the name of the standard is modified to AES-128, AES-192, AES-256 respectively.

• A number of AES parameters depend on the key length. For example, if the key size used is 128 then the number of ground is 10 whereas it is 12 and 14 for 192 and 256 bits respectively.

• At present the most common key size likely to be used is the 128-bit key.

• Rijndael was designed to have the following characteristics:

- Resistance against all known attacks
- Speed and code compactness on a wide range of platforms.
- Design simplicity.

• The input is a single 128-bit block both for decryption and encryption and is known as the in matrix (figure).

- This block is copied into a state array which is modified at each stage of the algorithm and then copied to an output matrix (figure).
- Both the plaintext and key are depicted as a 128-bit square matrix of bytes.
- This key is then expanded into an array of key schedule words (the w matrix).
- Ordering of bytes within the in matrix is by column. The same applies to the w matrix.

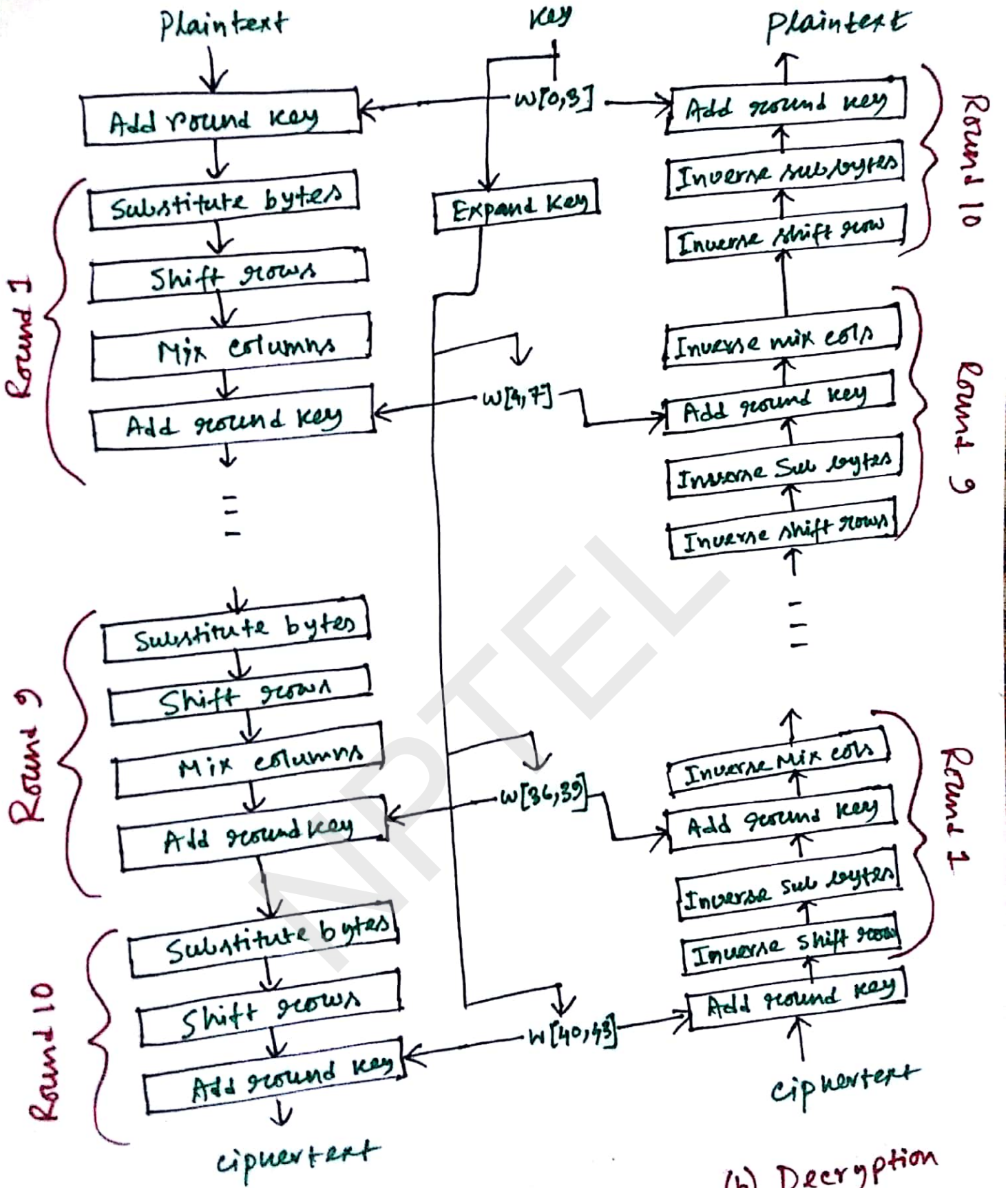
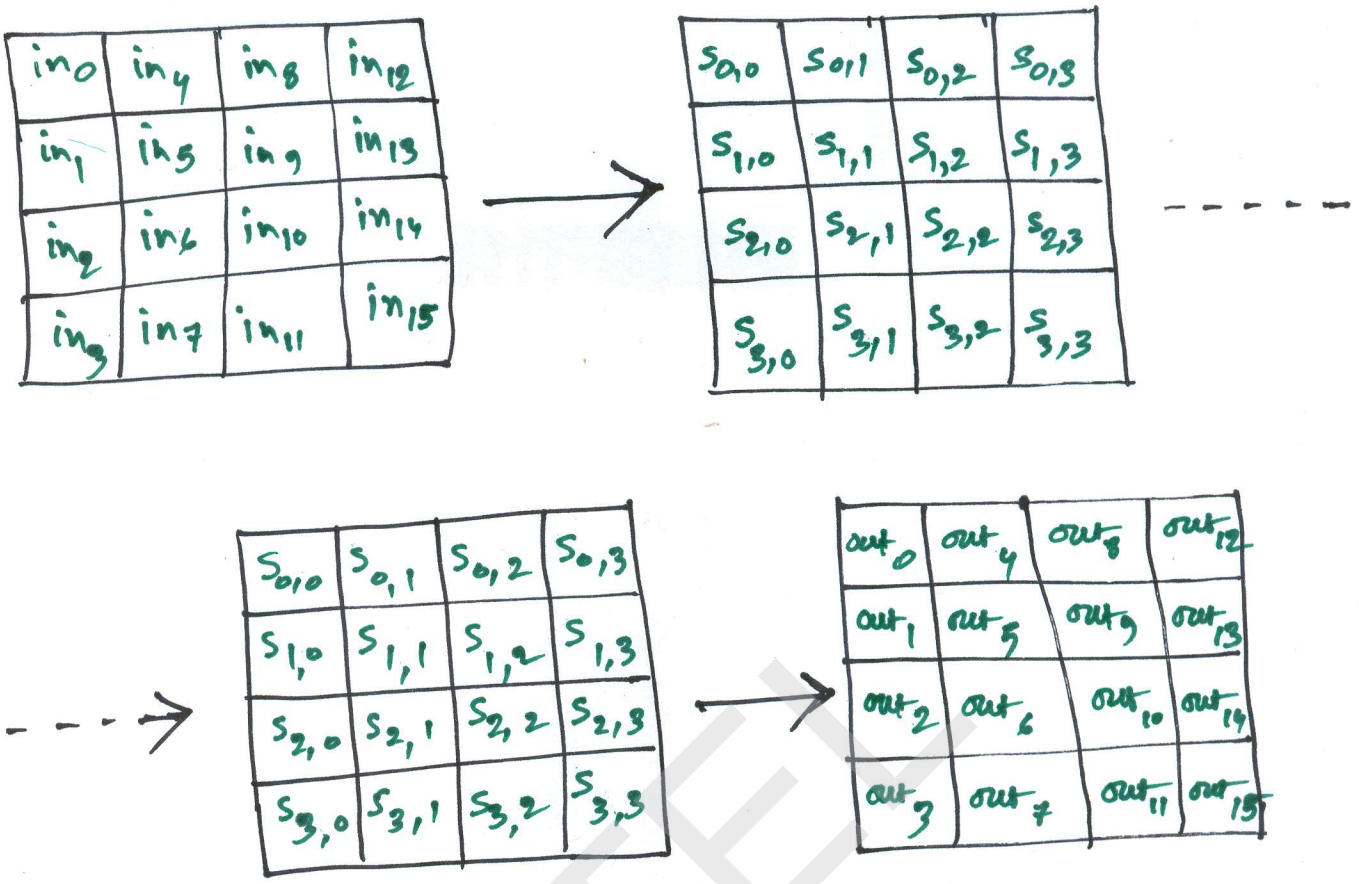
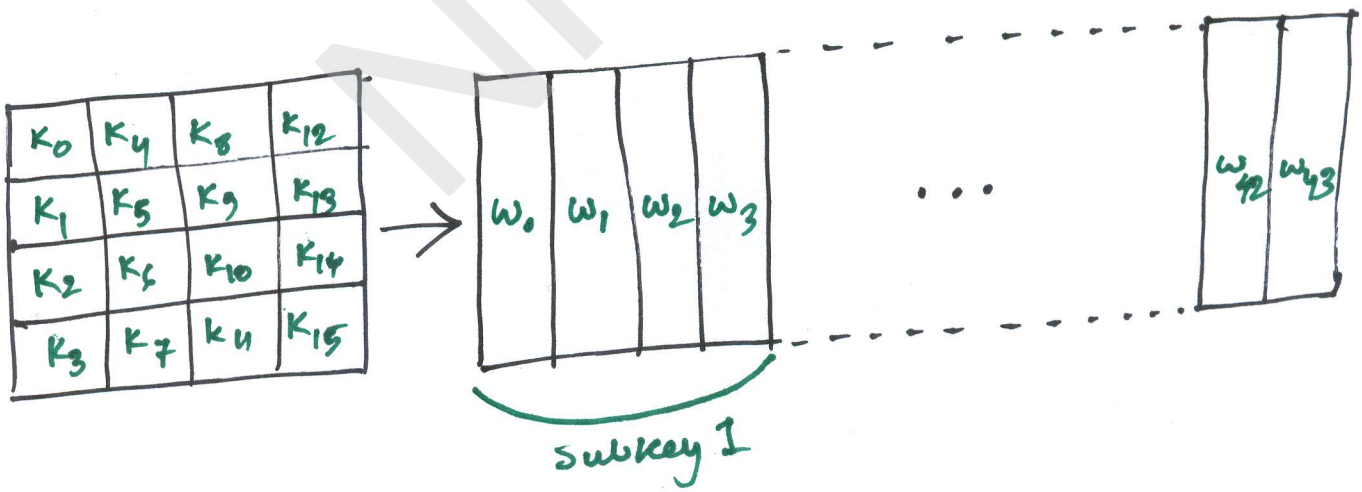


Figure 1 : Overall Structure of the AES Algorithm



(a) Input, state array and output



(b) Key and expand key

Figure 2: Data structures in the AES algorithm

High-level description of r-round AES

1. Given a plaintext X, initialize state to be X and perform an operation Add round key, which x-ors the round key with state.

2. For each of the first (r-1) rounds, perform a substitution operation called SubBytes on state using an S-box; perform a permutation ShiftRows on state; perform MixColumns on state; and perform AddRound key.

3. Perform SubBytes; perform ShiftRows; and perform Add Round key.

4. Define the ciphertext Y to be State.

- The plaintext X consists of 16 bytes.
- State is represented as a four by four array of bytes. (hexadecimal notation used to represent the content of a byte).

Inner Workings of a Round :

- The algorithm begins with an Add round key stage followed by 9 rounds of four stages and a tenth round of three stages.
- This applies for both encryption and decryption with the exception that each stage of a round the decryption algorithm is the inverse of its counterpart in the encryption algorithm.
- The four stages are as follows:
    1. Substitution bytes
    2. Shift rows
    3. Mix Columns
    4. Add Round key
- The tenth round simply leaves out the Mix Columns stage. The first nine rounds of the decryption algorithm consists of the following:
    1. Inverse Shift rows
    2. Inverse Substitution bytes
    3. Inverse Add Round key
    4. Inverse Mix Columns.

- Again the tenth round simply leaves out the Inverse Mix Columns stage.

Substitute Bytes (SubBytes):

- This stage is simply a table lookup using a 16×16 matrix of byte values called an S-box.
- This matrix consists of all the possible combinations of an 8-bit sequence ($2^8 = 16 \times 16 = 256$).
- State is effected by each round in the following way: each byte is mapped into new byte — the leftmost nibble of the byte is used to specify a particular row of the S-box and the rightmost nibble specifies a column.
- For example, the byte {95} selects row 9 column 5 which turns out to contain the value {2A}.
- This is then used to update the State matrix. (Figure 3).

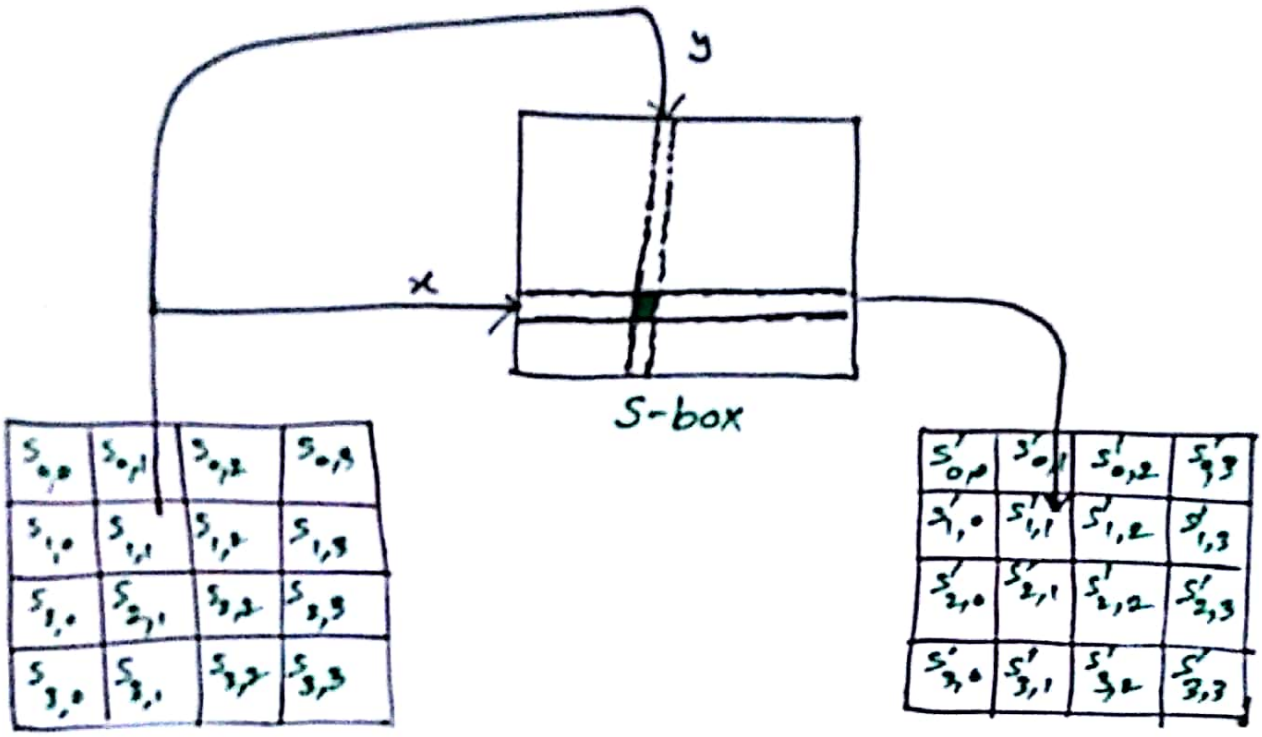


Figure 3: Substitute Bytes Stage of the AES algo.

- The Inverse substitute byte transformation (InvSubBytes) makes use of an inverse s-box.
- In this case what is desired is to select the value {2A} and get the value {95}.
- Table 4 shows the two s-boxes.
- The s-box is designed to be resistant to known cryptanalytic attacks.

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

(a) S-box

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	FA	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

(b) Inverse S-box

Figure 4: AES S-box both forward and inverse.

Algebraic formulation of AES S-box

- AES S-box involves operations in the finite field:

$$F_{28} = \mathbb{Z}_2[x] / (x^8 + x^4 + x^3 + x + 1)$$

- Let FieldInv denote the multiplicative inverse of a field element.

- Let BinaryToField convert a byte to a field element; and FieldToBinary perform the inverse conversion.

- The field element $\sum_{i=0}^{7} a_i x^i$ corresponds to the byte $a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0$,

where $a_i \in \mathbb{Z}_2 = \{0, 1\}$ for $0 \leq i \leq 7$.

- SubBytes $(a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0)$

$= (b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0)$

Algorithm: SubBytes $(a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0)$

external : FieldInv , BinaryToField , FieldToBinary

$z \leftarrow$ BinaryToField

if $z \neq 0$ then $z \leftarrow$ FieldInv (z)

$(a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0) \leftarrow$ FieldToBinary (z)

$(c_7 c_6 c_5 c_4 c_3 c_2 c_1 c_0) \leftarrow (0\,1\,1\,0\,0\,0\,1\,1)$

Comment: all subscripts are to be reduced modulo 8

for $i \leftarrow 0$ to 7

do $b_i \leftarrow (a_i + a_{i+4} + a_{i+5} + a_{i+6} + a_{i+7} + c_i) \bmod 2$

return $(b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0)$

Example:

- $\{53\} = 01010011$ in binary
- corresponding field element is
$$x^6 + x^4 + x + 1$$
- Multiplicative inverse in F_{28} is
$$x^7 + x^6 + x^3 + x$$
- binary notation $(a_7 a_6 a_5 a_4 a_3 a_2 a_1) = (11001010)$
- $b_0 = a_0 + a_4 + a_5 + a_6 + a_7 + c_0 \pmod 2 = 1$
$b_1 = a_1 + a_5 + a_6 + a_7 + a_0 + c_1 \pmod 2 = 0$
.....
- $(b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0) = (11101101) = \{ED\}$ (in Hexadecimal)

- We perform the following affine transformation:

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

(perform addition mod 2)

Algorithm: InvSubBytes

external : FieldInv, BinaryToField, FieldToBinary

$(d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0) \leftarrow (00000101)$

comment: all subscripts are to be reduced modulo 8

for $i \leftarrow 0$ to 7

do $b_i' \leftarrow (b_{i+2} + b_{i+5} + b_{i+7} + d_i) \bmod 2$

$z \leftarrow$ BinaryToField $(b_7' b_6' b_5' b_4' b_3' b_2' b_1' b_0')$

if $z \neq 0$ then $z \leftarrow$ FieldInv

$(a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0) \leftarrow$ FieldToBinary

return $(a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0)$

- We have the following affine transformation

$$\begin{pmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{pmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

(perform addition mod 2)

Shift Row Transformation:

- It works as follows:
  - The first row of state is not altered.
  - The second row is shifted 1 bytes to the left in a circular manner.
  - The third row is shifted 2 bytes to the left in a circular manner.
  - The fourth row is shifted 3 bytes to the left in a circular manner.

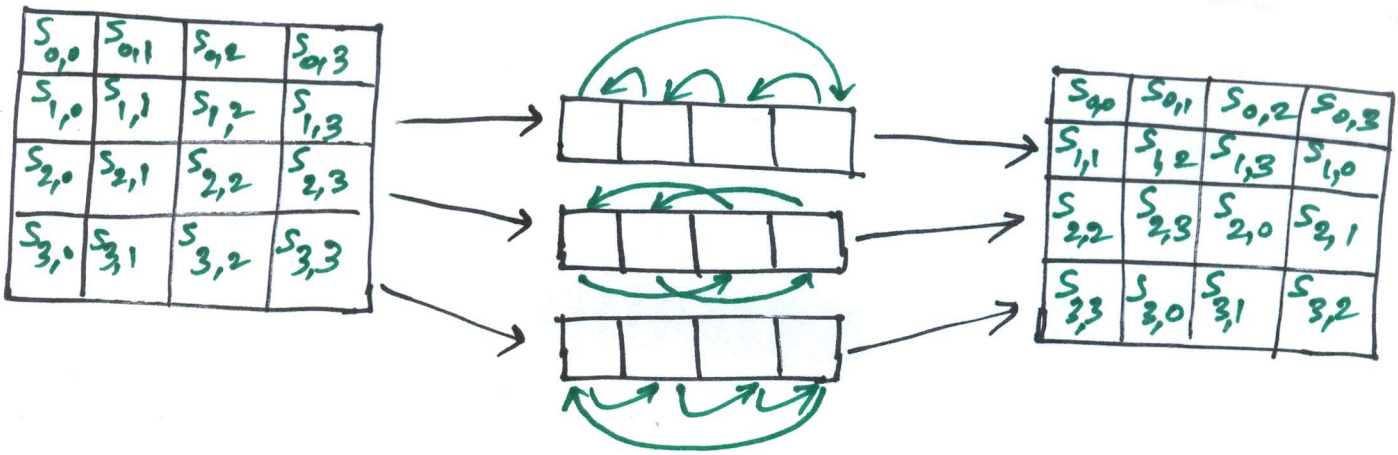


Figure 5: Shift rows stage.

- The Inverse Shift Rows transformation (InvShiftRows) performs these circular shifts in the opposite direction for each of the last three rows (the first row was unaltered to begin with).

- The transformation ensures that the four bytes of one column are spread out to four different columns.

(Remember that state is treated as an array of four byte columns)

Mix Column Transformation :

- The transformation can be determined by the following matrix multiplication on State

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

- In this case the individual additions and multiplications are performed in $GF(2^8)$

- The ~~Matrix~~ MixColumns transformation of a single column j $(0 \le j \le 3)$ of state can be expressed as :

$$s'_{0,j} = (2 \bullet s_{0,j}) \oplus (3 \bullet s_{1,j}) \oplus s_{2,j} \oplus s_{3,j}$$

$$s'_{1,j} = s_{0,j} \oplus (2 \bullet s_{1,j}) \oplus (3 \bullet s_{2,j}) \oplus s_{3,j} \quad (1)$$

$$s'_{2,j} = s_{0,j} \oplus s_{1,j} \oplus (2 \bullet s_{2,j}) \oplus (3 \bullet s_{3,j})$$

$$s'_{3,j} = (3 \bullet s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \bullet s_{3,j})$$

Algorithm : MixColumn (c)

external: FieldMult, BinaryToField, FieldToBinary

for $i \leftarrow 0$ to 3

do $t_i \leftarrow$ BinaryToField$(s_{i,c})$

$u_0 \leftarrow$ FieldMult$(x, t_0) \oplus$ FieldMult$(x+1, t_1) \oplus t_2 \oplus t_3$

$u_1 \leftarrow$ FieldMult$(x, t_1) \oplus$ FieldMult$(x+1, t_2) \oplus t_3 \oplus t_0$

$u_2 \leftarrow$ FieldMult$(x, t_2) \oplus$ FieldMult$(x+1, t_3) \oplus t_0 \oplus t_1$

$u_3 \leftarrow$ FieldMult$(x, t_3) \oplus$ FieldMult$(x+1, t_0) \oplus t_1 \oplus t_2$

for $i \leftarrow 0$ to 3

do $s_{i,c} \leftarrow$ FieldToBinary(u_i)

Example: let first column to be $s_{0,0} = \{87\}$

- $s_{1,0} = \{6E\}$, $s_{2,0} = \{46\}$, $s_{3,0} = \{A6\}$
- $s_{0,0} = \{87\}$ —mapped to→ $s'_{0,0} = \{47\}$, from (1) with $j=0$
- $(02 \bullet 87) \oplus (03 \bullet 6E) \oplus 46 \oplus A6 = 47$

$\{02\} = x$, $\{87\} = x^7 + x^2 + x + 1$

(Represent each Hexadecimal num by a poly.)

Multiply: $x(x^7 + x^2 + x + 1) = x^8 + x^3 + x^2 + x$

Reduce it by the irreducible poly. $m(x) = x^8 + x^4 + x^3 + x + 1$.

See $(x^8 + x^3 + x^2 + x) \pmod{m(x)}$

$= x^4 + x^2 + 1.$

$\equiv$ 0001 0101 (in binary)

$= \{15\}$

In this way,

$02 \cdot 87 = 0001\ 0101$

$03 \cdot 6E = 1011\ 0010$

$46 = 0100\ 0110$

$A6 = 1010\ 0110$

$\oplus$ 0100 0111 $= \{47\}$

- There is an easier way to do multiplication modulo $m(x)$.
- If we were multiplying by $\{02\}$ then all we have to do is a 1-bit left shift followed by a conditional bitwise XOR with (00011011) if the left most bit of the original value (prior to the shift) was 1.
- Multiplication by other number is repeated application of this method.
- Multiplication operation has been reduced to a shift and an XOR operation.

- The InvMixColumns is defined by the following matrix multiplication:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix} \quad \text{---(8)}$$

- This first matrix of equation (8) can be shown to be the inverse of the first matrix in equation (5).

- If we label these A and A^{-1} respectively and states as S, S' then,

$$A\,S = S'$$

so $$A^{-1}S' = A^{-1}(A\,S) = S.$$

Add Round Key Transformation :

- In this stage (Add Round Key) the 128 bits of state are bitwise XORed with the 128 bits of the round key.

- The operation is viewed as a columnwise operation between the 4 bytes of a State column and one word of the round key.

- The AES key expansion algorithm takes as input a 4-word key and produces a linear array of 44 words. Each round uses 4 of these words as shown in figure 2.

- Each word contains 32 bytes which means each subkey is 128-bits long. Figure 7 show pseudocode for generating the expanded key from the actual key.

Add Round Key Transformation :—

```
KeyExpansion ( byte key[16], word w[44] )
{
    word temp
    for ( i=0; i<4; i++)
        w[i] = ( key[4*i], key[4*i+1], key[4*i+2], key[4*i+3]);
    for (i=4; i<44; i++ )
    {
        temp = w[i-1]
        if (i mod 4 =0)  temp = subword (RotWord(temp)) ⊕
                                                    Rcon[i/4];
        w[i] = w[i-4] ⊕ temp;
    }
}
```

Figure 7 : Key expansion Pseudocode

- The key is copied into the first four words of the expanded key.
- The remainder of the expanded key is filled in four words at a time.
- Each added word w[i] depends on the immediately preceding word w[i-1] and the word four positions back w[i-4].
- In three out of four cases, a simple XOR is used.
- For a word whose position in the w array is a ~~simple~~ multiple of 4, a more complex function is used.

- **Algorithm** KeyExpansion (Key)

external : RotWord , SubWord

Rcon[1] ← 01000000 ; Rcon[3] ← 02000000
Rcon[3] ← 04000000 ; Rcon[4] ← 08000000
Rcon[5] ← 10000000 ; Rcon[6] ← 20000000
Rcon[7] ← 40000000 ; Rcon[8] ← 80000000
Rcon[9] ← 1B000000 ; Rcon[10] ← 36000000

for i ← 0 to 3
  do w[i] ← (key[4i], key[4i+1], key[4i+2], key[4i+3])
for i ← 4 to 43 do
  temp ← w[i-1]
  if i ≡ 0 (mod 4)
    then temp ← SubWord (RotWord (temp)) ⊕ Rcon[i/4]
  w[i] ← w[i-4] ⊕ temp
end do
return (w[0], w[1], . . . , w[43])

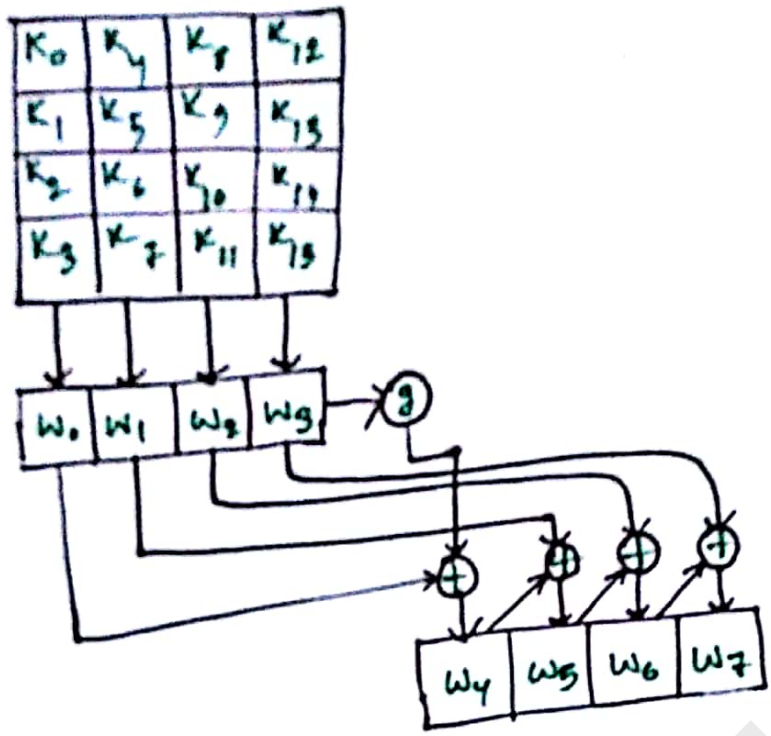


Figure 8: AES key expansion

- Figure 8 illustrates the generation of the first eight words of the expanded key using the symbol g to represent that complex function.
- The function g consists of the following subfunction:
  1. RotWord performs a one-byte circular shift in left direction on a word, i.e., $[b_0, b_1, b_2, b_3] \rightarrow [b_1, b_2, b_3, b_0]$

2. SubWord performs a byte substitution on each byte of its input word, using the s-box described earlier.

3. The result of step 1 and 2 is XORed with round constant, Rcon[j].

- Note: $Rcon[j] = (RC[j], 0, 0, 0)$
  with $RC[1] = 1$, $RC[j] = 2 \cdot RC[j-1]$
  (multiplication defined over the field $GF(2^8)$)

- The key expansion was designed to be resistant to known cryptanalytic attacks.

Public-key Cryptography :

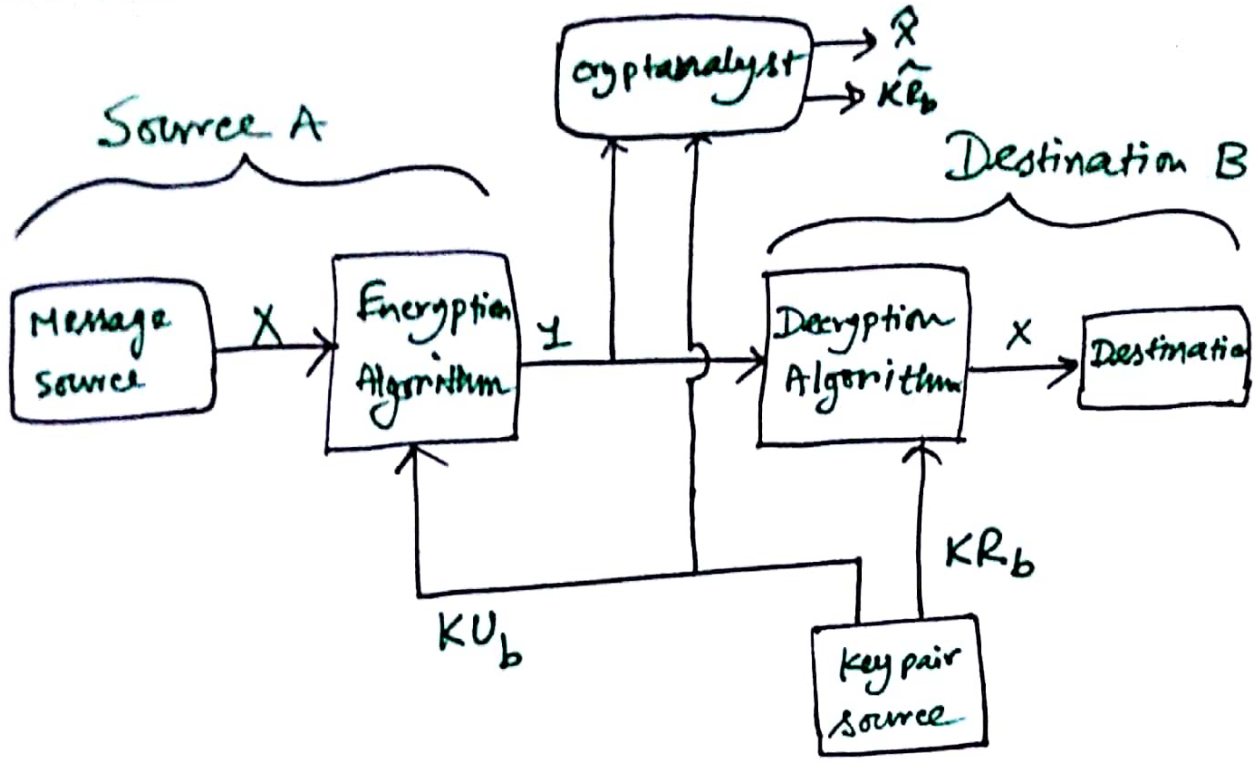
The process of PK cryptosystem are :

1. Each system generates a pair of keys.

2. Each system publishes its encryption key (public key) keeping its companion key private.

3. If A wishes to send a message to B it encrypts the message using B's public key.

4. When B receives the message, it decrypts the message using its private key. No one else can decrypt the message because only B knows its private key.



- B generates KU_b = public key
  KR_b = private key
- A : $Y = E_{KU_b}(X)$ and send it to B
- B : $X = D_{KR_b}(Y)$ and no one else gets X.

Figure 1: Public key Cryptography

Diffie Hellman Key Exchange :

- The first published P-K algorithm appeared in the paper by Diffie-Hellman that defined public key cryptography.

- The D-H key exchange consists of two publicly known numbers : a prime number p and an integer α that is a primitive root of p.

- It is based on Discrete Logarithm problem

  Instance : Given a large prime p and a generator of $\mathbb{Z}_p^*$ as g, an element $A \in \mathbb{Z}_p^*$.

  Question : Find the unique integer a, $0 \leq a \leq p-2$ such that $A = g^a \pmod p$.

- Suppose A and B wish to exchange a key.
- A selects a random integer $X_A < q$ and computes $Y_A = \alpha^{X_A} \pmod q$
- B selects a random integer $X_B < q$ and computes $Y_B = \alpha^{X_B} \pmod q$.
- Each side keeps the X values private and makes Y values available publicly to the other side.
- A computes the common key as

$$K = (Y_B)^{X_A} \pmod q$$

B computes the common key as

$$K = (Y_A)^{X_B} \pmod q$$

See,

$$\begin{aligned} K &= (Y_B)^{X_A} \pmod q \\ &= (\alpha^{X_B} \bmod q)^{X_A} \pmod q \\ &= (\alpha^{X_B})^{X_A} \pmod q \\ &= (\alpha^{X_A} \bmod q)^{X_B} \pmod q \\ &= (Y_A)^{X_B} \pmod q \end{aligned}$$

- Furthermore, because X_A and X_B are private an opponent is forced to take discrete logarithm to determine the key.

- For example, attacking the secret key of user B the opponent must compute :

$$X_B = \text{ind}_{\alpha,q}(Y_B)$$

where $\text{ind}_{\alpha,q}(Y_B)$ is the discrete logarithm or index, of Y_B for the base $\alpha \pmod q$.

- **Example:** Take $q = 353$, $\alpha = 3$

Note, α is a primitive root of prime q.

A: $X_A = 97$, $Y_A = \alpha^{X_A} \pmod q = 40$

B: $X_B = 233$, $Y_B = \alpha^{X_B} \pmod q = 248$

Then, $K = 248^{97} \pmod{353}$

$= 40^{233} \pmod{353}$

$= 160$

Subset Sum Problem :

- Merkle-Hellman knapsack cryptosystem is based on the subset sum problem:

Problem Instance: $I = (a_1, a_2, \ldots, a_n, S)$, where $a_1, \ldots, a_n$ and S are positive integers. The a_i's are called sizes and S is called the target sum.

Question: Is there a 0-1 vector $X = (x_1, x_2, \ldots, x_n)$ such that

$$\sum_{i=1}^{n} a_i x_i = S \; ?$$

- Subset sum problem is NP-complete problem, there is no polynomial-time algorithm that solves it.

- But certain special cases can be solved in polynomial time.

- A list of sizes $(a_1, \ldots, a_n)$ is said to be superincreasing if

$$a_j > \sum_{i=1}^{j-1} a_i$$

for $2 \leq j \leq n$. If the list of sizes is superincreasing, then Subset Sum search problem can be solved in time $O(n)$, and a solution X (if it exists) must be unique.

- **Algorithm 1 for the Subset Sum Problem:**

```
for i=n down to 1 do
    if S > a_i then
        S = S - a_i
        x_i = 1
    else
        x_i = 0
if S=0 then
    X = (x_1, ..., x_n) is the solution
else
    there is no solution.
```

- Send messages in blocks of n bits.

  - Cargo vector: $a = (a_1, a_2, \cdots, a_n)$ where a_i is an integer.

  - Plaintext: message block $x = (x_1, x_2, \cdots, x_n)$ where x_i is a binary digit.

  - Ciphertext: $S = a \cdot x = \sum_{i=1}^{n} (a_i x_i)$

Example: $a = (455, 341, 284, 132, 82, 56)$

$x = (x_1, x_2, x_3, x_4, x_5, x_6)$

$S = 821$

- a is public.
- Send message x as the ciphertext

$$S = a \cdot x = \sum_{i=1}^{n} a_i x_i$$

- Receiver must recover x from S and a.
- If $S = 3$, $a = (1, 3, 2, 5)$ then the pair has <u>two solutions</u> $x = (0, 1, 0, 0)$ and $x = (1, 0, 1, 0)$, which we don't want.
- Choose a as superincreasing sequence to get unique solution for x.

- Take $a' = (171, 197, 459, 1191, 2410)$ which is superincreasing.
- Suppose $S' = a' \cdot x' = 3798$
- Then apply Algorithm 1 to get $x' = 01011$
- Any one can decrypt the ciphertext, hence it is completely insecure.

The solution is as follows:

1. a', a superincreasing vector (private, chosen)
2. m, an interger larger than the sum of all a_i's (private, chosen)
3. w, an integer relatively prime to m (private, chosen)
4. w^{-1}, the inverse of w, modulo m (private, calculated)
5. $a = wa' \pmod m$ (public, calculated)

- private key $= (w^{-1}, m, a')$

public key $= a$

- B wishes to send message x to A.
- B calculate $S = a \cdot x$
- The determination of x given S and a is difficult as a is not superincreasing.
- A is able to decrypt easily.

Defining, $S' \equiv w^{-1} S \pmod m$

$$S = a \cdot x = w\, a' \cdot x$$

$$S' \equiv w^{-1} S \pmod m$$

$$\equiv w^{-1} w\, a' \cdot x \pmod m$$

$$= a' \cdot x$$

Now A runs Algorithm 1 with the instance $(a_1', \ldots, a_n', S')$ and obtain x.

(Since a' is superincreasing)

Merkle-Hellman Knapsack Cryptosystem

- Let $a' = (a'_1, \ldots, a'_n)$ be a superincreasing list of integers.
- Let m and w be two integers such that $m > \sum_{i=1}^{n} a'_i$ and $\gcd(w, m) = 1$.
- For $1 \leq i \leq n$, define $a_i = w\, a'_i \pmod m$ and denote $a = (a_1, \ldots, a_n)$
- Let $P = \{0,1\}^n$, $C = \{0, \ldots, n(m-1)\}$ and $K = \{(a', m, w, a)\}$. Here a is public and a', m, w are secret.
- For $K = (a', m, w, a)$ we define the encryption function:

$$e_K(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i a_i$$

- For a ciphertext $S \in C$, i.e., $0 \leq S \leq n(m-1)$ define $S' = w^{-1} S \pmod m$ and solve subset sum problem $(a'_1, \ldots, a'_n, S')$ obtaining $d_K(S) = (x_1, \ldots, x_n)$

Example:

- Let $a' = (2, 5, 9, 21, 45, 103, 215, 450, 946)$ is secret superincreasing sequence.
- $m = 2003$, $w = 1289$. Then,

$$a = (575, 436, 1586, 1030, 1921, 569, 721, 1183, 1570)$$

- B sends message $x = (1, 0, 1, 1, 0, 0, 1, 1, 1)$ computes $S = 575 + 1586 + 1030 + 721 + 1183 + 1570 = 6665$
- A receives S. Computes $S' = w^{-1} S \pmod m = 1643$
- A solves the instance $I = (a', S')$ of the subset sum problem using Algorithm 1 and get the plaintext as $(1, 0, 1, 1, 0, 0, 1, 1, 1)$

RSA (Ron Rivest, Adi Shamir, Len Adleman):

- For some plaintext $M < n$ and ciphertext $C < n$ we have:

$$C = M^e \pmod n$$
$$M = C^d \pmod n = (M^e)^d \pmod n$$
$$M = M^{ed} \pmod n$$

- Euler's theorem states that:

$$a^{\phi(m)} \equiv 1 \pmod m \quad \text{where } \gcd(a,m)=1$$

- A corollary to this theorem is: given two prime numbers p and q and integers $n = pq$ and m, with $0 < m < n$, the following relationship holds:

$$m^{\phi(n)+1} \equiv m^{(p-1)(q-1)+1} \equiv m \pmod n \quad \text{—(1)}$$

If $\gcd(m,n)=1$ then (1) directly follows from Euler's theorem.

· Let $\gcd(m,n) \neq 1$.

$(n = pq \;,\; \gcd(m,n) = 1)$

$\equiv$ (m is not a multiple of p) AND

(m is not a multiple of q)

Therefore $\gcd(m,n) \neq 1 \Rightarrow$

(m is a multiple of p) OR

(m is a multiple of q)

Consider, m is a multiple of p

$\Rightarrow m = cp$ for some positive integer c.

In this case $\gcd(m,q) > 1$

as m is a multiple of $q \Rightarrow m > pq$

But $pq = n$. So, $m > n$, is not true.

Therefore $m = cp$ & $\gcd(m,q) = 1$

$\gcd(m,q) = 1 \Rightarrow$ by Euler's theorem

$$m^{\phi(q)} \equiv 1 \pmod{q}$$

$$\Rightarrow \left(m^{\phi(q)}\right)^{\phi(p)} \equiv 1 \pmod{q}$$

$$\Rightarrow m^{\phi(n)} \equiv 1 \pmod{q}$$

Therefore, $q \mid (m^{\phi(n)} - 1)$

$\Rightarrow m^{\phi(n)} = 1 + kq$ for some integer k.

Now $m = cp$ so,

$$m^{\phi(n)+1} = m + kmq$$
$$= m + kc(pq)$$
$$\Rightarrow m^{\phi(n)+1} \equiv m \pmod{pq}$$
$$\equiv m \pmod{n}.$$

Therefore, $m^{\phi(n)+1} \equiv m \pmod{n}$

Similar line of reasoning is used when m is a multiple of q.

- An alternative form of this corollary is directly relevant to RSA:

$$m^{k\phi(n)+1} \equiv (m^{\phi(n)})^k \times m \pmod{n}$$
$$\equiv 1^k \times m \pmod{n}$$
$$\equiv m \pmod{n}$$

- This can be achieved with

$ed \equiv 1 \pmod{\phi(n)}$ where $ed = K\phi(n)+1$

i.e., $d \equiv e^{-1} \pmod{\phi(n)}$.

Key Generation

- Select p, q — p, q are both primes
- Calculate $n = p \times q$
- calculate $\phi(n) = (p-1) \times (q-1)$
- Select integer e, where $\gcd(\phi(n), e) = 1$; $1 < e < \phi(n)$
- calculate d, where $d = e^{-1} \pmod{\phi(n)}$
- Public key, $KU = \{e, n\}$
- Private key, $KR = \{d, p, q\}$

Encryption

- plaintext $= M \ (< n)$
- ciphertext $= C = M^e \pmod{n}$

Decryption

- Ciphertext $= C$
- plaintext $= M = C^d \pmod{n}$

The RSA algorithm

- **Example:**

  - Select $p = 7$, $q = 17$
  - Calculate $n = pq = 119$
  - Calculate $\phi(n) = (p-1)(q-1) = 96$
  - Select $e = 5$, so that $\gcd(e, \phi(n)) = 1$
  - Determine d, so that $ed \equiv 1 \pmod{\phi(n)}$

    Use Extended version of Euclid's algo. to get

    $77 \times 5 = 385 = 4 \times 96 + 1$

    $\Rightarrow d = 77$

  - $KU = \{5, 119\}$, $KR = \{77\}$.
  - Let plaintext $M = 19$

    $C = M^e \pmod{n} = 19^5 \pmod{119}$

    $\Rightarrow$ ciphertext $C = 66$.

  - To obtain plaintext using C and KR compute,

    $C^{77} = 66^{77} \equiv 19 \pmod{119}$.

    $M = 19 = C^d \pmod{n}$.