BLOWFISH Algorithm.

Notebook
Date: / /
Page:

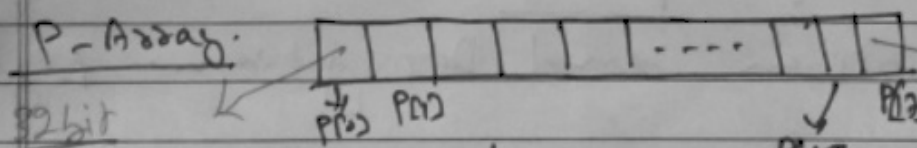
→ Symmetric key Algorithm
→ Block cipher (64 bit) Algorithm
→ Designed by Bruce Schneier in 1993
→ It is an alternative to DES
→ It follows feistel structure. (like DES)
→ Fast than DES
→ Compact (use less memory)
→ simple — (XOR, add operation)
→ Secure — variable length of key.
(Not fixe size)

→ Block size → 64 bit (plaintext size)
→ Key size. → variable (32 to 448) bits.
→ No of subKeys → 18 (P-Array)
$P_0, P_1, P_2 \ldots P_{17}$
No of rounds of feistel function. → 16
No of substitution boxes (S-box) → 4
each S-box having 256 entries and each entry having 32 bit.

Generation of SubKeys (18)
$P_0, P_1 \ldots P_{17}$ (in P-Array)

encryption
Decryption
use

P-Array:



32 bit

P-Array element length is 32 bit

$P[0] = H_1 H_2 \ldots H_8$ ← 8 Hexadecimal digit
8×4 = 32 bit, 1 Hexadecimal = 4

for example
P[0] = AB9832AC

Teacher's Signature.....

Each subKey of P-Array is chanted with respect to user Key.

for Example we have Key length is 448

P[0] = P[0] (XOR) 1st 32 bits of Key (1,32)

P[1] = P[1] (XOR) 2nd 32 bits of Key (33, 64)

⋮

P[13] = P[13] (XOR) 14th 32 bits of Key (416, 448)

P[14] = P[14] (XOR) 1st 32 bits of Key (1, 32)

⋮

so on

If we have Key length 448
then No of set of 32 bits = $\frac{448}{32} = 14$,
if size of Parray is fix that is P[0] - P[17] = 18
if size of Key is 32 bits then All
P-Array element XOR with same 32 bits.

Now P-Array is final and used for encryption and Decryption Algorithm.

Teacher's Signature

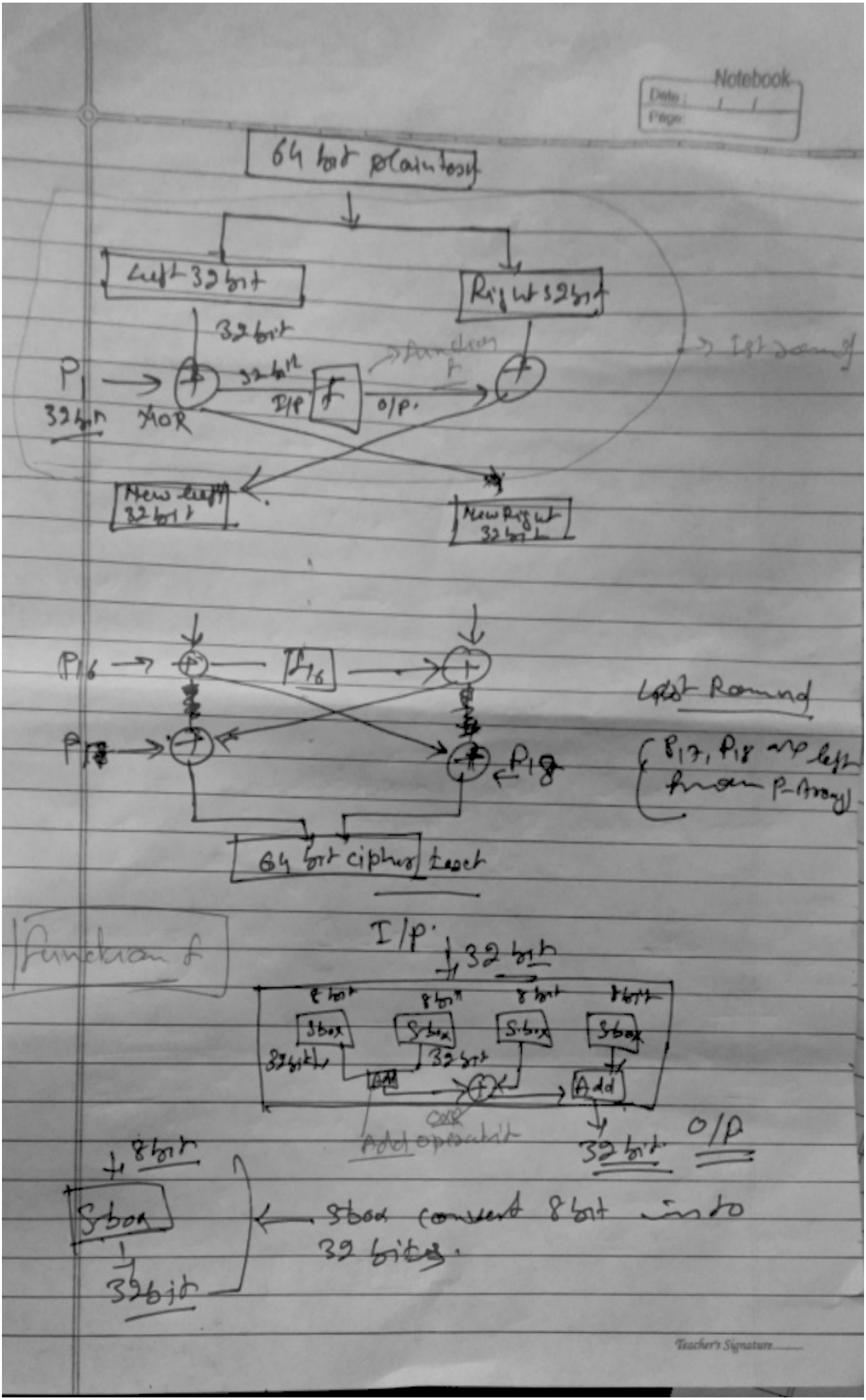
Each subKey of P-Array is chanted with respect to user Key.
for Example we have Key length is 448

P[0] = P[0] ⊕ (xor) 1st 32 bits of Key (1,32)
P[1] = P[1] ⊕ 2nd 32 bits of Key (33, 64)
⋮
P[13] = P[14] ⊕ 14th 32 bits of Key (415, 448)
P[14] = P[15] ⊕ 1st 32 bits of Key (1,32)

⋮ so on

If we have Key Length 448
then No of set of 32 bits = $\frac{448}{32} = 14$.
if no of Parray is fixed that is P[0] - P[17] = 18
if size of Key is 32 bits then all
P-Array element XOR with same 32 bits.

Now P-Array is final and used for encryption and Decryption Algorithm.


64 bit plaintext
Left 32 bit
Right 32 bit
32 bit
P1
32 bit
XOR
32 bit
I/P
Function f
O/P.
New Left 32 bit
New Right 32 bit
P16
P17
P18
Last Round
{P17, P18 are left from P-Array}
64 bit ciphertext
function f
I/P
32 bit
8 bit
8 bit
8 bit
8 bit
Sbox
Sbox
Sbox
Sbox
32 bit
Add
Add operation
32 bit
O/P
8 bit
Sbox
32 bit
← Sbox convert 8 bit into 32 bits.

RC5 Algorithm

→ is a block cipher notable for its simplicity.
→ Symmetric block cipher.
→ Designed by Ronald Rivest in 1994
→ RC stands for Rivest cipher or Ron's code
→ It is quite fast as it uses only primitive computer operation.
→ It allows a variable number of rounds and variable bit size key to add flexibility

Features

→ Suitable for H/W and S/W
→ fast
→ Adaptable to processors of different word lengths
→ variable number of rounds
→ variable length key
→ Simple
→ Low memory usage
→ High Security
→ Emphasis of data-dependent rotations.

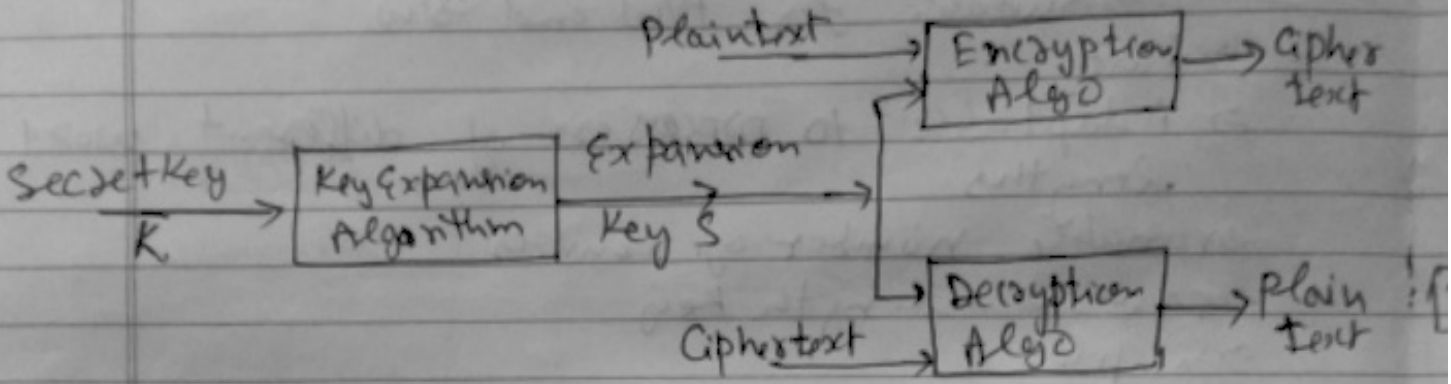
word size : w (16, 32, 64)
Number of rounds : r (0, 1, ... 255)
Number of bytes in Key K : b (0, 1, ... 255)
RC5 Algorithm notation : RC5-w/r/b
Example - RC5-32/16/7 ← 7 byte (56 bit) Key
↙ ↘
two 32-bit word) input/output & output
16 rounds

RC5-w/r/b
w - word size in bits
r - number of rounds
b - key size in bytes

RC5 Algorithm

→ three components of RC5
1. → Key Expansion Algorithm
2. → Encryption Algorithm
3. → Decryption Algorithm

→ fast Symmetric block cipher
  → same key for encryption and decryption
  → plain text and ciphertext are fixed length bit sequence (blocks)



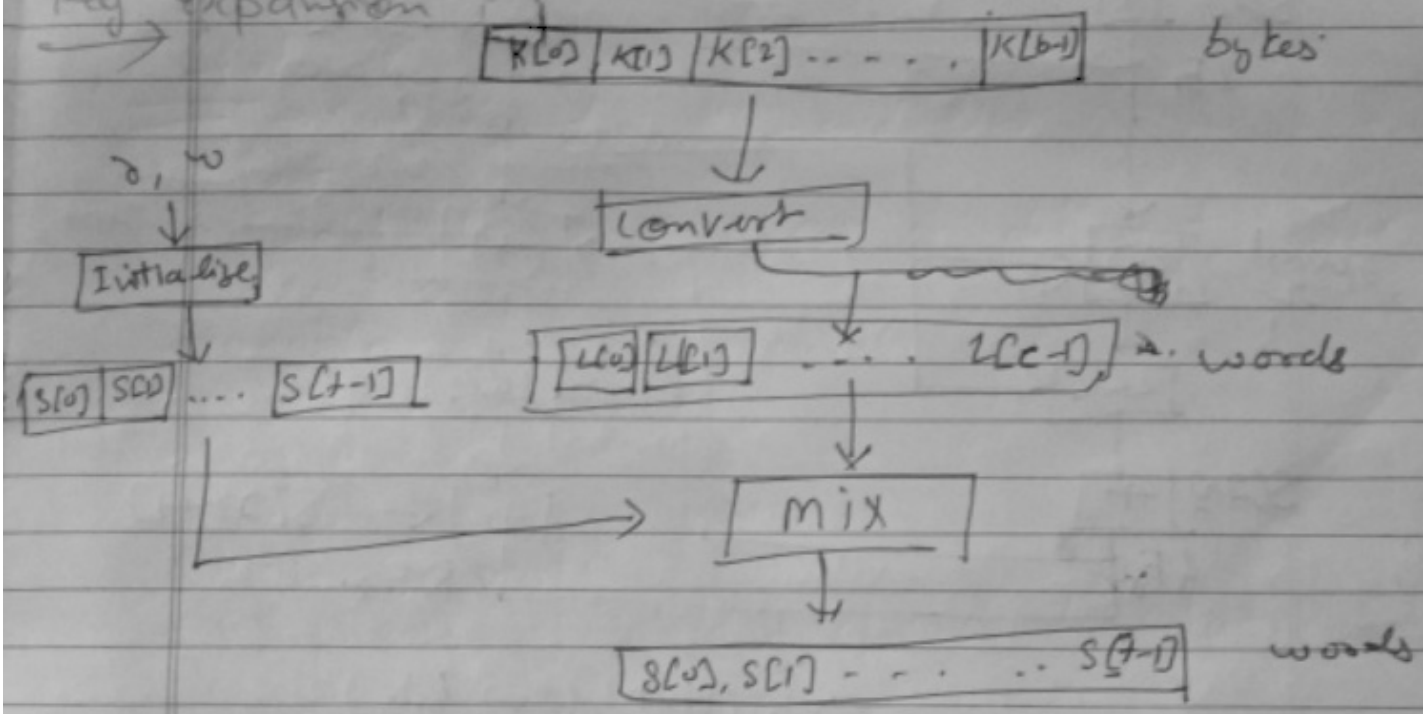
Key Expansion Algorithm →

→ RC5 performs a complex set of operations on the secret key to produce a total of t subkeys.
→ Two subkeys are used in each round and two subkeys are used on an additional operation that is not part of any round
  so $t = 2r + 2$
→ Each subkey is one word (w bits) in length.
→ In RC5, the plain text message is divided into two blocks A and B each of 32 bits.
→ Then two subkeys are generated S[0] and S[1]
→ These two subkeys are added into A and B respectively

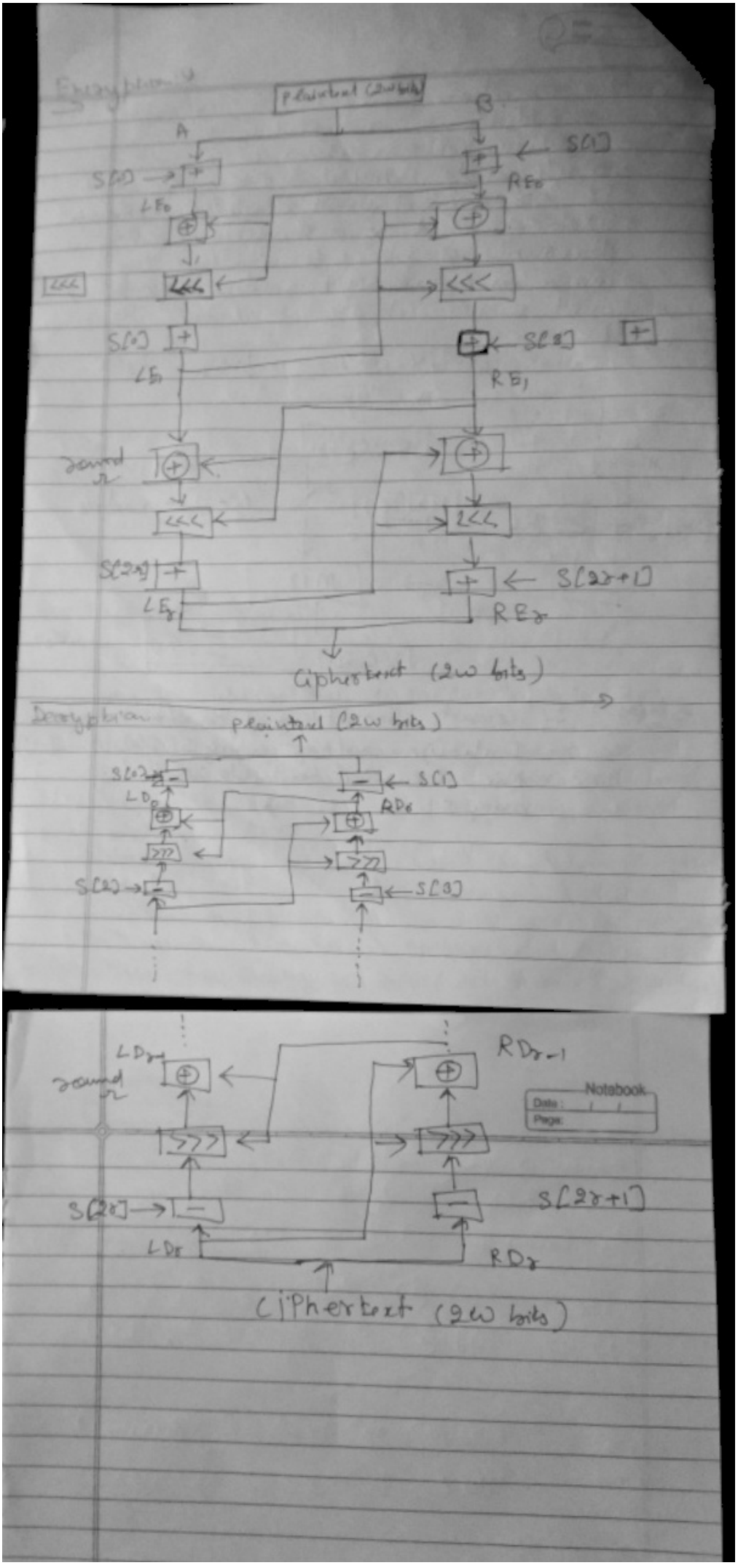
Primitive operations →

⇒ → Addition → denoted by +
→ Bitwise XOR → denoted by ⊕
→ Left rotation :- This is the cyclic left rotation of words, denoted by ~~<<<y~~ X<<<Y, where x is the word and y is the number of bits to be shifted. The inverse is cyclic right rotation X >>>Y

Key Expansion →



Steps :→ 1. Convert Secret Key bytes ~~to~~ words
2. Initialize subKey array S(S[0]S[1] ... S[t-1])
3. Mix the secret Key into sub key array S


Encryption
plaintext (2w bits)
A
B
S[0]
S[1]
LE0
RE0
<<<
S[2]
S[3]
LE1
RE1
round
S[2r]
S[2r+1]
LEr
REr
ciphertext (2w bits)
Decryption
plaintext (2w bits)
LD0
RD0
>>>
LDr
RDr
RDr-1
S[2r]
S[2r+1]
ciphertext (2w bits)