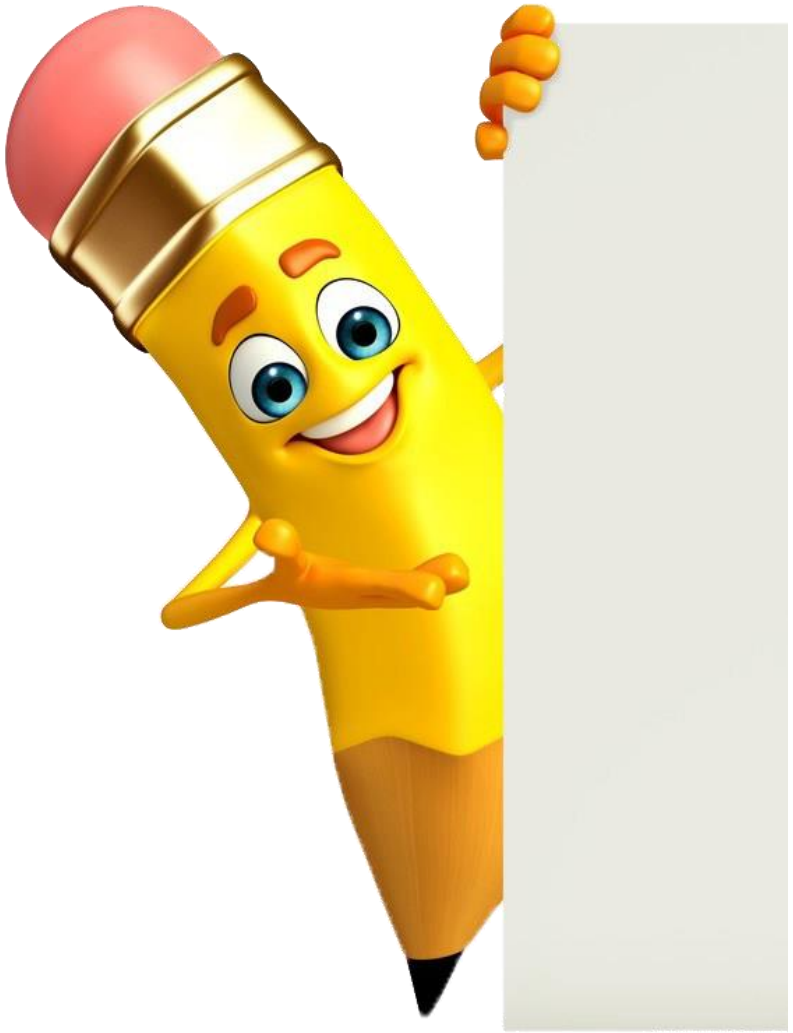


CSCI361-Cryptography and Secure Applications

K-bit OFB Encryption

Block:	0
Plaintext (Hex)	12345678
Plaintext (Long):	305419896
Plaintext (Binary):	0001 0010 0011 0100 0101 0110 0111 1000
kBit:	7
cipherTextBlock:	5
Key[]:	{0xab1a16be, 0xc4163a89, 0x87e5b018, 0x65ed8705}
IV	11111111111111111111111111111111111111111111111111111111111
Output:	1011110 1110000 0101011 1111001 0000001 1010000 1100000 0000110 0100001 0
plaintext:	0000001 0010001 1010001 0101100 1111000
Output^plaintext Kbit	1011111
cipherText:	1011111

Block:	1
IV:	11111111111111111111111111111111111111111111111111111110000000
Output-kBit:	00000000000000000000000000000000000000000000000000000001011110
IV:	11111111111111111111111111111111111111111111111111111111011110
Key[]:	{0xab1a16be, 0xc4163a89, 0x87e5b018, 0x65ed8705}
IV	11111111111111111111111111111111111111111111111111111111011110
Output:	1001100 0000100 0001100 0000101 1001101 1011000 0001111 1111111 0000111 1
plaintext:	0010001 1010001 0101100 1111000 0000000
Output^plaintext Kbit	1011101
cipherText:	1011111 1011101

Block:	2
IV:	11111111111111111111111111111111111111111111111111111011 1100000000
Output-kBit:	00000000000000000000000000000000000000000000000000000000 0001001100
IV:	11111111111111111111111111111111111111111111111111111011 1101001100
Key[]:	{0xab1a16be, 0xc4163a89, 0x87e5b018, 0x65ed8705}
IV	11111111111111111111111111111111111111111111111111111011 1101001100
Output:	1110100 0110111 1010010 1001001 0100101 0111001 0001101 0011110 1110101 1
plaintext:	1010001 0101100 1111000 0000000 0000000
Output^plaintext Kbit	0100101
cipherText:	1011111 1011101 0100101

Block:	3
IV:	11111111111111111111111111111111111111111110111101001 1000000000
Output-kBit:	00000000000000000000000000000000000000000000000000 0001110100
IV:	11111111111111111111111111111111111111111110111101001 1001110100
Key[]:	{0xab1a16be, 0xc4163a89, 0x87e5b018, 0x65ed8705}
IV	11111111111111111111111111111111111111111110111101001 1001110100
Output:	0000111 0101001 0011010 0100111 0101100 0110010 0100000 1010001 1100110 1
plaintext:	0101100 1111000 0000000 0000000 0000000
Output^plaintext Kbit	0101011
cipherText:	1011111 1011101 0100101 0101011

Block:	4
IV:	11111111111111111111111111111111111101111010011001101000000000
Output-kBit:	00000000000000000000000000000000000000000000000000000000111
IV:	11111111111111111111111111111111111101111010011001101000000111
Key[]:	{0xab1a16be, 0xc4163a89, 0x87e5b018, 0x65ed8705}
IV	11111111111111111111111111111111111101111010011001101000000111
Output:	0001101 1110101 0101010 1000000 1010111 1001110 0110101 0110000 0010001 1
plaintext:	1111000 0000000 0000000 0000000 0000000
Output^plaintext Kbit	1110101
cipherText (Bin):	1011111 1011101 0100101 0101011 1110101
cipherText(Hex):	5fba955f5

Block:	0
Plaintext (Hex)	12345678
Plaintext (Long):	305419896
Plaintext (Binary):	0001 0010 0011 0100 0101 0110 0111 1000
kBit:	7
cipherTextBlock:	5
Key[]:	{0xab1a16be, 0xc4163a89, 0x87e5b018, 0x65ed8705}
IV	11111111111111111111111111111111111111111111111111111111 11111111
Output:	$Output = enrypt(IV, key)$
plaintext:	0000001 0010001 1010001 0101100 1111000
Output^plaintextKbit	$OutputKbit = Output \,\&\, \left(allOne \ll (64 - kBit)\right)$ $OutputKbit = rotateRight(OutputKbit, 64 - kBit)$ $temp = Output\text{^}plaintext$ $temp = rotateRight(temp, 64 - kBit)$
cipherText:	$(cipherText \ll kBit)\|temp$

Block:	1
IV: (Shift kBit to left)	$IV = IV \ll kBit$
OutputKbit:	0000000000000000000000000000000000000000000000000000000001011110
IV:	$IV = iv \mid OutputKbit$
Key[]:	{0xab1a16be, 0xc4163a89, 0x87e5b018, 0x65ed8705}
IV	1111111111111111111111111111111111111111111111111111111111011110
Output:	$Output = enrypt(IV, key)$
plaintext:	$plaintext = (plaintext \ll kBit)$
Output^plaintext Kbit	$OutputKbit = Output \,\&\, \left(allOne \ll (64 - kBit)\right)$ $OutputKbit = rotateRight(OutputKbit, 64 - kBit)$ $temp = Output\text{^}plaintext$ $temp = rotateRight(temp, 64 - kBit)$
cipherText:	1011111 1011101

- The steps should be the same for the rest of the iterations.

TEA in C++ - Encryption function

```
void encrypt(unsigned int* v, unsigned int* k) {
    unsigned int v0=v[0], v1=v[1], sum=0; /* setup */
    unsigned int delta=0x9e3779b9; /* key schedule constant */
    unsigned int k0=k[0], k1=k[1], k2=k[2], k3=k[3]; /* cache key */

    for (int i=0; i < 32; i++) { /* basic cycle start */
        sum += delta;
        v0 += (v1<<4)+k0 ^ v1+sum ^ (v1>>5)+k1;
        v1 += (v0<<4)+k2 ^ v0+sum ^ (v0>>5)+k3; /* end cycle */
    }
    v[0]=v0; v[1]=v1;
}
```

TEA in C++ - Decryption function

```
void decrypt(unsigned int* v, unsigned int* k) {
    unsigned int v0=v[0], v1=v[1], sum=0xC6EF3720; /* setup */
    unsigned int delta=0x9e3779b9; /* key schedule constant */
    unsigned int k0=k[0], k1=k[1], k2=k[2], k3=k[3]; /* cache key */

    for (int i=0; i<32; i++) { /* basic cycle start */
        v1 -= (v0 << 4)+k2 ^ v0+sum ^ (v0 >> 5)+k3;
        v0 -= (v1 << 4)+k0 ^ v1+sum ^ (v1 >> 5)+k1;
        sum -= delta; /* end cycle */
    }
    v[0]=v0; v[1]=v1;
}
```

TEA in Java – Encryption function

```
private final static int DELTA = 0x9e3779b9;
private final static int DECRYPT_SUM_INIT = 0xC6EF3720;
private final static long MASK32 = (1L << 32) - 1;
public static long encrypt(long in, int[] k) {
    int v1 = (int) in;
    int v0 = (int) (in >>> 32);
    int sum = 0;
    for (int i = 0; i < 32; i++) {
        sum += DELTA;
        v0 += ((v1 << 4) + k[0]) ^ (v1 + sum) ^ ((v1 >>> 5) + k[1]);
        v1 += ((v0 << 4) + k[2]) ^ (v0 + sum) ^ ((v0 >>> 5) + k[3]);
    }
    return (v0 & MASK32) << 32 | (v1 & MASK32);
}
```

TEA in Java – Decryption function

```
public static long decrypt(long in, int [] k) {
        int v1 = (int) in;
        int v0 = (int) (in >>> 32);
        int sum = DECRYPT_SUM_INIT;
        for (int i=0; i<32; i++) {
            v1 -= ((v0<<4) + k[2]) ^ (v0 + sum) ^ ((v0>>>5) + k[3]);
            v0 -= ((v1<<4) + k[0]) ^ (v1 + sum) ^ ((v1>>>5) + k[1]);
            sum -= DELTA;
        }
        return (v0 & MASK32) << 32 | (v1 & MASK32);
    }
```

CSCI361-
Cryptography and Secure Applications

K-Bit OFB Decryption

Block:	0
cipherText (Hex)	5fba955f5
ciphertext (Long):	25697015285
ciphertext (Bin):	0101 1111 1011 1010 1001 0101 0101 1111 0101
kBit:	7
cipherTextBlock:	5
Key[]:	{0xab1a16be, 0xc4163a89, 0x87e5b018, 0x65ed8705}
IV	11111111111111111111111111111111111111111111111111111111111
Output:	1011110 1110000 0101011 1111001 0000001 1010000 1100000 0000110 0100001 0
cipherText (Bin):	1011111 1011101 0100101 0101011 1110101
Output^ciphertextKbit	0000001
plaintext (Bin):	0000001

Block:	1
IV:	11111111111111111111111111111111111111111111111111111110000000
Output-kBit:	00000000000000000000000000000000000000000000000000000001011110
IV:	11111111111111111111111111111111111111111111111111111111011110
Key[]:	{0xab1a16be, 0xc4163a89, 0x87e5b018, 0x65ed8705}
IV	11111111111111111111111111111111111111111111111111111111011110
Output:	1001100 0000100 0001100 0000101 1001101 1011000 0001111 1111111 0000111 1
cipherText (Bin):	1011101 0100101 0101011 1110101 0000000
Output^ciphertextKbit	0010001
plaintext (Bin):	0000001 0010001

Block:	2
IV:	11111111111111111111111111111111111111111111111111111011 1100000000
Output-kBit:	00000000000000000000000000000000000000000000000000000000 0001001100
IV:	11111111111111111111111111111111111111111111111111111011 1101001100
Key[]:	{0xab1a16be, 0xc4163a89, 0x87e5b018, 0x65ed8705}
IV	11111111111111111111111111111111111111111111111111111011 1101001100
Output:	1110100 0110111 1010010 1001001 0100101 0111001 0001101 0011110 1110101 1
cipherText (Bin):	0101011 1110101 0000000 0000000 0000000
Output^ciphertextKbit	1010001
plaintext (Bin):	0000001 0010001 1010001

Block:	3
IV:	11111111111111111111111111111111111111111111101111010011000000000
Output-kBit:	00000000000000000000000000000000000000000000000000000001110100
IV:	11111111111111111111111111111111111111111111101111010011001110100
Key[]:	{0xab1a16be, 0xc4163a89, 0x87e5b018, 0x65ed8705}
IV	11111111111111111111111111111111111111111111101111010011001110100
Output:	0000111 0101001 0011010 0100111 0101100 0110010 0100000 1010001 1100110 1
cipherText (Bin):	0101100 1111000 0000000 0000000 0000000
Output^ciphertextKbit	0101100
plaintext (Bin):	0000001 0010001 1010001 0101100

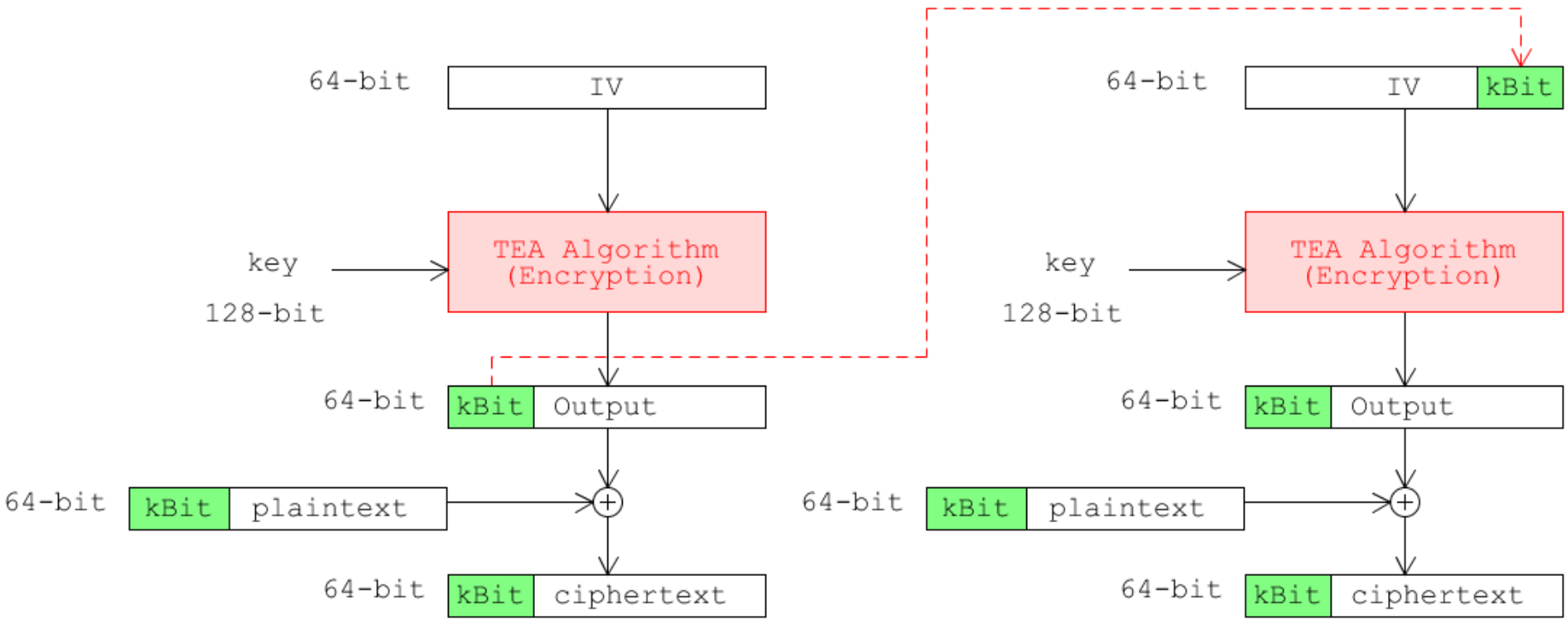
Block:	4
IV:	11111111111111111111111111111111111101111010011001101000000000
Output-kBit:	00000000000000000000000000000000000000000000000000000000111
IV:	11111111111111111111111111111111111101111010011001101000000111
Key[]:	{0xab1a16be, 0xc4163a89, 0x87e5b018, 0x65ed8705}
IV	11111111111111111111111111111111111101111010011001101000000111
Output:	0001101 1110101 0101010 1000000 1010111 1001110 0110101 0110000 0010001 1
plaintext:	1110101 0000000 0000000 0000000 0000000
Output^plaintext Kbit	1111000
plaintext (Bin):	0000001 0010001 1010001 0101100 1111000
Plaintext (Hex):	12345678

Block:	0
cipherText (Hex)	5fba955f5
ciphertext (Long):	25697015285
ciphertext (Bin):	0101 1111 1011 1010 1001 0101 0101 1111 0101
kBit:	7
cipherTextBlock:	5
Key[]:	{0xab1a16be, 0xc4163a89, 0x87e5b018, 0x65ed8705}
IV	11111111111111111111111111111111111111111111111111111111 11111111
Output:	$Output = enrypt(IV, key)$
ciphertext:	1011111 1011101 0100101 0101011 1110101
Output^ciphertextKbit	$OutputKbit = Output \,\&\, \left(allOne \ll (64 - kBit)\right)$ $OutputKbit = rotateRight(OutputKbit, 64 - kBit)$ $temp = Output\text{^}ciphertext$ $temp = rotateRight(temp, 64 - kBit)$
plaintext (Bin):	$(plainText \ll kBit)\|temp$

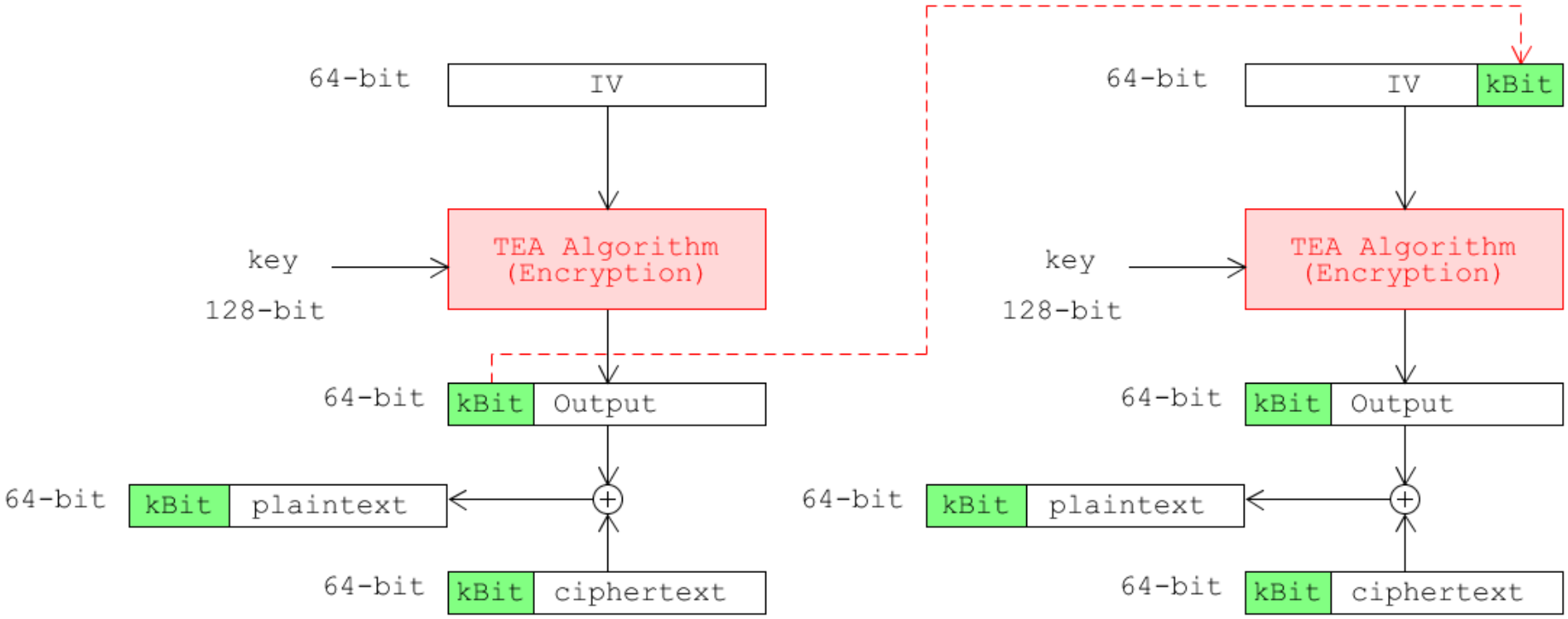
Block:	1
IV: (Shift kBit to left)	$IV = IV \ll kBit$
OutputKbit:	0000000000000000000000000000000000000000000000000000000001011110
IV:	$IV = iv \mid OutputKbit$
Key[]:	{0xab1a16be, 0xc4163a89, 0x87e5b018, 0x65ed8705}
IV	1111111111111111111111111111111111111111111111111111111111011110
Output:	$Output = enrypt(IV, key)$
ciphertext:	$ciphertext = (ciphertext \ll kBit)$
Output^ciphertextKbit	$OutputKbit = Output \,\&\, \left(allOne \ll (64 - kBit)\right)$ $OutputKbit = rotateRight(OutputKbit, 64 - kBit)$ $temp = Output \text{^} ciphertext$ $temp = rotateRight(temp, 64 - kBit)$
plaintext (Bin):	$(plainText \ll kBit) \mid temp$

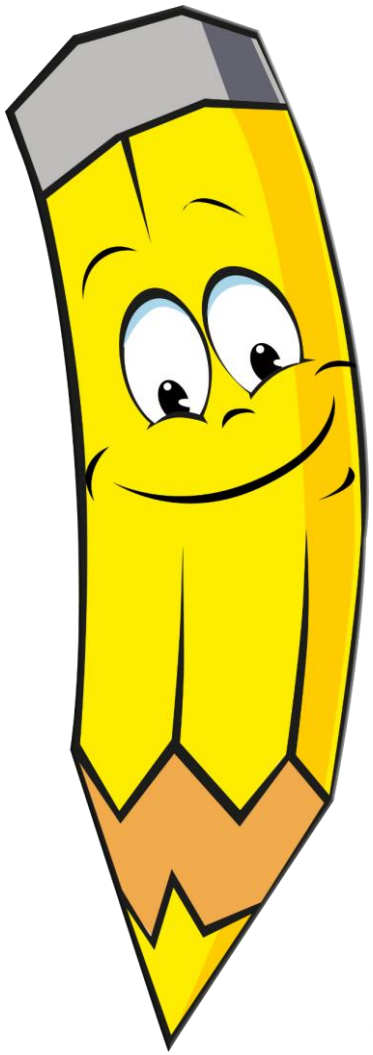
- The steps should be the same for the rest of the iterations.

kBit-OFB Encryption



kBit-OFB Decryption

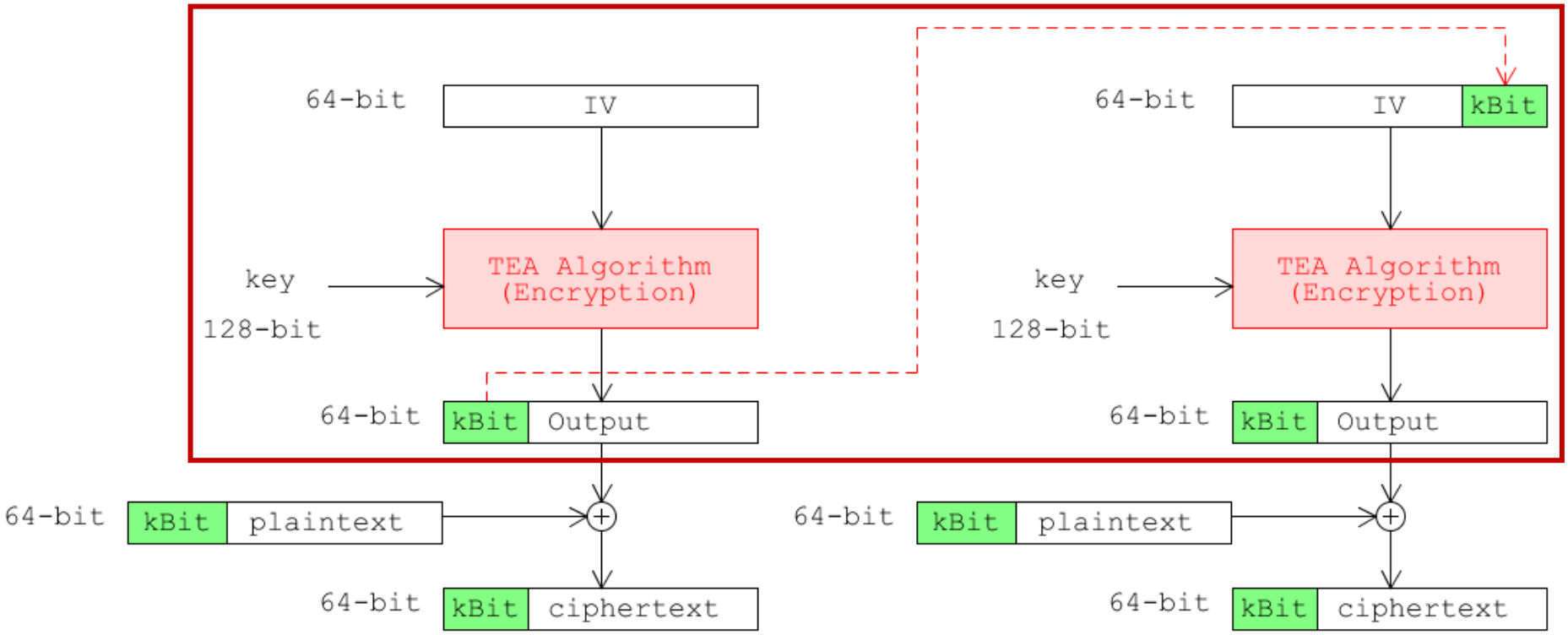




K-Bit OFB

Only need to use encryption algorithm to encrypt and decrypt in OFB mode

kBit-OFB Encryption



kBit-OFB Decryption