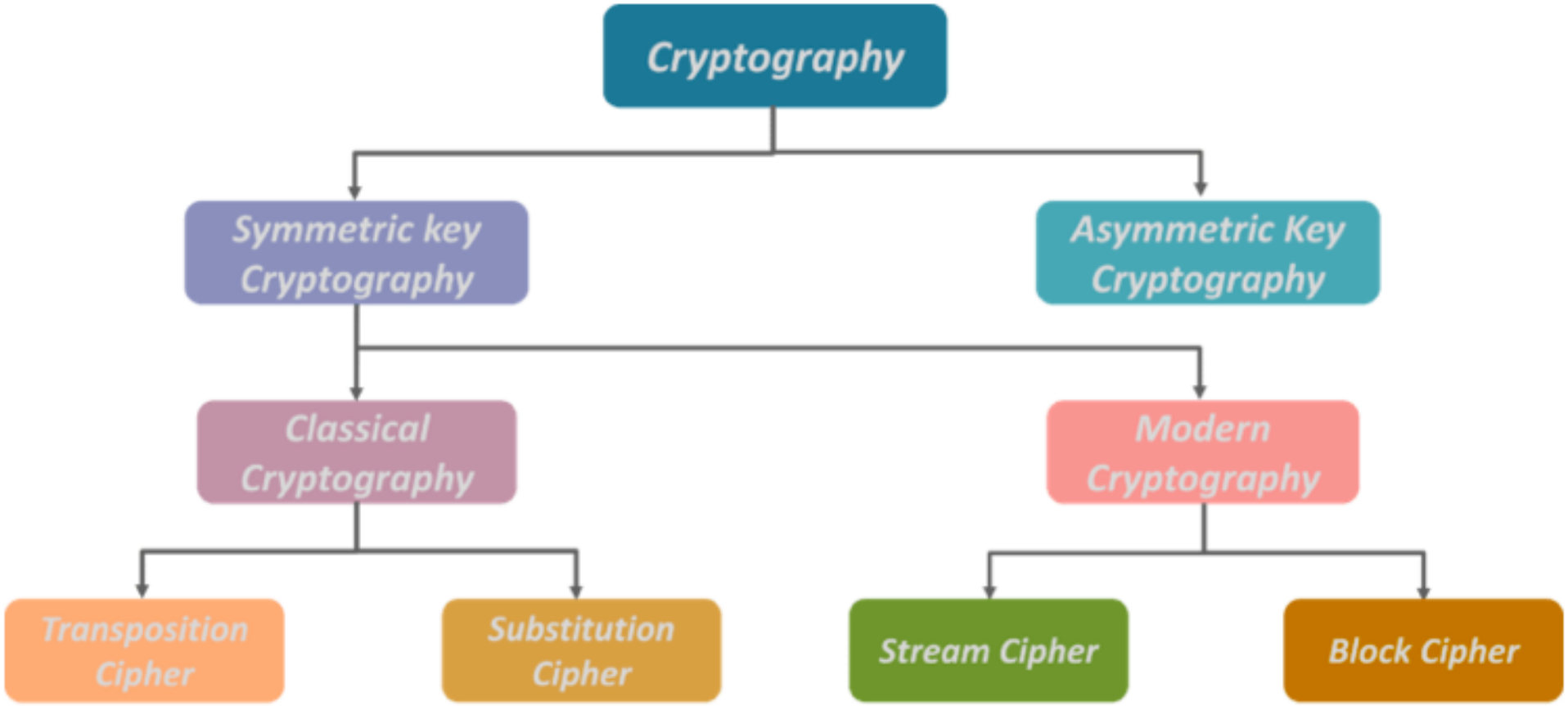
Practical Cryptography

Handout 4 – Block Cipher Operation Modes

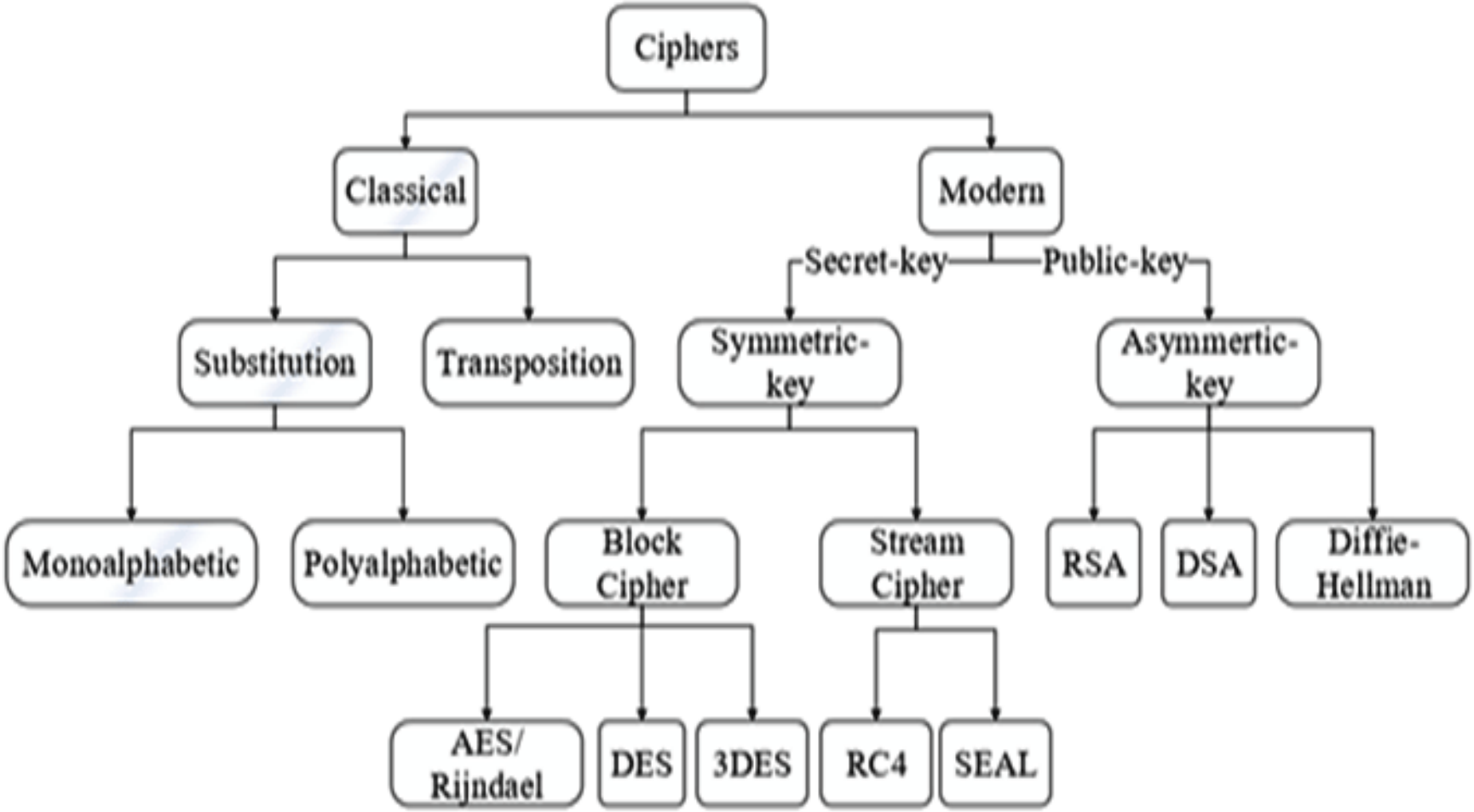
Kasun de Zoysa
kasun@ucsc.cmb.ac.lk

UNIVERSITY OF COLOMBO SCHOOL OF COMPUTING

Cryptography

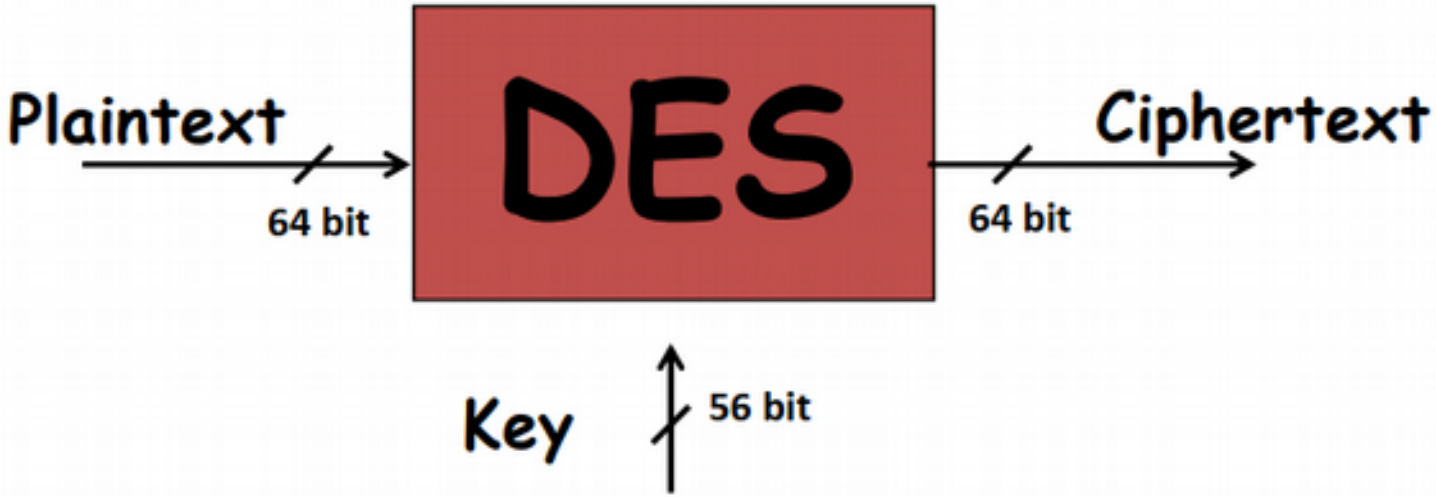


Ciphers


Ciphers
Classical
Modern
Secret-key
Public-key
Substitution
Transposition
Symmetric-key
Asymmertic-key
Monoalphabetic
Polyalphabetic
Block Cipher
Stream Cipher
RSA
DSA
Diffie-Hellman
AES/ Rijndael
DES
3DES
RC4
SEAL

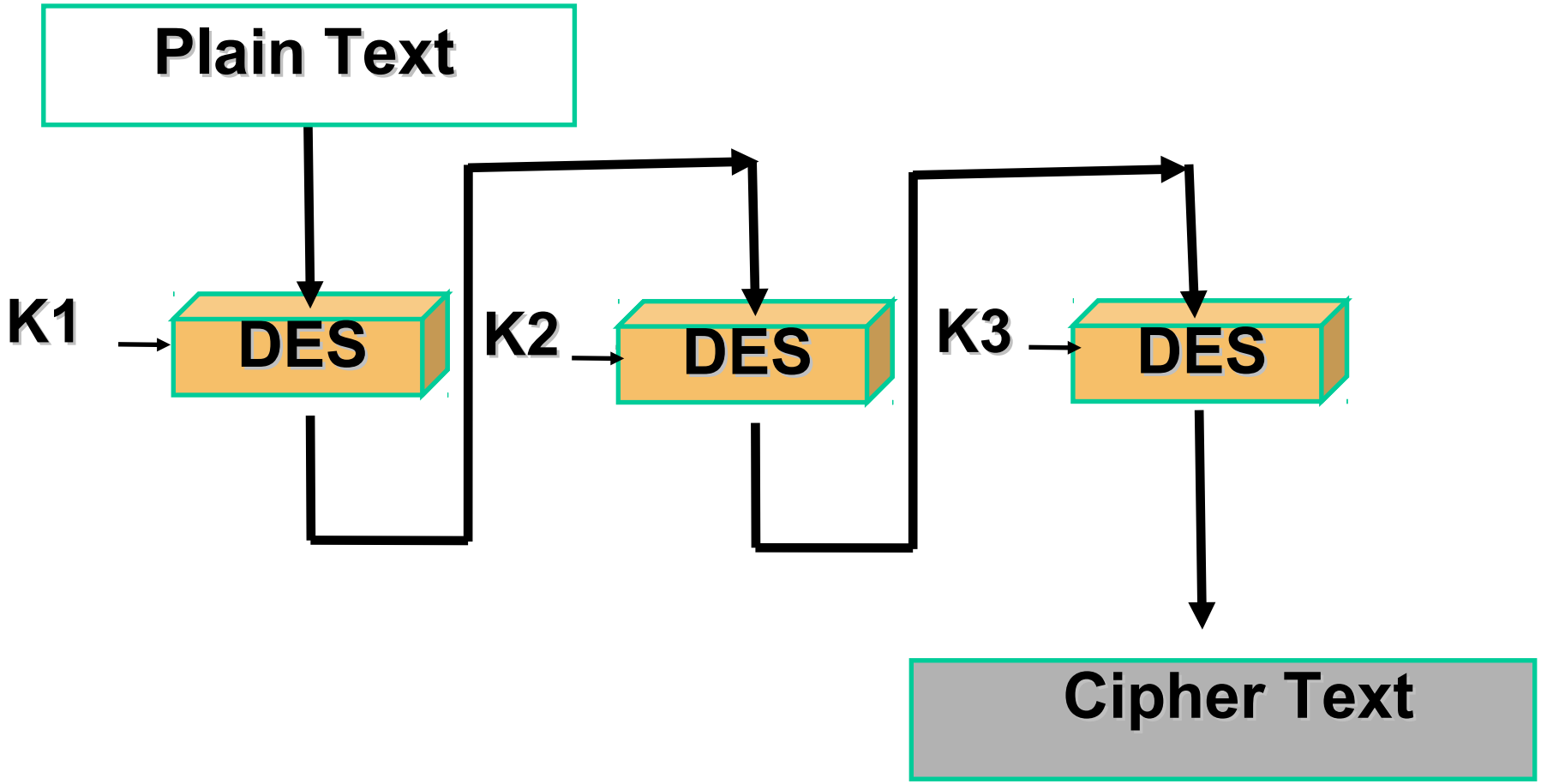
DES Features

- Features:
  - Block size = 64 bits
  - Key size = 56 bits (in reality, 64 bits, but 8 are used as parity-check bits for error control, see next slide)
  - Number of rounds = 16
  - 16 intermediary keys, each 48 bits



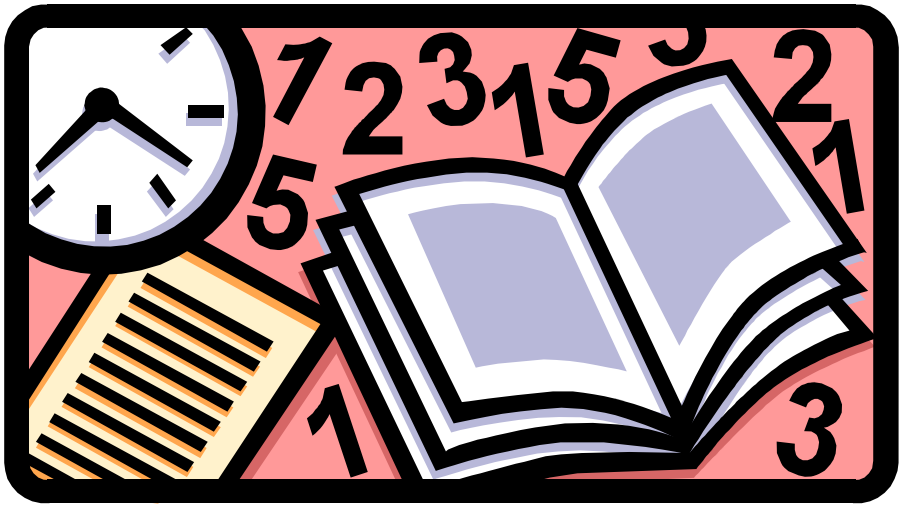
8

Triple DES - Encryption



Advance Encryption Standard (AES)

- In 2001, National Institute of Standards and Technology (NIST) issued AES known as FIPS 197
- AES is based on Rijndael proposed by Joan Daemen, Vincent Rijmen from Belgium



Advance Encryption Standard (AES)

- AES has block length 128
- Supported key lengths are 128, 192 and 256
- AES requires 10 rounds of processing
- Key is expanded into 10 individual keys
- Decryption algorithm uses the expanded keys in reverse order
- Decryption algorithm is not identical to the encryption algorithm

Block Ciphers - Modes of Operation

- Block ciphers encrypt fixed size blocks
  - E.g. DES encrypts 64-bit blocks, with 56-bit key

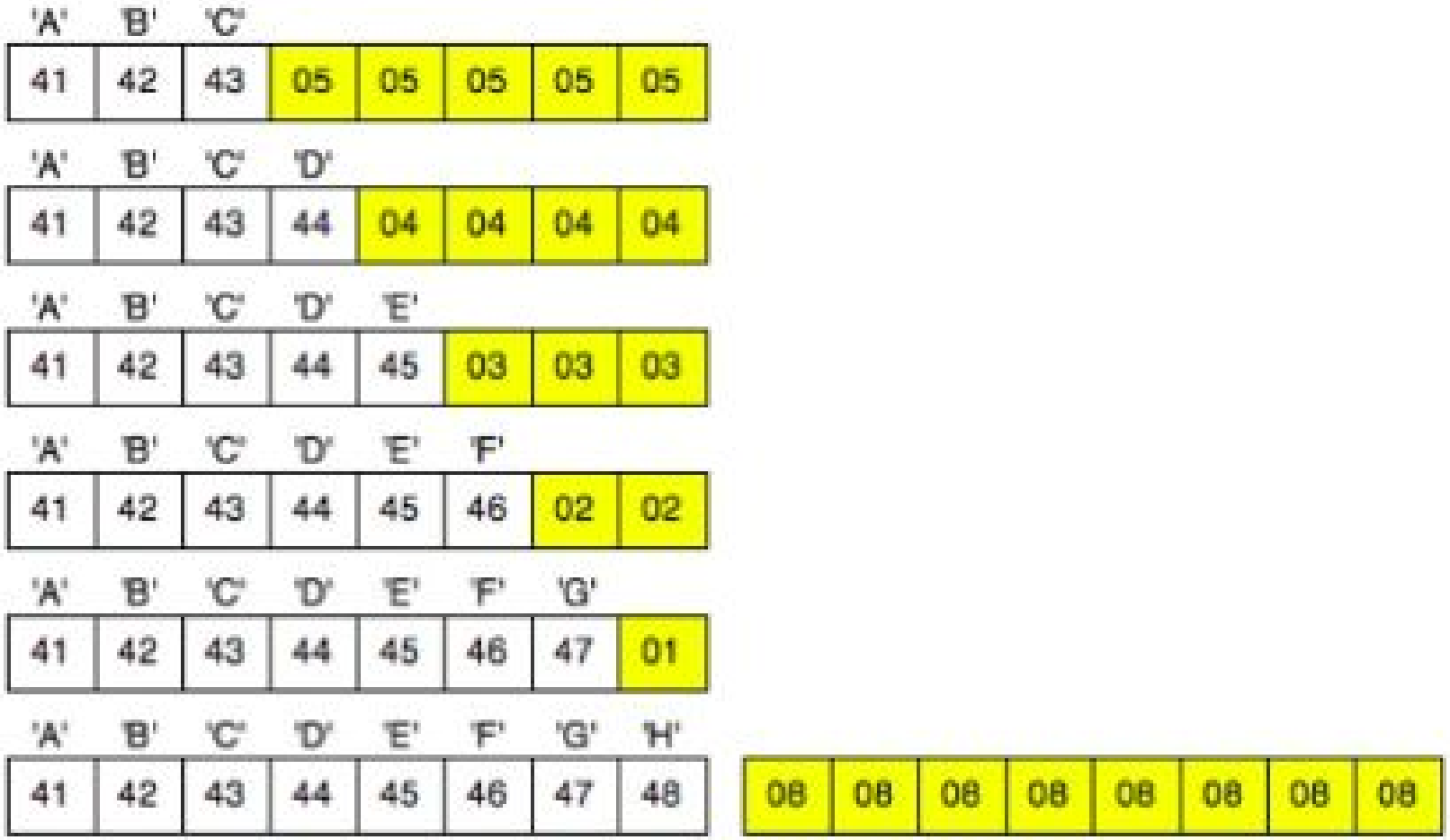
- Given that one needs to encrypt arbitrary amount of information, how do we use in practice,
  - Four modes were defined for DES in ANSI standard
  - **ANSI X3.106-1983 Modes of Use**
  - Subsequently now have 5 for DES and AES

PKCS5 Padding Scheme

- Assume block cipher is 64-bits
- Any message not a multiple of 8 bytes is padded
- Valid pad:
- 1 byte needed: 0x1
- 2 bytes needed: 0x2 0x2
- 3 bytes needed: 0x3 0x3 0x3
- ....
- No padding: 0x8 0x8 0x8 0x8 0x8 0x8 0x8 0x8

  (If the length of the original data is an integer multiple of the block size B, then an extra block of bytes with value B is added.)

PKCS5 Padding Scheme



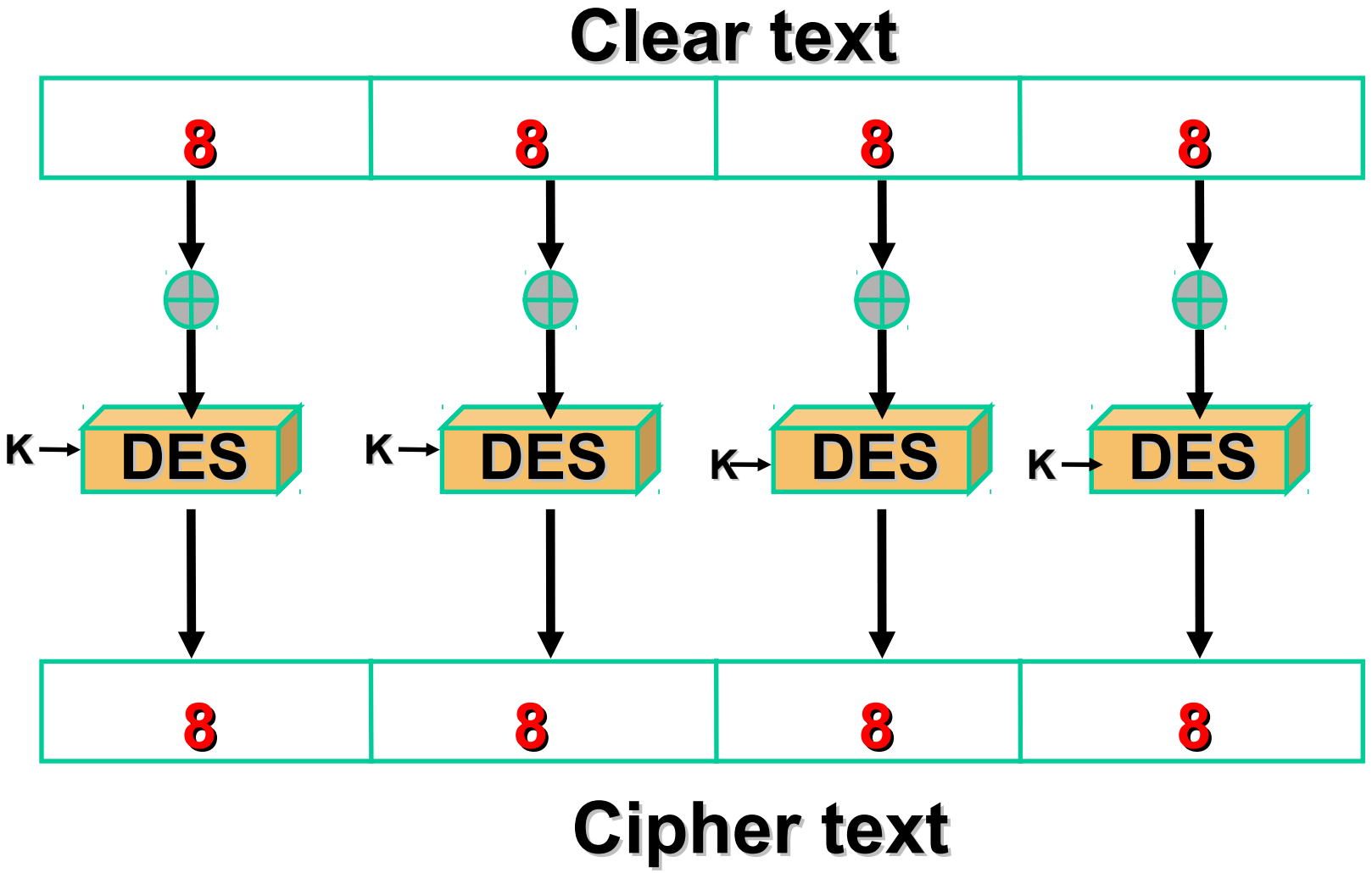
Electronic Codebook Book (ECB)

- Message is broken into independent blocks which are encrypted
- Each block is a value which is substituted, like a codebook, hence name
- Each block is encoded independently of the other blocks

$$C_i = DES_K(P_i)$$

- Uses: secure transmission of single values

Electronic Code Book Mode (ECB)

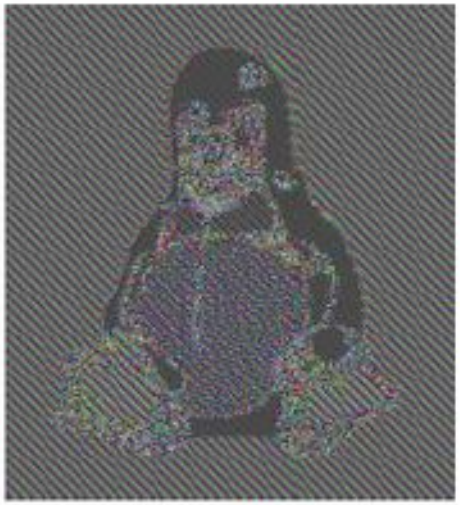


Advantages and Limitations of ECB

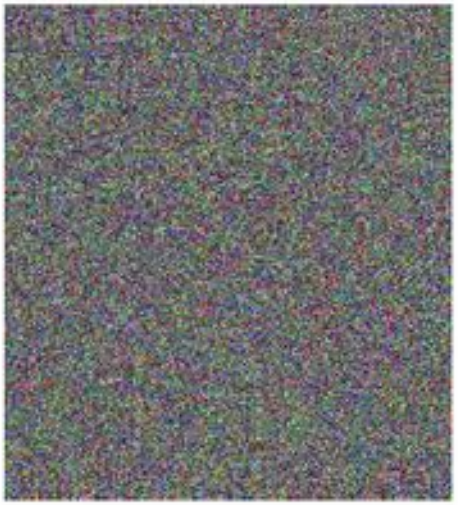
- Repetitions in message may show in ciphertext if aligned with message block particularly with data such graphics or with messages that change very little
- Weakness due to encrypted message blocks being independent
- Main use is sending a few blocks of data

ECB vs CBC

Original



Encrypted using ECB mode



Encrypted using other modes

Electronic codebook (ECB), Cipher block chaining (CBC), Cipher feedback (CFB), Output feedback (OFB)

Cipher Block Chaining (CBC)

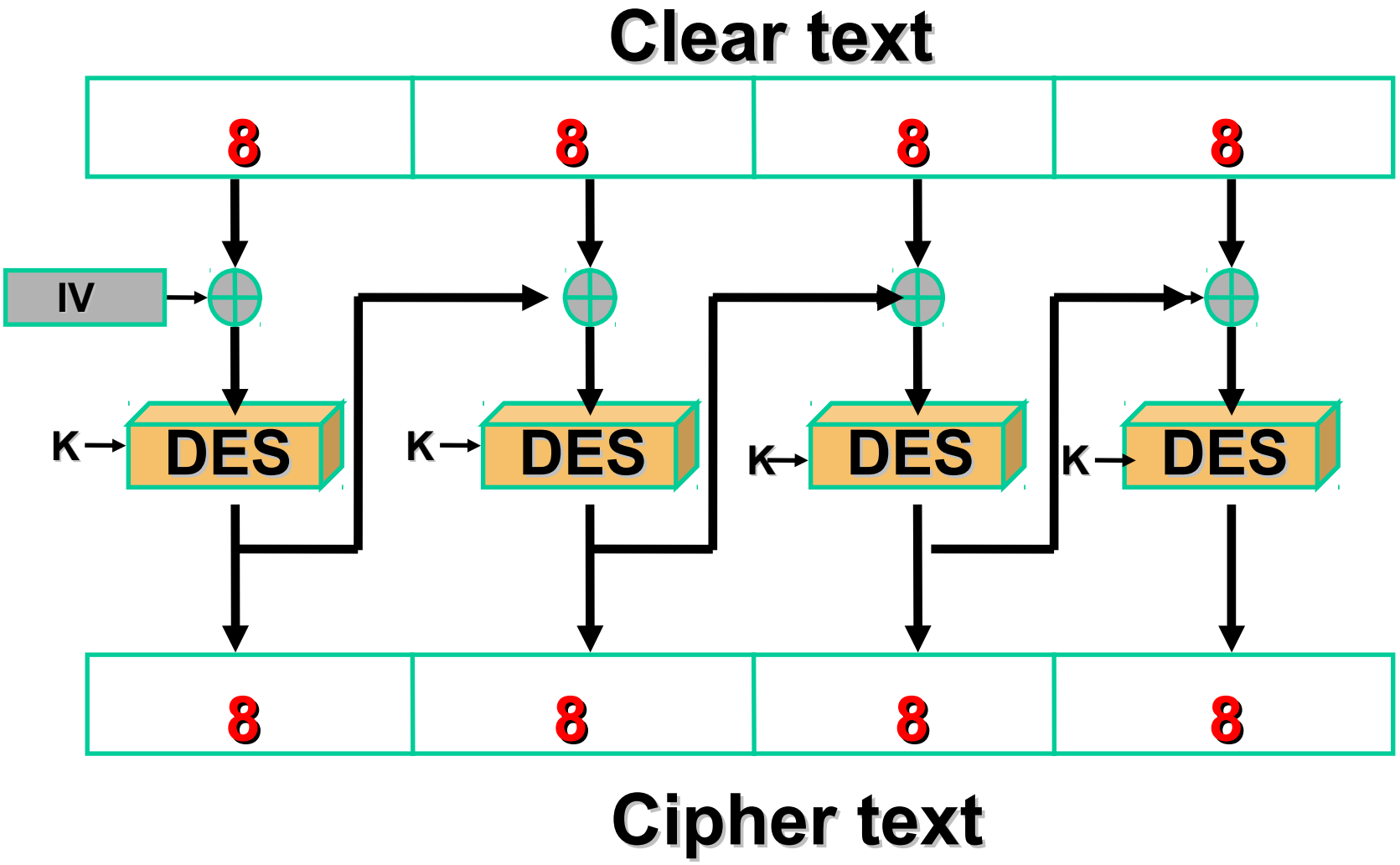
- Message is broken into blocks
- But these are linked together in the encryption operation
- Each previous cipher blocks is chained with current plaintext block, hence name
- Use Initial Vector (IV) to start process

$$C_i = DES_K(P_i \; XOR \; C_{i\text{-}1})$$

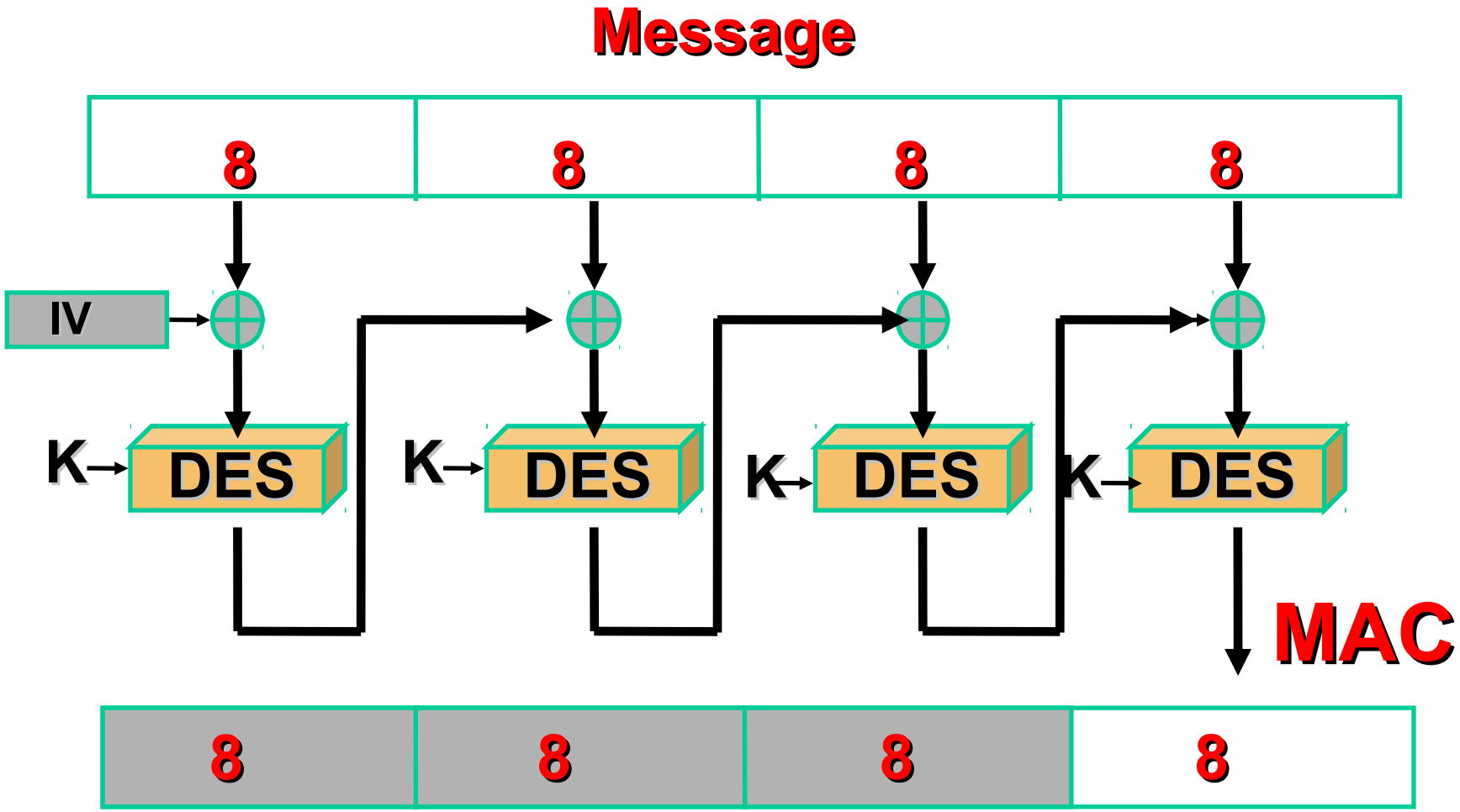
$$C_{-1} = IV$$

- Uses: bulk data encryption, authentication

Cipher Block Chaining Mode (CBC)



MAC based on CBC



Advantages and Limitations of CBC

- Each ciphertext block depends on **all** preceding message blocks thus a change in the message affects all ciphertext blocks after the change as well as the original block

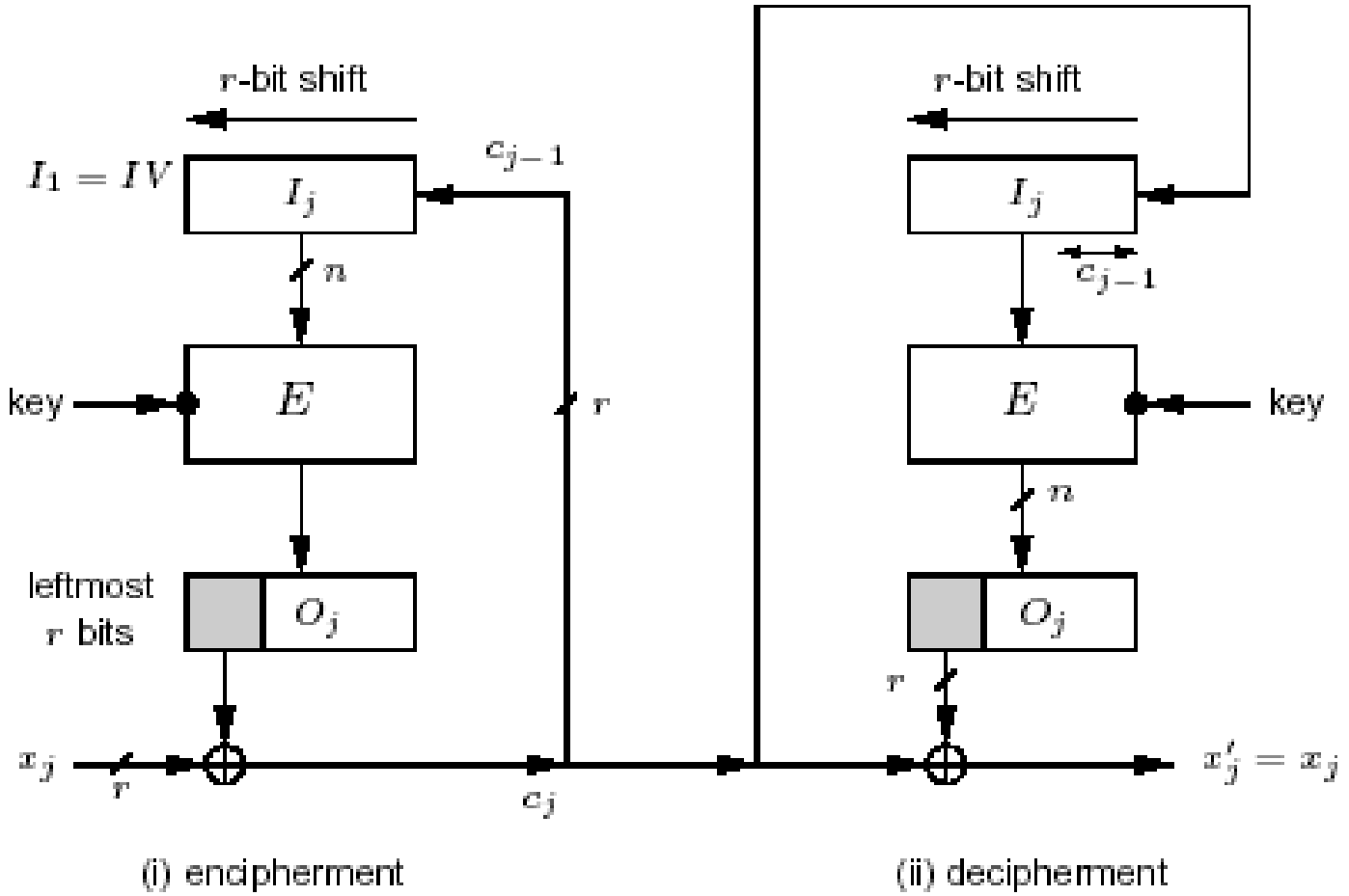
- Need **Initial Value** (IV) known to sender & receiver however if IV is sent in the clear, an attacker can change bits of the first block, and change IV to compensate hence either IV must be a fixed value or it must be sent encrypted in ECB mode before rest of message

- At end of message, handle possible last short block by padding either with known non-data value (e.g. nulls) or pad last block with count of pad size

Cipher feed back (CFB) mode

- A Stream Cipher where the Ciphertext is used as feedback into the Key generation source to develop the next Key Stream
- The Ciphertext generated by performing an XOR on the Plaintext with the Key Stream the same number of bits as the Plaintext
- Errors will propagate in this mode

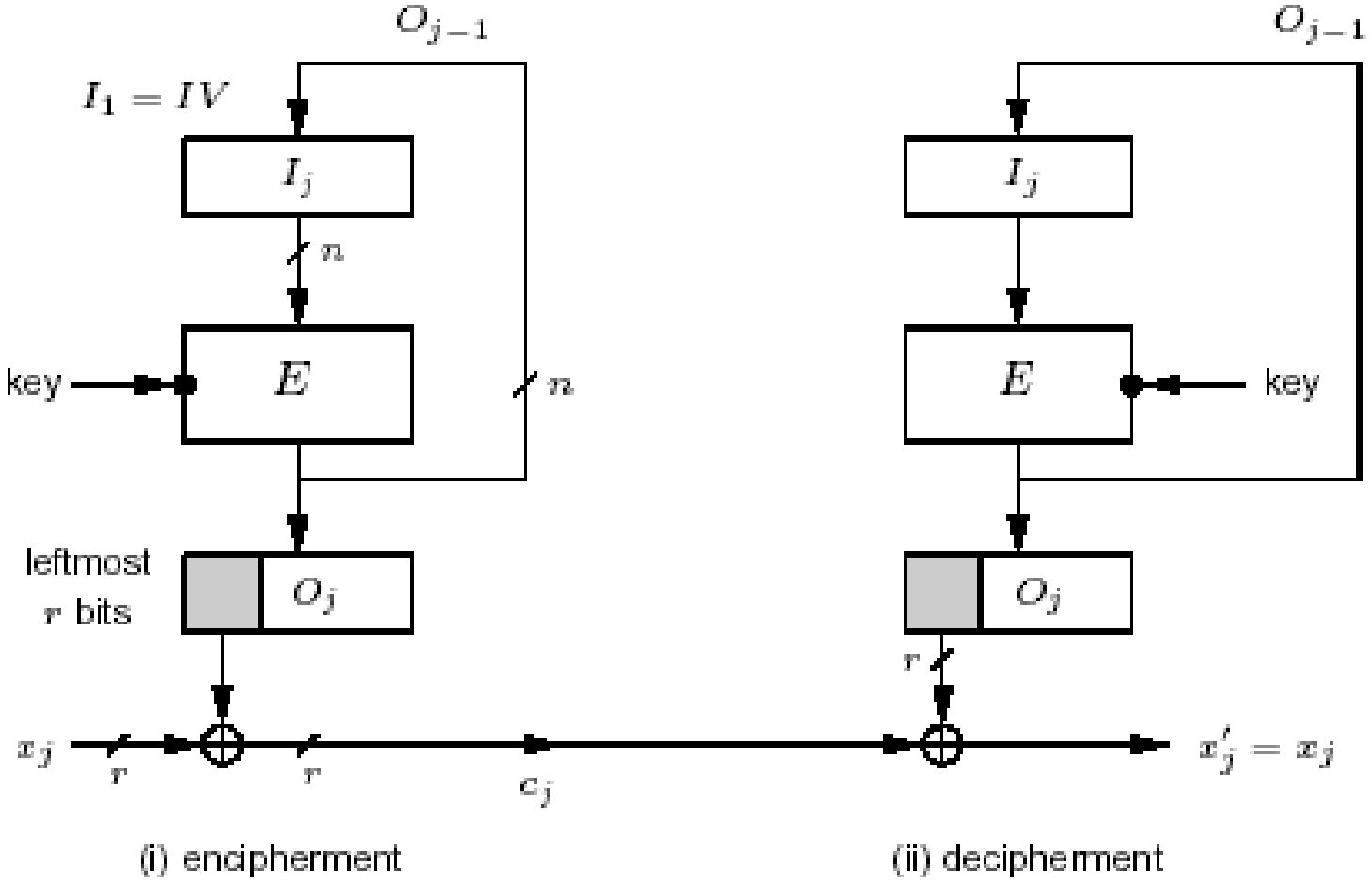
Cipher Feedback Mode (CFB)



Output Feed Back(OFB) mode

- A Stream Cipher that generates the Ciphertext Key by XORing the Plaintext with a Key Stream.
- Requires an Initialization Vector
- Feedback is used to generate the Key Stream – therefore the Key Stream will vary
- Errors will not propagate in this mode

Output Feedback Mode (OFB)



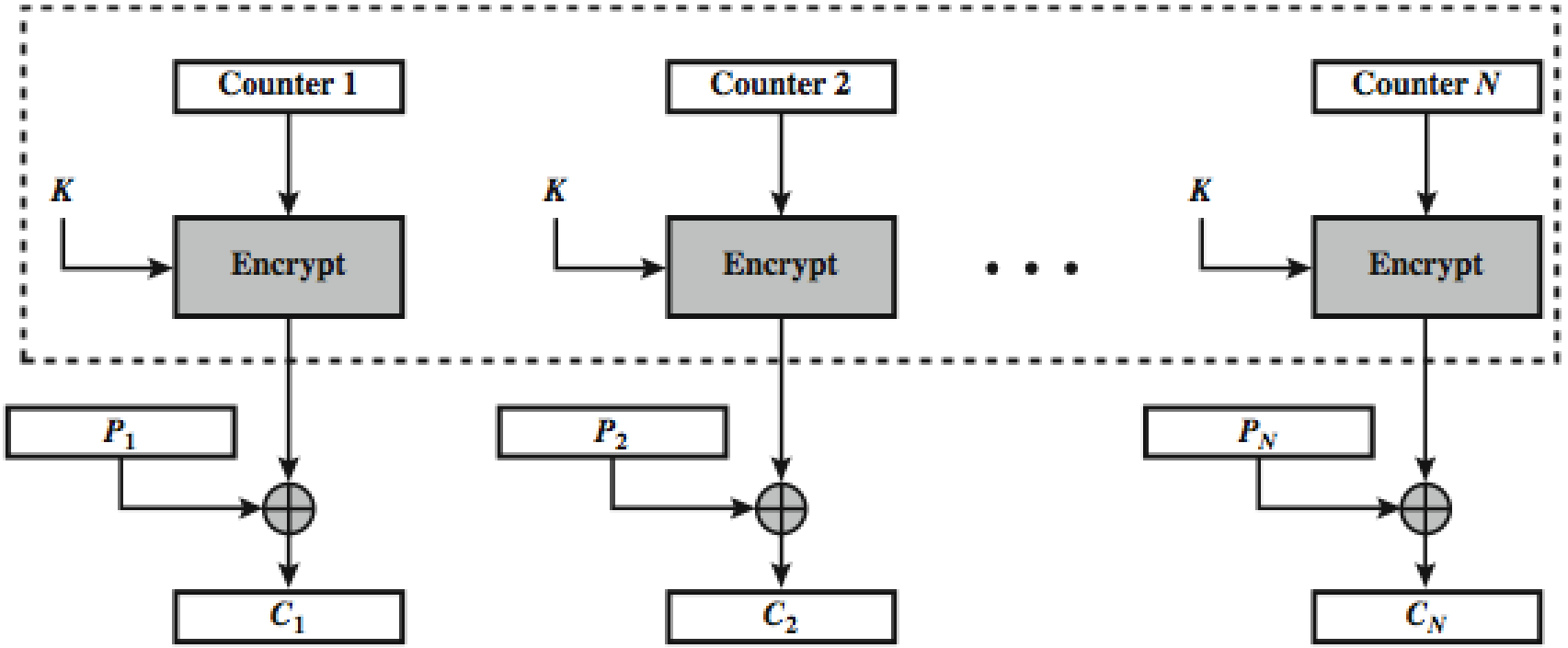
Counter (CTR)

a “new” mode, though proposed early on similar to OFB but encrypts counter value rather than any feedback value

Oi = EK(i)
Ci = Pi XOR Oi

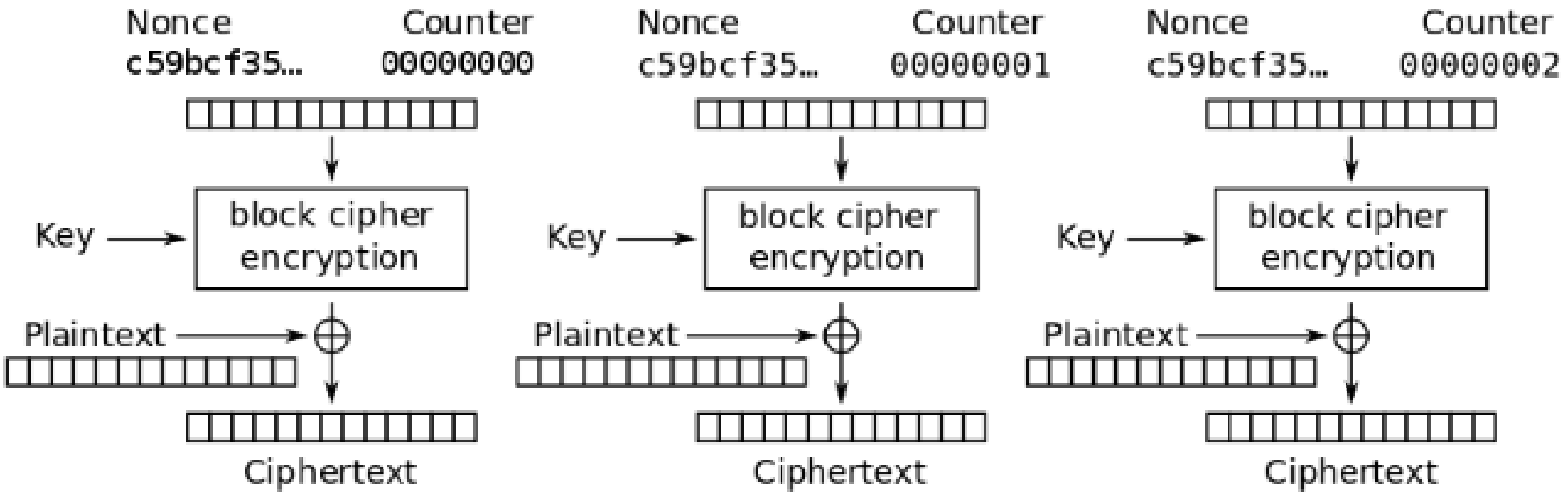
must have a different key & counter value for every plaintext block (never reused) again
uses: high-speed network encryptions

CTR



(a) Encryption

CTR


Nonce
c59bcf35...
Counter
00000000
Key
block cipher encryption
Plaintext
Ciphertext
Nonce
c59bcf35...
Counter
00000001
Key
block cipher encryption
Plaintext
Ciphertext
Nonce
c59bcf35...
Counter
00000002
Key
block cipher encryption
Plaintext
Ciphertext

Advantages and Limitations of CTR

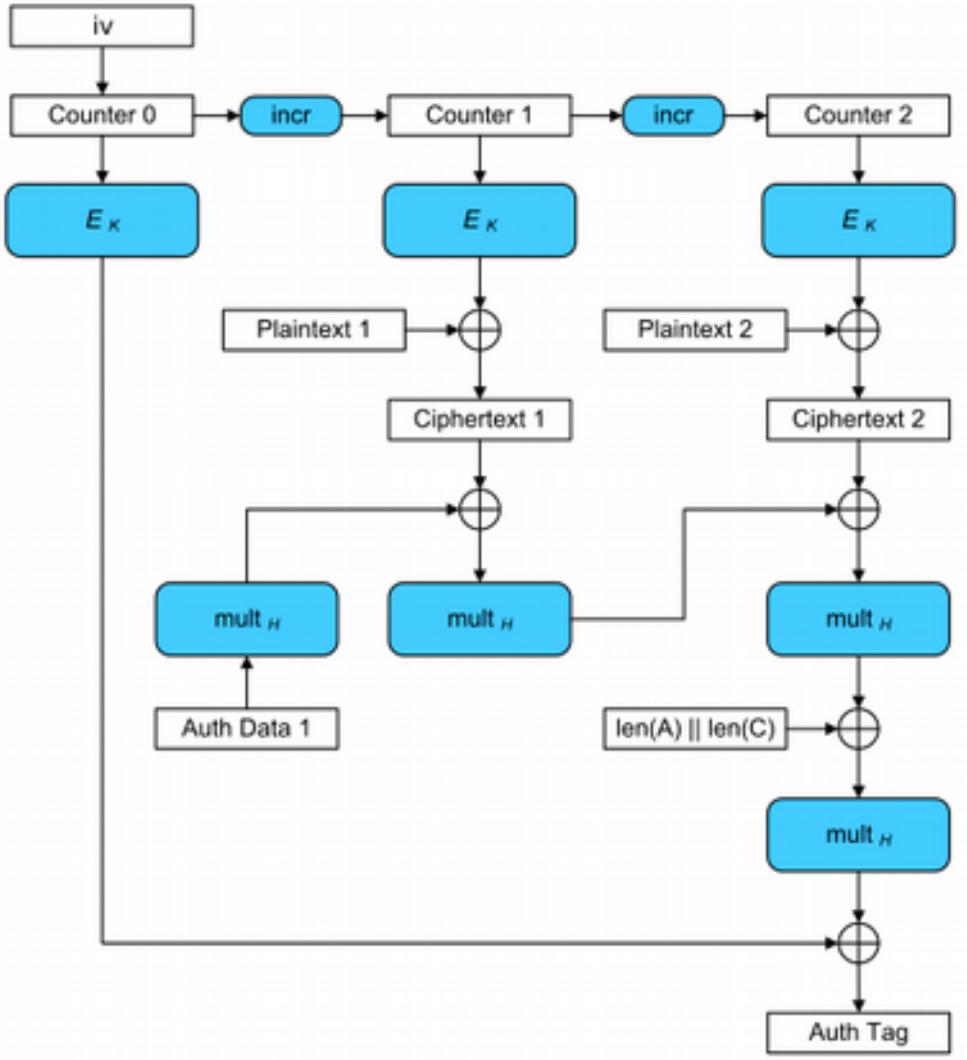
- can do parallel encryptions in h/w or s/w
- can preprocess in advance of need
- good for high speed links
- random access to encrypted data blocks
- provable security (good as other modes)
- but must ensure never reuse key/counter values, otherwise could break

GCM (Galois/Counter) Block Mode

The GCM mode uses a counter, which is increased for each block and calculated a message authentication tag (MAC code) after each processed block.

The final authentication tag is calculated from the last block. Like all counter modes, GCM works as a stream cipher, and so it is essential that a different IV is used at the start for each stream that is encrypted.

The key-feature is the ease of parallel-computation of the Galois field multiplication used for authentication.



AES-GCM Authenticated Encryption

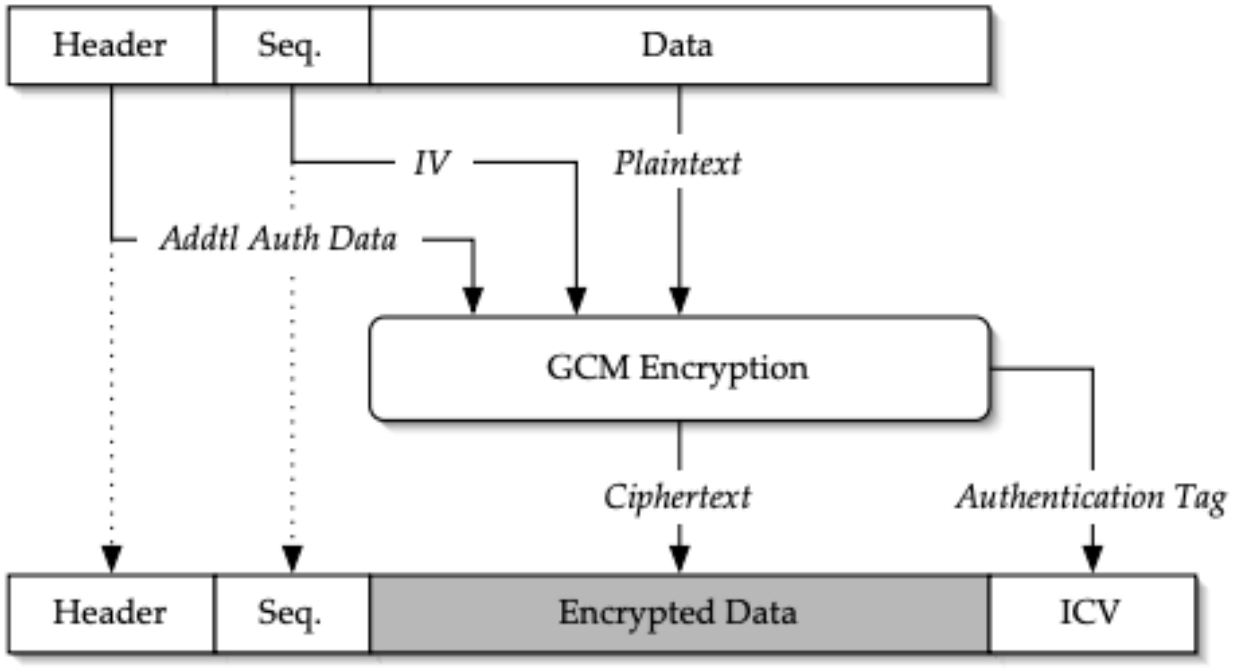
- AES-GCM Authenticated Encryption (D. McGrew & J. Viega)
  - Designed for high performance (Mainly with a HW viewpoint)
  - A NIST standard FIPS 800-38D (since 2008)
    - Included in the NSA Suite B Cryptography.
- Also in:
  - IPsec (RFC 4106)
  - IEEE P1619 Security in Storage Working Group http://siswg.net/
  - TLS 1.2
- How it works:
  - Encryption is done with AES in CTR mode
  - Authentication tag computations - "Galois Hash" :
    - A Carter-Wegman-Shoup universal hash construction: polynomial evaluation over a binary field
    - Uses GF(2^{128}) defined by the "lowest" irreducible polynomial

$$g = g(x) = x^{128} + x^7 + x^2 + x + 1$$

  - Computations based on GF(2^{128}) arithmetic

But not really the standard GF(2^{128}) arithmetic

AES- GCM



AES-GCM is the best performing Authenticated Encryption combination among the NIST standard options (esp. compared to using HMAC SHA-1)

Other Symmetric Block Ciphers

- International Data Encryption Algorithm (IDEA)
  - 128-bit key
  - Used in PGP
- Blowfish
  - Easy to implement
  - High execution speed
  - Run in less than 5K of memory

Other Symmetric Block Ciphers

- RC5
  - Suitable for hardware and software
  - Fast, simple
  - Adaptable to processors of different word lengths
  - Variable number of rounds
  - Variable-length key
  - Low memory requirement
  - High security
  - Data-dependent rotations
- Cast-128
  - Key size from 40 to 128 bits
  - The round function differs from round to round

Stream Ciphers

- Process the message bit by bit (as a stream)
- Typically have a (pseudo) random **stream key**
- Combined (XOR) with plaintext bit by bit
- Randomness of **stream key** completely destroys any statistically properties in the message

$$C_i = M_i \text{ XOR } \text{StreamKey}_i$$

- But must never reuse stream key

otherwise can remove effect and recover messages

Stream Cipher Properties

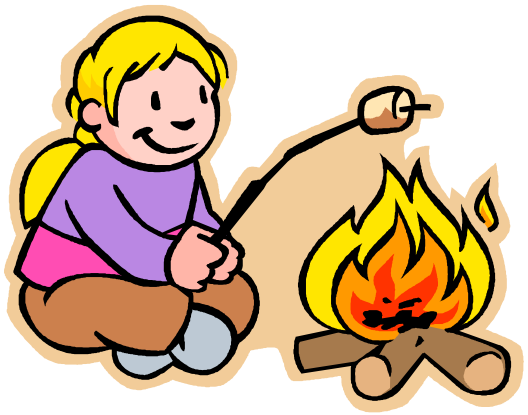
- Some design considerations are:
  - long period with no repetitions
  - statistically random
  - depends on large enough key
  - large linear complexity
  - correlation immunity
  - confusion
  - diffusion
  - use of highly non-linear Boolean functions

RC4

- A proprietary cipher owned by RSA DSI
- Another Ron Rivest design, simple but effective
- Variable key size, byte-oriented stream cipher
- Widely used (web SSL/TLS, wireless WEP)
- Key forms random permutation of all 8-bit values
- Uses that permutation to scramble input information processed a byte at a time

RC4 Security

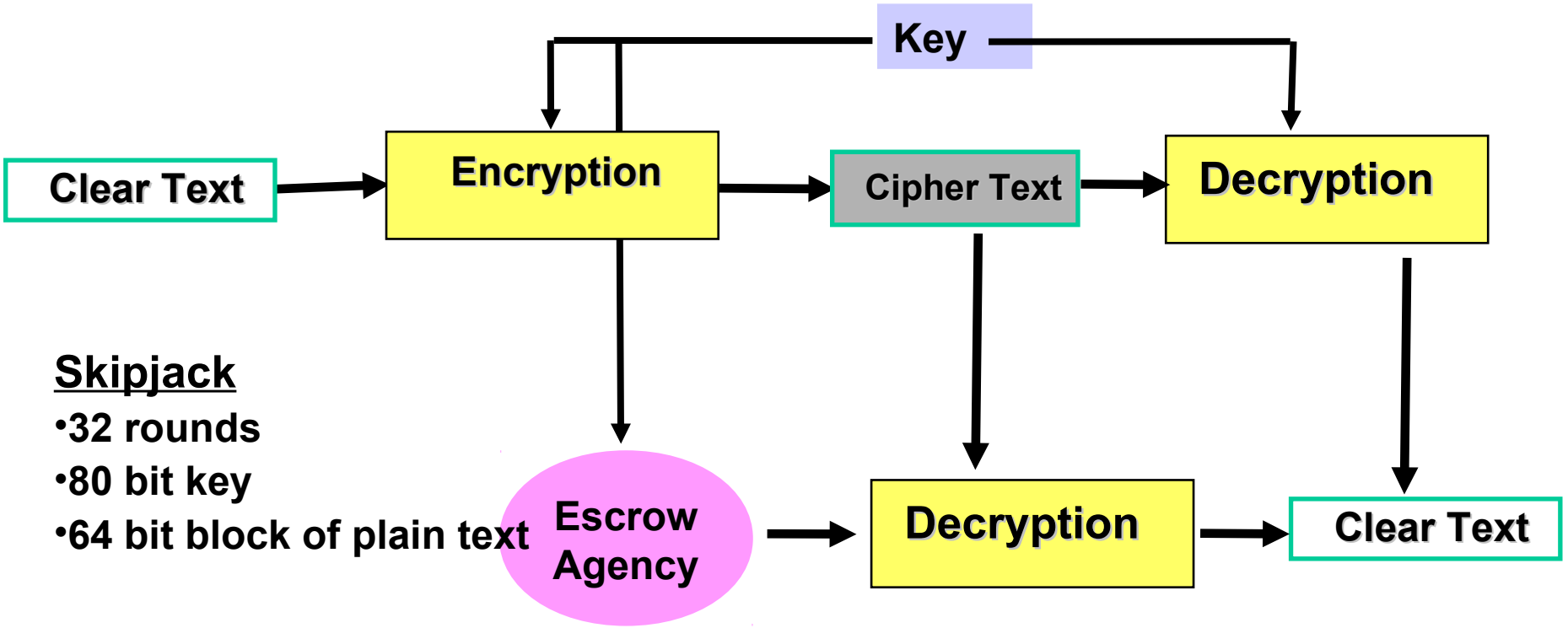
- Claimed secure against known attacks
  - have some analyses, none practical
- Result is very non-linear
- Since RC4 is a stream cipher, must **never reuse a key**



Key Escrow

- Separate agencies maintain components of private key, which, when combined, can be used to decrypt ciphertext
- Stated reason is to decrypt drug related communications
- Clipper chip is an example
  - secret algorithm
  - Unpopular, unused
- Issues include key storage, Big Brother

Key Escrow Standard



Skipjack

- 32 rounds
- 80 bit key
- 64 bit block of plain text

Random Number Generator (RNG)

The SecureRandom class is an engine class that provides the functionality of a Random Number Generator (RNG). It differs from the Random class in that it produces cryptographically strong random numbers.

Random Number Generator (RNG)

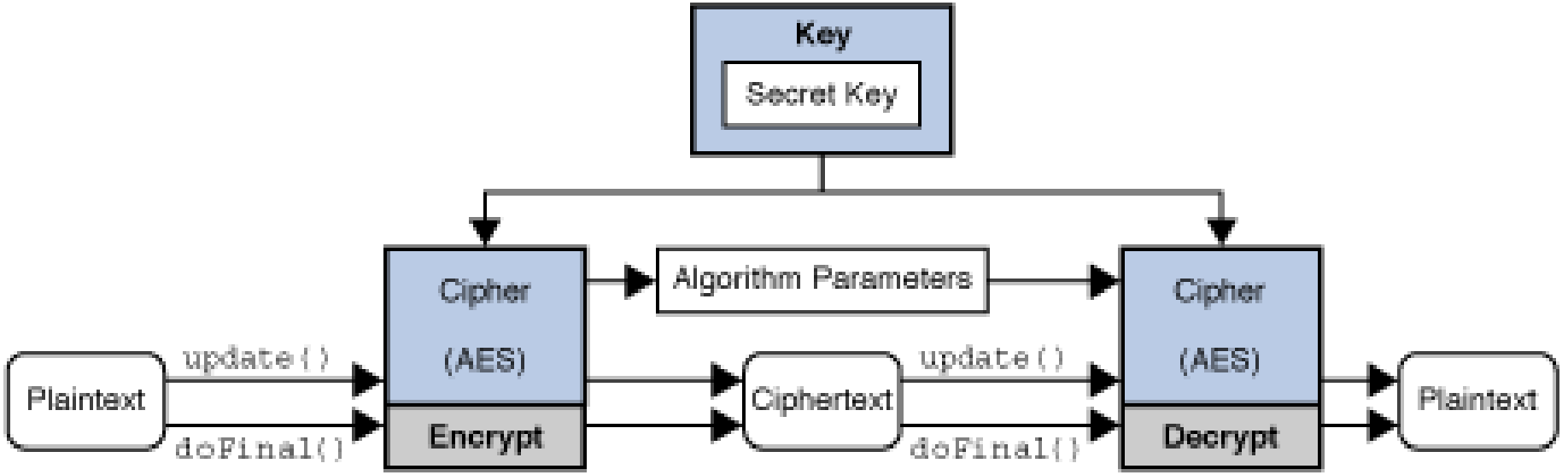
```
//Initialize secure random generator
 SecureRandom sr =   SecureRandom.getInstance("SHA1PRNG");

//Generate and set seed value
   int seedByteCount = 10;
   byte[] seed = sr.generateSeed(seedByteCount);
   sr.setSeed(seed);

// Get 256 random bits
   byte[] bytes = new byte[256/8];
   sr.nextBytes(bytes);

// Get next 256 random bits
   sr.nextBytes(bytes);
```

Encryption: AES ECB



AES-ECB Encryption

Encryption:

1. Key Generation

```
KeyGenerator    generator = KeyGenerator.getInstance("AES");
generator.init(128);
Key key = generator.generateKey();
```

2. Obtain the cipher engine

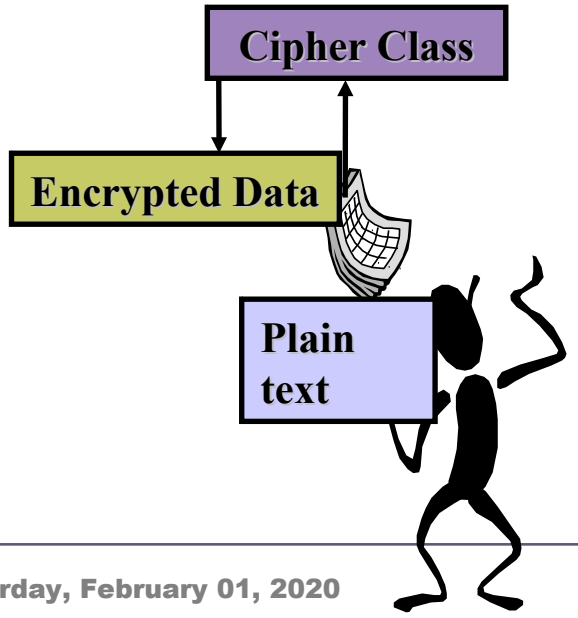
```
Cipher c = Cipher.getInstance("AES/ECB/PKCS5Padding")
```

3. Initializing the cipher engine for encryption

```
c.int(Cipher.ENCRYPT_MODE, key)
```

4. Do the padding and finish the encryption

```
byte[] cipherText = c.doFinal(input);
```



AES-ECB Decryption

Decryption:

1. Generation the same key

```
KeyGenerator    generator = KeyGenerator.getInstance("AES");
generator.init(128);
Key key = generator.generateKey();
```

2. Obtain the cipher engine

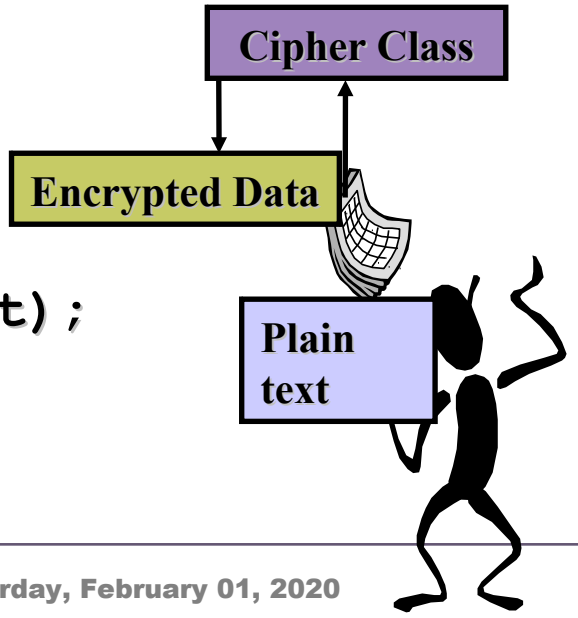
```
Cipher c = Cipher.getInstance("AES/ECB/PKCS5Padding")
```

3. Initializing the cipher engine for decryption

```
c.int(Cipher.DECRYPT_MODE, key)
```

4. Remove the padding and finish the decryption

```
byte[] plainText = c.doFinal(cipherTest);
```



Advantages & Disadvantages

Advantages

Algorithms are fast

- ***Encryption & decryption are handled by same key***
- ***As long as the key remains secret, the system also provide authentication***

Disadvantages

Key is revealed, the interceptors can decrypt all encrypted information

- ***Key distribution problem***
- ***Number of keys increases with the square of the number of people exchanging secret information***

Discussion