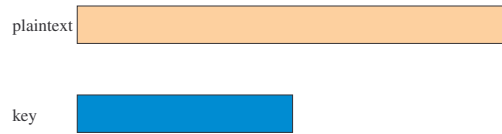


Symmetric Algorithms

- ∅ Stream ciphers: polyalphabetic
 - ∅ Work on message a bit or a byte at a time
 - ∅ Same bit/byte will encrypt differently, depending on the position of the key
- ∅ Block ciphers: polygram
 - ∅ Work on message block by block
 - ∅ Block size is usually the same as key size
 - ∅ Same plaintext block will always encrypt into the same ciphertext block
- ∅ A cryptographic mode combines basic cipher, some sort of feedback and some simple operations

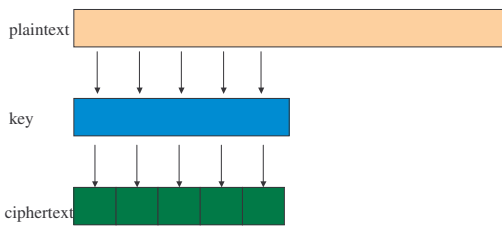
1

Stream Cipher Example



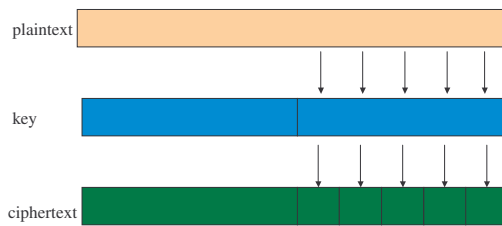
2

Stream Cipher Example



3

Stream Cipher Example



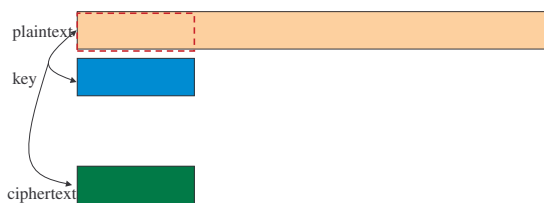
4

Block Cipher Example



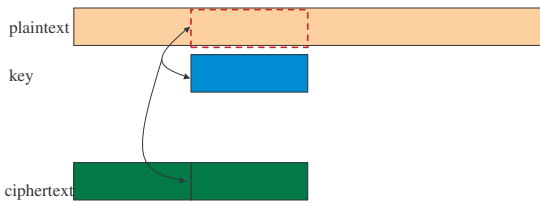
5

Block Cipher Example

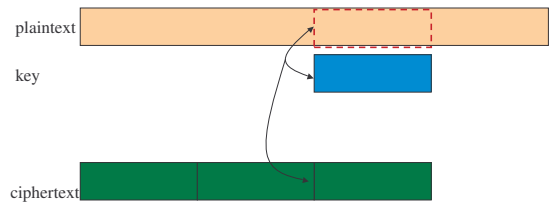


6

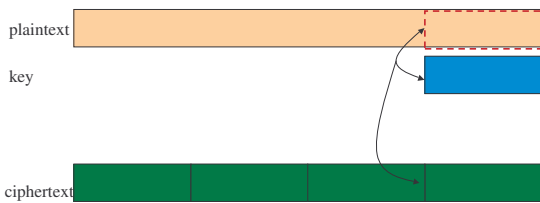
Block Cipher Example



Block Cipher Example



Block Cipher Example



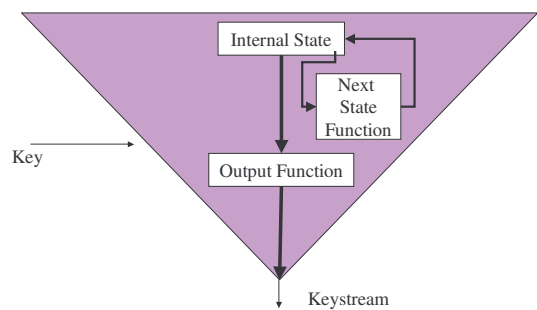
Requirements for Symmetric Algorithm

- ∅ Efficiency
- ∅ Patterns in plaintext should be concealed
- ∅ Loss or modification of one block/byte/bit should not invalidate the whole message

Stream Ciphers

- ∅ Work on one bit/byte at a time, XOR-ing it with key
- ∅ Security depends entirely on RNG generating the key
- ∅ Key should be pseudorandom – hard to break but easily reproduced for decryption
- ∅ If Eve can get hold of plaintext/ciphertext pair she can retrieve the key
- ∅ Keystream is generated continuously and is the function of the secret stored inside the RNG

Keystream Generator



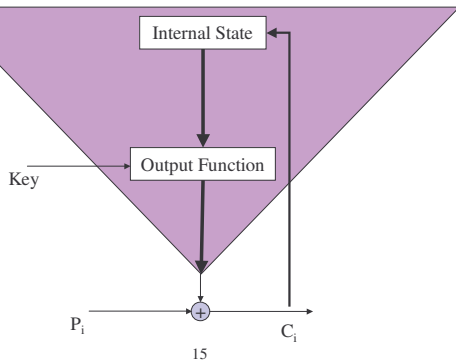
Synchronous Stream Cipher

- ∅ Keystream is generated from the key K
- ∅ Sender and receiver must be synchronized
- ∅ One-bit error in ciphertext produces one-bit error in plaintext
- ∅ Upon loss of synchronization both sides start afresh with a new key
- ∅ Any deletions and insertions will cause loss of synchronization
- ∅ Mallory can toggle/change bits

Self-Synchronizing Stream Cipher

- ∅ Internal state is the function only of the previous n ciphertext bits
- ∅ The function depends on the key K
- ∅ Decryption keystream generator will completely synchronize with encryption KG after receiving n bits
- ∅ Drawback:
 - ∅ Error extension – one-bit error in ciphertext produces n errors in plaintext
 - ∅ Mallory can also capture and replay any message if timestamps are not used

Self-Synchronizing Stream Cipher

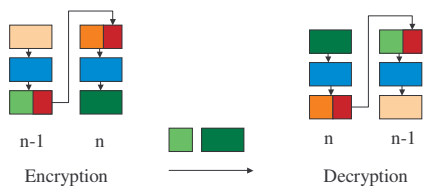


Electronic Code Book (ECB)

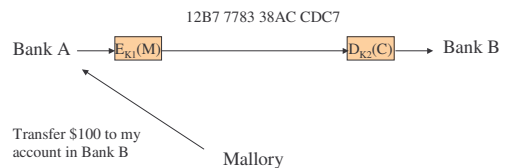
- ∅ Precompute and store mapping for every possible block
 - ∅ Fast encryption/decryption – just a table lookup
 - ∅ Ability to process text in any order and in parallel
 - ∅ Table size could be enormous even for 64 bit keys
 - ∅ Eve can detect which blocks map to other blocks, by seeing several plaintext and corresponding ciphertext messages
 - ∅ Due to language redundancy even partial decryption might provide enough information
 - ∅ Bit errors invalidate one block, bit losses are not recoverable

Dealing with Short Blocks

- ∅ Plaintext has to be **padding** to block boundary
 - ∅ Add a number of bytes with any content, last byte carries the padding length
- ∅ **Ciphertext stealing** (avoids transmission overhead)

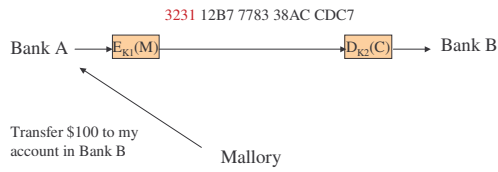


Block Replay



- ∅ Mallory does this couple of times, looks for similar block sequences.
- ∅ She can now replay 12B7 7783 38AC CDC7 at will

Block Replay

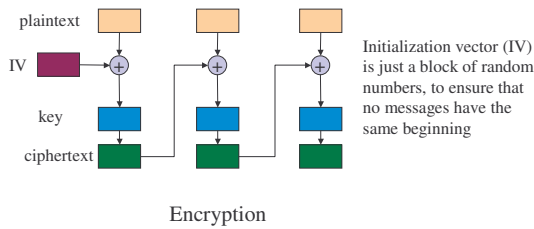


- ∅ Bank adds timestamps
- ∅ Mallory picks specific blocks of message carrying his name and account number and replaces those in other messages between Bank A and Bank B

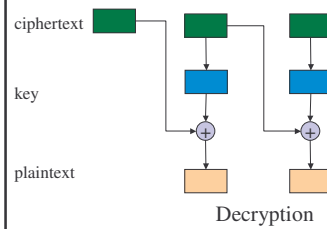
Cipher Block Chaining (CBC)

- ∅ Mallory can replace, add or drop blocks at will
- ∅ Chaining prevents this by adding feedback
- ∅ Each ciphertext block depends on all previous blocks

Cipher Block Chaining (CBC)

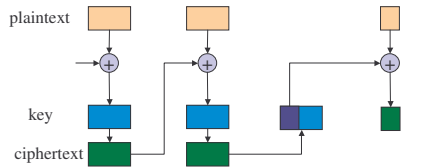


Cipher Block Chaining (CBC)



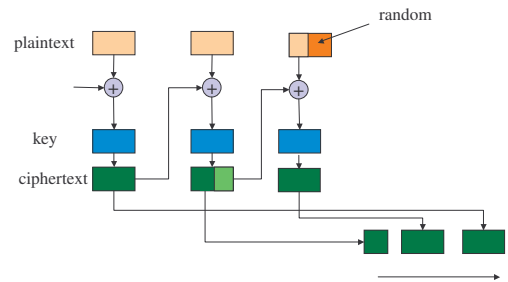
Dealing with Short Blocks

- ∅ Sometimes the ciphertext must be the same length as plaintext, we can't use padding
- ∅ Last plaintext block is of length j
- ∅ Encrypt last full plaintext block twice, take leftmost j bits and XOR with last block



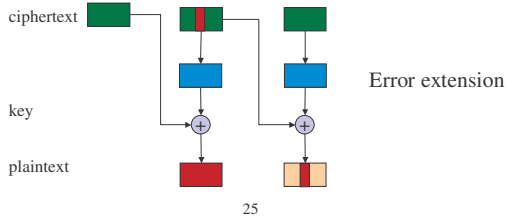
Dealing with Short Blocks

- ∅ Ciphertext stealing



Error Recovery

- ∅ An error in plaintext affects the rest of the message but are easily spotted and removed after decryption
- ∅ An error in ciphertext affects one block and several bits of plaintext

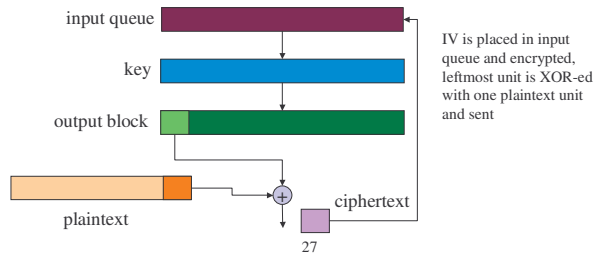


Potential Problems

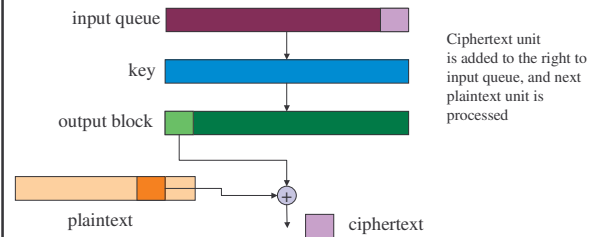
- ∅ Mallory can:
 - ∅ Add blocks
 - ∅ Drop blocks
 - ∅ Introduce bit errors

Cipher-Feedback Mode (CFB)

- ∅ Self-Synchronizing Block Cipher
- ∅ Enables encryption one unit at a time, where unit is smaller than the block



Cipher-Feedback Mode (CFB)

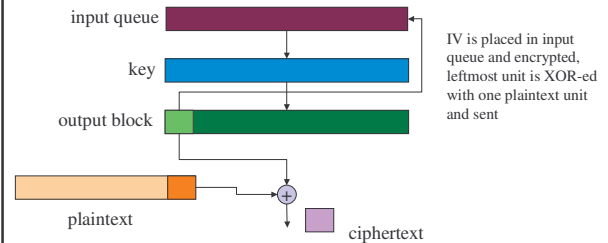


Cipher-Feedback Mode (CFB)

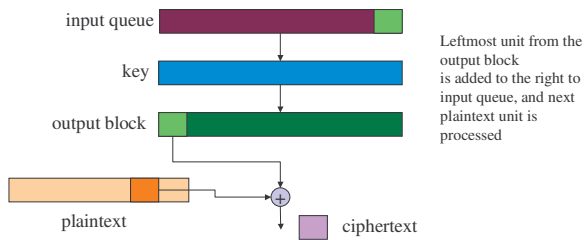
- ∅ IV must be unique, otherwise it opens vulnerability
- ∅ Drawback:
 - ∅ Error extension – one-bit error in ciphertext produces one-bit error in plaintext and $m/n-1$ subsequent plaintext units are garbled (m is the block size, n is the unit size)

Output-Feedback Mode (OFB)

- ∅ Similar to CFB but unit is taken from the output queue, not from the ciphertext



Output-Feedback Mode (OFB)



Output-Feedback Mode (OFB)

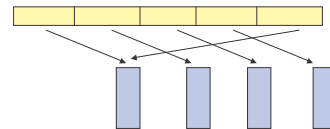
- ∅ Most of the work can be done offline, plaintext is then just XOR-ed when it arrives
- ∅ One-bit error in ciphertext produces one-bit error in plaintext
- ∅ Synchronization errors are not recoverable, must be detected and sender/receiver input queues synchronized

Which Mode is the Best?

- ∅ Stream ciphers can be analysed mathematically and can be efficiently implemented in hardware
- ∅ Block ciphers are more general and can be efficiently implemented in software
- ∅ ECB is easiest and fastest but also weakest. Can be used for encrypting random data, such as other keys.
- ∅ CBC is good for encrypting files, no danger of lack of synchronization
- ∅ CFB is good for encrypting streams of characters
- ∅ OFB is good if error propagation cannot be tolerated

Interleaving

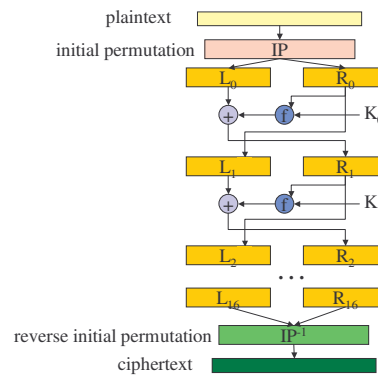
- ∅ All chaining ciphers depend on encryption results of previous blocks – this hinders parallel processing
- ∅ The trick is to divide plaintext stream into several parallel streams and feed it into several encryption processors



Data Encryption Standard (DES)

- ∅ Block cipher, symmetric algorithm
- ∅ Encrypts data in 64-bit blocks
- ∅ Key length 56-bits
- ∅ Combination of substitution and transposition – **round**
- ∅ Round is repeated on same block 16 times

DES Overview



Initial Permutation

- ∅ Traverses the plaintext in steps 8-bits wide and recombines bits

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**,

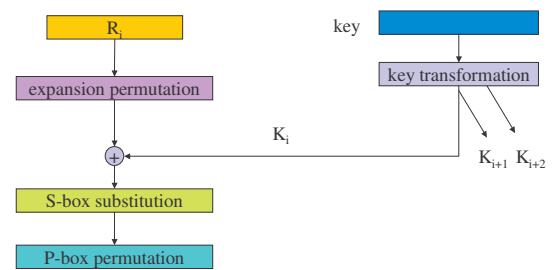
61, 53, 45, 37, 29, 21, 13, **5**, 63, 55, 47, 39, 31, 23, 15, **7**

- ∅ It does not affect security, just makes it easier to load data in byte-sized pieces on the chip

- ∅ Many software implementations leave out IP

37

Encryption Function (f)



38

Key Transformation

- ∅ 64-bit key K is reduced to 56-bit key by ignoring every 8-th bit (these can be used for parity check)
- ∅ 16 different 48-bit **subkeys** ($K_0 \dots K_{15}$) are generated from K
 - ∅ K is divided in two halves, each 26-bit long
 - ∅ Halves are shifted 1 or 2 bits left
 - ∅ 48 out of 56 bits are selected (compression permutation)
- ∅ This effectively generates new key for each round

39

Expansion Permutation

- ∅ Expands R_i from 32 to 48 bits
 - ∅ Makes the block length equal to key length
 - ∅ Provides redundancy that can be compressed later
 - ∅ Main purpose is to make every bit of ciphertext depend on every bit of plaintext

40

S-Box Substitution

- ∅ Critical operation for security – non-linear
- ∅ Performed by 8 boxes, each has 6-bit input and 4-bit output
- ∅ Bits 2,3,4,5 are substituted
- ∅ Bits 1 and 6 choose one out of four substitution functions

41

P-Box Permutation

- ∅ Recombines 32-bit output of S-box
- ∅ No bits are added or deleted

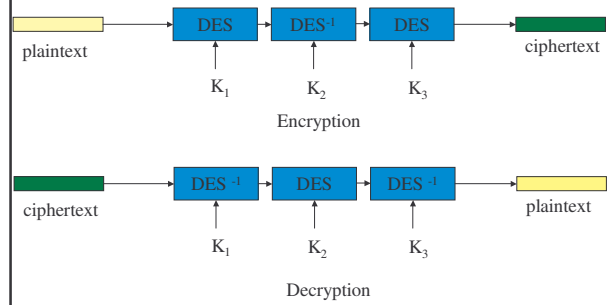
42

More on DES

- ∅ Decryption is same as encryption, just keys have to be fed in reverse
- ∅ DES can be used in ECB, CBC, OFB and CFB modes (think of the whole DES as performing encryption step)
- ∅ There are some weak keys that generate less than 16 different keys in key transformation step
- ∅ Also if K_1 is a complement of K , and P_1 is a complement of P , then C_1 will be a complement of C
 - ∅ This means that attackers must test only half of the keys

43

Triple DES



44

Combining ciphers

- ∅ Several encryptions can be performed to increase strength
- ∅ With multiple different keys

45