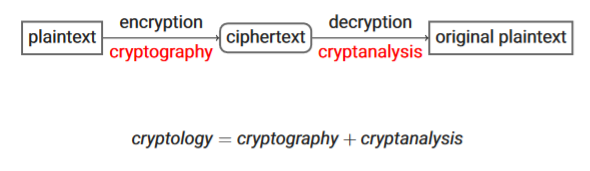
Symmetric Cryptography

Recap



(Dis)advantages of symmetric cryptography

Advantages

* Fast to compute

Disadvantages

* Key distribution is difficult
* Non-repudiation is NOT given
* Lots of keys needed
* When shared key is used, it must be regenerated when someone leaves the group

Basic techniques

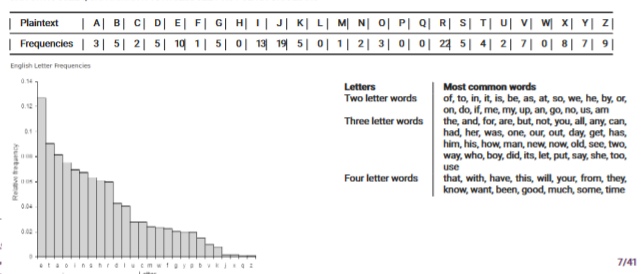
Three basic techniques used:

1. Substitution: exchange characters of the alphabet
2. Transposition: exchange the position of characters, by units of plaintext (block-by-block)
3. One-time-pad (OTP): combine each bit or character of the alphabet with the corresponding bit or character from the key using modular addition

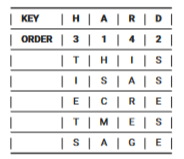
Substitution

Caesar cipher: simple shift of letters 🡪 Substitution: instead of shift, randomise characters, e.g.,



Weakness: can be cracked by frequency analysis (letters, two-letter words, three-letter words)

Transposition

Most famous example is columnar transposition:

* Pick a key
* Arrange the plaintext in a matrix and pad with a character, e.g., X
* Read the columns in alphabetical order from top to bottom
* Examples
  + Key: hard
  + Plaintext: this is a secret message
  + Ciphertext: HSCMAIAREGTIETSSSESE

Weakness:

* Does not change letter frequency, indicating that it might be a transposition
* Via bigram (2-letter combinations) frequencies it is possible to reconstruct the order of the columns

One-Time-Pad (OTP)

OTP takes a plaintext P, a key K, that is the same length as P generated uniformly at random and produces a ciphertext as: , Decryption as:

Examples:

* Encryption:
  + Plaintext: 1010
  + Key: 1100
  + Ciphertext: 0110
* Decryption:
  + Ciphertext: 0110
  + Key: 1100
  + Plaintext: 1010

Strength of OTP: Perfect secrecy: even with unlimited computing power, an attacker cannot learn anything about the plaintext other than the length, because of XOR every bit has a probability ½

Weakness:

* Key must be truly random
* Keys cannot be reused, or even part of keys
* Need a lot of key material

Modern day crypto

* Modern day security of cypher should only rely on secrecy of key, not on secrecy of cipher
* Kerckhoff’s principles:
  + Ciphers must arise from public competition (AES, SHA3, Salsa20)
  + Non-public ciphers must be distrusted
  + Do not trust constants in ciphers, that are not explained (Dual\_EC\_DRBG)

Block ciphers

Algorithms

Encryption: the encryption algorithm E takes a key K and a plaintext block P and produces ciphertext C: C = E(K, P)

Decryption: the decryption algorithm D, in the inverse of the encryption algorithm, takes a key K and the ciphertext C and produces the plaintext block P: P = D(K, C)

Construction

Block ciphers are not gigantic algorithms, but are a repetition of rounds, a short sequence of operations that is weak on its own, but not in big numbers

Two main techniques:

* Substitution-permutation networks (as in AES)
* Feistel schemes (as in DES)

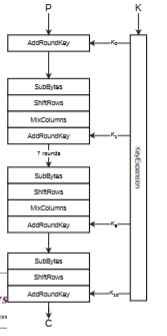
Substitution-permutation networks

Goal of networks is about confusion and diffusion

Confusion:

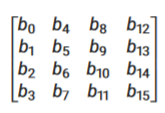
* Each bit of the ciphertext should depend on several parts of the key, e.g., if a single bit in a key is changed, the calculation of most or all of the bits in the ciphertext will be affected
* Goal: hide the relationship between the ciphertext and the key
* Means: Substitution

Diffusion:

* If we change a single bit of the plaintext, then about half of the bits in the ciphertext should change. The transformation depends equally on all bits of the input
* Goal: hide the relationship between the ciphertext and plaintext, e.g., hide any patterns that might exist in the plaintext
* Means: Permutation

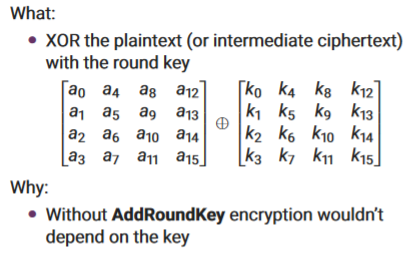
Advanced Encryption Standard (AES)

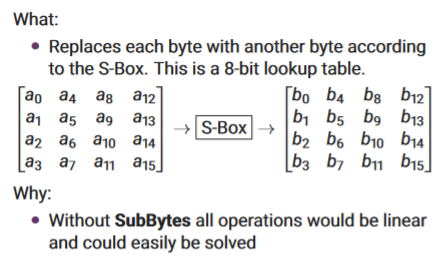
* Uses block of 128 bits
* Secret key either 128, 192 or 256 bits
* Internal state represented by 16 bytes as a two-dimensional array

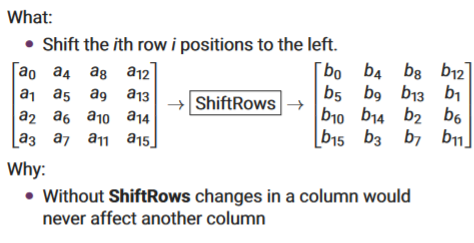


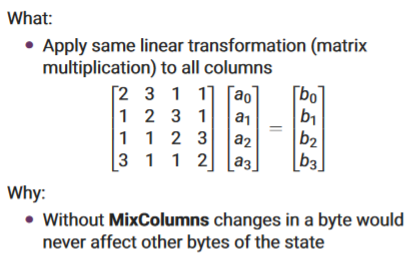
Key expansion:

* What:
  + Key is expanded to multiple round keys
  + Every round uses its own round key
  + At the first step, the key K0 is the original key
* Why:
  + Without it all rounds would depend on the same key, vulnerable to slide attacks

AddRoundKey:

SubBytes:

ShiftRows:

MixCollumns:

Decryption of AES:

* Apply operations in reverse
* Use the inverse functions

Block cipher – Modes

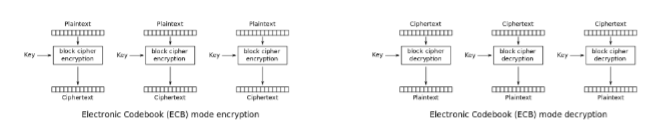
Electronic Code Book (ECB)

Pros:

* Easy to implement
* Easy to parallelize

Cons:

* Completely insecure
* Some plaintext block creates same ciphertext

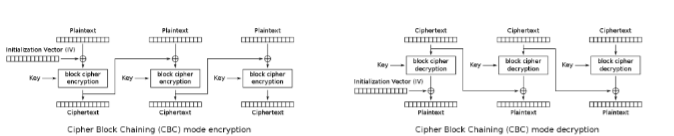


Cipher Block Chaining (CBC)

Pros:

* Initialization vector (IV) makes sure that the same plaintext is encrypted to different ciphertext
* Making ciphertext Ci dependent on ciphertext Ci-1 makes sure that same plaintext is encrypted to different ciphertexts

Cons:

* Encryption is slow, because has to wait until previous block is finished

Security protocols

What are security protocols

* Communication in the form of “A 🡪 B : m” which means Alice sends message m to Bob
* A sequence of such messages is intended to achieve a security goal
  + Confidentiality, integrity, authenticity, nonrepudiation
* After every step of the protocol the beliefs of the participants change
* When something goes wrong, the protocol is aborted

Attacker model

* There is an attacker Eve who tries to undermine the goal of protocol
  + Capabilities follow the Dolev-Yao model. Attacker capabilities are
    - Read
    - Delete
    - Copy
    - Rebuild
  + Attacker cannot break encryption (with unknown key) or hashes

Protocol for confidentiality

Assume Alice and Bob share a secret key , which is user for symmetric encryption. Confidential exchange of a message m vie protocol: A 🡪 B : {m}; achieves confidentiality

Protocol for integrity

A 🡪 B : {m} does not guarantee integrity, because attacker could change one or multiple bits in the message. If Bob now tries to decrypt the message it might still make sense, but Bob has no way of verifying this

Suppose Alice and Bob wish to be really sure that what Bob receives has been sent by Alice. They use: A 🡪 B : m,{m}

Protocol for confidentiality and integrity

Obvious combinations:

* A 🡪 B : {m}, {{m}}
* A 🡪 B : {m,{m}}

This does not work with OTP encryption

Should use two different keys, one for confidentiality and one for integrity

Authentication via shared secret

It is quite common to use a shared secret for authentication (Passwords, PIN Codes, …). Problem: in every authentication the secret is used in the clear

Authentication by challenge-response

Achieve authentication without using a secret in the clear

* Idea: send a riddle that can only be solved with a secret key
* The riddle must be fresh upon every reuse
* Typically, freshness is achieved via a nonce: a number used once
  + Range of numbers is relevant
  + Must be random / unpredictable
* Other means are
  + Timestamps
    - Require reliable / secure / synchronised clocks
  + Sequence numbers
    - Are predictable
    - Can wrap around
    - Need to keep track of state

Protocols for authentication

Basic protocol: A 🡪 B : A, ( is the fresh nonce)

At this stage, Alice knows she is talking to Bob, because only Bob has the shared key and can thus compute ,

Other variations:

* A 🡪 B : A, {}
* B 🡪 A :
* -----------
* A 🡪 B :
* B 🡪 A : + 1}

Seite 66