Network and Computer Security

Summary

Contents

Chapter 1

Cryptography

It is a widespread and dangerous belief that encrypting everything provides protection against anything. In reality, when the algorithm is known it is necessary to prevent several types of attacks, such as ciphertext-only, known-plaintext, chosen-plaintext or chosen-ciphertext. Cryptography must, then, protect the information against unauthorized insertion of information, modification of information in transit, replay of information and access to information.

1.1 Cryptographic Services

To this end, we need the following cryptographic services:

- Confidentiality: a service used to keep the content of the information from all, but those entities authorized to have it. Has the drawbacks of making debugging harder and may lead to information loss if the key is lost. Can be assured through symmetric of assymetric cipher.
- Integrity: a service that detects data manipulation by unauthorized entities (not the same thing as error detection codes). An intruder should not be able to substitute a false message for a legitimate one. Can be assured through MIC or a digital signature.
- Authenticity: a service used to ascertain the identity or the origin of a message. Can be assured through MIC or a digital signature, tough freshness requires adding a nonce to the message.
  - Entity authentication: verify the identity of an entity
  - Data origin authentication: confirm the creator of the message
  - Non-repudiation: a service which prevents an entity from denying. Can be assured by a digital signature. previous commitments or actions

1.2 Cryptographic Building Blocks

1.2.1 Symmetric Cipher

Uses the same key to cipher and decipher. In cryptographic function notation:

- $E(M, K)$: cipher message M with key K
- $D(C, K)$: decipher cryptogram C with key K

1.2.2 Asymmetric Cipher

A public/private pair of keys is used (KU/KR). In cryptographic function notation:

- $AE(M, KR)$: cipher message M with private key KR
- $AD(C, KU)$: decipher cryptogram C with public key KU

It is also possible to cipher with the public key and then decipher with the private key.

1.2.3 Cryptographic Hash

A cryptographic hash function does not use a key, instead receives an input message and returns a digest of the data. In cryptographic function notation:

- $H(M)$: hash of message M

The digest value is deterministic, has fixed size, a unique representation, is non-reversible and sensible to input changes.

1.2.4 Composite Building Blocks

- Hybrid cipher: a random symmetric key is generated and used to cipher the message. This key is then ciphered with the public key of the receiver and is sent along with the ciphered message. The receiver must decipher the symmetric key with its private key and use it do decipher the message
- Message integrity code: the MIC is created by creating the digest of the message and ciphering the result. It can then be used to check the integrity of the message by generating a new hash and comparing to the sent one.
- Digital signature: a digital signature is created by ciphering a digest with the private key. The receiver can then decipher with the public key and verify the identity of the sender.

1.3 Symmetric Criptography

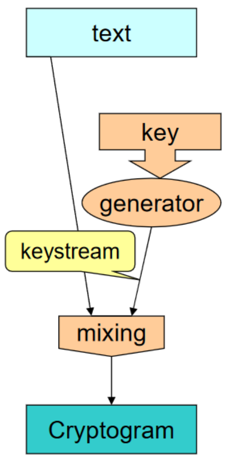
Symmetric ciphers use a single secret key that may be shared by 2 or more communicating parties. This allows confidentiality to all those who have the key and message authentication, with good performance. The main problem lies in key distribution

For N communicating parties, $N \times (N - 1)/2$ keys are needed for them to be able to communicate 1 - 1 secretly.

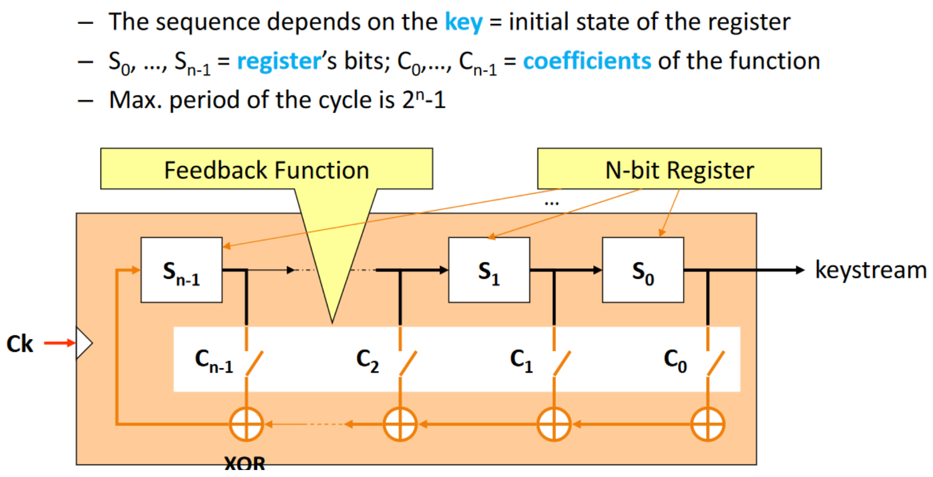
An example of a symmetric cipher is the one-time pad, where the message is XOR'd with the key. The security is based on the assumption that the key is never reused.

1.3.1 Symmetric Stream Ciphers

Symmetric stream ciphers are pratical approximations to the one-time pad. Keystreams are generated in a deterministic way from a fixed size key (approximation to real random sequence generators).



In this type of cipher, if the plain text is known, the keystream is exposed. The secure pseudo-random generators are sometimes based on LFSRs (Linear Feedback Shift Registers). It consists on a state machine that produces a cyclic sequence of bits:



- The sequence depends on the **key** = initial state of the register
- S_0, …, S_{n-1} = **register**'s bits; C_0,…, C_{n-1} = **coefficients** of the function
- Max. period of the cycle is 2^n-1

1.3.2 Symmetric Block Ciphers

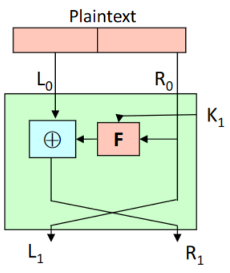
These are also based on approximations, using Shannon's notions of confusion (repeated application of a complex function to a large block) and diffusion (permutation, substitution, expansion and compression).

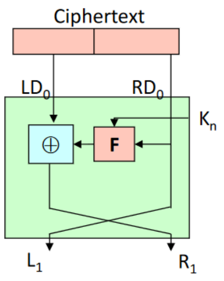
Feistel Network

The feistel network is a complex function most commonly used in block cipher algorithms. It applies a round function F over multiple rounds:

- Each round uses a different round key (K_i), obtained from the key (K)
- Text is split in left (L) and right (R) parts

The cipher and decipher processes are the same, with the keys used in the inverse order.





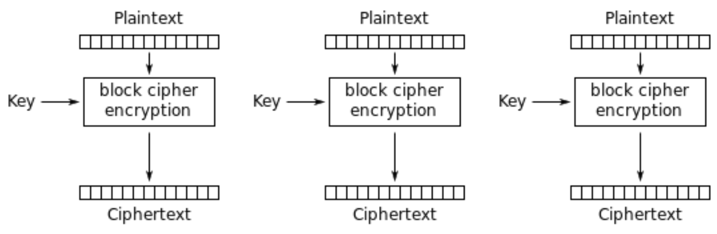
DES

The DES algorithm uses 64 bit blocks with 56 bit keys. It applies the feistel network algorithm on several rounds, after an initial permutation of the 64 bits, according to a fixed table.

Block Cipher Modes

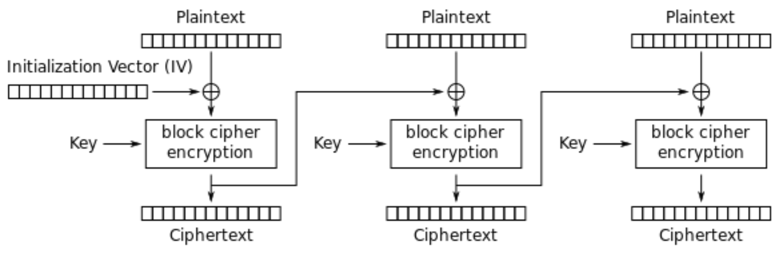
When using a plaintext of different size than the block, a cipher mode must be used.

- *Electronic Code Book* (ECB)
  - Encryption using independent blocks
  - Weakness in reproducing patterns from the original text; identical blocks produce the same ciphertext



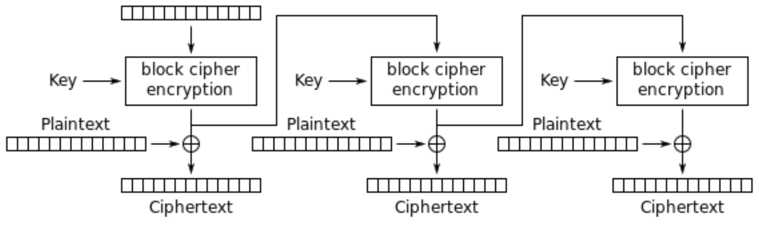
Electronic Codebook (ECB) mode encryption

- *Cipher Block Chaining* (CBC)
  - Plaintext is XORed with the ciphertext of the previous block before encryption
  - Reduces the risk of pattern replication
  - Uses an initialization vector in the first block (necessary for decryption) and requires padding: bits to compose entire blocks of the size required by the algorithm



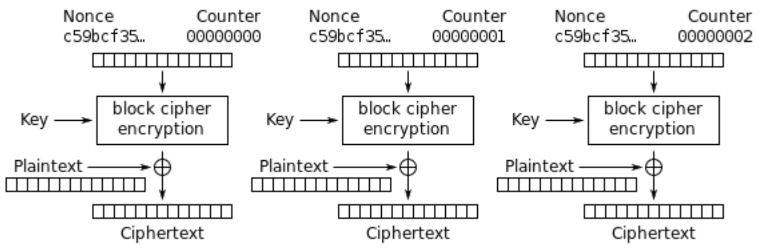
Cipher Block Chaining (CBC) mode encryption

- *Output Feedback* (OFB)
  - Retains the advantages of CFB, and the ciphertext is not used in the next block, allowing block cipher operations to be performed in advance, enabling parallel XOR as soon as the text (plaintext or ciphertext) is available



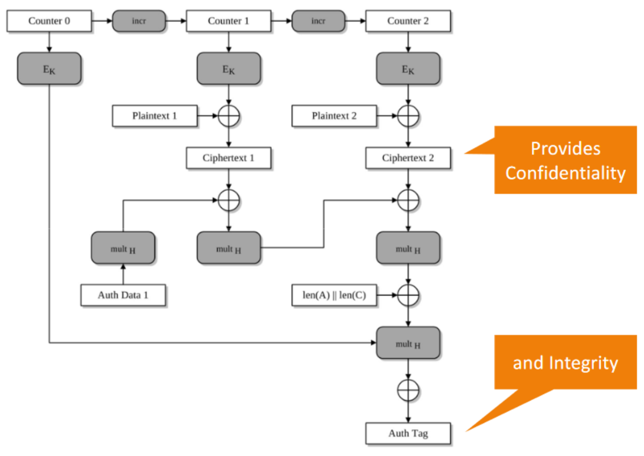
Output Feedback (OFB) mode encryption

- *Counter Mode* (CTR)
  - Standard cipher mode in AES
  - Uses a nonce and a counter, which must be different in each cipher operation



Counter (CTR) mode encryption

- *Galois Counter Mode* (GCM)



Padding

Sometimes it may be necessary to pad the last block, when the message size is not divisible by the block length. This may be done by:

- Padding with zero (null) bytes, spaces (0x20), all bytes of the same value
- Padding with random bits
- Padding with 0x80 (1000 0000) followed by zero (null) characters
- Padding with the PKCS#5 scheme: Sequence of bytes, each of which equal to the number of padding bytes (e.g. if 24 bits of padding need to be added, the padding string is "03 03 03": 3 bytes times 8 bits equals 24)

AES

AES is the current encryption standard. Supports keys of 128, 192 and 256 bits with 128 bit blocks. It runs in 10, 12 or 14 rounds, in which it substitutes bytes, shifts rows, mixes columns and adds the round key XOR state with key material)

1.4 Hash Functions

These are cryptagraphic functions, but not ciphers. They must be:

- Collision resistance: Computationally infeasible to find two inputs that give the same hash
- Preimage resistance: Given a hash, it's computationally infeasible to find an input that produces that hash
- Seconde preimage resistant: Given a hash value and the corresponding input, it's computationally infeasible to find a second input that generates that same hash

The most common algorithms are:

- MD5 (128 bits): very weak
- SHA-1 (160 bits): weak
- SHA-2 (256 to 512 bits)
- SHA-3 (256 to 512 bits)

Attacks on hash functions are sometimes done by brute force ($P(collision) = 2^{m-1}$) and most often through the birthday attack (pick M, M', M'', M'''... and obtain hashes until any 2 are identical) ($P(collision) = 2^{m/2}$).

1.4.1 Message Integrity Codes

The objective is to detect changes to a message. It allows the checking of a message's integrity and, with freshness can provide authenticity, thus, it's sometimes called MAC.

Under the assumption that the sender and recipient have a shared secret key K, the idea is to send the messgae and the MAC; if the message (or the MAC) is modified by an attacker, the recipient will be able to detect it. Attacker cannot create a valid MIC because he does not have K.

Implementations of MAC could be as follows:

- Hash the message and encrypt the digest
- Using a keyed-function (CMAC, usually with CBC)
- Using a keyed-hash (HMAC)

1.5 Asymmetric Ciphers

Asymmetric cryptography uses a pair of public/private keys. It allows for confidentiality, authentication and integrity (digital signatures). Has the advantage of only needing N key pairs, but the performance is much worse.

For confidentiality, the public key is used to cipher and the private to decipher. For authentication it's the opposite.

1.5.1 One-way Functions

Also known as trapdoor functions. The idea is for the function to be easy to compute in one direction, but hard to invert. These are usually used in asymmetric cryptography.

1.5.2 Algorithms

RSA

In this algorithm, the plaintext is divided into blocks, which are treated as a number. The keys are generated as follows:

- Choose two prime numbers p and q
- Define $n = pq$ and $z = \phi(n) = (p-1)(q-1)$
- Choose $e < n$ such that e is coprime with z
- Calculate d such that $ed \mod z = 1$
- The public key is $K_u = (e, n)$, and the private key is $K_r = (d, n)$

To encrypt, compute:

$$E(K_u, m) = m^e \mod n = c$$

And to decrypt:

$$D(K_r, c) = c^d \mod n = m$$

RSA is often used to produce signatures. To sign M we calculate $S = (hash(M))^d \mod n$ and to verify we check $hash(M) == S^e \mod n$. Only the owner of the private key can sign, but anyone with the public key can verify the signature.

ECC

Elliptic curve cryptography offers the same security as RSA with smaller bit sizes. In this case, the "hard" problem is the elliptic curve logarithm:

- $Q = k \cdot P$, where Q, P belong to an elliptic curve (e.g. $y^2 = x^3 - 3x + b \mod p$)
- Easy to compute Q, given k and P
- Hard to find k, given Q and P
- P is a base point (parameter of the curve)

1.5.3 Digital Signatures

Digital signatures use an asymmetric cipher and a hash function. The basic algorithm is as follows:

- Sign: $S(doc) = E(K_r, hash(doc))$
- Validate: $D(K_u, S(doc)) == hash(doc)$

Today often ECDSA is adopted (with elliptic curve encryption).