Crittografia

```
Encrypt(Plain Text, Keys) = Encrypted Text
Decrypt(Encrypted Text, Keys) = Plain Text
```

La crittografia si occupa di sviluppare schemi, metodi, algoritmi per la codifica sicura dei messaggi. La *crittoanalisi* studia gli attacchi che tentano di violare la crittografia attraverso lo studio di messaggi crittati, decifrando singoli messaggi, scoprendo debolezze di un algoritmo crittografico o dell'ambiente che lo esegue, scoprendo la chiave o tentando di dedurre caratteristiche dei messaggi criptati senza necessariamente decifrarli.

Bruteforce attack: tentare tutte le possibili chiavi

Problemi "risolti" dall'encryption

Consider the steps involved in sending messages from a sender, S, to a recipient, R. If S entrusts the message to T, who then delivers it to R, T then becomes the transmission medium. If an outsider, O, wants to access the message (to read, change, or even destroy it), we call O an *interceptor* or intruder. Any time after S transmits the message via T, it is **vulnerable** to exploitation, and O might try to access it in any of the following ways:

- **block** it, by preventing its reaching R, thereby affecting the availability of the message
- **intercept** it, by reading or listening to the message, thereby affecting the confidentiality of the message
- **modify** it, by seizing the message and changing it in some way, affecting the message's integrity
- **fabricate** an authentic-looking message, arranging for it to be delivered as if it came from S, thereby also affecting the integrity of the message

Attacco a messaggi criptati

Lo schema di attacco può variare in base alle informazioni disponibili all'attaccante:

- `cyphertext only`, solo testo cifrato
- `known-plaintext`, testo in chiaro (anche parziale) per un messaggio campione
- `chosen-cyphertext`, testo cifrato e algoritmo
- Coppie `<plaintext-ciphertext>` per un insieme di messaggi crittati con la stessa chiave

Gli algoritmi "degni di fiducia" sono quelli

- basati su matematica consolidata
- analizzati da esperti
- che hanno superato la "prova del tempo"
- la cui complessità temporale è commisurata alla sicurezza necessariamente
- che non impongono requisiti/limitazioni alle chiavi/testo da crittare
- trasparenti all'utente/che forniscono un processo semplice per l'utilizzo.

Algoritmi crittografici

Algoritmi simmetrici a chiave segreta

Si usa la stessa chiave per la decodifica e codifica. Garantisce: - **confidenzialità** solo chi possiede la chiave può decifrare - **integrità** non si può manomettere un messaggio criptato - **autenticazione e non ripudio** solo chi conosce la chiave può aver scritto quel messaggio

Non è possibile garantire nessun requisito indipendentemente dagli altri.

Per n individui sono necessiarie $n(n-1)/2$ chiavi.

DES: Data Encryption Standard

Basato su confusione e diffusione dell'informazione, codifica i messaggi in bocchi da 64 bit applicando 16 volte una funzione conmbinatoria, usando ogni volta come chiave uno dei parametri di F.

La versione originale usa una chiave da 56 bit (+8 parity check), consiste in operazioni combinatorie/logiche facilmente implementabili efficientemente in hw/sw.

Fa le stesse operazioni per codificare e decodificare i dati.

Violabile con forza bruta (2^{56} chiavi totali)

AES: Advanced Encryption Standard

Basato su trasformazioni (sostituzioni, scorrimento, mescolamento bit) applicabili a blocchi di 128 bit. Utilizza chiavi da 128, 192 o 256 bit (estendibile oltre) in un numero di cicli tra 10 e 14 (estendibile).

Algoritmi asimmetrici a chiave pubblica

Esiste una coppia di chiavi `<privateKey, publicKey`, in cui quella pubblica permette di cifrare un messaggio, che potrà essere poi decriptato solo con la chiave privata. `n individui, n coppie di chiavi`.

Garantiscono

- **Confidenzialità**, il messaggio è decodificabile solo da chi è in possesso della `privateKey`
- **Integrità**
- **Autenticazione e non ripudio**, solo chi possiede `privateKey` può produrre messaggi crittati con `privateKey`

È possibile gestire ogni requisito indipendentemente dagli altri.

RSA

Si generano chiave prima e privata come risultati di funzioni di due numeri primi molto grandi scelti casualmente* (60~200 cifre ognuno). Chiave pubblica e privata sono intercambiabili:

```
decrypt(k_priv, encrypt(k_pub, p)) = p
decrypt(k_pub, encrypt(k_priv, p)) = p
```

todo

Checksum

A checksum is a small-sized datum derived from a block of digital data for the purpose of detecting errors which may have been introduced during its transmission or storage. It is usually applied to an installation file after it is received from the download server. By themselves, checksums are often used to verify data integrity but are not relied upon to verify data authenticity.

The actual procedure which yields the checksum from a data input is called a checksum function or checksum algorithm. Depending on its design goals, a good checksum algorithm will usually output a significantly different value, even for small changes made to the input. This is especially true of cryptographic hash functions, which may be used to detect many data corruption errors and verify overall data integrity; if the computed checksum for the current data input matches the stored value of a previously computed checksum, there is a very high probability the data has not been accidentally altered or corrupted.

Checksum functions are related to hash functions, fingerprints, randomization functions, and cryptographic hash functions. However, each of those concepts has different applications and therefore different design goals. For instance a function returning the start of a string can provide a hash appropriate for some applications but will never be a suitable checksum. Checksums are used as cryptographic primitives in larger authentication algorithms. For cryptographic systems with these two specific design goals, see HMAC.

Check digits and parity bits are special cases of checksums, appropriate for small blocks of data (such as Social Security numbers, bank account numbers, computer words, single bytes, etc.). Some error-correcting codes are based on special checksums which not only detect common errors but also allow the original data to be recovered in certain cases.

Cryptographic hash function (Message digest)

A cryptographic hash function is a special class of hash function that has certain properties which make it suitable for use in cryptography. It is a mathematical algorithm that maps data of arbitrary size to a bit string of a fixed size (a hash function) which is designed to also be a one-way function, that is, a function which is infeasible to invert. The only way to recreate the input data from an ideal cryptographic hash function's output is to attempt a brute-force search of possible inputs to see if they produce a match, or use a rainbow table of matched hashes. Bruce Schneier has called one-way hash functions "the workhorses of modern cryptography".The input data is often called the message, and the output (the hash value or hash) is often called the message digest or simply the digest.

The ideal cryptographic hash function has five main properties:

- it is **deterministic** so the same message always results in the same hash
- it is quick to compute the hash value for any given message
- it is **infeasible to generate a message from its hash value** except by trying all possible messages
- a small change to a message should **change** the hash value **so extensively** that the new hash value appears uncorrelated with the old hash value
- it is infeasible to find two different messages with the same hash value

Cryptographic hash functions have many information-security applications, notably in digital signatures, message authentication codes (MACs), and other forms of authentication. They can also be used as ordinary hash functions, to index data in hash tables, for fingerprinting, to detect duplicate data or uniquely identify files, and as checksums to detect accidental data corruption. Indeed, in information-security contexts, cryptographic hash values are sometimes called (digital) fingerprints, checksums, or just hash values, even though all these terms stand for more general functions with rather different properties and purposes.

Applications:

- Verifying the **integrity** of files or messages
- Password verification
- Proof-of-work
- File or data identifier
- Pseudorandom generation and key derivation

MD5

Applica una trasformazione combinatoria complessa a blocchi di 512 bit del messaggio e genera un output a 128 bit.

> It is conjectured that is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest. (RFC 1231)