# 1 Cybersecurity essentials

## 1.1 Definizioni di sicurezza

La **sicurezza informatica** è l'insieme dei servizi, delle regole organizzative e dei comportamenti individuali che proteggono i sistemi informatici di un'azienda. Ha il compito di proteggere le risorse da accessi indesiderati, garantire la riservatezza delle informazioni, assicurare il funzionamento e la disponibilità dei servizi a fronte di eventi imprevedibili.

## 1.2 Proprietà di sicurezza

**Autenticazione** Il servizio di **autenticazione** si preoccupa dell'autenticità della comunicazione. Vengono definiti due tipi di autenticazione: **autenticazione delle controparti** e **autenticazione dell'origine dei dati**.

**Controllo degli accessi** Il controllo degli accessi è l'abilità di limitare e controllare l'accesso ai sistemi e alle applicazioni tramite canali di comunicazione. Ogni entità interessata ad accedere ad un servizio deve prima essere autenticata.

**Confidenzialità** La confidenzialità è l'atto di proteggere i dati trasmessi dagli attacchi passivi.

**Integrità dei dati** Il principio di integrità è applicabile ad un flusso di dati

# 2 Crittografia

## 2.1 Crittografia simmetrica

Crittografia con chiave comune e unica, a basso carico di elaborazione.

### 2.1.1 Algoritmi di crittografia simmetrica

**DES** Chiave a 56 bit + 8 di parità, se applicato 3 volte viene detto 3DES. 2DES è vulnerabile a un attacco di tipo known-plaintext detto meet in the middle.

**IDEA** Chiave a 128 bit, blocco dati da 64 bit, utilizza XOR, addizione mod16 e moltiplicazione $\mathrm{mod} 2^{16} + 1$.

**RC2, RC4** Più veloci di DES, con chiave a lunghezza variabile e blocchi da 64 bit.

### 2.1.2 Applicazione algoritmi a blocchi

**ECB (Electronic Code Book)** Per cifrare dati in quantità superiore. Ogni blocco viene cifrato con lo stesso algoritmo separatamente. Sconsigliato perché cifra allo stesso modo blocchi identici.

**CBC (Cipher Block Chaining)** Per cifrare dati in quantità superiore. Richiede IV, XOR tra blocco cifrato precedente e blocco da cifrare, poi applicazione algoritmo di cifratura.

**Padding** Per cifrare dati in quantità inferiore. Aggiungo bit per riempire lo spazio vuoto. Alcuni tipi offrono controllo d'integrità, applicando padding a tutti i blocchi.

**CTS (Cipher Text Stealing)** Per cifrare dati in quantità inferiore. L'ultimo blocco è riempito con byte del penultimo blocco, questi due blocchi vengono scambiati durante la cifratura.

**CTR (Counter mode)** Per cifrare dati in quantità inferiore. Accesso random al testo cifrato, usa algoritmo a blocchi per cifrare n bit alla volta

### 2.1.3 Algoritmi stream

Operano su un flusso di dati senza richiederne la divisione in blocchi, tipicamente su un bit o byte.

**Salsa20 e ChaCha20** Chiavi da 128 o 256 bit. Operazione base: add-rotate-xor su 32 bit. Effettuano 20 volte mixing su input.

### 2.1.4 Distribuzione chiavi

Per una comunicazione privata tra $n$ persone occorrono $\frac{n(n-1)}{2}$ chiavi. Avviene tramite algoritmi per scambio chiavi.

## 2.2 Crittografia asimmetrica

Le chiavi sono diverse e hanno funzionalità reciproca. È possibile generare un messaggio segreto per uno specifico destinatario conoscendone solo la chiave pubblica.

### 2.2.1 Algoritmi a chiave pubblica

**DSA** Elevamento a potenza e logaritmo del risultato, utilizzato solo per firma digitale.

**RSA** Può solo cifrare dati il cui valore sia inferiore al modulo pubblico. Funzionamento:

1. modulo pubblico $n = pq$, con $p$ e $q$ primi, grandi e segreti
2. $\phi = (p-1)(q-1)$
3. esponente pubblico $e$ tale che $1 < e < \phi$, $e$ coprimo $\phi$
4. esponente privato: $d = e^{-1}\phi$
5. chiave pubblica: $(n, e)$, chiave privata: $(n, d)$

Solitamente le chiavi pubbliche hanno un $e$ che contiene solo due bit a 1 per ottimizzare le prestazioni.

RSA è debole se vengono utilizzati esponenti piccoli, stesse chiavi per firma e cifratura. Per renderlo più forte, aggiungere sempre del padding fresco prima di cifrare il messaggio e non firmare dati grezzi.

### 2.2.2 Distribuzione chaivi per crittografia asimmetrica

**Diffie-Hellman** Sfrutta la difficoltà di risoluzione del problema dell'algoritmo discreto.

**Curve ellittiche** Problema del logaritmo discreto sulla curva, più complesso, permette di avere chiavi più corte. Firma digitale: ECDSA, key agreement: ECDH, key distribution: ECIES.

## 2.3 Funzioni di hash e digest

### 2.3.1 Digest

È un riassunto a lunghezza fissa del messaggio da proteggere. Deve essere veloce da calcolare, difficile da invertire e non generare troppe collisioni (digest uguali). Un algoritmo di digest a $n$ bit è insicuro quando vengono generati più di $2^{\frac{n}{2}}$ digest perché si ha una probabilità di collisione pari al 50%.

### 2.3.2 Funzioni di hash

Dividono il messaggio in blocchi e applicano la funzione base per ottenere il valore di hash.