Università degli studi di Perugia
Facoltà di Ingegneria

Appunti di Sicurezza Informatica

Burani Alessio

Anno Accademico 2014/2015

Indice

Capitolo 1

Introduzione alla sicurezza informatica

1.1 Premessa

1.1.1 Definizione

Che cosa significa sicuro? La parola sicurezza e l'aggettivo sicuro vengono sempre associati a beni che si desidera proteggere da possibili danni, danneggiamenti, perdita, e via dicendo. In particolare un bene è al sicuro se è ben protetto, e la sua messa in sicurezza non deve impedirne l'utilizzo. Esistono molteplici definizioni di sicurezza informatica, ad esempio:

- Security is the degree of protection against danger, damage, loss, and criminal activity [**?**].
- Security is a form of protection where a separation is created between the assets and the threat [**?**].

La parola sicurezza deriva dal latino sine cura: senza preoccupazione. La sicurezza di un sistema può essere definita come la conoscenza del fatto che l'evoluzione del sistema non produrrà stati indesiderati. Le cause che possono minare la sicurezza sono molteplici e spesso non prevedibili, quindi non si può parlare di sicurezza in senso assoluto, ma solo relativo (*L'unico computer sicuro è un computer spento*).

Trasversalità della tematica: Le problematiche di sicurezza interessano molteplici campi (attività lavorative, vita domestica, hobby, giochi, sport, etc.). Di fatto, ogni settore della vita moderna ha delle implicazioni relative alla sicurezza. Il livello di sicurezza di un'organizzazione dipende dai livelli di sicurezza di tutti i suoi comparti/settori. Il livello di sicurezza di un sistema è determinato dal livello di sicurezza dal suo comparto meno sicuro (principio dell'anello debole).

1.2 Concetti fondamentali

1.2.1 Sistema informativo e informatico

- Per sistema informativo (information system) di un'organizzazione si intende l'insieme delle informazioni prodotte ed elaborate e delle risorse umane, materiali e immateriali, coinvolte nel processo di elaborazione di tali informazioni
- Per sistema informatico (information and communication technology system) s'intende l'insieme delle varie tecnologie coinvolte nel sistema informativo (il sistema informativo è parte del sistema informatico).

Questo corso verterà sulla sicurezza dei sistemi informativi. Tuttavia, si approfondiranno maggiormente questioni inerenti la sicurezza dei sistemi informatici. Per sicurezza di un sistema

informativo si intende il grado di protezione contro qualsiasi minaccia ai suoi asset. Richiede il soddisfacimento dei seguenti requisiti:

- **confidenzialità, integrità e disponibilità**
- **assicurazione, autenticità e anonimato**

Quando si parla di **attacco** solitamente si intende la violazione di uno o più di questi requisiti.

1.2.2 Vulnerabilità, Minacce, Attacchi, Difesa

Vulnerabilità

Una vulnerabilità (o falla o breccia) è una **debolezza intrinseca** di un sistema, che potrebbe essere sfruttata per provocare perdite o danni. Scaturisce spesso da errate procedure di sicurezza e/o da errori di progettazione/implementazione, e in alcuni casi è intimamente legata alla natura del sistema. Ad esempio:

- un sistema potrebbe essere vulnerabile alla manipolazione non autorizzata dei dati causa un bug nella procedura di autenticazione dell'utente
- un calcolatore è vulnerabile all'acqua

Nel primo caso (vulnerabilità scaturite da errate procedure di sicurezza) l'insorgenza delle vulnerabilità può essere mitigata adottando adeguati standard e norme di qualità.

Minacce (Threats)

Per minaccia (threat) ad un sistema informatico/informativo si intende quell'insieme di circostanze che potrebbero arrecare danni ai suoi asset: eventi potenziali, accidentali o deliberati, che, nel caso accadessero, produrrebbero perdite e danni. il realizzarsi di una minaccia generalmente avviene sfruttando una o più vulnerabilità del sistema. Si parla quindi di situazioni ipotetiche che potrebbero avvenire in determinate circostanze. Ad esempio:

- esecuzione di codice malevole che invia dati sensibili ad un'organizzazione criminale
- accesso a dati riservati da parte di entità non autorizzate
- perdita di dati a causa della rottura di un apparato hardware o al crash di uno specifico software

Attacchi (Attacks)

Un attacco (attack) è un atto deliberato teso ad arrecare un danno al sistema. Consiste, di fatto, nella realizzazione di una **minaccia**. Generalmente, un attacco viene perpetrato attraverso lo sfruttamento di una o più vulnerabilità. Spesso si classificano in base all'entità del danno:

- **attacco attivo (active attack)**: altera le risorse o ne modifica il processo di gestione/elaborazione
- **attacco passivo (passive attack)**: ottiene le informazioni/dati senza alterarli e senza modificare il relativo processo di gestione/elaborazione

Un'altra importante classificazione è in base al luogo da cui viene iniziato l'attacco:

- **attacco dall'interno (inside attack)**: attacco iniziato da un'entità all'interno del perimetro di sicurezza di un sistema informativo di una data organizzazione
- **attacco dall'esterno (outside attack)**: attacco iniziato da un'entità all'esterno del perimetro di sicurezza

Ovviamente, è molto più difficile prevenire e rilevare gli attacchi interni di quelli esterni. Ciò anche a causa del fatto che le misure di prevenzione per questo tipo di attacchi limita notevolmente l'usabilità del sistema (si pensi, ad esempio, alla struttura gerachica in ambiente militare, in cui ogni risorsa conosce il minimo indispensabile per svolgere i propri compiti. In questo modo nel caso la risorsa venga compromessa, si limita il danno. Ovviamente tutto ciò rallenta il processo di funzionamento del sistema).

Tecniche di difesa

Diverse contromisure (o misure protettive) possono essere attuate per proteggere un sistema informativo da eventi accidentali e da attacchi deliberati. Tali misure devono essere strutturate all'interno di un piano di sicurezza redatto dopo un'attenta analisi costi/benefici (textitcost-effective solutions). Le tecniche di difesa possono essere di tipo:

- **preventivo**: effettuano una serie di **controlli** per evitare **a priori** che attacchi noti o immaginabili possano essere sferrati con successo (e.g. controlli aeroportuali, controllo di accessi e permessi negli OS).
- **a posteriori**: sono tese a ridurre gli effetti di un attacco che è riuscito a eludere le misure preventive di cui sopra; devono monitorare un sistema ed essere in grado di **rivelare** comportamenti anomali.

Un **meccanismo di sicurezza** è un qualsiasi metodo, strumento, o procedura teso a rilevare, prevenire o porre rimedio agli effetti di un attacco alla sicurezza del sistema. La strategia di difesa, qualunque essa sia, combina in modo opportuno uno o più meccanismi di sicurezza. molti meccanismi di sicurezza consistono in controlli hardware/software.

1.2.3 Sicurezza informatica: definizione classica e requisiti CIA

La sicurezza informatica si fonda sulla protezione dei seguenti macro-requisiti di un sistema informativo (informatico):

- **Confidenzialità (Confidentiality)**
- **Integrità (Integrity)**
- **Disponibilità (Availability)**

Spesso si utilizza l'acronimo C.I.A. per denotarli in modo compatto. Ovviamente, come si può

Figura 1.1: Diagramma di Venn dei macro-requisiti di sicurezza

notare dal diagramma di Venn in Figura 1.1 spesso vengono richiesti contemporaneamente più di questi requisiti, cosi come un attacco può violare più requisiti.

Confidenzialità (Confidentiality)

Def. **Confidentiality**: the property that information is not made available or disclosed to unauthorized individuals, entities, or processes. Per confidenzialità si intende quindi la garanzia che alle risorse informatiche accedano solo le parti autorizzate ad accedervi. E' talvolta denominata segretezza, riservatezza o privacy. Nota bene: per accesso non si intende solo la lettura, ma anche la visualizzazione, la stampa o la semplice consapevolezza dell'esistenza di una data risorsa nel sistema.

Attacchi alla confidenzialità: Si ha un attacco alla confidenzialità quando una entità (persona, processo o risorsa) tenta di accedere senza autorizzazione a informazioni protette (mentre queste sono memorizzate, oppure durante l'elaborazione, oppure ancora durante una comunicazione). La protezione della confidenzialità avviene utilizzando in modo appropriato i seguenti strumenti (meccanismi) di sicurezza informatica:

- **Cifratura (Encryption)**
- **Controllo degli accessi (Access Control)**
- **Sicurezza fisica (Physical security)**

Integrità (Integrity)

Def. **Integrity**: safeguarding the accuracy and completeness of information and processing methods.
Def. **Integrity**: the property of safeguarding the accuracy and completeness of assets.

Per integrità si intende quindi la garanzia che le risorse possano essere modificate solo dalle parti autorizzate e solo nei modi prestabiliti. Le modifiche comprendono la scrittura, la variazione, il cambiamento dello stato, l'eliminazione e la creazione. Nel caso di file, conviene includere anche i metadati associati (proprietario, ultimo utente ad averlo letto, data ultima modifica,data di creazione), in modo che un accesso non autorizzato al contenuto possa essere rivelato da un controllo di integrità applicato ai metadati.

Attacchi all'integrità: Si ha un attacco all'integrità quando una entità (persona, processo o risorsa) tenta di modificare senza autorizzazione una o più risorse del sistema informativo.

La protezione della confidenzialità avviene utilizzando in modo appropriato i seguenti strumenti (meccanismi) di sicurezza informatica (tutti basati su un uso corretto della ridondanza):

- **Backup**
- **Somma di controllo o Checksum**
- **Codici a correzione di errore (Corruzioni non deliberate ma accidentali, possono essere facilmente elusi da attacchi intelligenti)**
- **Codici di autenticazione dei messaggi o Message Authentication Code MAC**

Disponibilità (Availability)

Def. **Availability**: ensuring that authorized users have access to information and associated assets when required.

Per disponibilità (availability) si intende quindi che le risorse siano accessibili, nei tempi e nei modi prestabiliti, alle parti autorizzate ogni volta che le richiedono: se una persona o un sistema dispone dei diritti di accesso ad una risorsa, l'accesso non deve essergli impedito. Spesso la disponibilità viene citata tramite il suo opposto: la **negazione di servizio** (**Denial of Service** o **DoS**). La disponibilità può assumere significati/sfumature diverse; una risorsa può trovarsi in uno stato intermedio tra i due opposti stati di piena disponibilità e di piena indisponibilità.

Conflittualità requisiti CIA

Spesso i requisiti di confidenzialità, integrità e disponibilità possono essere in conflitto tra loro. E' quindi importante trovare il compromesso ottimo per la garanzia di tutti e tre. E' facile garantire la confidenzialità di una risorsa impedendo a chiunque di accedervi (e.g. se chiudo la risorsa in un blocco di cemento, sicuramente resterà confidenziale. Tuttavia la disponibilità della stessa verrebbe compromessa in senso assoluto).

1.2.4 Requisiti AAA

In aggiunta ai concetti CIA, sovente viene richiesto il soddisfacimento di ulteriori requisiti che vanno sotto l'acronimo A.A.A.

Assicurazione (Assurance)

Per assicurazione (assurance) si intende come viene fornita e gestita la fiducia reciproca tra le varie parti coinvolte nel sistema informativo. Tale requisito sopperisce la mancanza di regolamentazione, da parte dei requisiti C.I.A., dell'uso delle risorse di un sistema. E' teso quindi ad evitare un uso non consono dei vari asset del sistema. Tale concetto è ovviamente bidirezionale: così come il sistema deve fidarsi degli utenti (del fatto che non lo usino in modo inappropriato), gli utenti devono fidarsi del sistema (e.g.:trattamento dei dati personali). Il concetto di fiducia è tuttavia difficile da quantificare ed è legato al grado di confidenza sul comportamento che ci si attende dal sistema. Il requisito di assicurazione viene regolato agendo sui seguenti strumenti:

- **Politiche (Policies):** specifiche comportamentali che regolano l'operato degli attori del sistema
- **Permessi (Permissions):** descrivono le operazioni ammesse/concesse e quelle proibite
- **Protezioni (Protections):** implementazione dei meccanismi di sicurezza tesi ad applicare e far rispettare le politiche e i permessi di cui sopra
- **Qualità del software:** lo sviluppo del software che rispetta standard di qualità rigorosi rende più difficile che il sistema si discosti dai comportamenti attesi

Autenticità (Authenticity)

L'autenticità è la capacità di provare che dichiarazioni, politiche e autorizzazioni rilasciate da persone o sistemi siano veritiere. Se non è garantita persone/sistemi possono sostenere argomentazioni non vere senza essere contraddetti da prove oggettive. Se è garantita, si dice anche che è soddisfatto il requisito del textbfnon-ripudio (non-ripudiation). Per non-ripudio si intende che dichiarazioni autentiche rilasciate da persone e/o sistemi NON possono essere negate. La firma digitale è il principale meccanismo crittografico che permette di ottenerlo. Un contesto in cui tale requisito è importante è, ad esempio, la PEC.

Anonimato (Anonymity)

Per anonimato (anonymity) si intende che non è possibile ottenere o risalire all'dentità di un individuo pur avendo accesso a determinati record/transazioni di un sistema informativo. Se un'organizzazione deve rendere pubblici alcuni dati dei suoi clienti/membri senza violarne la privacy, può adottare alcuni dei seguenti strumenti:

- **Aggregazione (Aggregation):** combinare mediante somme e/o medie i dati di diversi individui
- **Miscelazione (Mixing):** permutare i dati dei singoli utenti in modo da preservare l'esito di predeterminate query
- **Mediazione (Proxies):** utilizzare degli agenti fidati che si interpongono tra l'individuo e i sistemi con i quali interagisce

- **Pseudonimi (Pseudonyms):** utilizzare delle identità fittizie eventualmente autenticate da una terza entità che funge da garante

1.2.5 Tipologie di Minacce e Attacchi

E' possibile classificare diverse minacce e attacchi rispetto a quali requisiti violano. Vediamone alcune:

Intercettazione (Eavesdropping)

Si ha un'intercettazione quando un'entità non autorizzata ha ottenuto l'accesso ad una risorsa. Generalmente questo tipo di attacchi avviene nella fase di trasmissione dell'informazione, e rappresenta una violazione alla confidenzialità. Esempi:

- copia illecita di file di programma o dati
- intercettazione di una comunicazione telefonica
- sniffing di pacchetti di dati

Un'intercettazione potrebbe essere molto difficile da rilevare, poiché un intercettatore "silenzioso" potrebbe essere molto abile e non lasciare tracce o concellarle.

Alterazione (Alteration)

Si ha una alterazione o modifica quando un'entità non autorizzata, oltre ad accedere a una risorsa, interferisce con essa modificandola. Rappresenta una violazione alla confidenzialità. Esempi:

- attacchi di tipo **man-in-the-middle**: un flusso di dati viene intercettato, modificato e ritrasmesso (rappresenta anche una minaccia alla confidenzialità)
- virus informatici che modificano file di sistema critici in modo da eseguire azioni maliziose
- cambiare i valori di un database o modificare un programma in modo che esegua dei calcoli diversi

Alcuni casi di alterazione possono essere facilmente rilevati, mentre altre modifiche, più sottili, possono essere estremamente difficili da individuare.

Interruzione (Denial-of-service)

In un'interruzione, una risorsa del sistema viene degradata o eliminata, diventando non disponibile o inutilizzabile. Rappresenta una minaccia alla disponibilità e/o all'integrità. Esempi:

- la distruzione o il sabotaggio di un dispositivo
- la cancellazione di un file
- la congestione di un Web server causata da un numero enorme di richieste artificiali

Falsificazione (Masquerading)

La falsificazione consiste nella contraffazione di risorse (dati/hardware) da parti non autorizzate. Costituisce principalmente una violazione di autenticità, e a volte anche alla confidenzialità e/o all'integrità. Esempi:

- phishing: creazione di siti Web apparentemente identici agli originali allo scopo di frodarne gli utenti
- textbfspoofing: spedizione di pacchetti dati con falsi indirizzi di ritorno

A volte le falsificazioni sono facilmente rilevabili, ma se attuate con abilità potrebbero essere del tutto indistinguibili da normali operazioni reali legittime.

Ripudio (Ripudiation)

Il ripudio consiste nel negare di aver effettuato una data azione. Ll'azione potrebbe essere la ricezione o la spedizione di un messaggio, così come l'esecuzione di una transazione. Spesso, riguarda il tentativo di recedere da un contratto (accordo) precedentemente assunto in cui era comunque previsto l'uso di ricevute (o simili) per dimostrare l'esecuzione di determinate operazioni. Si tratta di un attacco all'assicurazione.

Inferenza (Inference), Correlation and traceback

Per inferenza o attacco inferenziale si intendono un insieme di tecniche che utilizzando la statistica e l'algebra permettono di ricavare/stimare **informazioni sensibili** a partire da dati non sensibili. Rappresenta un attacco alla confidenzialità, e spesso è attuato nel contesto dei databases. Similmente, per correlation and traceback si intendono un insieme di tecniche basate sulla statistica e sull'algebra che permettono di determinare la sorgente di una particolare informazione o di un particolare flusso di dati.

N.B.: C'è una differenza, in senso giuridico, tra dati personali e dati sensibili. Tale distinzione dipende dalla giurisdizione dei paesi. In particolare in Italia il dato personale viene indicato come un'informazione che permette di identificare un individuo (anagrafica), mentre un dato sensibile rappresenta un'informazione su aspetti della vita privata dell'individuo (orientamento sessuale, opinione politica, credo religioso, etc.)

1.2.6 Principi della Sicurezza Informatica

Mediazione completa (Complete mediation):

Ogni accesso ad una risorsa deve essere controllato verificando che sia conforme alle politiche di sicurezza stabilite; diffidare da miglioramenti nell'efficienza ottenuti salvando autorizzazioni precedentemente acquisite, poiché i permessi possono variare nel tempo.

Struttura aperta (Open design):

L'architettura, il progetto e l'implementazione dei meccanismi di sicurezza di un sistema devono essere resi pubblici.

- la sicurezza deve fondarsi sulla segretezza di pochi elementi chiave
- maggior feedback favoriscono l'individuazione di bug, falle e vulnerabilità, aumentando la robustezza e la sicurezza del sistema
- un meccanismo di protezione ritenuto sicuro da molti è preferibile ad uno noto solo a pochi. E' quindi bene evitare meccanismi di sicurezza basati sulla segretezza (security by obscurity).

Separazione dei privilegi (Separation of privilege):

Più condizioni dovrebbero essere richieste per concedere l'accesso a risorse limitate o ottenere il permesso di effettuare una data azione. In genere questo principio comporta una separazione logico/funzionale delle componenti di un sistema.

Minimo privilegio (Least privilege):

Ogni parte di un sistema deve avere i privilegi minimi necessari allo svolgimento dei propri compiti. Per attività inusuali che richiedono maggiori privilegi conviene assegnare autorizzazioni temporanee fortemente limitate nel tempo. In questo modo si riduce il rischio di attacchi basati sulla scalata di privilegi.

Minimo meccanismo comune (Least common mechanism):

I meccanismi di sicurezza che per l'accesso e la gestione di risorse condivise non dovrebbero essere a loro volta condivisi o dovrebbero essere condivisi il meno possibile. E' quindi buona norma adottare tecniche di isolamento quali la virtualizzazione e il sandboxing (e.g. browser web: nessuna applicazione web può agire sul filesystem, eccetto attraverso i cookies). In questo modo vengono mitigati rischi derivanti da comportamenti malevoli di utenti cui spetta comunque l'accesso a una data risorsa condivisa.

Usabilità (Usability, Psychological acceptability):

I meccanismi di sicurezza non devono rendere più difficile l'accesso alle risorse. Le interfacce utente devono essere ben progettate, intuitive e di facile utilizzo, mentre i parametri di configurazioni inerenti aspetti di sicurezza devono essere di semplice comprensione e facilmente modificabili.

Fattore lavoro (Work factor):

Il costo necessario ad aggirare un meccanismo di sicurezza deve essere confrontabile alle risorse di cui dispone un potenziale attaccante. Un meccanismo di sicurezza deve avere un livello di sofisticazione, e pertanto un costo, che tenga conto del valore degli asset da proteggere e delle risorse a disposizione di potenziali attaccanti.

Monitoraggio (Compromise recording):

A volte può convenire effettuare un monitoraggio dettagliato piuttosto che investire in sofisticati meccanismi di sicurezza di tipo preventivo.

Penetrazione più semplice:

un attaccante utilizzerà qualsiasi mezzo di penetrazione disponibile: non necessariamente i mezzi più ovvi (prevedibili), non necessariamente i mezzi per i quali sono state installate le difese più solide. (e.g.: E' inutile avere un sistema informatico sicuro se il personale non viene formato e fornisce a chiunque credenziali di accesso). L'applicazione di tale principio presenta le seguenti difficoltà:

- saper anticipare l'avversario, cioè riuscire a prevederlo
- gestire in modo equilibrato la sicurezza delle diverse parti di un sistema; rafforzare le difese di una parte può indurre gli avversari ad attaccare un'altra parte (più debole) del sistema.

Temporalità:

le risorse di un sistema informativo devono essere protette solo fino a quando possiedono un valore, e in modo proporzionale al loro valore. Generalmente tale principio si riferisce ai dati: il loro valore può subire brusche variazioni; si consideri ad esempio il valore dei dati sull'andamento dei mercati prima e dopo la loro divulgazione.

Anello più debole:

la sicurezza di un sistema articolato NON può essere più forte del suo anello più debole. La gestione della sicurezza deve tener conto del sistema nel suo insieme. Un elevato grado di sicurezza delle singole parti non implica un elevato grado di sicurezza globale, ma è necessario prevedere anche una strategia oculata di coordinamento della varie parti che non introduca vulnerabilità (l'insieme è più della somma delle parti).

1.3 Fondamenti di Crittografia

La parola Crittografia deriva dal greco, e significa scrittura segreta. La crittografia è quindi l'arte e la scienza dello scrivere in modo segreto, ovvero l'arte di offuscare le informazioni in modo incomprensibile prevedendo una tecnica segreta per ricostruirle in modo esatto. L'applicazione storica della crittografia è la protezione della confidenzialità dei messaggi. Con **crittografia classica** si intende, infatti, l'insieme di tecniche che sopperiscono la necessità di nascondere il contenuto di una comunicazione a soggetti non autorizzati. Con l'avvento dei calcolatori e del digitale, si è passati alla crittografia moderna, che presenta molte applicazioni aggiuntive di non immediata comprensione, ma estremamente utili:

- firma digitale
- controllo di integrità sicuro
- autenticazione

1.3.1 Terminologia e definizioni preliminari

- **Testo in chiaro (plaintext o cleartext):** messaggio nella sua forma originale
- **Testo cifrato (ciphertext):** messaggio cifrato (criptato o crittografato)
- **Cifratura o crittazione o criptazione (encryption):** processo che produce il testo cifrato a partire dal testo in chiaro
- **Decifratura o decrittazione o decriptazione (decryption):** processo inverso della cifratura
- **Crittografo (cryptographer):** esperto di crittografia
- **Crittoanalisi o crittanalisi (cryptanalysis):** l'arte o la scienza di violare testi cifrati; ricostruire il testo in chiaro senza disporre del segreto necessario alla fase di decifratura
- **Crittoanalista:** esperto di crittoanalisi
- **Crittologia (cryptology):** l'arte o la scienza delle scritture nascoste nella sua accezione pi'u generale; include la crittografia e la crittoanalisi
- **Chiave segreta:** la segretezza dell'elaborazione nell'algoritmo è di solito concentrata nella segretezza di una chiave. E' stata introdotta perché è difficile concepire ogni volta nuovi algoritmi di cifratura ed è difficile spiegare rapidamente il funzionamento di un nuovo algoritmo ad un soggetto con il quale si desidera comunicare in modo sicuro

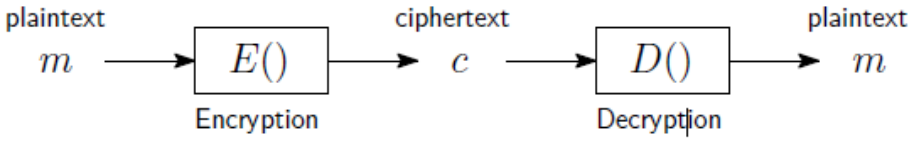


Figura 1.2: Schema a blocchi che illustra il processo cifratura-decifratura

1.3.2 Definizione formale di un sistema crittografico

: Formalmente, un sistema crittografico è costituito da sette componenti (a volte alcune di queste sono assenti o coincidono):

- l'insieme dei possibili input (plaintext)

- l'insieme dei possibili output (ciphertex)
- l'insieme delle possibili chiavi di cifratura
- l'insieme delle possibili chiavi di decifratura
- la corrispondenza tra chiavi di cifratura e di decifratura
- l'algoritmo di cifratura da usare
- l'algoritmo di decifratura da usare

1.3.3 Complessità computazionale

Uno schema crittografico anche assumendo che sia privo di vulnerabilità intrinseche, può essere (quasi) sempre sottoposto ad attacchi a forza bruta. Affinché ciò possa avvenire è necessario disporre di un test affidabile che permetta di dire se una chiave scelta a caso è quella corretta; cioè, essere in grado di riconoscere il testo in chiaro. Se tale precondizione è verificata si può perseguire un approccio esaustivo in cui si tentano tutte le chiavi. Uno schema crittografico è allora tanto più sicuro quanto è maggiore il costo computazionale necessario a violarlo. Contemporaneamente deve garantire l'efficienza di cifratura/decifratura. Formalizzando, possiamo definire un sistema **computazionalmente sicuro** se:

- il costo per rendere inefficace il cifrario supera il valore dell'informazione cifrata
- il tempo richiesto per rendere inefficace il cifrario supera l'arco temporale in cui l'informazione ha una qualche utilità

Tuttavia, stimare lo sforzo computazionale richiesto per effettuare con successo la crittoanalisi del testo cifrato è molto difficile. Il tempo richiesto per un approccio a forza bruta fornisce soltanto un limite superiore: è realistico solo se l'algoritmo non presenta delle debolezze intrinseche di tipo matematico; in questo caso si possono effettuare stime ragionevoli su tempi e costi. In un approccio a forza bruta mediamente si devono provare metà di tutte le possibili chiavi. Ovviamente in questo processo prende parte, come variabile fondamentale, la lunghezza della chiave. Alcuni schemi crittografici prevedono una chiave avente lunghezza variabile: aumentandola aumenta la sicurezza, ma diminuisce l'efficienza. Altri schemi stabiliscono a priori la lunghezza della chiave, se si desidera aumentarla è necessario sviluppare algoritmi simili che utilizzano chiavi di lunghezza diversa, oppure è possibile combinare in modo opportuno tali schemi ottenendo uno schema risultante con una chiave globale più lunga (attenzione al modo in cui si combinano!).

1.3.4 Algoritmo pubblico o segreto?

Ottenere la sicurezza dalla segretezza cioè custodendo gelosamente e nascondendo la metodologia di cifratura/decifratura, **security by obscurity**, è assolutamente da evitare. Questo perché:

- mantenere la segretezza è difficile, poiché le tecniche di reverse engineering permettono di risalire al codice
- è una strategia in chiara opposizione al fundamental tenet of cryptography

Oggigiorno, nelle applicazioni di uso civile/commerciale si usano schemi crittografici di dominio pubblico. La segretezza, laddove prevista, è conseguenza di altre cause (e.g. protezione del segreto industriale, contesti militari).

1.3.5 Attacchi a sistemi crittografici

Si possono distinguere cinque categorie di attacchi in base al tipo di informazioni in possesso dell'attaccante (di cui gli ultimi due meno frequenti):

Solo testo cifrato (ciphertext only)

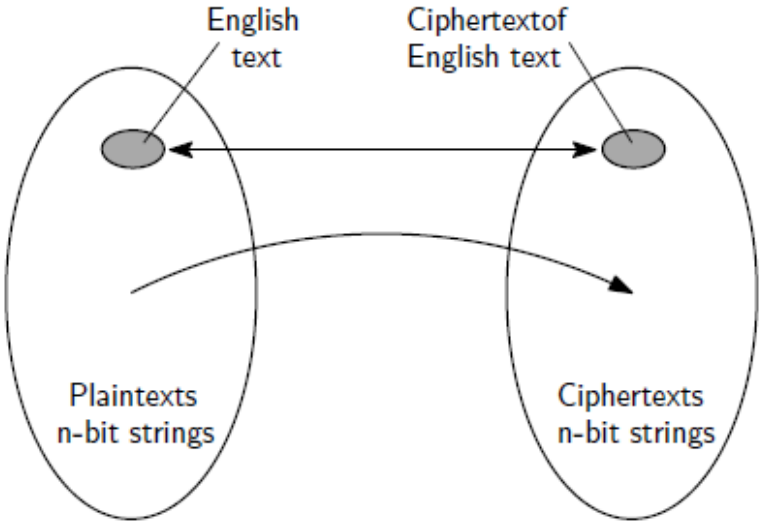
L'attaccante conosce l'algoritmo di cifratura e dispone soltanto di una certa quantità di testo cifrato. Nell'ipotesi che non vi siano vulnerabilità intrinseche nello schema crittografico, l'attaccante può tentare un attacco forza bruta, a patto che disponga di un test affidabile per il riconoscimento del testo in chiaro e di potenza di calcolo sufficiente. Tale attacco è anche noto come **testo in chiaro riconoscibile (recognizable plaintext)**. Riguardo la riconoscibilità del testo in chiaro valgono le seguenti osservazioni:

- nel caso di messaggi testuali, è molto improbabile che una chiave di decifratura errata permetta di ottenere un messaggio verosimile
- è essenziale avere una sufficiente quantit'a di testo in chiaro

Esempio: il messaggio m è una frase di un qualche linguaggio naturale. Ipotizzando che i caratteri di m siano codificati in ASCII a 8 bit e il testo cifrato c abbia la stessa lunghezza di m (quasi sempre è così). Siano:

- t: il numero di caratteri di m e di c
- $n = 8t$: il numero di bit di m e di c
- $2^{\alpha n}$: il numero totali di messaggi distinti, nel linguaggio naturale considerato, di lunghezza n bit

Si osservi che nominalmente esistono 2^n stringhe binarie di n bit, ma solo una piccolissima frazione di queste è la codifica di un messaggio nel linguaggio naturale, $0 < \alpha < 1$ è un fattore correttivo usato per modellare tale fenomeno, per la lingua inglese $\alpha = 0.16$.



Il seguente rapporto:

$$\frac{2^{\alpha n}}{2^n} = \frac{1}{2^{(1-\alpha)n}} \tag{1.1}$$

rappresenta la probabilità che un stringa random di n bit costituisca la codifica di un messaggio nel linguaggio naturale considerato. Se la chiave di decifratura ha una lunghezza di k bit, effettuando un attacco a forza bruta (2^k tentativi), il numero atteso di messaggi verosimili (nel linguaggio naturale dato) associabili al testo cifrato c è pari a:

$$\frac{2^k}{2^{(1-\alpha)n}} \tag{1.2}$$

Essendo k fissato, al crescere di n tale numero tende rapidamente a zero, quindi il test di riconoscibilità diventa quasi infallibile se si dispone di una sufficiente quantità di testo cifrato. In alcuni casi, si potrebbe tentare un approccio a forza bruta computazionalmente meno costoso se:

- la chiave di decifratura coincide con quella di cifratura, spesso è così
- la chiave di cifratura 'e derivata da una password di utente segreta con una procedura nota (si può tentare un approccio a forza bruta sullo spazio delle password, sensibilmente più piccolo dello spazio delle chiavi)

IMPORTANTE: uno schema crittografico deve SEMPRE essere sicuro contro attacchi di tipo **ciphertext only**, poiché il testo cifrato è sempre facilmente intercettabile e ottenibile.

Testo in chiaro conosciuto (known plaintext)

In questo caso, il crittoanalista conosce:

- l'algoritmo di cifratura
- un certo insieme di coppie $\langle plaintext, ciphertext \rangle$, ma non ha la facoltà di decidere lui il tipo specifico di plaintext
- una certa quantità di testo cifrato

Alcuni schemi crittografici potrebbero essere molto resistenti ad attacchi di tipo **ciphertext only** ma rivelarsi vulnerabili ad attacchi di tipo **known plaintext**. In caso di impiego, è fondamentale prevenire che un attaccante ottenga coppie $\langle plaintext, ciphertext \rangle$.

Testo in chiaro selezionato (chosen plaintext)

In questo caso, il crittoanalista conosce:

- l'algoritmo di cifratura
- una certa quantità di testo cifrato (con chiave segreta K)

e può scegliere a piacimento il testo in chiaro ed ottenere il relativo testo cifrato (sempre con la chiave K). In altre parole può scegliere quali coppie $\langle plaintext, ciphertext \rangle$ conoscere. E' possibile che un sistema crittografico sia sicuro contro attacchi di tipo **ciphertext only** e **known plaintext**, ma sia vulnerabile ad attacchi di tipo **chosen plaintext**.

Testo cifrato selezionato (chosen ciphertext)

L'attaccante dispone (oltre che dell'algoritmo di cifratura e del testo cifrato da decifrare) di testo cifrato, con significato, scelto da lui, e corrispondente testo in chiaro.

Testo selezionato (chosen text)

L'attaccante dispone sia di testo in chiaro scelto da lui (e corrispondente testo cifrato) sia di testo cifrato scelto da lui (e corrispondente testo in chiaro).

1.3.6 Tipi di funzioni crittografiche

- funzioni a chiave pubblica (public key functions): richiedono l'uso di due chiavi, una pubblica e una privata (approfodire non ripudio su firma digitale)
- funzioni a chiave segreta (secret key functions): richiedono l'uso di una singola chiave
- funzioni hash (hash functions): non usano alcuna chiave

Tipologia di attacco	Informazioni in possesso al crittoanalista
solo testo cifrato (*ciphertext only*)	• algoritmo di cifratura • testo cifrato da decifrare
testo in chiaro conosciuto (*known plaintext*)	• algoritmo di cifratura • testo cifrato da decifrare • uno o più testi in chiaro e corrispondenti testi cifrati
testo in chiaro selezionato (*chosen plaintext*)	• algoritmo di cifratura • testo cifrato da decifrare • testo in chiaro scelto dal crittoanalista e relativo testo cifrato
testo cifrato selezionato (*chosen ciphertext*)	• algoritmo di cifratura • testo cifrato da decifrare • testo cifrato, con significato, scelto dal crittoanalista e corrispondente testo in chiaro
testo selezionato (*chosen text*)	• algoritmo di cifratura • testo cifrato da decifrare • testo in chiaro scelto dal crittoanalista e relativo testo cifrato • testo cifrato, con significato, scelto dal crittoanalista e corrispondente testo in chiaro

Figura 1.3: Tabella riassuntiva tipologie di attacchi

Key Size (bits)	Number of Alternative Keys	Time required at 1 decryption/µs	Time required at 10^6 decryptions/µs
32	$2^{32} = 4.3 \times 10^9$	2^{31} µs = 35.8 minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	2^{55} µs = 1142 years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	2^{127} µs = 5.4×10^{24} years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	2^{167} µs = 5.9×10^{36} years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	2×10^{26} µs = 6.4×10^{12} years	6.4×10^6 years

Figura 1.4: Tempo medio per una ricerca esaustiva

1.4 Modelli di controllo degli accessi (Access control models)

Il controllo degli accessi alla varie risorse di un sistema informativo costituisce un fondamentale meccanismo di sicurezza di tipo **preventivo**. Previene attacchi alla confidenzialità, all'integrità e all'anonimato. L'idea di base è restringere l'accesso solo a coloro che hanno la necessità di accedere e/o modificare specifiche risorse, in accordo al principio del **minimo privilegio**.

1.4.1 Matrice di controllo degli accessi

Una matrice di controllo degli accessi (Access Control Matrix ACM) è una tabella che definisce i permessi di accesso dei vari **soggetti** di un sistema informativo ai suoi **oggetti**.

- un **soggetto** è un utente, un gruppo o una generica entità attiva che desidera effettuare una data azione su un data risorsa
- un **oggetto** è un file, un documento, un dispositivo, una generica risorsa o più in generale un'entità passiva sulla quale si desidera compiere una data azione

Ogni riga della tabella è associata ad un soggetto, e ogni colonna è associata ad un oggetto. Ogni cella stabilisce quindi il tipo di azione consentita; il soggetto e l'oggetto sono implicitamente definiti dalla cella. Diverse modalità di accesso sono possibili: lettura, scrittura, copia, esecuzione, cancellazione, annotazione. Una cella vuota stabilisce che non viene assegnato alcun tipo di accesso.

	/etc/passwd	/home/mario/	/admin/
root	read, write	read, write, exec	read, write, exec
pippo	read		
mario	read	read, write, exec	
backup	read	read, exec	read, exec
...	...	...	...

Figura 1.5: Esempio access control matrix

Pro e Contro ACM

L'adozione di una ACM offre, in linea di principio, i seguenti vantaggi:

- immediatezza nel valutare e modificare i diritti di accesso per una data coppia soggetto-oggetto
- facilità di visualizzazione e gestione: semplifica la vita all'amministratore

Purtroppo, il grande svantaggio delle ACM, che ne limita fortemente l'applicabilità, è la mancanza di scalabilità. E' del tutto ragionevole pensare che un computer server possa avere ordine di 10^3 soggetti (per lo più utenti) e 10^6 oggetti (files e directories), richiedendo una ACM di 10^9 celle. In questi ordini di grandezza i precedenti vantaggi decadono (non scalabilità).

1.4.2 Liste di controllo degli accessi (Access Control List ACL)

Una lista di controllo degli accessi (access control list ACL) segue un approccio di tipo object-centered per garantire una buona scalabilità, in particolare:

- ad ogni oggetto **o** viene associata un lista **L**, detta la lista di accesso di **o**, che enumera tutti i soggetti che hanno un qualche diritto di accesso ad **o**, e

- per ciascuno di tali soggetti, **s**, sono specificati i tipi di azioni che **s** può compiere su textbfo

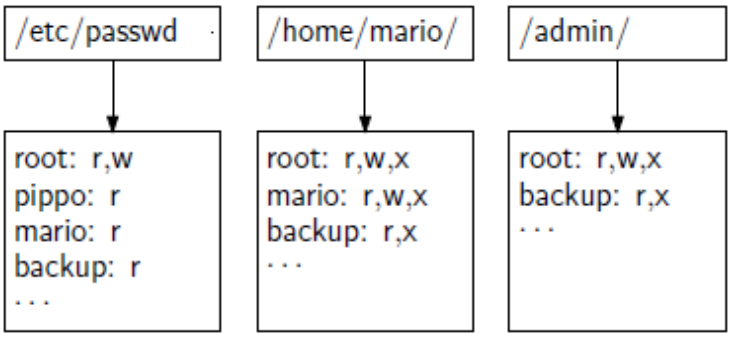


Figura 1.6: Esempio access control list

Nella pratica, il modello ACL comporta un significativo risparmio di memoria rispetto al modello ACM: ogni lista del modello ACL è ottenibile scandendo la relativa colonna del modello ACM ignorando però le celle vuote.

Pro e Contro ACL

Il principale vantaggio del modello ACL rispetto al modello ACM è la minor occupazione di memoria che si registra nella pratica (in teoria, nel caso peggiore l'occupazione è la medesima). La memoria occupata complessivamente nel modello ACL, $\sum_O size(L_O)$, è proporzionale a $C_{nv}(ACM)$ cioè al numero di celle non vuote della ACM. Nella pratica $C_{nv}(ACM)$ è molto minore di $C(ACM)$ (che denota il numero totale di celle della ACM); cioè nella pratica $C_{nv}(ACM) << C(ACM)$. Un altro vantaggio è la facilità di gestione delle ACL da parte del sistema operativo:

- sono incorporate nei metadati associati all'oggetto in questione (i.e. filesystem)
- per verificare i diritti di accesso ad un oggetto o non è necessario consultare una struttura dati centralizzata associata a tutti gli oggetti
- ma basta manipolare una struttura dati molto più snella, distribuita ed associata soltanto all'oggetto **o**

Il principale svantaggio delle ACL è che non consente di enumerare in modo efficiente i diritti di accesso di un dato soggetto. Tale operazione deve essere espletata ogni qual volta un utente viene rimosso da un sistema. I diritti di accesso di un soggetto **s** possono ottenersi soltanto effettuando una scansione di tutte le liste di accesso (associate a tutti gli oggetti **o**), e selezionando i diritti di accesso di **s** nelle liste in cui tale soggetto compare. Nel modello ACM, invece, ciò poteva banalmente ottenersi esaminando la riga della matrice relativa al soggetto **s**.

1.4.3 Controllod egli accessi basato su liste di capacità (Capabilities Access Control CAC)

Il modello basato sulle liste di capacità segue un approccio "subject-centered", complementare (ortogonale) a quello del modello ACL, per offrire una buona scalabilità. Ad ogni soggetto **s** viene associata una lista, detta **C-list** di **s**, contenente soltato gli oggetti per i quali **s** ha un qualche diritto di accesso. Per ciascun oggetto **o** della **C-list** di **s** viene specificato il tipo di azione che **s** può esercitare su **o**. Similmente al modello ACL, anche il modello CAC comporta un significativo risparmio di memoria rispetto al modello ACM nelle situazioni pratiche: ogni lista del modello CAC è ottenibile scandendo la relativa riga del modello ACM ignorando però le celle vuote.

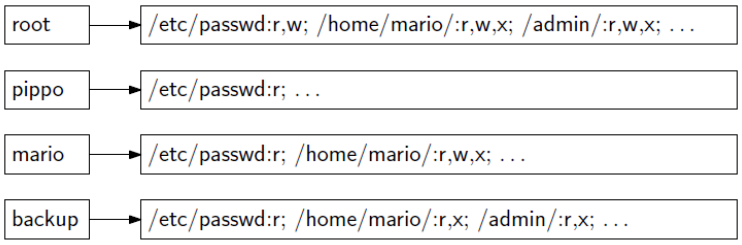


Figura 1.7: Esempio CAC

Pro e Contro CAC

Rispetto al modello ACM, anche il modello CAC richiede una minor occupazione di memoria nella pratica. la memoria occupata complessivamente nel modello CAC, $\sum_s size(L_s)$, è proporzionale a $C_{nv}(ACM)$, cioè al numero di celle non vuote della ACM (vedi considerazioni su ACL). Facilita inoltre il compito dell'amministratore di sistema nell'analizzare e gestire i diritti di accesso di un dato soggetto. Inoltre, l'autorizzazione ad accedere ad un dato oggetto o da parte di un soggetto **s** richiede tempi ragionevolmente brevi qualora la **C-list** di **s** abbia una dimensione contenuta.

Il principale svantaggio del modello CAC è che le **C-list** non sono direttamente associate agli oggetti. Ciò comporta che:

- non è possibile implementarle in modo distribuito incorporandole nei metadati degli oggetti
- l'unico modo per determinare chi e come può accedere ad un dato oggetto o è effettuare una ricerca su tutte le C-list (di tutti i soggetti); nel modello ACM tale operazione richiede semplicemente di esaminare la colonna relativa ad o (problema ortogonale a quello delle ACL)

1.4.4 Controllod egli accessi basato sui ruoli (Role-Based Access Control RBAC)

Per controllo degli accessi basato sui ruoli si intende che l'assegnazione dei diritti di accesso avviene in modo indiretto ed è funzione del ruolo attribuito ad un dato soggetto **s**. L'amministratore definisce i ruoli e specifica i diritti di accesso per tali ruoli, anziché quelli per i singoli soggetti (utenti). Ogni ruolo dovrebbe rappresentare una classe di soggetti (utenti) con la medesima mansione. I soggetti vanno assegnati ai vari ruoli coerentemente alle loro mansioni, e un soggetto può ricevere più ruoli dato che può ricoprire più mansioni. Definiti i ruoli, i diritti di accesso vanno assegnati secondo una logica ruolo-oggetto e NON soggetto-oggetto. I diritti di accesso di un dato soggetto sono l'unione dei diritti di accesso dei suoi ruoli. Pertanto:

- textbfil modello RBAC non 'e alternativo ai modelli ACM, ACL e CAC, ma si pone ad un livello di astrazione superiore: RBAC deve necessariamente sfruttare un modello per il controllo degli accessi, basato su **ACM**,**ACL** o **CAC**, ove i soggetti saranno però sostituiti dai ruoli.
- RBAC deve inoltre mantenere/gestire l'elenco dei ruoli associati ai vari soggetti

Architettura RBAC

Il modello RBAC 'e realizzabile utilizzando un generico framework per la gestione del controllo degli accessi (ACM, ACL, CAC), ove i soggetti sono, come già detto, sostituiti con i ruoli, più un layer superiore che gestisce l'elenco dei ruoli associati ai vari soggetti e che traduce le richieste di accesso per un dato soggetto in quelle relative ai suoi ruoli.

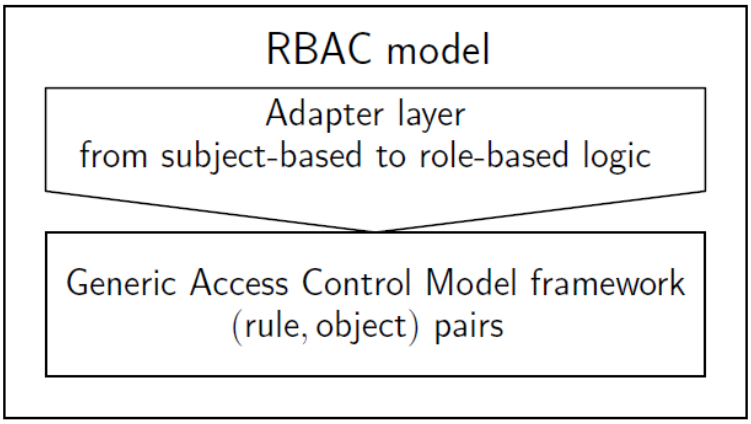


Figura 1.8: Architettura RBAC

Gerarchia dei ruoli

Generalmente è conveniente organizzare i ruoli in una struttura gerarchica, così da riflettere l'organigramma (gerarchico) di una data organizzazione. I diritti di accesso vengono più agevolmente gestiti e assegnati ai vari ruoli sfruttando l'ereditarietà.

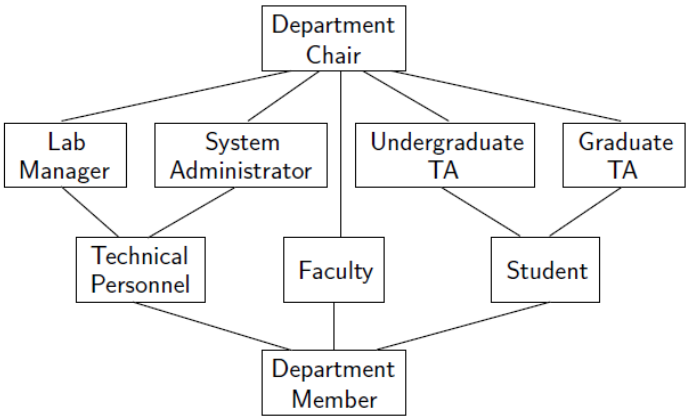


Figura 1.9: Architettura RBAC

Pro e Contro RBAC

Indipendentemente dal framework di basso di livello utilizzato per il controllo degli accessi, i vantaggi nell'uso del modello RBAC sono:

- la riduzione drastica del numero totale di regole di accesso da gestire; nella pratica il numero dei ruoli è significativamente inferiore a quello dei soggetti, inoltre la loro organizzazione gerarchica ne semplifica ulteriormente la gestione.
- l'overhead per determinare se un dato soggetto ha un dato diritto è contenuto, è sufficiente consultare se almeno uno dei suoi ruoli ha tale diritto.

Lo svantaggio principale è che non viene implementato dagli attuali sistemi operativi!

Capitolo 2

Secret Key Cryptography

2.1 Introduzione

La crittografia a chiave segreta richiede l'uso di UNA sola chiave: dato un messaggio (il testo in chiaro) e la chiave, la cifratura produce dati non intellegibili (il testo cifrato). Il testo cifrato ha circa la stessa lunghezza di quello in chiaro e la decifratura è l'inverso della cifratura, ed usa la stessa chiave. La crittografia a chiave segreta è talvolta chiamata crittografia **convenzionale** o crittografia **simmetrica**.

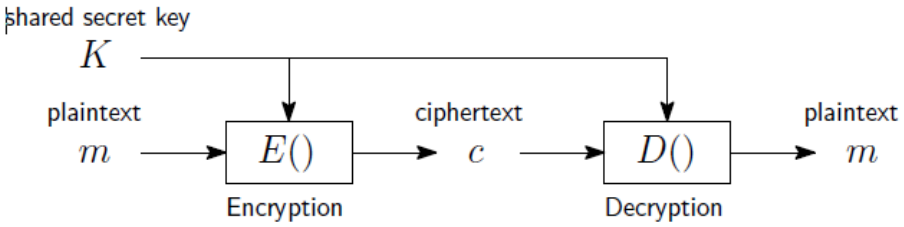


Figura 2.1: Schema a blocchi crittografia a chiave segreta

2.1.1 Impieghi crittografia a chiave segreta

La crittografia a chiave segreta è alla base di molti meccanismi di sicurezza, i suoi principali impieghi sono:

- protezione della **confidenzialità** (l'uso classico di queste tecniche è rappresentato dalle **comunicazioni su un canale insicuro**, uno degli usi più moderni invece dalla **memorizzazione sicura su supporto insicuro**)
- protezione dell'**integrità**: tramite queste tecniche possono essere eseguiti test per rilevare eventuali modifiche non consentite.
- autenticazione: tramite l'implementazione di protocolli per verificare l'identità di persone e/o processi.

Comunicazioni su un canale insicuro

In molte circostanze, due entità devono comunicare attraverso un canale insicuro correndo il rischio di essere ascoltate da una terza parte. Questo contesto è molto comune: si pensi alle reti LAN, che trasmettono dati in broadcast. La crittografia a chiave segreta permette a due entità che condividono un segreto (la chiave) di comunicare attraverso un canale insicuro, ove non può

essere garantita l'assenza di intercettazioni/ascoltatori (eavesdropper), avendo la garanzia che il contenuto della comunicazione rimarrà confidenziale.

Memorizzazione sicura

Si supponga di disporre di un supporto di memorizzazione non protetto (ad esempio accessibile a molti utenti). Se si desidera salvare i dati proteggendone la confidenzialità si può definire una chiave segreta, salvare i dati dopo averli crittografati con tale chiave e custodire la chiave segreta in un luogo protetto. Il rischio dell'uso di questa tecnica consiste nella possibilità di smarrimento della chiave. In tal caso i dati sarebbero irrevocabilmente persi.

Autenticazione forte

Per **autenticazione forte (strong authentication)** si intende che si è in grado di provare la conoscenza di un segreto, che contraddistingue l'identità di una data entità, senza rivelarlo. L'autenticazione forte è ottenibile utilizzando la crittografia a chiave segreta, ed è particolarmente utile quando due processi devono comunicare su uba rete insicura. A rigore il segreto che contraddistingue l'identitò che si desidera autenticare dovrebbe essere noto solo a quest'ultima. Nella crittografia chiave segreta tale requisito non piò essere soddisfatto. Il segreto in questione è una chiave crittografica che deve essere nota anche all'entità autenticante.

Esempio

Si supponga che Alice e Bob condividano una chiave segreta K_{AB} e che vogliano autenticarsi reciprocamente, cioè ciascuno vuole accertarsi dell'identità dell'altro.
ipotesi: K_{AB} è nota solo ad Alice e Bob. Alice deve dimostrare a Bob di conoscere K_{AB} senza rivelarla e viceversa.
Strategia a sfida e risposta: ciascuno dimostra di conoscere K_{AB} rispondendo ad una sfida posta dall'altro. La **sfida** è un numero/stringa random **r** non prevedibile e sempre diversa. La **risposta** alla sfida è la sfida stessa cifrata $E(K_{AB}, R)$. In Figura 2.2 è riportato un possibile schema di autenticazione a sfida e risposta (challenge-response) a chiave segreta. La procedura segue i seguenti passi:

- Alice genera un numero random r_A (la sfida) e la invia al presunto Bob
- il presunto Bob critta la sfida con la sua chiave segreta K'_{AB} e restituisce ad Alice la risposta $E(K'_{AB}, r_A)$
- Alice riceve la risposta del presunto Bob e la decritta con la chiave K_{AB}, cioè calcola $D(K_{AB}, E(K'_{AB}, r_A))$. Se ottiene r_A, allora il presunto Bob è realmente Bob poiché con elevatissima probabilità, se $D(K_{AB}, E(K'_{AB}, r_A)) = r_A$, allora $K'_{AB} = K_{AB}$. In caso negativo deduce invece che il presunto Bob è un impostore. In modo analogo Bob verifica l'identità di Alice.

La sicurezza del precedente protocollo si fonda sulle seguenti condizioni che non devono venire meno:

- solo e soltanto Alice e Bob devono conoscere la chiave segreta K_{AB}
- le sfide generate devono essere **randomiche**, e di conseguenza non prevedibili, e **non ripetibili**, cioè la probabilità che due sfide si ripetano deve tendere a zero. L'attaccante potrebbe infatti collezionare molte coppie testo in chiaro/testo cifrato.
- è importante quindi che il numero di bit di una sfida sia superiore ad una data soglia (almeno 64 bit)

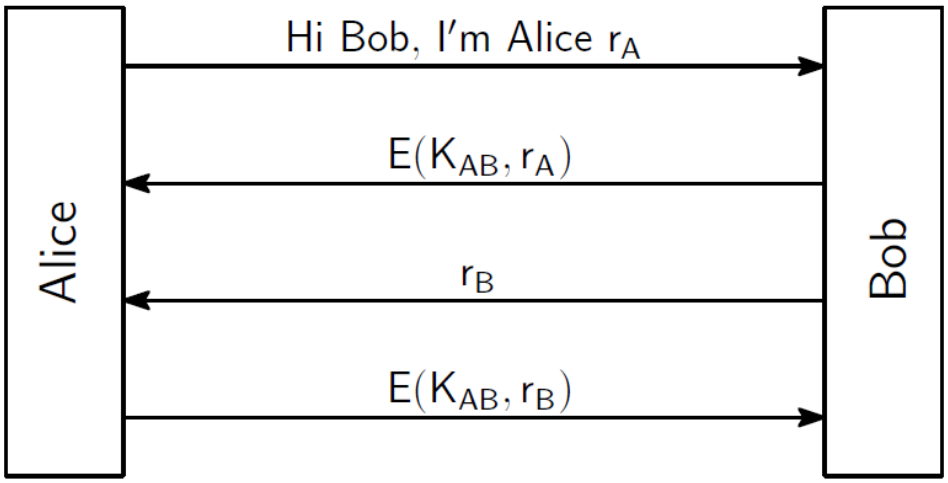


Figura 2.2: Schema a blocchi crittografia a chiave segreta

Controllo di integrità

La verifica dell'integrità di un messaggio inviato o di un file è un problema ricorrente in telecomunicazioni e in informatica. Per rilevare eventuali modifiche accidentali si fa uso generalmente di codici (somme) di controllo, detti anche **checksum**. Un **checksum** associa ad un qualsiasi messaggio $\mathbf{m} \in \{0,1\}^*$ un codice di lunghezza prefissato di b bit (generalmente $b = 32, 64, 128$ bit): $ck(\mathbf{m}) \in \{0,1\}^b$. Un buon **checksum** dovrebbe variare in modo significativo anche a fronte di iminime variazioni dell'input.

La sorgente del messaggio m_s rende pubblico/invia il corrispondente checksum $ck(m_s)$. Chi riceve il messaggio m_r calcola il checksum e verifica se vale l'uguaglianza $ck(m_s) = ck(m_r)$. In caso affermativo, conclude che $m_s = m_r$. Si noti tuttavia che, seppur improbabile, è possibile ottenere dei falsi positivi, cioè $ck(m_s) = ck(m_r)$ anche se non è vero.

Checksum segreti e non segreti

I codici di controllo servono per proteggere l'hardware da difetti e da inevitabili errori/guasti. Esistono codici di controllo molto sofisticati come i **CRC (Cyclic Redundancy Check)** per i quali la probabilità di falsi positivi è estremamente ridotta. Esistono anche i codici **FEC (Forward Error Correction)** che permettono di correggere eventuali errori oltre che a rilevarli (aggiungendo ulteriore ridondanza). Tuttavia entrambe queste tecniche **non** sono utilizzabili per la protezione contro attacchi intelligenti. Essendo pubblici, infatti, un avversario intelligente che vuole cambiare un messagio potrebbe modificare anche il codice di controllo in maniera coerente.

Per la protezione contro modifiche maliziose ad un messaggio, è richiesto un codice di controllo (checksum) segreto. Se l'algoritmo non è noto, nessuno può calcolare il checksum corretto per il messaggio modificato. Chiaramente, come nel caso degli algoritmi di cifratura, anziché un algoritmo segreto conviene avere un algoritmo noto a tutti che richiede la conoscenza di una chiave segreta per il calcolo di un codice di controllo (Vedi pagina 8, principio **Open Structure**).

In ciò consiste appunto un checksum cifrato, detto anche **MIC (Message Integrity Code)**. Il funzionamento del MIC è il seguente:

- l'algoritmo produce un codice di autenticazione di lunghezza fissa $MIC(K, m)$, denominato anche **MAC (Message Authentication Code)**
- il codice MIC MIC(K,m) viene trasmesso insieme al messaggio m stesso

- formalmente l'input è una coppia $(K, m) \in \{0,1\}^k \times \{0,1\}^*$, dove k denota il numero di bit della chiave segreta K, mentre l'output è sempre una stringa binaria di lunghezza prefissata, cioè $MIC(K, m) \in \{0,1\}^b$

2.1.2 Cifrari a blocchi e cifrari a flusso

Un **cifrario a blocchi** elabora un blocco di elementi in ingresso per volta, producendo un blocco di uscita per ciascun blocco di ingresso. Questa metodologia implica quindi che:

- il testo in chiaro deve essere preliminarmente suddiviso in blocchi
- il testo cifrato si ottiene combinando i vari blocchi cifrati

DES, IDEA e AES sono esempi di cifrari a blocchi simmetrici. Un **cifrario a flusso**, invece, elabora continuamente gli elementi in ingresso, producendo in uscita un flusso di elementi cifrati. Gli elementi cifrati vengono prodotti singolarmente, uno alla volta, man mano che la cifratura procede.

2.2 Cifratura a blocchi

2.2.1 Introduzione e concetti generali

L'algoritmo di cifratura converte un blocco di testo in chiaro in un blocco di testo cifrato. La chiave K non deve essere troppo corta (e.g. se K ha lunghezza 4 bit, sono sufficienti $2^4 = 16$ tentativi per individuarla). Analogamente la lunghezza (fissata) di un blocco non deve essere troppo piccola (e.g. se un blocco ha lunghezza 8 bit, ottenendo delle coppie plaintext - ciphertext si potrebbe costruire una tabella di $2^8 = 256$ coppie utilizzabile per la decfiratura).
D'altra parte, avere blocchi esageratamente lunghi oltre a non essere necessario dal punto di vista della sicurezza, comporta una gestione più complicata e può degradare le prestazioni. **64 bit è una lunghezza ragionevole per un blocco**:

- è improbabile ottenere ordine di 2^{64} coppie $\langle plaintext, ciphertext \rangle$ per costruire una tabella di decifratura, e
- anche se fosse possibile, la sua memorizzazione richiederebbe una spazio enorme (2^{64} record da 64 bit),
- come pure l'ordinamento per consentire ricerche efficienti

Il modo più generale per cifrare un blocco da 64 bit è definire una **biiezione** $\gamma : \{0,1\}^{64} \to \{0,1\}^{64}$. Tuttavia, memorizzare la definizione della *biiezione* in una struttura dati è impraticabile: sarebbero richiesti $2^8 4 \times 64 = 2^{70}$ bit. Ad essere precisi ce ne vogliono un po' di meno, trattandosi di un **biiezione**, comunque almeno 2^{69} bit sono richiesti (perché?!?!). Inoltre, così facendo la chiave è incorporata nella biiezione: per renderla parametrica rispetto alla chiave è necessario memorizzare una biiezione per ogni possibile chiave.

I sistemi di crittografia a chiave segreta sono concepiti per usare una chiave ragionevolmente lunga (ad esempio 64 bit e generare una **biiezione** che appare, a chi non conosce la chiave, completamente random. Se la biiezione fosse realmente random due input i e i' nei quali cambia un solo bit (uno qualunque) sarebbero **statisticamente indipendenti**: non può succedere, ad esempio, che il terzo bit dell'output cambia **sempre** quando il dodicesimo bit dell'input cambia. Gli algoritmi crittografici sono pensati per **diffondere/spargere** in tutti i bit dell'output il valore di ogni bit dell'input: si cerca di far si che ogni bit dell'output dipenda allo stesso modo da tutti i bit dell'input.

2.2.2 Sostituzione e Permutazione

Sostituzioni e permutazioni sono due trasformazioni base applicabili ad un blocco di dati.

Si assuma di dover cifrare un blocco di k bit. Una **sostituzione** specifica, per ciascuno dei 2^k possibili valori dell'input, i k bit dell'output. Per specificare una sostituzione **completamente random** sono necessari circa $k2^k$ bit. E' quindi impraticabile implementare una sostituzione per blocchi di 64 bit, mentre è fattibile per blocchi di lunghezza di 8 bit. Esempio di crittografia per sostituzione è il **Cifrario di Cesare**.

Una **permutazione** specifica, per ciascuna delle k posizioni dei bit in input, la posizione del corrispondente bit nell'output. Per specificare una permutazione **completamente random** per un blocco di lunghezza k bit sono necessari $k \log_2 k$ bit. Infatti per ciascuno dei k bit va specificata la sua posizione nell'output; ogni posizione richiede $k \log_2 k$. Ad esempio: essendo $2^6 = 64$, sono necessari $6 = \log_2 64$ bit per specificare la nuova posizione che l'i-esimo bit in input avrà in output.

Si noti che una **permutazione** è un caso particolare di **sostituzione** in cui ogni bit dell'output ottiene il suo valore da esattamente un bit dell'input.

2.2.3 Cifrario a blocchi – schema generale

Un algoritmo di cifratura a chiave segreta può funzionare come segue:

- scompone il blocco in input in pezzi più piccoli (e.g. blocchi da 8 bit)
- applica una sostituzione (tramite una **rete combinatoria**) a ciascun pezzo da 8 bit (la sostituzione dipenderà dal valore della chiave)
- gli output delle sostituzioni vengono riuniti in un unico blocco (64 bit)
- tale blocco viene permutato in un permutatore a 64 bit (che ha il compito di diffondere le modifiche eseguite nelle sostituzioni)
- il processo viene ripetuto un certo numero di volte riportando l'output in ingresso

2.2.4 Cifrario a blocchi – esempio

Ogni attraversamento del cifrario viene detto **round**. In riferimento alla Figura 2.3 si fanno le seguenti considerazioni. Con un solo round, un bit b_x di input può influenzare soltanto 8 bit $b_{x1}, b_{x2}, \ldots, b_{x8}$ dell'output, poiché b_x ha attraversato soltanto un blocco di sostituzione. In generale i bit $b_{x1}, b_{x2}, \ldots, b_{x8}$ non sono consecutivi essendo stati mescolati nel permutatore. Alla fine del secondo round, assumendo che i bit $b_{x1}, b_{x2}, \ldots, b_{x8}$ siano smistati in blocchi di sostituzione distinti, il bit b_x iniziale influenza tutti i bit in output.

2.3 Data Encryption Standard - DES

DES fu pubblicato nel 1977 dal **National Bureau of Standards**, ora rinominato **N**ational **I**nstitute of **S**tandards and **T**echnology (**NIST**), per usi commerciali e altre applicazioni del governo statunitense. progettato da IBM, si basa sul precedente cifrario *Lucifer* ed è frutto della collaborazione con consulenti della NSA. DES usa una chiave di 56 bit, e mappa un blocco di input da 64 bit in un blocco di output da 64 bit (l'algoritmo è quindi abbastanza rigido, al contrario di altri cifrari a blocchi). La **chiave** è in realtà costituita da una sequenza di 64 bit, ma un bit in ogni ottetto (il blocco da 64 bit è formato da 8 sequenze di 8 bit dette ottetti) è usato come **odd parity** su ciascun ottetto. Di fatto, soltanto 7 bit in ogni ottetto sono quindi veramente significativi come chiave.

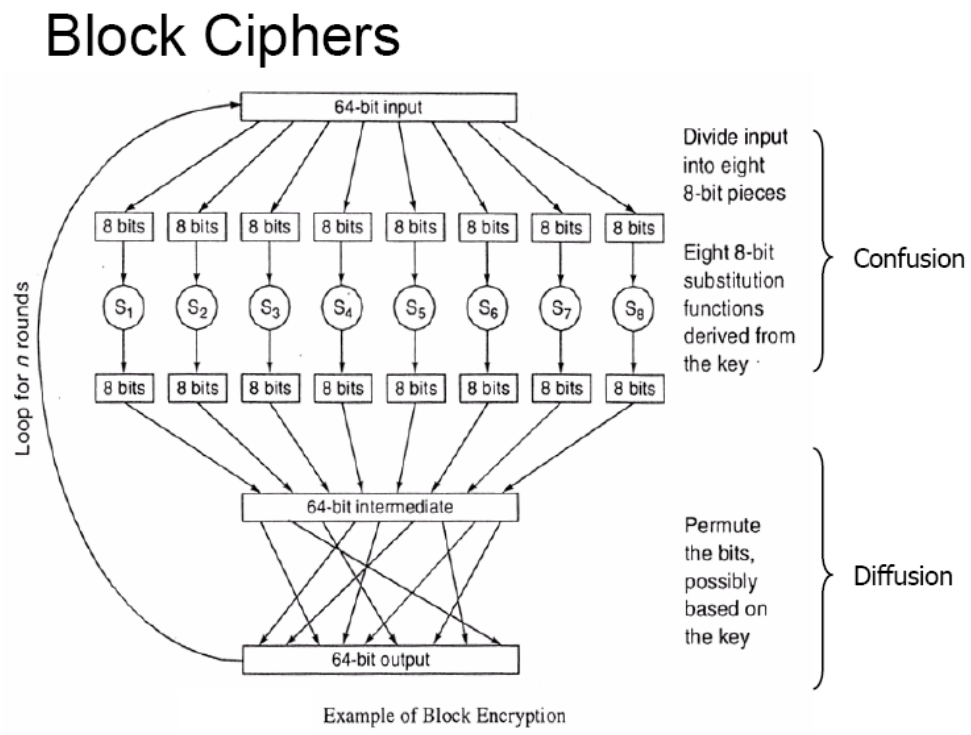


Figura 2.3: Schema a blocchi crittografia a chiave segreta

DES è efficiente se realizzato in hardware, ma relativamente lento se implementato in software. Sebbene l'essere difficilmente implementabile come software non era un requisito specificato nel progetto molti sostengono che in realtà era un fatto voluto, forse per limitarne l'uso ad organizzazioni in grado di realizzare sistemi hardware, o forse perché rese più facile controllare l'accesso alla tecnologia. Ad ogni modo, l'aumento delle capacità di calcolo delle CPU rese possibile realizzare una versione software di DES. Una 500-MIPS (Million Instruction Per Second) CPU può infatti cifrare ad un tasso di circa 30 KB/s e forse più, dipende dai dettagli architetturali della CPU e dalla intelligenza dell'implementazione. Un processore Intel Core i7 Extreme Edition i980EE ha una capacità di calcolo di circa 150 MIPS, quindi può cifrare ad un tasso di circa 9 KB/s (per cifrare 1 MB impiega circa 110 secondi). L'implementazione software è pertanto attualmente adeguata a molte applicazioni.

2.3.1 Perché chiavi da 56 bit?

La scelta di una chiave da 56 bit causò molte controversie. Prima che DES fu adottato, le persone al di fuori della *intelligence community* lamentavano che 56 bit non offrivano una sicurezza adeguata. Perché solo 56 dei 64 bit di una chiave DES sono effettivamente usati nell'algoritmo? Lo svantaggio di usare 8 bit della chiave per un controllo di parità è che ciò rende DES molto meno sicuro (256 volte meno sicuro contro una ricerca esaustiva). Ma qual è il vantaggio di usare 8 bit per un controllo di parità? Una possibile risposta è che permette di verificare che la chiave non sia corrotta. Tuttavia questa spiegazione non regge. Se si considerassero infatti 64 bit a caso invece della chiave, c'è una probabilità su 256 che il controllo di parità dia esito positivo. La probabilità che la chiave sia comunque errata (nonostante il controllo di parità dia esito positivo) è troppo alta. Inoltre avere una chiave corrotta non comporta un problema di sicurezza, semplicemente la cifratura/decifratura non viene eseguita correttamente. Chiaramente è anche ridicolo sostenere che la scelta di 56 bit sia stata fatta per risparmiare memoria.

La risposta ormai condivisa è che il governo statunitense abbia deliberatamente indebolito la sicurezza di DES di una quantità appena sufficiente da consentire alla NSA di violarlo.

2.3.2 Violabilità e sicurezza di DES

Gli avanzamenti tecnologici dell'industria dei semiconduttori hanno reso ancora più critico il problema della lunghezza della chiave di DES. la velocità dei chip e un po' di furbizia permettono di violare (individuare) le chiavi DES con approcci a forza bruta in tempi ragionevoli. Il rapporto prestazioni/prezzo dell'hardware cresce del 40% per anno. La lunghezza delle chiavi dovrebbe aumentare di 1 bit ogni 2 anni. Assumendo che 56 bit erano appena sufficienti nel 1979 (quando DES fu standardizzato), 64 bit erano adeguati nel 1995, e 128 bit dovrebbero essere sufficienti fino al 2123.

Ragioniamo ora sulla sicurezza di DES. Se si dispone di un singolo blocco $\langle plaintext, ciphertext \rangle$ quanto è difficile trovare la chiave? Un approccio a forza bruta dovrebbe provare ordine di $2^{56} \approx 10^{17}$ chiavi. Se ogni tentativo richiede una singola istruzione sono necessarie ordine di 1000 MIPS-year istruzioni.

- 1 MIPS = 1 Milione di Istruzioni Per Secondo
- 1 MIPS-year = numero di istruzioni eseguite in un anno ad un tasso pari a 1 MIPS
- 1 MIPS-year = 1 MIPS $\times(365 \times 86400)$ secondi in un anno $= 3,1536 \times 10^{13}$ istruzioni
- $2^{56} \approx 10^{17} \approx 10^3$ MIPS-year