

Dati Sicuri

[Return](#)

Indice

- Dati Sicuri
- Indice
 - Trasformazioni per la sicurezza
 - Crittografia e Crittoanalisi
 - Principi della difesa
 - Integrità
 - Riservatezza
 - Autenticazione
 - Identificazione
 - Calcoli impossibili
 - Trasformazioni Segrete e Chiavi
 - Crittanalisi
 - Tre livelli di Gerarchia
 - Generatore di numeri casuali (RNG)
 - True Random Number Generator
 - Pseudo Random Number Generator
 - Funzione Hash
 - Attacco alle funzioni hash
 - Algoritmi di hash
 - Servizi di Identificazione
 - Identificazione passiva
 - Identificazione attiva
 - One-time password
 - Sfida e risposta
 - Zero knowledge

Trasformazioni per la sicurezza



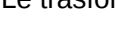
Trasformazione: Operazione che trasforma un messaggio in un altro messaggio

Arbitro: Terzo ente che verifica che le trasformazioni siano corrette

Se Sorgente o Destinazioni sono fidate, potrebbe non servire l'arbitro



Crittografia e Crittoanalisi



Crittografia: Scienza che studia le trasformazioni per la sicurezza **Crittoanalisi**: Scienza che studia come rompere le trasformazioni per la sicurezza

Principi della difesa



Le trasformazioni per chi è autorizzato sono calcoli facili da eseguire, per chi non è autorizzato diventano calcoli computazionalmente molto complicati.

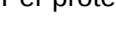
Integrità

Preservare il dato da alterazioni accidentali o intenzionali.

Il caso più semplice per proteggere da errori casuali potrebbe essere un checksum o un CRC (Cyclic Redundancy Check, hash semplice), ma non sono sufficienti per proteggere da attacchi intenzionali.

Per proteggere da errori intenzionali si utilizzando **funzioni hash crittografiche**. Producono un riassunto univoco.

Dato un messaggio di lunghezza *m* producono un hash di lunghezza *n* (dove *n* è molto più piccolo di *m*). Per rendere difficile il compito dell'attaccante l'hash si deve comportare da **oracolo casuale**, utilizzare simboli equiprobabili.



Il messaggio *m* viene inviato da S a D su un canale, il mittente calcola l'hash e lo trasmette su un canale sicuro, il destinatario calcola l'hash e lo confronta con quello ricevuto. Se sono uguali il messaggio è integro. Un attaccante proverà a mandare un messaggio alterato che produca lo stesso hash, con funzioni hash crittografichè è pressochè impossibile.

Riservatezza

Per proteggere la riservatezza delle informazioni è necessario ricorrere a metodi di cifratura. La sorgente concorda con la destinazione un metodo di rappresentazione.

I calcoli da svolgere devono essere facili per sorgente e destinazione, ma computazionalmente impossibili per un attaccante.



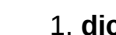
Ally cifra il messaggio con un *segeto* producendo il *ciphertext* e lo invia a Bob. Bob decifra il messaggio con lo stesso *segreto* concordato e ottiene il messaggio originale.

Per proteggere contemporaneamente **riservatezza** e **integrità** si procede:

- producendo *p* concatenando il messaggio *m* e l'hash *h* (*m*||*H*(*m*))
- cifrando *p* per produrre il *ciphertext* *c* (*c*=*E*(*p*)) Bob riceve *c** con * perchè non sa se è stato alterato o no
- p** = *D*(*c**) = *m**||*H**(*m*) D = Decrypt
- Si confrontano *H*(*m*)* e *H**(*m*), se sono uguali il messaggio è integro, altrimenti è stato alterato.

Autenticazione

Per certificare che l'autore di un messaggio sia autenticato si utilizzano le firme digitali. Anche in questo caso i calcoli per dimostrare la propria identità devono essere semplici mentre computazionalmente impossibili quelli per creare un falso autentico.



La sorgente trasmette il messaggio *m* su un canale, genera l'hash del messaggio e lo firma, lo cifra e lo trasmette al destinatario mediante un canale sicuro, Il destinatario riconosce la firma, decifra il messaggio e verifica che l'hash sia corretto. Se sono uguali il messaggio è integro e l'autore è autenticato.

Identificazione

L'identificazione permette di identificare un utente a real time (non come l'autenticazione). Si basa su tre concetti principali:

- conoscenza**: l'utente deve conoscere un segreto (password, PIN, ecc.)
- possessione**: l'utente deve possedere un oggetto (carta di credito, smart card, ecc.)
- conformità** : l'utente deve essere conforme a un modello (impronta digitale, riconoscimento facciale, ecc.)

Un protocollo di identificazione prevede una fase di **registrazione** in cui **identificando** e **verificatore** concordano un segreto. Poi si procede in 3 fasi:

- dichiarazione**: l'utente dichiara di voler essere identificato
- interrogazione**: il verificatore interroga l'utente
- dimostrazione**: l'identificando comunica il segreto e il verificatore lo confronta con quello atteso

Calcoli Impossibili

Abbiamo parlato di calcoli difficili per un utente malevolo. Definiamo questo concetto.

Una funzione *f* è detta **unidirezionale** o **one-way function** se:

- è invertibile
- facile da calcolare
- per quasi tutti gli x appartenenti al dominio è **difficile** calcolare *y* = *f**(*x*)

| esempio: **elenco telefonico** trovare il numero di telefono di una persona ha complessità *O*(*n*), mentre trovare il nome di una persona a partire dal numero di telefono ha complessità *O*(*n*^2)

Non esistono funzioni unidirezionali perfette ma **pseudo unidirezionali** o **trapdoor one-way**.

I problemi vengono classificati in:

- facili** se esistono algoritmi polinomiali in grado di risolverli su macchine deterministiche
- difficili** se non sono stati individuati algoritmi che li risolvono in tempo polinomiale su macchine deterministiche.

Definiamo quindi:

- Tempo di esecuzione di un algoritmo**: numero di operazioni *N* eseguite in funzione della dimensione *n* dell'input.
- Tempo di esecuzione nel caso peggiore**: è il numero massimo di operazioni \$N_{(max)}\$ eseguite in funzione della dimensione *n* dell'input.

Sulla base di queste grandezze si studia l'andamento asintotico del tempo di esecuzione al crescere della dimensione di *n*, lo si definisce **Ordine di grandezza del tempo di esecuzione**: \$T(n) = O(g(n))\$, dove *g*(*n*) è una funzione tale che \$0<(n)<ctimes g(n)\$ per ogni \$n>n_0\$, dove *c* è una costante positiva e \$n_0\$ è un numero naturale.

Ogni algoritmo che consente di **difendere** una proprietà critica deve avere tempo polinomiale, metre gli algoritmi che consentono di **rompere** una proprietà critica devono avere tempo esponenziale.

Utilizzeremo quindi funzioni pseudo-unidirezionali la cui risoluzione è polinomiale e la cui inversione è esponenziale o semi-esponenziale.



Trasformazioni Segrete e Chiavi

La cosa migliore è che la responsabilità del segreto sia dell'utente, utilizzando un algoritmo pubblico ma che funzioni con un parametri privato noto solo all'utente.

Esistono due tipi di algoritmi:

- a chiavi simmetriche**: le due chiavi \$k_s\$ e \$k_d\$ sono uguali, la chiave è nota a sorgente e destinatario, ma non all'attaccante.
- a chiavi asimmetriche**: le due chiavi \$k_s\$ e \$k_d\$ sono diverse, la chiave di cifratura è nota a tutti (**chiave pubblica**), mentre la chiave di decifratura è nota solo al destinatario (**chiave privata**). Ogni soggetto ha una chiave pubblica e una privata.



Crittanalisi

Studia come decifrare testi cifrati senza conoscere la chiave. Escludendo i calcoli, le chiavi possono essere indovinate, intercettate o dedotte. Questo ovviamente deve essere impossibile.

I segreti possono essere indovinati facendo una ricerca esauriente (brute force), utilizzando dizionari composti con le parole più probabilmente utilizzate, oppure utilizzando le chiavi più comuni (es. 123456, password, ecc.).

Siamo comunque esposti a diversi tipi di attacco:

- con solo testo cifrato**: si studia il presunto linguaggio del messaggio e si sfruttano calcoli sulle probabilità di occorrenza, (XOR tra il testo cifrato e il testo in chiaro dedotto)
- con solo testo in chiaro noto**: se si hanno coppie di testo in chiaro e cifrato si può dedurre la chiave (es. XOR tra il testo in chiaro e il testo cifrato)
- con testo in chiaro scelto**: se si ha la possibilità di avere la versione in chiaro di un determinato testo cifrato si può ricostruire il segreto.
- con testo cifrato scelto**: se si ha la possibilità di cifrare un testo a piacere e di ricevere il testo cifrato, si può dedurre la chiave.

Tre livelli di Gerarchia

Si può strutturare un file system cifrato in tre livelli di gerarchia dei segreti:

- Primo Livello**: L'utente impara a memoria una password di cui viene memorizzato un hash
- Terzo Livello**: dove vengono salvati dati sensibili. Sono archiviati facendo uso di una chiave scelta ogni volta da un **RNG**
- Secondo Livello**: per accedere al terzo livello si utilizza una chiave memorizzata in un portachiavi generata utilizzando un **RNG** e un segreto, al secondo livello.

Serve una forma di recovery.

Generatore di numeri casuali (RNG)

Alla stringa generata da un RNG sono richieste 2 proprietà:

- Causalità**: ogni valore deve avere la stessa rpobabilità di verificarsi ed essere statisticamente indipendente da tutti gli altri.
- Imprevedibilità**: deve essere computazinalmente impossibile prevedere il valore successivo.

per verificare la casualità sono stati definiti alcuni *test statistici*:

- Monobit**: valuta se il numero di 0 e 1 sono uguali.
- Poker**: divide la suquenza in blocchi da \$M\$ bit e valuta se il numero di volte che compare ciascuna delle \$2^M\$ configurazone è lo stesso.
- Run**: considera le stringhe di bit consecutivi uguali e valuta se il numero di 0 e 1 è lo stesso.
- Long Run**: considera il più lungo run di \$1\$ e valuta se la lunghezza è compatibile con quella attesa per una sequenza casuale.
- Autocorrelazione**: valuta se la sequenza è casuale considerando le differenze tra i bit.
- Trasformata discreta di Fourier**: valuta se la sequenza è casuale considerando le differenze tra i bit.

L'imprevedibilità viene valutata con **next-bit test**: data \$L\$ bit non deve eseistere nessun algoritmo polinomiale che permetta di predire \$L+1\$ con probabilità maggiore di \$1/2\$.

True Random Number Generator

Il TRNG si basa su fenomeni fisici. I generatori hardware digitalizzano un segnale analogico fornito da una sorgente. A volte viene fatto un post processing per rendere equiprobabili 1 e 0 (skewing).

Pseudo Random Number Generator

Il PRNG si basa su algoritmi deterministicci che producono sequenze di bit casuali. La sequenza è determinata da un seme iniziale, che deve essere scelto in modo casuale. La sequenza è periodica e il periodo è limitato dalla lunghezza del seme.

Oggi è conveniente l'uso di generatori algoritmici, quindi sono stati trovati nuovi algoritmi per conseguire **casualità** e **imprevedibilità**. I generatori algoritmici sono più veloci e più facili da implementare, ma meno sicuri.

Funzione Hash

L'output di una funzione di Hash è detta **riassunto** o **impronta** (*digest* o **fingerprint**).

Tuttavita riducendo una stringa di \$m\$ bit a una di \$n\$ bit, con \$n<m\$, è possibile che due stringhe diverse producano lo stesso hash. Questo fenomeno è detto **collisione**.

Si distinguono quindi funzioni di hash **semplici** e **crittografiche** a seconda che l'individuazione di collisioni sia facile o difficile.

Per proteggere l'**integrità** è necessario che l'individuazione di collisioni sia difficile. Per proteggere la **riservatezza** deve essere difficile il colcolo dell'inversa.

Altra proprietà che devono avere le funzioni di Hash è la **non invesibilità**.

Tutti gli algoritmi di has si basano sul principio della **compressione iterata**, ovvero si applica una funzione di compressione a blocchi di lunghezza fissa, che produce un hash di lunghezza fissa. La funzione di compressione è iterata più volte fino a ottenere l'hash finale.

Attacco alle funzioni hash

Un attacco a cui sono vulnerabili alcune funzioni di hash è il **length extension attack**.

Algoritmi di hash

- MD5**: produce un hash di 128 bit, è stato progettato per essere veloce e sicuro, ma sono stati trovati attacchi che ne compromettono la sicurezza.
- SHA-1**: produce un hash di 160 bit, è stato progettato per essere sicuro, ma sono stati trovati attacchi che ne compromettono la sicurezza.
- RIPEMD**: produce un hash di 128, 160, 256, 384 o 512 bit, è stato progettato per essere sicuro e veloce.

Servizi di Identificazione

Un protocollo di identificazione ha lo scopo di identificare un'entità A verso un'entità B. Deve però rispettare alcuni requisiti:

- Se le entità in gioco sono fidate, B deve poter completare il protocollo di identificazione certo dell'identità di A.
- B non può riutilizzare lo scambio di identificazione avuto con A per impersonare illegittimamente A.
- la probabilità che C riesca a completare il protocollo spacciandosi per A deve essere prossima allo 0.
- tutti gli obiettivi devono rimanere validi se un numero elevato di identificazioni tra A e B sono state osservate e se C è stato precedentemente coinvolto in sessioni di identificazioni con A o B.

Un protocollo di identificazione si svolge in 3 fasi:

- A dichiara a B di volersi identificare
- B interroga A
- A fornisce a B una prova della sua identità

L'identificazione può essere **passiva** o **attiva**

Identificazione passiva

Un meccanismo di identificazione passiva si basa sull'inserimento di una password che quindi sarà inviata all'ente verificatore tramite un messaggio del tipo \$A||PSW\$

I possibili attacchi sono:

- Replay attack**: l'attaccante registra il messaggio e lo reinvia al verificatore, che lo accetta come valido.
- attacco di forza bruta**: l'attaccante prova tutte le possibili combinazioni di password fino a trovare quella giusta.
- attacco a dizionario**: l'attaccante prova tutte le parole di un dizionario fino a trovare quella giusta.
- accesso al file di password**: l'attaccante accede al file di password e trova la password giusta, o concatena il suo ID e la password per ottenere l'autorizzazione.

Identificazione attiva

Consiste nel cambiare continuamente la prova d'identità. Può essere fatta in vari modi:

- one-time password**
- sfida e risposta**
- zero knowledge**

Tutte e tre devono rispettare che il calcolo della proa sia facile per chi conosce il segreto e difficile per chi non lo conosce o dispone solo delle prove precedenti.

One-time password

Funziona o tramite funzioni unidirezionali o tramite un **cifrario** con chiavi di sessione.

Il primo metodo prevede che, in fase di registrazione, A scelga un numero casuale \$X_A\$ e impieghi una funzione unidirezionale \$F\$ per calcolare \$F(X_A)\$, \$F^1(X_A)\$, ... , \$F^n(X_A)\$. Su B sarà memorizzato \$F^n(X_A)\$ e A potrà utilizzare \$F^n(X_A)\$ a partire da \$i=n-1\$ fino a \$i=0\$ per identificarsi. B memorizza \$F^n(X_A)\$ e lo conoscendo \$F\$ verifica che \$F(F^n(X_A))\$ sia \$F^{n+1}(X_A)\$, poi sostituisce l'ultimo.

Il secondo metodo prevede che i corrispondenti, all'inizio di ogni sessione modifichino il segreto con il quale viene cifrata la medesima password. La password viene cifrata con una chiave di sessione, che viene generata in modo casuale.

Sfida e risposta

Zero knowledge

Si tratta di dare testimonianza della propria capacità di risolvere un problema ritenuto difficile.

- L'identificando fornisce una testimonianza di ciò che sa fare.
- Il verificatore lancia una sfida che possa convincerlo.
- L'identificando fornisce una risposta alla sfida.