

(1-2_intro-2_0.pdf)

Perché studiare la computer security? perché è ovunque dai computer casalinghi, per i servizi bancari, nelle telecomunicazioni e anche gli enti governativi hanno bisogno di essere protetti con la sicurezza informatica.

La computer security si occupa della prevenzione e del rilevamento di azioni non autorizzate da parte di un utente in un sistema di un PC.

L'autorizzazione è fondamentale per la definizione, sono permesse solo le azioni concesse tramite delle **security policy** (politiche di sicurezza) che definiscono chi (o cosa) possa eseguire quale azione.

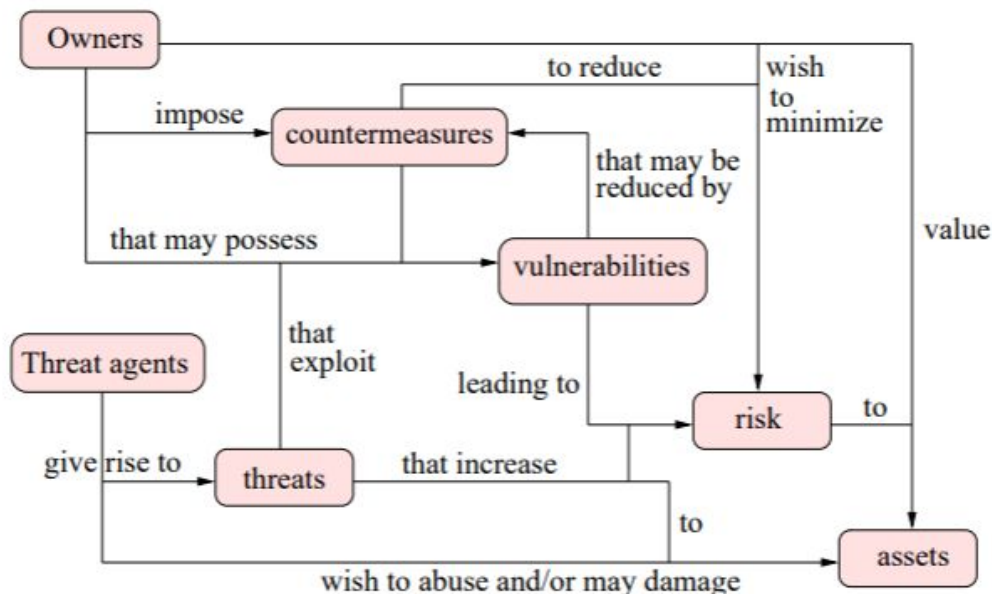
La sicurezza informatica (information security) è più generale e si occupa delle informazioni (anche i singoli dati, ma non solo, l'informazione è più generale dei dati) indipendentemente dal sistema di elaborazione.

La sicurezza informatica si preoccupa di proteggere le risorse dalle minacce.

Il possessore dà valore alle proprie risorse e le vuole proteggere, anche l'hacker dà valore all'asset del possessore e prova a trovarlo e prenderlo.

Il possessore analizza le minacce e cerca le contromisure per ridurre le vulnerabilità al rischio.

Ovviamente qualche rischio può rimanere a causa di vincoli come fattibilità e spesa, l'obiettivo del possessore è quello di ridurre il rischio.



Proprietà della sicurezza:

- riservatezza, le informazioni non devono essere apprese da attori non autorizzati;
- integrità, i dati non devono essere (maliziosamente) alterati;
- autenticazione, gli attori e/o l'origine dei dati devono poter essere rintracciati;
- disponibilità, i dati/servizi devono poter essere acceduti quando desiderato;
- responsabilità, le azioni possono essere ricondotte agli attori responsabili.

Di solito si vogliono garantire le proprietà con meccanismi specifici, ma dobbiamo ricordare che la sicurezza è un problema dell'intero sistema (software, hardware, personale, ambiente esterno, etc...).

Contromisure di protezione:

- prevenzione, prevenire le violazioni alla sicurezza con tecnologie e protocolli adatti (es. firewall per prevenire gli accessi esterni in una rete aziendale);
- rilevamento, nel caso avvenisse una violazione vogliamo assicurarci che venga rilevata. I file di log e file di MAC i metodi di rilevamento primari anche se i sistemi di rilevamento degli intrusi attivi sono più comuni.
- risposta, in seguito ad una violazione dobbiamo rispondere all'attacco o recuperare l'asset. Le risposte spaziano da ripristinare i backups a informare le parti interessate o le autorità competenti.

Riservatezza:

- una sua violazione può essere caratterizzata da una lettura non autorizzata dei dati quando consideriamo le misure per il controllo degli accessi, ma in generale siamo preoccupati dall'estrapolazione non autorizzata delle informazioni, che è un concetto più fine.
- la riservatezza presume la nozione di parte autorizzata, più in generale, una politica di sicurezza dice chi o cosa può avere accesso ai dati (la security policy è usata per l'access control).
- talvolta con privacy si intende la riservatezza per gli individui, mentre con secrecy viene intesa la segretezza per le organizzazioni. Con privacy, ancora, si intende l'anonimato, mantenere la propria identità privata.
- Esempio: le tue cartelle cliniche sono ottenute dal tuo capo senza il tuo consenso.

Integrità:

- nella computer security ci preoccupiamo di prevenire alterazioni malevole ai dati da qualcuno che non sia autorizzato a farlo.
- l'integrità può essere descritto come una scrittura non autorizzata dei dati e, di nuovo, le politiche di sicurezza dicono a chi o a cosa è concesso alterare i dati.
- Esempio, un sistema di pagamento online altera la lettura di un assegno elettronico 10.000€ invece di 100€.

Autenticazione:

- l'autenticazione è la verifica dell'identità di una persona o di un sistema.
- i metodi di autenticazione sono spesso caratterizzati da:
 - qualcosa che hai, e.g. una chiave;
 - qualcosa che sai, e.g. una password;
 - qualcosa che sei, e.g. un'impronta digitale, firma, o chiave biometrica.
- anche la posizione geografica in cui ti trovi potrebbe essere controllata esplicitamente o implicitamente per un'ulteriore verifica di sicurezza.
- Più metodi possono essere combinati per una maggiore sicurezza.
- Esempio: pretendendo di essere qualcun altro (furto di identità) fingendo e-mail o rubare le credenziali.

Disponibilità:

- i dati o i servizi devono poter essere acceduti in modo affidabile e tempestivo.
- le minacce alla disponibilità possono essere esterne (e.g. fuoco, spegnimento del server per mancata corrente) così come accidentali o malevoli attacchi software (e.g. malware); nella computer security ci preoccupiamo di quest'ultimi.

- garantire la disponibilità significa prevenire attacchi DoS (denial of service), nella misura in cui è possibile. È più complesso garantire sicurezza su questo tipo di attacchi in quanto potrebbe non essere semplice distinguere un attacco da un legittimo utilizzo del servizio.
- Esempio: i micidiali attacchi DoS (DDoS) sui servizi online; interferire con il routing.

Responsabilità:

- se i metodi di prevenzione dell'accesso e del controllo falliscono è necessario avere un traccia di controllo sicura degli eventi, affinché si possa fare marcia indietro fino alla parte responsabile.
- una forma maggiore di responsabilità è la non-ripudiazione, quando una parte non può negare una azione dopo averla eseguita.
- creare una traccia di controllo con una macchina di log è problematico: se un sistema è compromesso, anche i log potrebbero essere manomessi. Un modo semplice per aggirare il problema è quello di mandare i messaggi di log su un altro server distante dal primo in una stampante isolata e tenere così traccia degli eventi.
- Esempio: una traccia di controllo viene manomessa, persa o non è possibile stabilire dove si è verificata una violazione della sicurezza.

Gestione della sicurezza: implementare una soluzione

- analisi della sicurezza: esamina i rischi per determinare le risorse da proteggere e propone politiche e soluzioni ad un giusto costo;
- modello di rischio: documenta le possibili minacce ad un sistema, immaginando tutte le vulnerabilità che potrebbero essere sfruttate.
- valutazione del rischio: studia la probabilità di ogni minaccia al sistema e assegna un valore di costo, per trovare i rischi.
- politica di sicurezza: definisce un insieme coerente di contromisure per affrontare le minacce.
- i costi delle contromisure vengono rapportati ai costi dei rischi e viene valutato un ragionevole compromesso (+ o - protetto in base ai valori delle risorse da proteggere in relazione alla spesa dell'implementazione delle contromisure).
- Ciò consente di progettare una soluzione di sicurezza, implementando tecnologie appropriate in un costo adeguato. In parte questo è un esercizio di bilancio; ma è anche importante spendere sforzi nel posto giusto.

(2_Cos_è il cyber-risk)

Il termine **cyber-risk** si riferisce alla possibilità di **danno** subito da un'organizzazione a seguito di un malfunzionamento dei suoi sistemi informatici causato da un agente ostile. La consapevolezza del cyber-risk è cresciuta rapidamente negli ultimi anni a causa sia dell'aumentata dipendenza dai servizi digitali da parte delle aziende che dal crescente numero di incidenti cyber di alto profilo.

Comprendere e valutare i fattori che contribuiscono al cyber-risk è il presupposto indispensabile alla pianificazione di azioni tese a mitigarlo. Tali fattori sono le minacce, le vulnerabilità e l'impatto.

Le **minacce** (*threats*) possono essere portate da una molteplicità di **agenti ostili** (*threat actors*), quali ad esempio, nazioni, criminali (singoli o organizzati in gruppi), attivisti (ad es. il gruppo *anonymous*), o un dipendente infedele (*insider*). Molteplici possono essere le

motivazioni che guidano gli agenti ostili: profitto, obiettivi politici o ideologici, spionaggio governativo o aziendale, finalità militari, per citarne alcuni.

DEF **vulnerabilità**: sono difetti delle infrastrutture tecnologiche, delle procedure o dei processi aziendali che consentono l'esecuzione di comportamenti indesiderati nelle infrastrutture informatiche. Gli agenti ostili **sfruttano** (*exploit*) le vulnerabilità per lanciare **attacchi cyber**.

DEF **impatto**: è la conseguenza di un attacco cyber, quali la perdita finanziaria, il danno reputazionale, l'interruzione del servizio, la sottrazione di segreti industriali, e la divulgazioni di dati sensibili.

L'impatto è normalmente legato alla violazione di una o più delle proprietà di **confidenzialità** (*confidentiality*), **integrità** (*integrity*) e **disponibilità** (*availability*) delle **risorse digitali** (*digital assets*) dell'organizzazione. La confidenzialità si riferisce all'esigenza di restringere l'accesso alle informazioni ad un insieme ristretto di soggetti autorizzati, l'integrità alla garanzia che i dati siano accurati e affidabili, la disponibilità alla garanzia che i dati o i servizi siano accessibili e quindi fruibili al personale autorizzato.

Gli accessi fisici (recarsi in loco) non autorizzati sono una minaccia concreta e sottovalutata, tuttavia la maggior parte degli attacchi avviene da remoto sfruttando le debolezze dei servizi informatici, un vettore molto efficace è la posta elettronica che permette di inviare malware come allegati.

(3_Cyber Attacks)

Gli attacchi cyber colpiscono tanto le organizzazioni governative quanto le aziende e l'esposizione alla minaccia non dipende dalla dimensione o dal settore in cui opera l'organizzazione. In altre parole, nessuna organizzazione moderna può ritenersi immune agli attacchi cyber, ovvero non soggetta a cyber-risk.

ransomware: particolare tipologia di malware progettata per penetrare nelle infrastrutture informatiche della vittima, produrre disservizi quali la cifratura del patrimonio informativo e richiedere il pagamento di un riscatto in Bitcoin per rientrare nella disponibilità dei dati cifrati.

Advanced Persistent Threat (APT), sono caratterizzati dalla capacità degli attaccanti di penetrare l'infrastruttura informatica della vittima sfruttando vulnerabilità dell'infrastruttura informatica e l'involontaria partecipazione dei dipendenti della banca (mediante tecniche di *social engineering*) e di mantenere in essa la propria presenza senza essere notati anche per lunghi periodi di tempo (in alcuni casi anche per mesi) durante i quali gli attaccanti identificano le vulnerabilità nelle procedure e nei processi aziendali e sviluppano tecniche di attacco specifiche.

L'esfiltrazione o l'alterazione di dati sensibili custoditi dalle organizzazioni è un obiettivo spesso perseguito dagli attaccanti cyber. Si pensi che nel **Dark Web** (porzione del web dove, grazie all'anonimato che lo distingue, esiste un fiorente commercio di beni e servizi illegali) è possibile acquistare (illegalmente) dati sanitari ad un costo che attualmente oscilla tra i 250 e i 1.000 dollari per individuo. L'attacco condotto recentemente al sito dei pagamenti della *American Medical Collection Agency (AMCA)*, ha portato all'esfiltrazione di dati sensibili di più 20 milioni di cittadini americani. Oltre al costo diretto causato dalla gestione dell'incidente informatico, il danno reputazionale è stato così ingente da portare l'azienda al fallimento.

(4_crypto-intro.pdf)

Per rendere sicuro un canale non affidabile sono necessarie:

- confidenzialità
- integrità
- autenticazione
- e altre proprietà sono desiderabili (e.g. non-ripudiazione)

La **crittografia** è la tecnologia che permette questo.

DEF Crittologia: è la disciplina che si occupa delle scritture nascoste.

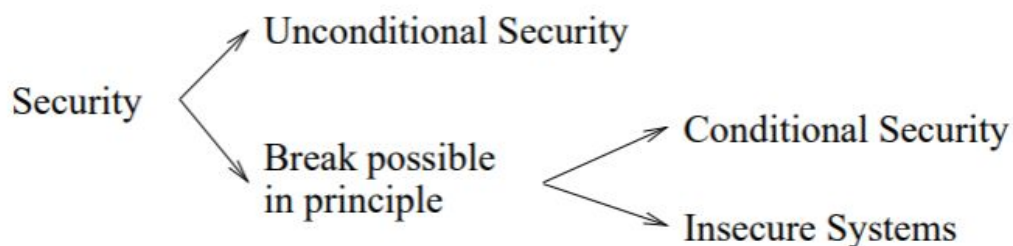
DEF Crittografia: branca della crittologia che tratta di scritture nascoste, insieme di metodi per “offuscare” il messaggio, rendendolo incomprensibile a persone non autorizzate al leggerlo.

DEF Steganografia: tecnica con l’obiettivo di scrivere messaggi nascosti altri messaggi e di nascondere la comunicazione tra due interlocutori.

La sicurezza di uno schema crittografico dipende dalla chiave, non dall’algoritmo.

Possono esserci algoritmi:

- simmetrici: in cui le chiavi di mittente e destinatario sono uguali, o possono inviarsele facilmente,
- asimmetrici o a chiave pubblica: le chiavi in possesso degli interlocutori sono differenti e non possono essere inviate, la chiave pubblica può essere diffusa senza compromettere l’efficacia della chiave privata.
- la cifratura e decifratura devono essere semplici se si è in possesso delle chiavi.



La sicurezza può essere:

- incondizionata: il sistema è sicuro anche se l’attaccante ha potenza di calcolo illimitata, questo perché il testo cifrato non permette di ricavare il corrispondente testo in chiaro. La sicurezza è misurata tramite la teoria dell’informazione.
- condizionata: il sistema può essere bucato in linea teorica, ma la potenza di calcolo richiesta è maggiore di quella che un hacker reale possa disporre. La sicurezza è misurata tramite la teoria della complessità.

Crittoanalisi: insieme di tecniche che vogliono testare un sistema di sicurezza cercando di recuperare del testo in chiaro a partire da un messaggio cifrato senza possedere la chiave.

Oss: solitamente l’oggetto è più di un messaggio.

Approcci tipici: brute force, attacchi crittoanalitici.

Brute force: è sempre possibile, il suo costo varia al variare della dimensione della chiave che si deve trovare e assume che il testo in chiaro sia noto o riconoscibile.

Key size (bits)	Number of keys	Time required at 1 decryption/ μ s	Time required at 10^6 decryptions/ μ s
32	$2^{32} = 4.3 \times 10^9$	$2^{31} \mu s = 35.8 \text{ min}$	2.15 ms
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55} \mu s = 1142 \text{ years}$	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127} \mu s = 5.4 \times 10^{24} \text{ years}$	$5.4 \times 10^{18} \text{ years}$
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167} \mu s = 5.9 \times 10^{36} \text{ years}$	$5.9 \times 10^{30} \text{ years}$
26 chars (permut.)	$26! = 4 \times 10^{26}$	$2 \times 10^{26} \mu s = 6.4 \times 10^{12} \text{ years}$	$6.4 \times 10^6 \text{ years}$

Attacco crittanalitico: all'attaccante è noto l'algoritmo usato, questo tipo di attacco è il peggiore caso realistico per un sistema. L'algoritmo dovrebbe essere pubblicato per permettere una più semplice valutazione dell'effettiva sicurezza (in contrasto con la sicurezza tramite segretezza).

Il modello di attacco può essere diviso in tre punti:

- **input:** ciò che l'attaccante sa dall'inizio (es. chiave pubblica, algoritmo usato, etc...);
- **oracolo:** ciò che l'attaccante ottiene durante l'attacco (diverse informazioni in base al bersaglio scelto, es. testo in chiaro se si è attaccato un testo cifrato);
- **output:** ciò che l'attaccante vuole ottenere (es. chiave privata, informazioni in chiaro), egli vince se ottiene ciò che desidera.

Tipi di attacco:

- **dato un testo cifrato:** ottenere il testo in chiaro o un algoritmo per calcolare il testo in chiaro.
- **dato una parte di testo in chiaro e il testo cifrato:** dedurre una "chiave inversa" per ottenere il testo in chiaro dato del testo cifrato.
- **dato una parte di testo in chiaro a scelta e il testo cifrato:** come sopra, ma la parte in chiaro è scelta dal crittoanalista.
- **testo in chiaro scelto adattivamente:** l'attaccante può non solo scegliere in testo in chiaro, ma anche scegliere il testo in chiaro a lui più conveniente per la decodifica in base ai risultati delle cifrature.
- **testo cifrato scelto:** l'hacker può scegliere tra diversi testi cifrati da decifrare e ottenere accesso al testo decifrato.

Come verificare che un sistema è sicuro:

1. specificare un oracolo (un tipo di attacco),
2. definire di che cosa ha bisogno l'attaccante per vincere il gioco (es. un condizione sul suo output),
3. il sistema si può definire sicuro se ogni efficiente avversario vince il gioco con una probabilità trascurabile (sufficientemente bassa).

Definizioni in merito alla cifratura/decifratura:

- A, alfabeto, insieme finito.
- M è contenuto in A^* è l'insieme dei caratteri che compongono tutti i possibili messaggi. M appartiene a M e il testo in chiaro (messaggio).

- C è l'insieme dei caratteri che compongono tutti i possibili testi cifrati, il cui alfabeto può differire da quello di M .
- K denota l'insieme dei caratteri che compongono tutte le possibili chiavi.
- Per ogni e appartenente a K determina una funzione bigettiva da M a C , denotata come E_e essa è la funzione di cifratura (o trasformazione).
- Per ogni d appartenente a K determina D_d , funzione bigettiva da C a M detta funzione di decifratura.
- Applicare E_e è detto cifratura, applicare D_d è decifratura.
- Uno schema di cifratura (o cifrario) consiste in un insieme di funzioni di decifratura $\{D_d: d \text{ appartenente a } K\}$ e funzioni di cifratura $\{E_e: e \text{ appartenente a } K\}$ con la proprietà che per ogni e appartenente a K esiste un unico d appartenente a K tale che $D_d = E_e^{-1}$ (es. $D_d(E_e(m)) = m$ per tutti gli m appartenenti a M).
- le chiavi e e d formano la coppia delle chiavi (anche rappresentata come (e,d)) possono essere identiche (es. la chiave simmetrica).
- Per costruire un cifrario occorre fissare M, C, K così come le funzioni di cifratura e decifratura.

Let $\mathcal{M} = \{m_1, m_2, m_3\}$ and $\mathcal{C} = \{c_1, c_2, c_3\}$. There are $3! = 6$ bijections from \mathcal{M} to \mathcal{C} . The key space $\mathcal{K} = \{1, 2, 3, 4, 5, 6\}$ specifies these transformations.

E_1			E_2			E_3		
m_1	\rightarrow	c_1	m_1	\rightarrow	c_1	m_1	\rightarrow	c_2
m_2	\rightarrow	c_2	m_2	\rightarrow	c_3	m_2	\rightarrow	c_1
m_3	\rightarrow	c_3	m_3	\rightarrow	c_2	m_3	\rightarrow	c_3

E_4			E_5			E_6		
m_1	\rightarrow	c_2	m_1	\rightarrow	c_3	m_1	\rightarrow	c_3
m_2	\rightarrow	c_3	m_2	\rightarrow	c_1	m_2	\rightarrow	c_2
m_3	\rightarrow	c_1	m_3	\rightarrow	c_2	m_3	\rightarrow	c_1

Suppose Alice and Bob agree on the transformation E_6 . To encrypt m_1 , Alice computes $E_6(m_1) = c_3$. Bob decrypts c_3 by reversing the arrows on the diagram for E_6 and observing that c_3 points to m_1 .

Cifratura a chiave simmetrica

- Uno schema di cifratura è a chiave simmetrica se per ogni coppia (e,d) è computazionalmente facile determinare d conoscendo solo e e viceversa. In pratica $e = d$.
- Anche detto: a chiave condivisa o singola.
- Mittente e destinatario condividono la chiave in comune.
- tutti gli algoritmi di cifratura classici sono simmetrici (il solo tipo di cifratura prima dell'invenzione della chiave pubblica nel 1970).

DEF Cifrario a blocchi: la cifratura del messaggio avviene dopo averlo diviso in blocchi di lunghezza fissata t , solo in seguito alla suddivisione il testo in chiaro viene criptato un blocco alla volta.

DEF Cifrario di flusso: un cifrario a blocchi in cui la lunghezza del blocco è 1.

DEF Codice: cifrario che divide il testo in chiaro in blocchi di lunghezza variabile anziché fissa.

Cifrari di sostituzione:

- Cifrario di Cesare: ogni carattere del testo in chiaro è rimpiazzato da un carattere di tre posizioni più a destra e modulo 26.
- ROT13: sposta ogni lettera di 13 posti.
- Alfanumerico: sostituisce i numeri con le lettere.

Cifrario di sostituzione monoalfabetico:

Si basa sull'idea del cifrario di cesare, ma permette una sostituzione arbitraria.

Sia K l'insieme di tutte le permutazioni dell'alfabeto A^* . Definiamo per ogni e appartenente a K una funzione di cifratura E_e sulle stringhe $m=m_1m_2\dots m_n$ appartenenti a M come

$$E_e(m) = e(m_1)e(m_2)\dots e(m_n) = c_1c_2\dots c_n = c$$

per decifrare c , occorrerà eseguire la permutazione inversa $d = e^{-1}$ ossia

$$D_d = d(c_1)d(c_2)\dots d(c_n) = m$$

E_e è un cifrario a sostituzione semplice o monoalfabetico.

Un esempio di cifrario a sostituzione semplice è il cifrario affine.

Nonostante lo spazio delle chiavi sia tipicamente molto vasto (con 26 lettere possiamo avere $26!$ possibili chiavi) si può riuscire a decifrare un testo cifrato con facilità basandosi sull'analisi della frequenza dei simboli, ad esempio se sappiamo che un testo è scritto in inglese sapremo che statisticamente conterrà più 'e' e 't' dunque se trovassimo una coppia di lettere che si ripetono più spesso di altre, queste potrebbero proprio essere 'e' e 't'.

Cifrario di sostituzione omofonico: ogni simbolo dell'alfabeto A viene rimpiazzato da un simbolo preso da un insieme di simboli $H(a)$, il quale viene scelto tramite una funzione nell'insieme di $H(a)$, preso un simbolo di A restituisce un simbolo di $H(a)$. Per decifrare c (simbolo cifrato) occorre sapere a di A tale appartenga a $H(a)$. Le chiavi del cifrario sono gli insiemi $H(a)$. Permette una più complessa analisi della frequenza, ma i dati inviati crescono velocemente ed è più difficile da decifrare.

Cifrario a sostituzione polialfabetica: nasconde la distribuzione usando una famiglia di mappature. Esso è un cifrario a blocchi su un alfabeto A dove:

- K consiste in una sequenza di permutazioni su A nella forma (e_1, \dots, e_t) ;
- la cifratura di m (blocco del messaggio) con e è $E_e(m) = c_1c_2\dots$, dove $c_i = e_{(i \bmod t)}(m_i)$ per $i = 1, 2, \dots$ (ovvero ogni carattere viene codificato con una chiave diversa)
- la decifratura è $d = (e_1^{-1}, e_2^{-1}, \dots)$.

Il testo è diviso in blocchi, ogni carattere del blocco usa una chiave diversa.

Es. cifrario di Vigenère.

One-time pad (Cifrario di Vernam): è un cifrario a flusso su un alfabeto $\{0, 1\}$ che prende il messaggio ed effettua una somma diretta (ADB bitwise) con una stringa binaria casuale che sarà la chiave.

Siccome ogni chiave è equamente probabile, anche ogni testo in chiaro lo è.

In teoria è un tipo di sicurezza incondizionata se non viene riusata la stessa chiave.

Le comunicazioni tra Mosca-Washington erano prima codificate così.

Il problema di questo cifrario è lo scambio e la sincronizzazione sicuri di chiavi lunghe.

Se viene utilizzata una chiave usata in precedenza e un terzo soggetto malevolo legge due scambi di messaggi avvenuti cifrando i testi con questa, può fare come segue per ottenere entrambi i messaggi decifrati:

$$C_1 \oplus C_2 = (S \oplus M_1) \oplus (S \oplus M_2) = M_1 \oplus M_2$$

DEF Malleabile: Una funzione di codifica $e(k, m)$ si dice malleabile sse esistono due funzioni

$$f(x) \text{ e } g(x) \text{ t.c. } f(e(k, m)) = e(k, g(m))$$

es. $e(k, m) = k + m$ è malleabile, infatti, sia $f(x) = g(x) = n + x$ per ogni n di dimensione appropriata, si ha che:

$$f(e(k, m)) = n + (k + m) = k + (n + m) = e(k, g(m))$$

Corollario: è possibile trasformare un testo cifrato in un altro conoscendo il messaggio in chiaro del primo e potendo scegliere quello del secondo anche se non si conosce la chiave? Sì come segue:

$$C_1 \oplus (M_1 \oplus M_2) = (K \oplus M_1) \oplus (M_1 \oplus M_2) = K \oplus M_2$$

Cifrari a trasposizione: per ogni blocco di dim t , sia K un insieme di permutazioni su $\{1, \dots, t\}$. per ogni e appartenente a K e m appartenente a M .

$$E_e = m_{e(1)} m_{e(2)} \dots m_{e(t)}$$

L'insieme di tutte le trasformazioni è detto cifrario di trasposizione.

Per decifrare c occorre eseguire $D_d(c) = c_{d(1)} c_{d(2)} \dots c_{d(t)}$ (dove d è la permutazione inversa).

In poche parole: sposti le lettere nel testo cifrato in un posto diverso da quello in chiaro; le lettere sono le stesse, solo in posizione "sbagliata".

Cifrari composti: questi sono basati sulla combinazione di sostituzione e trasposizione.

- Una coppia di sostituzioni è più complicata di una.
- Una coppia di trasposizioni è più complicata di una.
- Una sostituzione seguita da una trasposizione è più complicata delle precedenti soluzioni.

Quindi vengono messi in sequenza combinazioni di sostituzioni-trasposizioni.

Iniziano ad essere difficoltosi da gestire manualmente, dunque, con l'avvento di questi cifrari vengono inventate le prime macchine cifratrici.

(5_crypto-symm.pdf)

Cifrari a blocchi vs cifrari a flusso

- i cifrari a blocchi processano il messaggio in blocchi, ognuno dei quali è de/cryptato;
- come una sostituzione su un alfabeto molto grande
- 64 bit o più
- cifrari a stream processano il messaggio un bit o byte alla volta quando de/cryptano;
- ad oggi sono più diffusi i cifrari a blocchi
- possono essere usati su una più vasta gamma di applicazioni

Conclusione: ad oggi meglio i cifrari a blocchi.

Il cifrario a blocchi ideale è simile ad un enorme cifrario a sostituzione, necessiterebbe una tabella con 2^n voci per un blocco di n bit e con una dimensione di chiave di $n \cdot 2^n$ con 2^n !

trasformazioni possibili. Nel cifrario ideale l'informazione statistica sul testo in chiaro dovrebbe essere persa, ma è infattibile.

Struttura della rete di Feistel.

Idea: si approssima il cifrario a blocchi ideale utilizzando il concetto di cifrario prodotto. Combinando insieme più cifrari semplici, otteniamo un cifrario più robusto.

In pratica: costruisco un cifrario con una lunghezza k , e i blocchi saranno di lunghezza n , ciò mi permetterà di avere un totale di 2^k trasformazioni anziché 2^n !

Molti dei cifrari a blocchi simmetrici si basano su quest'idea per permettere una decifratura più efficiente del messaggio.

Claude Shannon ha introdotto l'idea della rete di sostituzione-permutazione (S-P) nel '49, sulla quale si basano i moderni cifrari a blocchi.

La rete S-P si basa sul sostituire (S-box, "mescola" i bit in input) e sul permutare (P-box, diffonde i bit attraverso gli input delle S-box) messaggio e chiave.

Siccome il cifrario deve cercare di oscurare le statistiche del messaggio originale (OTP riesce a pieno), Shannon suggerisce le combinazioni di S-P per ottenere:

- diffusione: dissipare la struttura delle statistiche del testo in chiaro su tutto il testo cifrato;
- confusione: rendere la corrispondenza tra testo cifrato e chiave il più complessa possibile.

Feistel con il suo cifrario (basato sul concetto di cifrario prodotto invertibile) esegue le seguenti operazioni:

- divide il blocco in input in 2 parti;
- procede con più iterazioni nelle quali:
 - esegue una sostituzione nella prima metà dei dati;
 - esegue una funzione F che prende in input la seconda metà e una sottochiave,
 - dunque effettua una permutazione scambiando le metà.

Così facendo implementa il concetto S-P di Shannon.

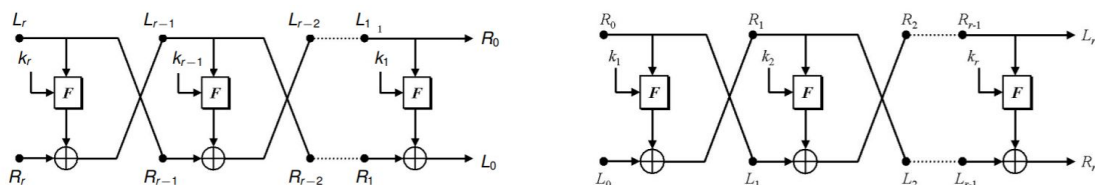
La funzione F può essere una rete S-P o un qualsiasi cifrario.

La cifratura e decifratura sono strutturalmente identiche, sebbene la sottochiave usata durante ogni round è presa al contrario durante la decifratura.

Nello specifico: l'input della decodifica è la coppia (R_r, L_r) anziché (R_0, L_0) , l' i -esima sottochiave per la decodifica è k_{r-i+1} , anziché k_i e all' i -esimo round nella decodifica avremo (R_{r-i}, L_{r-i}) , mentre nella codifica avremo (R_i, L_i) .

(decifratura)

(cifratura)

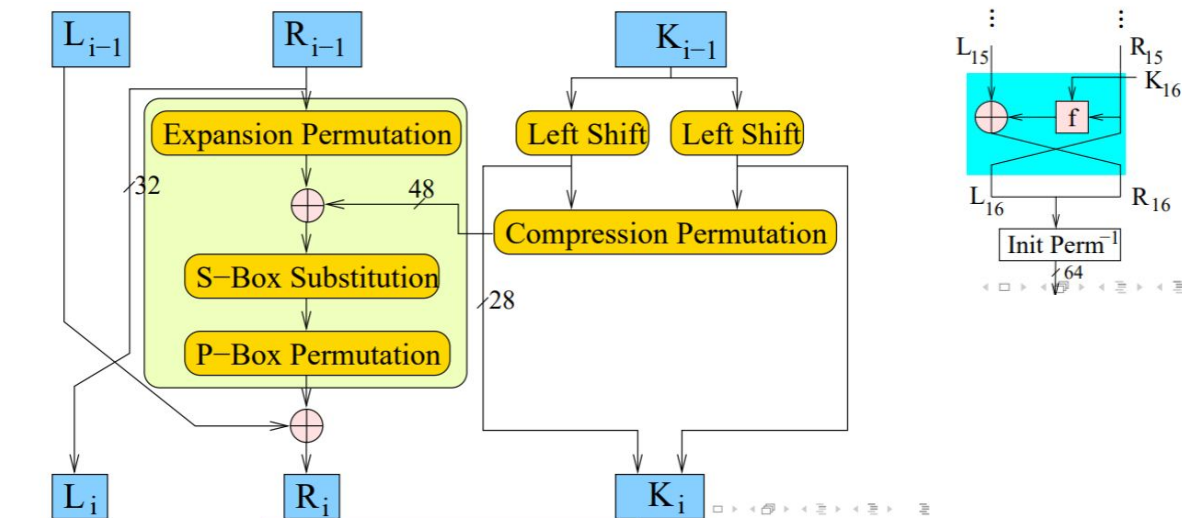


Riassumendo il cifrario di Feistel si compone di: dimensione del blocco, dimensione della chiave, numero di iterazioni, algoritmo di generazione di sotto chiave, funzione F del round software veloce di de/cifratura e un'analisi semplice.

DEF Data Encryption Standard (DES): è un algoritmo di cifratura a blocchi che si compone di blocchi da 64 bits e usa chiavi a 56 bits (espresso con 64 bits, per il controllo di parità). Ampiamente usato nelle applicazioni bancarie con l'estensione triple-DES per superare il problema delle chiavi corte.

Come si articola:

- 16 iterazioni del cifrario di Feistel + generatore di sottochiavi,
- il generatore di sottochiavi genera K_i a partire da K ,
- una permutazione viene effettuata prima delle iterazioni e una permutazione inversa al termine di queste,
- F consiste in 2 permutazioni (P-box) e una sostituzione (S-box).



Il DES dispone di un buon effetto valanga (proprietà chiave di un algoritmo di cifratura), in quanto, ha la proprietà che se viene cambiato un bit nella chiave o nell'input, il risultato sarà modificato per circa la metà dei suoi bits.

La chiave del DES a 56 bits ha $2^{56} = 7.2 \cdot 10^{16}$ valori, ma verso la fine degli anni 90 sono riusciti a decifrare un messaggio così crittografato in sole 22 ore con la forza bruta (per la crescente potenza di calcolo dei PC). Dunque sono necessarie delle alternative al DES.

Ad oggi sono presenti diversi attacchi analitici al DES che utilizzano una profonda comprensione del cifrario come ottenere informazioni sulla cifratura e parte o tutti i bit delle sottochiavi. Questi attacchi generalmente sono statistici e includono: crittoanalisi differenziale e lineare e attacchi relativi alla chiave.

Il timing attack colpisce l'effettiva implementazione del cifrario e usa la conoscenza dell'implementazione per derivare informazioni circa tutti/alcuni bits sottochiavi.

Nello specifico usa il fatto che il calcolo può variare a seconda del valore degli input. Particolarmente problematico sulle tessere magnetiche.

Varianti: Double DES (112 bits key), Triple DES (se k_1, k_2, k_1 112 bits key) (3DES)
 $C = E_{k_1}(D_{k_2}(E_{k_3}(P)))$ (se $k_1 = k_2 = k_3$ è come DES), AES.

DEF Advanced Encryption Standard (AES):

- 128 bits dimensione del blocco,
- 3 lunghezze differenti per la chiave 128, 192 e 256 bits.

Veloce sia per il software che per l'hardware, inoltre AES ha soppiantato globalmente DES.

AES si basa sulla rete S-P ma non usa la rete Feistel.

Sebbene AES abbia una dimensione del blocco fissa di 128 bit e una dimensione della chiave di 128, 192 o 256 bit, Rijndael funziona con blocchi e dimensioni di chiavi che possono essere qualsiasi multiplo di 32 bit, entrambi con un minimo di 128 e un massimo di 256 bit.

La dimensione della chiave utilizzata per una crittografia AES specifica il numero di ripetizioni di cicli di trasformazione che convertono il testo in chiaro in testo cifrato:

- 10 cicli di ripetizione per chiavi a 128 bit.
- 12 cicli di ripetizione per chiavi a 192 bit.
- 14 cicli di ripetizione per chiavi a 256 bit.

Modalità di funzionamento nel caso in cui la dimensione del messaggio ecceda la dimensione del blocco, vediamo due modalità:

Electronic Codebook Mode: divide il messaggio in m blocchi ognuno cifrato individualmente. Ha 2 principali limitazioni: i blocchi di testo cifrato identici rappresentano lo stesso testo in chiaro e non si può sapere se il testo è stato corrotto (modificato, duplicato o cancellato).

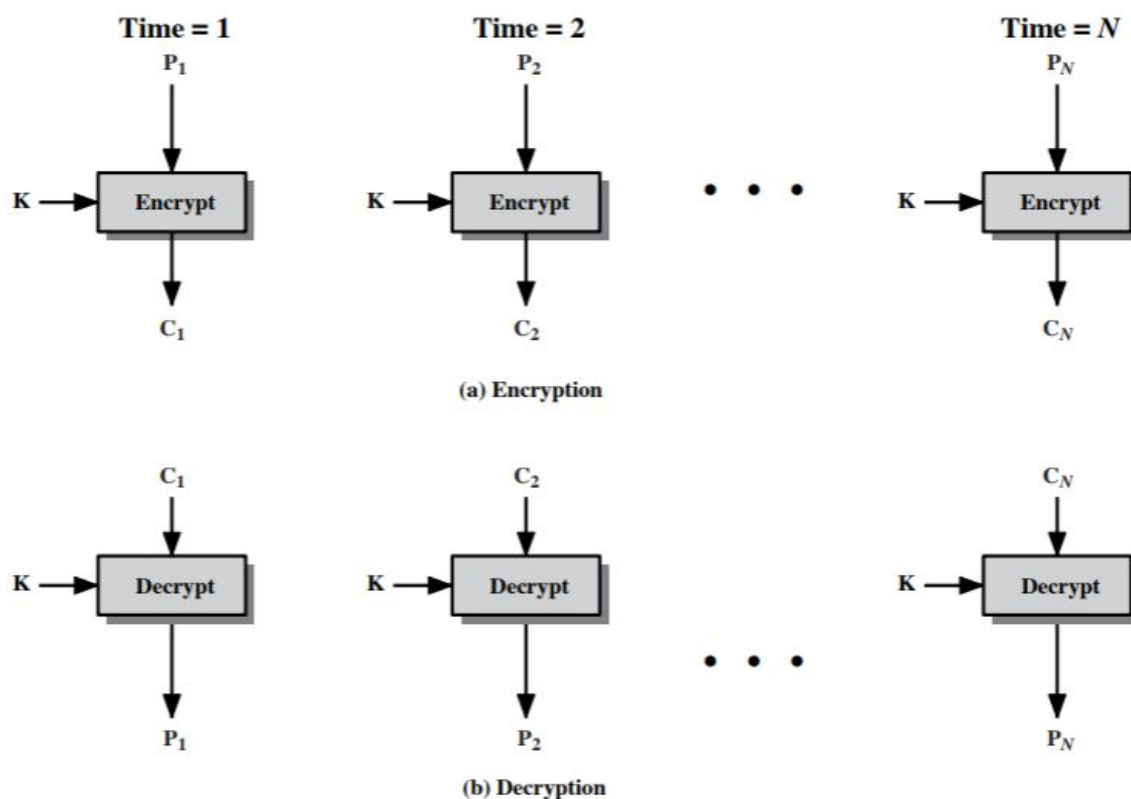


Figure 6.3 Electronic Codebook (ECB) Mode

Cipher-block chaining: l'input del cifrario è lo XOR tra blocco testo in chiaro e testo cifrato in precedenza.

Dati C_0 =vettore vuoto e $i = 1, \dots, m$ avremo:

- cifratura $\Rightarrow C_i = E_K(P_i \oplus C_{i-1})$
- decifratura $\Rightarrow P_i = C_{i-1} \oplus D_K(C_i)$

Proprietà:

- due blocchi di testo in chiaro uguali formano del testo cifrato differente,
- Catena di dipendenza: C_j dipende da tutti i testi in chiaro precedenti,
- se avviene un errore in un blocco, i blocchi successivi vengono decrittati correttamente.

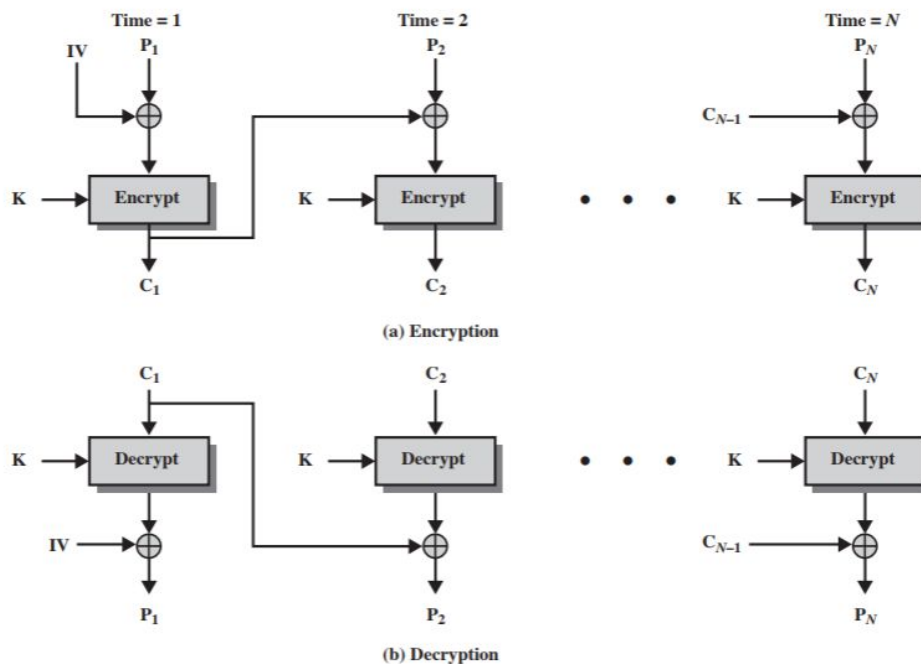
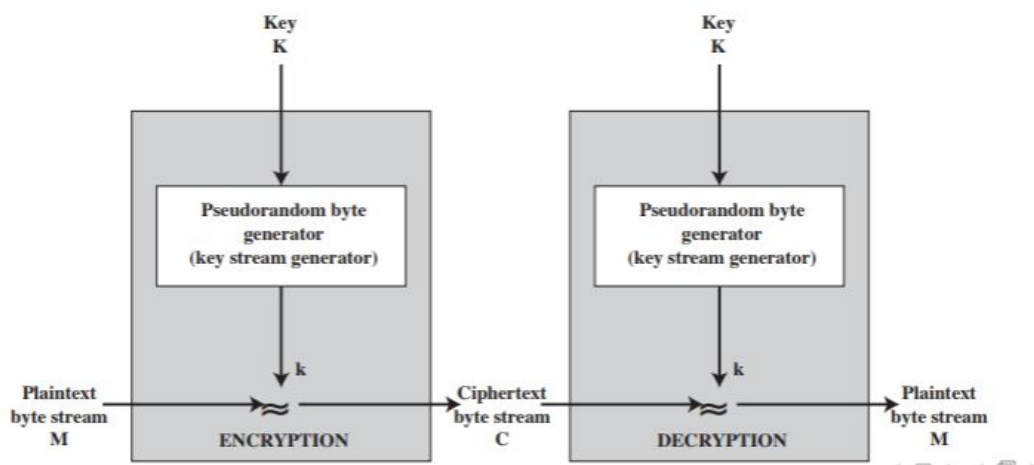


Figure 6.4 Cipher Block Chaining (CBC) Mode

Stessa idea del cifrario Vernam ma usa un generatore pseudocasuale (al posto di un generatore veramente casuale), i.e.

$$E_{k1 \dots kn}(m_1 \dots m_n) = (m_1 \oplus k_1) \cdot \dots \cdot (m_n \oplus k_n)$$

$$D_{k1 \dots kn}(c_1 \dots c_n) = (c_1 \oplus k_1) \cdot \dots \cdot (c_n \oplus k_n)$$



I cifrari a flusso sono solitamente più veloci e di facile implementazione di quelli a blocchi, inoltre con un appropriato generatore pseudocasuale di chiavi un cifrario a flusso è più sicuro di uno a blocchi con una chiave di confrontabile lunghezza (in quest'ultimo le chiavi possono essere riusate).

Posizionamento della crittografia

Nel modello TCP/IP abbiamo la crittografia link ai livelli 1 e 2 (fisico e datalink) e la crittografia end-to-end ai livelli 3, 4 e 5 (rete, trasporto, applicazione).

Come saliamo nel modello TCP/IP meno informazione va criptata, c'è più sicurezza, ma è più complessa con più elementi è chiavi.

Quando usiamo la cifratura end-to-end gli headers non devono essere criptati per poter inviare il messaggio correttamente (routing), quindi, sebbene l'informazione inviata sarà criptata, il modello del traffico di flusso non lo sarà.

Idealmente noi vorremmo la end-to-end per proteggere i messaggi su tutto il percorso e fornire un meccanismo di autenticazione e la protezione link per evitare che venga spiato il nostro traffico.

Link e End-to-End a confronto

- la sicurezza nei sistemi finali ed intermedi:
 - link: non cripta il testo messaggio nella rete
 - end-to-end: cripta il messaggio nella rete
- ruolo dell'utente:
 - link: essa è trasparente all'utente (non la vede) e può essere fatta attraverso dell'hardware, inoltre o viene cifrato tutti i messaggi che circolano sulla rete o nessuno.
 - end-to-end: è l'utente che deve applicare questo tipo di crittografia, deve deciderne l'algoritmo, lo schema di cifratura e può decidere se cifrare o meno ogni singolo messaggio.
- difficoltà implementative:
 - link: richiede una coppia di chiavi per ogni nodo intermedio che incontra e una per mittente-intermedio e intermedio-destinatario, permette l'autenticazione dell'host.
 - end-to-end: richiede una coppia di chiavi (mittente-destinatario), permette l'autenticazione dell'utente.

DEF L'analisi del traffico: monitoraggio del flusso di comunicazione delle parti, utile in campo militare e commerciale può essere usato per creare un canale segreto.

La cifratura a livello fisico con copre l'analisi del volume del traffico (può essere appannata a costo di un continuo traffico).

Distribuzione delle chiavi

Gli schemi simmetrici necessitano che le parti condividano una chiave segreta. Il problema è come distribuire questa chiave in sicurezza, infatti, spesso un errore della sicurezza di un sistema è dovuto alla riuscita estrapolazione di informazioni da uno schema di distribuzione delle chiavi.

Alternative di distribuzione:

- A invia fisicamente la chiave a B,
- una terza parte invia la chiave ad A e a B,
- se A e B hanno già comunicato in precedenza, possono usare la precedente chiave per cifrare quella nuova,
- se A e B hanno un canale di comunicazione sicuro con C, egli può intermediare lo scambio della chiave tra A e B.

Tipicamente viene usato una gerarchia delle chiavi:

DEF Chiave di sessione: usata per la cifratura dei dati tra gli utenti in una sessione logica, dopodichè scartata.

DEF Chiave master: usata per cifrare lo scambio di chiavi tra l'utente e il centro di distribuzione chiavi (KDC).

Problemi di distribuzione chiave:

- Gerarchie di KDC sono necessarie per reti di grandi dimensioni, ma devono fidarsi l'una dell'altra,
- La durata delle chiavi di sessione dovrebbe essere limitata per una maggiore sicurezza,
- Usa la distribuzione automatica delle chiavi per conto degli utenti, ma deve essere un sistema fidato (il KDC deve essere autorevole),
- Uso della distribuzione delle chiavi decentralizzata,
- Controllo dell'utilizzo delle chiavi.

(6_crypto-pk.pdf)

Crittografia a chiave pubblica

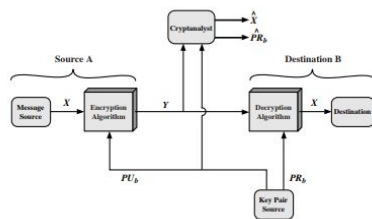
Siano $\{E_e : e \in K\}$ e $\{D_d : d \in K\}$ base per uno schema di cifratura.

Consideriamo la coppia (E_e, D_d) in cui conoscendo E_e è impossibile, dato un $c \in C$, trovare un $m \in M$ tc $E_e(m) = c$. Questo implica che è impossibile dato determinare d dato e . E_e è una funzione unidirezionale (facile da calcolare in un verso, infattibile l'inversa).

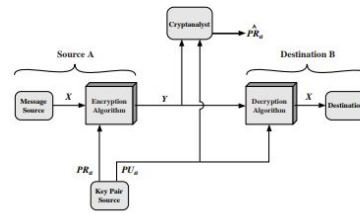
La chiave pubblica e può essere un'informazione nota a chiunque.

Cifratura tradizionale (simmetrica)	Cifratura chiave pubblica (asimmetrica)
<p>Necessario perché funzioni:</p> <ol style="list-style-type: none"> 1. lo stesso algoritmo con la stessa chiave per de/cifratura, 2. mittente e destinatario devono condividere algoritmo e chiave. <p>Necessario per sicurezza:</p> <ol style="list-style-type: none"> 1. la chiave dev'essere segreta 2. deve essere impossibile o quasi decifrare il messaggio se non si hanno informazioni riguardo 3. la conoscenza dell'algoritmo + esempi di testo cifrato devono bastare per determinare la chiave 	<p>Necessario perché funzioni:</p> <ol style="list-style-type: none"> 1. 1 algoritmo per la de/cifratura con una coppia di chiavi una per cifrare l'altra per decifrare, 2. mittente e destinatario devono possedere una chiave a testa per coppia di chiavi (non la stessa) <p>Necessario per la sicurezza:</p> <ol style="list-style-type: none"> 1. una chiavi deve essere segreta, 2. dev'essere impossibile decifrare il messaggio se non si hanno info, 3. conoscere l'algoritmo + una chiave + dei testi cifrati non bastano per determinare l'altra chiave

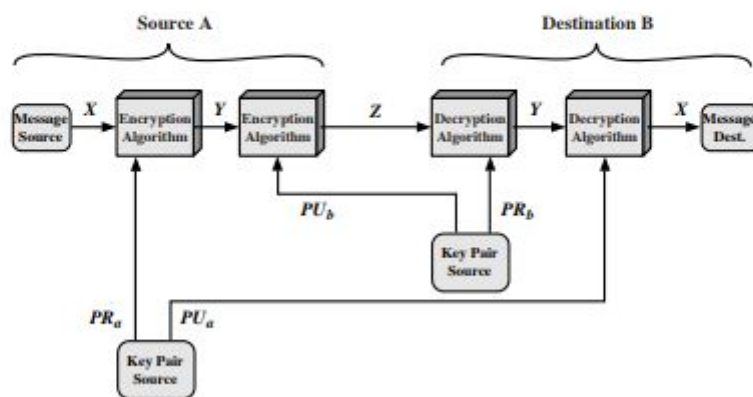
Public-Key Cryptosystem: Secrecy



Public-Key Cryptosystem: Authentication



Public-Key Cryptosystem: Secrecy and Authentication



Requisiti per la crittografia a chiave pubblica:

1. Dev'essere facile per B generare una coppia (PR_B, PU_B) .
2. Dev'essere facile per A conoscendo PU_B e M generare: $C = E(PU_B, M)$
3. Dev'essere facile per B decifrare C usando PR_B per recuperare M
 $M = D(PR_B, C) = D(PR_B, E(PU_B, M))$
4. Dev'essere impossibile per un attaccante, ricavare PR da PU.
5. Dev'essere infattibile per un attaccante, decifrare C conoscendo PU.
6. (utile ma non necessario) le chiavi possono essere applicate in entrambi gli ordini
 $M = D(PU_b, E(PR_b, M)) = D(PR_b, E(PU_b, M))$

Essendo dei requisiti difficili, pochi algoritmi hanno ricevuto un'ampia accettazione (es. RSA)

DEF Funzione unidirezionale: è una funzione "facile da calcolare" ma "difficile da invertire".

DEF Funzione Botola: è una funzione facile da computare in una direzione, ma difficile da calcolare nella direzione opposta se non si conoscono determinate informazioni, chiamate appunto botole.

Attacchi possibili:

- forza bruta: come contromisura si usano chiavi più grandi, ma non troppo, per non complicare le operazioni di cifratura e decifratura;

- calcolare la chiave privata a partire dalla chiave pubblica: non ci sono prove che questo attacco sia impossibile! (anche per RSA);
- Attacco di messaggio probabile: per piccoli messaggi inviati con la chiave pubblica, l'attaccante può provare tutte le combinazioni di messaggi brevi, cifrandoli con la chiave pubblica; quando genera una corrispondenza tra il messaggio da lui cifrato e da lui intercettato ha scoperto il testo in chiaro (pericoloso perché il breve messaggio potrebbe essere una chiave per un DES). Come contromisura si possono aggiungere dei bit casuali in fondo al testo in chiaro (per appesantirlo e rendere la ricerca della corrispondenza più onerosa e meno probabile).

DEF Fattorizzazione a numeri primi: scrive un numero come prodotto di potenze di numeri primi (es $10 = 2 * 5$).

DEF Numeri relativamente primi: sono due numeri che non hanno divisori in comune a parte 1 (es 8 e 15).

DEF Massimo comune divisore: confronto tra i fattori primi di 2 numeri usando la potenza massima (es $300 = 2^2 * 3^1 * 5^2$, $18 = 2^1 * 3^2$ quindi $MCD(18, 300) = 2^1 * 3^1 * 5^0 = 6$).

DEF Congruo modulo n: se $a, b \in \mathbb{Z}$ sono congruenti modulo n, se $a \% n = b$.

DEF Funzione di Eulero: è una funzione definita, per ogni intero positivo n, come il numero degli interi compresi tra 1 e n che sono coprimi con n. (es $\phi(8)=4$ poiché i numeri coprimi di 8 sono quattro: 1, 3, 5, 7).

Teorema di Eulero: afferma che se $n \in \mathbb{N}$, a è coprimo rispetto ad n, allora:

$a^{\phi(n)} \equiv 1 \pmod n$ dove $\phi(n)$ indica la funzione phi di Eulero e \equiv la relazione di congruenza modulo n.

Algoritmo RSA

Prende il nome dai suoi 3 inventori. L'algoritmo a chiave pubblica più usato.

La sicurezza si basa sulla difficoltà di fattorizzare grandi numeri. Le chiavi sono funzioni di coppie di numeri (100 o più cifre) primi.

Procedimento:

- sia n un numero noto a mittente e destinatario.
- il testo in chiaro viene diviso in blocchi di dimensione: parte intera superiore($\log_2(n)$) bits, così ciascun blocco rappresenta un numero M tc $M < n$.
- de/cifratura sono così definite:
 $C = M^e \pmod n$
 $M = C^d \pmod n = (M^e)^d \pmod n = M^{ed} \pmod n$
per qualche valore (propriamente scelto) di e, d.
- è un algoritmo di chiave pubblica con:
 - $PU = (e, n)$,
 - $PR = (d, n)$

Perché possa lavorare, ha bisogno dei seguenti requisiti:

1. è possibile trovare i valori e, d, n tc: $M^{ed} \pmod n = M$ per tutti gli $M < n$
2. è relativamente facile calcolare $M^e \pmod n$, $C^d \pmod n$ per tutti gli $M < n$
3. è infattibile determinare d dati e, n.

DEF inverso moltiplicativo (?reciproci?): x, y sono reciproci mod r sse $xy \pmod r = 1$.

Teorema Correttezza RSA: se d, e sono reciproci mod $\phi(n)$, allora $M^{ed} \pmod n = M \quad \forall M < n$
es. se $ed \pmod \phi(n) = 1$, allora $M^{ed} \pmod n = M$ per tutti gli $M < n$.

Lemma siano p, q numeri primi, $n = pq$. Se d, e sono reciproci mod $\phi(n)$, allora $M^{ed} \equiv_p M$, ma anche $M^{ed} \equiv_q M$, es $(M^{ed} - M)$ è multiplo sia di p sia di q .

Prova della correttezza: se d, e sono reciproci mod $\phi(n)$. per il lemma $(M^{ed} - M)$ è multiplo di p, q . Siccome p, q sono primi, allora $(M^{ed} - M)$ è anche un multiplo di pq quindi di n .

Proof of Lemma.

Let d and e are multiplicative inverses mod $\phi(n)$. Then, there exists an integer k such that $ed = k\phi(n) + 1$.

We must prove that $M^{ed} = M^{k\phi(n)+1} \equiv_p M$. Two cases:

Case 1: M and p are relatively prime.

$$\begin{aligned} M^{k\phi(n)+1} \bmod p &= M \cdot M^{k(p-1)(q-1)} \bmod p \\ &= M \cdot (M^{p-1})^{k(q-1)} \bmod p \\ &= M \cdot (M^{\phi(p)})^{k(q-1)} \bmod p \text{ since } p \text{ is prime} \\ &= M \cdot (1)^{k(q-1)} \bmod p \text{ by Euler Theorem} \\ &= M \bmod p \end{aligned}$$

Case 2: M and p are not relatively prime. Then M is a multiple of p , i.e. $M \bmod p = 0$ and hence $M^{k\phi(n)+1} \bmod p = M \bmod p$. □

RSA algorithms: Example

- **Generate a public/private key pair:**
 - ① Generate $p = 47, q = 71$
 - ② Compute $n = pq = 3337$ and $\phi = (p-1)(q-1) = 46 * 70 = 3220$
 - ③ Choose $e = 79$ (randomly in the interval $[1..3220]$)
 - ④ Compute d such that $79 \cdot d \bmod 3220 = 1$: $d = 1019$
 - ⑤ Public key $(e, n) = (79, 3337)$, private key $(d, n) = (1019, 3337)$
- **Encryption with key $(e, n) = (79, 3337)$**
 - ① Break message M into blocks, e.g. 688 232 687 966 668 ...
 - ② Compute $C_1 = 688^{79} \bmod 3337 = 1570, C_2 = \dots$
- **Decryption with key $(d, n) = (1019, 3337)$:**
 - ① Compute $M_1 = 1570^{1019} \bmod 3337 = 688, M_2 = \dots$

Sicurezza dell'RSA:

- calcolare la d secreta data la coppia (e, n)
 Difficile come la fattorizzazione. Se possiamo calcolare $n = qp$ allora possiamo sapere $\phi = (p-1)(q-1)$ e quindi d
 Non esiste un algoritmo polinomiale che lo calcoli, ma dati i progressi nella fattorizzazione, n dovrebbe avere almeno 1024 bits.
- Calcolare M_i , dato C_i e la coppia (e, n)
 non ci sono prove che sia necessario calcolare d , ovvero fattorizzare n
 I progressi nella teoria dei numeri potrebbe rendere RSA insicuro.

L'RSA è malleabile infatti:

RSA encryption, namely $E((e, n), M) = M^e \bmod n$ is clearly malleable.
Let $F(X) = X * (M_1^e \bmod n)$ for any given M_1 .

$$\begin{aligned} F(E((e, n), M)) &= E((e, n), M) * (M_1^e \bmod n) = \\ &= (M^e \bmod n) * (M_1^e \bmod n) = (M * M_1)^e \bmod n = \\ &= E((e, n), M * M_1) = E((e, n), G(M)) \end{aligned}$$

Per questo è usato con altri metodi per offuscare i dati.

Vedremo due algoritmi per la distribuzioni di chiavi asimmetriche:

- distribuzione di chiavi simmetriche con RSA:
 - cifratura con PU
 - decifratura con PR
 - Es. A sceglie una chiave segreta che cifra con la PU di B, la sessione successiva si baserà sulla chiave segreta.
 - Problema: se la PR di B è compromessa allora la chiave segreta può essere usata per osservare il traffico

- Scambio di chiavi Diffie-Hellman:

1. gli attori condividono un numero primo q e la sua radice primitiva α , possono essere entrambi di dominio pubblico;

2. A e B generano numeri casuali X_a e X_b entrambi $< q$;

3. A calcola $Y_A = \alpha^{X_a} \bmod q$, B calcola $Y_B = \alpha^{X_b} \bmod q$.

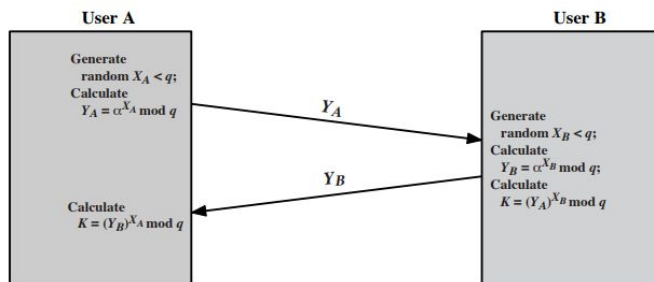
4. A e B si scambiano i risultati.

5. A calcola $K_A = Y_B^{X_a} \bmod q$, B calcola $K_B = Y_A^{X_b} \bmod q$.

Le chiavi sono uguali, $K_A = K_B$

La sicurezza dipende dalla difficoltà di elaborazione di log discreti.

$$\begin{aligned} K_A &= Y_B^{X_A} \bmod q \\ &= (\alpha^{X_B})^{X_A} \bmod q \\ &= (\alpha^{X_A})^{X_B} \bmod q \\ &= Y_A^{X_B} \bmod q = K_B \end{aligned}$$



La chiave segreta condivisa è creata dal nulla, non viene mai trasmessa.

Debolezza di Diffie-Hellman:

Le chiavi non sono autenticate e così è vulnerabile agli attacchi man-in-the-middle.

Soluzione: firmare gli esponenti, ma richiede delle chiavi condivise a monte.

Es.

- ① A transmits Y_A to B
- ② I intercepts Y_A and transmits Y_{D_1} to B. I also calculates $K_2 = (Y_A)^{X_{D_2}} \bmod q$.
- ③ B receives Y_{D_1} and calculates $K_B = (Y_{D_1})^{X_B} \bmod q$
- ④ B transmits Y_B to A
- ⑤ I intercepts Y_B and transmits Y_{D_2} to A. I calculates $K_1 = (Y_B)^{X_{D_1}} \bmod q$.
- ⑥ A receives Y_{D_2} and calculates $K_A = (Y_{D_2})^{X_A} \bmod q$

Now A and B think that they share a secret key, but instead **A shares secret key K_2 with I** and **B shares secret key K_1 with I.**

(7_msg-auth-dsign.pdf)

L'autenticazione dei messaggi si preoccupa di:

- proteggere l'integrità del messaggio,
- valida l'identità del mittente,
- è permette la non ripudiazione del mittente.

Ogni autenticazione di messaggio o meccanismo di firma digitale ha a che fare con una funzione di autenticazione per generare un autenticatore (es. valore usato per autenticare il messaggio).

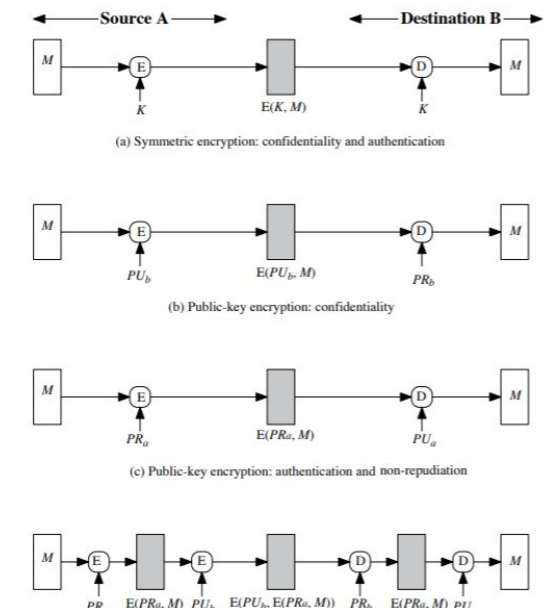
Vedremo le seguenti funzioni di autenticazione:

- cifratura del messaggio: il testo cifrato del messaggio intero serve da autenticatore,
- codice di autenticazione del messaggio: una funzione che prende il messaggio e una chiave segreta e genera un valore a lunghezza fissa che funge da autenticatore,
- funzione crittografica di hash: una funzione che mappa il messaggio qualsiasi lunghezza in un valore hash di lunghezza fissata, che fungerà da autenticatore.

Cifratura del messaggio

4 approcci possibili:

- $A \rightarrow B: E(K, M)$ (chiave simmetrica)
fornisce confidenzialità (solo A e B condividono K), autenticazione (il msg può arrivare solo da A), ma non prevede la firma (il destinatario può falsificare il messaggio e il mittente ripudiarlo).
- $A \rightarrow B: E(PU_B, M)$ (chiave pubblica per la confidenzialità)
fornisce confidenzialità (solo B può decifrarlo), ma non l'autenticazione (chiunque può cifrare un messaggio con la PU_B e fingersi A).
- $A \rightarrow B: E(PR_A, M)$ (chiave privata per l'autenticazione e firma)
fornisce autenticazione e firma (solo A può cifrare il messaggio), non la segretezza (chiunque può decifrarlo usando la PU_A)
- $A \rightarrow B: E(PU_B, E(PR_A, M))$ (PU e PR per autenticazione, firma e segretezza)
fornisce autenticazione, firma e confidenzialità.



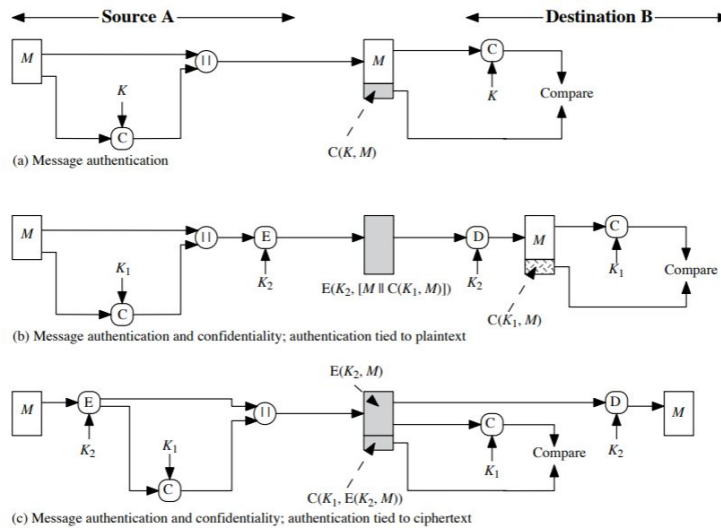
Per controllare che non vi siano stati errori durante la trasmissione del messaggio vengono aggiunti dei controlli, per esempio viene aggiunto in fondo al messaggio un 'checksum' prima della cifratura.

Codice di autenticazione del messaggio (MAC):

Una funzione MAC prende in input un messaggio di dimensione qualsiasi M e una chiave segreta K, e produce un output di dimensione fissa C(M, K).

è una funzione multi-a-uno, più messaggi possono condividere lo stesso MAC, ma trovarli è difficile.

C(M, K) è chiamato codice di autenticazione del messaggio (MAC) o checksum crittografico.



L'algoritmo di autenticazione sui dati basato sul DES è stato ampiamente usato negli anni, poi si è scoperta una grave falla di sicurezza.

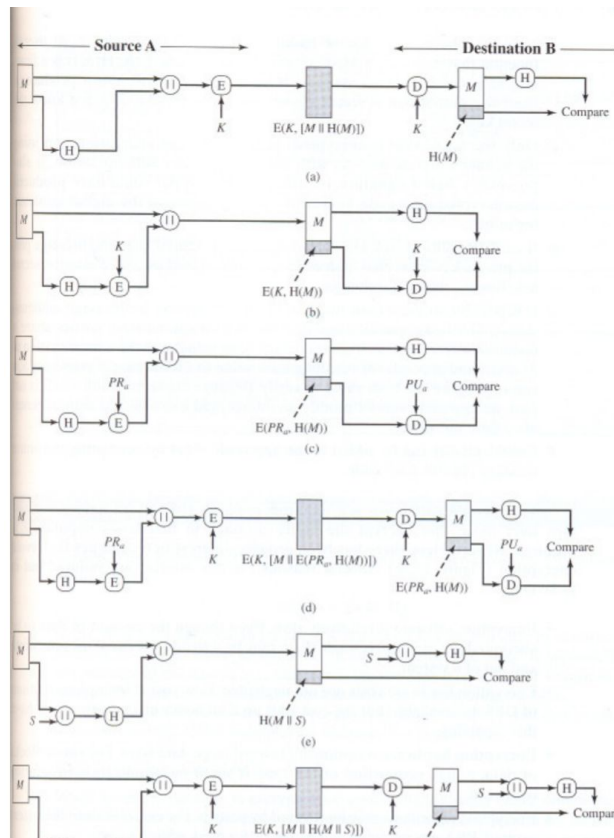
Oss: per applicare la funzione basta condividere chiave e funzione di checksum, il testo lo si "recupera" dal testo inviato o generato.

Funzione di hash crittografica:

Come nel caso del MAC, la funzione di hash accetta un msg di qualsiasi dimensione M in input e produce un output di dimensione fissa $H(M)$.

A differenza del MAC, non usa una chiave.

$H(M)$ è chiamato hash code, ma anche message digest.



Lo scopo è creare un'impronta digitale per un oggetto (es. file, msg, altro...)

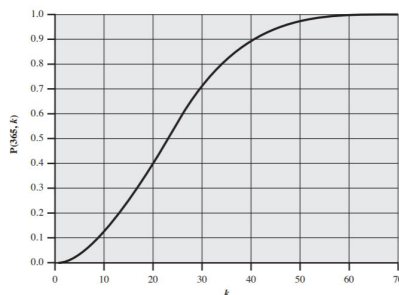
Proprietà che una funzione di hash crittografica dovrebbe possedere:

1. H può essere applicata a blocchi di dati di ogni dimensione,
2. H produce un output di dimensione fissata,
3. $H(x)$ è facile dato x ,
4. Dev'essere una funzione unidirezionale, dato y , è infattibile trovare x tc $H(x) = y$,
5. Resistenza alla preimmagine: per ogni x , è impossibile trovare $y \neq x$ tc $H(y) = H(x)$,
6. Resistenza alla 2^a preimmagine: è infattibile trovare qualsiasi coppia (x, y) tc $H(y) = H(x)$.

Applicazioni:

- la proprietà di unidirezionalità è importante nelle tecniche di autenticazione del messaggio che coinvolgono l'uso di un valore segreto. es:
 $A \rightarrow B : M || H(M || S)$ dove S è il segreto tra A e B
- la resistenza alla preimmagine previene la manomissione quando del codice hash cifrato è usato (non può essere ricifrato da terzi). es:
 $A \rightarrow B : M || E(K, H(M))$ $A \rightarrow B : M || E(PR_a, H(M))$
- la resistenza alla preimmagine è utile anche per proteggere i file delle password.
- la resistenza alla 2^a preimmagine è utile contro l'attacco del compleanno.

DEF S/Key: sistema otp per stupidi terminali e computer pubblici inaffidabili che non richiedono una password a lungo termine, perché ogni password è usata solo una volta (è inutile cercare di carpirle, "sniffarle". Le password sono mostrate su un computer o dispositivo mobile (sicuro e fidato).



DEF Paradosso del compleanno: se H ha 2^m possibili output. H dev'essere applicata a $2^{m/2}$ input per avere una probabilità di collisione maggiore di 0.5.

Dunque su un codice di hash con 64 bit. Per la resistenza alla preimmagine sono necessari 2^{63} tentativi, mentre per la resistenza alla 2^a preimmagine ne "bastano" 2^{32} per avere 0.5 di probabilità di avere una collisione.

Costruire una funzione di hash crittografica secondo lo schema generale di Merkle.

Divido il messaggio in blocchi di lunghezza fissa $M =$

Y_0, \dots, Y_{L-1} e applico:

$CV_0 = IV$ (=vettore vuoto)

$CV_i = f(CV_{i-1}, Y_{i-1})$ for $i = 1, \dots, L$

$H(M) = CV_L$

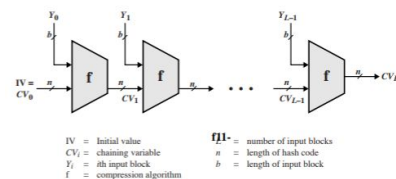


Figure 11.9 General Structure of Secure Hash Code

Esempi: MD5(usa 128 bits, ma è insicuro), SHA(usa 160 bits, assunto come sicuro).

Digital Signature

L'autenticazione del messaggio protegge la 2 parti da una terza, ma non le protegge tra di loro.

Diverse forme di dispute tra le due sono possibili, per esempio: B potrebbe falsificare un messaggio e dire che gli è stato mandato da A, viceversa, A potrebbe negare l'invio di un messaggio, perchè anch'esso potrebbe essere falsificato da B.

Requisiti di alto livello per il meccanismo di firma digitale:

- fornire i mezzi per verificare l'autore e la data della firma
- autenticare il contenuto al momento della firma
- essere verificabile da terze parti, per risolvere dispute

Requisiti per il meccanismo di firma digitale:

- deve essere uno schema di bit che dipende dal messaggio da firmare
- deve usare informazioni uniche per il mittente, per evitare falsificazione e ripudio.
- dev'essere facile produrre la firma digitale
- dev'essere facile riconoscere e verificare la firma digitale
- dev'essere infattibile falsificare la firma
- dev'essere pratico conservare una copia della firma digitale in arrivo

Due tipi di firma digitale: firma digitale diretta e firma digitale arbitraria.

Firma digitale diretta:

- il destinatario deve conoscere la chiave pubblica del mittente
- la firma è formata:
 - o dalla cifratura dell'intero messaggio con la chiave privata del mittente,
 - o dalla cifratura del codice hash del messaggio con la PR del mittente
- la sicurezza del sistema dipende dalla sicurezza della PR del mittente
- ogni messaggio dovrebbe contenere un timestamp e la compromissione della chiave dovrebbe essere prontamente riportata all'autorità centrale.
- ma, un attaccante potrebbe rubare la chiave al tempo T e falsificare il messaggio assegnandogli un tempo precedente

Firma digitale arbitraria:

il problema della firma digitale diretta può essere affrontato con l'identificazione di un arbitro, il quale può vedere il messaggio nel caso della codifica convenzionale.

Nota: nella diretta il mittente si firma da sé, nell'arbitraria, l'arbitro garantisce la firma per il mittente.

Pretty Good Privacy (PGP) è una famiglia di software di crittografia per autenticazione e privacy è probabilmente il crittosistema più adottato al mondo probabilmente il più vicino alla crittografia di livello militare.

Table 13.1 Arbitrated Digital Signature Techniques

(1) $X \rightarrow A: M \parallel E(K_{xa}, [ID_X \parallel H(M)])$ (2) $A \rightarrow Y: E(K_{ay}, [ID_X \parallel M \parallel E(K_{xa}, [ID_X \parallel H(M)]) \parallel T])$
--

(a) Conventional Encryption, Arbitrator Sees Message

(1) $X \rightarrow A: ID_X \parallel E(K_{xy}, M) \parallel E(K_{xa}, [ID_X \parallel H(E(K_{xy}, M))])$ (2) $A \rightarrow Y: E(K_{ay}, [ID_X \parallel E(K_{xy}, M)]) \parallel E(K_{xa}, [ID_X \parallel H(E(K_{xy}, M)) \parallel T])$

(b) Conventional Encryption, Arbitrator Does Not See Message

(1) $X \rightarrow A: ID_X \parallel E(PR_x, [ID_X \parallel E(PU_y, E(PR_x, M))])$ (2) $A \rightarrow Y: E(PR_a, [ID_X \parallel E(PU_y, E(PR_x, M)) \parallel T])$

(c) Public-Key Encryption, Arbitrator Does Not See Message


(8_Digital Certificates and PKI)

I certificati digitali sono oggetti che forniscono: distribuzione delle chiavi e autenticazione (con non ripudiazione); essi giocano un ruolo chiave nella sicurezza web.

L'infrastruttura delle chiavi pubbliche (PKI) fornisce gli elementi tecnici e legali per permettere un utilizzo sicuro del web.

Un certificato digitale permette di verificare l'affidabilità di una chiave pubblica, associando alla chiave pubblica il nome del proprietario, normalmente l'ente emittente del certificato è un'autorità certificativa.

Siano PR_E e PU_E dell'emittente. Scriviamo (Nome, PU_N , Emittente)[PR_E] sta a \rightarrow
Usando la PU_E possiamo validare o rifiutare l'affidabilità della chiave di 'Nome'.

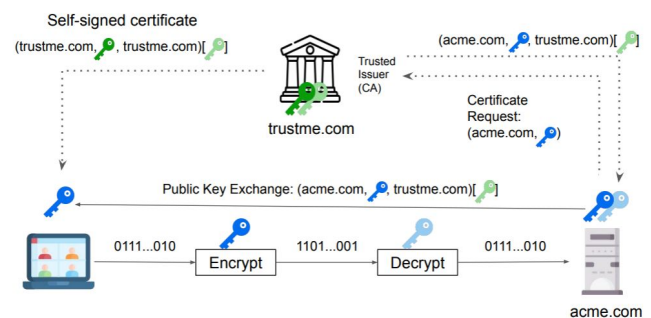
Owner	acme.com
Public Key	00...01001001 
Issuer	trustme.com
....
Signature	10...011101101

Prima di fidarsi di un certificato digitale occorre verificare la sua validità, ciò può essere fatto verificando la validità della firma digitale generata dall'emittente e confrontandola con quella presente nel certificato. Ovviamente, un certificato emesso da un emittente non fidato deve essere scartato.

Ciclo di vita della certificazione:

- Un sito richiede di essere certificato (acme.con, PU_a);
- l'ente certificatore lo certifica (acme.con, PU_a , trustme.com)[PR_T];
- il sito certificato invia la propria PU_A certificata all'utente (acme.con, PU_a , trustme.com)[PR_T];
- l'ente fornisce al client a propria PU_T certificata (trustme.com, PU_T , trustme.com)[PU_T];
- a questo punto il client può inviare al sito ciò che deve, cifrando con la sua chiave pubblica PU_A , ormai certificata ed affidabile.

Certificate Life-cycle



Oss: Questo protocollo resiste agli attacchi man-in-the-middle.

Differenti tipi di certificati diversi sono rilasciati da autorità certificative a seconda dell'oggetto del certificato:

- Validazione di dominio (DV): sono i certificati più comuni, l'ente certificatore deve solo verificare che il richiedente sia effettivamente in possesso del dominio, ma non è richiesto che sappia anche l'identità del richiedente nel mondo reale.
- Validazione dell'organizzazione (OV) o validazione estesa (EV): certificato in cui viene anche verificata l'identità reale del richiedente (oltre che quella informatica).

(9_security-protocols.pdf)

I protocolli di sicurezza possono essere usati per rendere sicuri:

- comunicazioni bancarie;
- comunicazioni in una rete di sensori;
- accoppiamenti di dispositivi wireless;
- un sistema di controllo degli accessi in un'area vasta per gli ski lifts;
- per i pagamenti online....

Gli algoritmi distribuiti per promuovere la sicurezza coinvolgono i protocolli di sicurezza come: IPSec, SSH, PGP, SSL, Kerberos.

DEF Protocollo: un protocollo consiste in un insieme di regole (convenzioni) che determinano lo scambio di messaggi tra 2 o più attori. In poche parole, sono un algoritmo distribuito con particolare enfasi sulla comunicazione.

DEF Protocolli di sicurezza (o crittografici): usano meccanismi crittografici per raggiungere gli obiettivi di sicurezza (es. integrità, autenticazione, non ripudiazione,...). Sono piccole ricette, ma non banali da progettare e comprendere.

I messaggi sono costituiti da:

- Nome: A, B...
- Keys: K e l'inversa K^{-1}
- Chiave simmetrica: $\{M\}_{K_{AB}}$, dove K_{AB} è condivisa tra A e B
- Cifratura: $\{M\}_K$, cifratura con la chiave pubblica di A: $\{M\}_{K_A}$
- Firma: $\{M\}_{K^{-1}_A}$, firmare con la chiave privata di A: $\{M\}_{K^{-1}_A}$
- Nonces: N_A , oggetto usato per sfidare/rispondere (generalmente un numero casuale)
- Timestamp: T, denota il tempo.
- Concatenazione dei messaggi: $\{M_1, M_2\}$, $M_1||M_2$, o $[M_1, M_2]$.

Un esempio è: $\{A, T_A, K_{AB}\}_{K_B}$

Prendendo come esempio il meccanismo di sblocco di un'auto, analizziamo come rendere lo scambio di messaggi tra il telecomando e la vettura sicuro.

Come primo obiettivo vogliamo che la macchina venga sbloccata dal ricevitore solo se il proprietario ha premuto il bottone di sblocco sul telecomando. Un attaccante potrebbe simulare il click e aprire la macchina.

1. Infatti se ponessimo come chiave condivisa tra telecomando (KF) e ricevitore (R) della vettura un numero seriale (SN) l'attaccante potrebbe semplicemente origliare il valore e riprodurlo successivamente; i problemi sono 2: la segretezza di SN è compromessa e R non può controllare l'autenticità della richiesta.
2. Potremmo allora far sì che KF cifri SN con una chiave condivisa K e invii il risultato a R. A questo punto la segretezza di SN è garantita, ma un attaccante potrebbe comunque origliare la richiesta cifrata ed mimarla in un secondo momento, infatti, nonostante R possa verificare l'autenticità, questo non basta a fermare un attacco malintenzionato.

Rivisitiamo il nostro obiettivo: vogliamo che R apra la macchina solo se KF ha recentemente inoltrato la richiesta.

3. KF cifra con K un T e invia il risultato a R. Non abbiamo più bisogno di SN e T previene gli attacchi mima, ma richiede che i clock di KF e R siano sincronizzati.
4. KF chiede a R un nonces (N), R invia a KF un N e KF risponde con N cifrato con K. L'utilizzo di N previene gli attacchi mima e non è necessario sincronizzare gli orologi di KF e R, ma sono necessari dei passaggi in più.

Gli eventi fondamentali sono la comunicazione tra le parti, le quali ricoprono ruoli (iniziatore e risponditore). La comunicazione è asincrona. Il protocollo specifica le azioni degli attori; equivalentemente, un protocollo definisce un insieme di eventi in sequenza (traccia).

Assunzioni (per attori):

- sanno le proprie PR e le PU degli altri;
- possono generare/controllare nonces e timestamps, de/cifrare con le chiavi note;
- quelli onesti implementato correttamente il protocollo.

Obiettivi: cosa il protocollo dovrebbe portare a casa:

- autenticazione dei messaggi
- garantire la tempestività dei messaggi
- garantire la segretezza di certi oggetti.

Assunzioni (per attaccanti):

- conoscono il protocollo, ma non possono romperlo
- sono passivi, ma origliano tutte le comunicazioni
- sono attivi e intercettano e generano messaggi
- potrebbero anche essere attori che usano il protocollo.

Modello di attacco standard:

- L'attaccante è attivo. Cioè può:
 - intercettare e leggere tutti i messaggi;
 - decomporre i messaggi nelle loro parti;
 - costruire nuovi messaggi con differenti parti (es T diverso);
 - mandare messaggi in ogni momento.
- Anche chiamato Dolev-Yao, sono le assunzioni più forti sull'attaccante; i protocolli corretti funzionano nella maggior parte dei casi.

Tipi di attacco:

- mimo: riusa parti di messaggi precedenti;
- Man-in-the-middle: l'attaccante si mette in mezzo alla comunicazione tra A e B;
- riflesso: rimanda le informazioni trasmesse indietro al mittente;
- type flaw: sostituisce gli elementi in un campo del messaggio. Es. usa un nome come nonce.

DEF Protocollo NSPK: protocollo proposto negli anni '70 e usato per decenni ha l'obiettivo di fornire un'autenticazione (delle entità) mutuale (= reciproco).

Correttezza (informale):

- A invia N_A a B, cifrato con la chiave pubblica di B;
- B risponde ad A con il suo N_B (che se l'ha letto vuol dire che B era effettivamente B, solo lui ha la sua chiave privata con la quale può decifrare e leggere il messaggio, in questo caso il messaggio è proprio N_A) e un nuovo N_B , il tutto cifrato con PU_A .
- Infine A risponde a B con N_B (che può essere letto solo da A), cifrato con PU_B

```
1.  $A \rightarrow B : \{A, N_A\}_{K_B}$ 
2.  $B \rightarrow A : \{N_A, N_B\}_{K_A}$ 
3.  $A \rightarrow B : \{N_B\}_{K_B}$ 
```

Questo protocollo è però vulnerabile all'attacco Man-in-the-middle.

Per proteggerci usiamo dunque il protocollo NSPK con l'aggiustamento di Lowe.

Qui questa modifica comporta che nella risposta di B ci sia anche il suo nome, cifrato sempre con la sua chiave pubblica.

```
1.  $A \rightarrow B : \{A, N_A\}_{K_B}$ 
2.  $B \rightarrow A : \{N_A, N_B, B\}_{K_A}$ 
3.  $A \rightarrow B : \{N_B\}_{K_B}$ 
```

Attacchi Type flaws: Siccome un messaggio può essere visto come una sequenza di messaggi più piccoli, come una sequenza di bit questo tipo di attacco prevede di non modificare il contenuto del messaggio, ma di far sì che vengano mal interpretati i tipi dei campi, ad esempio che si scambi il nome per un nonce. Quindi il messaggio arriva correttamente, ma le parti lo interpretano diversamente.

DEF Protocollo Otway-Rees: Protocollo lato server che fornisce la distribuzione delle chiavi autenticata (con autenticazione e freschezza delle chiavi), ma senza l'autenticazione delle chiavi o la conferma delle chiavi.

M1. $A \rightarrow B : I, A, B, \{N_A, I, A, B\}_{K_{AS}}$
M2. $B \rightarrow S : I, A, B, \{N_A, I, A, B\}_{K_{AS}}, \{N_B, I, A, B\}_{K_{BS}}$
M3. $S \rightarrow B : I, \{N_A, K_{AB}\}_{K_{AS}}, \{N_B, K_{AB}\}_{K_{BS}}$
M4. $B \rightarrow A : I, \{N_A, K_{AB}\}_{K_{AS}}$

In cui le chiavi del server sono note e I è l'identificatore dell'esecuzione del protocollo.

Sono possibili 2 attacchi a questo protocollo:

- Riflessione/type-flaw: supponendo che $|I, A, B| = K_{AB}$, un attaccante può mimare i punti 1 e 4, saltando 2 e 3. A vedendo N_A accetta $\{I, A, B\}$ come chiave di sessione.

M1. $A \rightarrow M(B) : I, A, B, \{N_A, I, A, B\}_{K_{AS}}$
M4. $M(B) \rightarrow A : I, \{N_A, I, A, B\}_{K_{AS}}$

- L'attaccante può giocare il ruolo di S in 2 e 3 riflettendo le componenti cifrate a B. Quindi M può decifrare tutti i messaggi, autenticazione e riservatezza della chiave falliscono.

M1. $A \rightarrow B : I, A, B, \{N_A, I, A, B\}_{K_{AS}}$
M2. $B \rightarrow M(S) : I, A, B, \{N_A, I, A, B\}_{K_{AS}}, \{N_B, I, A, B\}_{K_{BS}}$
M3. $M(S) \rightarrow B : I, \{N_A, I, A, B\}_{K_{AS}}, \{N_B, I, A, B\}_{K_{BS}}$
M4. $B \rightarrow A : I, \{N_A, I, A, B\}_{K_{AS}}$

DEF Protocollo RPC Andrew Secure: Stabilisce le chiavi di sessione K'_{AB} e un nonce N'_B per sessioni future; ha l'obiettivo di scambiare una chiave condivisa, segreta, autenticata e fresca tra 2 parti che condividono una chiave simmetrica.

M1. $A \rightarrow B : A, \{N_A\}_{K_{AB}}$ M1. $A \rightarrow B : A, \{N_A\}_{K_{AB}}$
M2. $B \rightarrow A : \{N_A + 1, N_B\}_{K_{AB}}$ M2. $B \rightarrow A : \{N_A + 1, N_B\}_{K_{AB}}$
M3. $A \rightarrow B : \{N_B + 1\}_{K_{AB}}$ M3. $A \rightarrow M(B) : \{N_B + 1\}_{K_{AB}}$
M4. $B \rightarrow A : \{K'_{AB}, N'_B\}_{K_{AB}}$ M4. $M(B) \rightarrow A : \{N_A + 1, N_B\}_{K_{AB}}$

Attacco flaw-type: supponendo che le chiavi e i nonces siano rappresentati da sequenze di bit della stessa lunghezza, un attaccante può registrare 2, intercettare 3, e mimare 2 come 4. A è dunque ingannato e accetterà il valore $N_A + 1$ come chiave di sessione. Anche se questa chiave non è autenticata, questo attacco non viola la proprietà di riservatezza.

Scambio di chiavi con Autorità certificate (Denning & Sacco):

- S: server dell'autorità certificativa;

- K_{AB} : chiave di sessione segreta;
- K_B : PU_B , il compagno con il quale si vuole comunicare;
- K_A^{-1} : firma con PR_A ;
- C_A, C_B : Certificati di A e B (nome, PU, ...);
- T_A : Timestamp generato da A.

$$\begin{aligned} A \rightarrow S &: A, B \\ S \rightarrow A &: C_A, C_B \\ A \rightarrow B &: C_A, C_B, \{\{T_A, K_{AB}\}_{K_A^{-1}}\}_{K_B} \end{aligned}$$

Obiettivo:

- K_{AB} è nota solo ad A e B (chiave di sessione);
- B può essere sicuro che gli è stato inviato da A:
 - B può decifrarlo usando la chiave pubblica di A;
 - K_A è vincolato ad A dal certificato C_A .
- B sa che il messaggio era rivolto a lui perché è stata usata la sua PU_B .
- T può essere usato come limite per la chiave di sessione.

Attacco Man-in-the-middle: se all'ultimo passaggio A inoltra a C anziché a B, la chiave diventa né segreta né autenticata e C può ascoltare tutta la conversazione.

The attack

$$\begin{aligned} A \rightarrow S &: A, B \\ S \rightarrow A &: C_A, C_B \\ A \rightarrow B &: C_A, C_B, \{\{T_A, K_{AB}\}_{K_A^{-1}}\}_{K_B} \end{aligned}$$

$$\begin{aligned} \text{alice} \rightarrow \text{charlie} &: C_{\text{alice}}, C_{\text{charlie}}, \{\{T_{\text{alice}}, K_{\text{alice charlie}}\}_{K_{\text{alice}}^{-1}}\}_{K_{\text{charlie}}} \\ \text{charlie} \rightarrow \text{bob} &: C_{\text{alice}}, C_{\text{bob}}, \{\{T_{\text{alice}}, K_{\text{alice charlie}}\}_{K_{\text{alice}}^{-1}}\}_{K_{\text{bob}}} \end{aligned}$$

Per difenderci A dovrà modificare il suo ultimo messaggio così:

$$A \rightarrow B: C_A, C_B, \{\{A, B, T_A, K_{AB}\}_{K_A^{-1}}\}_{K_B}$$

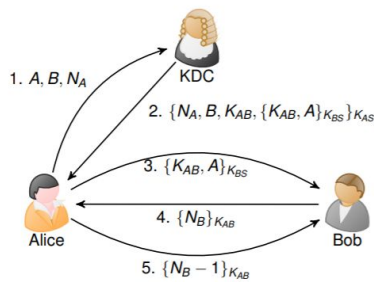
In questo modo B può sapere se la comunicazione ed il canale sono compromessi confrontando il certificato e i nomi cifrati.

DEF la prudente ingegneria dei protocolli di sicurezza sono i seguenti principi:

1. ogni messaggio dovrebbe dire ciò che intende;
2. le condizioni per l'invio di un messaggio dovrebbero essere esplicite;
3. menzionare esplicitamente il nome dell'attore se è essenziale per il significato;
4. essere chiari sul perché la cifratura deve essere fatta.
Riservatezza, autenticazione, associazione dei messaggi (es. $\{x, y\}_{K^{-1}}$ vs $\{x\}_{K^{-1}}, \{y\}_{K^{-1}}$);
5. essere chiari sulle proprietà che assumi tramite i nonces;
6. le quantità prevedibili per rispondere alle richieste dovrebbero essere protette dall'attacco mima;
7. i timestamp devono tener conto dell'ora locale e della manutenzione degli orologi;
8. una chiave potrebbe essere stata usata di recente, ma essere vecchia;
9. se una codifica è usata per presentare il significato di un messaggio, allora dovrebbe essere possibile dire quale codifica è stata usata.
10. Il designer del protocollo dovrebbe sapere su quali relazioni di fiducia si basa il suo protocollo.

I protocolli di sicurezza posso fornire proprietà che la crittografia classica da sola non offre.

La validità dei protocolli di sicurezza è difficile da testare in maniera tradizionale, per questo si usano tecniche automatizzate, più efficienti ed efficaci.



DEF protocollo NSSK: protocollo di sicurezza con lo scopo di effettuare scambio di chiavi autenticato.

In questo schema può far breccia un attacco mima da C a B, ricopiando $\{K_{AB}, A\}_{K_{BS}}$ il quale può essere escluso dal protocollo implementando l'utilizzo di un timestamp.

DEF Kerberos: Protocollo per il controllo dell'autenticazione/accesso per applicazioni client/server.

Esso ha "tre teste" a guardia della porta di rete: autenticazione, contabilità e verifica (le ultime 2 mai implementate); ampiamente usato (es. Microsoft Windows).

Obiettivi:

- le password non devono mai viaggiare in rete;
- le password non devono mai essere salvate in nessuna forma nel client;
- le password non dovrebbero mai essere salvate in chiaro, anche in un database autenticato;
- all'utente è chiesta la password una volta per sessione di lavoro;
- l'informazione di autenticazione risiede solo nel server autenticato, l'applicazione server non deve contenere le informazioni di autenticazione dei loro utenti ed è essenziale per:
 - l'amministratore può rimuovere gli utenti più facilmente;
 - al cambio di password di un utente, viene cambiata per tutti i suoi servizi;
 - non c'è ridondanza delle informazioni;
- Autenticazione mutuale: non solo gli utenti devono dimostrare chi sono, anche l'applicazione server deve provare la propria autenticità al client.
- dopo autenticazione e autorizzazione, client e server possono stabilire una connessione sicura (cifrata).

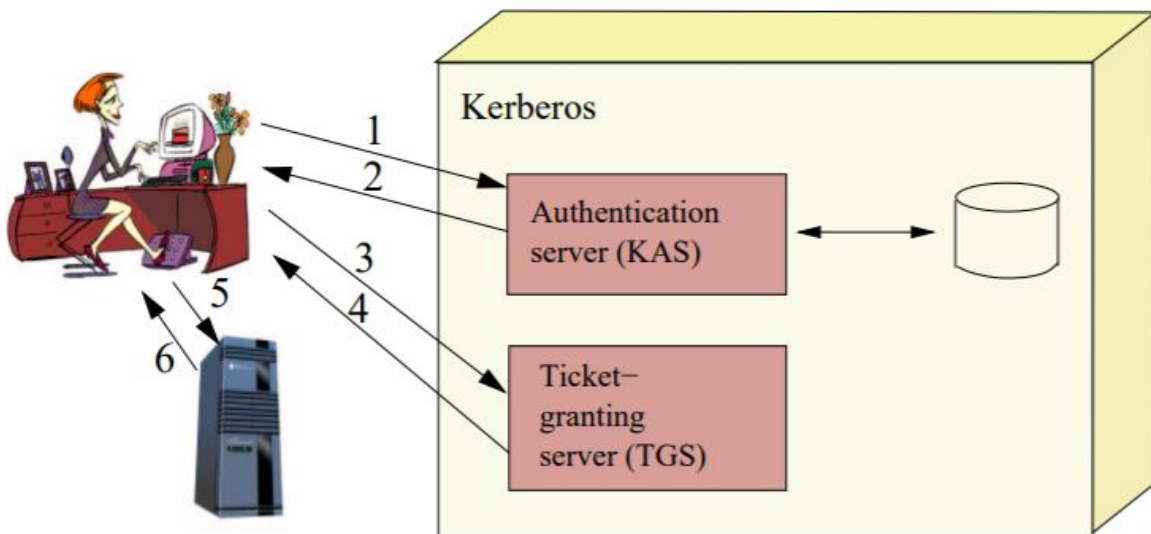
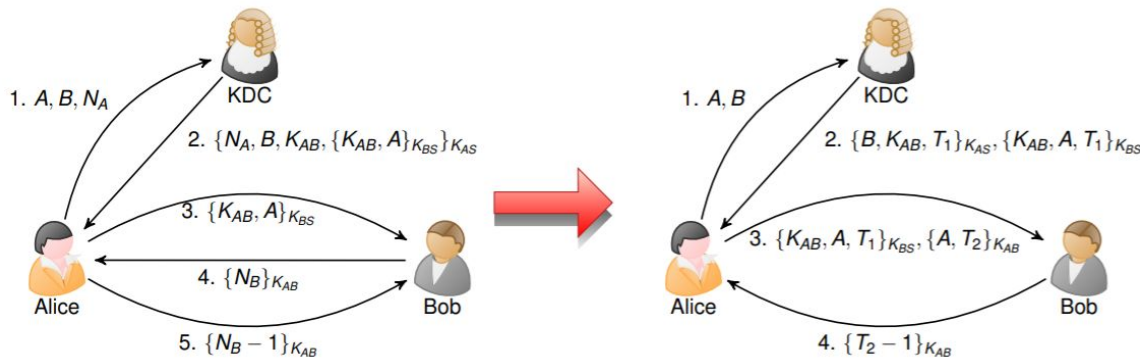
Requisiti del Kerberos:

- sicuro: un intercettatore, non dovrebbe poter ottenere info per impersonare un utente
- affidabile: diversi servizi dipendono dal Kerberos per la gestione degli accessi, esso deve essere affidabile e supportare architetture distribuite, dove un sistema può eseguire il backup di un altro;
- trasparente: ogni utente dovrebbe immettere una password singola per ottenere i servizi della rete, altrimenti dovrebbe essere all'oscuro degli altri protocolli;
- scalabile: può cresce con sistemi grandi, modulari e architetture distribuite con facilità

Kerberos v4:

- autenticazione: usa KAS, Server di Autenticazione di Kerberos;
- autorizzazione: usa TGS, Server di Autorizzazione di Ticket;
- controllo degli accessi: dove il server controlla i ticket del TGS.

Il protocollo di autenticazione di Kerberos li basa sul NSSK, ma usa i Timestamp anziché i nonces per assicurare la freschezza delle chiavi di sessione e rimuove la cifratura annidata.



Autenticazione nei passaggi 1 e 2 (una volta per sessione di login).

Autorizzazione nei passaggi 3 e 4 (una volta per ogni tipo di servizio).

Servizio (controllo dell'accesso) nei passaggi 5 e 6 (una volta per sessione di servizio).

Di seguito estele le 3 parti (semplificate):

- fase di autenticazione:

1. $A \rightarrow KAS : A, TGS$

2. $KAS \rightarrow A : \{K_{A,TGS}, TGS, T_1\}_{K_{AS}}, \underbrace{\{A, TGS, K_{A,TGS}, T_1\}_{K_{KAS,TGS}}}_{AuthTicket}$

- A logga nella stazione di lavoro e richiede una risorsa di rete;
- KAS accede ad DB e invia ad A una chiave di sessione $K_{A,TGS}$ (che vive per alcune ore, a seconda dell'app) e un ticket cifrato *AuthTicket*.
- K_{AS} deriva dalla password dell'utente (es. $K_{AS} = h(\text{Password}_A || A)$)
- sia la chiave user sia la chiave server devono essere salvate sul DB.
- A digita la pwd sul client per decifrare i risultati. Il ticket e la password di sessione vengono salvate e la password dell'utente è dimenticata. A effettua il logout quando $K_{A,TGS}$ scade.

- fase di autorizzazione:

$$3. A \rightarrow TGS : \underbrace{\{A, TGS, K_{A,TGS}, T_1\}_{K_{KAS,TGS}}}_{AuthTicket}, \underbrace{\{A, T_2\}_{K_{A,TGS}}}_{authenticator}, B$$

$$4. TGS \rightarrow A : \underbrace{\{K_{AB}, B, T_3\}_{K_{A,TGS}}}_{AuthTicket}, \underbrace{\{A, B, K_{AB}, T_3\}_{K_{BS}}}_{ServTicket}$$

Prima del primo accesso di A alla risorsa di rete B:

- A mostra AuthTicket a TGS con un autenticatore, il quale vive per qualche secondo (esso è utile a prevenire attacchi mimi, il server li salva per prevenire immediati attacchi mimi).
- TGS emette ad A una nuova chiave di sessione K_{AB} (vive per minuti) e un nuovo ticket *ServTicket*. K_{BS} è condivisa tra TGS e la risorsa di rete.
- fase di servizio (controllo accessi):

$$5. A \rightarrow B : \underbrace{\{A, B, K_{AB}, T_3\}_{K_{BS}}}_{ServTicket}, \underbrace{\{A, T_4\}_{K_{AB}}}_{authenticator}$$

$$6. B \rightarrow A : \{T_4 + 1\}_{K_{AB}}$$

A per accedere alla risorsa di rete B:

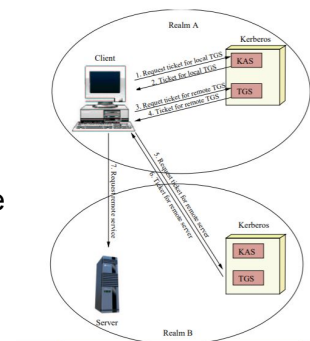
- A mostra K_{AB} a B con un altro autenticatore (con eventuali altre informazioni utili al server).
- B risponde, autenticando il servizio (permette l'uso della risorsa).

DEF reame: è definito da un server Kerberos che salva le password dell'utente e del server dell'applicazione per il reame.

Grandi reti possono essere divise in reami amministrativi.

Kerberos supporta protocollo tra reami facendo sì che i server si registrino reciprocamente e A può accedere a B in un altro reame, il TGS di A riceve la richiesta e il ticket per accedere al TGS del reame di B.

Estendere il Kerberos su più reami è semplice, ma per n reami il problema della distribuzione delle chiavi ha complessità $O(n^2)$.



Limiti:

- nel primo punto non è necessaria la cifratura, ma un attaccante potrebbe sommergere il server di richieste;
- nel secondo punto la cifratura annidata non è necessaria (infatti è eliminata nella versione 5 del Kerberos).
- Si basa su un clock sincronizzato e non compromesso, ma se l'host viene compromesso, allora anche l'orologio può essere facilmente compromesso.

Per concludere, i protocolli di sicurezza sono degli elementi chiave per la sicurezza dei sistemi distribuiti e sono onnipresenti (si estendono a praticamente tutti i domini delle applicazioni), infine sono (ingannevolmente) semplici: vanno usati i protocolli stabiliti quando e dove possibile, ma è necessario essere in grado di progettarne di nuovi se necessario.

(10_websec.pdf)

DEF Sicurezza Web: non è ben definita come la crittografia, nella pratica la sicurezza web e di rete dipende da: dettagli standard della rete, dettagli implementativi, versioni concrete dei browser e dei server,.....

Gli attacchi che tendono a minare la sicurezza web sono quelli contro la privacy, la sicurezza e la qualità del servizio.

Essendo web e rete oggetti "in movimento" e d in continua mutazione, non esiste una "soluzione per sempre", in maniera analoga per gli attacchi, c'è sempre una modifica e innovazione nella difesa e nell'attacco della sicurezza web.

DEF HyperText Transfer Protocol (HTTP): è un protocollo del livello applicativo che permette di trasferire richieste e informazioni tra server e browser.

Il client inizia le comunicazioni scegliendo il metodo (GET, HEAD, POST, PUT) e può navigare attraverso gli url, si ricorda inoltre che, HTTP non supporta le sessioni.

Il server invia i dati (di qualsiasi dimensione, statici come pagine HTML e immagini o dinamici come dati calcolati a run time) su richiesta dell'utente.

Lo script può essere sia lato client (js), sia lato server (php).

I dati sono mandati all'applicazione tramite un metodo HTTP, poi processati dal relativo script e, infine, i risultati sono ritornati al browser dell'utente.

Il GET espone le informazioni nell'URL, mentre il POST mette le informazioni nel corpo della richiesta e non nell'URL; per questo motivo si impone l'utilizzo di HTTPS POST per il trasporto di dati sensibili

Per ogni richiesta il client invia un header (che contiene informazioni come: lingua, OS, Browser usato, cookie, ...) al server che può non essere cifrato (HTTP) o esserlo (HTTPS).

L'header HTTP può contenere delle informazioni sensibili come: informazioni del mittente (come la mail, usabile per lo spam), informazioni di autenticazione (AUTHORIZATION, in HTTP sono sinonimi autorizzazione e autenticazione), i cookie e le pagine visitate dal client. Una combinazione di queste informazioni permette al sito un buon controllo del comportamento dei clienti.

I cookie sono stati introdotti per permettere acquisti online, funzionano così:

- il server può, in ogni risposta, includere un cookie.
- un client invia, in ogni richiesta, il cookie al server.
- il cookie può contenere qualsiasi cosa (fino a 4 kb)
- il cookie ha vita limitata e specificata.

Questo strumento è criticato per i possibili attacchi alla privacy degli utenti in quanto possono essere tracciati (es. lettura dei log dei server). i cookie andrebbero trattati come informazioni confidenziali e quelli con una scadenza troppo lontana sono sospetti.

HTTP supporta 2 modalità di autenticazione:

- basica:
 - basata su login/password;
 - l'informazione è inviata non cifrata;
 - le credenziali sono inviate per ogni richiesta per ogni reame.
 - estremamente diffuso e quasi totalmente supportato da client/server.

- Autenticazione Digest:
 - il server invia un nonce.
 - il client applica la funzione di hash sul nonce basandosi su login/password.
 - il client invia l'hash crittografati attraverso la rete.
 - è usato raramente.

Considerazioni generali: bisogna stare attenti quando si naviga nel web, i siti visitati sono salvati (nella cache, nella cronologia), bisogna usare gli strumenti di memorizzazione delle password con cautela, molte minacce sono causate da componenti attive malintenzionate (es in javascript).



Questi sono i 10 più pericolosi rischi alla sicurezza per le applicazioni web, essi:

- non attaccano direttamente né la crittografia né l'autorizzazione;
- sfruttano difetti di programmazione o configurazione;
- Sono tutti relativamente facili da sfruttare;
- danneggiano gravemente l'app: rivelano segreti o minano la qualità del servizio;
- possono essere sicuri solo i sistemi ben disegnati.

Input non valido:

Siccome le applicazioni web usano dati presi in input tramite HTTP anche gli attaccanti possono sfruttare questa interazione per danneggiare la web app.

I possibili attacchi che sfruttano l'inoltro di dati inaspettati sono: inserzione di comandi di sistema, SQL injection, cross-site scripting (XSS), Cross-Site Request Forgery (XSRF), Clickjacking.

Per difendere il sito si può fare affidamento ad una validazione dell'input lato client (JavaScript) e/o lato server (che può prevenire gli attacchi precedentemente descritti), inoltre l'applicazione di un firewall può permettere la validazione di alcuni parametri.

Falla di iniezione: l'attaccante prova a iniettare un comando al sistema di back end, tipo:

- il sistema operativo del server;
- il DB;
- linguaggi di script utilizzati;

L'attaccante prova a far eseguire del codice sul sistema del server.

Per prevenire si possono: filtrare gli input, evitare chiamate a interpreti esterni, scegliere chiamate sicure a sistemi esterni, controllare i valori di ritorno per riuscire a trovare gli attacchi e nello specifico, per i DB, preferire statement SQL preparati.

JavaScript: linguaggio di scripting più noto, funziona con tutti i browser, pensato per aggiungere interattività alle pagine HTML. Esso è dinamico, debolmente tipato, interpretato, viene eseguito sulla macchina virtuale nel browser.

Esso può accedere a vari elementi di una pagina come un form, un elemento del form e inoltre può gestire i cookie (leggendoli, impostando o eliminandoli).

JavaScript è stato creato per funzionare in una sandbox: un ambiente protetto in cui lo script non può accedere alle risorse del computer del browser.

DEF politica dell'origine comune: permette agli script in esecuzione in pagine che provengono dallo stesso sito di accedere ai reciproci metodi e proprietà senza specifiche restrizioni, impedendo invece l'accesso alla maggior parte dei metodi e delle proprietà tra pagine provenienti da siti diversi. Il termine "origine" è definito usando il nome di dominio, il protocollo di livello di applicazioni usato, e (nella maggior parte dei browser) la porta TCP del documento HTML di cui vengono eseguiti gli script. Due risorse sono considerate avere la stessa origine se e solo se tutti e tre i valori sono esattamente gli stessi.

Al centro di un attacco XSS tradizionale si trova uno script vulnerabile: il lo script legge parte della richiesta HTTP e la riporta alla pagina di risposta, senza prima igienizzando.

Ad esempio potrebbe essere mostrato il cookie di un utente a video, o addirittura, potrebbe essere inviato il cookie all'attaccante il quale ruberebbe le informazioni (es esempio, potrebbe aprire inviare il cookie ad un suo sito malevolo e salvarlo lì).

Contromisure: utilizzare l'header di risposta X-XSS-Protection per abilitare la protezione interna del browser e X-Content-Security-Policy per istruire il browser su quale politica sul contenuto dei siti è attiva (ad esempio sposta tutti gli script in linea in file esterni).

DEF La falsificazione delle richieste nel dominio (CSRF): attacco che avviene quando programma, sito, blog malintenzionato fa sì che il browser dell'utente esegua azioni non volute su siti fidati sui quali è correntemente autenticato.

Contromisure inefficaci: utilizzo di cookie segreti, accettare solo dati in post, transazioni a più step e riscrittura degli url.

Contromisura falsificazione delle richieste nel dominio (CSRF) pattern sincronizzatore di token: ha l'obiettivo di fornire ad un'applicazione un controllo su cosa l'utente intenda realmente inviare. Tutto ciò avviene generando dei token che vengono associati alla sessione dell'utente, questi vengono inseriti nei form HTML e nei link associati a operazioni sensibili, dunque, quando vengono invocate queste operazioni la richiesta HTTP include anche il token, sarà quindi responsabilità del server verificare l'esistenza e la correttezza del token.

DEF Clickjacking: Il clickjacking (un sottoinsieme della "correzione dell'interfaccia utente") è una tecnica dannosa che consiste in ingannare un utente web facendogli interagire con qualcosa di diverso da ciò che crede l'utente con cui sta interagendo.

L'header di risposta HTTP X-Frame-Option protegge dalla maggior parte delle classi di Clickjacking, inoltre si può usare la direttiva frame-ancestor nel X-Content-Security-Policy.

Rottura del controllo degli accessi e della gestione delle sessioni.

Dei meccanismi degli accessi affidabili sono difficili da implementare, configurare e mantenere. La politica del controllo degli accessi dovrebbe essere ben documentata.

La gestione dell'autenticazione e della sessione è inclusa nelle pagine web per:

- il cambiamento delle password;
- la gestione delle password dimenticate;
- l'aggiornamento dei dati personali.

La complessità di questi sistemi è spesso sottovalutata e un attaccante può dirottare una sessione di un utente e la sua identità.

Per evitare queste minacce una web app dovrebbe:

- richiedere l'immissione della password su ogni pagina di gestione;
- richiedere password forti;
- implementare un meccanismo di cambio password;
- salvare le password come hash;
- proteggere le credenziali e l'ID di sessione in transito nella rete;
- evitare il caching.

Siccome HTTP è stateless (non ha memoria) le web app creano e gestiscono le sessioni che vengono salvate sul server (ognuna con un unico ID) e il client sa, al momento della creazione, quando questa è instaurata (in ogni richiesta allega l'ID di sessione).

Ci sono 3 possibili modi per trasportare gli ID di sessione:

- codificarli nell'URL come parametro GET; presenta svantaggi:
 - visibili nella barra dell'url e nella cache;
- nasconderli nei campi del form: funziona solo con le richieste POST;
- cookies: metodo preferito, ma può essere rifiutato dal client.

Tipi di attacchi di sessione mirati a rubare l'ID di sessione:

- Intercettazione: intercettare richieste e risposte e estrarre l'ID (usare SSL per difendersi, attivarlo con l'header di risposta HTTP Strict-transport-security);
- Predizione: predire in pochi tentativi l'ID (usare un imprevedibile ID da un generatore di numeri casuali);
- Forza bruta: provare molte volte a trovare l'ID;
- Fissazione: far sì che la vittima usi un certo ID;
- i primi 3 possono essere raccolti nel gruppo di attacchi "dirottamento di sessione".

Gestione degli errori:

- i messaggi degli errori non devono mostrare dettagli sull'app;
- non bisogna distinguere tra "file non trovato" e "accesso negato";
- il sistema deve rispondere con messaggi di errore chiari e concisi all'utente;
- gli errori di esecuzione potrebbero essere un input prezioso per il sistema di rilevamento delle intrusioni.

Un uso insicuro della memoria potrebbe essere dovuto a: il salvataggio di dati sensibili senza in chiaro, salvataggio insicuro di chiavi e certificati, salvataggio improprio di segreti in

memoria, scelta di algoritmi crittografici devoli e poca randomicità, o ancora il tentativo di creare una “nuova” crittografia e l'impossibilità di cambiare le chiavi.

Si può prevenire un uso insicuro della memoria con: massimizzare la cifratura, minimizzare il numero di dati salvati, scegliere algoritmi crittografici affidabili, assicurarsi che certificati, password e chiavi siano salvati in sicurezza e dividere il segreto in parti e comporlo se necessario (segreto può essere una chiave condivisa).

DEF Denial of Service (Dos): attacco attraverso la rete, invia più richieste HTTP possibili per sovraccaricare il server. Ad oggi chiunque potrebbe effettuare un attacco DoS, è dunque necessario testare la propria app con alti carichi. Un bilanciamento del carico può aiutare (ad esempio limitando il numero di richieste da parte degli utenti/client/sessione).

Il corretto mantenimento del software è un problema complicato. Occorre che:

- non venga mai eseguito del codice con dei problemi;
- stare attenti a configurazioni errate del server;
- rimuovere tutti gli accounts e le passwords di default;
- controllare le configurazioni di default per evitare insidie (“di fabbrica”);
- rimuovere i file non necessari;
- controllare i permessi impropri per file e cartelle;
- controllare le configurazioni errate per i certificati SSL.

Conclusione:

Molti dei problemi sono causati dalla complessità della costruzione dei sistemi: per esempio unire sistemi più piccoli in uno grande o eseguire implementazioni incompatibili (anche se di poco) e il problemi di riuscire ad eseguire correttamente configurazioni complesse.

NB: i sistemi sono sicuri come il link più debole.

Oggi la crittografia è difficile da rompere, ma i sistemi concreti sono ancora vulnerabili e molti degli attacchi effettuati con successo sono avvenuti per errori nella configurazione.

Linee guida della sicurezza:

- design
 - principio KISS;
 - la sicurezza per occultazione non funziona;
 - usare i privilegi il meno possibile;
 - separare i privilegi (creare più classi di privilegi).
- Implementazione
 - valida input e output del sistema;
 - non fidarti della validazione lato client;
 - fai fallire il sistema in sicurezza;
 - usa e riusa componenti fidate (e affidabili);
 - testa il sistema (es usando tool di attacco);
- altre tecniche:
 - non dovresti fidarti solo sul firewall standard, devi filtrare con attenzione a livello applicativo!
 - i firewall a livello business possono aiutare, ma non sono una panacea;
 - applica il rilevamento delle intrusioni.
- I problemi alla sicurezza cambiano ogni giorno: sii aggiornato!
- controlla la tua configurazione regolarmente!

(11_buffer-overflow.pdf)

Nell'implementazione di un SW è possibile incappare in errori che causano un'insicurezza del prodotto, anche se si ha un buon design in quanto quest'ultimo lascia molte opzioni aperte e il concretizzare le idee porta nuove vulnerabilità.

La comprensione delle vulnerabilità è vitale per lo sviluppo e la valutazione contromisure. La mancata considerazione porta a un codice insicuro o eccesso di fiducia nelle soluzioni parziali. Lo sfruttamento delle vulnerabilità (= pirateria informatica) è un reato penale.

DEF Buffer: è una regione di memoria contigua che memorizza i dati dello stesso tipo.

Un buffer overflow si verifica quando i dati vengono scritti oltre la fine del buffer.

Il danno risultante dipende da:

- Dove si riversano i dati;
- Come viene utilizzata questa regione di memoria (ad es. Flag per il controllo dell'accesso);
-
- Quali modifiche vengono apportate

Un esempio di programma vulnerabile potrebbe esserlo uno scritto in C che ha un array di caratteri di dimensione fissata (es. `char buff[8]`), ma che non controlla il numero di caratteri immessi nel buffer da parte dell'utente.

Questo problema potrebbe essere esteso a casi di manomissione di programmi, in quanto, senza un controllo del limite di scrittura, un utente può andare a scrivere fuori dall'area di memoria preposta e sovrascrivere altre variabili (ad esempio forzare un risultato se questo non viene correttamente gestito dal programma o, con una stringa molto lunga, addirittura andare a sovrascrivere tutte le variabili del programma compreso l'indirizzo di ritorno).

Come sfruttare il codice per effettuare attacchi malevoli:

- l'attaccante potrebbe volere che il programma salti in un codice che genera shell;
- il codice malevolo verrà messo nel buffer stesso (che è in overflow);
- l'indirizzo di ritorno dev'essere il punto di ingresso del codice in cui si vuole saltare

non è banale da attuare nella pratica.

Il codice malevolo può essere anche piazzato:

- nello stack: in parametri o altre variabili locali;
- nello heap: in qualche regione di memoria allocata dinamicamente;
- nelle variabili di ambiente (nello stack).

Un'altra variabile di attacco è abusare del codice esistente (saltare in librerie o programmi).

Difesa:

- usare un canarino: valore dello stack che deve essere verificato prima di ritornare la funzione (può essere casuale o determinato).
- evitare funzioni non sicure delle librerie (es. `strcpy`, `gets`; usare invece `strncpy`, `fgets`).
- Controllare sempre il limite degli array quando si esegue l'iterazione su di essi (anche con strumenti automatici, anche se questi non sono sempre affidabili).
- Programmare con attenzione, controllare e usare tool (es. `grep`)
- evitare la dichiarazione di array locali, ma bensì dichiararli dinamicamente, così l'indirizzo di ritorno della funzione non può essere sovrascritto.
- segnare come non eseguibile il pezzo di memoria che contiene il buffer così non potranno essere eseguiti programmi malevoli lì.

Conclusione:

- gli array possono essere sovrascritti in C;
- ciò causa che i dati e il flusso di controllo possono essere alterati fuori controllo;
- occorre programmare in maniera difensiva.
- soppesare bene pro e contro del PL usato.

(12_ipsec.pdf)

Rete

- fisicamente: una collezione di punti che trasmettono flussi di bit.
- logicamente: un mezzo di comunicazione tra 2 parti.
- un canale sicuro e un'altra astrazione ancora, altre astrazioni possono riguardare disponibilità, privacy del compagno di comunicazione, etc.

La funzionalità logica è costruita in modo stratificato, con il protocollo TCP/IP a 4 livelli (link, network, trasporto, applicazione) come modello di riferimento (modello alternativo OSI: con 3 livelli in più: fisico, presentazione e sessione).

DEF Internet: confederazione di reti che usano il protocollo TCP/IP. Non esiste un dominio globalmente affidabile, differenti sottoreti possono o meno essere affidabili e siccome per inviare un pacchetto, questo deve fare almeno 15 salti in altri nodi, come possiamo rendere sicura la comunicazione/applicazione? Possiamo utilizzare applicazioni sicure su canali insicuri (kerberos, PGP), ma anche rendere sicuri gli altri livelli è possibile.

TCP e IP da soli non provvedono a rendere una comunicazione sicura, ma questi due protocolli sì:

- SSL (o TLS/SSH): non è implementato a livello OS, ma a livello applicativo. L'API SSH è un insieme di porte API per TCP.
- IPsec: è implementato a livello OS. Non avvengono modifiche a livello applicativo o per il TCP.

Applicazione (o end-to-end):

- non sono necessarie assunzioni circa la sicurezza dei protocolli usati, i percorsi,...👍
- le decisioni della sicurezza possono essere basate su ID utente, data,...👍
- le applicazioni devono essere pensate per essere "consapevolmente sicure".👎

Tra il livello applicazione e trasporto: es. SSL

- non vengono eseguite modifiche al OS. modifiche minime nell'applicazione. 👍
- problemi di interazione con TCP. SSL potrebbe rifiutare quello che TCP accetterebbe e di conseguenza potrebbe cadere la connessione => attacco DoS facile 👎

IPsec:

- sicurezza a livello di trasporto senza modifiche al livello applicativo. 👍
- autenticazione dei soli IP, non c'è l'autenticazione dell'utente 👎
- Si possono implementare altri controlli, ma richiede modifiche di API e dell'app.

DEF IPsec: fornisce un canale sicuro per tutte le applicazioni, cifra e/o autentica il traffico.

Permette il filtraggio basandosi sulle politiche del database, come se avessimo un firewall tra i due estremi. è installato:

- nel sistema operativo: per la sicurezza end-to-end;
- nel gateway di sicurezza: quindi nei firewall o nei routers.

Il secondo viene usato per implementare le VPNs.

L'IPsec può essere usato per rendere più sicuro un e-commerce, rendere sicuro l'accesso attraverso la rete ad un dispositivo remoto.

IPsec è uno standard IETF. Specifiche complesse che coprono protocolli per diversi scopi:

- Authentication Header (AH): protegge l'integrità e l'autenticità di un datagramma (ma non la sua riservatezza);
- Encapsulating Security Payload (ESP): protegge riservatezza e a volte l'integrità;
- Key Management (IKE): Protocollo di scambio di chiavi in internet.

Table 16.1 IPSec Services

	AH	ESP (encryption only)	ESP (encryption plus authentication)
Access control	✓	✓	✓
Connectionless integrity	✓		✓
Data origin authentication	✓		✓
Rejection of replayed packets	✓	✓	✓
Confidentiality		✓	✓
Limited traffic flow confidentiality		✓	✓

DEF security association (SA): è una relazione unidirezionale tra mittente e destinatario per definire i servizi di sicurezza, specifica cose tipo: l'algoritmo di autenticazione (AH), l'algoritmo di cifratura (ESP), le chiavi, la durata delle chiavi, la durata della security association, il modalità di protocollo (tunnel o trasporto),...

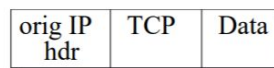
Viene incluso l'indice dei parametri di sicurezza e l'indirizzo di destinazione nei campi AH/ESP. SA è stabilito usando IKE, o qualche altro protocollo.

AH

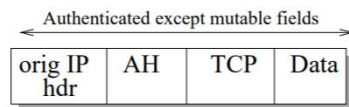
Un header extra tra il 3° e il 4° livello (IP e TCP), permette di riservare sufficiente informazioni in merito al SA usato. AH garantisce solo l'integrità, ma protegge anche e parti dell'header IP.

Modalità di AH

Datagramma originale (per TCP):

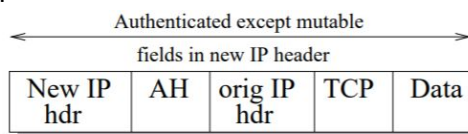


Modalità trasporto:



- AH è inserito dopo l'header IP, ma prima del payload dell'IP;
- un MAC (num a dim fissa) è preso per tutto il pacchetto (tranne per i campi mutabili);
- fornisce protezione end-to-end tra sistemi che hanno attivo IPsec.

Modalità tunnel:



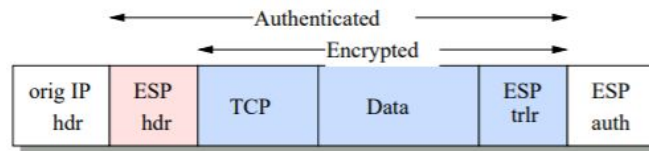
- tutto il pacchetto originale è autenticato ed è aggiunto un nuovo IP header esterno;
- l'header interno ha l'informazione dell'indirizzo mittente/destinatario originale;
- il nuovo header IP è anch'esso protetto (ad eccezione dei campi mutabili) e può contenere differenti indirizzi IP, es. firewalls o gateways.

AH è usato per fornire un canale autenticato o end-to-end (tipicamente con la modalità trasporto) o, con la modalità tunnel, per un gateway sicuro.

ESP

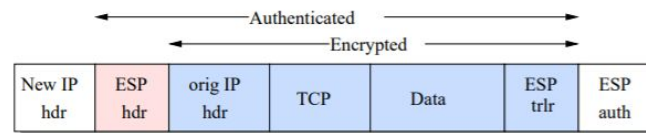
L'header specifica crittografia e opzionalmente autenticazione

Modalità trasporto: cifra solo i dati del payload di ogni pacchetto, ma non tocca gli header



fornisce una crittografia end-to-end tra host, che supportano IPSec

Modalità tunnel: tutto il datagramma è incapsulato con ESP. Quindi sia intestazione che payload sono cifrati (e facoltativamente autenticati).



può essere usato per impostare una VPN

Gli SA possono essere combinati e macchine diverse possono implementarne diversi contemporaneamente per usare differenti metodi seconda dell'interlocutore.

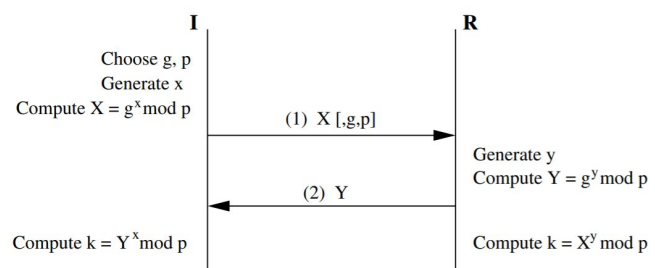
L'IKE stabilisce non solo le chiavi, ma anche l'SA.

L'IKE è molto flessibile, es. supporta l'autenticazione basata su una varietà di pre-condivisi segreti (ad esempio chiavi master), ma è anche molto complesso (molte opzioni, alternativi sotto-protocolli, ...). Tra i protocolli che hanno dato origine ad IKE ci sono:

- ISAKMP (Internet Security Association and Key Management Protocol): fornisce un framework e un protocollo di negoziazione generico per stabilire SA e chiavi crittografiche, ma non prescrive un meccanismo di autenticazione particolare.
- Oakley: una suite di protocolli di concordanza di chiave in cui due parti generano una chiave congiuntamente.

In parole povere, IKE combina i formati di pacchetto di ISAKMP e scambi di OAKLEY, basati su Diffie-Hellman.

Scambio di chiavi Diffie-Hellman di base: iniziatore I e risponditore R scambiano le "mezze chiavi" per arrivare alla mutua chiave di sessione k.



Se questo protocollo subisse un attacco DoS, l'attaccante invierebbe richieste diverse falsificando l'IP e R sarebbe costretto a sprecare esponenzialmente lo spazio in memoria. Difese deboli: farsi che il mittente esegua gli stessi calcoli, richiedere risposte a un indirizzo. Difere: cookie (numeri casuali scambiati tra I e R, o calcolati con una funzione di hash), I deve rispondere con il cookie, altrimenti R non computa Y, quindi l'attaccante deve essere all'indirizzo e rispondere con il cookie.

Lo scambio di chiavi non autenticate permette un attacco man-in-the-middle, dal quale ci si può difendere aggiungendo due step di autenticazione: I invia ID_I e R risponde ID_R dopo lo

scambio di Y (entrambi gli ID sono firmati tramite la chiave del mittente e possono essere crittografati).

Se si condividono già le chiavi?

Si usa il Perfect Forward Secrecy (PFS): se qualcuno scopre le chiavi private di I e R non sarà in grado di decifrare nulla.

Senza PFS: SSL in cui I sceglie un segreto, lo cifra PU_R e il resto della sessione è protetto sul segreto.

Inoltre, generazione periodica di nuove chiavi e materiale per la codifica complica la crittoanalisi.

(13_access.pdf)

Per garantire confidenzialità, integrità e disponibilità abbiamo visto approcci crittografici ben pensati (traffico di rete), possibili (proteggere file in un computer) e mal pensati (controllando processi o operazioni su dati e altri processi).

Una politica di sicurezza definisce cosa è concesso. Definisce quali esecuzioni sono accettabili, o complementariamente, non accettabili, è analoga ad una legge e solitamente definita in termini di regole di alto livello o requisiti. Descrivono la restrizione degli accessi, es. una relazione tra soggetto e oggetto.

DEF Modello di sicurezza: fornisce una rappresentazione formale di una (o più) politica di sicurezza.

DEF Meccanismo di sicurezza: definisce le funzioni di basso livello sw/hw che implementano i controlli imposti dalle politi e formalmente descritti nel modello.

Le politi del controllo degli accessi possono essere raggruppate in 3 classi principali:

- Discrezionale (DAC) (basate su autorizzazione): si basano sull'identità del richiedente e le regole descrivono cosa egli può (o non può) fare.
- Obbligatoria (MAC): si basano su normative obbligatorie decise dall'autorità centrale.
- Basate sui Ruoli (RBAC): si basano sul ruolo che l'utente ricopre nel sistema e descrivono cosa può fare un utente a seconda del suo ruolo.

DAC e RBAC sono spesso associate a delle politiche amministrative che definiscono chi può delineare autorizzazioni/ruoli.

Come si compone il meccanismo di controllo degli accessi?

Il sistema sa chi è l'utente (è autenticato). Ogni richiesta passa da un componente fidato denominato reference monitor che deve godere di queste proprietà:

- resistente alle manomissioni: non deve essere possibile modificarlo (o almeno non può essere modificato senza che nessuno se ne accorga);
- non bypassabile: deve mediare tutti gli accessi al sistema e alle sue risorse;
- security kernel: deve essere contenuto in una parte limitata del sistema;
- piccolo: deve essere di dimensione limitata per poter subire verifiche rigorose.

DEF stato di un sistema: è l'insieme di valori di tutta la memoria (registri, celle,...). Il sottoinsieme della memoria che si occupa della protezione è il sistema di protezione degli stati (es. in un file system è la parte che determina chi può eseguire azioni su quali oggetti).

Sia P lo spazio del sistema e Q contenuto in P sia lo stato in cui il sistema è autorizzato a risiedere; finché il sistema s è in Q esso è sicuro, quando il sistema è in P, ma non in Q, non è sicuro.

Q è definito da una politica di sicurezza, quindi le politiche dividono gli stati del sistema in autorizzati o sicuri e non autorizzato o insicuri.

Un meccanismo di sicurezza previene che un sistema passi in uno stato non sicuro.

Un sistema sicuro si ha quando esso parte in uno stato sicuro e non può andare in uno stato non sicuro.

ES. kerberos, politiche di sicurezza per dati proprietari.

Controllo dell'accesso discrezionale (DAC)

Premessa: le risorse degli utenti sono di loro proprietà e ne possono controllare l'accesso.

Il proprietario della risorsa può cambiarne i permessi a sua discrezione (anche passare al proprietario). è un meccanismo flessibile, ma aperto a negligenza, errori e abusi.

Necessità che tutti gli utenti conoscano e rispettino le politiche di sicurezza e i meccanismi di controllo degli accessi. Es di abuso. trojan: un programma può ingannare un utente per trasferirgli i permessi.

Accesso di controllo - matrice degli accessi

Semplice schema per descrivere la protezione di un sistema, descrivendo i privilegi dei soggetti sugli oggetti.

- soggetti: utenti, processi, agenti, gruppi,...
- oggetti: dati, blocchi di memoria, altri processi,...
- privilegi (o permessi/diritti): lettura, scrittura, modifica,...

Usata per rappresentare la relazione finita: $AC \subseteq \text{Subjects} \times \text{Objects} \times \text{Privileges}$ in tabella.

		Objects					
		File 1	File 2	File 3	File 4	Account 1	Account 2
Subjects	Alice	Own R W		W X		Inquiry Credit
	Bob	R	Own R W	W	R	Inquiry Debit	Inquiry Credit
	Charlie	R W	R		Own R X		Inquiry Debit

Uno stato di protezione (relativo ad un insieme di privilegi) è una tripla (S, O, M)

- S: insieme di soggetti,
- O: insieme di oggetti,
- M: matrice definisce i privilegi per ogni (s, o) appartenente a $S \times O$ in cui $M(s, o) = \{p \text{ appartiene all'insieme dei privilegi} \mid (s, o, p) \text{ appartiene a } AC\}$

Una transizione di stato descrive un comando tipo:

- aggiungi un privilegio p in $M(s, o)$;
- crea il soggetto s;
- distruggi l'oggetto o.

Questi trasformano uno stato in un altro cambiando le sue parti.

Accesso di controllo - modello Harrison-Ruzzo-Ullman

Definisce le autorizzazione del sistema formalizzando come cambiano i diritti o come creare e eliminare soggetti e oggetti.

Stato (S, O, M) , per soggetto S , oggetto O , matrice M .

Transizione descritta dai comandi nella forma:

```

command  $c(x_1, \dots, x_k)$ 
  if  $r_1$  in  $M(x_{s_1}, x_{o_1})$  and ...  $r_m$  in  $M(x_{s_m}, x_{o_m})$ 
  then  $op_1; \dots op_n$ 
end

```

per i permessi r_i , interi s_i e o_i , le operazioni primitive op_i , per esempio, si inserisce il privilegio p in $M(s, o)$ o si crea il soggetto s .

Istanze dello schema sopra riportato sono, esempio:

```

command create.file( $s, o$ )
  create  $o$ 
  enter  $Own$  into  $M(s, o)$ 
  enter  $R$  into  $M(s, o)$ 
  enter  $W$  into  $M(s, o)$ 
end

command confer.write( $s1, s2, o$ )
  if  $Own \in M(s1, o)$ 
  then enter  $W$  into  $M(s2, o)$ 
end

```

Operation	Conditions	New State
enter r into $M(s, o)$	$s \in S$ $o \in O$	$S' = S$ $O' = O$ $M'(s, o) = M(s, o) \cup \{r\}$ $M'(s_1, o_1) = M(s_1, o_1)$ for $(s_1, o_1) \neq (s, o)$
delete r from $M(s, o)$	$s \in S$ $o \in O$	$S' = S$ $O' = O$ $M'(s, o) = M(s, o) \setminus \{r\}$ $M'(s_1, o_1) = M(s_1, o_1)$ for $(s_1, o_1) \neq (s, o)$
create subject s'	$s' \notin S$	$S' = S \cup \{s'\}$ $O' = O \cup \{s'\}$ $M'(s, o) = M(s, o)$ for $s \in S, o \in O$ $M'(s', o) = \emptyset$ for $o \in O'$ $M'(s, s') = \emptyset$ for $s \in S'$
destroy subject s'	$s' \in S$	$S' = S \setminus \{s'\}$ $O' = O \setminus \{s'\}$ $M'(s, o) = M(s, o)$ for $s \in S', o \in O'$
create object o'	$o' \notin O$	$S' = S$ $O' = O \cup \{o'\}$ $M'(s, o) = M(s, o)$ for $s \in S, o \in O$ $M'(s, o') = \emptyset$ for $s \in S'$
destroy object o'	$o' \in O$ $o' \notin S$	$S' = S$ $O' = O \setminus \{o'\}$ $M'(s, o) = M(s, o)$ for $s \in S', o \in O'$

Oltre alle matrici degli accessi esistono altre strutture dati:

Lista del controllo degli accessi (ACL): viene usata una lista per esprimere ogni oggetto, l'i-esimo elemento della lista restituisce un soggetto e i suoi permessi sull'oggetto.

è solo il proprietario a poter garantire, revocare o decrementare i privilegi al file F agli altri utenti. Utilizzare linux per la gestione della memoria virtuale.

Capability list: vista del soggetto di una matrice del controllo degli accessi, è meno comune che la ACL (difficile capire chi ha i permessi su un oggetto e revocare permessi).
è usato nelle impostazioni delle applicazioni distribuite.

Unix fornisce un meccanismo basato su ACL semplificate adatto per una classe limitata di Criteri DAC.

- Controlla l'accesso per oggetto (file, directory, ...) utilizzando lo schema di autorizzazione proprietario/gruppo/altro.
- Gli oggetti vengono assegnati a un singolo utente (proprietario) e a un singolo gruppo (normalmente il gruppo della directory che lo contiene). Possono essere modificati usando, rispettivamente, `chown` e `chgrp` comandi.
- Ogni utente può essere assegnato a più gruppi (cfr. `Useradd`).
- Bit di autorizzazione assegnati agli oggetti dai rispettivi proprietari o dall'amministratore (`root`), (cfr. `chmod`).

Il file mode bits ha 2 parti:

File dei bit di permesso che controlla l'accesso ai file ordinario, i permessi di:

- lettura, per cartelle vuol dire poter vederne il contenuto;
- scrittura, modifica i file o, per le cartelle, crea e rimuove file;
- esecuzione, esegue un file o, per le cartelle, accedere al loro contenuto.

Special mode bits che riguarda solo gli eseguibili e le cartelle.

- imposta l'ID dell'user nel file al momento dell'esecuzione;
- imposta l'ID del gruppo nel file al momento dell'esecuzione;
- quando un utente diverso dal proprietario esegue il programma, verrà eseguito con tutti i permessi conferiti al proprietario;
- Impedisce agli utenti non privilegiati di rimuovere o rinominare un file in una directory a meno che possiedono il file o la cartelle; questo è chiamato "flag di eliminazione limitata" per cartelle e si trova comunemente su cartelle scrivibili da tutti come `/tmp`.

Problemi con le politiche discrezionali: Trojan

Le politiche discrezionali non distinguono gli utenti dai soggetti.

I primi sono entità passive per le quali sono specificabili autorizzazioni e chi si può connettere al sistema; quando gli utenti si collegano al sistema generano processi (soggetti) che vengono eseguiti per loro conto e inviano richieste al sistema.

Questo genere di politiche non fa distinzione fra le due entità, il processo verrà eseguito se l'utente che ne richiede l'esecuzione ne ha il permesso.

Questo rende questa politica vulnerabile a esecuzioni malevoli di programmi che sfruttano le autorizzazioni degli utenti per contro delle proprie esecuzioni.

Il sistema AC può dunque essere bypassato da un trojan incorporato nei programmi.

DEF Trojan Horse: è un programma che contiene funzioni nascoste che sfruttano surrettiziamente le legittime autorizzazioni del processo di invocazione.

Esso può usare qualsiasi autorizzazione dell'utente che ha eseguito il programma (es. può anche cancellare tutti i dati dell'utente).

Siccome il DAC non esegue controlli sulle esecuzioni, è possibile che un processo faccia trapelare informazioni, facendole leggere a utenti non autorizzati.

Tutto ciò avviene senza che l'amministratore o proprietario possa accorgersene, nonostante ogni singola richiesta sia controllata per quanto concerne le autorizzazioni.

Modello di sicurezza di Android: è un sistema operativo a privilegio-separato in cui ogni applicazione viene eseguita con un'identità distinta (allo stesso modo sono separate in identità distinte le parti del sistema).

Linux isola le applicazioni le une dalle altre e dal sistema (nessuna applicazione, di default, può interagire con altre applicazioni, con il sistema, o l'utente),

Altre caratteristiche di sicurezza a grana fine sono:

- un meccanismo di permessi che applichi restrizioni su operazioni specifiche che un processo particolare può eseguire, e
- permessi attraverso URI (Uniform Resource Identifier) per garantire accessi a specifici dati.

Mandatory Access Control (MAC)

Le decisioni del controllo degli accessi sono formalizzate (controllate) confrontando le security labels che indicano la criticità di un oggetto, con l'autorizzazione formale (es. autorizzazioni di sicurezza dei soggetti).

Restrizione dell'accesso a livello di sistema agli oggetti. Obbligatorio (Mandatory) perché i soggetti non possono trasferirsi i loro diritti di accesso. È più rigido del DAC, ma anche più sicuro.

Modelli del controllo degli accessi, di vari tipi:

- che raggruppano politiche per segretezza (Bell-LaPadula) o per integrità (Biba, Clark-Wilson);
- applicabili ad ambienti con politiche statiche (Bell-LaPadula), o dinamiche (Chinese Wall);
- possono essere informali (Clark-Wilson), semi-formali o formali (Bell-LaPadula, Harrison-Ruzzo-Ullman).

MAC: ordinamento lineare

Sono richiesti 2 principi per mantenere la confidenzialità:

- leggere in basso: l'autorizzazione di un soggetto deve essere superiore al grado di sicurezza dell'oggetto letto (es. un soggetto che ha la possibilità di leggere documenti top secret potrà leggere documenti non classificati)
- scrivere in alto: l'autorizzazione di un soggetto deve essere inferiore al livello di sicurezza dell'oggetto scritto (spesso ristretta ad un'uguaglianza di livello, per evitare che un soggetto sovrascriva dati che non può leggere).

Nel caso in cui un soggetto di livello superiore volesse scrivere documenti di livello inferiore, dovrebbe poter cambiare dinamicamente livello.

Per l'integrità è il contrario: leggere in alto, scrivere verso il basso.

DEF Reticolo (L, \leq): è un insieme di livelli di sicurezza L e un ordine parziale, tali che per ogni due elementi a, b appartenenti a L esiste un limite superiore minimo u appartiene a L e un massimo limite inferiore i appartenete a L . es.

$$\begin{aligned} a \leq u, b \leq u \quad \text{and} \quad (a \leq v \ \& \ b \leq v) \rightarrow (u \leq v) \text{ for all } v \in L \\ i \leq a, i \leq b \quad \text{and} \quad (k \leq a \ \& \ k \leq b) \rightarrow (k \leq i) \text{ for all } k \in L \end{aligned}$$

limite minimo superiore: dati 2 oggetti, cerco livello min sogg per leggerli entrambi.

limite massimo inferiore: dati 2 soggetti, cerco livello max ogg per essere entrambi letti.

Modello Bell-LaPadula (BLP)

Modella l'aspetto della riservatezza di un sistema multi utente, es. in un DBMS o OS. Probabilmente uno dei più famosi ed influenti modelli di sicurezza.

BLP modella la confidenzialità combinando aspetti di DAC e di MAC:

- accessi e permessi sono definiti entrambi tramite la matrice AC e i livelli di sicurezza.
- sicurezza multi-livello (MLS): politiche obbligatorie per prevenire il flusso di informazioni da un livello di sicurezza più alto ad uno più basso.
- BLP è un modello statico: i livelli di sicurezza non cambiano mai.

BLP definisce diverse proprietà di sicurezza per uno stato, per esempio:

- simple security property (ss-property): un soggetto s può leggere un oggetto o solo se il livello di sicurezza I_s di s è maggiore del livello I_o di o , es. $I_o \leq I_s$ (anche nota come no-read-up NRU, "non puoi leggere dati più protetti").
- *-property (star-property): un soggetto s può scrivere un oggetto o solo se il livello di sicurezza I_o di o è maggiore del livello di sicurezza di I_s di s , es. $I_s \leq I_o$ (anche nota come no-write-down NWD, "non puoi scrivere dati meno protetti").

NWD impedisce che soggetti di grado maggiore inviino dati a soggetti di grado inferiore e per far sì che i primi possano usare tutti i documenti necessari si può:

- far temporaneamente diminuire il grado dei primi ad uno inferiore;
- identificare un gruppo di soggetti fidati ai quali è permesso violare la *-property.

Il fondamento logico di queste proprietà è il non voler far trapelare informazioni.

Nonostante BLP sia adatto per modellare la riservatezza nei OS o nei DBMS, ha delle limitazioni, come:

- non specifica come cambiare i permessi o creare/distruggere oggetti/soggetti.
- è un modello statico (questa tranquillità ha sollevato molte controversie).
- contiene canali coperti, es. i flussi di informazione non vengono controllati.

Il primo problema può trovare soluzione usando il modello Harrison-Ruzzo-Ullman.

System Z ha solo una transizione di stato, che abbassa tutti i soggetti e gli oggetti al livello di sicurezza più basso, inserisce tutti i diritti di accesso in tutte le posizioni della matrice AC. il sistema Z soddisfa la nozione di sicurezza BLP.

caso contro BLP (McLean): BLP deve essere migliorato, come un sistema che può essere portato in uno stato in cui tutti possono leggere tutto ciò che non è sicuro.

pro caso di BLP (Bell): questo non è un problema di BLP ma piuttosto catturare in modo corretto il file requisiti di sicurezza. Se i requisiti dell'utente richiedono una tale transizione, allora dovrebbe essere consentita nel modello, altrimenti no.

il fondamento di questo disaccordo è uno stato di transizione che cambia i livelli di sicurezza (e i permessi di accesso). BLP è però un modello statico in cui i livelli di sicurezza sono fissi.

proprietà della tranquillità forte: i livelli di sicurezza di soggetti e oggetti mai cambiare durante il funzionamento del sistema.

proprietà della tranquillità debole: i livelli di sicurezza non cambiano mai in modo tale da violare una politica di sicurezza definita.

Questa limitazione viene eliminata nei modelli dinamici basati su BLP.

Talvolta non basta nascondere il contenuto degli oggetti, ma occorre nascondere anche la loro esistenza. In BLP, il meccanismo di AC può essere usato per costruire un canale coperto, dove i flussi di informazione passano da un livello di sicurezza più alto e scendono. Dire a un soggetto che una certa operazione non è consentita è già flusso di informazioni.

Modello Biba

Biba propone una classe di modelli volti a garantire l'integrità con le seguenti regole:

- modello di integrità obbligatorio: no-read-down e no-write-up
- Relax no-read-down: i soggetti non possono leggere ad un livello inferiore.
- Relax no-write-up: i soggetti non possono scrivere ad un livello superiore.

Biba e BLP possono essere combinati per un modello per la riservatezza e l'integrità.

Mira a garantire l'integrità degli accessi da parte dei soggetti agli oggetti usando un modello simile al BLP.

Affronta l'integrità in termini di accesso da parte dei soggetti agli oggetti utilizzando un modello simile a quello di BLP.

- Un reticolo (L, \leq) di livelli di sicurezza.
- $fS: S \rightarrow L$ e $fO: O \rightarrow L$ assegnano livelli di integrità a soggetti e oggetti.
- Le informazioni possono fluire solo verso il basso nel reticolo dell'integrità.

A differenza di BLP, non esiste una singola policy di integrità di alto livello ma piuttosto una varietà di policy (alcune addirittura incompatibili tra loro).

- Livelli a integrità statica.
- Livelli a integrità dinamica.
- Politiche per l'invocazione

Livelli a integrità statica

Politiche in cui i livelli di integrità non cambiano mai (come BLP tranquillity).

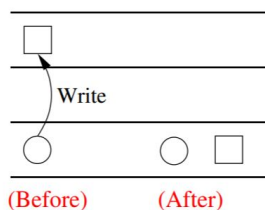
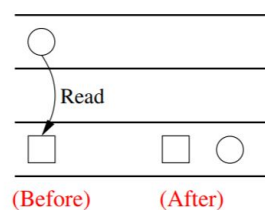
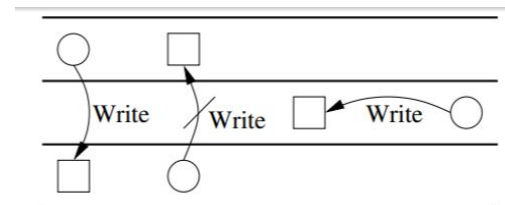
Due proprietà (dall'insieme delle politiche obbligatorie del BLP);

- no-write-up: s può modificare o se $fS(s) \geq fO(o)$
- no-read-down: s può leggere o se $fO(o) \geq fS(s)$

Livelli a integrità dinamica

Le proprietà del livello minimo (simili a Chinese Wall) regolano automaticamente il livello di integrità di un'entità se è entrata in contatto con informazioni di basso livello:

- Proprietà del livello minimo del soggetto: relax no-read-down.
Consenti a un soggetto di leggere, ma prima abbassa il suo livello di integrità a quello dell'oggetto da leggere.
Le operazioni di scrittura sono vincolate in base al principio di no-write-up.
Formalmente: s può leggere una o a qualsiasi livello di integrità.
Il nuovo livello di integrità di s è $\min(fS(s), fO(o))$, dove $fS(s)$ e $fO(o)$ sono i livelli di integrità prima dell'operazione.
- Proprietà del livello minimo dell'oggetto: relax no-write-up.
Abbassa il livello dell'oggetto a quello del soggetto che scrive.
Le operazioni di lettura sono vincolate secondo il principio di no-read-down.
 s può modificare una o a qualsiasi livello di integrità.
Il nuovo livello di integrità di o è $\min(fS(s), fO(o))$, dove $fS(s)$ e $fO(o)$ sono i livelli di integrità prima dell'operazione.



Modello Muraglia Cinese

Un modello di riservatezza di ispirazione commerciale (mentre la maggior parte dei modelli commerciali si concentra sull'integrità).

Modella le regole di accesso in un'azienda di consulenza in cui gli analisti devono assicurarsi che non sorgano conflitti di interesse quando lavorano con società diverse.

Informalmente, i conflitti sorgono perché: i clienti sono concorrenti diretti nello stesso mercato, o per la proprietà delle società.

Regola: non deve esserci flusso di informazioni che causi un conflitto di interessi.

Derivato dal Bell-LaPadula, con 3 livelli di astrazione:

1. compagnie, soggetti e oggetti:
Un insieme C di compagnie, e un insieme S di soggetti (gli analisti).
un insieme O di oggetti, con informazioni riguardanti una singola compagnia.
2. Tutti gli oggetti sulla stessa compagnia sono collezionati in un company dataset.
 $cd: O \rightarrow C$ restituisce l'insieme dei dati di ogni oggetto.
3. le classi di conflitto d'interesse indicano quali compagnie sono in competizione.
 $cic: O \rightarrow P(C)$ restituisce la classe di conflitto di interesse per ogni oggetto o , es.
l'insieme di aziende che non dovrebbero conoscere il contenuto di o .

la label di sicurezza di un oggetto o è la coppia $(cic(o), cd(o))$.

- Ogni oggetto dipende da un unico dataset aziendale.
- Ogni dataset aziendale è contenuto in un'unica classe di conflitto di interesse.
- Ogni classe di conflitto di interesse deve contenere una o più dataset aziendali.

i conflitti sorgono sia per gli oggetti attualmente affrontati, sia per quelli acceduti in passato.

Una matrice booleana $N: S \times O$ contiene le azioni dei soggetti:

$N(s, o) = \text{vero}$, se il soggetto s ha accesso a o , falso altrimenti

Uno stato di sicurezza iniziale imposta $N(s, o) = \text{falso}$ su tutta la matrice ($\forall s \in S, \forall o \in O$).

I permessi di accesso cambiano dinamicamente e devono essere riesaminati ad ogni transizione di stato: come un soggetto accede ad un oggetto, un altro oggetto che precedentemente era accessibile, ora non lo è più.

Una politica semplice per prevenire il conflitto di interessi: un soggetto può accedere a qualsiasi informazione purché non abbia mai avuto accesso alle informazioni di una società diversa nella stessa classe di conflitto.

Così, l'accesso è garantito se la l'oggetto o richiesto appartiene a:

- un dataset aziendale già posseduto da s (es. o appartiene ad un dataset di un'azienda già "visitata" da s),
- oppure ad una classe di conflitto d'interesse diversa (es. classe mai acceduta da s).

Formalmente, ss-property: s può accedere a o solo se per tutti o' con $N(s, o') = \text{true}$, si ha che $cd(o) = cd(o')$ oppure $cd(o) \notin cic(o')$.

Nonostante questa proprietà sono ancora possibili dei flussi di informazioni (es. due compagnie diverse si affidano alla stessa banca, chi ha accesso ai dati della banca potrebbe avere accesso ai dati sensibili, in relazione alla banca, delle due aziende concorrenti).

DEF informazione sanificata: Le informazioni sono sanificate se sono state eliminati i dettagli sensibili e non sono soggette a restrizioni di accesso. $cic(o) = \emptyset$ per un oggetto o sanificato.

Pertanto, concedere l'accesso in scrittura a un oggetto solo se non è possibile leggere nessun altro oggetto che si trova in un set di dati aziendale diverso e contiene informazioni non sanificate.

Formalmente *-property: a s viene concesso l'accesso in scrittura a o solo se s non ha accesso in lettura a un oggetto o' con $cd(o) \neq cd(o')$ e $cic(o') \neq \emptyset$.

Riassumendo:

- BLP: diritti di accesso sono assunti come statici.
- Chinese Wall: diritti di accesso cambiano dinamicamente, e devono essere riesaminati in ogni transizione di stato.

Modello per l'integrità di Clark-Wilson

è un modello informale per l'integrità, per le aziende, è differente da quelli orientati ai livelli (come BLP o Biba).

Gli oggetti sono divisi tra elementi di dati vincolati (CDI) e elementi di dati non vincolati (UDI).

Nove regole, basate sulle "best practice" aziendali, che vincolano

- il modo in cui l'integrità dei CDI deve essere convalidata,
- come e da chi possono essere modificati i CDI,
- come gli UDI possono essere modificati in CDI.

Ad esempio (regola 2): l'applicazione di una procedura di trasformazione a qualsiasi CDI deve mantenere l'integrità di tale CDI.

Controllo degli accessi basato sul ruolo (RBAC)

Questa modalità di AC è un'alternativa alle tradizionali DAC, MAC che è ampiamente usata nelle applicazioni commerciali; permette di specificare e forzare politiche di sicurezza specifiche dell'azienda in un modo che mappi naturalmente la struttura aziendale.

Per l'AC è più importante sapere che ruolo ricopre l'utente, piuttosto che sapere chi sia.

RBAC unisce la flessibilità di autorizzazioni esplicite (DAC) con vincoli aziendali addizionali (MAC).

Idea: scomporre la relazione soggetto/oggetto utilizzando i ruoli.

NB: s ha il permesso p su o se esiste un ruolo r tc. s ha il ruolo r e r ha il permesso p sull'oggetto o.

Ruoli e gruppi a confronto:

- i gruppi definiscono un insieme di utenti, i ruoli definiscono un insieme di privilegi.
- i ruoli posso essere attivati e disattivati dagli utenti a loro discrezione.
- l'appartenenza ad un gruppo è sempre attiva, non può essere dis/attivata a piacere.
- Tuttavia, poiché è possibile definire ruoli che corrispondono a figure organizzative (ad esempio, segretario, presidente e docente), uno stesso "concetto" può essere visto sia come gruppo che come ruolo.

RBAC semplifica di molto la gestione delle politiche di sicurezza:

- quando un nuovo utente si aggiunge all'organizzazione, l'amministratore deve solo concedergli il ruolo corrispondente al suo lavoro.
- quando un utente cambia lavoro, l'amministratore deve cambiare solo il ruolo associatogli.
- quando una nuova applicazione o task viene aggiunta al sistema, l'amministratore deve solo decidere a quale ruolo permetterne l'esecuzione.

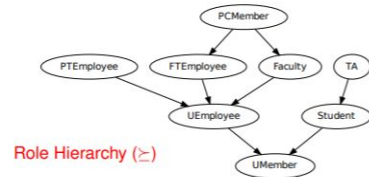
L'uso di gerarchie semplifica di molto la relazione di assegnazione dei permessi: s ha il permesso p su o sse esistono ruoli r e r' tc. $r \geq r'$, s ha il ruolo r, r' ha il permesso p sull'oggetto o.

Senza gerarchie

User Assignment (UA)		Permission Assignment (PA)	
User	Role	Role	Permission
Alice	PCMember	PCMember	GrantTenure
Bob	Faculty	PCMember	AssignGrades
Charlie	Faculty	PCMember	ReceiveHBenefits
David	TA	PCMember	UseGym
David	Student	Faculty	AssignGrades
Eve	UEmployee	Faculty	ReceiveHBenefits
Fred	Student	Faculty	UseGym
Greg	UMember	TA	AssignHWScores
		TA	Register4Courses
		TA	UseGym
		UEmployee	ReceiveHBenefits
		UEmployee	UseGym
		Student	Register4Courses
		Student	UseGym
		UMember	UseGym

Con gerarchie

User Assignment (UA)		Permission Assignment (PA)	
User	Role	Role	Permission
Alice	PCMember	PCMember	GrantTenure
Bob	Faculty	Faculty	AssignGrades
Charlie	Faculty	TA	AssignHWScores
David	TA	UEmployee	ReceiveHBenefits
David	Student	Student	Register4Courses
Eve	UEmployee	UMember	UseGym
Fred	Student		
Greg	UMember		



Ulteriori vantaggi di RBAC:

- Il minimo privilegio. I ruoli consentono a un utente di accedere con il privilegio minimo richiesto per la particolare attività che deve eseguire. Gli utenti autorizzati a ruoli potenti non devono esercitarli fino a quando questi privilegi non sono effettivamente necessari. Ciò riduce al minimo il pericolo di danni dovuti a errori involontari o Trojan.
- Separazione dei compiti. Principio secondo cui a nessun utente dovrebbe essere dati abbastanza privilegi per potergli permettere di abusare del sistema. Ad esempio, la persona che autorizza uno stipendio non dovrebbe essere la stessa persona che può prepararli. La separazione dei compiti può essere applicata staticamente o dinamicamente.
- Applicazione dei vincoli. I ruoli forniscono una base per la specifica e l'applicazione di ulteriori requisiti di protezione. Es, vincoli di cardinalità, che limitano il numero di utenti autorizzati ad attivare un ruolo o il numero di ruoli autorizzati ad esercitare un dato privilegio. I vincoli possono essere dinamici, es. essere imposti all'attivazione dei ruoli piuttosto che alla loro assegnazione.

La semplice relazione gerarchica non è sufficiente per modellare i diversi tipi di relazioni che possono verificarsi.

Ad esempio, potrebbe essere necessario consentire a una segretaria di scrivere documenti specifici per conto del suo manager, ma nessuno dei due ruoli è una specializzazione dell'altro.

Quindi dovrebbero essere supportati diversi modi di propagare i privilegi.

In alcuni casi, l'identità del richiedente deve essere considerata anche quando viene adottata una politica basata sui ruoli.

Ad esempio, un medico può essere autorizzato a specificare i trattamenti e accedere ai file, ma può essere limitato ai trattamenti e ai file per i propri pazienti, dove il rapporto medico-paziente è definito in base alla loro identità.

RBAC amministrativo: modifica le policy dei soggetti in policy amministrative.

Esistono diversi modelli: ARBAC97, SARBAC, Oracle DBMS, UARBAC, ...

Problema chiave: definizione di dominio amministrativo. Per esempio, per ARBAC: dominio basato sui ruoli, mentre per UARBAC il dominio è basato sugli attributi.

Sommario:

- Abbiamo visto diversi modelli di sicurezza per la riservatezza e l'integrità.
- Molti altri modelli di sicurezza (e tipi di modelli) sono stati proposti per affrontare varie minacce alla sicurezza:
- Modelli di non interferenza per sistemi deterministici.
- Modelli possibilistici per sistemi non deterministici.
- Modelli probabilistici per sistemi non deterministici
-
- Quale modello o quale combinazione di modelli scegliere dipende dalla particolare applicazione di sicurezza e dagli obiettivi.

Qualche parola sull'auditing (???):

- 2 componenti:
 - collezione e organizzazione di dati di verifica,
 - analisi dei dati per scoprire o diagnosticare una violazione della sicurezza.
- Rilevamento delle intrusioni:
 - passiva: analisi offline per rilevare possibili intrusioni o violazioni.
 - attiva: analisi in tempo reale per ottenere una risposta protettiva immediata.

Possibili soluzioni:

Rilevamento anomalie: lo sfruttamento delle vulnerabilità comporta un utilizzo anomalo del sistema.

Rilevamento degli abusi: basato su regole che specificano sequenze di eventi o proprietà osservabili del sistema, sintomatiche di violazioni.

I dati di verifica devono essere protetti dalla modifica da parte di un intruso.

(14_CS_System_Security-Access_Control_in_UNIX_&_Windows_NT.pdf)

Guardare direttamente lì

.