Appunti di network security

Riccardo Torre

21 novembre 2025

Indice

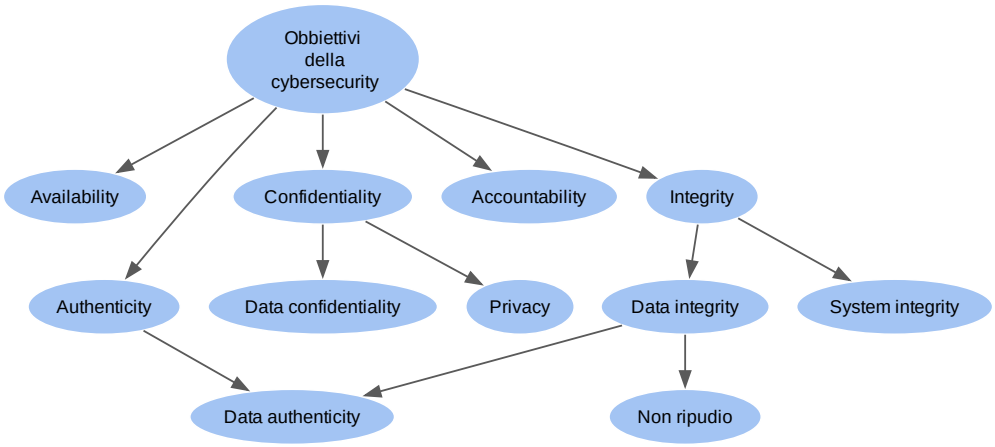
1 Introduzione

Cos'è la cybersecurity La *cybersecurity* o *sicurezza informatica*, è l'insieme dei mezzi, delle tecnologie e delle procedure tesi alla protezione dei sistemi informatici in termini di *confidenzialità*, *integrità* e *disponibilità* dei beni o asset informatici.

Information security Preserva la confidenzialità, l'integrità e la disponibilità delle *informazioni*. In aggiunta, altre proprietà come l'autenticità, il non ripudio e l'affidabilità possono essere incluse.

Network security Protegge le reti e i relativi servizi da modifiche non autorizzate, da distruzioni o divulgazioni non autorizzate e garantisce che la rete svolga correttamente le sue funzioni critiche e che non vi siano effetti collaterali dannosi.

1.1 Punti principali della cybersecurity



Availability (disponibilità) Garantisce che i sistemi funzionino tempestivamente e che il servizio non venga negato agli utenti autorizzati.

Confidenziality (condidenzialità) Si suddivide in:

- **data confidentiality:** assicura che le informazioni private e riservate non vengano rese disponibili o divulgate a soggetti non autorizzati;
- **privacy:** garantisce che gli individui controllino o influenzino quali informazioni a loro relative possano essere raccolte e archiviate e da che e a chi tali informazioni possano essere divulgate.

Integrity Si suddivide in:

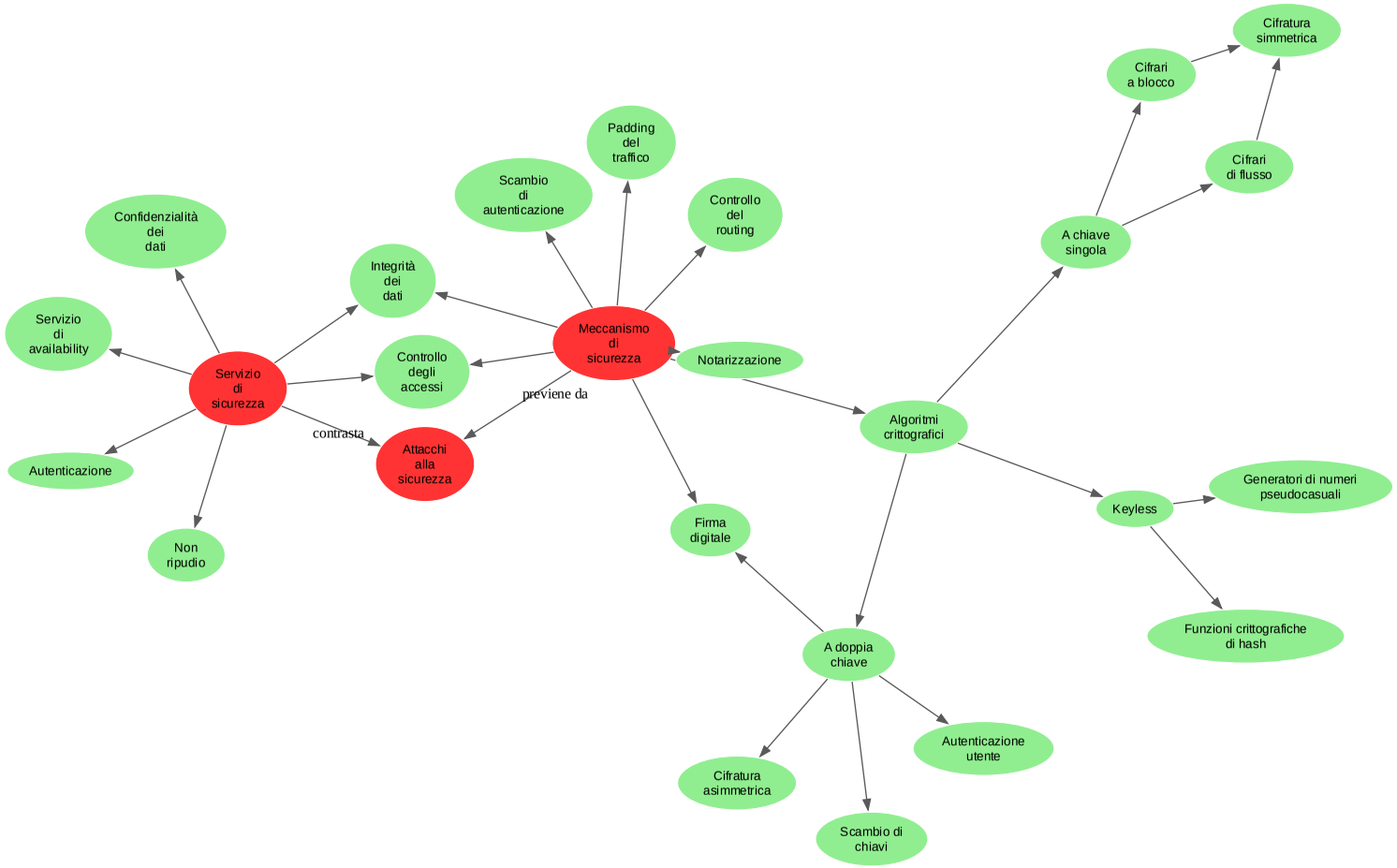
- **data integrity:** garantisce che dati e programmi vengano modificati solo in modo specificato e autorizzato. Questo concetto comprende anche *l'autenticità dei dati*, ovvero che un oggetto digitale è effettivamente ciò che dichiara di essere o ciò che viene dichiarato di essere, e il *non ripudio*, ovvero la garanzia che al mittente delle informazioni venga fornita una prova di consegna e al destinatario una prova dell'identità del mittente, in modo che nessuno dei due possa successivamente negare di aver elaborato le informazioni;
- **system integrity (integrità di sistema):** assicura che un sistema svolga la sua funzione prevista in modo inalterato, libero da manipolazioni non autorizzate, deliberate o involontarie.

Authentity (autentitcità) La proprietà di essere autentico e di poter essere verificato e considerato attendibile; la fiducia nella validità di una trasmissione, di un messaggio o del mittente del messaggio. Ciò significa verificare che gli utenti siano chi dicono di essere e che ogni input che arriva al sistema provenga da una fonte attendibile.

Accountability (responsabilità) È l'obbiettivo di sicurezza che genera il requisito che le azioni di un'entità siano riconducibili in modo univoco a tale entità. Ciò supporta il *non ripudio*, la deterrenza, l'isolamento dei guasti, il rilevamento e la prevenzione delle intrusioni, il ripristino post-azione e le azioni legali. Poiché sistemi veramente sicuri non sono ancora un obbiettivo raggiungibile, dobbiamo essere in grado di ricondurre una violazione di sicurezza al responsabile. I sistemi devono conservare registrazioni delle loro attività per consentire successive analisi forensi volte a rintracciare le violazioni della sicurezza o a facilitare la risoluzione di controversie sulle transazioni.

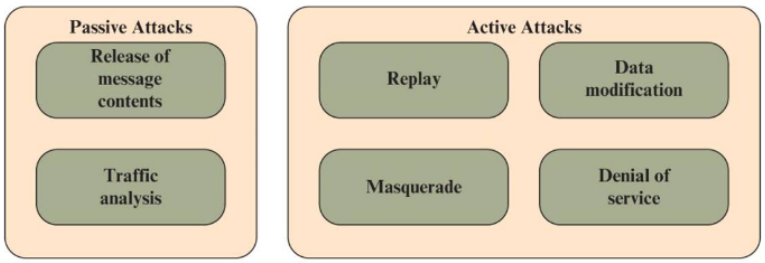
1.2 Architettura OSI di sicurezza

- **Security attack:** è una qualsiasi azione che compromette la sicurezza delle informazioni possedute da un'organizzazione.
- **Security mechanism:** è un processo (o un dispositivo che incorpora tale processo) che è progettato per rilevare, prevenire, o ripristinare da un security attack.
- **Security service:** è un servizio di elaborazione o comunicazione che migliora la sicurezza dei sistemi di elaborazione dati e dei trasferimenti di informazione di un'organizzazione. È destinato a contrastare gli attacchi alla sicurezza e utilizza uno o più meccanismi di sicurezza per fornire il servizio.


Servizio di sicurezza
Confidenzialità dei dati
Servizio di availability
Autenticazione
Non ripudio
Integrità dei dati
Controllo degli accessi
Attacchi alla sicurezza
contrasta
Meccanismo di sicurezza
previene da
Scambio di autenticazione
Padding del traffico
Controllo del routing
Notarizzazione
Algoritmi crittografici
A chiave singola
Cifrari a blocco
Cifratura simmetrica
Cifrari di flusso
Keyless
Generatori di numeri pseudocasuali
Funzioni crittografiche di hash
Firma digitale
A doppia chiave
Autenticazione utente
Cifratura asimmetrica
Scambio di chiavi

1.3 Minacce e attacchi

- **Minaccia:** un potenziale per la violazione della sicurezza, che esiste quando c'è una circostanza, capacità, azione o evento che potrebbe compromettere la sicurezza e causare danni. Cioè, una minaccia è un possibile pericolo che potrebbe sfruttare una vulnerabilità.
- **Attacco:** è un assalto alla sicurezza del sistema che deriva da una minaccia intelligente; cioè, un atto intelligente che è un tentativo deliberato (soprattutto nel senso di un metodo o tecnica) di eludere i servizi di sicurezza e violare la politica di sicurezza di un sistema.



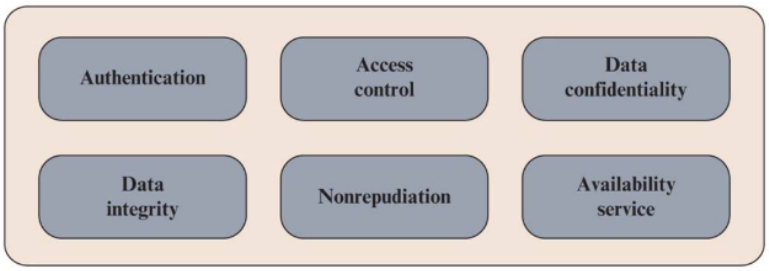
(a) Attacks

Attacchi passivi Tentano di apprendere o utilizzare informazioni dal sistema ma non influenzano sulle risorse del sistema ascoltando o monitorando le trasmissioni; lo scopo dell'attaccante è di ottenere le informazioni che sono state trasmesse. Vi sono due tipi di attacchi passivi:

- **rilascio dei contenuti dei messaggi:** possono contenere informazioni confidenziali o sensibili); ad esempio una conversazione al telefono, una mail elettronica, un file trasferito; i messaggi nono sono mascherati;
- **l'analisi del traffico:** i messaggi sono mascherati con la crittografia; l'attaccante inferisce il contenuto dei messaggi in base a frequenza e lunghezza;

Attacchi attivi Tentano di alterare le risorse del sistema o influenzare il loro funzionamento comportando una modifica del flusso di dato o la creazione di un flusso falso. Sono difficili da prevenire a causa della vasta gamma di potenziali vulnerabilità fisiche, software e di rete. I tipi di attacco possono essere:

- **masquerade:** si verifica quando un'entità finge di essere un'altra entità e di solito include una delle altre forme di attacco attivo;
- **replay:** comporta la cattura passiva di un'unità di dati e la sua successiva ritrasmissione per produrre un effetto non autorizzato;
- **data modification:** alcune parti di un messaggio legittimo vengono alterate, oppure i messaggi vengono ritardati o riordinati per produrre un effetto non autorizzato;
- **denial of service:** preclude o inibisce l'uso normale o la gestione delle strutture di comunicazione.



(b) Services

Autenticazione si occupa di garantire che una comunicazione sia autentica:

- nel caso di un singolo messaggio, assicura che il messaggio provenga dalla fonte che afferma di essere;
- nel caso di un'interazione continua, assicura che le due entità siano autentiche e che la connessione non sia interferita in modo tale che una terza parte possa mascherarsi come una delle due parti legittime;

e si suddivide in:

- **autenticazione delle entità peer:** fornisce la conferma dell'identità di un'entità peer in un'associazione. Due entità sono considerate peer se implementano lo stesso protocollo in sistemi diversi. L'autenticazione delle entità peer è prevista per l'uso al momento dell'instaurazione di una connessione, o in determinati momenti durante la fase di trasferimento dei dati in una connessione. Essa cerca di fornire fiducia che un'entità non stia eseguendo né masquerading, né una ripetizione non autorizzata di una connessione precedente.
- **autenticazione all'origine dei dati:** fornisce la conferma della fonte di un'unità dati. Non offre protezione contro la duplicazione o la modifica delle unità di dati. Questo tipo di servizio supporta applicazioni come la posta elettronica, dove non ci sono interazioni in corso tra le entità comunicanti.

Controllo degli accessi La capacità di limitare e controllare l'accesso ai sistemi host e alle applicazioni tramite collegamenti di comunicazione. Per raggiungere questo obbiettivo, ogni entità che cerca di ottenere accesso deve prima essere identificata o autenticata, in modo che i diritti di accesso possano essere personalizzati per l'individuo.

Riservatezza dei dati Deve prevenire la riservatezza dei dati da attacchi passivi proteggendo il contenuto dei messaggi scambiati tra gli utenti, andando a proteggere l'intero messaggio o anche limitandosi a specifici campi al suo interno; da attacchi attivi onde evitare che un attaccante non possa inferire informazioni interessanti (sorgente, destinazione, frequenza, lunghezza,…) facendo analisi del traffico.

Integrità dei dati Può applicarsi a un flusso di messaggi, a un singolo messaggio o a campi selezionati all'interno di un messaggio.

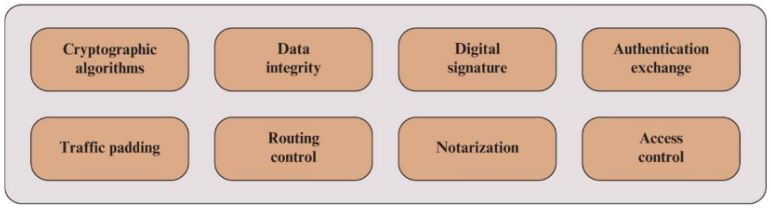
- **Servizio di integrità orientato alla connessione:** si occupa di un flusso di messaggi, assicura vengano ricevuti così come sono stati inviati, senza duplicazioni, inserimenti, modifiche, riordini o ripetizioni.

- **Servizio di integrità senza connessione:** si occupa di messaggi individuali senza considerare un contesto più ampio, fornisce generalmente protezione contro la modifica dei messaggi.

Non ripudio Previene che il mittente o il destinatario possano negare un messaggio trasmesso. Quando un messaggio viene inviato, il destinatario può dimostrare che il presunto mittente ha effettivamente inviato il messaggio. Quando un servizio viene ricevuto, il mittente può dimostrare che il presunto destinatario ha effettivamente ricevuto il messaggio.

Servizio di disponibilità Protegge un sistema per garantire la sua disponibilità. Questo servizio affronta le preoccupazioni di sicurezza sollevate dagli attacchi di negazione del servizio e dipende dalla corretta gestione e controllo delle risorse di sistema, quindi dipende dal servizio di controllo degli accessi e da altri servizi di sicurezza.

1.4 Meccanismi di sicurezza



(c) Mechanisms

Algoritmi crittografici Si differenziano in meccanismi crittografici *reversibili* e *irreversibili*. Un meccanismo crittografico reversibile è un algoritmo di crittografia che consente di crittografare i dati e successivamente decrittografarli. I meccanismi crittografici irreversibili includono *algoritmi di hash* e *codici di autenticazione dei messaggi*, che sono utilizzati nelle applicazioni di firma digitale e autenticazione dei messaggi.

Integrità dei dati Include meccanismi utilizzati per garantire l'integrità di un'unità dati o di un flusso di dati.

Firma digitale Consiste nell'aggiungere dati o effettuare una trasformazione crittografica ad un'unità di dati per provare la fonte e l'integrità dell'unita dati e proteggerla dalla falsificazione.

Scambio di autenticazione Meccanismo destinato a garantire l'identità di un'entità mediante uno scambio di informazioni.

Padding del traffico L'inserimento di bit in spazi vuoti in un flusso di dati per scoraggiare tentativi di analisi del traffico.

Controllo del routing Consente la selezione di percorsi fisicamente o logicamente sicuri per determinati dati e permette modifiche al routing, specialmente quando si sospetta una violazione della sicurezza.

Notarizzazione L'uso di una terza parte fidata per garantire determinate proprietà di uno scambio di dati.

Controllo degli accessi Una varietà di meccanismi che applicano i diritti di accesso alle risorse.

1.4.1 Algoritmi senza chiave

Tra gli algoritmi senza chiave vi sono:

- **funzioni deterministiche:** hanno determinate proprietà utili per la crittografia. Un tipo di algoritmo senza chiave è la *funzione crittografica di hash* che trasforma una quantità variabile di testo in un valore di piccole dimensioni e di lunghezza fissa chiamato hash, codice hash, o *digest*. Una funzione hash crittografica può essere parte di un altro algoritmo crittografico, come un codice di autenticazione dei messaggi o una firma digitale;
- **generatore di numeri pseudocasuali:** produce una sequenza deterministica di numeri o bit che ha l'apparenza di essere una sequenza veramente casuale.

1.4.2 Algoritmi a chiave singola

Algoritmi a chiave simmetrica Dipendono dall'uso di una chiave segreta e sono denominati solitamente *algoritmi di crittografia simmetrica*. Un algoritmo di crittografia simmetrica prende come input alcuni dati da proteggere e una chiave segreta e produce una trasformazione illeggibile su quei dati. Un corrispondente algoritmo di decrittazione prende i dati trasformati e la stessa chiave segreta e recupera i dati originali.

I crittosistemi a chiave singola possono essere classificati in:

- **cifrario a blocchi:** opera sui dati come una sequenza di blocchi. Nella maggior parte delle versioni del cifrario a blocchi, conosciute come modalità di operazione, la trasformazione dipende non solo dall'attuale blocco di dati e dalla chiave segreta, ma anche dal contenuto dei blocchi precedenti;
- **cifrario a flusso:** opera sui dati come una sequenza di bit; la trasformazione dipende da una chiave segreta.

Codice MAC Algoritmo crittografico a chiave singola. Un MAC è un elemento di dati associato a un blocco dati o a un messaggio. Viene generato da una trasformazione crittografica che coinvolge l'uso di una chiave segreta e, tipicamente, una funzione hash crittografica del messaggio. Il MAC è progettato in modo che chiunque sia in possesso della chiave segreta possa verificare l'integrità del messaggio. Il destinatario del messaggio con il MAC può eseguire la stessa computazione sul messaggio; se il MAC calcolato corrisponde al MAC che accompagna il messaggio, ciò fornisce la certezza che il messaggio non sia stato alterato.

1.4.3 Algoritmi asimmetrici

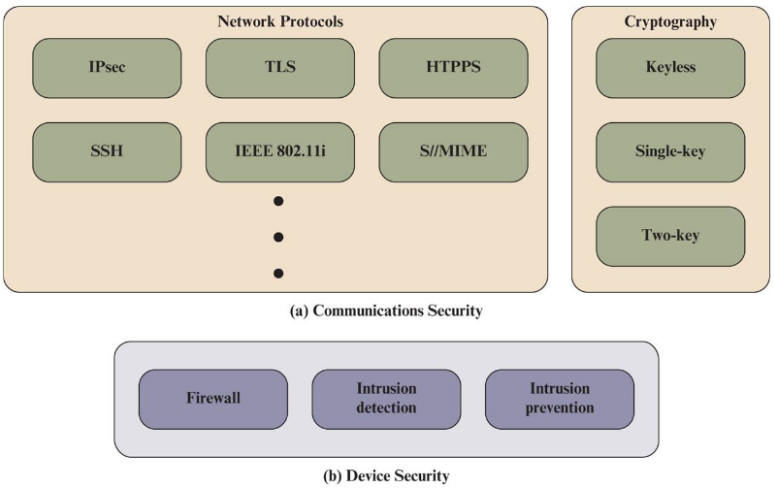
Sono algoritmi di crittografia che usano una coppia di chiavi.

Algoritmo di firma digitale Calcola la *firma digitale* e la associa ad un oggetto di dati in modo tale che qualsiasi destinatario dei dati possa utilizzare la firma per verificare l'origine e l'integrità dei dati.

Scambio di chiavi È il processo di distribuzione sicura di una chiave simmetrica a due o più parti.

Autenticazione dell'utente È il processo di autenticazione che verifica che un utente che tenta di accedere ad un'applicazione o a un servizio sia genuino e, allo stesso modo, che l'applicazione o il servizio sia genuino.

1.5 Elementi chiave della sicurezza delle reti



Sicurezza delle comunicazioni Si occupa della protezione delle comunicazioni attraverso la rete comprese le misure per proteggere contro attacchi sia passivi che attivi. La sicurezza delle comunicazioni è principalmente implementata utilizzando protocolli di rete. Un *protocollo di rete* consiste nel formato e nelle procedure che governano la trasmissione e la ricezione di dati tra punti in una rete. Un *protocollo* gestisce la struttura delle singole unità di dati e i comandi di controllo che gestiscono il trasferimento dei dati.

Rispetto alla sicurezza di rete, un protocollo di sicurezza può essere un miglioramento che fa parte di un protocollo esistente o autonomo.

Sicurezza dei device Protegge i dispositivi di rete come router e switch, sistemi finali connessi alla rete, come client e server.

Le principali preoccupazioni di sicurezza sono gli intrusi che ottengono accesso al sistema per eseguire azioni non autorizzate, inserire software dannoso (malware) o sovraccaricare le risorse di sistema per ridurre la disponibilità.

Tre tipi di sicurezza dei dispositivi sono:

- **firewall:** capacità hardware/software che limita l'accesso tra una rete e i dispositivi collegati alla rete, in conformità con una specifica politica di sicurezza. Il firewall agisce come un filtro che consente o nega il traffico dati, sia in entrata che in uscita, basandosi su un insieme di regole basate sul contenuto del traffico e/o sul modello di traffico;

- **rilevamento delle intrusioni:** prodotti hardware o software che raccolgono e analizzano informazioni da varie aree all'interno di un computer o di una rete al fine di trovare e fornire avvisi in tempo reale o quasi in tempo reale di tentativi di accesso alle risorse di sistema in modo non autorizzato;
- **prevenzione delle intrusioni:** prodotti hardware o software progettati per rilevare attività intrusive e tentare di fermare l'attività, idealmente prima che raggiunga il suo obbiettivo.

2 Crittografia simmetrica

I due requisiti fondamentali per un uso sicuro della crittografia simmetrica sono:

- un algoritmo di crittografia forte;
- il mittente e il destinatario devono aver ottenuto copie della chiave segreta in modo sicuro e devono mantenere la chiave al sicuro.

I sistemi crittografici sono generalmente classificati lungo tre dimensioni indipendenti:

1. **sostituzione:** ogni elemento nel testo in chiaro è mappato in un altro elemento;
2. **trasposizione:** gli elementi nel testo in chiaro vengono riarrangiatati. Il requisito fondamentale è che non venga persa alcuna informazione.

I **sistemi a prodotto** comportano più fasi di sostituzioni e trasposizioni.

2.1 Tecniche di attacco della crittoanalisi

Ciphertext only Il crittoanalista ha accesso solo al testo cifrato. È il caso più frequente ed è quello che fornisce meno informazioni all'attaccante.

Known plaintext L'attaccante ha un insieme di testi cifrati dei quali conosce i corrispondenti testi in chiaro. A primo avviso può sembrare una situazione improbabile, ma i casi in cui l'attaccante può conoscere sia il testo in chiaro che quello cifrato non sono rari: ad esempio, molti messaggi di e-mail iniziano con frasi ricorrenti o terminano con firme predefinite.

Chosen plaintext In questo caso l'attaccante può scegliere arbitrariamente i testi da cifrare. Anche questa situazione non è improbabile: nella crittografia asimmetrica, la chiave pubblica è nota a tutti e può essere utilizzata per cifrare quanti messaggi si desidera.

Chosen ciphertext L'attaccante può ottenere i testi in chiaro corrispondenti da un insieme arbitrario di testi cifrati. L'informazione ignota è in questo caso la chiave crittografica.

Chosen text Questo attacco combina gli approcci dei precedenti. Il crittoanalista può scegliere sia un messaggio in chiaro che un testo cifrato e ottenere le rispettive versioni cifrate e decifrate. Questo fornisce una quantità significativa di informazioni, rendendo più facile per il crittoanalista analizzare e compromettere la crittografia.

2.2 Crittoanalisi

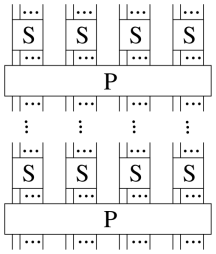
Uno schema di crittografia è considerato sicuro dal punto di vista computazionale se il testo cifrato generato dallo schema soddisfa uno o entrambi i seguenti criteri:

- il costo per rompere il cifrario supera il valore delle informazioni cifrate;
- il tempo necessario per rompere il cifrario supera la vita utile delle informazioni.

Attacco di forza bruta Comporta il tentativo di ogni possibile chiave fino a ottenere una traduzione comprensibile del testo cifrato in testo in chiaro. In media, deve essere provata metà di tutte le chiavi possibili per raggiungere il successo. A meno che non venga fornito un testo in chiaro noto, l'analista deve essere in grado di riconoscere il testo in chiaro come tale. Per integrare l'approccio di forza bruta è necessaria una certa conoscenza sul testo in chiaro previsto e avere un mezzo per distinguere automaticamente il testo in chiaro dal testo confuso.

2.3 Claude Shannon e i cifrari a permutazione e sostituzione

Nel 1949, Calude Shannon introdusse l'idea delle reti di sostituzione-permutazione (S-P), le quali formano la base dei cifrari a blocchi moderni. Le reti S-P si basavano sulle operazioni delle due operazioni crittografiche primitive di **sostituzione (S-box)** e **permutazione(P-box)** che forniscono confusione e diffusione del messaggio.

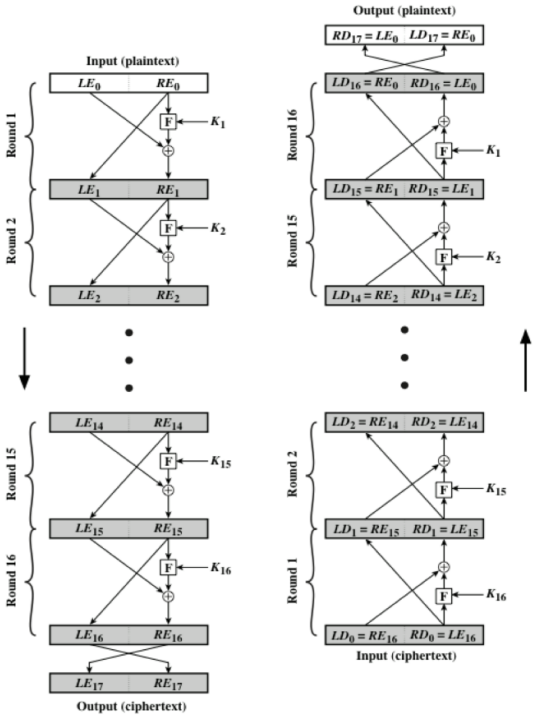


Operazione	Box	Effetto
Sostituzione	S-box	Confusione: rende la relazione statistica tra il testo cifrato e la chiave il più complessa possibile
Permutazione	P-box	Diffusione: dissipa la struttura statistica del testo in chiaro su gran parte del testo cifrato

Cifrari a prodotto Combinano catene di sostituzioni e trasposizioni. Le S-box confondono i bit di input e le P-box diffondono i bit attraverso gli input delle S-box. Un pad monouso oscura completamente le proprietà statistiche del messaggio originale.

Cifrario di Feistel Ideato da Horst Feistel, è basato sul concetto di cifrario di prodotto invertibile. Il crittosistema prende in input un blocco che viene diviso a metà, successivamente elabora attraverso più round eseguendo una sostituzione sulla metà sinistra, mentre sulla metà destra viene applicata la funzione di round a cui viene applicata la sottochiave. Quindi le due metà vengono scambiate. Questo cifrario implementa il concetto di rete di sostituzione-permutazione di Shannon. Gli elementi del cifrario di Feistel sono:

- **dimensione del blocco:** dimensioni di blocco maggiori significano maggiore sicurezza (maggiore diffusione) ma riducono la velocità di crittografia/decrittazione;
- **dimensione della chiave:** dimensioni di chiave maggiori significano maggiore sicurezza (maggiore resistenza ai bruteforce attacks e maggiore confusione) ma possono diminuire la velocità di crittografia/decrittazione;



- **numero di round:** l'essenza di un cifrario a blocchi simmetrico è che un singolo round offre sicurezza inadeguata, mentre più round offrono una sicurezza crescente;
- **algoritmo di generazione delle sottochiavi:** maggiore complessità in questo algoritmo dovrebbe portare a una maggiore difficoltà nella crittoanalisi;
- **funzione di round:** maggiore complessità generalmente significa maggiore resistenza alla crittoanalisi; una dimensione tipica è di 16 rounds;
- **facilità di analisi:** se l'algoritmo può essere spiegato in modo coinciso e chiaro, è più facile analizzarlo per vulnerabilità crittoanalitiche e quindi sviluppare un livello più elevato di garanzia sulla sua robustezza.

2.4 Algoritmi di crittografia a blocchi simmetrici

Cifrario a blocchi Elabora l'input in chiaro in blocchi di dimensioni fisse e produce un blocco di testo cifrato di dimensioni uguali per ogni blocco di testo in chiaro. Sono gli algoritmi di crittografia simmetrica più comunemente usati. I tre cifrari a blocco simmetrici più importanti sono:

- DES – Data Encryption Standard;
- 3DES – Triple DES;
- Advanced encryption Standard AES.

2.4.1 DES – Data Encryption Standard

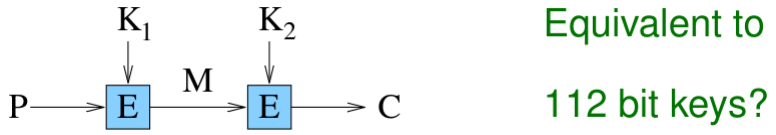
La sicurezza del DES è stata a lungo messa in discussione. Ci sono state molte speculazioni sulla lunghezza della chiave, sul numero di iterazioni e sul design delle S-box (coinvolgimento dell'NSA).

La struttura del DES è una leggera variazione della rete Feistel. Ci sono 16 round di elaborazione, il testo in chiaro è lungo 64 bit (bit di input), la chiave è lunga 56 bit. DES è stato attaccato nel 1999 in 22 ore con un computer da un milione di dollari.

I punti critici di DES sono:

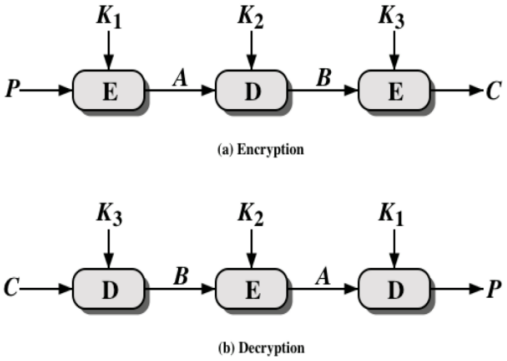
- **chiavi a 56 bit:** lo spazio delle chiavi è circa $7,2 \times 10^{16}$; un singolo PC che esegue una cifratura al μs impiegherebbe oltre 1000 anni, ma macchine parallele o hardware moderno riducono molto i tempi. Un singolo PC moderno può rompere DES in circa un anno; con più PC o supercomputer il tempo scende a ore. Le chiavi grandi almeno 128 bit sono praticamente inattaccabili con brute-force;
- **natura dell'algoritmo (S-box):** le otto S-box di DES sono state oggetto di sospetti perché i criteri di progettazione non furono pubblici; sono state trovate regolarità ma non sono emerse debolezze decisive sfruttabili per una crittoanalisi completa;
- **attacchi di timing:** osservando i tempi di decrittazione si possono ricavare informazioni (es. peso di Hamming della chiave). Studi indicano che DES è relativamente resistente a questi attacchi e la tecnica, al momento, non sembra praticabile per compromettere DES, Triple DES o AES.

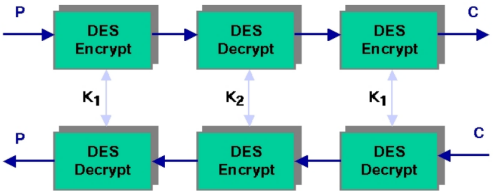
2DES Per aumentare la sicurezza del DES, si passò al 2DES – Double DES. L'idea è quella di eseguire due crittografie. Si creano due chiavi K_1 e K_2 e si utilizza un processo di crittografia $E_{K_2}(E_{K_1}(M))$. Questo non è equivalente ad utilizzare chiavi da 112 bit. Infatti è possibile attaccare DDES con un attacco di tipo **meet-in-the-middle**



Attacco plaintext contro il 2DES (meet-in-the-middle) Per $C = E_{K_2}(E_{K_1}(P))$ si pone $X = E_{K_1}(P) = D_{K_2}(C)$. Dato un P e un C noti, crittografare P per tutte le 2^{56} possibili K_1. Memorizzare in una tabella ordinata per X. Decrittare C con tutte le 2^{56} possibili K_2 e cercare una corrispondenza. Ogni corrispondenza è una soluzione candidata. Validare con un ulteriore coppia di plaintext/ciphertext.

3DES–Triple DES Utilizza tre fasi di crittografia invece di due. La compatibilità viene mantenuta con il DES standard (si pone $K_2 = K_1$). **Vantiaggi:** la lunghezza della chiave di 168 bit supera la vulnerabilità agli attacchi di forza bruta del DES e l'algoritmo di crittografia sottostante è lo stesso del DES. **Svantaggi:** l'algoritmo è lento in software e utilizza una dimensione di blocco di 64 bit.





2.4.2 AES – Advanced Encryption Standard

Nato per sostituire il 3DES. Nel 1997 il NIST ha emesso una richiesta di proposte per un nuovo AES:

- dovrebbe avere una forza di sicurezza pari o superiore a quella del 3DES e un'efficienza significativamente migliorata;
- deve essere una crittografia simmetrica a blocchi con una lunghezza di blocco di 128 bit e supporto per lunghezze di chiave di 128, 192 e 256 bit;
- i criteri di valutazione includevano sicurezza, efficienza computazionale, requisiti di memoria, idoneità hardware e software, e flessibilità.

Il NIST ha selezionato **Rinjdael** come algoritmo AES proposto (2 sviluppatori belgi Dr. Joan Daemen e Dr. Vincent Rijmen).

Svantaggi del DES Algoritmo progettato per l'implementazione hardware degli anni '70. Esegue lentamente nelle implementazioni software. 3DES è 3 volte più lento a causa dei 3 round. La dimensione del blocco di 64 bit deve essere aumentata per velocizzare le operazioni.

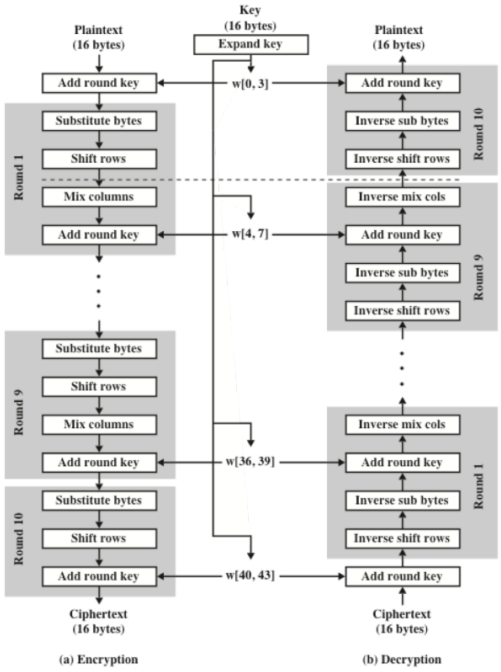
Panoramica dell'AES Ha dimensioni del blocco di 128, 192 e 256 bit (128 è il più comune). Non è una struttura di Feistel, elabora l'intero blocco in parallelo. La chiave di 128 bit viene espansa in 44 parole da 32 bit con 4 parole utilizzate per ogni round.

2.5 Numeri casuali e pseudocasuali

Un certo numero di algoritmi di sicurezza di rete basati sulla crittografia fa uso di numeri casuali. Ad esempio nella generazione di chiavi per l'algoritmo di crittografia a chiave pubblica RSA e altri algoritmi a chiave pubblica e nella generazione di una chiave simmetrica da utilizzare come chiave di sessione temporanea; utilizzata in diverse applicazioni di rete come la sicurezza del livello di trasporto, WiFi, sicurezza delle e-mail e sicurezza IP.

In diversi scenari della distribuzione delle chiavi , come Kerberos, i numeri casuali vengono utilizzati per l'handshake per prevenire attacchi di replay.

Due requisiti distinti e non necessariamente compatibili per una sequenza di numeri casuali sono:



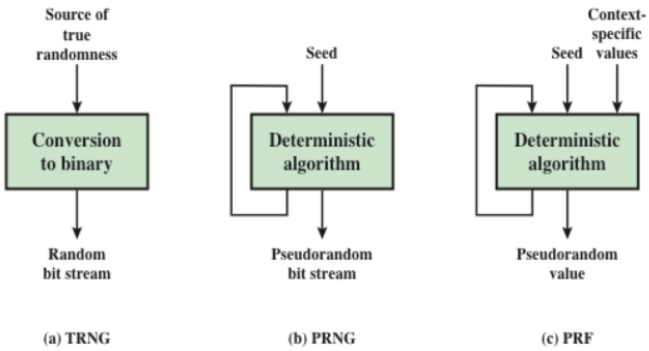
Randomness Tradizionalmente, la preoccupazione nella generazione di una sequenza di numeri presumibilmente casuali è stata che la sequenza di numeri fosse casuale in qualche ben definito senso statistico. I seguenti due criteri sono utilizzati per convalidare che una sequenza di numeri sia casuale:

- **distribuzione uniforme:** la distribuzione dei bit nella sequenza dovrebbe essere uniforme; cioè, la frequenza di occorrenza di uno e zero dovrebbe essere approssimativamente uguale.
- **indipendenza:** nessuna sottosequenza nella sequenza può essere dedotta dalle altre.

Non predicibilità In applicazioni come l'autenticazione reciproca e la generazione di chiavi di sessione, il requisito non è tanto che la sequenza di numeri sia statisticamente casuale, ma che i membri successivi della sequenza siano imprevedibili. Con le sequenze "vere" casuali, ogni numero è statisticamente indipendente dagli altri numeri nella sequenza e quindi imprevedibile. È necessario prestare attenzione affinché un avversario non possa prevedere elementi futuri della sequenza sulla base degli elementi precedenti.

TRNG, PRNG, PRF Le applicazioni crittografiche fanno uso di tecniche algoritmiche per la generazione di numeri casuali che sono deterministiche e che producono sequenze di numeri che non sono statisticamente casuali e che vengono definiti numeri pseudocasuali. Non sono realmente casuali ma sembrano casuali.

TRNG (True Random Number Generator) Prende come input una sorgente che è casuale e che è solitamente chiamata **sorgente di entropia**. Sostanzialmente, una sorgente di entropia è presa dall'ambiente fisico di un computer e può includere ad esempio i tempi con cui vengono battuti i tasti della tastiera, le attività elettriche del disco, i movimenti del mouse e valori



- The seed of PRNG is often generated by a TRNG
- PRF are often used to build symmetric keys and nonces, where context specific values are user ID or app ID.

istantanei del clock di sistema. La sorgente o una combinazione di sorgenti, serve come input all'algoritmo che produce un output binario casuale (questo è realmente casuale). I TRNG potrebbero coinvolgere conversioni di una sorgente analogica per produrre un output binario. I TRNG potrebbero includere processamento aggiuntivo per far fronte a qualsiasi bias nella sorgente.

PRNG Un PRNG prende come input un valore fissato chiamato **seme** e produce una sequenza di bit in output usando un algoritmo deterministico. Molte volte il seme viene generato da una TRNG. Tipicamente esiste un percorso di feedback tramite il quale alcuni dei risultati prodotti dall'algoritmo vengono reimmessi come input mentre vengono prodotti ulteriori bit di output. È importante notare che il flusso di bit in uscita è determinato esclusivamente dal valore o dai valori di input, in modo che un avversario che conosce l'algoritmo e il seme possa riprodurre l'intero flusso di bit.

PRF Un PRF è usato per produrre una stringa di bit pseudocasuale di una qualche lunghezza fissata. Esempi sono la crittografia delle chiavi simmetriche e i nonces. Di solito, il PRF prende come input un seme più alcuni valori dal contesto specifico, come ad esempio l'ID dell'utente o l'ID di un'applicazione.

Per creare i PRNG vengono solitamente utilizzate tre categorie di algoritmi crittografici:

- cifrari simmetrici a blocchi;
- cirfrari asimmetrici;
- funzioni hash e codici di autenticazione dei messaggi.

Considerazioni sul design dei cifrari a flusso

- La sequenza di crittografia dovrebbe avere un lungo periodo: più lungo è il periodo di ripetizione, più difficile sarà effettuare la crittoanalisi.

- Il flusso di chiavi dovrebbe approssimare le proprietà di un vero flusso di numeri casuali il più possibile: più il flusso di chiavi appare casuale, più il testo cifrato è randomizzato, rendendo la crittoanalisi più difficile.
- Il generatore di numeri pseudocasuali è condizionato dal valore della chiave di input: per proteggersi dagli attacchi di forza bruta, la chiave deve essere sufficientemente lunga; con la tecnologia attuale, è auspicabile una lunghezza della chiave di almeno 128 bit.

Algoritmo RC4

- Dal 2015 non viene più utilizzato in TLS perché sono state trovate diverse vulnerabilità di sicurezza.
- È un cifrario a flusso progettato nel 1987 da Ron Rivest per RSA security, con dimensione della chiave variabile e operazioni orientate ai byte.
- L'algoritmo si basa sull'uso di una permutazione casuale.
- È utilizzato negli standard Secure Sockets Layer/Transport Layer Security (SSL/TLS) definiti per la comunicazione tra browser Web e server.
- Utilizzato anche nel protocollo WEP (Wired Equivalent Privacy) e nel più recente protocollo WPA (WiFi Protected Access), che fanno parte dello standard IEEE 802.11 per le reti LAN wireless.

2.6 Modalità di operazione dei cifrari a blocco

Un cifrario simmetrico a blocchi elabora un blocco di dati alla volta. Nel caso di DES e 3DES, la lunghezza del blocco è di 64 bit. Per AES, la lunghezza del blocco è di 128 bit. Per quantità maggiori di testo in chiaro, è necessario suddividere il testo in blocchi di b bit, aggiungendo padding all'ultimo blocco se necessario.

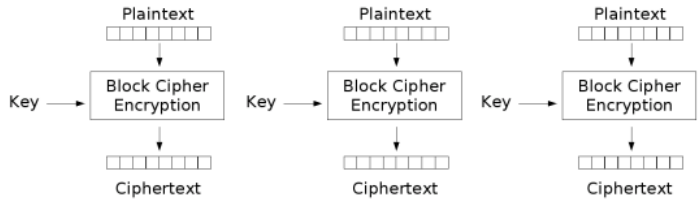
2.6.1 Le 5 modalità di operazione definite dal NIST

Cinque modalità di operazione sono state definite dal NIST, destinate a coprire praticamente tutte le possibili applicazioni di crittografia per le quali un cifrario a blocchi potrebbe essere utilizzato, inclusi tutti i cifrari simmetrici a blocchi tra cui 3DES e AES.

ECB (Electronic Codebook) La modalità ECB è la più semplice. Il testo in chiaro viene gestito 64 bit per volta; ognuno dei blocchi di 64 bit viene cifrato con la stessa chiave. Per messaggi più lunghi di 64 bit, si procede suddividendo il messaggio in blocchi di 64 bit, utilizzando, se necessario, bit di riempimento nell'ultimo blocco. Per una data chiave, esiste un unico testo cifrato per ogni blocco di testo in chiaro di 64 bit; in altre parole, se nel messaggio compare più volte lo stesso blocco di 64 bit di testo in chiaro, verrà prodotto sempre lo stesso testo cifrato. Per questo motivo, il metodo ECB è ideale per limitati volumi di dati (ad esempio, la trasmissione di una chiave di cifratura), poiché per messaggi più lunghi potrebbe essere insicuro: se si trattasse di dover cifrare un messaggio molto strutturato, l'analisi crittografica potrebbe sfruttarne le regolarità.

L'ECB presenta i seguenti vantaggi:

- cifratura parallelizzabile;



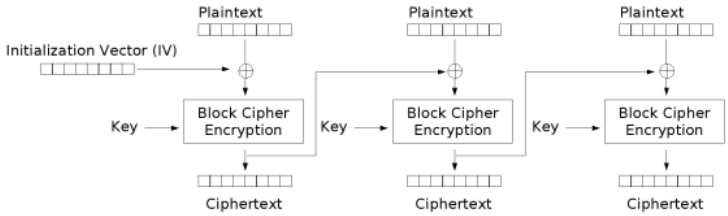
Electronic Codebook (ECB) mode encryption

- gli errori rimangono localizzati, quindi non si ha propagazione di errore sui diversi blocchi.

Presenta anche svantaggi:

- non è garantita l'integrità del messaggio, poiché un attaccante potrebbe invertire dei blocchi e la vittima non se ne accorgerebbe;
- usa una chiave fissa che quindi può essere predisposta ad attacchi di analisi crittografica;
- non è sicuro contro attacchi di forza bruta.

CBC (Cipher Block Chaining) Per superare i limiti di sicurezza di ECB, è necessario l'utilizzo di una tecnica in cui lo stesso blocco di testo in chiaro, se ripetuto, produce blocchi di testo cifrato differenti. Questo, è ciò che accade con la modalità CBC, in cui l'input dell'algoritmo di crittografia è il risultato dello XOR tra il blocco di testo in chiaro corrente e il blocco di testo cifrato precedente; per ciascun blocco viene utilizzata la stessa chiave. In fase di decifratura,



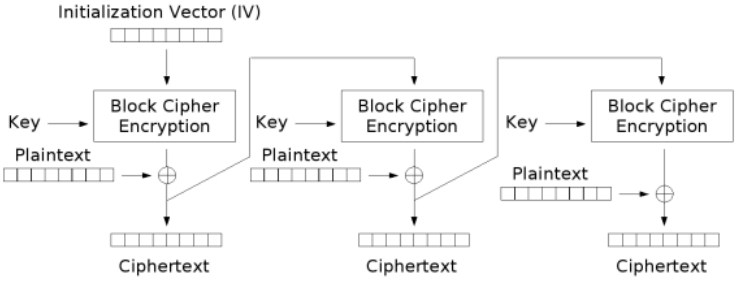
Cipher Block Chaining (CBC) mode encryption

ciascun blocco di testo cifrato passa attraverso l'algoritmo di decrittografia; il risultato subisce uno XOR con il blocco di testo precedente, per produrre il blocco di testo in chiaro.

Per produrre il primo blocco di testo cifrato, lo XOR viene effettuato tra un vettore di inizializzazione IV (dall'inglese *initialization vector*) e il primo blocco di testo in chiaro. In decifratura, l'IV subisce uno XOR con l'output dell'algoritmo di decrittografia in modo da ottenere nuovamente il primo blocco di testo in chiaro. Il vettore di inizializzazione IV deve essere dunque noto non solo al mittente ma anche al destinatario, che tipicamente lo riceve assieme alla chiave; entrambi i valori vengono cifrati in modalità ECB. Per aumentare il livello di sicurezza ed evitare il replay attack, si utilizza un IV casuale per ogni processo di cifratura. Questo permette di produrre un testo cifrato differente anche a parità di testo in chiaro e chiave di crittografia. In questi casi è desiderabile poter evitare di condividere con il destinatario ogni IV generato;

questo è possibile, semplicemente premettendo al testo in chiaro un blocco di testo casuale: in questo modo verrà generato un primo blocco cifrato comunque utilizzabile nel processo di cifratura. Implementando questa modalità, in fase di decifratura sarà necessario generare un nuovo IV casuale, quindi scartare il primo blocco di testo in chiaro che verrà prodotto. È possibile utilizzare un IV casuale e senza la necessità di condividerne ogni valore con il destinatario, anche nei cifrari a blocco in modalità CFB e OFB. In conclusione, questa modalità è la più appropriata per cifrare messaggi più lunghi di 64 bit. Inoltre, la modalità CBC può essere utilizzata anche per l'autenticazione.

CFB (Cipher Feedback) La modalità CFB è stata ideata per convertire idealmente una cifratura a blocchi in una cifratura a flusso. La cifratura a flusso non necessita di eseguire riempimenti e può inoltre operare in tempo reale. Nell'operazione di cifratura, l'input della funzione di crittografia è un registro a scorrimento a 64 bit che inizialmente viene importato con un vettore di inizializzazione IV. Gli s bit più significativi (ovvero quelli più a sinistra) dell'output subiscono uno XOR con il primo segmento di testo in chiaro P_1 per produrre la prima unità di testo cifrato C_1. Il contenuto del registro di scorrimento viene fatto scorrere a sinistra di s bit,



Cipher Feedback (CFB) mode encryption

e negli s bit meno significativi (quelli più a destra) del registro viene inserito C_1. Il processo viene reiterato fino all'esaurimento di tutte le unità di testo in chiaro. All'atto della decifratura, si utilizza il medesimo schema, tranne per il fatto che le unità di testo cifrato ricevute sono sottoposte ad uno XOR con l'output della funzione di crittografia. Non si utilizza dunque la funzione di decrittografia, perché se $S_s(x)$ sono gli s bit più significativi di X, allora:

$$C_1 = P_1 \oplus S_s(E_K(IV))$$

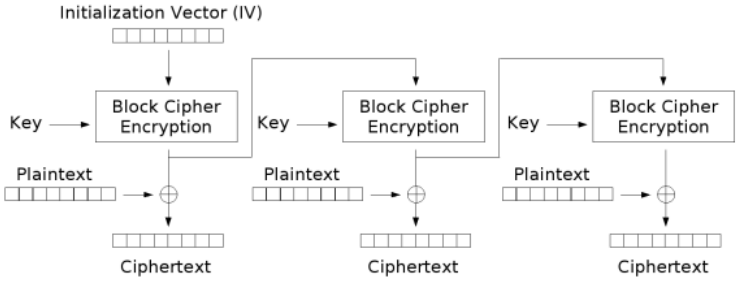
Pertanto

$$P_1 = C_1 \oplus S_s(E_K(IV))$$

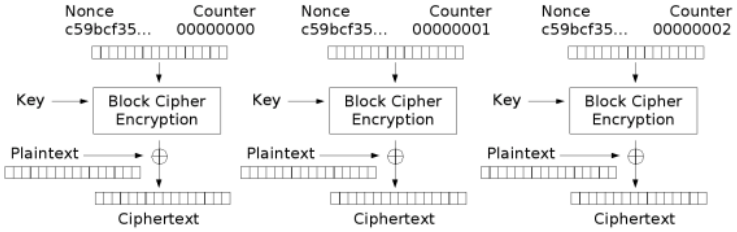
e lo stesso ragionamento vale per i passi successivi.

OFB La modalità OFB è molto simile alla CFB. Il vantaggio della OFB è che non propaga gli errori di trasmissione dei bit. Il suo svantaggio è che è più vulnerabile a un attacco a modifica del flusso dei messaggi.

CTR (Counter) Questa quinta modalità è stata introdotta successivamente alle altre quattro, per poter essere applicata a ATM (Asynchronous Transfer Mode) e a IPsec (IP security). In questa modalità viene utilizzato un contatore corrispondente alle dimensioni del blocco di testo



Output Feedback (OFB) mode encryption



Counter (CTR) mode encryption

in chiaro. Il requisito essenziale è che il suo valore sia differente per ciascun blocco da cifrare; in genere viene inizializzato con un determinato valore e poi incrementato di un'unità per ogni blocco successivo (modulo 2^b dove b corrisponde alle dimensioni del blocco). Per la cifratura, il contatore viene crittografato e poi si applica uno XOR col blocco di testo in chiaro per produrre il blocco di testo cifrato. Per la decifratura si utilizza la stessa sequenza di valori del contatore ai quali si applica lo XOR con i blocchi di testo cifrato.

I vantaggi del CTR sono:

- efficienza dell'hardware;
- efficienza del software;
- pre-elaborazioni;
- accesso diretto;
- sicurezza dimostrabile;
- semplicità.

Sicurezza dimostrabile (provable security) In crittografia un sistema crittografico presenta una sicurezza dimostrabile se i suoi requisiti di sicurezza possono essere fissati formalmente in un modello con precisi assunti, dove colui che cerca di violare il sistema (comunemente denominato "avversario") ha accesso allo stesso ed ha abbastanza risorse computazionali per cercare di forzarlo. La dimostrazione di sicurezza (chiamata "riduzione") è che questi requisiti di sicurezza

Modalità	Applicazioni tipiche
ECB	Trasmissione sicura di singoli valori
CBC	• Trasmissione di carattere generale orientata ai blocchi • Autenticazione
CFB	• Trasmissione di carattere generale orientata al flusso di dati • Autenticazione
OFB	Trasmissione orientata al flusso di dati su canali rumorosi
CTR	• Trasmissione di carattere generale orientata ai blocchi • Utile per requisiti di alta velocità

siano soddisfatti stabilito che le ipotesi riguardanti l'accesso dell'avversario al sistema siano soddisfatte e che siano chiaramente indicate quelle inerenti alla difficoltà computazionale di alcuni calcoli. Esempi di requisiti sono stati pubblicati nel 1989 da Shafi Goldwasser e Silvio Micali.

La sicurezza dimostrabile si basa principalmente su due nozioni di sicurezza:

- la sicurezza semantica;
- la sicurezza computazionale.

La prima nozione di sicurezza, antecedente a quella di sicurezza dimostrabile, è quella di sicurezza perfetta ed è stata introdotta da Claude Shannon nel suo celebre articolo *Comunication theory of secrecy systems* pubblicato nel 1949. Come egli stesso ha dimostrato, esiste solo un sistema che è stato provato sia incondizionatamente sicuro, il cifrario di Vernam, dove la chiave crittografica è lunga quanto il testo da cifrare: senza di essa è provatamente impossibile risalire ad alcuna informazione inerente al messaggio in chiaro.

La nozione di sicurezza dimostrabile, invece, si basa sul fatto che, se non si dispone che di una limitata capacità computazionale, sarà possibile risalire al messaggio in chiaro solo con una probabilità detta trascurabile, ovvero minima.

Posizionamento della crittografia

- **Crittografia a livello di collegamento:** avviene in modo indipendente su ogni collegamento. Il traffico deve essere decrittografato e ricrittografato a ogni collegamento. Richiede molti dispositivi, ma con chiavi abbinate.
- **Crittografia end-to-end:** avviene tra la sorgente originale e la destinazione finale. Sono necessari dispositivi a ciascun estremo. Richiede chiavi condivise tra i due punti finali della comunicazione.

3 Distribuzione delle chiavi

Gli schemi crittografici simmetrici richiedono che entrambe le parti condividano una chiave segreta comune, che deve essere distribuita in modo sicuro attraverso un canale affidabile prima dell'inizio della comunicazione cifrata.

Approcci alla distribuzione delle chiavi Le parti A e B hanno diverse alternative per la distribuzione delle chiavi:

- A può selezionare la chiave e consegnarla fisicamente a B;
  - richiede la consegna manuale della chiave. È ragionevole per la cifratura di collegamento (*link encryption*) tra due dispositivi partner ma scomodo e poco pratico per la cifratura end-to-end su una rete distribuita, dove sono necessarie molte chiavi dinamiche;
- una terza parte può selezionare e consegnare la chiave ad A e B;
  - richiede la consegna manuale della chiave ma è inefficace per la cifratura end-to-end a causa della necessità di distribuire un gran numero di chiavi dinamicamente in sistemi distribuiti;
- se A e B hanno comunicato in precedenza, possono utilizzare la chiave precedente per crittografare una nuova chiave;
  - è una possibilità sia per la cifratura di collegamento che per quella end-to-end. Tuttavia, se un aggressore ottiene l'accesso a una chiave, tutte le chiavi successive saranno rilevate. Inoltre, rimane irrisolto il problema della distribuzione iniziale di milioni di chiavi;
- se A e B hanno comunicazioni sicure con una terza parte C, C può trasmettere la chiave tra A e B;
  - ampiamente adottata per la cifratura end-to-end. Utilizza un centro di distribuzione chiavi (KDC) responsabile della consegna delle chiavi quando necessario. Ogni utente deve condividere una chiave univoca con il KDC per la distribuzione delle chiavi.

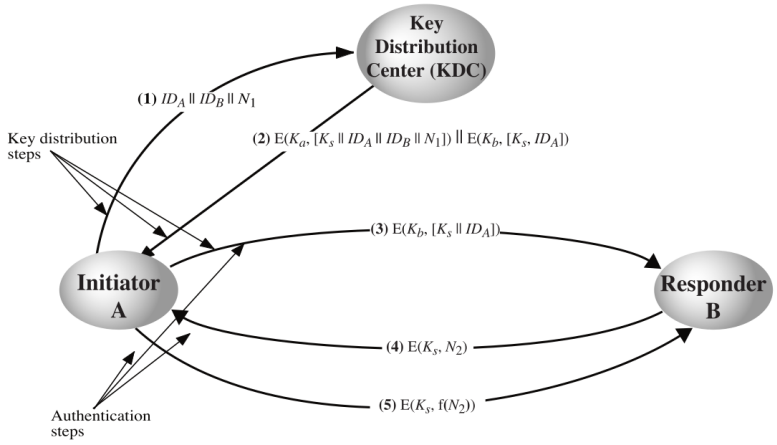
Tipicamente viene utilizzata una gerarchia delle chiavi:

- **Chiave di sessione:** utilizzata per la crittografia dei dati tra gli utenti per una sessione logica, quindi scartata.
- **Chiave master:** utilizzata per crittografare le chiavi di sessione.

La gerarchia delle chiavi viene condivisa tra l'utente e il centro di distribuzione delle chiavi.

Problemi di distribuzione delle chiavi

- Sono necessarie gerarchie di KDC (Key Distribution Center) per reti di grandi dimensioni, ma devono fidarsi l'uno dell'altro.
- La durata delle chiavi di sessione dovrebbe essere limitata per una maggiore sicurezza.
- Utilizzo della distribuzione automatica delle chiavi per conto degli utenti, ma è necessario fidarsi del sistema.
- Utilizzo della distribuzione decentralizzata delle chiavi.
- Controllo dell'uso delle chiavi.



Problema della scalabilità La scala del problema dipende dal livello di cifratura:

- **Cifratura a livello di rete (IP):** è necessaria una chiave per ogni coppia di host, con un numero di chiavi pari a $\frac{n(n-1)}{2}$, dove n è il numero di host.
- **Cifratura a livello di applicazione:** è necessaria una chiave per ogni coppia di utenti/processi. Poiché ci sono molti più utenti/processi che host, il numero di chiavi richieste è molto più alto (ad esempio, 50 milioni di chiavi per 10.000 applicazioni rispetto a mezzo milione per 1000 nodi).

3.1 Algoritmo RSA

Algoritmo di crittografia asimmetrica, inventato nel 1977 da Ronald Rivest, Adi Shamir e Leonard Aldeman utilizzabile per cifrare o firmare informazioni.

L'algoritmo RSA si basa sulla difficoltà di fattorizzare un numero molto grande in due numeri primi. Quindi, anche se qualcuno ha accesso all'informazione cifrata e alla chiave pubblica, è molto difficile per loro scoprire la chiave privata che è necessaria per decodificare il messaggio. Questa caratteristica rende l'algoritmo RSA molto sicuro e per questo viene utilizzato per proteggere molte comunicazioni online, come per esempio i pagamenti online o le comunicazioni via e-mail.

Funzionamento base

1. si scelgono a caso due numeri primi p e q abbastanza grandi da garantire la sicurezza dell'algoritmo (per esempio, il più grande numero RSA, RSA-2048, utilizza due numeri primi lunghi più di 300 cifre);
2. si calcola il loro prodotto $n = pq$, chiamato *modulo* (dato che tutta l'aritmetica seguente è modulo n), e il prodotto $\varphi(n) = (p-1)(q-1)$, dove $\varphi(n)$ è la funzione toziente;
3. si considera che la fattorizzazione di n è segreta e solo chi sceglie i due numeri primi, p e q, la conosce;
4. si sceglie poi un numero e (chiamato *esponente pubblico*), coprimo con $\varphi(n)$ e più piccolo di $\varphi(n)$;

5. si calcola il numero d (chiamato esponente *privato*) tale che il suo prodotto con e sia congruo a 1 modulo $\varphi(n)$ ovvero che $ed \equiv 1 \mod \varphi(n)$; per calcolare d si utilizza l'algoritmo esteso di Euclide.

La chiave pubblica è (n, e), mentre la chiave privata è (n, d).

La forza dell'algoritmo sta nel fatto che per calcolare d da e (o viceversa) non basta la conoscenza di n ma serve il numero $\phi(n) = (p-1)(q-1)$, e che il suo calcolo richiede tempi molto elevati; infatti fattorizzare in numeri primi (cioè scomporre un numero nei suoi divisori primi) è un'operazione computazionalmente costosa.

Un messaggio m viene cifrato attraverso l'operazione $m^e \mod n$ trasformandolo nel messaggio cifrato c. Una volta trasmesso, c viene decifrato con $c^d \mod n = m$. Il procedimento funziona solo se $m < n$ e la chiave e utilizzata per cifrare e la chiave d utilizzata per decifrare sono legate tra loro dalla relazione $ed \equiv 1 \mod \varphi(n)$, quindi quando un messaggio viene cifrato con una delle due chiavi può essere decifrato solo utilizzando l'altra. L'algoritmo si basa sull'assunzione mai dimostrata (nota come assunzione RSA) che il problema di calcolare $\sqrt[e]{c} \mod n$, con n numero composto di cui non si conoscono i fattori, sia computazionalmente non trattabile.

La firma digitale non è altro che l'inverso: il messaggio viene crittografato con la chiave privata, in modo che chiunque possa, utilizzando la chiave pubblica conosciuta da tutti decifrarlo e, oltre a poterlo leggere in chiaro, essere certo che il messaggio è stato mandato dal possessore della chiave privata corrispondente a quella pubblica utilizzata per leggerlo.

Per motivi di efficienza e comodità, normalmente viene inviato il messaggio in chiaro con allegata la firma digitale di un hash del messaggio stesso; in questo modo il ricevente può direttamente leggere il messaggio (che è in chiaro), utilizzare la chiave pubblica per estrarre l'hash della firma e verificare che questo sia uguale a quello calcolato localmente sul messaggio ricevuto. Se l'hash è crittograficamente sicuro, la corrispondenza dei due valori conferma che il messaggio ricevuto è identico a quello originalmente firmato e trasmesso.

Fondamenti matematici La decifratura del messaggio è assicurata grazie ad alcuni teoremi matematici; infatti dal calcolo si ottiene:

$$e^d \mod n = (m^e)^d \mod n = m^{ed} \mod n$$

Ma sappiamo che $ed \equiv 1 \mod (p-1)(q-1)$, di conseguenza abbiamo che $ed \equiv 1 \mod (p-1)$ e che $ed \equiv 1 \mod (q-1)$. Quindi per il piccolo teorema di Fermat:

$$m^{ed} \equiv m \mod p \quad \text{e} \quad m^{ed} \equiv \mod q$$

Siccome p e q sono numeri diversi e primi, possiamo applicare il teorema cinese del resto, ottenendo che

$$m^{ed} \equiv m \mod pq$$

e quindi che

$$e^d \equiv m \mod n$$

Teorema 1 (Piccolo teorema di Fermat) *Se p è un numero primo, allora per ogni intero a:*

$$a^p \equiv a \mod p$$

Questo significa che se si prende un qualunque numero a, lo si moltiplica per se stesso p volte e si sottrae a, il risultato è divisibile per p. È spesso espresso nella forma equivalente: se p è primo e a è un intero coprimo con p, allora:

$$a^{p-1} \equiv 1 \mod p$$

Teorema 2 (Teorema cinese del resto) *Si supponga che $n_1, ..., n_k$ siano interi a due a due coprimi (il che significa che $\gcd(n_i, n_j) = 1$ quando $i \neq j$). Allora, comunque si scelgano degli interi $a_1, \ldots, a_k$, esiste un intero x soluzione del sistema di congruenze*

$$x \equiv a_i \mod n_i \quad \textit{per } i = 1, \ldots, k$$

Inoltre, tutte le soluzioni x di questo sistema sono congruenti modulo il prodotto $n = n_1, \ldots, n_k$.

Si può trovare una soluzione x come segue. Per ogni i gli interi n_i e $\frac{n}{n_i}$ sono coprimi, e utilizzando l'algoritmo di Euclide esteso si possono trovare due interi r e s tali che $rn_i + s\frac{n}{n_i} = 1$. Ponendo $e_i = s\frac{n}{n_i}$, si ottiene

$$e_i \equiv 1 \mod n_i \quad \text{e} \quad e_i \equiv 0 \mod n_j \quad \text{per} \quad j \neq i$$

Una soluzione del sistema di congruenze è quindi:

$$x = \sum_{i=1}^{k} a_i e_i$$

Usare il teorema cinese del resto nell'algoritmo di RSA permette di trasportare i calcoli dall'anello Z_n all'anello $Z_p \times Z_q$. La somma delle dimensioni in bit di p e q è la dimensione in bit di n, in questo modo i calcoli vengono molto semplificati.

Riepilogo RSA

- seleziona p, q dove p e q sono entrambi numeri primi, $p \neq q$
- calcola $n = p \times q$
- calcola $\varphi(n) = (p-1)(q-1)$
- seleziona un intero e tale che il massimo comun divisore tra $\varphi(n)$ e e è 1, ovvero

  $$\gcd(\varphi(n), e) = 1$$

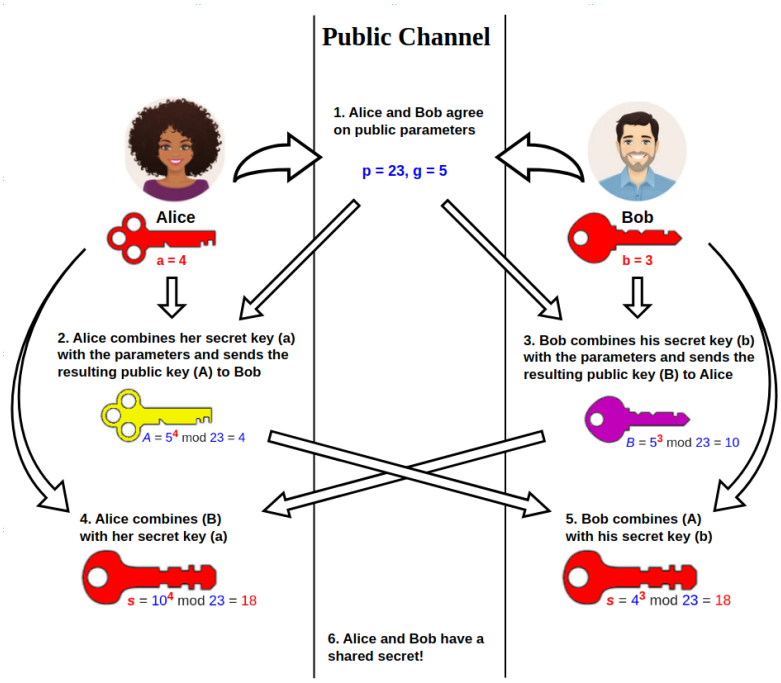
  e

  $$1 < e < \varphi(n)$$

- calcola d tale che:

  $$de \mod \varphi(n) = 1$$

- chiave pubblica $P_k = \{e, n\}$
- chiave privata $S_k = \{d, n\}$
- plaintext $M < n$
- cifratura: $C = M^e \mod n$
- decifratura:$M = C^d \mod n$



Considerazioni sulla sicurezza La sicurezza di RSA dipende dal suo utilizzo in modo da contrastare potenziali attacchi. Gli approcci di attacco sono:

- **attacchi matematici:** fattorizzazione dei numeri primi per derivare p e q da n; per contrastare la fattorizzazione, utilizzare chiavi di grande dimensione: almeno 2048 bit;
- **attacchi di temporizzazione:** per dedurre la dimensione delle chiavi dal tempo di decrittazione;
- **attacchi chosen-ciphertext:** per messaggi piccoli, potrebbe essere fattibile eseguire attacchi di forza bruta; utilizzare padding.

3.2 Scambio di chiavi Diffie-Hellman

Il protocollo Diffie-Hellman-Merkle (comunemente noto come DH) è un protocollo crittografico che consente a due entità (spesso chiamate Alice e Bob) che non si conoscono in precedenza di stabilire congiuntamente una chiave segreta condivisa su un canale di comunicazione non sicuro (pubblico). Questa chiave segreta può essere poi utilizzata per cifrare le comunicazioni successive tramite un algoritmo di cifratura simmetrica.

Storia Il concetto è stato pubblicato per la prima volta da Whitfield Diffie e Martin Hellman nel 1976. È stato il primo metodo pratico per lo scambio di chiavi segrete su un canale non autenticato senza conoscenza preliminare tra le parti, ed è considerato una delle invenzioni fondamentali della crittografia a chiave pubblica. Successivamente è stato riconosciuto che Ralph Merkle aveva sviluppato un metodo simile in precedenza, portando talvolta alla denominazione più completa di "scambio di chiavi Diffie-Hellman-Merle".

Descrizione del funzionamento Il metodo si basa sulla difficoltà computazionale del **problema del logaritmo discreto.**

1. **Parametri pubblici:** Alice e Bob concordano pubblicamente su due numeri:
   - p: un numero primo grande;
   - g: una radice primitiva modulo p (o un generatore).
2. **Chiavi private:**
   - Alice sceglie un intero segreto a;
   - Bob sceglie un intero segreto b.
3. **Chiavi pubbliche scambiate:**
   - Alice calcola il suo valore pubblico $A = g^a \mod p$ e lo invia a Bob;
   - Bob calcola il suo valore pubblico $B = g^b \mod p$ e lo invia ad Alice.
4. **Calcolo della chiave segreta condivisa:**
   - Alice calcola la chiave segreta K usando il valore pubblico di Bob e la sua chiave privata: $K_A = B^a \mod p$;
   - Bob calcola la chiave segreta K usando il valore pubblico di Alice e la sua chiave privata: $K_B = A^b \mod p$.

Grazie alle proprietà dell'aritmetica modulare, si ha che:

$$K_A = (g^b)^a \mod p = g^{ba} \mod p$$
$$K_B = (g^a)^b \mod p = g^{ab} \mod p$$

Quindi $K_A = K_B$, e Alice e Bob arrivano allo stesso segreto condiviso $K = g^{ab} \mod p$.

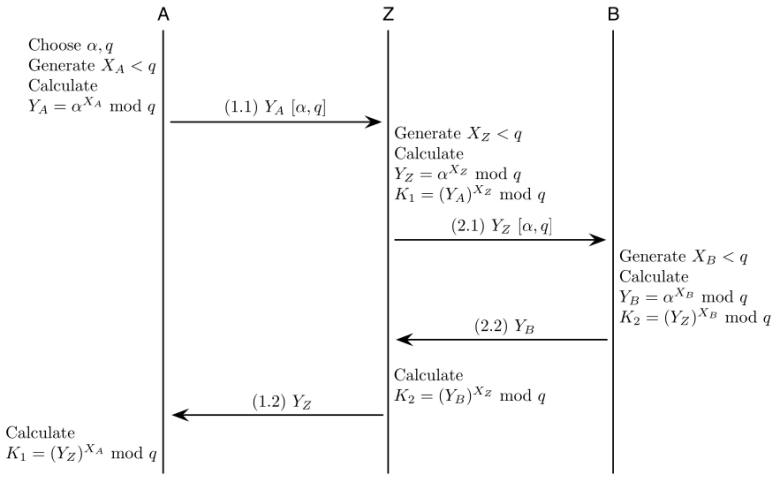
Livello di sicurezza La sicurezza si basa sul fatto che un ascoltatore (Eve) che intercetta i valori pubblici g, p, A, B non può facilmente calcolare la chiave segreta K perché dovrebbe risolvere il problema del logaritmo discreto per trovare a o b.

Attacco Man-in-the-Middle Il Diffie-Hellman nella sua forma base è suscettibile all'attacco Man-in-the-Middle. Eve può:

- intercettare A (il valore di Alice) e inviare il suo valore $E_A = g^{ea} \mod p$ a Bob;
- intercettare B (il valore di Bob) e inviare il suo valore $E_B = g^{eb} \mod p$ a Alice.

A questo punto, Eve ha una chiave segreta condivisa con Alice e una chiave segreta diversa condivisa con Bob, consentendole di decifrare, leggere, e ricifrare tutti i messaggi tra i due senza che loro se ne accorgano.

Per mitigare questo, si utilizzano versioni del protocollo con **autenticazione**, come il DHE (DH effimero) o l'ECDH (Elliptic Curve Diffie-Hellman) , spesso sono implementati all'interno di protocolli come TLS/SSL.



3.2.1 Perfect Forward Secrecy

Il Perfect Forward Secrecy (PFS), in crittografia, è una proprietà di un sistema di scambio di chiavi che garantisce che una chiave si sessione compromessa non comprometta anche le chiavi di sesisone precedenti e future. In termini più semplici, se un attaccante riesce a registrare la comunicazione cifrata oggi, e domani, ottiene la chiave a lungo termine (chiave privata del server), non sarà in grado di decifrare i dati registrati.

Funzionamento Per ottenere la PFS, un sistema deve utilizzare chiavi di sessione temporanee ed effimere che vengono generate in modo indipendente per ogni sessione di comunicazione.

- **Chiavi effimere:** invece di usare sempre la stessa chiave privata a lungo termine del server per cifrare la chiave di sessione, vengono usate chiavi di sessione uniche, spesso derivate tramite un algoritmo di scambio effimero.
- **Distruzione:** dopo la fine della sessione, le chiavi effimere utilizzate per quella sessione vengono immediatamente distrutte (non vengono memorizzate da nessuna parte).

Se un aggressore ruba la chiave privata permanente del server in un momento successivo (compromissione a posteriori), tale chiave gli permetterà di decifrare le future comunicazioni non protette da PFS, ma non gli darà alcuna informazione utile per decifrare le comunicazioni passate, poiché le chiavi temporanee usate in precedenza sono state eliminate.

Protocolli e algoritmi La PFS è implementata tipicamente nei protocolli di sicurezza come TLS (Transport Layer Security), la base per HTTPS. Gli algoritmi di scambio di chiavi che offrono PFS sono:

- **DHE (Diffie-Hellman Effimero):** utilizza chiavi Diffie-Hellman diverse e temporanee per ogni sessione.
- **ECDHE (Elliptic Curve Diffie-Hellman Effimero):** simile al DHE, ma utilizza la crittografia a curve ellittiche, che è più efficiente.

I moderni browser e server web preferiscono l'utilizzo di ECDHE per l'equilibrio tra sicurezza, velocità ed efficienza.

Mancanza di PFS Quando un protocollo non ha PFS, generalmente utilizza la chiave privata a lungo termine del server (o una master key statica) per proteggere direttamente o derivare la chiave di sessione. Un esempio di non-PFS è l'utilizzo di RSA statico per lo scambio di chiavi in TLS: se un aggressore ruba la chiave privata RSA del server, può derivare tutto il traffico registrato in precedenza che è stato scambiato con quel server.

Dettagli aggiuntivi su Forward secrecy

1. **Definizione formale e sinonimi:**
   - **Sinonimi:** PFS è spesso chiamato semplicemente Forward Secrecy (FS) o Public-Key Forward Secrecy (PFS) quando applicato a protocolli a chiave pubblica.
   - **Obbiettivo tecnico:** un sistema ha la proprietà di FS se l'ispezione (cioè la decifratura) in chiaro dello scambio di dati che avviene durante la fase di accordo chiave all'inizio della sessione non rileva la chiave utilizzata per cifrare il resto della sessione.
2. **Origine storica e protocolli:**
   - **Padri fondatori:** il concetto di Forward Secrecy fu introdotto formalmente da Whitfield Diffie, Paul van Oorschot e Micheal James Wiener nel 1992, descrivendolo come una proprietà del protocollo Station-to-Station.
   - **Adozione diffusa:** la PFS è una caratteristica cruciale adottata da numerosi protocolli e applicazioni:
     - è una caratteristica opzionale in IPsec;
     - è utilizzata in SSH (Secure Shell);
     - è un pilastro di protocolli di messaggistica moderna come OTR[1] (Off-the-Record Messaging) e il Protocollo Signal[2], che garantiscono l'indipendenza di ogni singolo messaggio.
   - **La mandatorietà di TLS 1.3:** Il protocollo TLS 1.3, la versione più recente e sicura del Transport Layer Security (che è la base di HTTPS), ha elevato la PFS da "opzionale" a obbligatoria.
     - L'IETF ha imposto che TLS 1.3 consenta solo le suite di cifrature effimere (come ECDHE).
     - Questo significa che lo scambio di chiavi RSA statico (che non offriva PFS è stato eliminato in TLS 1.3, rendendo la Perfect Forward Secrecy uno standard *de facto* per la navigazione web sicura moderna.)
   - **Il concetto di Backward Secrecy:** sebbene il PFS protegga le comunicazioni passate da una compromissione futura della chiave a lungo termine, esiste un concetto correlato:
     - **Backward Secrecy (o Future Secrecy):** è la proprietà che assicura che un'intercettazione e/o compromissione della chiave di sesisone attuale non permetta di decifrare le sessioni di comunicazione che avverranno in futuro.

[1] Protocollo crittografico che fornisce una cifratura alle conversazioni di messagistica istantanea utilizzando una combinazione di crittografia simmetrica AES con chiavi di 128 bit, scambio di chiavi Diffie-Hellman e funzione crittografica di hash SHA1. Oltre all'autenticazione e alla cifratura, l'OTR fornisce anche perfect forward secrecy.

[2] Protocollo critografico non federato che fornisce la crittografia end-to-end per i messaggi e le chiamate di messaggistica istantanea. Il protocollo è stato sviluppato da Open Whisper Systems nel 2013 ed è stato introdotto nell'app open source TextStructure, la quale in sequito è stata rinominata Signal. A partire dal 2018 è stato sviluppato dalla fondazione Signal ed è distribuito con licenza libeera AGPLv3.

– Alcuni protocolli avanzati, come il Protocollo Signal, implementano sia il Forward Secrecy che il Backward Secrecy per fornire una protezione completa contro la compromissione delle chiavi in qualsiasi momento.

3.3 Standard di Firma Digitale (DSS)

FIPS PUB 186 è conosciuto come Standard di Firma Digitale. Fa uso di SHA-1 e presenta una nuova tecnica di firma digitale, l'Algoritmo di Firma Digitale (DSA). Proposto originariamente nel 1991 e revisionato nel 1993 e nuovamente nel 1996. Utilizza un algoritmo progettato per fornire solo la funzione di firma digitale. A differenza di RSA, non può essere utilizzato per la crittografia o lo scambio di chiavi.

Evoluzione dello standard Le versioni successive come FIPS 186-3 (2009) e FIPS 186-4 (2013) hanno introdotto l'uso di SHA-2 (SHA-224, SHA-256, SHA-384 e SHA-512) al posto di SHA-1 per una maggiore sicurezza. Il più recente standard, FIPS 186-5 (2023), ha ritirato l'uso di DSA per le nuove implementazioni a causa della sua crescente complessità rispetto a RSA ed ECDSA, e ha introdotto l'uso potenziale di nuovi schemi di firma basati sulla crittografia post-quantistica (come gli schemi basati su reticoli).

3.3.1 Firme digitali

NIST FIPS PUB 186-4 (Standard di Firma Digitale DSS) definisce una firma digitale come: *"il risultato di una trasformazione crittografica dei dati che, se implementata correttamente, fornisce un meccanismo per verificare l'**autenticità dell'origine**, l'**integrità dei dati** e la **non ripudiabilità del firmatario**"*. Pertanto, una firma digitale è un modello di bit dipendente dai dati, generato da un agente come funzione di un file, messaggio o altra forma di blocco di dati.

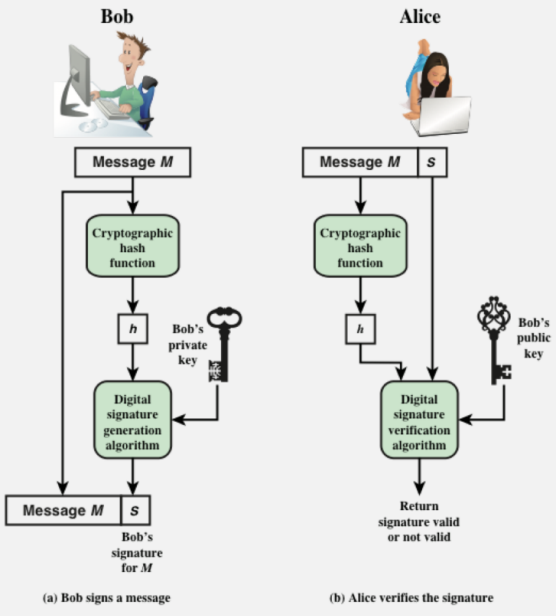
FIPS 186-4 specifica l'uso di uno dei tre algoritmi di firma digitale:

- algoritmo di firma digitale (DSA);
- algoritmo di firma digitale RSA;
- algoritmo di firma digitale a curva ellittica (ECDSA).

Processo di firma Il processo generale di firma digitale coinvolge due fasi principali.

1. **Fase di firma:** il documento (o messaggio) viene prima sottoposto a hashing utilizzando una funzione crittografica come SHA-2. L'hash risultante (noto come *message digest*) viene quindi crittografato con la chiave privata del firmatario (utilizzando uno degli algoritmi specificati: DSA, RSA o ECDSA). La firma digitale è l'hash crittografato.
2. **Fase di verifica:** chiunque può verificare la firma decrittografando l'hash con la chiave pubblica del firmatario. L'hash decrittografato viene poi confrontato con un nuovo hash generato dal documento ricevuto. Se i due hash corrispondono, la firma è considerata valida, provando l'autenticità e l'integrità.

Crittografia a curva ellittica (ECC) La tecnica si basa sull'uso di una costruzione matematica nota come **curva ellittica** (specificatamente sul problema del logaritmo discreto su curve ellittiche, ECDLP). L'attrazione principale dell'ECC rispetto a RSA è che sembra offrire una sicurezza equivalente per una dimensione di bit molto più piccola, riducendo così il sovraccarico



di elaborazione. Ad esempio, una chiave ECC a 256 bit è considerata equivalente in sicurezza a una chiave RSA a 3072 bit. Il livello di fiducia nell'ECC non è ancora alto come quello in RSA (le chiavi di *backdoor* sono state diffuse in alcuni momenti, come con l'algoritmo Dual EC DRBG).

Algorimto ECDSA l'Algoritmo di Firma Digitale a Curva Ellittica (ECDSA) è la variante di DSA che utilizza l'ECC. È oggi ampiamente preferito in molti protocolli moderni (come TLS/SSL e Bitcoin) grazie alle dimensioni ridotte della chiave e delle firme, che lo rendono ideale per ambienti con larghezza di banda limitata o per dispositivi mobili.

4 Crittografia a chiave pubblica e autenticazione dei messaggi

La crittografia protegge contro attacchi passivi (intercettazione). Un requisito diverso è proteggere contro attacchi attivi (falsificazione di dati e transazioni). La protezione contro tali attacchi è nota come **autenticazione dei messaggi**.

4.1 Autenticazione dei messaggi

È una procedura che consente alle parti comunicanti di verificare che i messaggi ricevuti siano autentici. I due aspetti importanti sono:

1. verificare che il contenuto del messaggio non sia stato alterato;
2. verificare che la fonte sia autentica;

Altri requisiti sono la tempestività, l'assenza di ripetizioni o il riordino.

Approcci all'autenticazione dei messaggi

- **Utilizzando la crittografia convenzionale:** la crittografia simmetrica da sola non è uno strumento adatto per l'autenticazione dei dati. Si assume che solo che il mittente e il destinatario condividano una chiave, quindi solo il mittente genuino sarebbe in grado di crittografare un messaggio con successo. Il destinatario presume che non siano state apportate alterazioni e che la sequenza sia corretta se il messaggio include un codice di rilevamento degli errori e un numero di sequenza. Se il messaggio include un timestamp, il destinatario presume che il messaggio non sia stato ritardato oltre il tempo normalmente previsto per il transito in rete.
- **Senza crittografia del messaggio:** viene generato un tag di autenticazione e aggiunto a ciascun messaggio per la trasmissione. Il messaggio stesso non è crittografato e può essere letto a destinazione indipendentemente dalla funzione di autenticazione a destinazione. Poiché il messaggio non è crittografato, la riservatezza del messaggio non è garantita.

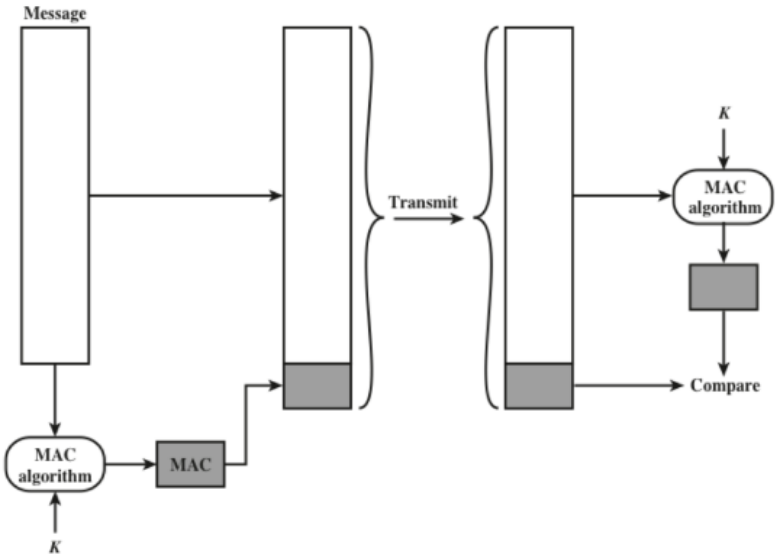
4.1.1 MAC (Message Authentication Code)

Il MAC, in crittografia, è un piccolo blocco di dati utilizzato per garantire l'autenticazione, e l'integrità di un messaggio digitale, ma non la sua confidenzialità (riservatezza). Viene generato attraverso un meccanismo di crittografia simmetrica che richiede una chiave segreta condivisa tra mittente e destinatario.

Come funziona il MAC

1. **Generazione del MAC (mittente):** il mittente prende il messaggio da inviare e lo elabora con un algoritmo MAC (ad esempio, HMAC, CBC-MAC e una chiave segreta K. L'algoritmo produce in output un valore chiamato MAC (o tag). Il mittente invia al destinatario il messaggio in chiaro e il MAC associato.
2. **Verifica del MAC (destinatario):** il destinatario riceve il messaggio e il MAC. Utilizzando lo stesso algoritmo MAC e la stessa chiave segreta K ricalcola il MAC sul messaggio ricevuto. Confronta il MAC ricalcolato con il MAC allegato al messaggio dal mittente.
3. **Esito della verifica:**
    - Se i due MAC coincidono, il destinatario ha la certezza che il messaggio non è stato modificato durante la trasmissione (**integrità**) e che proviene effettivamente da chi non possiede la chiave segreta (quindi, il mittente legittimo).
    - Se i due MAC non coincidono, significa che il messaggio è stato alterato o che non proviene dal mittente legittimo.

MAC e riservatezza È fondamentale notare che il MAC da solo non cripta il contenuto del messaggio; protegge solo da modifiche e falsificazioni. Per garantire anche la riservatezza delle informazioni (cioè per impedire che terzi leggano il messaggio), il MAC viene solitamente utilizzato in combinazione con un algoritmo di crittografia simmetrica (ad esempio, AES) che provvede a cifrare l'intero messaggio.



Caratteristica	MAC (Message Authentication Code)	Funzione di Hash (message digest)	Firma digitale
Obbiettivo principale	Autenticazione e integrità	Integrità (senza chiave)	Integrità, autenticazione e non ripudio
Uso della chiave	Sì (chiave segreta simmetrica)	No	Sì (chiave privata asimmetrica)
Non ripudio	No (chi verifica può anche generare)	No	Sì (solo il proprietario della chiave privata può firmare)
Tipo di crittografia	Simmetrica	Nessuna (solo funzione matematica)	Asimmetrica

4.1.2 Funzioni Hash unidirezionali

Le funzioni hash unidirezionali (o one-way hash functions), sono algoritmi fondamentali in crittografia e sicurezza informatica. Sono funzioni matematiche che prendono un input di lunghezza arbitraria (il messaggio, un file, una password, ecc…) e producono in output una stringa di caratteri di lunghezza fissa chiamata **valore hash** o **digest**. Il concetto di "unidirezionale" è la proprietà di sicurezza cruciale.

Una funzione è considerata unidirezionale se è:

1. **Facile da calcolare:** dato un input x, è estremamente rapido e semplice calcolare l'output $h(x)$.
2. **Impossibile da invertire:** dato l'output $h(x)$, è computazionalmente impossibile risalire all'input originale x in un tempo ragionevole (richiederebbe un attacco a forza bruta con tempi superiori alla vita dell'universo).

Questa proprietà garantisce che, anche se un attaccante intercetta o ottiene il valore hash, non può ricostruire il messaggio o la password originali.

Requisiti di sicurezza (funzioni hash sicure) Per essere considerate sicure e utilizzabili in crittografia, le funzioni hash devono soddifare tre proprietà principali:

1. **Resistenza alla preimmagine (one-way):** dato un valore hash h, è difficile trovare un messaggio m tale che $h(m) = h$ (impossibilità di invertire la funzione).
2. **Resistenza alla seconda preimmagine(weak collision resistance):** dato un messaggio originale m_1, è difficile trovare un secondo messaggio diverso m_2 tale che $h(m_1) = h(m_2)$ (impossibilità di alterare un messaggio lasciando inalterato l'hash).
3. **Resistenza alle collisioni (strong collision resistance):** è difficile trovare una coppia di messaggi qualsiasi (m_1, m_2) tali che $h(m_1) = h(m_2)$ (questa è la proprietà più restrittiva e cruciale).

Effetto valanga Un'altra caratteristica importante è l'**effetto valanga** (*"avalanche effect"*): anche una piccola modifica all'input (ad esempio, cambiare un singolo bit nel messaggio) deve causare un drastico e imprevedibile cambiamento nell'output hash.

Applicazioni principali Le funzioni hash unidirezionali sono il pilastro di molti meccanismi di sicurezza:

- **Verifica dell'integrità dei dati:** un hash (spesso chiamato *checksum*) viene calcolato su un file prima della trasmissione/memorizzazione ricalcolato dopo. Se i due hash coincidono, il file è integro.
- **Memorizzazione sicura delle password:** i sistemi non memorizzano mai le password degli utenti in chiaro. Memorizzano invece l'hash della password (spesso chiamato "salato" con un valore casuale, *"salt"*). Qaundo un utente accede, la password viene passata alla funzione di hash e confrontata con l'hash memorizzato. Poiché la funzione è unidirezionale, anche se il database viene violato, gli aggressori non ottengono le password reali.
- **Firme digitali:** invece di firmare digitalmente l'intero documento (che è inefficiente), l'algoritmo di firma digitale si applica solo all'hash (o *digest*) del documento.

- **Blockchain e criptovalute:** vengono utilizzate intensamente per concatenare i blocchi, per l'identificazione delle transazioni e nel processo di *mining*(Proof-of-Work).

Esempi noti di hash crittograficamente sicuri includono la famiglia SHA-2 (come SHA-256) e SHA-3. Algoritmi più vecchi come MD5 e SHA-1 sono stati considerati in gran parte abbandonati per la resistenza alle collisioni[3].

Approcci per attaccare una funzione hash sicura Può avvenire tramite crittoanalisi (trovando debolezze logiche nell'algoritmo di hash) oppure con attacco di forza bruta (la resistenza di una funzione hash contro questo attacco dipende esclusivamente dalla lunghezza del codice hash prodotto dall'algoritmo).

SHA-1,SHA-2,SHA-3 SHA (Secure Hash Standard) è stata sviluppata dal NIST ed è stata pubblicata come standard federale di elaborazione delle informazioni (FIPS 180) nel 1993. È stata revisionata nel 1995 come SHA-1 e pubblicata come FIPS 180-1. Si basa sulla funzione hash MD4 e il suo design modella strettamente MD4. Produce valori hash di 160 bit. Nel 2005, il NIST ha annunciato l'intenzione di eliminare l'approvazione di SHA-1 e di passare a una dipendenza da SHA-2 (256,384,512 bit) entro il 2010. (Nel 2015, il NIST ha rilasciato una nuova versione, SHA-3).

Funzionamento di SHA2

1. **Imbottitura:** al messaggio originale vengono aggiunti dei bit di "imbottitura" affinché la lunghezza finale del messaggio risulti congruenti a 448 modulo 512, così facendo la lunghezza in bit di "messaggio + imbottitura" divisa per 512 darà resto 448.

2. **Aggiunta lunghezza:** alla sequenza di bit (messaggio+imbottitura) viene aggiunto un intero unsigned di 64 bit contenente la lunghezza del messaggio originale. Alla fine di questi due primi passi otteniamo una sequenza di bit che è un multiplo di 512.

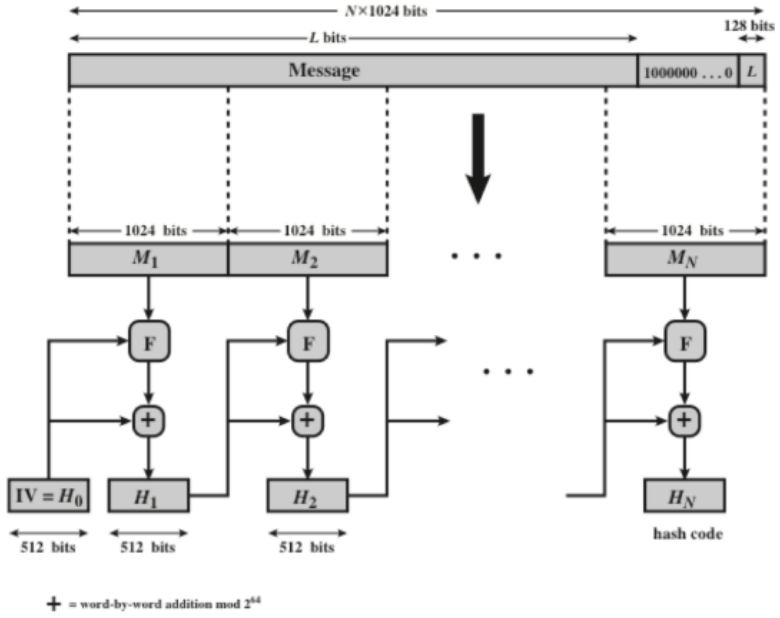
3. **Inizializzazione del buffer MD:** un buffer di 160 bit suddiviso in 5 registri da 32 bit ciascuno viene creato per la memorizzazione di alcuni passaggi intermedi. I 5 registri verranno convenzionalmente indicati con (A,B,C,D,E) ed inizializzati con i seguenti valori esadecimali:

   (a) A=67452301
   (b) B=EFCDAB89
   (c) C=98BADCFE
   (d) D=10325476
   (e) E=C3D2E1F0

4. **Elaborazione dei blocchi da 512 bit:** la sequenza di bit che comprende il messaggio, l'imbottitura e la lunghezza del messaggio, viene divisa in blocchi da 512 bit, che identificheremo con B_n con $0 \leq n \leq L$. Il fulcro dell'algoritmo SHA-1 è chiamato *compression function* ed è formato da 4 cicli di 20 passi cadauno. I cicli hanno una struttura molto

[3]In pratica, è diventato fattibile per un attaccante trovare due input diversi che producono lo stesso identico valore di hash. Quando ciò accade, l'algoritmo non è più considerato sicuro per la maggior parte delle applicazioni crittografiche poiché un attaccante potrebbe creare un file dannoso con lo stesso hash di un file legittimo, falsificando l'autenticità.

simile tra di loro se non per il fatto che utilizzano una differente funzione logica primitiva. Ogni blocco viene preso come parametro di input da tutti e 4 i cicli insieme ad una costante K e i valori dei 5 registri. Alla fine della computazione otterremo dei nuovi valori per A,B,C,D,E, che useremo per la computazione del blocco successivo sino ad arrivare al blocco finale F.



Nel 2001 il NIST pubblicò quattro funzioni di *hash* addizionali facenti parti della famiglia SHA, ognuna con un digest più lungo di quello originale, collettivamente denominate SHA-2. Queste varianti sono note con la lunghezza in bit del digest generato a seguire la sigla ufficiale dell'hash. Gli algoritmi SHA-256 e SHA-512 lavorano rispettivamente con word di 32 e 64 bit: utilizzano un numero differente di rotazioni e di costanti addizionali, ma la loro struttura è sostanzialmente identica.

SHA3 deve poter sostituire SHA-2 e deve preservarne la sua natura.

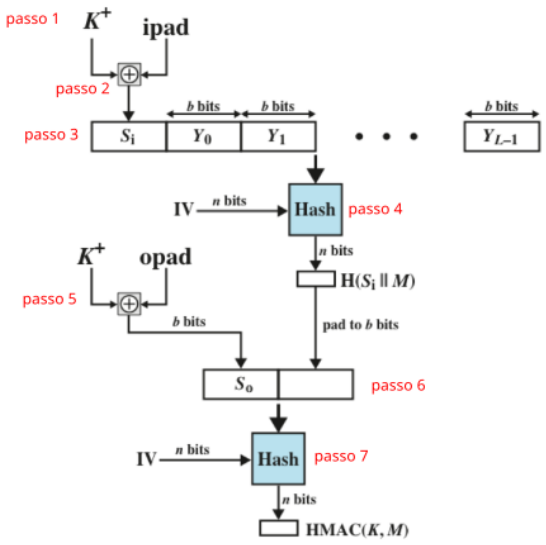
4.1.3 HMAC (Hash-based Message Authentication Code)

L'HMAC è uno schema specifico per creare un MAC utilizzando una funzione hash crittografica (come SHA-256) in combinazione con una chiave segreta.

Obbiettivo HMAC è stato progettato per risolvere i problemi di sicurezza che sorgono quando si tenta semplicemente di concatenare la chiave segreta al messaggio e poi calcolare l'hash (es. hash(chiave + messaggio)). Tali metodi semplici si sono dimostrati vulnerabili a vari attacchi. HMAC è più robusto e utilizza la chiave segreta in due passaggi separati per mescolarla in modo efficace con il messaggio e prevenire attacchi noti.

Come funziona La formula generale di HMAC è:

$$\text{HMAC}_K(M) = \text{Hash}((K \oplus opad)||\text{Hash}(K \oplus ipad)||M)$$



$$\text{HMAC}(K, M) = \text{H}[\ (K+ \oplus \text{opad}) \ \ || \ \text{H}[(K+ \oplus \text{ipad}) \ \ || \ M]\ ]$$

- K+ = K padded with zeros on the right so that the result is b bits long
- ipad = 00110110 (36 in hexadecimal) repeated b/8 times
- opad = 01011100 (5C in hexadecimal) repeated b/8 times

dove K è la chiave segreta, M è il messaggio, Hash è la funzione hash sottostante (es. SHA-256), *ipad* e *opad* sono costanti fisse definite nello standard, $\oplus$ rappresenta l'operazione di XOR e $||$ rappresenta la concatenazione.

Algoritmo

1. Aggiungere zeri all'estremità sinistra di K in modo tale che il risultato K^+ sia lungo b bit (ad esempio, se K ha una lunghezza di 160 bit e $b = 512$, allora K sarà completato con 44 zeri ovvero $\frac{512-160}{8} = \underbrace{\frac{352}{8}}_{8 \text{ bit} = 1 \text{ byte}} = 44$ byte.
2. Eseguire l'operazione di XOR (bitwise exclusive-OR) tra K^+ e *ipad* per produrre il blocco di b bit S_i.
3. Aggiungere M a S_i.
4. Applicare H al flusso generato nel passo 3.
5. Eseguire l'operazione di XOR tra K^+ e *opad* per produrre il blocco di b bit S_0.
6. Aggiungere il risultato dell'hash ottenuto dal passo 4 a S_0.
7. Applicare H al flusso generato nel passo 6 e restituire il risultato.

Nota che l'operazione XOR con *ipad* comporta il ribaltamento della metà dei bit di K. Allo stesso modo, l'operazione XOR con *opad* comporta il ribaltamento della metà dei bit di K, utilizzando

un insieme diverso di bit. In effetti, passando S_i e S_0 attraverso la funzione di compressione dell'algoritmo di hash, abbiamo generato pseudocasualmente due chiavi da K.

Vantaggi principali

- **Integrità e autenticazione:** come tutti i MAC, l'HMAC garantisce che il messaggio non sia stato alterato e che provenga da un mittente che possiede la chiave segreta condivisa.
- **Sicurezza migliorata:** la sua struttura a doppio hash (inner e outer) garantisce che, anche se vengono scoperte debolezze nell'algoritmo di hash sottostante (ad esempio, se si trova una collisione), l'HMAC rimanga difficile da falsificare.
- **Flessibilità:** l'HMAC può utilizzare qualsiasi funzione hash crittografica come base (si parla infatti di HMAC-MD5, HMAC-SHA1, HMAC-SHA256, ecc…). Attualmente, HMAC-SHA256 è lo standard più raccomandato.

HMAC è ampiamente utilizzato in protocolli come IPsec, TLS (per la protezione dei dati), e nella generazione di token di sicurezza come i JSON Web Tokens (JWT).

4.1.4 CMAC (Cipher-based Message Authentication Code)

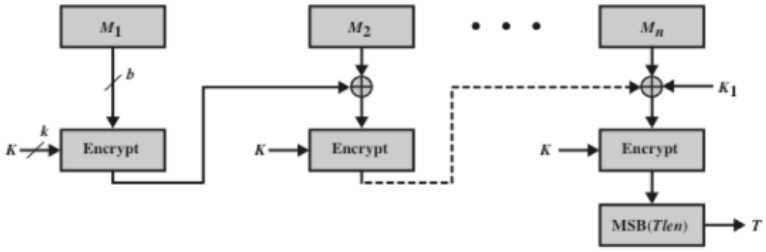
Il CMAC è uno schema specifico per generare un Message Authentication Code (MAC) utilizzando un cifrario a blocchi simmetrico (come AES) in modalità CBC (Cipher Block Chaining). È stato sviluppato per superare le debolezze di sicurezza e le complessità legali (legate ai brevetti) del suo predecessore, il CBC-MAC standard.

Caratteristiche fondamentali

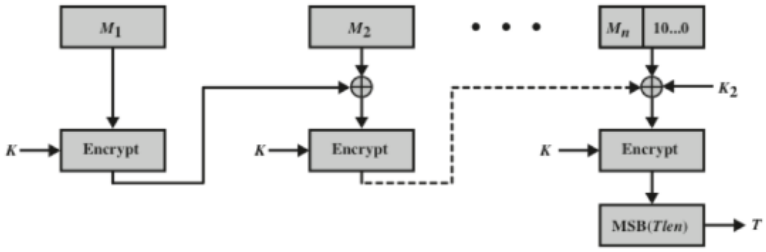
- **Algoritmo sottostante:** il CMAC utilizza un cifrario a blocchi simmetrico. L'implementazione più comune e raccomandata è AES-CMAC (che usa AES come cifrario).
- **Scopo:** garantisce sia l'integrità dei dati che l'autenticazione del messaggio grazie all'uso di una chiave segreta condivisa.
- **Standardizzazione:** è stato formalmente specificato dal NIST nella pubblicazione NIST SP 800-38B e da IETF nella RFC 4493, il che ne assicura la robustezza e l'affidabilità.

Come funziona A differenza dell'HMAC che usa funzioni hash, il CMAC usa un cifrario a blocchi:

- **Crittografia a catena (CBC):** il CMAC elabora il messaggio in blocchi, proprio come la modalità CBC. Ogni blocco viene crittografato e il suo output viene combinato in XOR con il blocco di dati successivo prima che quest'ultimo venga crittografato.
- **Chiavi derivate:** il CMAC si distingue dal CBC_MAC standard per il modo in cui gestisce l'ultimo blocco. Per evitare vulnerabilità di sicurezza, il CMAC utilizza due chiavi aggiuntive K_1 e K_2, che vengono derivate dalla chiave principale del cifrario K utilizzando operazioni matematiche.
- **L'ultimo blocco:**
  - se il messaggio è diviso esattamente in blocchi completi, l'ultimo blocco viene elaborato con la chiave derivata K_1;



(a) Message length is integer multiple of block size



(b) Message length is not integer multiple of block size

  - se il messaggio è incompleto (non riempe l'ultimo blocco), viene aggiunto un padding standard e il blocco viene elaborato con la chiave derivata K_2.

- **Generazione del MAC:** l'output finale della crittografia dell'ultimo blocco, dopo essere stata combinata con la chiave derivata appropriata, è il valore CMAC (il tag di autenticazione).

Vantaggi

- **Efficienza:** poiché utilizza gli stessi algoritmi di crittografia a blocchi già presenti nell'hardware per la riservatezza, spesso è molto efficiente in termini di implementazione e velocità di esecuzione.

- **Robustezza:** la corretta gestione dell'ultimo blocco (con l'uso di K_1 e K_2) garantisce che il CMAC sia matematicamente dimostrato sicuro sotto l'ipotesi che il cifrario a blocchi sia un buona *funzione pseudocasuale*.

CMAC è ampiamente utilizzato negli standard di sicurezza e nei dispositivi hardware (come i moduli di sicurezza e hardware o HSM) dove l'efficienza è cruciale.

4.2 Applicazioni per i sistemi di crittografia a chiave pubblica

I sistemi a chiave pubblica sono caratterizzati dall'uso di un algoritmo crittografico con due chiavi: una mantenuta privata e una disponibile pubblicamente. A seconda dell'applicazione, il mittente utilizza la chiave privata del mittente, o la chiave pubblica del destinatario, o entrambe per eseguire un certo tipo di funzione crittografica:

- **crittografia/decrittografia:** il mittente crittografa un messaggio con la chiave pubblica del destinatario;

- **firma digitale:** il mittente "firma" un messaggio con la propria chiave privata;
- **scambio di chiavi:** due parti collaborano per scambiare una chiave di sessione.

Algoritmo	Crittografia/decrittografia	Firma digitale	Scambio di chiavi
RSA	Sì	Sì	Sì
Diffie-Hellman	No	No	Sì
DSS	No	Sì	No
Curva ellittica	Sì	Sì	Sì

5 Gestione e distribuzione delle chiavi crittografiche

Gestione delle chiavi crittografiche L'uso sicuro degli algoritmi di chiave crittografica dipende dalla protezione delle chiavi crittografiche; la loro gestione è il processo di amministrazione o gestione delle chiavi crittografiche per un sistema crittografico: comporta la generazione, la creazione, la protezione, l'archiviazione, lo scambio, la sostituzione e l'utilizzo delle chiavi e consente restrizioni selettive per determinate chiavi. Oltre alla restrizione di accesso, la gestione delle chiavi implica anche il monitoraggio e la registrazione dell'accesso, dell'uso e del contesto di ciascuna chiave. Un sistema di gestione delle chiavi includerà anche server di chiavi, procedure per gli utenti e protocolli. La sicurezza del sistema crittografico dipende dalla gestione efficace delle chiavi.

Tecnica di distribuzione delle chiavi Affinché la crittografia simmetrica funzioni, le due parti coinvolte in uno scambio devono condividere la stessa chiave che deve essere protetta dall'accesso di terzi. Inoltre, cambi frequenti della chiave sono solitamente auspicabili per limitare la quantità di dati compromessi se un attaccante riesce a scoprire la chiave. Pertanto, la forza di qualsiasi sistema crittografico dipende dalla tecnica di distribuzione delle chiavi, un termine che si riferisce ai mezzi per consegnare una chiave a due parti che desiderano scambiare dati, senza consentire ad altri di vedere la chiave.

Distribuzione della chiave simmetrica Per due parti A e B, la distribuzione della chiave può essere realizzata in diversi modi:

1. A può selezionare una chiave e consegnarla fisicamente a B;
2. una terza parte può selezionare la chiave e consegnarla fisicamente sia ad A che a B;
3. se A e B hanno precedentemente e recentemente utilizzato una chiave, una delle parti può trasmettere la nuova chiave all'altra, cifrata utilizzando la vecchia chiave;
4. se A e B hanno ciascuno una connessione crittografata con una terza parte C, C può consegnare una chiave sui collegamenti crittografati a A e B.

Le opzioni 1 e 2 richiedono la consegna manuale di una chiave. Per la crittografia dei collegamenti, questo è un requisito ragionevole, poiché ogni dispositivo di crittografia dei collegamenti scambierà dati solo con il suo partner all'altro capo del collegamento. Tuttavia, per la crittografia end-to-end su una rete, la consegna manuale è scomoda. In un sistema distribuito, qualsiasi

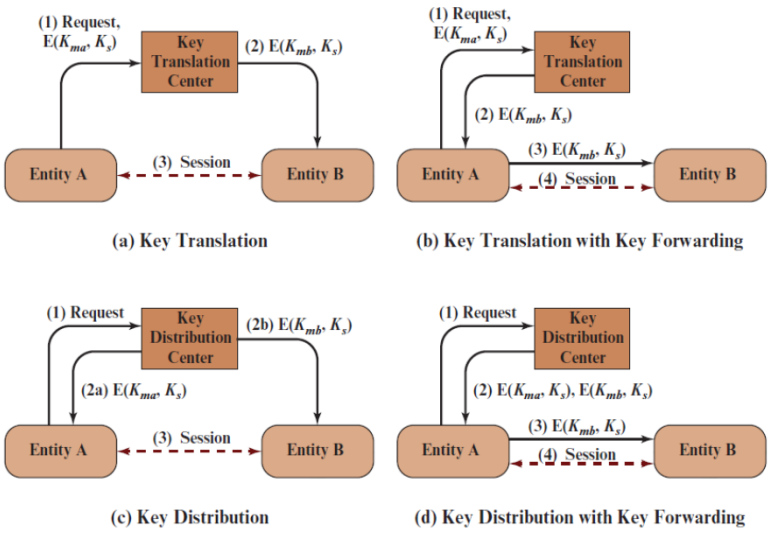


Figura 2: opzioni per la distribuzione delle chiavi.

utente o server può dover partecipare a scambi con molti altri utenti e server nel tempo. Pertanto, ogni endpoint ha bisogno di un certo numero di chiavi fornite dinamicamente. Il problema è particolarmente difficile in un sistema distribuito su una vasta area.

La scala del problema dipende dal numero di coppie comunicanti che devono essere supportate. Se la crittografia end-to-end viene eseguita a livello di rete o IP, allora è necessaria una chiave per ogni coppia di host sulla rete che desidera comunicare. Pertanto, se ci sono n host, il numero di chiavi richieste è:

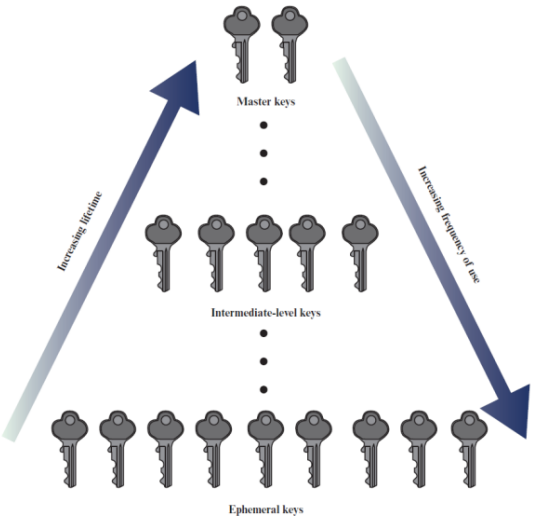
$$\frac{n(n-1)}{2}$$

Se la crittografia viene eseguita a livello di applicazione, allora è necessaria una chiave per ogni coppia di utenti o processi che richiedono comunicazione. Pertanto, una rete può avere centinaia di host ma migliaia di utenti e processi. Una rete che utilizza la crittografia a livello di nodo con 1000 nodi potrebbe concepibilmente dover distribuire fino a mezzo milione di chiavi. Se quella stessa rete supporta 10.000 applicazioni, allora potrebbero essere necessarie fino a 50 milioni di chiavi per la crittografia a livello di applicazione.

L'opzione 3 è una possibilità sia per la crittografia dei collegamenti che per la crittografia end-to-end, ma se un attaccante riesce mai a ottenere l'accesso a una chiave, tutte le chiavi saranno rilevate. Inoltre, la distribuzione iniziale di potenzialmente milioni di chiavi deve ancora essere effettuata.

Per la crittografia end-to-end, una qualche variazione dell'opzione 4 è stata ampiamente adottata. In questo schema, un centro di distribuzione delle chiavi è responsabile della distribuzione delle chiavi a coppie di utenti (host, processi, applicazioni) secondo necessità. Ogni utente deve condividere una chiave unica con il centro di distribuzione delle chiavi per scopi di distribuzione delle chiavi.

La fig. 2 illustra due diverse opzioni, ciascuna con due variazioni per la distribuzione delle chiavi. I numeri lungo le linee rappresentano i passaggi dello scambio. In questi esempi, esiste una connessione tra le entità A e B, che desiderano scambiare informazioni utilizzando tecniche crittografiche. A questo scopo, necessitano di una chiave di sessione temporanea che durerà per la durata di una connessione logica, come una connessione TCP. A e B condividono ciascuno



una chiave master di lunga durata con una terza parte coinvolta nella fornitura della chiave di sessione. La chiave di sessione è etichettata come K_S e le chiavi master tra le entità A e B e la terza parte sono etichettate come K_{ma} e K_{mb} rispettivamente. Un centro di traduzione chiavi (KTC) trasferisce chiavi simmetriche per future comunicazioni tra due entità, almeno una delle quali ha la capacità di generare o acquisire chiavi simmetriche autonomamente. L'entità A genera o acquisisce una chiave simmetrica da utilizzare come chiave di sessione per la comunicazione con B. A cripta la chiave utilizzando la chiave master che condivide con il KTC e invia la chiave criptata al KTC. Il KTC decripta la chiave di sessione, la cripta nuovamente con la chiave master che condivide con B e invia quella chiave di sessione criptata nuovamente ad A affinché A la inoltri a B, oppure la invia direttamente a B.

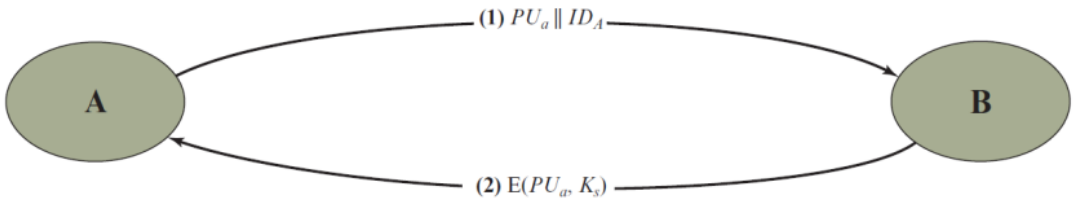
KDC (Key Distribution Center) Un centro di distribuzione chiavi (KDC) genera e distribuisce chiavi di sessione. L'entità A invia una richiesta al KDC per una chiave simmetrica da utilizzare come chiave di sessione per la comunicazione con B. Il KDC genera una chiave di sessione simmetrica, quindi la cripta con la chiave master che condivide con B e la invia a B. In alternativa, invia entrambi i valori di chiave criptati ad A, e A inoltra la chiave di sessione criptata con la chiave master condivisa dal KDC e B a B.

Vi sono un certo numero di dettagli omessi, ad esempio, le parti che scambiano chiavi devono autenticarsi reciprocamente. I timestamp sono spesso utilizzati per limitare il tempo in cui può avvenire uno scambio di chiavi e/o la durata di vita di una chiave scambiata.

Gerarchia delle chiavi Un requisito comune in una varietà di protocolli come IEEE 802.11 e IPsec è la crittografia di una chiave simmetrica in modo che possa essere distribuita a due parti per future comunicazioni. Spesso, un protocollo richiede una gerarchia di chiavi con chiavi più basse nella gerarchia utilizzate più frequentemente e cambiate più spesso per contrastare gli attacchi. Una chiave di livello superiore, che viene utilizzata raramente, e quindi è più resistente alla crittoanalisi, viene utilizzata per crittografare una chiave di livello inferiore appena creata in modo che possa essere scambiata tra le parti che condividono la chiave di livello superiore. Il termine chiave effimera si riferisce a una chiave che viene utilizzata solo una volta o al massimo ha una vita molto breve.

5.1 Distribuzione delle chiavi simmetriche

Utilizzo della crittografia asimmetrica A causa dell'inefficienza dei sistemi crittografici a chiave pubblica, essi sono quasi mai utilizzati per la crittografia diretta di blocchi di grandi dimensioni, ma sono limitati a blocchi relativamente piccoli. Uno degli usi più importanti di un sistema crittografico a chiave pubblica è quello di crittografare chiavi segrete per la distribuzione.



Distribuzione semplice delle chiavi segrete Uno schema estremamente semplice è stato proposto da Merkle. Se A desidera comunicare con B, viene impegata la seguente procedura:

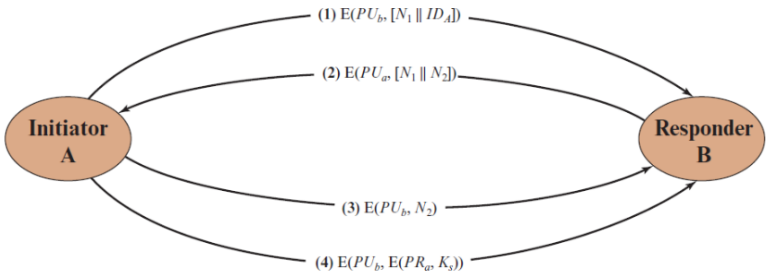
1. A genera una coppia di chiavi pubblica/privata $\{PU_A, PR_A\}$ e trasmette a B un messaggio contenente PU_A e un indentificatore di A, ID_A;
2. B genera una chiave segreta K_S, e la trasmette ad A, che è criptata con la chiave pubblica di A;
3. A calcola $D(PR_A, E(PU_A, K_S))$ per recuperare la chiave segreta. Poiché solo A può decriptare il messaggio, solo A e B conosceranno l'identità di K_S;
4. A scarta PU_A e PR_A e B scarta PU_B.

A e B possono ora comunicare in modo sicuro utilizzando la crittografia convenzionale e la chiave di sessione K_S. Al termine dello scambio, sia A che B scartano K_S. Nonostante la sua semplicità, questo è un protocollo attraente. Non esistono chiavi prima dell'inizio della comunicazione e nessuna esiste dopo il completamento della comunicazione . Pertanto il rischio di compromissione delle chiavi è minimo. Allo stesso tempo, la comunicazione è sicura da intercettazioni. Ma il protocollo risulta essere insicuro contro un avversario che può intercettare i messaggi e può sia inoltrare il messaggio intercettato, sia sostituirlo con un altro messaggio. Tale attacco è noto come attacco **man-in-the-middle**[4]. Se un avversario D ha il controllo del canale di comunicazione intermedio, allora D può compromettere la comunicazione senza essere rivelato:

1. A genera una coppia di chiavi pubblica/privata $\{PR_A, PR_B\}$ e trasmette un messaggio destinato a B contenente PU_A e un identificatore di A, ID_A;
2. D intercetta il messaggio, crea la propria coppia di chiavi pubblica/privata $\{PU_D, PR_D\}$ e trasmette $PU_D||ID_A$ a B;
3. B genera una chiave segreta K_S,e trasmette $E(PU_D, K_S)$;
4. D intercetta il messaggio e appende K_S calcolando $D(PR_D, E(PU_D, K_S))$;
5. D trasmette $E(PU_A, K_S)$ ad A.

[4]Attacco informatico in cui qualcuno segretamente ritrasmette o altera la comunicazione tra due parti che credono di comunicare direttamente tra di loro.

Il risultato è che sia A che B conoscono K_S e non sono a conoscenza del fatto che K_S è stato rivelato anche a D. A e B possono ora scambiarsi messaggi utilizzando K_S. D non interferisce più attivamente con il canale di comunicazione, ma semplicemente ascolta. Conoscendo K_S, D può decriptare tutti i messaggi, e sia A che B non sono a conoscenza del problema. Pertanto, questo semplice protocollo è utile solo in un ambiente in cui l'unica minaccia è l'intercettazione.



Needam-Schroeder Public Key Protocol: also this protocol is vulnerable to a MITM attack.

Distribuzione della chiave segreta con riservatezza e autenticazione

1. A utilizza la chiave pubblica di B per crittografare un messaggio a B contenente un identificatore di A ID_A e un nonce N_1 che viene utilizzato per identificare univocamente questa transazione;

2. B invia un messaggio ad A crittografato con PU_A e contenente il nonce di A N_1 così come un nuovo nonce generato da B N_2. Poiché solo B potrebbe aver decriptato il messaggio (1), la presenza di N_1 nel messaggio (2) assicura ad A che il corrispondente è B;

3. A restituisce N_2, crittografato utilizzando la chiave pubblica di B, per assicurare a B che il suo corrispondente è A;

4. A seleziona una chiave segreta K_S e invia $M = E(PU_B, E(PR_A, K_S))$ a B. La crittografia di questo messaggio con la chiave pubblica di B assicura che solo B possa leggerlo; la crittografia con la chiave privata di A assicura che solo A possa averlo inviato (firma di A);

5. B calcola $D(PU_A, D(PR_B, M))$ per recuperare la chiave segreta. Ricorda che si applica dapprima $D(PR_B, M) = D(PR_B, E(PU_B, E(PR_A, K_S)))$ per ottenere $E(PR_A, K_S)$ e successivamente $D(PU_A, E(PR_A, K_S))$ per ottenere K_S.

Il risultato è che questo schema garantisce sia la riservatezza che l'autenticazione nello scambio di una chiave segreta.

Distribuzione delle chiavi pubbliche

1. **Annuncio pubblico delle chiavi pubbliche:** Se esiste un algoritmo di chiave pubblica ampiamente accettato, come RSA, qualsiasi partecipante può inviare la propria chiave pubblica a qualsiasi altro partecipante o trasmettere la chiave alla comunità in generale. Sebbene questo approccio sia conveniente, presenta una grande debolezza. Chiunque

può falsificare un tale annuncio pubblico. Cioè, un utente potrebbe fingersi l'utente A (**masquerading attack** e più specificatamente **spoofing attack**[5]) e inviare una chiave pubblica a un altro partecipante o trasmettere tale chiave pubblica. Fino a quando l'utente A non scopre la falsificazione e avvisa gli altri partecipanti, il falsario è in grado di leggere tutti i messaggi crittografati destinati ad A e può utilizzare le chiavi falsificate per l'autenticazione.

2. **Directory pubblicamente disponibile:** un maggiore grado di sicurezza può essere raggiunto mantenendo una directory dinamica di chiavi pubbliche, pubblicamente disponibile. La manutenzione e la distribuzione della directory pubblica dovrebbe essere responsabilità di un'entità o organizzazione fidata. Tale schema includerebbe i seguenti elementi:

   (a) l'autorità mantiene una directory con un'entrata nome, chiave pubblica per ciascun partecipante;

   (b) ogni partecipante registra una chiave pubblica presso l'autorità della directory. La registrazione dovrebbe avvenire di persona o tramite qualche forma di comunicazione sicura e autenticata;

   (c) un partecipante può sostituire la chiave esistente con una nuova in qualsiasi momento, sia per il desiderio di sostituire una chiave pubblica che è già stata utilizzata per una grande quantità di dati, sia perché la chiave privata corrispondente è stata compromessa in qualche modo;

   (d) i partecipanti potrebbero anche accedere alla directory elettronicamente. A tal fine, è obbligatoria una comunicazione sicura e autenticata dall'autorità al partecipante.

3. **Autorità delle chiavi pubbliche:** una sicurezza più forte per la distribuzione delle chiavi pubbliche può essere raggiunta fornendo un controllo più rigoroso sulla distribuzione delle chiavi pubbliche dalla directory. Come prima lo scenario assume che un'autorità centrale mantenga una directory dinamica delle chiavi pubbliche di tutti i partecipanti. Inoltre, ogni partecipante conosce in modo affidabile una chiave pubblica per l'autorità, con solo l'autorità che conosce la corrispondente chiave privata. Si verificano i seguenti passaggi:

   1. A invia un messaggio con timestamp all'autorità delle chiavi pubbliche contenente una richiesta per la chiave pubblica attuale di B.
   2. L'autorità risponde con un messaggio che è crittografato utilizzando la chiave privata dell'autorità PR_{AUTH}. Pertanto, A è in grado di decriptare il messaggio utilizzando

[5]Tipo di attacco informatico che si basa sulla falsificazione dell'identità di un dispositivo, utente o servizio all'interno di una rete, con l'obiettivo di ingannare la vittima o il sistema per ottenere accesso non autorizzato a informazioni riservate, dati sensibili o per lanciare ulteriori attacchi

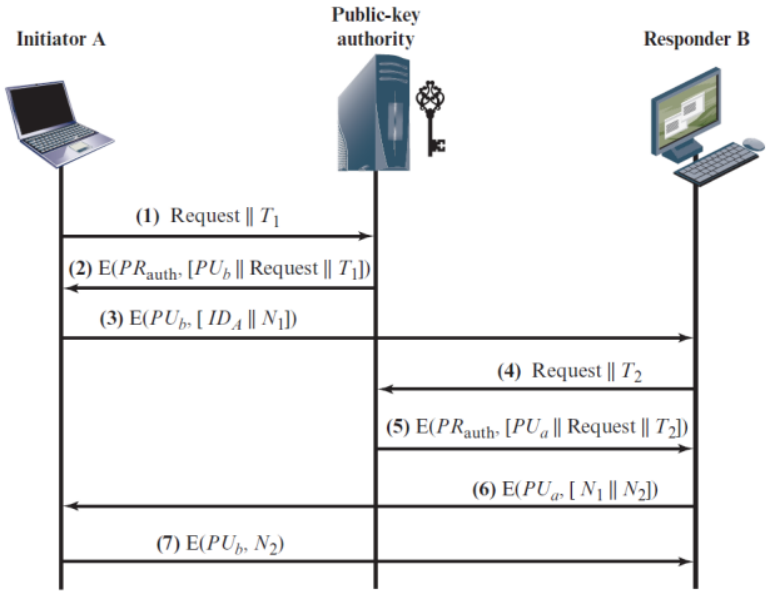


Figura 3: autorità delle chiavi pubbliche.

la chiave pubblica dell'autorità. Di conseguenza, A è assicurato che il messaggio provenga dall'autorità. Il messaggio include:

- la chiave pubblica di B, PU_B che A può utilizzare per crittografare i messaggi destinati a B;
- la richiesta originale utilizzata per consentire ad A di abbinare questa risposta con la corrispondente richiesta precedente e per verificare che la richiesta originale non sia stata alterata prima della ricezione da parte dell'autorità;
- il timestamp originale fornito affinché A possa determinare che questo non è un messaggio vecchio dell'autorità contenente una chiave diversa dalla chiave pubblica attuale di B.

3. A memorizza la chiave pubblica di B e la utilizza anche per crittografare un messaggio a B contenente un identificatore di A ID_A e un nonce N_1, che viene utilizzato per identificare univocamente questa transazione;

4 e 5. B recupera la chiave pubblica di A dall'autorità nello stesso modo in cui A ha recuperato la chiave pubblica di B. A questo punto, le chiavi pubbliche sono state consegnate in modo sicuro ad A e B, e possono iniziare il loro scambio protetto. Tuttavia, sono desiderabili due ulteriori passaggi;

6. B invia un messaggio ad A crittografato con PU_A e contenente il nonce di A N_1 così come un nuovo nonce generato da B N_2. Poiché solo B potrebbe aver decriptato il messaggio (3), la presenza di N_1 nel messaggio (6) assicura ad A che il corrispondente è B;

7. A restituisce N_2, crittografato utilizzando la chiave pubblica di B, per assicurare a B che il suo corrispondente è A.

Pertanto, sono richiesti un totale di sette messaggi. Tuttavia, i primi cinque messaggi devono essere utilizzati solo raramente, poiché sia A che B possono salvare la chiave pubblica

dell'altro per un uso futuro – una tecnica nota come caching. Periodicamente, un utente dovrebbe richiedere copie fresche delle chiavi pubbliche dei suoi corrispondenti per garantire la loro attualità.

5.2 Certificati di chiave pubblica

Lo scenario nella fig. 3 presenta alcuni svantaggi. L'autorità delle chiavi pubbliche potrebbe rappresentare un collo di bottiglia nel sistema, poiché un utente deve fare riferimento all'autorità per ottenere la chiave pubblica di ogni altro utente che desidera contattare. Come prima, la directory di nomi e chiavi pubbliche mantenuta dall'autorità è vulnerabile a manomissioni.

Un approccio alternativo, suggerito per la prima volta da Kohnfelder, è quello di utilizzare certificati che possono essere usati dai partecipanti per scambiare chiavi senza contattare un'autorità delle chiavi pubbliche, in un modo che è affidabile come se le chiavi fossero ottenute direttamente da un'autorità delle chiavi pubbliche. In sostanza, un certificato consiste in una chiave pubblica, un identificatore del proprietario della chiave e l'intero blocco firmato da una terza parte fidata. Tipicamente, la terza parte è un'autorità di certificazione, come un'agenzia governativa o un'istituzione finanziaria, che è fidata dalla comunità degli utenti. Un agente può presentare la propria chiave pubblica all'autorità in modo sicuro e ottenere un certificato. L'utente può ottenere il certificato e verificare che sia valido tramite la firma fidata allegata. Un partecipante può anche trasmettere le proprie informazioni sulla chiave a un altro trasmettendo il proprio certificato. Altri partecipanti possono verificare che il certificato sia stato creato dall'autorità.

Possiamo porre i seguenti requisiti su questo schema:

1. qualsiasi partecipante può leggere un certificato per determinare il nome e la chiave pubblica del proprietario del certificato;
2. qualsiasi partecipante può verificare che il certificato provenga dall'autorità di certificazione e non sia contraffatto;
3. solo l'autorità di certificazione può creare e aggiornare i certificati;
4. qualsiasi partecipante può verificare la validità temporale del certificato.

Nello schema di certificato (fig. 4), ogni partecipante fa richiesta all'autorità di certificazione, fornendo una chiave pubblica e richiedendo un certificato. La richiesta deve avvenire di persona o tramite qualche forma di comunicazione sicura e autenticata. Per il partecipante A, l'autorità fornisce un certificato della forma:

$$C_A = E(PR_{AUTH}, [T||ID_A||PU_A])$$

dove PR_{AUTH} è la chiave privata utilizzata dall'autorità e T è un timestamp. A può quindi trasmettere questo certificato a qualsiasi altro partecipante, che legge e verifica il certificato come segue:

$$D(PU_{AUTH}, C_A) = D(PU_{AUTH}, E(PR_{AUTH}, [T||ID_A||PU_A])) = (T||ID_A||PU_A)$$

Il destinatario utilizza la chiave pubblica dell'autorità, PU_{AUTH} per decifrare il certificato. Poiché il certificato è leggibile solo utilizzando la chiave pubblica dell'autorità, questo verifica che il certificato provenga dall'autorità di certificazione. Gli elementi ID_A e PU_A forniscono al destinatario il nome A e la chiave pubblica del titolare del certificato. Il timestamp T convalida l'attualità del certificato. Il timestamp contrasta il seguente scenario: la chiave privata di A

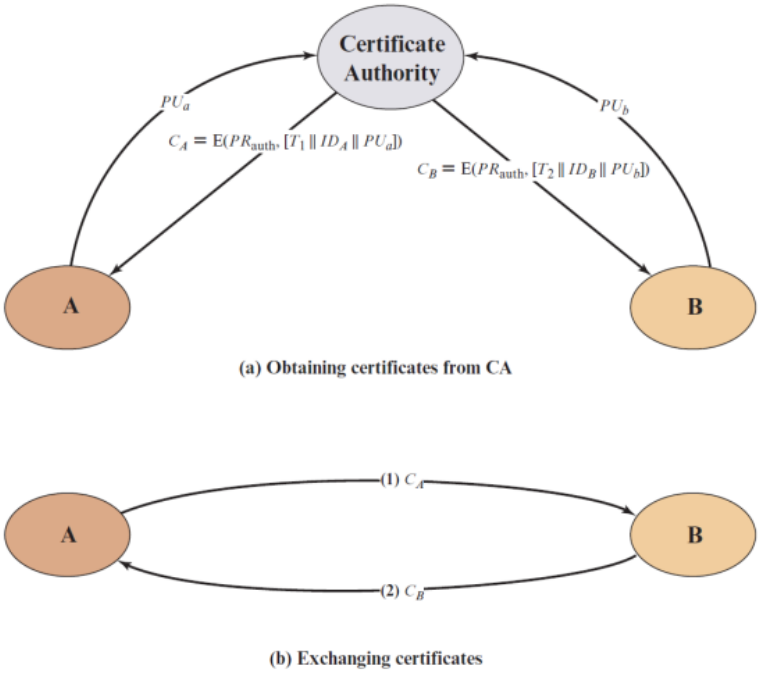


Figura 4: schema di certificato.

viene appresa da un avversario. A genera una nuova coppia di chiavi private/pubbliche e fa richiesta all'autorità di certificazione per un nuovo certificato. Nel frattempo, l'avversario riproduce il vecchio certificato a B. Se B crittografa i messaggi utilizzando la vecchia chiave pubblica compromessa, l'avversario può leggere quei messaggi.

In questo contesto, la compromissione di una chiave privata è paragonabile alla perdita di una carta di credito. Il proprietario annulla il numero della carta di credito ma è a rischio fino a quanto tutti i possibili comunicanti non sono a conoscenza del fatto che la vecchia carta di credito è obsoleta. Pertanto, il timestamp funge da data di scadenza. Se un certificato è sufficientemente vecchio, si presume che sia scaduto.

Uno schema è diventato universalmente accettato per la formattazione dei certificati di chiave pubblica: lo standard X.509. I certificati X.509 sono utilizzati nella maggior parte delle applicazioni di sicurezza di rete, inclusi la sicurezza IP, la sicurezza del livello di trasporto (TLS) e S/MIME.

5.2.1 Certificati X.509

La raccomandazione ITU-T X.509 fa parte della serie di raccomandazioni X.500 che definiscono un servizio di directory. La **directory** è, in effetti, un server o un insieme distribuito di server che gestisce un database di informazioni sugli utenti. Le informazioni includono una mappatura dal nome utente all'indirizzo di rete, oltre ad altri attributi e informazioni sugli utenti.

X.509 definisce un framework per la fornitura di servizi di autenticazione da parte della directory X.500 ai suoi utenti. La directory può fungere da repository di certificati a chiave pubblica. Ogni certificato contiene la chiave pubblica di un utente ed è firmato con la chiave privata di un'autorità di certificazione (CA) fidata. Inoltre, X.509 definisce protocolli di autenticazione

alternativi basati sull'uso di certificati a chiave pubblica. X.509 è uno standard importante per-

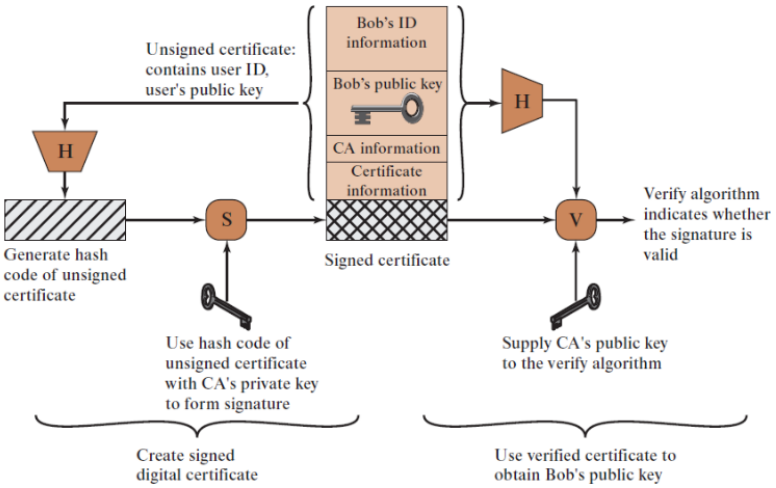


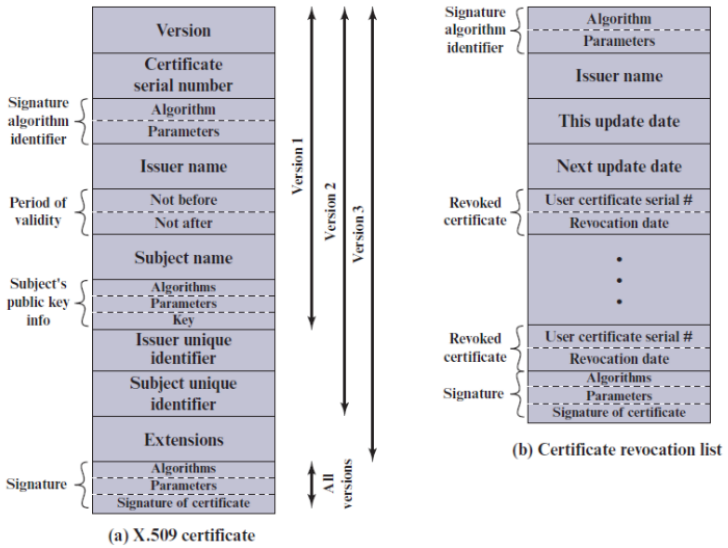
Figura 5: funzionamento di X.509.

ché la struttura dei certificati e i protocolli di autenticazione definiti in X.509 sono utilizzati in una varietà di contesti. Ad esempio, il formato del certificato X.509 è utilizzato in S/MIME, IP Security e SSL/TLS.

X.509 si basa sull'uso della crittografia a chiave pubblica e delle firme digitali. Lo standard non impone l'uso di uno specifico algoritmo di firma digitale né di una specifica funzione hash. La fig. 5 illustra lo schema generale X.509 per la generazione di un certificato a chiave pubblica. Il certificato per la chiave pubblica di Bob include informazioni di identificazione univoche per Bob, la chiave pubblica di Bob e informazioni di identificazione sulla CA, oltre ad altre informazioni come spiegato di seguito. Queste informazioni vengono quindi firmate calcolando un valore hash delle informazioni e generando una firma digitale utilizzando il valore hash e la chiave privata della CA. Bob può quindi trasmettere questo certificato ad altri utenti o allegare il certificato a qualsiasi documento o blocco di dati che firma. Chiunque abbia bisogno di utilizzare la chiave pubblica di Bob può essere certo che la chiave pubblica contenuta nel certificato di Bob sia valida perché il certificato è firmato dalla CA fidata.

Elementi di un certificato X.509

- **Versione:** differenzia tra le versioni successive del formato del certificato; il valore predefinito è la versione 1. Se l'identificatore univoco dell'emettente o l'identificatore univoco del soggetto sono presenti, il valore deve essere la versione 2. Se sono presenti una o più estensioni, la versione deve essere la versione 3. Sebbene a specifica X.509 sia attualmente alla versione 7, non sono state apportate modifiche ai campi che compongono il certificato dalla versione 3.
- **Numero di serie:** un valore intero unico all'interno della CA emittente che è inequivocabilmente associato a questo certificato.
- **Identificatore dell'algoritmo di firma:** l'algoritmo utilizzato per firmare il certificato insieme a eventuali parametri associati. Poiché queste informazioni sono ripetute nel campo della firma alla fine del certificato, questo campo ha poca, se non alcuna, utilità.



- **Nome dell'emittente:** nome X.500 della CA che ha creato e firmato questo certificato.
- **Periodo di validità:** consiste in due date: la prima e l'ultima in cui il certificato è valido.
- **Periodo di validità:** consiste in due date: la prima e l'ultima in cui il certificato è valido.
- **Nome del soggetto:** nome dell'utente a cui si riferisce questo certificato. Cioè, questo certificato certifica la chiave pubblica del soggetto che detiene la corrispondente chiave privata.
- **Informazioni sulla chiave pubblica del soggetto:** la chiave pubblica del soggetto, più un identificatore dell'algoritmo per il quale questa chiave deve essere utilizzata, insieme a eventuali parametri associati.
- **Identificatore univoco dell'emittente:** n campo di stringa di bit opzionale utilizzato per identificare univocamente la CA emittente nel caso in cui il nome X.500 sia stato riutilizzato per entità diverse.
- **Identificatore univoco del soggetto:** un campo di stringa di bit opzionale utilizzato per identificare univocamente il soggetto nel caso in cui il nome X.500 sia stato riutilizzato per entità diverse.
- **Estensioni:** un insieme di uno o più campi di estensione. Le estensioni sono state aggiunte nella versione 3.
- **Firma:** copre tutti gli altri campi del certificato. Un componente di questo campo è la firma digitale applicata agli altri campi del certificato. Questo campo include l'identificatore dell'algoritmo di firma. I campi identificatori univoci sono stati aggiunti nella versione 2 per gestire il possibile riutilizzo dei nomi dei soggetti e/o degli emittenti nel tempo. Questi campi sono raramente utilizzati.

L'autorità di certificazione firma il certificato con la sua chiave privata. Se la corrispondente chiave pubblica è nota a un utente, allora quell'utente può verificare che un certificato firmato dalla CA sia valido.

Ottenere un certificato I certificati utente generati da una CA hanno le seguenti caratteristiche:

- qualsiasi utente con accesso alla chiave pubblica della CA può verificare la chiave pubblica dell'utente che è stata certificata;
- nessuna parte, tranne l'autorità di certificazione, piò modificare il certificato senza che ciò venga rilevato.

Poiché i certificati sono inalterabili, possono essere collocati in una directory senza la necessità che la directory faccia sforzi speciali per proteggerli. Inoltre, un utente può trasmettere il proprio certificato direttamente ad altri utenti. Una volta che B è in possesso del certificato di A, B ha fiducia che i messaggi che crittografa con la chiave pubblica di A saranno sicuri da intercettazioni e che i messaggi firmati con la chiave privata di A sono inalterabili.

5.2.2 Modelli di fiducia

Diversi modelli di fiducia determinano come gli utenti stabiliranno la validità dei certificati.

Fiducia diretta Se tutti gli utenti si abbonano alla stessa CA, allora c'è una fiducia comune in quella CA. Tutti i certificati utente possono essere collocati nella stessa directory per l'accesso da parte di tutti gli utenti. Inoltre, la trasmissione dei certificati da utente a utente (come in PGP) è possibile.

Fiducia gerarchica e albero di fiducia Per una grande comunità di utenti, è più pratico avere un numero di CA, ciascuna delle quali fornisce in modo sicuro la propria chiave pubblica a una frazione degli utenti. La fiducia si estende da un certo numero di certificati radice. Questi certificati possono certificare certificati stessi, oppure possono certificare certificati che certificano ancora altri certificati lungo una certa catena. La validità del certificato foglia è verificata tracciando all'indietro dal suo certificatore, ad altri certificatori, fino a quando non si trova un certificato radice direttamente fidato.

Cross-certificazione Supponiamo che A e B abbiano ottenuto certificati dalle CA X_1 e X_2 rispettivamente. Se A non conosce in modo sicuro la chiave pubblica di X_2, allora A non può convalidare il certificato di B. Se le CA hanno scambiato le proprie chiavi pubbliche, allora A può ottenere la chiave pubblica di B tramite una catena di certificati (**cross-certificazione**).

A ottiene dalla directory, il certificato X_2 firmato da X_1 e può così ottenere la chiave pubblica di X_2 (e verificarla tramite la firma di X_1 sul certificato). A poi torna alla directory e ottiene il certificato di B firmato da X_2, che A può ora verificare con la copia fidata della chiave pubblica di X_2.

In X.509 questa catena di certificati è espressa come:

$$X_1 \ll X_2 \gg X_2 \ll B \gg$$

B ottiene la chiave pubblica di A con la chiave inversa:

$$X_2 \ll X_1 \gg X_1 \ll A \gg$$

Una catena con n elementi è:

$$X_1 \ll X_2 \gg X_2 \ll X_3 \gg \ldots X_n \ll B \gg$$

dove ogni coppia (X_i, X_{i+1}) di CA nella catena ha creato certificato l'uno per l'altro (memorizzati nella directory).

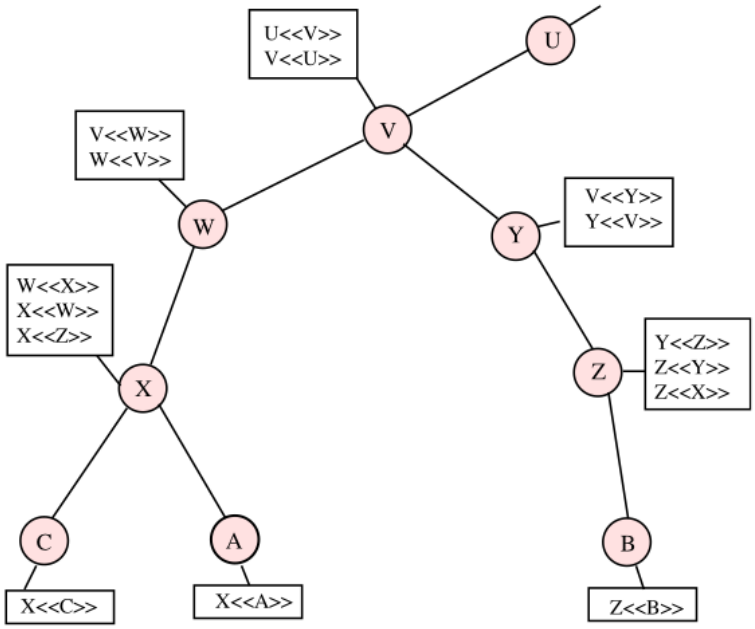


Figura 6: gerarchia X.509.

Gerarchia X.509 Cerchi connessi indicano la relazione gerarchica tra le CA. I rettangoli indicano i certificati mantenuti nella directory per ciascun ingresso della CA.

- **Certificati in avanti:** certificati di X generati da altre CA.
- **Certificati inversi:** certificati di altre CA generati da X.

Percorsi di connessione Facendo riferimento alla fig. 6

- Percorso $A \rightarrow B$: $X \ll W \gg W \ll V \gg V \ll Y \gg Y \ll Z \gg Z \ll B \gg$
- Percorso $B \rightarrow A$: $Z \ll Y \gg Y \ll V \gg V \ll W \gg W \ll X \gg X \ll A \gg$

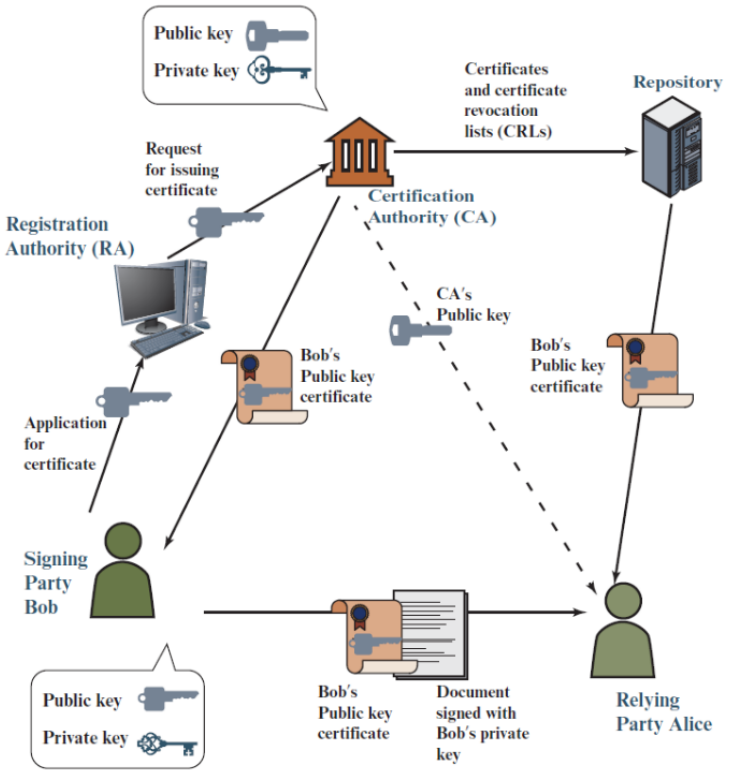
Revoca dei certificati

- Ogni certificato include un periodo di validità. Tipicamente, un nuovo certificato viene emesso poco prima della scadenza del vecchio.
- Può essere desiderabile, in alcune occasioni, revocare un certificato prima della scadenza, per uno dei seguenti motivi:
  - la chiave privata dell'utente si presume compromessa;
  - l'utente non è più certificato da questa CA;
  - il certificato della CA si presume compromesso.
- Ogni CA deve mantenere un elenco composto da tutti i certificati revocati ma non scaduti emessi da quella CA. Questi elenchi dovrebbero essere pubblicati nella directory.

X.509 v3 Il formato della versione 2 non trasmette tutte le informazioni che l'esperienza recente di progettazione e implementazione ha dimostrato essere necessarie. Invece di continuare ad aggiungere campi a un formato fisso, gli sviluppatori di standard hanno ritenuto necessario un approccio più flessibile. La versione 3 include un numero di estensioni opzionali. Le estensioni del certificato rientrano in tre categorie principali:

- informazioni su chiavi e politiche (trasmettono informazioni aggiuntive sulle chiavi del soggetto e dell'emittente, oltre a indicatori della politica del certificato);
- attributi del soggetto e dell'emittente (supportano nomi alternativi, in formati alternativi, per un soggetto del certificato o un emittente del certificato);
- vincoli sul percorso di certificazione (consentono di includere specifiche di vincolo nei certificati emessi per le CA da altre CA).

Ogni estensione consiste di un identificatore dell'estensione, un suo indicatore di criticità e un suo valore.



Interazione dei vari componenti Considera una parte affidabile, Alice, che ha bisogno di utilizzare la chiave pubblica di Bob. Alice deve prima ottenere in modo affidabile e sicuro una copia della chiave pubblica della CA. Questo può essere fatto in diversi modi e dipende dall'architettura PKI particolare e dalla politica aziendale.

Se Alice desidera inviare dati crittografati a Bob, controlla con il Repository per determinare se il certificato di Bob è stato revocato e, se non lo è, ottiene una copia del certificato di Bob.

Alice può quindi utilizzare la chiave pubblica di Bob per crittografare i dati inviati a Bob. Bob può anche inviare un documento ad Alice firmato con la chiave privata di Bob. Bob può includere il suo certificato con il documento o presumere che Alice abbia già o possa ottenere il certificato. In entrambi i casi, Alice utilizza prima la chiave pubblica della CA per verificare che il certificato sia valido, quindi utilizza la chiave pubblica di Bob (ottenuta dal certificato) per convalidare la firma di Bob.

Piuttosto che una singola CA, un'azienda potrebbe dover fare affidamento su più CA e più repository. Le CA possono essere organizzate in modo gerarchico, con una CA radice che è ampiamente fidata e firma il certificato della chiave pubblica delle CA subordinate. Molti certificati radice sono incorporati nei browser Web, quindi hanno una fiducia integrata in quelle CA. I server Web, i client di posta elettronica, gli smartphone e molti altri tipi di hardware e software supportano anche la PKI e contengono certificati radice fidati delle principali CA.

5.3 Web of trust e PGP

PGP utilizza un'infrastruttura di gestione delle chiavi basata su certificati per le chiavi pubbliche degli utenti. I certificati PGP (e la gestione delle chiavi) differiscono dai certificati X.509 in diversi modi importanti, ad esempio:

- una chiave PGP può avere più firme (anche "auto firma");
- ogni utente crea e firma certificati per le persone che conosce (pertanto, non è necessaria un'infrastruttura centrale);
- una nozione di "fiducia" è incorporata in ogni firma, e le firme per una singola chiave possono avere diversi livelli di fiducia (e gli utenti di un certificato agiscono in base al livello di fiducia);
- in un ambiente PGP, qualsiasi utente può agire come un'autorità di certificazione:
  - le firme digitali vengono usate come forma di introduzione: quando un utente firma la chiave di un altro, diventa introduttore di quella chiave;
  - man mano che questo processo continua, si stabilisce una rete di fiducia;
- qualsiasi utente PGP può convalidare il certificato della chiave pubblica di un altro utente, ma tale certificato è valido per un altro utente solo se riconosce il validatore come un introduttore fidato.

Sistema di reputazione in PGP Memorizzati nel keyring pubblico di ciascun utente ci sono indicatori che mostrano se l'utente considera una particolare chiave valida e il livello di fiducia che l'utente ripone nella chiave, in quanto ne è il proprietario, può fungere da certificatore delle chiavi di altri.

Esempio PGP Alice vuole comunicare con Bob. Ottiene il certificato della chiave pubblica PGP di Bob, ad esempio:

$$\text{Ellen, Fred, Giselle} \ll \text{Bob} \gg$$

Alice non conosce nessuno dei firmatari, quindi da un server di certificati ottiene il certificato PGP di Giselle:

$$\text{Henry, Irene, Giselle} \ll \text{Giselle} \gg$$

Conosce vagamente Henry, quindi per verificare il certificato di Giselle ottiene il certificato di Henry:

$$\text{Ellen, Henry} \ll \text{Henry} \gg$$

Dunque abbiamo:

- Ellen, Fred, Giselle $\ll$ Bob $\gg$
- Henry, Irene, Giselle $\ll$ Giselle $\gg$
- Ellen, Henry $\ll$ Henry $\gg$

Alice nota che la firma di Henry è al livello di fiducia **casuale**, quindi decide di cercare conferma altrove. Ottiene il certificato di Ellen:

$$\text{Jack, Ellen} \ll \text{Ellen} \gg$$

Riconosce immediatamente Jack come suo cugino e, poiché ha il certificato di Jack, lo utilizza per convalidare il certificato di Ellen: nota che la sua firma è al livello di fiducia **positivo**, e quindi accetta il certificato di Ellen come valido e lo utilizza per convalidare quello di Bob.

Nota che anche Ellen ha firmato il certificato con fiducia **positiva**, quindi conclude che il certificato, e la chiave pubblica che contiene, sono **affidabili**.

Alice ha seguito due catene di firme:

- Henry $\ll$ Henry $\gg\!\!\longrightarrow$ Henry $\ll$ Giselle $\gg\!\!\longrightarrow$ Giselle $\ll$ Bob $\gg$
- Jack $\ll$ Ellen $\gg\!\!\longrightarrow$ Ellen $\ll$ Bob $\gg$

(dove le firme non verificate sono state omesse).

I livelli di fiducia hanno influenzato il modo in cui Alice ha controllato il certificato.

Distinzione tra Certificati X.509 e PGP

- I certificati **X.509** includono un elemento di fiducia, ma la fiducia **non è indicata** nel certificato.
- I certificati **PGP** indicano il livello di fiducia, ma lo stesso livello di fiducia può avere **significati diversi** per diversi firmatari (cfr. anche risoluzione dei nomi).

Web of trust La Web of trust (rete di fiducia) è un concetto fondamentale nella crittografia a chiave pubblica, in particolare utilizzato da software come PGP e GnuPG (compatibili con OpenPGP), per stabilire l'autenticità del legame tra una chiave pubblica e il suo legittimo proprietario (utente). Si tratta di un sistema deventralizzato che funge da alternativa o complemento all'affidamento esclusivo su un'autorità di certificazione (CA) centralizzata, tipica delle infrastrutture a chiave pubblica (PKI) tradizionali.

- **Funzionamento principale:** la rete di fiducia non si basa su una singola autorità, ma sul concetto di fiducia reciproca tra gli utenti:
  - **firma del certificato:** un certificato di identità OpenPGP, che contiene la chiave pubblica e le informazioni sul suo proprietario, può essere firmato elettronicamente da altri utenti.

- **Attestazione:** apponendo la propria firma, un utente attesta di aver verificato (solitamente di persona, in eventi noti come *key signing party*) che quella specifica chiave pubblica appartiene effettivamente a quella specifica persona.
- **Livelli di fiducia:** ciascun utente configura il proprio livello di fiducia in modo indipendente, decidendo di quali chiavi firmate fidarsi. Ad esempio, un utente può decidere che un certificato è valido se è stato firmato da:
  * un conoscente di cui si fida completamente;
  * oppure, da tre conoscenti di cui si fida parzialmente.

Questo approccio si basa sull'idea dei 6 gradi di separazione, permettendo a un utente di fidarsi indirettamente di una chiave che non ha mai incontrato, purché sia stata validata da una catena di persone fidate (i propri contatti e i loro contatti).

La differenza principale con la PKI tradizionale sta nella struttura e nella flessibilità

Caratteristica	Web of trust (PGP/GnuPG)	PKI (X.509)
Autorità	Decentralizzata. La fiducia è risposta negli individui.	**Centralizzata/gerarchica**. La fiducia è risposta nelle CA (autorità di certificazione).
Validazione	Multipla e configurabile. Le firme provengono da una rete di utenti.	Singola. Un certificato è firmato da una CA, risalendo fino a un root certificate preinstallato.
Flessibilità	Alta. L'utente sceglie i propri parametri di fiducia (quante firme servono e da chi).	Bassa. L'utente deve accettare implicitamente tutte le CA include nel browser o software.

In sintesi, la Web of trust pone il controllo e la responsabilità della validazione della chiave nelle mani dell'utente individuale, a differenza delle PKI che impongono la fiducia nelle autorità centrali.

6 Sicurezza della rete e dei protocolli Internet

Definizione 1 (Protocollo) *Un protocollo consiste in un insieme di regole (convenzioni) che determinano lo scambio di messaggi tra due o più principali (entità in comunicazione).*

I protocolli di sicurezza (o crittografici) utilizzano meccanismi crittografici per raggiungere obbiettivi di sicurezza.

6.1 Costruzione di un protocollo di stabilimento delle chiavi

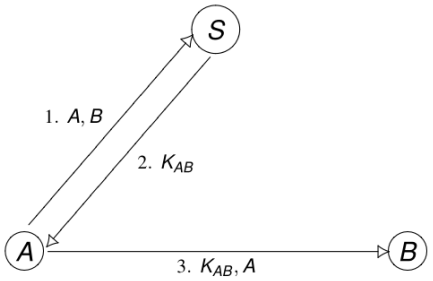
Il completamento con successo dello stabilimento della chiave dell'autenticazione dell'entità è solo l'inizio di una sessione di comunicazione sicura: una volta che una nuova chiave è stata stabilita, il suo utilizzo serve a proteggere i dati da comunicare con meccanismi crittografici scelti. A tal fine, gli utenti interagiscono con un server fidato, ovvero è fidato per non impegnarsi in alcuna altra attività che comprometterebbe deliberatamente la sicurezza degli utenti, e per generare la

nuova chiave in modo tale che sia sufficientemente casuale da prevenire che un attaccante ottenga informazioni utili su di essa.

Pertanto il protocollo coinvolge 3 principali:

- Alice (A);
- Bob (B);
- il server fidato (S).

Il ruolo di S è quello di generare la nuova chiave di sessione K_{AB} e inviarla ad A e a B. Il protocollo dovrebbe permettere di conoscere la chiave solo ai principali A e B e a nessun altro, con la possibile eccezione di S. A e B dovrebbero sapere che K_{AB} è stata generata di recente.



Un primo tentativo

- A contatta S inviando le identità delle 2 parti che condivideranno la chiave di sessione;
- S invia la chiave K_{AB} ad A.
- A passa K_{AB} a B.

K_{AB} non contiene alcuna "informazione" su A e su B, ma è semplicemente un nome per la stringa di bit che rappresenta la chiave di sessione. Prima di esaminare la mancanza di sicurezza di questo protocollo, si può notare che questa è una specifica del protocollo significativamente incompleta.

- Solo i messaggi scambiati in un'esecuzione riuscita del protocollo sono specificati. Non c'è alcuna descrizione di cosa succede nel caso in cui venga ricevuto un messaggio di formato errato o che non venga ricevuto alcun messaggio.
- Questo è standard per i protocolli di sicurezza, mentre nella specifica dei protocolli di comunicazione ordinari è comune includere queste informazioni (poiché è essenziale per dimostrare le proprietà funzionali di base).
- Nessuna specifica delle azioni interne dei principali. Le azioni interne sono abbastanza ovvie per molti protocolli (ad esempio calcolare ciò che p necessario per generare il prossimo messaggio), ma per altri protocolli ci sono numerose alternative e la scelta può avere rilevanza critica per la sicurezza.

- A e B sanno che i messaggi ricevuti fanno parte del protocollo. È comune omettere tali dettagli che sarebbero necessari affinché un computer in rete possa tracciare il progresso di un'esecuzione particolare del protocollo. Questo può includere dettagli su quale chiave dovrebbe essere utilizzata per decrittografare un messaggio ricevuto che è stato crittografato.
- Nonostante le evidenti limitazioni associate alla specifica dei protocolli mostrando solo i messaggi di un'esecuzione riuscita, rimane il metodo più popolare per descrivere i protocolli di sicurezza. Tuttavia, ci sono molti lavori che hanno affrontato questi problemi per "disambiguare" la notazione.

Una rappresentazione equivalente è la seguente

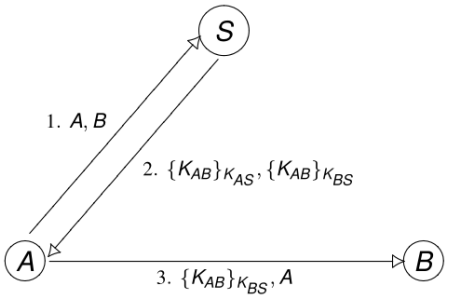
1. $A \to B$: A, B
2. $S \to A$: K_{AB}
3. $A \to B$: K_{AB}, A

Notare che i nomi di mittente/ricevente $A \to B$ non fanno parte del messaggio e non è detto che i messaggi raggiungano automaticamente la loro destinazione in modo sicuro.

Problemi I problemi sono di trasporto della chiave di sessione. Infatti la chiave di sessione K_{AB} deve essere trasportata ad A e B, ma a nessun'altra parte. Un'assunzione realistica nei sistemi di comunicazione tipici, come Internet e le reti aziendali.

Assunzione di sicurezza 1 *L'avversario è in grado di intercettare tutti i messaggi inviati in un protocollo di sicurezza.*

Se questa possibilità può essere esclusa allora, probabilmente non c'è bisogno di applicare alcuna sicurezza. È quindi possibile utilizzare un algoritmo crittografico e le chiavi associate.



Secondo tentativo Assumiamo che S condivida inizialmente una chiave segreta con ciascun utente del sistema:

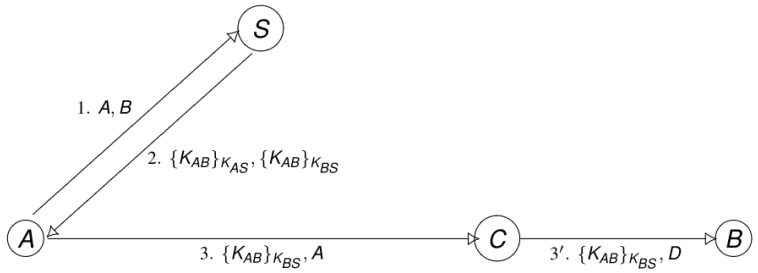
- K_{AS} con A;
- K_{BS} con B;
- crittografia del messaggio 2.

Problema Il problema non è che il protocollo riveli al chiave segreta K_{AB}, ma che le informazioni su chi altro possiede K_{AB} non sono protette. L'avversario potrebbe non solo essere in grado di intercettare i messaggi inviati, ma anche di catturare i mesaggi e alterarli.

Assunzione di sicurezza 2 *L'avversario è in grado di alterare tutti i messaggi inviati in un protocollo di sicurezza utilizzando qualsiasi informazione disponibile.*

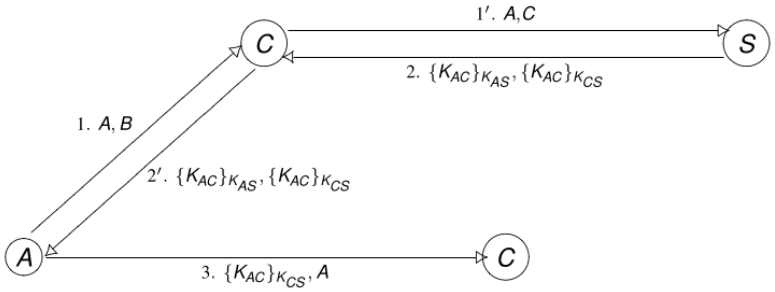
Può anche reindirizzare qualsiasi messaggio a qualsiasi altro principale (o può semplicemente intercettarli). Questo include la capacità di generare e inserire messaggi completamente nuovi.

A differenza dei protocolli di comunicazione ordinari, nei protocolli di sicurezza è coinvolto un principale sconosciuto e imprevedibile. Sebbene ci possano essere non più di 4 o 5 messaggi coinvolti in un'esecuzione legittima del protocollo, ci sono un numero infinito di variazioni in cui partecipa l'avversario.



Attacchi al secondo tentativo L'avversario C intercetta semplicemente il messaggio da A a B e sostituisce l'identità di D con quella di A (dove D potrebbe essere qualsiasi identità, inclusa quella di C stesso). B crede di condividere la chiave con D, mentre in realtà la sta condividendo con A. Il risultato di questo attacco dipenderò dallo scenario in cui viene utilizzato il protocollo, ma potrebbe includere azioni come B che fornisce informazioni ad A che avrebbero dovuto essere condivise solo con D. Sebbene C non ottenga K_{AB}, possiamo comunque considerare il protocollo compromesso poiché non soddisfa il requisito che gli utenti dovrebbero sapere chi altro conosce la chiave di sessione.

Ma c'è anche un altro attacco più serio. C altera il messaggio da A a S in modo che S



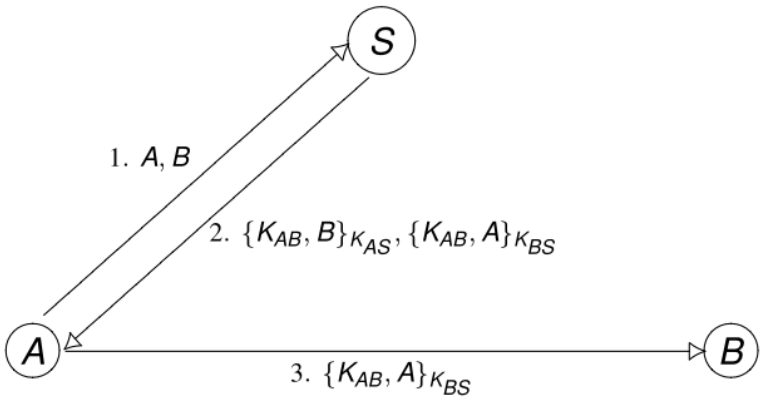
crittografi la chiave K_{AC} con la chiave di C K_{CS}, invece che on la chiave di B. Poiché A non può distinguere tra i messaggi crittografati destinati ad altri principali, non rileverà l'alterazione.

Nota anche che K_{AC} è semplicemente un nome formale per la stringa di bit che rappresenta la chiave di sessione, quindi sarà accettato da A. C intercetta il messaggio da A destinato a B in modo che B non rilevi alcuna anomalia. Pertanto A crederà che il protocollo sia stato completato

con successo con B, mentre C conosce K_{AC} e può quindi spacciarsi per B, oltre a poter apprendere tutte le informazioni che A invia destinate a B. A differenza dell'attacco precedente, questo avrà successo se C è un utente legittimo del sistema conosciuto da S, il che è un'assunzione realistica:

Assunzione di sicurezza 3 *L'avversario può essere un partecipante legittimo al protocollo (un insider), o una parte esterna (un outsider), o una combinazione di entrambi.*

Per superare questo attacco, i nomi dei principali che devono condividere K_{AB} devono essere legati crittograficamente alla chiave. Nessuno dei 2 attacchi precedenti ha successo sul seguente protocollo:



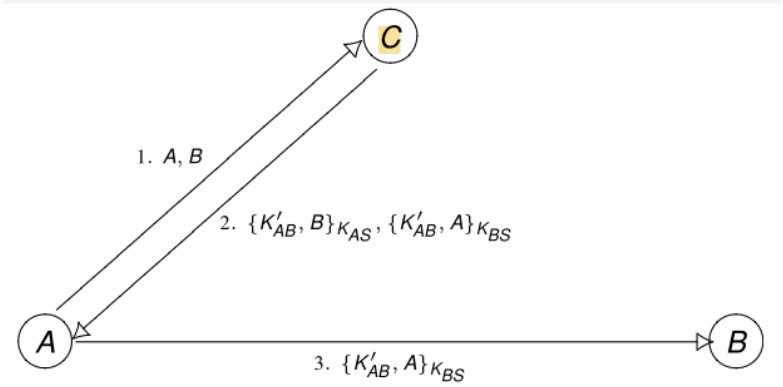
Legare crittograficamente i nomi dei principali alla chiave Il protocollo è migliorato al punto in cui un avversario non è in grado di attaccarlo intercettando o alterando i messaggi inviati tra le parti oneste. Tuttavia, anche ora il protocollo non è sufficientemente buono per garantire la sicurezza in condizioni operative normali.

Problema con le chiavi di sessione Il problema deriva dalla differenza di qualità tra le chiavi di crittografia a lungo termine condivise inizialmente con S e le chiavi di sessione K_{AB} generate per ciascuna esecuzione del protocollo. I motivi per l'uso delle chiavi di sessione sono:

- si prevede che siano vulnerabili ad attacchi (da crittoanalisi);
- le comunicazioni in diverse sessioni dovrebbero essere separate; in particolare non dovrebbe essere possibile riprodurre messaggi da sessioni precedenti;
- un'intera classe di attacchi diventa possibile quando vecchie chiavi (o altri dati rilevanti per la sicurezza) possono essere riprodotte in una sessione successiva.

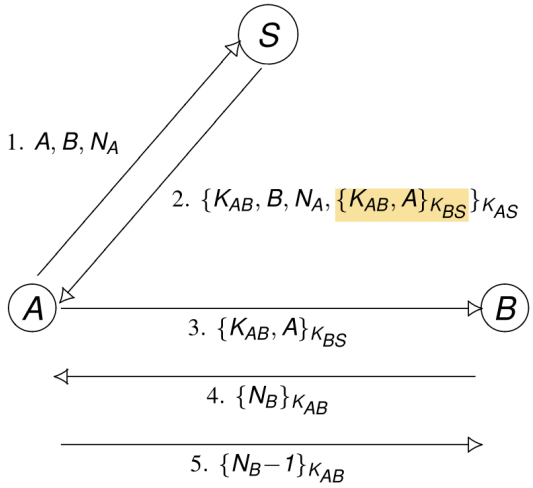
Assunzione di sicurezza 4 *L'avversario è in grado di ottenere il valore della chiave di sessione K_{AB} utilizzata in qualsiasi esecuzione "sufficientemente vecchia" del protocollo.*

C intercetta il messaggio da A a S (che non gioca alcun ruolo in questa esecuzione).



Attacco di replay $K_{AB}{}'$ è una chiave vecchia utilizzata da A e B in una sessione precedente. Secondo l'assunzione di sicurezza 1 ci si può aspettare che C conosca i messaggi crittografati tramite i quali $K_{AB}{}'$ è stato consegnato ad A e a B.

Secondo l'assunzione di sicurezza 4 ci si può aspettare che C conosca il valore di $K_{AB}{}'$. Pertanto, quando A completa il protocollo con B, C è in grado di decrittografare le informazioni successive crittografate con $K_{AB}{}'$ o di inserire o alterare i messaggi la cui integrità è protetta da $K_{AB}{}'$.



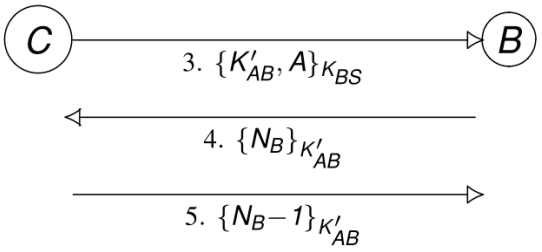
Utilizzo del nonce nel protocollo A invia il suo nonce N_A a S con la richiesta di una nuova chiave. Se lo stesso valore viene ricevuto con la chiave di sessione, allora A può dedurre che la chiave non è stata riprodotta. Questa deduzione sarà valida finché la chiave di sessione e il nonce sono legati insieme in modo crittografico in maniera tale che solo S possa aver formato un messaggio del genere.

Problema Il problema però è che N_A è solo un numero! Non c'è nulla in N_A che identifichi chi lo ha creato pertanto si può scrivere anche N_1 o N1. Se la chiave crittografata per B è inclusa nella parte crittografata del messaggio di A, allora A può avere la certezza che sia recente. È allettante credere che A possa trasmettere questa certezza a B in un ulteriore handshake:

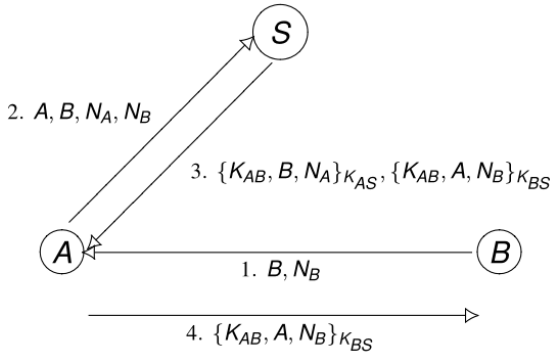
- B genera un nonce N_B e lo invia ad A protetto da K_{AB} stesso;
- A utilizza K_{AB} per inviare una risposta a B.

Questo è uno dei protocolli di sicurezza più celebri: il **protocollo di Needham Schroeder del 1978**.

Attacco al protocollo di Needham Schroeder Sfortunatamente questo protocollo è vulnerabile all'attacco di Denning e Sacco. Il problema è che l'assunzione "solo A sarà in grado di formare una risposta corretta al messaggio 4. di B" è sbagliata. Poiché si prevede che l'avversario C conosca il valore di una **vecchia chiave di sessione** (**replay attack**), questa assunzione è irrealistica. C si maschera da A e così persuade B a utilizzare la vecchia chiave K_{AB}:

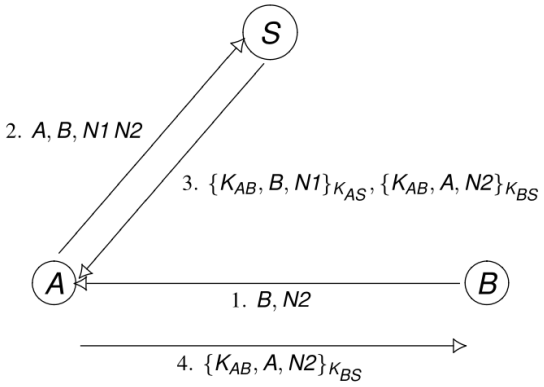


Approccio diverso Abbandonare che l'assunzione che sia scomodo per entrambi B e A inviare le loro sfide a S. Il protocollo è ora avviato da B, che invia prima il suo nonce N_B ad A. A aggiunge



il suo nonce N_A e invia entrambi a S, che ora è in grado di restituire K_{AB} in messaggi separati per A e B. Entrambi i messaggi possono essere verificati come freschi dai rispettivi destinatari.

Problema A poteva verificare non solo che la chiave è nuova e conosciuta solo da A, B e S, ma anche che B ha effettivamente ricevuto la chiave. Questa proprietà di conferma della chiave è raggiunta grazie all'uso della chiave da parte di B nel messaggio 4, assumendo che $\{N_B\}_{K_{AB}}$ non possa essere formata senza la conoscenza di K_{AB}. Nel protocollo finale, né A né B possono dedurre alla fine di un'esecuzione riuscita del protocollo che l'altro ha effettivamente ricevuto K_{AB}. Questo protocollo evita tutti gli attacchi visti finora, a condizione che l'algoritmo crittografico utilizzato fornisca le proprietà di riservatezza e integrità e che il server S agisca correttamente.



6.2 Protocollo di chiave pubblica di Needham Schroeder (NPSK)

Con il termine generico di protocollo di Needham-Schroeder si possono identificare due protocolli di comunicazione progettati per permettere comunicazioni cifrate su reti non sicure. I protocolli furono proposti da Roger Needham e Micheal Schroeder nel 1978.

- Il *protocollo di Needham-Schroeder a chiave segreta* è basato sulla crittografia simmetrica ed è alla base del protocollo di rete Kerberos. Il protocollo permette di stabilire una chiave di sessione utilizzabile da due entità di rete per proteggere le successive comunicazioni.
- Il *protocollo di Needham-Schroeder a chiave pubblica* è invece basato sulla crittografia asimmetrica e permette di assicurare la mutua autenticazione tra due entità di rete. Nella sua forma proposta non è sicuro.

6.2.1 Il protocollo a chiave segreta

Scenario

- Alice (A) e Bob (B) sono due entità di rete che devono comunicare in modo sicuro utilizzando un collegamento di rete non sicuro.
- S è un server fidato, ovvero che gode della fiducia di entrambe le parti.
- K_{AS} è una chiave simmetrica nota esclusivamente ad A e S.
- K_{BS} è una chiave simmetrica nota esclusivamente a B e S.
- K_{AB} è una chiave simmetrica di sessione generata da S durante l'esecuzione dei passi del protocollo.
- N_A e N_B sono nonce crittografici, ovvero numeri casuali da usare una volta sola.

Si suppone che sia A a cominciare la comunicazione.

Descrizione A spedisce un messaggio a S con la sua identità e quella di B, per dichiarare che intende comunicare con B. Allega anche un numero N_A:

$$A \to S\colon A, B, N_A$$

Il server genera la chiave di sessione K_{AB} e risponde ad A inviandole:

- la chiave K_{AB} appena generata;
- la coppia (K_{AB}, A) criptata con la chiave K_{BS}, in modo che possa essere inoltrata a B perché venga reso partecipe;
- il nonce N_A che assicura ad A che il messaggio è nuovo e che S sta rispondendo a quel particolare messaggio;
- B: l'inclusione dell'identificatore di B dice ad A con chi lei sta condividendo la chiave.

Il tutto è criptato con la chiave segreta K_{AS}, nota solo ad A e a S:

$$S \to A\colon \{N_A, K_{AB}, B, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$$

A comunica a B la chiave di sessione e il proprio identificativo, criptati con la chiave K_{BS}, come comunicata dal server con il precedente messaggio. B può decriptare il messaggio e il fatto che sia stato cifrato da una entità fidata (ovvero S) lo rende autentico:

$$A \to B\colon \{K_{AB}, A\}_{K_{BS}}$$

B risponde ad A con un nonce criptato con la chiave di sessione K_{AB}, per mostrare che è in possesso della chiave:

$$B \to A\colon \{N_B\}_{K_{AB}}$$

A decifra il nonce di B, lo modifica con una semplice operazione, lo cifra nuovamente e lo rispedisce indietro, provando così che è ancora attiva e in possesso della chiave:

$$A \to B\colon \{N_B - 1\}_{K_{AB}}$$

Da questo momento in poi la comunicazione è pienamente stabilita e viene condotta da entrambe le parti inviando messaggi cifrati con la chiave di sessione K_{AB}.

Vulnerabilità Il protocollo è suscettibile ad attacchi di tipo replay. Se un attaccante E conosce una vecchia chiave ${K_{AB}}'$ scambiata in una precedente esecuzione del protocollo, allora:

- E può replicare il messaggio $\{{K_{AB}}', A\}_{K_{BS}}$ a B;
- B accetta la chiave ${K_{AB}}'$ non sapendo che è compromessa (quindi non sa che la chiave non è nuova).

Soluzione Utilizzare i timestamp all'interno dei messaggi (ciò si vedrà come si utilizza nel protocollo di rete Kerberos).

6.2.2 Il protocollo a chiave pubblica

Scenario

- A e B sono due entità di rete che devono comunicare in modo sicuro utilizzando un collegamento di rete non sicuro.
- S è un server che gode della fiduccia di entrambi e si occupa di distribuire chiavi pubbliche su richiesta.
- K_{PA} e K_{SA} sono rispettivamente la chiave pubblica e la chiave segreta di A.

- K_{PB} e K_{SB} similmente sono le chiavi di B.
- K_{PS} e K_{SS} similmente sono le chiavi di S.

È importante specificare che mentre A e B usano la chiave pubblica per cifrare e quella privata per decifrare, S usa la chiave privata K_{SS} per cifrare e la chiave pubblica K_{PS} per decifrare; così facendo il server firma le proprie comunicazioni.

Descrizione A chiede a S la chiave pubblica di B:

$$A \to S\colon A, B$$

S risponde, inviando anche l'identificativo di B per conferma. A può usare la chiave pubblica K_{PS} per verificare la firma di S e verificarne l'autenticità confrontando l'id di B ricevuto con quello in suo possesso:

$$S \to A\colon \{K_{PB}, B\}_{K_{SS}}$$

Alice genera un nonce N_A e lo invia a B cifrandolo con la chiave appena ricevuta:

$$A \to B\colon \{N_A, A\}_{K_{PB}}$$

B decifra il messaggio mediante la sua chiave privata, vede che è di A e richiede la sua chiave pubblica a S:

$$B \to S\colon B, A$$

S soddisfa la richiesta di B:

$$S \to B\colon \{K_{PA}, A\}_{K_{SS}}$$

B genera un nonce N_B e lo invia ad A insieme a N_A, per provare che è in possesso della chiave privata K_{SB}:

$$B \to A\colon \{N_A, N_B\}_{K_{PA}}$$

A conferma N_B a B per provare a sua volta che è in possesso della chiave privata K_{SA}

$$A \to B\colon \{N_B\}_{K_{PB}}$$

Da questo momento in poi A e B si sono autenticati a vicenda e sono gli unici a conoscenza di N_A e N_B.

Vulnerabilità Il protocollo è suscettibile ad attacchi di tipo man-in-the-middle: un impostore I può ingannare A e convincerla a iniziare una sessione di comunicazione con lui e successivamente inoltrare i messaggi a B convincendolo di essere in comunicazione con A.

A parte le comunicazioni con S, che rimangono inalterate, l'attacco si svolge come segue: A invia N_A a I, che decifra il messaggio con K_{SI}:

$$A \to I\colon \{N_A, A\}_{K_{PI}}$$

I inoltra il messaggio a B, cifrandolo con la corrispettiva chiave pubblica, fingendo che sia A a voler comunicare con lui:

$$I \to B\colon \{N_A, A\}_{K_{PB}}$$

B risponde inviando N_B. Questo messaggio arriva a I, che non può decifrarlo perché non è in possesso di K_{SA}:

$$B \to I\colon \{N_A, N_B\}_{K_{PA}}$$

I lo inoltra ad A:

$$I \to A\colon \{N_A, N_B\}_{K_{PA}}$$

A crede che N_B sia il nonce di I (**attacco di binding**[6]), quindi lo conferma come da protocollo:

$$A \to I\colon \{N_B\}_{K_{PI}}$$

Ora I conosce N_B, lo cifra con K_{PB} e invia a B, che vede una conferma valida

$$I \to B\colon \{N_B\}_{K_{PB}}$$

I agisce da "uomo nel mezzo" e intercetta tutti i messaggi di B, che crede di essere in comunicazione sicura con A.

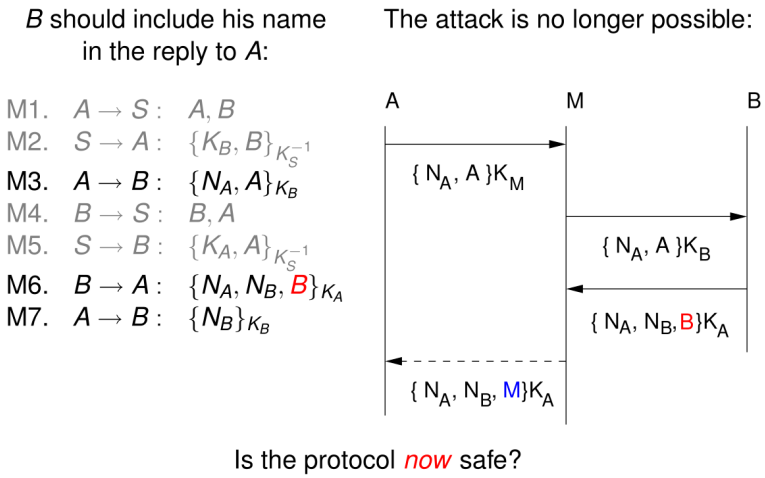
Gavin Lowe L'attacco è stato descritto per la prima volta da Gavin Lowe nel 1995. La versione corretta del protocollo, chiamata Needham-Schroeder-Lowe, sostituisce il sesto messaggio

$$B \to A\colon \{N_A, N_B\}_{K_{PA}}$$

con

$$B \to A\colon \{N_A, N_B, B\}_{K_{PA}}$$

In questo modo A si può accorgere che i messaggi non arrivano dall'intruso, con cui aveva iniziato la comunicazione, ma da B.

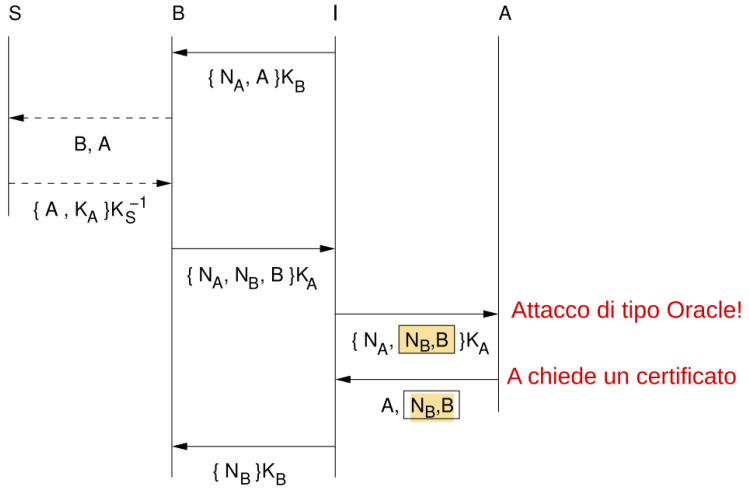


Il protocollo è ancora vulnerabile ad un attacco di tipo flow

Il protocollo non è ancora sicuro Un impostore I può ancora fare un altro attacco man-in-the-middle. I invia a B un messaggio spacciandosi per A (spoofing attack)

$$I \to B\colon \{N_A, A\}_{K_B}$$

[6]Un **attacco di binding** si verifica quando l'informazione segreta che in questo caso è il nonce non è legata crittograficamente all'identificatore di colui che quell'informazione l'ha generata.



B è convinto che A voglia effettuare uno scambio di chiavi con il protocollo Needham-Schroeder-Lowe e dunque ha bisogno della chiave pubblica di A (sta seguendo il protocollo, deve inviare in risposta il proprio nonce crittografato). Chiede ad S la chiave pubblica di A dicendo di essere B

$$B \to S\colon B, A$$

S risponde

$$S \to B\colon \{A, K_A\}_{K_S}^{-1}$$

ovvero la chiave pubblica di A e l'identificatore di A (per confermare che quella chiave è associata all'identità di A). Il messaggio è firmato con la chiave privata del server $K_S{\{} - 1\}$B risponde cifrando con la chiave pubblica di A la tripla $(N_A, N_B, B)_{K_A}$ e solo A può leggere tale messaggio. Quindi I riceve il messaggio da B

$$B \to I\colon \{N_A, N_B, B\}_{K_A}$$

e lo inoltra ad A (**Oracle attack**[7]). A legge la richiesta

$$I \to A\colon \{N_A, N_B, B\}_{K_A}$$

convinto che N_B, B sia l'identificatore di un principale legittimo (dato che ha ottenuto la chiave pubblica K_A, dunque deve aver interagito con il server S). Quello appena descritto rappresenta un **type flaw attack**[8]. Quindi A risponde con il messaggio

$$A \to I\colon A, N_B, B$$

in chiaro perché ha scambiato N_B, B per un identificatore, ed è convinto di inviare verso il server una richiesta per ottenere il certificato A, N_B, B ("io sono A, voglio ottenere il certificato di N_B, B"). Tale richiesta viene intercettata da I che invia a B il messaggio:

$$I \to B\colon \{N_B\}_{K_B}$$

chiudendo così il protocollo.

[7] A viene usato come un "oracolo" per ottenere la decifratura del messaggio che solo lui può decifrare.

[8] Un attacco **type flaw** si verifica quando il destinatario di un messaggio accetta quel messaggio come valido ma impone un'interpretazione diversa della sequenza di bit rispetto al principale che lo ha creato.