Appunti di network security

Riccardo Torre

18 novembre 2025

Indice

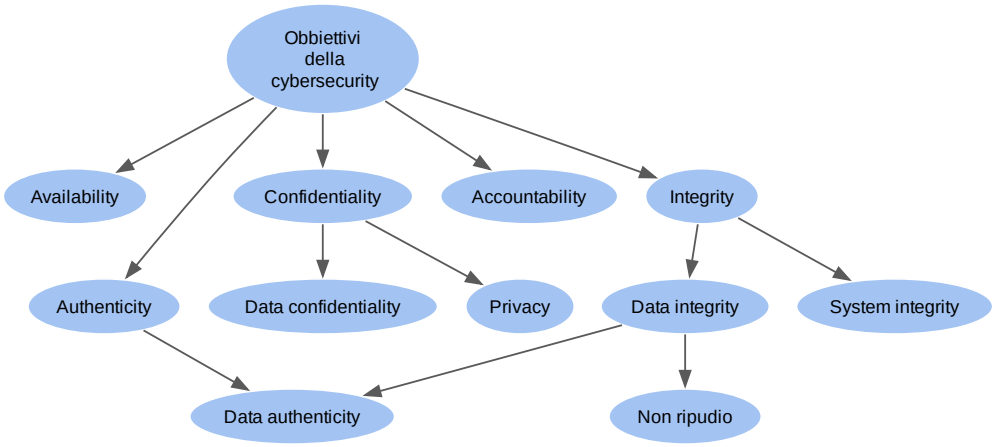
1 Introduzione

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.

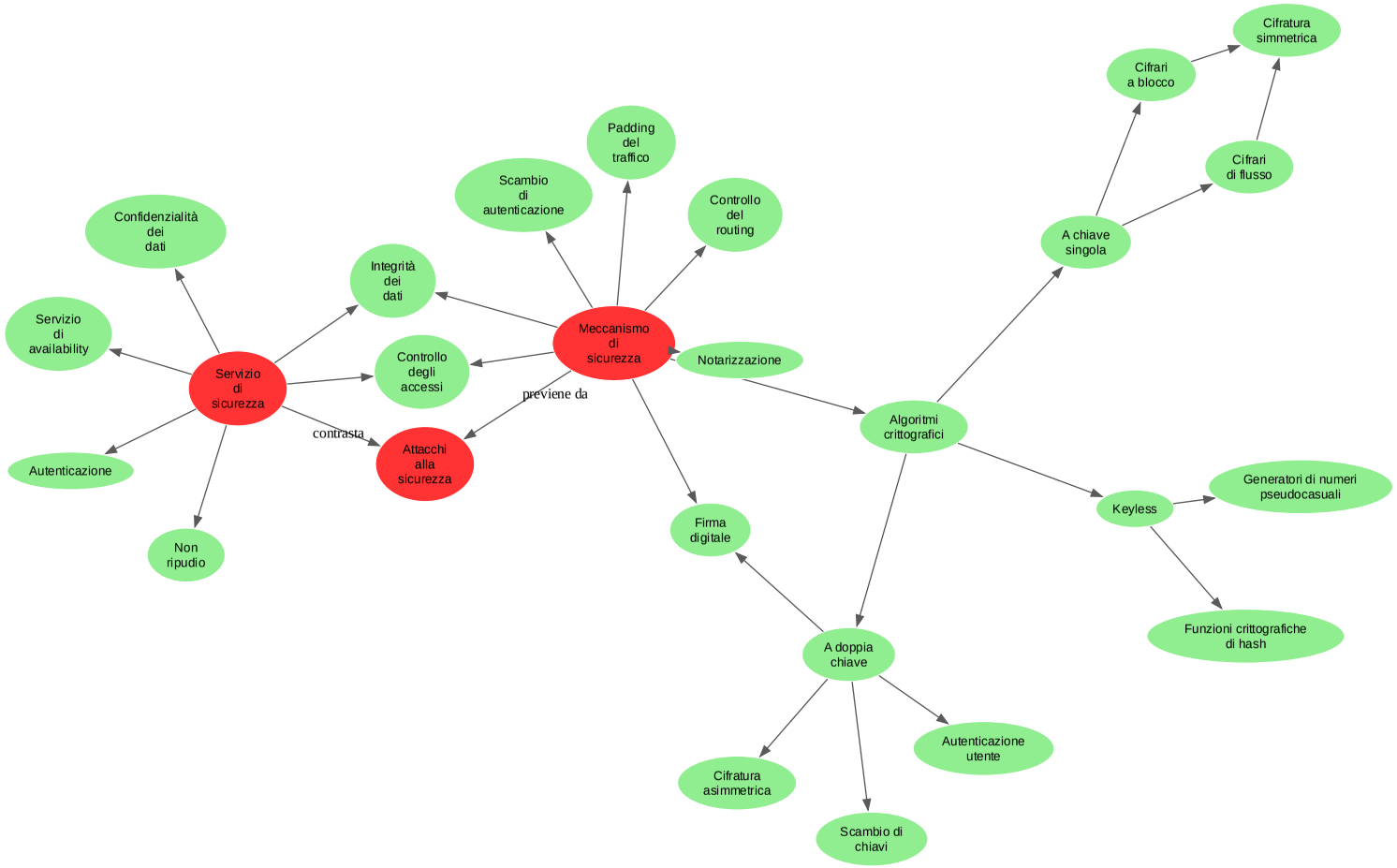
Accountability (responsabilità) È l'obbiettivo di sicurezza che genera il requisito che le azioni di un'entità siano riconducibili in modo univoco a tale entità. Ciò supporta il *non ripudio*, la deterrenza, l'isolamento dei guasti, il rilevamento e la prevenzione delle intrusioni, il ripristino post-azione 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.

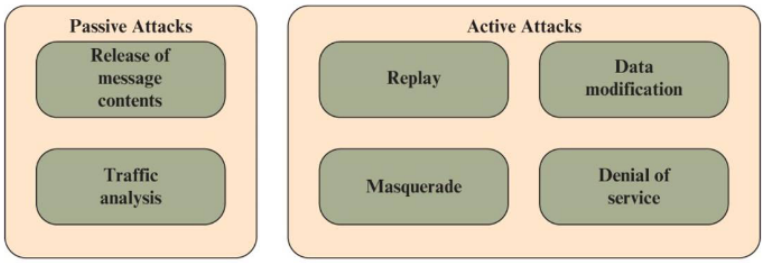
1.2 Architettura OSI di sicurezza

- **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.

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.


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



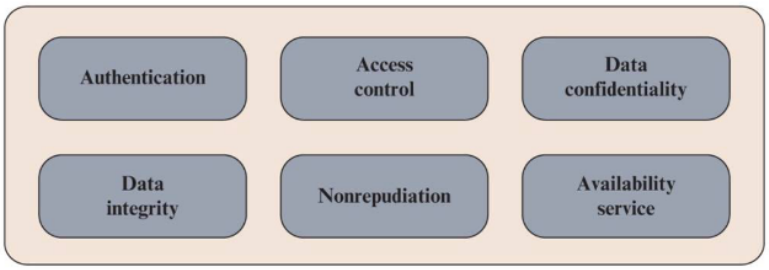
(a) Attacks

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 id 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 une 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 garantiche 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, o in determinati momenti durante la fase di trasferimento dei dati di 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 nono 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.

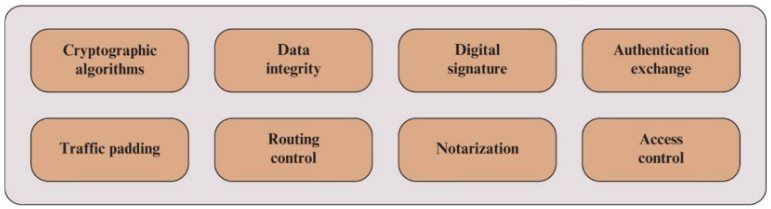
Integrità dei dati Può applicarsi a un flusso di messaggi, a un singolo messaggio o a campi selezionati all'interno di un messaggio.

- **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 si 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

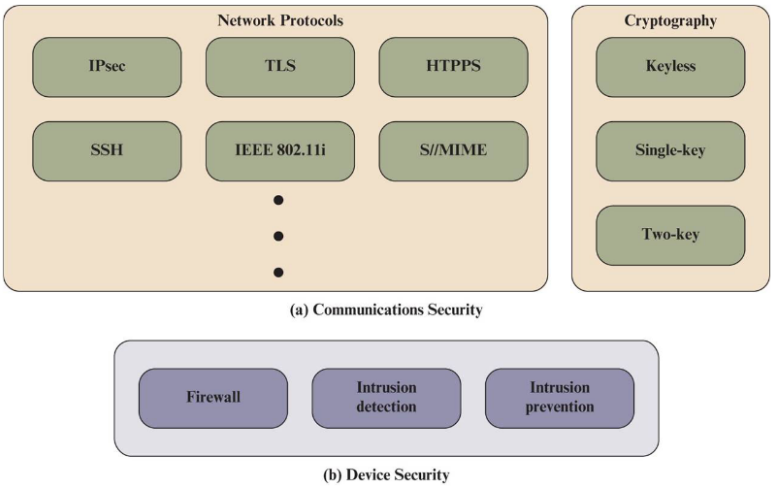
Sono algoritmi di crittografia che usano una coppia di chiavi.

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.

Chrosen 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

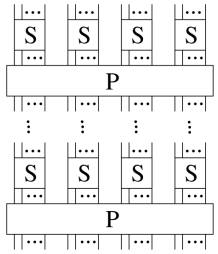
Uno schema di crittografia e considerato sicuro dal punto di vista computazionale se il testo cifrato generato dallo schema soddisfa uno o entrambi i seguenti criteri:

- 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.

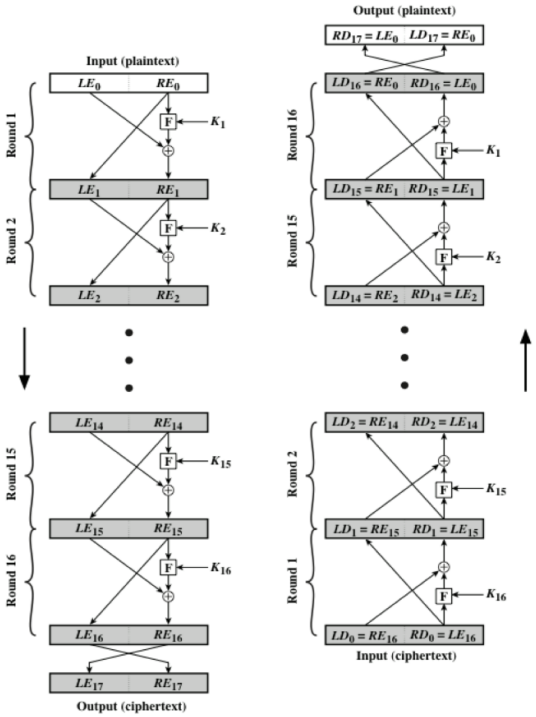


Operazione	Box	Effetto
Sostituzione	S-box	Confusione: rende la relazione statistica tra il testo cifrato e la chiave il più complessa possibile
Permutazione	P-box	Diffusione: dissipa la struttura statistica del testo in chiaro su gran parte del testo cifrato

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).

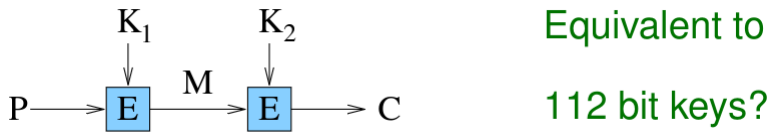
La struttura del DES è una leggera variazione della rete Feistel. Ci sono 16 round di elaborazione, il testo in chiaro è lungo 64 bit (bit di input), la chiave è lunga 56 bit.

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**



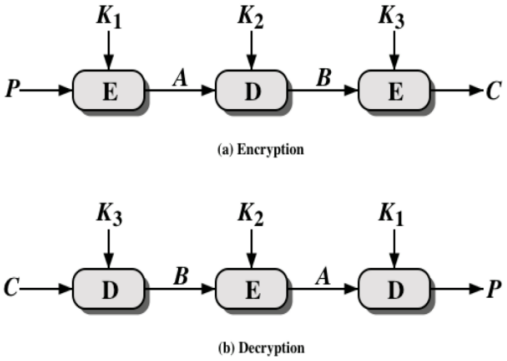
Attacco plaintext contro il 2DES (meet-in-the-middle) Per $C = E_{K_2}(E_{K_1}(P))$ si pone $X = E_{K_1}(P) = D_{K_2}(C)$. Dato un P e un C noti, crittografare P per tutte le 2^{56} possibili K_1. Memorizzare in una tabella ordinata per X. Decrittare C con tutte le 2^{56} possibili K_2 e cercare una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

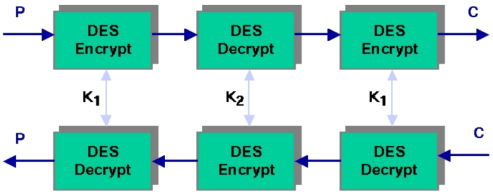
3DES–Triple DES Utilizza tre fasi di crittografia invece di due. La compatibilità viene mantenuta con il DES standard (si pone $K_2 = K_1$). **Vantiaggi:** la lunghezza della chiave di 168 bit supera la vulnerabilità agli attacchi di forza bruta del DES e l'algoritmo di crittografia sottostante è lo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



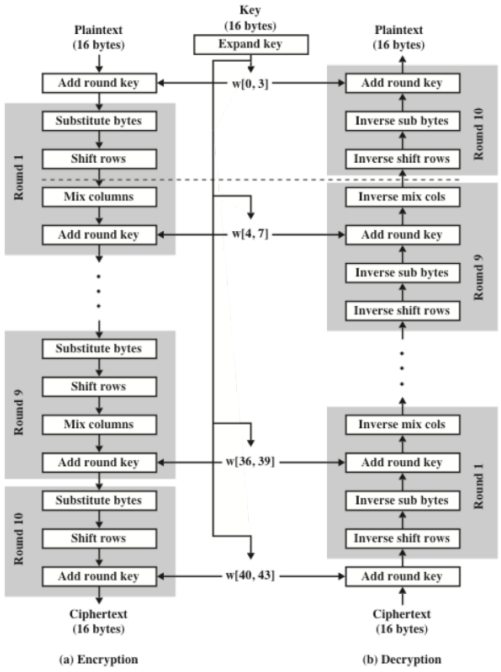


- i criteri di valutazione includevano sicurezza, efficienza computazionale, requisiti di memoria, idoneità hardware e software, e flessibilità.

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.

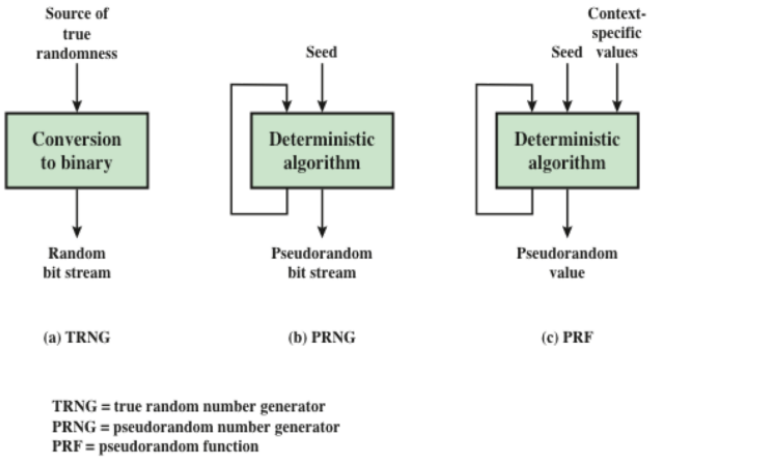
In diversi scenari della distribuzione delle chiavi , come Kerberos, i numeri casuali vengono utilizzati per l'handshake per prevenire attacchi di replay.

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.



- 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.

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 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 si 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.

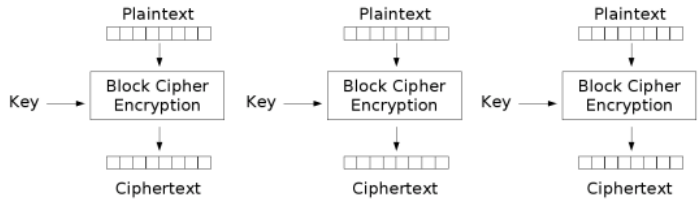
2.6 Modalità di operazione dei cifrari a blocco

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



Electronic Codebook (ECB) mode encryption

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.

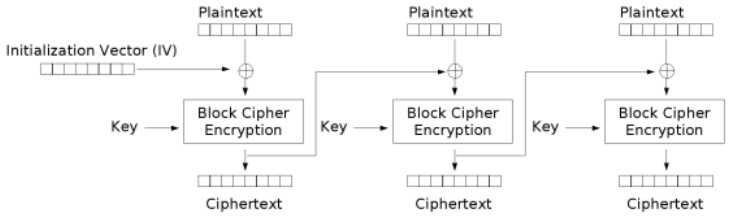
L'ECB presenta i seguenti vantaggi:

- cifratura parallelizzabile;
- 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,

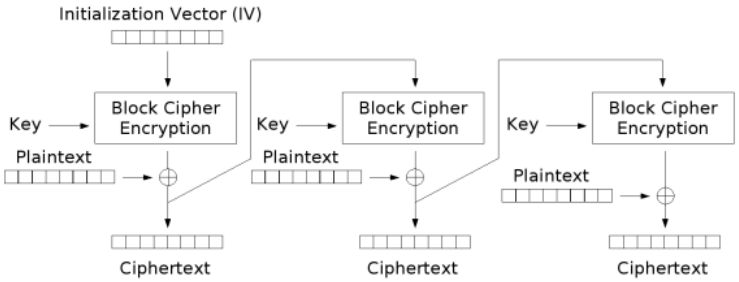


Cipher Block Chaining (CBC) mode encryption

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, perchéé se $S_S(x)$ sono gli s bit più significativi di X, allora:

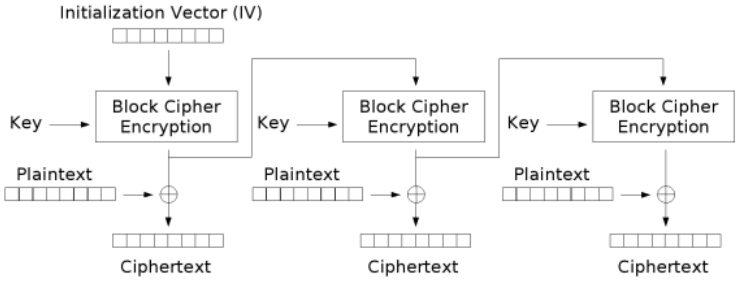
$$C_1 = P_1 \oplus S_S(E_K(IV))$$

Pertanto

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

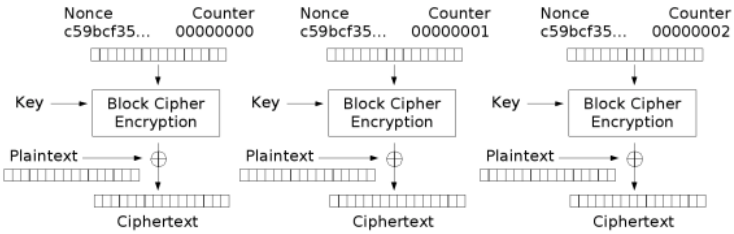
e lo stesso ragionamento vale per i passi successivi.

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.



Output Feedback (OFB) mode encryption

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 in chiaro. Il requisito essenziale è che il suo valore sia differente per ciascun blocco da cifrare;



Counter (CTR) mode encryption

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.

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

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 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;

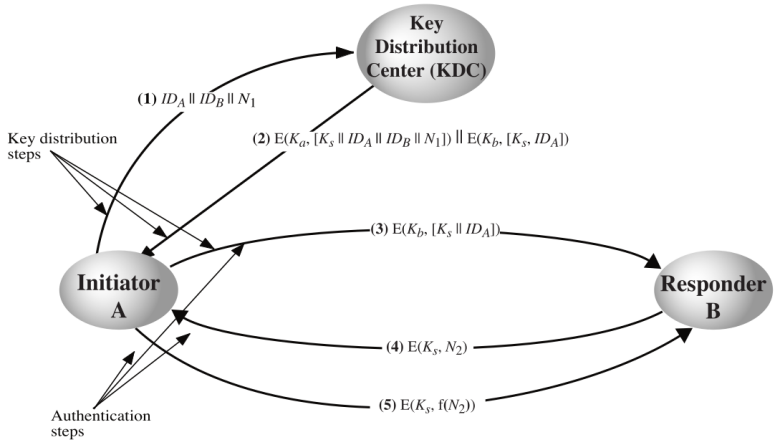
  - richiede la consegna manuale della chiave ma è inefficace per la cifratura end-to-end a causa della necessità di distribuire un gran numero di chiavi dinamicamente in sistemi distribuiti;

- 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 tuta 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

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

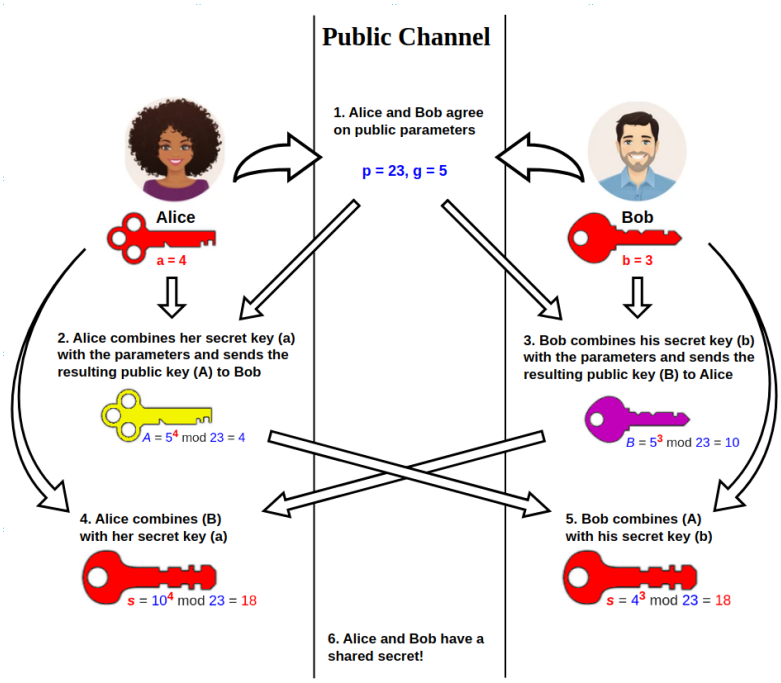
- calcola d tale che:

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

- 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.

t

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:

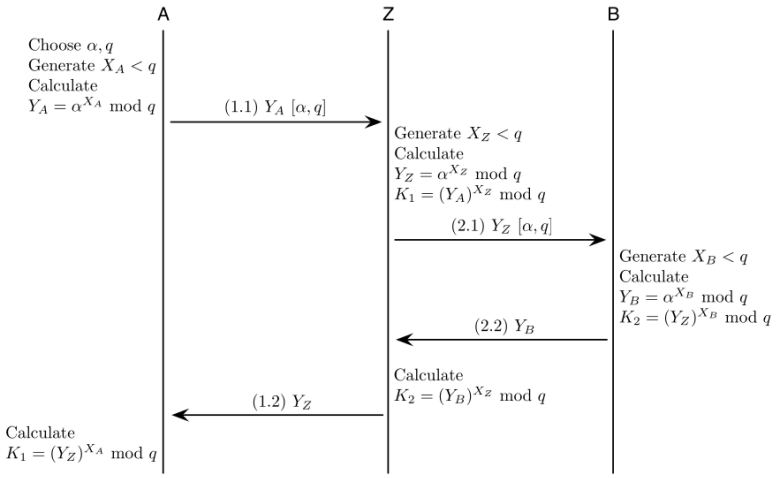
$$K_A = (g^b)^a \mod p = g^{ba} \mod p$$
$$K_B = (g^a)^b \mod p = g^{ab} \mod p$$

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-Hellma 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 spresos 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.
     - 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.

[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.

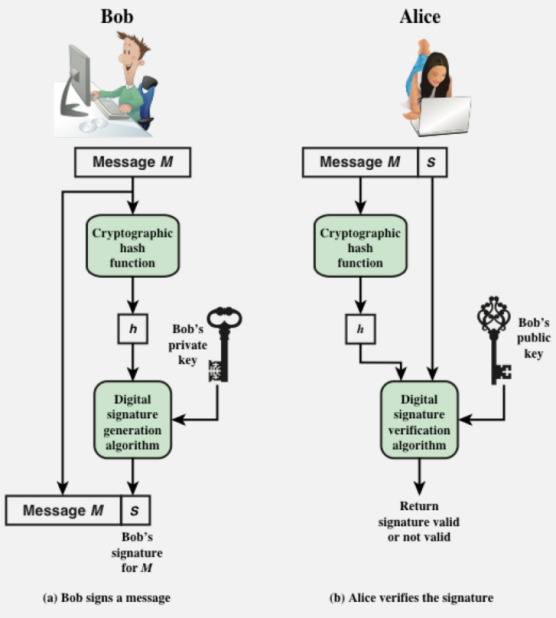
Evoluzione dello standard Le versioni successive come FIPS 186-3 (2009) e FIPS 186-4 (2013) hanno introdotto l'uso di SHA-2 (SHA-224, SHA-256, SHA-384 e SHA-512) al posto di SHA-1 per una maggiore sicurezza. Il più recente standard, FIPS 186-5 (2023), ha ritirato l'uso di DSA per le nuove implementazioni 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 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

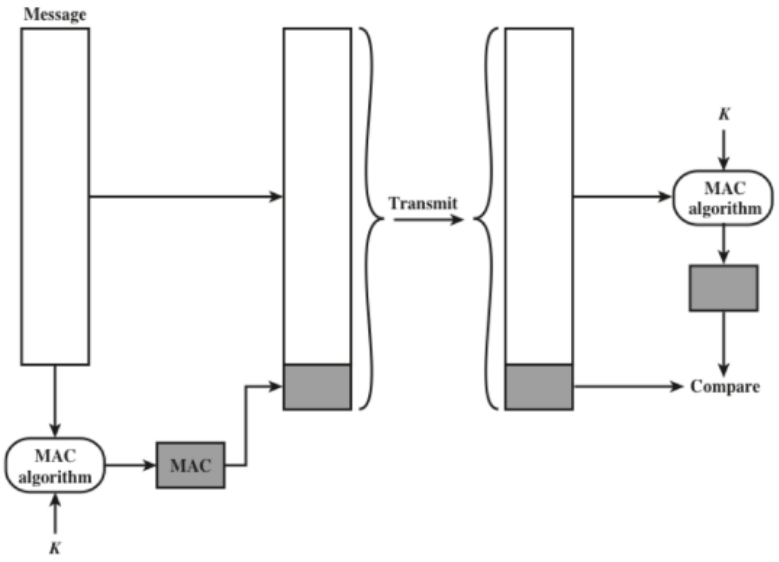
- **Utilizzando la crittografia convenzionale:** la crittografia simmetrica da sola non è uno strumento adatto per l'autenticazione dei dati. Si assume che solo che il mittente e il destinatario condividano una chiave, quindi solo il mittente genuino sarebbe in grado di crittografare un messaggio con 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 soddisfare 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 ri 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).

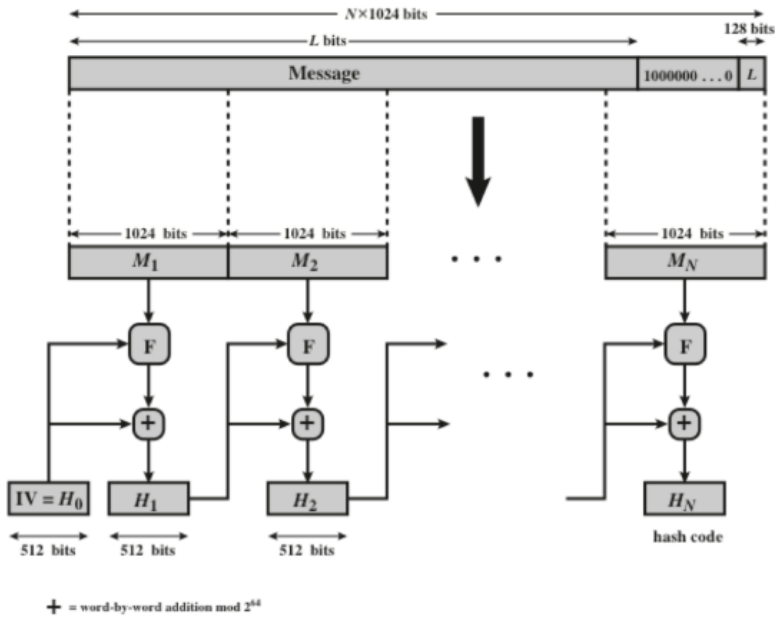
SHA-1,SHA-2,SHA-3 SHA (Secure Hash Standard) è stata sviluppata dal NIST ed è stata pubblicata come standard federale di elaborazione delle informazioni (FIPS 180) nel 1993. È stata revisionata nel 1995 come SHA-1 e pubblicata come FIPS 180-1. Si basa sulla funzione hash MD4 e il suo design modella strettamente 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]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.

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 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.



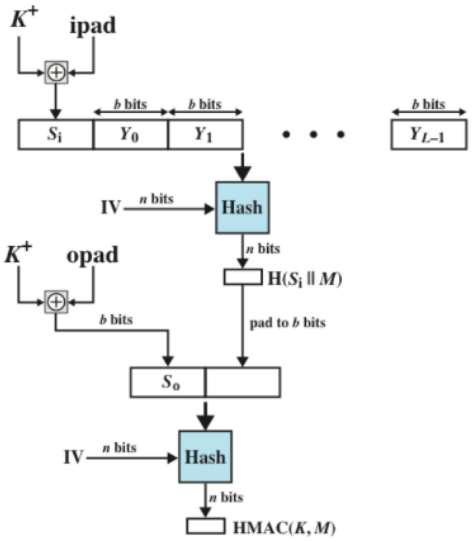
Nel 2001 il NIST pubblicò quattro funzioni di *hash* addizionali facenti parti della famiglia SHA, ognuna con un digest più lungo di quello originale, collettivamente denominate SHA-2. Queste varianti sono note con la lunghezza in bit del digest generato a seguire la sigla ufficiale dell'hash. Gli algoritmi 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.



HMAC(*K*, *M*) = H[(*K*+ ⊕ opad) || H[(*K*+ ⊕ 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

Come funziona La formula generale di HMAC è:

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

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.

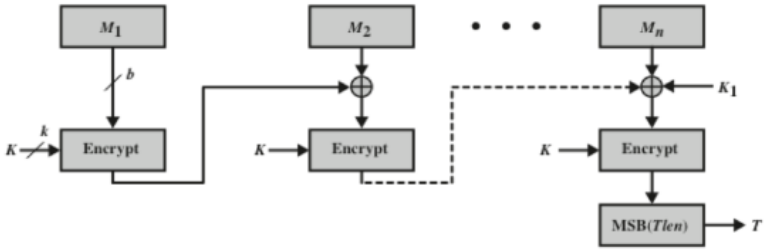
HMAC è ampiamente utilizzato in protocolli come IPsec, TLS (per la protezione dei dati), e nella generazione di token di sicurezza come i JSON Web Tokens (JWT).

4.1.4 CMAC (Cipher-based Message Authentication Code)

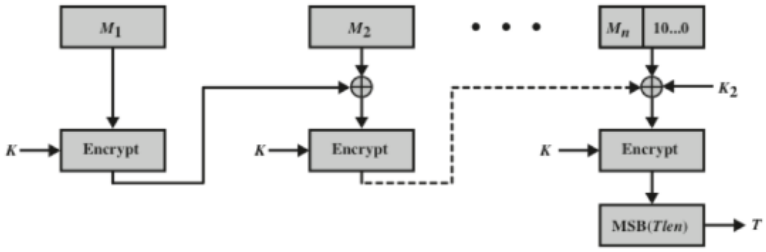
Il CMAC è uno schema specifico per generare un Message Authentication Code (MAC) utilizzando un cifrario a blocchi simmetrico (come AES) in modalità CBC (Cipher Block Chaining). È stato sviluppato per superare le debolezze di sicurezza e le complessità legali (legate ai brevetti) del suo predecessore,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



(a) Message length is integer multiple of block size



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

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;
  - 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