Lecture: concetti base di crittografia

Riccardo Torre

1 dicembre 2025

1 Lecture 13/05/2025

Le reti di sostituzione-permutazione di Claude Shannon Nel 1949 Claude Shannon introdusse l'idea di reti di permutazione-sostituzione (S-P) che sono basate su due operazioni crittografiche primitive ossia le sostituzioni con le S-box che sostituiscono i simboli di input confondendo il contenuto del messaggio e le permutazioni con le P-box che permettono di diffondere i bit tra le S-box dissipando la struttura statistica del contenuto del messaggio. I cifrari che hanno questa struttura si dicono **cifrari a prodotto**.

Feistel Nel 1973 Feistel crea il cifrario omonimo che si basa sull'idea di rete S-P proposta da Shannon. Ad ogni **round** si prende un blocco, lo si divide a metà. La parte di destra viene passata ad una funzione di round (funzione di Feistel) che prende in input oltre alla parte destra una chiave. La parte di sinistra viene messa in XOR con il risultato della funzione di Feistel. Successivamente la parte sinistra e destra vengono scambiate. Vengono fatti 16 round. Dopo il 16 esimo round (17°esimo passaggio) vengono scambiate le due metà. Se nell'operazione di cifratura vengono usate le chiavi nell'ordine $\mathrm{K}_1, \ldots, \mathrm{K}_{16}$, e al 17°esimo passaggio viene fatto lo scambio delle due metà del blocco, nella decifratura i passaggi devono essere svolti nell'ordine inverso, ovvero, si scambiano le due metà, si applicano i round usando le chiavi nell'ordine $\mathrm{K}_{16}, \ldots, \mathrm{K}_1$.

DES È l'implementazione del cifrario di Feistel, Data Encryption Standard. È stato attaccato per il rapido avanzamento tecnologico, non perché non fosse sicuro. Si decise di utilizzare una chiave da 56 bit perché la computazione con chiavi più lunghe per la tecnologia dell'epoca era abbastanza onerosa. Nel 1999 DES è stato attaccato con un brute force attack. Ci sono dei sospetti che il NSA avesse modificato le S-box introducendo una backdoor.

DDES È suscettibile ad attacco Meet-In-the-Middle. È un attacco **chosen plaintext**:

$$X = E_{Ki}(P) \xrightarrow{M} E_{Kj}(M) = C = E_{Kj}(E_{Ki}(P)) \qquad i, j \in \left[0; 2^{56} - 1\right]$$

1. Scegli un plaintext P, e applica l'algoritmo DDES per ottenere il relativo ciphertext C. Crea una tabella ordinata per X.
2. Decifra C con tutte le 2^{56} possibili chiavi k_j e verifica se $X = E_{Ki}(P) = D_{Kj}(C)$ (condizione Meet in the Middle).
3. Ciascuna corrispondenza della suddetta condizione è una soluzione candidata. Usare la stessa coppia di chiavi $(\mathrm{K}_i, \mathrm{K}_j)$ con altre coppie (P, C).

L'attacco ha una complessità spaziale e temporale di $O(2^{56})$. Quindi la chiave non ha una dimensione doppia di 112 bit.

Triple DES DES viene applicato in catena con tre chiavi diverse

Encryption $P \rightarrow \boxed{E_{K1}(P)} \xrightarrow{A} \boxed{D_{K2}(A)} \xrightarrow{B} \boxed{E_{K3}(B)} \rightarrow C$

Decryption $C \rightarrow \boxed{D_{K3}(C)} \xrightarrow{B} \boxed{E_{K2}(B)} \xrightarrow{A} \boxed{D_{K1}(A)} \rightarrow P$

o usando due chiavi

Encryption $P \rightarrow \boxed{E_{K1}(P)} \xrightarrow{A} \boxed{D_{K2}(A)} \xrightarrow{B} \boxed{E_{K1}(B)} \rightarrow C$

Decryption $C \rightarrow \boxed{D_{K1}(C)} \xrightarrow{B} \boxed{E_{K2}(B)} \xrightarrow{A} \boxed{D_{K1}(A)} \rightarrow P$

in questo modo, TDES è compatibile con lo standard DES ($\mathrm{K}_2 = \mathrm{K}_1$). Tuttora non si conosce un attacco pratico a TDES. Un brute-force attack potrebbe richiedere fino a 2^{112} operazioni.

AES Advanced Encryption Standard nato per sostituire TDES. È un cifrario a blocco con blocchi di lunghezza di 128 bit e supporta chiavi di 128, 192 e 256 bit. Non è una struttura di Feistel ma supporta le idee di Shannon di sostituizione e permutazione.

TRNG True Random Generator costruisce un flusso di bit casuale a partire da una sorgente casuale come i movimenti del mouse, la testina sul disco rigido, un qualche segnale. Quindi avviene semplicemente una conversione in digitale del segnale in ingresso.

PRNG È un algoritmo deterministico che usa il seme e una parte di elaborazione intermedia per generare un flusso pseudocasuale (è apparentemente casuale, ovvero distinguere un flusso di bit generato da un TRNG, o in generale da una sequenza veramente casuale di bit da un flusso generato da un PRNG è estremamente difficile).

PRF Le Pseudo Random Functions sono algoritmi deterministici che prendono in ingresso, un seme, una parte di elaborazione intermedia e valori specifici del contesto e restituiscono un valore pseudocasuale.

Message Authentication Code È un tag che permette di verificare l'integrità di un messaggio. Per autenticare un messaggio si potrebbe utilizzare la crittografia simmetrica inserendo nel messaggio un **sequence number** per evitare il riordinamento e un **error detection code** per verificare che il messaggio non sia stato alterato. Il **timestamp** può essere messo per proteggere dai **replay attack**. Oppure il messaggio potrebbe non essere crittografato (non viene garantita la confidenzialità). Si aggiunge un MAC. Per creare i MAC si usano gli algoritmi MAC (spesso funzioni hash). Il MAC viene aggiunto al messaggio. Chi riceve il messaggio utilizzerà lo stesso algoritmo MAC per calcolarlo sul messaggio e confrontarlo con il MAC ricevuto.

Funzioni hash one-way È una funzione che preso un messaggio restituiscono un *digest* di taglia fissata. Le funzioni hash in generale non prendono chiavi segrete in input. Ci sono tre modi per aggiungere il MAC al messaggio:

1. $C = E_K([M \parallel \mathrm{H}(M)]) \xrightarrow{invio} D_K(C)$ da cui si ricava $\mathrm{H}(M)$ che viene confrontato con il calcolo dell'hash su M.
2. $C = \mathrm{E}(\mathrm{K}_{PU}, [M \parallel \mathrm{H}(M)]) \xrightarrow{invio} \mathrm{D}(\mathrm{K}_{PR}, C)$ da cui si ricava $\mathrm{H}(M)$ che viene confrontato con il calcolo dell'hash su M.

3. Si concatena al messaggio uno shared secret S in modo da ottenere $[S \parallel M]$ e si calcola $\mathrm{H}([S \parallel M])$. Si invia il messaggio nella forma $[M \parallel \mathrm{H}([S \parallel M])]$. Il ricevente estrae il messaggio M, gli applica lo shared secret S, calcola $\mathrm{H}(S \parallel M)$ e lo confronta con quello ricevuto.

In generale, le funzioni **one-way** sono facili da calcolare ma difficili da invertire, ovvero esistono algoritmi che possono calcolare la funzione $f(x)$ in tempo polinomiale nella taglia dell'input ma anche che nessun algoritmo probabilisticamente polinomiale può calcolare una controimmagine di $f(x)$ a meno di una probabilità trascurabile, quando x viene scelta in modo casuale.

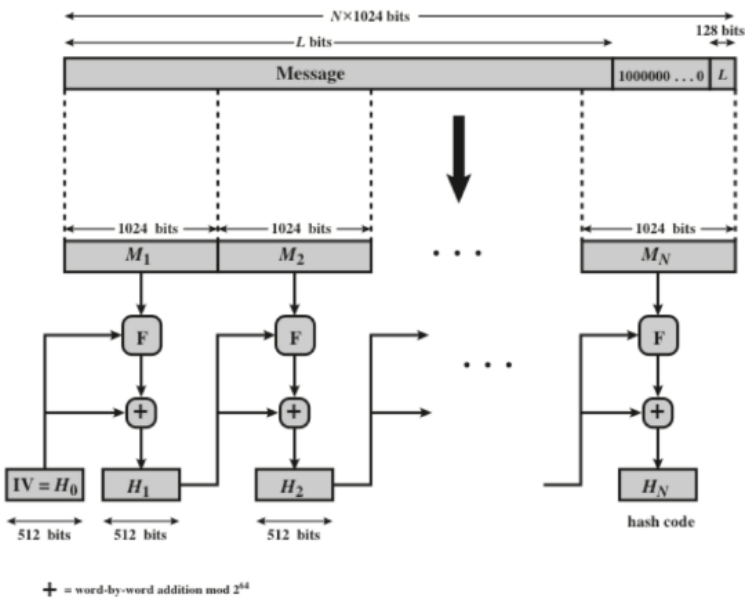
Funzioni hash sicure Una funzione hash è caratterizzata da uno dei livelli di sicurezza:

- **sicurezza debole:** dato un messaggio M, è computazionalmente "difficile" trovare un secondo messaggio M' tale che $\mathrm{H}(M) = \mathrm{H}(M')$;
- **sicurezza forte:** è computazionalmente "difficile" trovare una coppia di messaggi (M, M') tali che $\mathrm{H}(M) = \mathrm{H}(M')$;

e le seguenti proprietà:

- **resistenza alla preimmagine:** sia computazionalmente intrattabile la ricerca di una stringa in input che dia un hash uguale a un dato hash;
- **resistenza alla seconda preimmagine:** sia computazionalmente intrattabile la ricerca di una stringa in input che dia un hash uguale a quello di una data stringa;
- **resistenza alle collisioni:** sia computazionalmente intrattabile la ricerca di una coppia di stringhe in input che diano lo stesso hash.

SHA Vengono create con un meccanismo di coodifica a blocchi e concatenati come nella CBC (Cipher Block Chain). Il messaggio viene spezzato in blocchi della stessa dimensione. Più grande



è un hash, meglio è: dato che i testi possibili, con dimensione finita maggiore dell'hash, sono più degli hash possibili, per il principio dei cassetti ad almeno un hash corrisponderanno più testi possibili. Più grande è il blocco più grande è l'insieme degli hash possibili, perché è più grande la rappresentazione che codifica un codice hash.

	SHA-1	**SHA-224**	**SHA-256**	**SHA-384**	**SHA-512**
Message digest size	160	224	256	384	512
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	512	1024	1024
Word size	32	32	32	64	64
Number of steps	80	64	64	80	80

HMAC Si basa sul prendere una funzione hash (che hanno la caratteristica di essere veloci) e renderla crittografica attraverso l'utilizzo di una chiave. È utilizzato in IP Security, in TLS e in SET[1]. Ha un meccanismo che permette di scegliere la funzione hash da utilizzare come se fosse un plugin. È semplice rimpiazzare una funzione hash con un'altra. HMAC non altera le performance delle funzioni hash sottostanti.

$$\text{HMAC}(K, M) = \text{H}((K^+ \oplus \textit{opad}) \parallel \text{H}((K^+ \oplus \textit{ipad}) \parallel M))$$

HMAC permette di fare un hash crittografico che data una chiave e un messaggio, restituisce l'hash crittografico. Dapprima viene fatto il padding su K con degli zeri e viene indicata con K^+. In questo modo la lunghezza della chiave sarà sempre la stessa (quella desiderata). I parametri *ipad* e *opad* rappresentano rispettivamente i valori esadecimali 36 e 5C. Con $\parallel$ si indica la giustapposizione degli elementi. Le funzioni hash vengono annidate. La chiave è argomento delle funzioni di entrambe le funzioni hash che sono annidate.

CMAC Si prende il messaggio, lo suddivide in blocchi della stessa dimensione, ciascuno dei quali viene messo in XOR con il risultato precedente (tranne per il primo blocco) con il blocco sovrastante per essere poi passato ad un algoritmo di crittografia simmetrica come AES o TDES. Non è parallelizzabile. È più pesante rispetto ad HMAC.

Diffie-Hellman e l'idea delle chiavi pubbliche e private Al tempo non esisteva un meccanismo per distribuire le chiavi simmetriche e come fare le firme digitali. Diffie e Hellman proposero un'idea di algoritmo (**non un algoritmo!**) per fare la crittografia asimmetrica con due chiavi separate. In generale, più lunga è una chiave, più è sicura.

Algoritmo Diffie-Hellman per lo scambio di chiavi Permette di scambiare le chiavi segrete in modo sicuro (fig. 1) basando tale meccanismo sul calcolo dei logaritmi discreti. È computazionalmente difficile calcolare il logaritmo discreto (calcolare l'esponente conoscendo la base e il numero dietro al modulo). DH così come è stato proposto non è sicuro; è vulnerabile ad attacco man-in-the-middle. Questo perché manca autenticazione nel protocollo. Più volte a lezione si indica con "mezza chiave" quella che qui si indica con "chiave pubblica".

[1] Secure Eletronic Transaction.

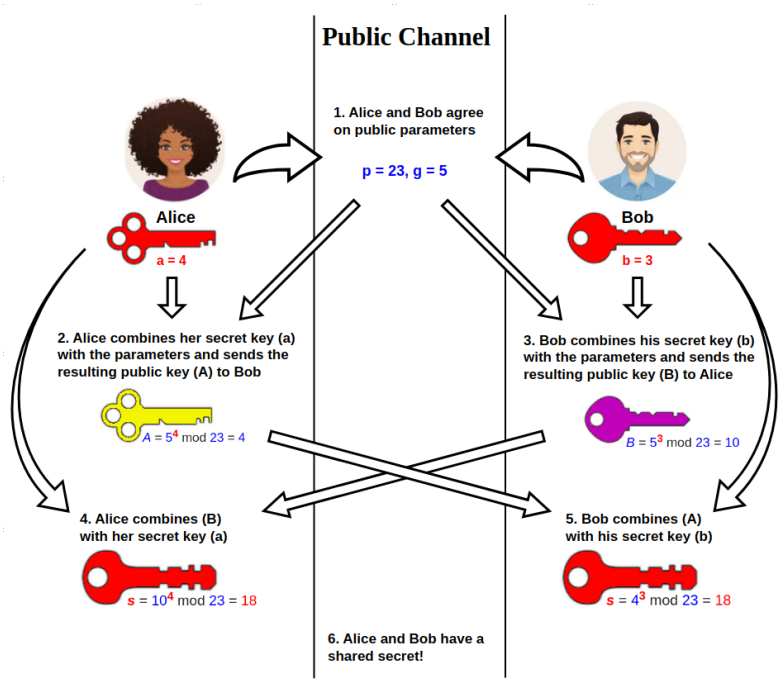


Figura 1: algoritmo Diffie Hellman.

Uso sistemi crittografici a chiave pubblica Usati per:

1. **crittografare/decrittografare:** il mittente cifra un messaggio con la chiave pubblica del destinatario;
2. **firma digitale:** il mittente firma un messaggio con la propria chiave privata;
3. **scambio di chiavi:** due parti cooperano per scambiare una chiave di sessione.

RSA Primo algoritmo ad implementare l'idea di Diffie e Hellman di chiave pubblica e chiave privata.