

Lecture: concetti base di crittografia e public key infrastructures

Riccardo Torre

3 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 K_1, \dots, 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 K_{16}, \dots, 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.

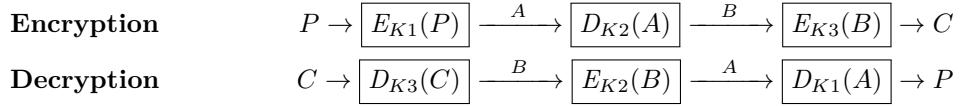
DDES È suscettibile ad attacco Meet-In-the-Middle. È un attacco **chosen plaintext**:

$$X = E_{K_i}(P) \xrightarrow{M} E_{K_j}(M) = C = E_{K_j}(E_{K_i}(P)) \quad i, j \in [0; 2^{56} - 1]$$

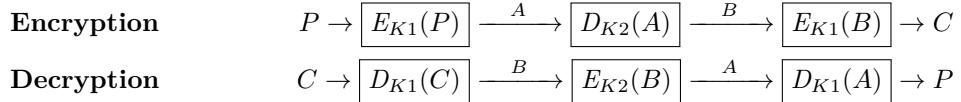
1. Scegli un plaintext P , una coppia di chiavi K_i, K_j e applica l'algoritmo DDES per ottenere il relativo ciphertext C . Crea una tabella ordinata per X .
2. Decifra C con tutte le 2^{56} possibili chiavi k_j e verifica se $X = E_{K_i}(P) = D_{K_j}(C)$ (condizione Meet in the Middle).
3. Ciascuna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



o usando due chiavi



in questo modo, TDES è compatibile con lo standard DES ($K_2 = K_1$). 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 sostituzione 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. $C = E_K([M \parallel H(M)]) \xrightarrow{\text{invio}} D_K(C)$ da cui si ricava $H(M)$ che viene confrontato con il calcolo dell'hash su M.

2. $C = E(K_{PU}, [M \parallel H(M)])$ $\xrightarrow{\text{invio}}$ $D(K_{PR}, C)$ da cui si ricava $H(M)$ 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 $H([S \parallel M])$. Si invia il messaggio nella forma $[M \parallel H([S \parallel M])]$. Il ricevente estrae il messaggio M , gli applica lo shared secret S , calcola $H(S \parallel M)$ 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 $H(M) = H(M')$;
- **sicurezza forte:** è computazionalmente “difficile” trovare una coppia di messaggi (M, M') tali che $H(M) = H(M')$;

e le seguenti proprietà:

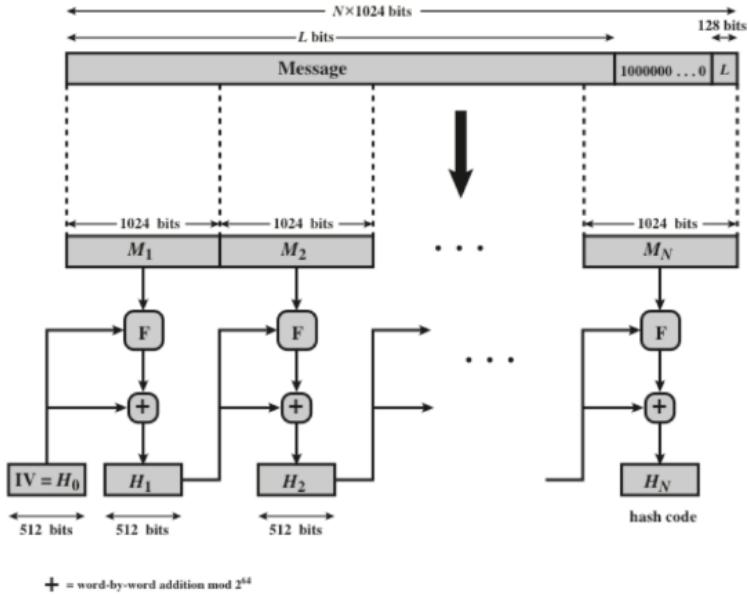
- **resistenza alla preimmagine:** sia computazionalmente intrattabile la ricerca di una stringa in input che dia un hash uguale a un dato hash;
- **resistenza alla seconda preimmagine:** sia computazionalmente intrattabile la ricerca di una stringa in input che dia un hash uguale a quello di una data 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 codifica 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 SET¹. Ha un meccanismo che permette di scegliere la funzione hash da utilizzare come se

¹Secure Eletronic Transaction.



fosse un plugin. È semplice rimpiazzare una funzione hash con un'altra. HMAC non altera le performance delle funzioni hash sottostanti.

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

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".

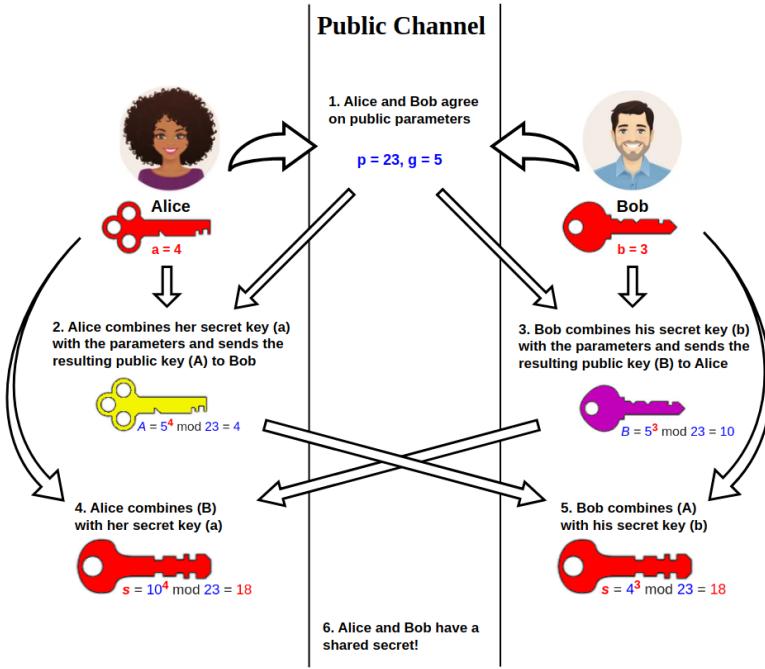


Figura 1: algoritmo Diffie Hellman.

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 crittosistema a chiave pubblica - chiave privata del 1976. Creato da Rivest, Shamir e Adleman e pubblicato nel 1997. RSA si basa sulla difficoltà di fattorizzare due numeri primi molto grandi (**problema della fattorizzazione**).

Generazione delle chiavi in RSA

1. Scegli due numeri primi p e q , $p \neq q$.
2. Computa $n = pq$.
3. Computa $\varphi(n) = (p - 1)(q - 1)$.
4. Seleziona un intero e tale che $\gcd(\varphi(n), e) = 1$ e $1 < e < \varphi(n)$.
5. Calcola d tale che $de \bmod \varphi(n) = 1$
6. Calcola la chiave pubblica come $K_{pu} = \{e, n\}$

7. Calcola la chiave privata come $K_{pr} = \{d, n\}$

Un attaccante può invalidare tutto se riesce a scoprire la fattorizzazione di n .

Cifratura in RSA Il plaintext è un $M < n$. Il ciphertext si ottiene come $C = M^e \pmod{n}$.

Decifratura in RSA Il ciphertext è C . Il plaintext si ottiene come $M = C^d \pmod{n}$.

Digital Signature Standard (DSS) Vengono usati esclusivamente per fare le firme digitali (non l'encryption). Devono essere in grado di garantire l'autenticazione dell'origine, la data integrity e la non repudiation. Si possono usare gli algoritmi *Digital Signature Algorithm* (DSA), RSA o *Elliptic Curve Digital Signature* (ECDSA). Le curve ellittiche sono estremamente efficienti nell'uso della chiave perché bastano delle chiavi piccole per ottenere una crittografia forte. In generale, le firme digitali si prestano bene a lavorare sull'hash perché RSA deve lavorare con il padding per evitare brute-force. Nell'esempio a seguire, si indica con PR la chiave privata di Bob e con PU la chiave pubblica. Bob vuole inviare il messaggio M ad Alice:

1. calcola $H(M)$;
2. firma $H(M)$ con la chiave privata di Bob $DSA_{PR}(H(M))$;
3. Invia ad Alice $[M \parallel DSA_{PR}(H(M))]$

Alice riceve il messaggio $[M \parallel DSA_{PR}(H(M))]$

1. prendi M e calcola $H(M)$;
2. confronta $H(M)$ appena calcolato, con il valore estratto dal messaggio $[M \parallel DSA_{PR}(H(M))]$ a cui viene applicato l'algoritmo DSA con la chiave pubblica di Bob $DSA_{PU}(DSA_{PR}(H(M))) = H(M)$; se i valori sono uguali, il messaggio è integro e proviene da Bob.

Gestione e distribuzione delle chiavi crittografiche

2 Lecture 20/03/2025

Certificati Servono per contrastare le compromissioni delle chiavi. I certificati possono essere revocati dalle CA quando una chiave viene compromessa o è vicina alla scadenza. Vengono emessi dalle CA.

X.509 Si basa sulla crittografia asimmetrica. Viene tuttora raccomandato RSA come algoritmo crittografico, mentre per le funzioni di hash non ci sono direttive sull'utilizzo di una specifica funzione di hash (che funzione hash utilizzare è contenuta nelle informazioni che riguardano il certificato), quindi le CA utilizzano quella che ritengono più opportuna. Il certificato contiene:

- informazioni che identificano il proprietario del certificato;
- la chiave pubblica del proprietario;
- informazioni che identificano la CA;
- informazioni che riguardano il certificato (durata, utilizzo);

- hash del certificato firmato con la chiave privata della CA.

Un certificato per essere autentico deve essere giustapposto al suo hash firmato con la chiave privata della CA:

$$E_{K_{PR_{CA}}}(H(\text{certificato di Bob}))$$

Per verificare la validità del certificato, il client utilizza la chiave pubblica della CA per verificare che la firma sia autentica (si decifra con RSA) e si utilizza la funzione di hash specificata nelle informazioni del certificato per calcolare il suo hash e confrontarlo con quello giustapposto.

I certificati possono essere posti in una directory pubblica, altamente accessibile, e non protetta perché i certificati sono già intrinsecamente sicuri.

Perché devo fare l'hash del certificato? Non posso firmare direttamente il certificato?
No, in quanto un certificato può avere delle dimensioni rilevanti e dunque si preferisce cifrare il suo hash (che ha una dimensione contenuta in pochi bit, ad esempio sha-256 è lungo 256 bit).

Nonce Name used only ONCE. I nonce si chiamano **challenge**, ovvero un'entità A sfida B proponendo un valore. B deve ritornare lo stesso valore ad A.

Compromissione di una CA Se una CA è compromessa, tutto ciò che è certificato da quella CA è compromesso.

Modelli di trust

- **Trust diretto:** tutti gli utenti sono iscritti alla stessa CA e sono legati dallo stesso legame di trust a quella CA, dunque quegli utenti hanno un legame di trust reciproco e diretto.
- **Trust gerarchico:** quando due utenti appartengono a CA diverse; si risale l'albero di trust per arrivare ad una radice comune; supponendo che A sia certificato dalla CA X_1 e B dalla CA X_2 e che X_1 e X_2 si conoscono e sono legate da un legame di trust, si scrive $X_1 \ll X_2 \gg X_2 \ll B \gg$ per dire che B è certificato da X_2 e X_2 è certificato da X_1 e che $X_2 \ll X_1 \gg X_1 \ll A \gg$ per dire che X_2 certifica X_1 e X_1 certifica A – catena di trust inversa; **forward certificate** i certificati di X generati dalle altre CA; **reverse certificates** i certificati delle altre CA generati da X ; in altre parole forward dall'alto verso il basso nella catena e reverse dal basso verso l'alto.
- **Web of trust:** è un concetto utilizzato da PGP per stabilire l'autenticità dell'associazione chiave pubblica-utente. Un certificato non viene firmato da una CA ma da diverse entità che conoscono il possessore del certificato in una relazione di trust più o meno forte (livelli di trust). Qualsiasi utente può certificare il certificato di qualcun altro.