Sicurezza e Crittografia (Rev. 0.3)

Michele Schimd

25 settembre 2022

1 Il problema della sicurezza

L'accesso alla rete Internet, usata da miliardi di persone, oggi avviene principalmente utilizzando reti *wireless* le quali, per loro stessa natura, rendono visibili a tutti i dati scambiati. Basta utilizzare un software di *sniffing* della rete, quale Wireshark (vedi Figura 1), per poter osservare tutto il *traffico di rete*. Di conseguenza è sempre più necessario proteggere le comunicazioni adottando strumenti per *offuscare* i dati scambiati.

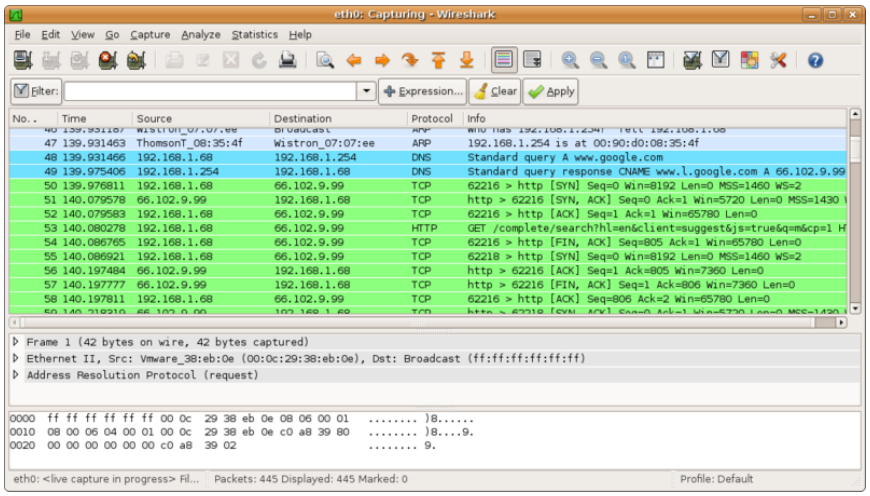

Figura 1: Il programma Wireshark per lo *sniffing* del traffico di rete.

1.1 La sigla CIAA

Possiamo schematizzare il problema della sicurezza nei sistemi di gestione delle informazione (ad esempio, nelle reti) con l'acronimo CIAA che

identifica quattro *requsiti* per la sicurezza:[1]

Confidentiality (*confidenzialità*) è la richiesta che i dati scambiati rimangano confidenziali, cioè noti soli a mittente e ricevente.

Integrity (*integrità*) è la richiesta che i dati che arrivano al ricevente siano gli stessi spediti dal mittente. Sotto questo requisito si trovano anche aspetti legati all'integrità del sistema (esempio dei router).

Availability (*disponibilità*) è la richiesta che i dati siano sempre disponibili a chi ha i diritti per consultarli. La disponibilità va intesa in senso ampio, se un server non è disponibile, nemmeno i dati che vi sono memorizzati sono disponibili.

Authority (*autorità*) è la richiesta che esista un meccanismo di verifica delle identità in modeo che mittente e ricevente siano proprio chi diachiarano di essere.

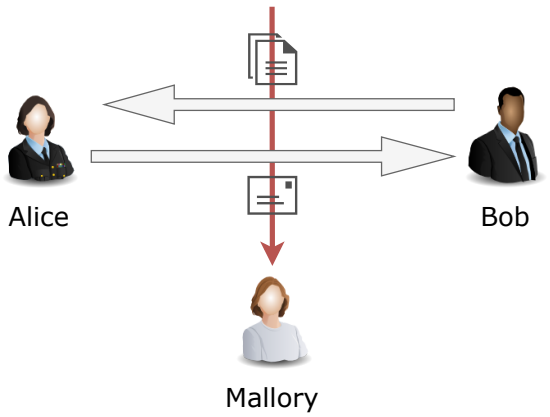

Figura 2: Scambio di messaggi tra Alice e Bob con Mallory che vuole sabotare la conersazione.

Per capire meglio i vari problemi che si presentano durante uno scambio di informazioni attraverso la rete, consideriamo la situazione in cui Alice e Bob vogliono scambiarsi delle informazioni, ma Mallory vuole sabotare e/o origliare la conversazione (vedi Figura 2).

[1]La sigla utilizzata più spesso è CIA (più facile da ricordare), che tuttavia non tiene conto del problema di *verifica dell'autorità* (*Authority*) di uno più dei componenti del sistema. Un ulteriore requisito, chiamato *accountability* cioè responsabilità, richiede che vi sia un responsabile (persona o organizzazione) per ogni operazione sui dati. Una discussione esaustiva della sigla CIA, e delle sue aggiunte, si trova nel libro di Stallings e Brown [3].

Violazione della confidenzialità: *eavesdropping* In questo caso Mallory è in grado di leggere i messaggi che Alice e Bob si scambiano. Questo è possibile se Mallory ha un dispositivo che accede alla stessa rete di Alice o di Bob. Utilizzando un programma come Wireshark, Mallory può catturare i pacchetti che Alice e Bob si stanno scambiando e, se questi non hanno *cifrato* la loro conversazione, Mallory sarà in grado di leggere il contenuto dei pacchetti, quindi della conversazione.

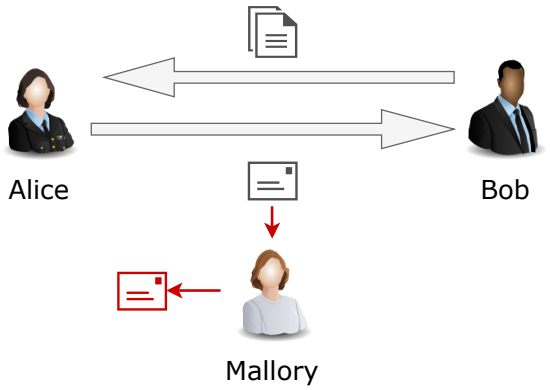

Figura 3: Mallory può origliare (*eavesdropping*) la conversazione tra Alice e Bob e può anche salvare delle copie dei messaggi scambiati.

Violazione dell'integrità: *tampering* In questo caso Mallory è in grado di modificare il contenuto di tutti o alcuni dei pacchetti che Alice e Bob si stanno scambiando. Questo è possibile, ad esempio, se Mallory ha modificato il codice di un *router* intermedio tra Alice e Bob ed è quindi in grado di sostituire i pacchetti originali con quelli finti. Questo non è ovviamente possibile se i pacchetti sono cifrati.

In questo caso è possibile che Mallory non sia in grado di decifrare il contenuto del messaggio perché cifrato, ma che possa comunque rendere insicura e compromessa la comunicazione. Mallory può ad esempio cambiare gli indirizzi IP dei pacchetti in modo che adesso lei sia il destinatario dei pacchetti di entrambi (*man-in-the-middle attcak*). Mallory può anche semplicemente farsi una copia dei pacchetti e magari tentare una loro decodifica *offline*, questo è utile se le informazioni scambiate tra Alice e Bob sono di valore anche dopo che la comunicazione è conclusa (ad esempio codici e numeri di una carta di credito).

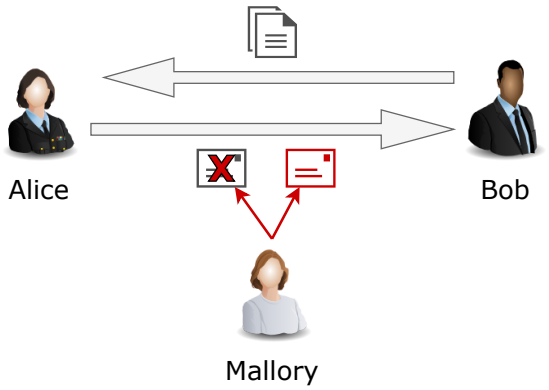

Figura 4: Mallory può sostituire un messaggio con una versione modificata (*tampering*).

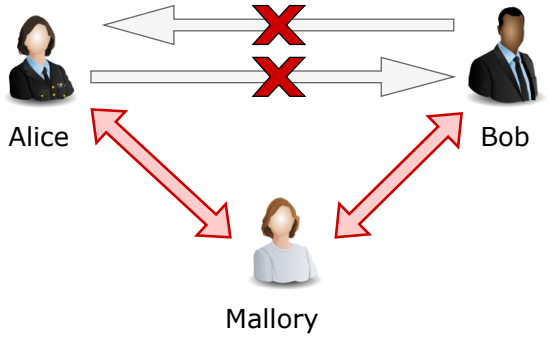

Figura 5: Mallory è in grado di frapporsi nella comunicazione tra Alice e Bob (*spoofing* e *man in the middle*).

Violazione dell'autorità: *spoofing* In questo caso Mallory riesce a impersonificare il ruolo di uno tra Alice e Bob. Questo è possibile, ad esempio, quando Mallory riesce a rubare le credenziali di uno dei due, oppure quando riesce a modificare il meccanismo di autorevolezza e/o autenticazione in uno dei dispositivi coinvolti nella comunicazione (vedi Figura 5).

Violazione della disponibilità: *Denial of Service (DoS)* In questo caso Mallory rende impossibile la comunicazione tra Alice e Bob. Questo è possibile, ad esempio, se Mallory può rendere inutilizzabile uno dei canali fisici di comunicazione. Nel caso di una comunicazione via cavo è possibile che Mallory interrompa fisicamente il collegamento, nel caso di connessioni wireless, Mallory può generare segnali di disturbo impedendo a Alice o Bob di utilizzare il mezzo fisico.

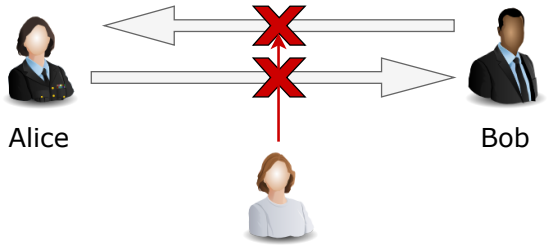

Figura 6: Mallory è in grado di interrompere la comunicazione tra Alice e Bob.

Un ulteriore (più frequente) tipo di attaco *Denial of Service* che Mallory può mettere in atto è l'attacco ai dispositivi di rete quali router o server. Tipicamente questi dispositivi vengono "bombardati" di richieste così da risultare sovraccarichi, quindi inutilizzatibili da Alice e Bob.

2 Algoritmi crittografici

Come è noto, un messaggio (anche se testo) viene interpretato da un computer come una sequenza di bit (cioè numeri). Più propriamente possiamo pensare ad un messaggio come ad una sequenza di numeri, uno per ogni carattere. Ad esempio la parola `Ciao` utilizzando la codifica ASCII è rappresentata dalla sequenza di numeri

```
103 151 141 111
```

Ad ogni numero n, possiamo pensare di applicare una **trasformazione** cioè una funzione che ne cambi il valore. Ad esempio, aggiungendo il valore 10 ad ognuno dei numeri sopra si otterrebbe la sequenza

```
113 161 151 121
```

che, tradotta in caratteri ASCII, diventa la parola `Msky`. Il processo di *offuscamento* del contenuto di un messaggio viene chiamato **cifratura** e degli algoritmi che fanno queste operazioni (ad esempio aggiungendo 10 al valore di ogni carattere) sono detti **algoritmi crittografici**.

Quindi un algoritmo crittografico è una *funzione* che ha come variabile un numero (o una sequenza di numeri) e produce un altro numero (o un'altra

sequenza di numeri). Nell'esempio sopra la funziona utilizzata è

$$f(x) = x + 10$$

per cui una sequenza di m numeri

$$x_1, x_2, \ldots, x_m$$

diventerà

$$x_1 + 10, x_2 + 10, \ldots, x_m + 10$$

In realtà la funzione usata finora permette solo di cifrare un messaggio, ma non di *decifrarlo*. Per questa operazione serve un secondo algoritmo che trasformi il *messaggio cifrato* (**ciphertext**) nel *messaggio in chiaro* (**palintext**). Nel nostro esempio questa funzione semplicemente prende il valore cifrato y e sottrae 10

$$g(y) = y - 10.$$

Ciò permette di ricostruire il messaggio originale ammesso che le funzioni f e g **siano una l'inversa dell'altra**. Matematicamente questo si traduce nel seguente fatto

$$g(f(x)) = x$$

cioè se calcolo $f(x)$ e poi uso il risultato di questo calcolo come argomento di g, allora ri-ottengo il valore x. Nella crittografia la funzione $f(x)$ prende il nome di *codifica* (**encryption**) e la funzione $g(y)$ prende il nome di *decodifica* (**decryption**). L'insieme delle funzioni di encryption e decrytpion viene spesso indicato con il termine di *schema di cifratura*. Gli algoritmi di encryption e decryption sono quegli algoritmi che "calcolano" le funzioni f e g, rispettavimante. Quindi, un algoritmo `Encrypt` avrà input x e ouput $f(x)$, mentre un algoritmo `Decrypt` avrà input y e output $g(y)$.

2.1 Chiave di cifratura

In tutti i casi pratici, gli algoritmi di encryption e decryption sono noti. Questo significa che la sicurezza dello schema di cifratura non è basato sul

fatto che non si può scrivere un programma che decifri un messaggio cifrato. Tuttavia ci deve essere un meccanismo che rende la comunicazione cifrata sicura, questo meccanismo è rappresentato dalla **chiave di cifratura**. La chiave di cifratura è un *parametro* dello schema di cifratura che **è noto solo ai partecipanti alla comunicazione cifrata**. Se la chiave viene "scoperta" da altri, la comunicazione non è più da considerarsi sicura e la chiave deve essere cambiata. Nell'esempio sopra la chiave di cifratura è il numero 10, questo numero infatti, è stato scelto "a caso" e non ha nessuna proprietà particolare. Qualsiasi altro numero k avrebbe dato un valido schema di cifratura definito delle funzioni

$$f(x) = x + k$$
$$g(y) = y - k$$

Come si vede da queste formule, le funzioni di encryption e decryption dipendono anche dal parametro k (cioè la chiave). Per questo motivo, spesso conviene indicare k esplicitamente usando $f(x, k)$ e $g(y, k)$ in modo da sottolineare il ruolo della chiave sia in f che in g.

Nella pratica gli schemi di cifratura funzionano bene in quanto **risulta difficile scoprire la chiave di cifratura k solo guardando i messaggi cifrati**. Ovviamente lo schema di cifratura che abbiamo usato come esempio non è particolarmente sicuro in quanto non è molto difficile scoprire la chiave k guardando ai messaggi cifrati.

2.2 Lo XOR per la cifratura

Ricordiamo la definizione dello **XOR** bit-a-bit tra due numeri $x = x_1 \dots x_m$ e $k = k_1 \dots k_m$ entrambe composti di m bit.

$$x \oplus k = (x_1 \oplus k_1), (x_2 \oplus k_2), \dots, (x_m \oplus k_m).$$

Ricordando la tabella di verità della funzione XOR

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

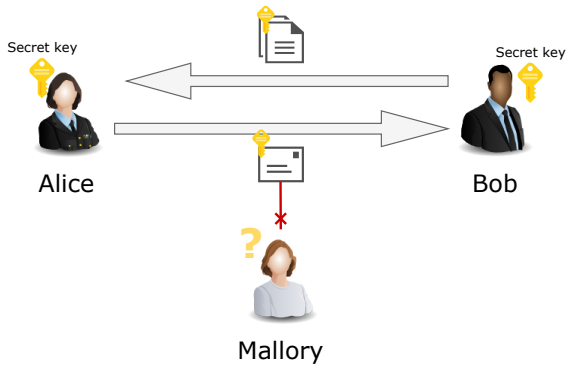

Figura 7: Alice e Bob usano una *chiave segreta* condivisa per cifrare tutti i messaggi che si scambiano. Mallory, non avendo la chiave, non è in grado di decifrare il contenuto del messaggio. Mallory, tuttavia, è ancora in grado di fare altre operazioni dannose, ad esempio può comunque memorizzare una copia del messaggio (magari un giorno scoprirà la chiave segreta) o interrompere la comunicazione con un attacco DoS.

si può facilmente dimostrare il seguente teorema.

Teorema 1. *Siano x e k due numeri interi con lo stesso numero di bit, allora*

$$(x \oplus k) \oplus k = x$$

Vulnerabilità di XOR Una delle maggiori vulnerabilità della funzione XOR è rappresentata proprio dalla proprietà enunciata nel Teorema 1. Infatti, essendo XOR commutativo, vale anche la seguente proprietà

$$(k \oplus x) \oplus x = k \tag{1}$$

che permette di "scoprire" la chiave in maniera relativamente semplice usando un testo x noto. Ad esempio, se identifico un pacchetto sulla rete di cui conosco il protocollo (magari lo "indovino" osservando le porte di comunicazione), è possibile che riesca a ricostruire il testo in chiaro x (es. `GET / HTTP/1.0`). A questo punto, conoscendo x e conoscendo $(k \oplus x)$, posso utilizzare la formula (1) per "scoprire" la chiave k.

2.3 Glossario dei concetti importanti

Cifratura Processo attraverso il quale il *messaggio in chiaro* (**plaintext**) viene *offuscato* e trasformato in un *messaggio cifrato* (**ciphertext**). Il processo inverso è la *decifratura* che permette di ri-ottenere il testo in chiaro a partire dal testo cifrato.

Encryption Funzione o algoritmo che permette di cifrare un messaggio.

Decryption Funzione o algoritmo che permette di decifrare un messaggio.

Schema di cifratura Insieme di meccanismi e/o algoritmi per l'encryption e il decryption di messaggi.

Chiave Parametro del meccanismo di cifratura che permette di avere una comunicazione sicura anche se lo schema è noto (ma non è nota la chiave).

Una raccolta estesa di definizioni utilizzate nell'ambito della sicurezza è disponibile nell'RFC4949 [2].

2.4 Esercizi

Esercizio 1: Cifrario di Giulio Cesare [4] La codifica presentata come esempio in questa sezione è noto come **Cifrario di Cesare** perché usato da Giulio Cesare. In realtà, l'imperatore di Roma usava un meccanismo di cifratura con chiave fissa $k = 3$. Come esercizio si chiede di realizzare un programma con un linguaggio di programmazione a scelta (es. Java) per creare il cifrario di Cesare utilizzando una chiave k qualsiasi tra 1 e 255. Il cifrario va poi applicato ai byte di un messaggio per cifrarlo prima e decifrarlo poi.
Attenzione Bisogna fare attenzione che i byte possono avere valori solo tra 0 e 255, quindi quando si fanno le somme $x + k$ (e anche le sottrazioni $y - k$) bisogna stare attenti a non superare il valore 255 oppure non andare sotto il valore 0. Lo studente deve trovare una soluzione a questo problema **garantendo che lo schema di cifratura sia invertibile** e cioè che sia sempre possibile tornare al messaggio in chiaro conoscendo messaggio cifrato e chiave di cifratura.

Esercizio 2: matematica e cifratura Lo studente proponga una coppia di funzioni matematiche $f(x)$ e $g(x)$ tali che l'una sia l'inversa dell'altra così da rappresentare un valido schema di cifratura. Lo schema proposto deve essere parametrizzato da una chiave k che deve essere un numero intero. Lo studente realizzi con un linguaggio di programmazione di propria scelta gli algoritmi di `Encrypt` a `Decrypt`.

Esercizio 3: cifratura a lunghezza fissa Gli schemi di cifratura utilizzati nella pratica devono avere dominio e codominio uguale. Ad esempio la cifratura di un blocco di 64 bit deve dare un altro blocco di 64 bit. Proporre un schema di cifratura basato su operazioni tra bit. Lo schema proposta deve essere invertibile e dipendere da una chiave k che si usi sia per cifrare sia per decifrare.

3 Le funzioni *hash*

Nel realizzare sistemi sicuri, oltre alle funzioni crittografiche viste nella sezione 2, sono molto usate le **funzioni di hash**. Normalmente queste funzioni trasformano un input di lunghezza qualsiasi (ad esempio un file o un messaggio) in una sequenza di bit a lunghezza fissa chiamata **digest**.

Un esempio di funzioni hash molto utilizzate nelle *tabelle di hash*, sono le funzioni *modulo*, cioè le funzioni che calcolano il resto di una divisione tra numeri interi. Consideriamo un numero intero m ed un numero intero n, sappiamo dal Teorema di Euclide (vedi [5]) che esistono due numeri interi q e r (chiamati *quoziente* e *resto*, rispettivamente) tali che

$$m = n \cdot q + r. \tag{2}$$

La proprietà importante di questa *divisione Euclidea* è che il resto r è sempre un numero intero compreso tra 0 e $n-1$. Per fare un esempio consideriamo il numero $m = 131$ e $n = 8$, il Teorema di Euclide ci dice che

$$131 = 8 \cdot 16 + 2$$

da cui abbiamo $q = 16$ e $r = 2$. Nelle tabelle hash la proprietà che $r \in [0, n-1]$ è utile perché permette di associare il numero m ad una posizione r nella tabella. Qualunque sia m sappiamo che ci basteranno n celle se associamo

ad m la cella r ottenuta dall'Equazione (2). Ovviamente è possibile che due numeri diversi abbiano lo stesso resto, ad esempio se $m = 155$ e $n = 8$, avremo $q = 19$ e $r = 3$, quando i due numeri 131 e 155 vengono *mappati* nella tabella di hash si dice che *collidono* sulla stessa cella corrispondente a $r = 3$. Una tabella di hash, quindi, deve prevedere un meccanismo per la gestione delle collisioni.

3.1 Funziona hash sicure

Quando ci si occupa di sicurezza, le funzioni di hash vengono usati in maniera diversa rispetto a quando si costruiscono tabelle di hash. Una funzione di hash $H(x)$ per algoritmi di sicurezza deve soddisfare le seguenti caratteristiche (vedi [3] p. 53)

1. H deve essere definita per ogni x, indipendente dalla dimensione.
2. H deve produrre un sequenza di bit a lunghezza fissa.
3. Esistono algoritmi *efficienti* per calcolare $H(x)$.
4. Non è praticabile utilizzare algoritmi che trovino x a partire da $H(x)$. In questo caso $H(x)$ si dice che è una funzione *one way*.
5. Non praticabile usare algoritmi che trovino due input x ed y che abbiano $H(x) = H(y)$. In questo caso si dice che $H(x)$ è una funzione *resistente alle collisioni*.

3.1.1 Esempi di funzioni hash sicure

Un classico esempio di funzione sicura, cioè che soddisfa ai requisiti sopra, è `MD5` (RFC 1321 [1]). Ad oggi tale algoritmo non è più ritenuto sicuro principalmente perché sono state proposte tecniche in grado di generare collisioni che rendono MD5 non più sicuro relativamente al punto 5 sopra [6].

Per questo motivo oggi sono da preferire altre funzioni hash sicure che prendono il nome di **Secure Hash Algorithm (SHA)** [8]. Vi sono diverse versioni di SHA che differiscono per il numero di bit del *codominio*, tipici valori per questo numero di bit sono $160, 256$ e 512.

Utilizzo delle funzioni hash per *block chain*

4 Crittografia a chiave simmetrica

Come suggerito dal nome, nella crittografia a **chiave simmetrica**, due entità che vogliono comunicare tra di loro devono possedere la stessa chiave che si usa sia per cifrare il messaggio in chiaro, sia per decifrare il messaggio cifrato. Ovviamente **chiunque possieda la chiave sarà in grado di decifrare le comunicazioni**, quindi è fondamentale che la chiave condivisa sia anche una chiave segreta. Il problema, quindi, è quello di *come scambiarsi le chiavi segreti necessarie per una comunicazione sicura?* Ci sono due modi per scambiarsi tali chiavi segrete

1. scambiarsele *offline* (cioè in modo "sicuro" rispetto agli attacchi della rete;

2. utilizzare meccanismo sicuro per lo scambio della chiave (es. basato su chiave pubblica/privata, descritto nella Sezione 5.3.

Ad esempio il protocollo *Transport Layer Security* (TLS) utilizzato da diversi protocolli applicativi TCP/IP (es. HTTP, IMAP, ...) utilizza un meccanismo di scambio delle chiavi segrete (necessarie ad ogni sessione) basato meccanismi di chiave pubblica/privata [9].

In un sistema di cifratura basato su **chiave simmetrica e condivisa**, sia per cifrare, sia per decifrare il messaggio si usa una stessa chiave k (vedi Figura 8).

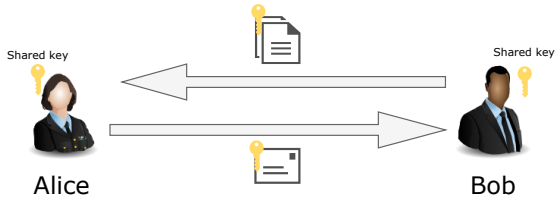

Figura 8: Utilizzo del sistema di cifratura basato su chiave simmetrica.

Le principali vulnerabilità di un sistema di cifratura a chiave simmetrica sono:

- meccanismo di scambio della chiave (come discusso sopra);

- compromissione dell'unica chiave condivisa.

5 Crittografia a chiave pubblica/privata

Uno schema di cifratura basato su **chiave pubblica privata** prevede che per ogni comunicazione cifrata servano due chiavi una *pubblica*, quindi nota a tutti, ed una *privata*, quindi nota **solo al possessore della coppia**. È importante notare che la chiave privata **non è nota a tutti e due gli interlocutori di una conversazione**. Osservando la Figura 9, notiamo che i messaggi vengono cifrati utilizzando la chiave privata, ma questa non è nota al destinatario del messaggio. Infatti, gli schemi a chiave pubblica/privata sono pensati in modo che il messaggio cifrato con una chiave privata si possa decifrare utilizzando la corrispondente chiave pubblica. Allo stesso modo, un messaggio che è stato cifrato con la chiave pubblica può essere decifrato solo usando la corrispondente chiave privata. Come tutti gli schemi di cifratura, anche uno schema basato su chiave pubblica/privata deve garantire che **sia difficile decifrare un messaggio senza possedere la chiave necessaria**.

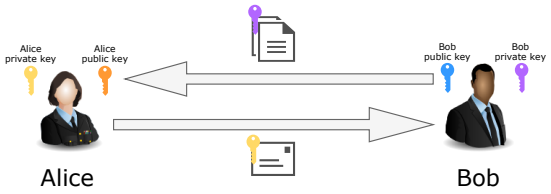

Figura 9: Utilizzo del sistema di cifratura basato su chiave pubblica/privata.

5.1 Confidenzialità utilizzando chiave pubblica/privata

Vediamo come possono Alice e Bob comunicare in maniera *confidenziale* utilizzando un meccanismo in cui entrambi possiedono una coppia di chiavi (ciascuna la propria coppia!).

1. Alice vuole mandare un messaggio cifrato a Bob e conosce la chiave pubblica di Bob che quindi usa per cifrare il messaggio.

2. Bob è l'unico in possesso della propria chiave privata e quindi è l'unico in grado di decifrare il messaggio di Alice.

3. Per rispondere ad Alice in modo confidenziale, anche Bob cifra la propria risposta usando la chiave pubblica di Alice.

4. Alice è l'unica a possedere la propria chiave privata ed è quindi l'unica in grado di decifrare il messaggio spedito da Bob.

Sebbene la confidenzialità si possa garantire con il protocollo appena visto, questo presenta alcuni inconvenienti.

- Sia mittente che ricevente devono possedere una coppia di chiavi pubblica/privata. Queste chiavi sono solitamente "difficili" o costose da generare.
- Gli algoritmi di cifratura a chiave pubblica/privata (es. RSA) richiedono considerevole tempo per la cifratura e la decifratura. Questo tempo aggiuntivo può rallentare eccessivamente la comunicazione.

5.2 Firma digitale (integrità e autenticazione)

Utilizzando una combinazione di funzioni hash sicure e chiave pubblica/privata, è possibile stabilire un protocollo di comunicazione che garantisce sia l'integrità sia l'autenticazione **senza necessariamente cifrare l'intero messaggio**. Supponiamo che Alice voglia mandare un messaggio a Bob e che voglia apporre la propria firma digitale in modo che Bob sia sicuro che è stata Alice a mandare quel messaggio. Inoltre è opportuno che vi sia una verifica di integrità vale a dire che il messaggio ricevuto da Bob sia proprio quello che ha spedito Alice. Per ottenere questo si procede nel seguente modo (si veda anche la Figura 10).

- Alice utilizza una funzione hash sicura per calcolare il *digest* del messaggio da spedire.
- Alice spedisce il messaggio in chiaro ed il digest che viene cifrato con la chiave privata di Alice.
- Bob riceve messaggio in chiaro e digest cifrato.
- Bob utilizza la stessa funzione hash usata da Alice per ricalcolare il digest sul messaggio.
- Bob decifra il digest ricevuto (utilizzando la chiave pubblica di Alice) e confronta il digest decifarto con quello che calcolato. Se i due digest coincidono, allora tutto è andato a buon in fine.

Si noti che:

- il digest cifrato può solo provenire da Alice in quanto è l'unica in possesso della sua chiave privata (garantendo l'identità del mittente).
- Siccome è difficile trovare un modo per creare un hash uguale per un messaggio diverso, quando il digest ricevuto coincide con il digest calcolato, è altamente probabile che sia stato calcolato utilizzando lo stesso messaggio che è stato ricevuto (garantendo l'integrità del messaggio ricevuto).

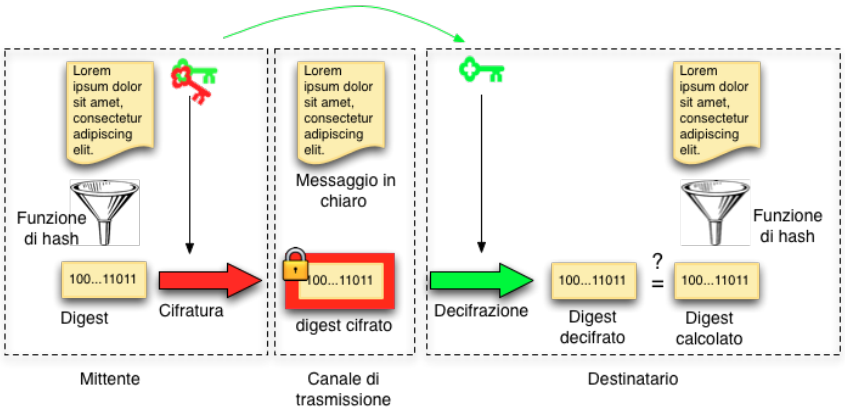

Figura 10: Funzionamento della firma digitale mediante l'utilizzo di funzioni hash sicure e chiave pubblica/privata.

5.2.1 Certification authority (CA)

Come visto, un meccanismo a chiave pubblica/privata permette di ottenere confidenzialità (anche se a fronte di un notevole sforzo di elaborazione), integrità e autorità. Tuttavia rimane un problema che non può essere risolto in quanto coinvolge proprio le chiavi (in particolare quella pubblica). Il problema è il seguente,*come facciamo ad essere sicuri che la chiave pubblica di Alice sia la sua e che Mallory non l'abbia sostituita con la propria chiave?*. Per garantire che la chiave pubblica di Alice sia esattamente la sua, è necessario che si preveda un meccanismo di *certificazione* delle chiavi che sono pubblicamente disponibili, per questo motivo esistono degli enti, chiamati **Certification Authorithy (CA)** che sono responsabili di garantire la corretta appartenenza delle chiavi pubblicate.

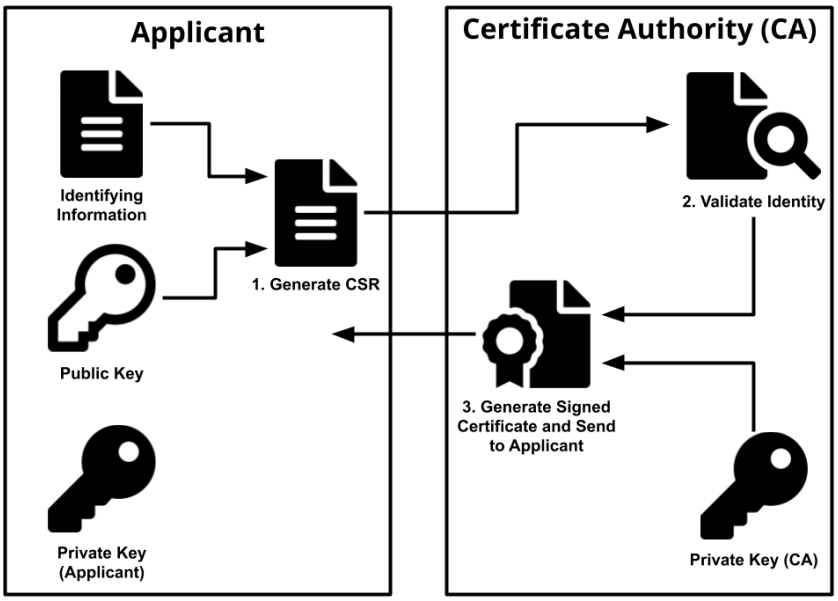

Figura 11: Meccanismo di validazione della chiave pubblic mediante Certification Authority (CA).

Il meccanismo secondo cui un CA garantisce la bontà delle chiavi è il seguente (si veda anche la Figura 11).

- Quando Alice vuole pubblicare la propria chiave pubblica, si rivolge ad una CA fornendo i propri dati (es. ragione sociale) e verificandone l'identità.
- Utilizzando un meccanismo sicuro (es. consegna a mano o mediante PEC), la CA consegna la chiave pubblica ad Alice, ma la CA **firma la chiave pubblica di Alice con la propria chiave privata**.
- A questo punto, chiunque recuperi la chiave pubblica di Alice potrà verificare la bontà della chiave in quanto firmata dal CA.

5.3 Protocollo di Diffie-Hellman

Il protocollo di Diffie-Hellman per lo scambio di chiava si basa sul meccanismo della coppia di chiave pubblica/privata [7].

Riferimenti bibliografici

[1] Ronald L. Rivest. The MD5 Message-Digest Algorithm. RFC 1321, April 1992.

[2] Robert W. Shirey. Internet Security Glossary, Version 2. RFC 4949, August 2007.

[3] William Stallings and Lawrie Brown. *Computer security: principles and practice*. Pearson Education Upper Saddle River, NJ, USA, 2012.

[4] Wikipedia. Cifrario di cesare — wikipedia, l'enciclopedia libera, 2021. [Online; in data 31-ottobre-2021].

[5] Wikipedia. Divisione euclidea — wikipedia, l'enciclopedia libera, 2021. [Online; in data 4-novembre-2021].

[6] Wikipedia. Md5 — wikipedia, l'enciclopedia libera, 2021. [Online; in data 10-novembre-2021].

[7] Wikipedia. Scambio di chiavi diffie-hellman — wikipedia, l'enciclopedia libera, 2021. [Online; in data 27-novembre-2021].

[8] Wikipedia. Secure hash algorithm — wikipedia, l'enciclopedia libera, 2021. [Online; in data 10-novembre-2021].

[9] Wikipedia contributors. Transport layer security — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Transport_Layer_Security&oldid=1057018034`, 2021. [Online; accessed 27-November-2021].