Azzolini Riccardo 2021-04-12

Cifratura asimmetrica

1 Cifratura asimmetrica

La **cifratura asimmetrica** è la più grande rivoluzione nei 3000 anni della crittografia. Essa utilizza **due chiavi** che sono legate non al canale, alla sessione di comunicazione (come avviene nel caso della cifratura simmetrica) bensì all'entità comunicante, all'utente: ogni entità ha una coppia di chiavi, una **pubblica** KP (talvolta indicata invece con KU) e una **privata** KR (la "R" sta per "Reserved", riservata).[1]

In genere, ciò che viene cifrato con la chiave pubblica di un utente può essere decifrato solo con la chiave privata dello *stesso* utente, e viceversa (ma non tutti gli algoritmi di cifratura asimmetrica permettono di "scambiare" in questo modo i ruoli delle chiavi: in alcuni algoritmi una delle due chiavi può essere usata solo per la cifratura, e l'altra solo per la decifratura). Si parla di cifratura "asimmetrica" proprio perché le due parti comunicanti effettuano operazioni diverse con chiavi diverse.

La cifratura asimmetrica fu introdotta negli anni '70 (quando la comunicazione digitale iniziava a essere importante), al fine di superare due limiti della cifratura simmetrica:

- per gestire le numerose chiavi segrete necessarie per la cifratura simmetrica servivano dei protocolli sicuri di distribuzione delle chiavi;
- serviva un metodo di *firma digitale*, per garantire che un messaggio fosse stato realmente prodotto dal mittente.

I primi a trattare questi problemi proponendo soluzioni basate sulla crittografia asimmetrica furono Whitfield Diffie e Martin Hellman, nel 1976, ma il primo vero e proprio algoritmo di cifratura asimmetrica fu RSA, sviluppato da Ron Rivest, Adi Shamir e Leonard Adleman.

Gli algoritmi di cifratura asimmetrica si basano su funzioni matematiche su numeri grandi, dunque la loro efficienza computazionale è minore rispetto a quella degli algoritmi simmetrici. Perciò, è importante capire che la cifratura asimmetrica *non sostituisce* quella simmetrica, bensì *la complementa*.

[1] Un'altra notazione è quella che indica queste due chiavi con K e K^{-1}, evidenziando il fatto che esse fungono l'una da inversa dell'altra.

2 Generazione e distribuzione delle chiavi

Per poter comunicare usando la crittografia asimmetrica, ogni utente deve prima generare una coppia di chiavi (pubblica e privata). La generazione non è casuale (come nel caso della cifratura simmetrica), bensì segue delle regole matematiche ben precise.

La chiave pubblica di ciascun utente viene poi distribuita in qualche modo (tramite registri, trasmissioni dirette, ecc.) agli altri utenti con cui vuole comunicare, mentre la chiave privata è tenuta segreta.

3 Segretezza

La cifratura asimmetrica può essere usata per realizzare la segretezza. Infatti, se Bob vuole mandare un messaggio riservato M ad Alice:

1. Bob cifra il messaggio con la chiave pubblica di Alice,

   $$C = E(M, KP_{\text{Alice}})$$

   e invia ad Alice il messaggio cifrato C.

2. Quando Alice riceve C, lo decifra con la sua chiave privata per ottenere il messaggio in chiaro:

   $$M = D(C, KR_{\text{Alice}})$$

Solo Alice ha la chiave privata KR_{Alice} necessaria per la decifratura, dunque solo Alice è in grado di decifrare il messaggio, ovvero si ha effettivamente la segretezza.

4 Autenticazione

Un altro servizio che la cifratura asimmetrica può fornire è l'autenticazione (la quale è invece difficile da realizzare con la cifratura simmetrica). Ad esempio, si supponga che Bob voglia mandare ad Alice un messaggio autenticabile, cioè tale che Alice possa verificare che il messaggio è stato effettivamente generato da Bob. Allora:

1. Bob cifra il messaggio M con la sua chiave privata,

   $$C = E(M, KR_{\text{Bob}})$$

   e invia C ad Alice.

2. Quando Alice (o chiunque altro) riceve C, lo decifra con la chiave pubblica di Bob:

   $$M = D(C, KP_{\text{Bob}})$$

Siccome la chiave pubblica KP_{Bob} di Bob è appunto pubblica, ottenibile da tutti, il messaggio può essere decifrato da chiunque: *non si ha segretezza*. Tuttavia, il fatto che il messaggio venga decifrato correttamente con KP_{Bob} da la prova che esso sia stato cifrato con la corrispondente chiave privata KR_{Bob}, e solo Bob è in possesso di tale chiave, dunque si ha la prova che il messaggio sia stato generato da Bob, ovvero sia autentico.

5 Autenticazione e segretezza

È anche possibile garantire sia la segretezza che l'autenticazione, cifrando il messaggio due volte. Tipicamente, se Bob vuole mandare ad Alice un messaggio segreto e autenticabile si procede in questo modo:

1. Bob cifra il messaggio M una prima volta con la propria chiave privata, per realizzare l'autenticazione, e cifra il risultato una seconda volta con la chiave pubblica di Alice, per realizzare la segretezza:

$$C = E(E(M, KR_{\text{Bob}}), KP_{\text{Alice}})$$

2. Quando Alice riceve C, lo decifra prima con la propria chiave privata, in modo da poter leggere il messaggio segreto, e poi con la chiave pubblica di Bob, ottenendo così una prova dell'autenticità del messaggio:

$$M = D(D(C, KR_{\text{Alice}}), KP_{\text{Bob}})$$

In teoria, le due operazioni di cifratura potrebbero essere effettuate anche nell'ordine inverso, cifrando prima con la chiave pubblica KP_{Alice} per la segretezza e poi con la chiave privata KR_{Bob} per l'autenticazione, ma in pratica ci sono vari motivi per cui è meglio applicare prima l'autenticazione e poi la segretezza. Ad esempio, se Alice memorizza il messaggio ricevuto per poi utilizzarne il contenuto (che potrebbe essere una chiave, un token di autenticazione, ecc.) in un secondo momento, magari anche più volte, è opportuno che verifichi nuovamente l'autenticazione al momento dell'uso, per accertarsi che nel frattempo il messaggio non sia stato modificato (accidentalmente a causa di errori di memorizzazione, oppure intenzionalmente da un attaccante). Allora:

- se l'autenticazione è stata applicata prima della segretezza, Alice può memorizzare il testo cifrato "intermedio", quello con solo l'autenticazione, così al momento dell'uso basta una singola operazione di decifratura per verificare l'autenticazione e accedere al messaggio;
- se invece l'autenticazione fosse stata applicata dopo la segretezza, Alice dovrebbe per forza memorizzare il messaggio cifrato due volte (non potendo "rimuovere" la segretezza senza prima rimuovere anche l'autenticazione), e quindi eseguire due operazioni di decifratura a ogni uso.

6 Uso delle chiavi

In sintesi, le due chiavi usate nella cifratura asimmetrica hanno le seguenti funzioni:

- la chiave pubblica viene impiegata per *cifrare i messaggi* al fine della segretezza e per *verificare le firme* (l'autenticazione[2]);
- la chiave privata viene impiegata per *decifrare i messaggi* e per *generare le firme*.

7 Applicazioni

Le principali applicazioni della cifratura asimmetrica sono tre:

- la **cifratura** al fine della segretezza;
- la **firma digitale**, che fornisce l'autenticazione;
- lo **scambio di chiavi**: due utenti si scambiano la chiave simmetrica (segreta) da usare per una sessione di comunicazione, che prende il nome di *chiave di sessione*.

In particolare, lo scambio delle chiavi può essere effettuato in due modi:

- Uno dei due utenti (oppure una terza entità) può generare la chiave simmetrica e inviarla all'altro utente (ai due utenti, nel caso di una terza entità) in modo segreto, tramite la cifratura con la chiave pubblica del destinatario;
- In alternativa, si possono usare degli apposito algoritmi che permettono ai due utenti di scambiare solo informazioni pubbliche, non segrete, e poi fare separatamente delle computazioni locali che portano entrambi alla generazione di una stessa chiave segreta. Tali algoritmi, come ad esempio Diffie-Hellman, non hanno la funzione di cifrare arbitrari messaggi, ma si considerano comunque algoritmi di crittografia asimmetrica perché hanno dei parametri privati e pubblici e prevedono lo scambio solo di quelli pubblici.

Oltre a RSA e Diffie-Hellman, che verranno presentati nel dettaglio, esistono vari altri algoritmi di crittografia asimmetrica, ciascuno con caratteristiche diverse che lo rendono adatto a particolari applicazioni. Alcuni di questi sono:

- *DSS / DSA* (*Digital Signature Standard/Algorithm*), un algoritmo che supporta solo la firma digitale (l'autenticazione), non permette di cifrare con la chiave pubblica, al contrario ad esempio di RSA, ma in compenso è più efficiente;
- la famiglia di algoritmi basati sulle *curve ellittiche*.

La seguente tabella riassume le applicazioni supportate dagli algoritmi menzionati finora:

[2] Si parla di "firme" perché l'autenticazione tramite cifratura asimmetrica viene solitamente effettuata con un meccanismo chiamato "firma digitale".

Algoritmo	Cifratura	Firma digitale	Scambio di chiavi
RSA	sì	sì	sì
Diffie-Hellman	no	no	sì
DSS	no	sì	no
Curve ellittiche	sì	sì	sì