Introduzione alla Crittografia Asimmetrica

Trascrizione della Presentazione

Indice

1 Crittografia Asimmetrica: Concetti di Base

La crittografia asimmetrica, nota anche come crittografia a chiave pubblica, si basa sull'uso di una coppia di chiavi distinte (una per la cifratura e una per la decifratura) per proteggere le informazioni.

Il processo fondamentale coinvolge un mittente (SRC) e un destinatario (DST).

- Il mittente cifra il messaggio in chiaro (Plaintext M) usando la **chiave di cifratura (KENC)**.
- L'operazione di cifratura è: $C = ENC(K_{ENC}, M)$, dove C è il testo cifrato (Ciphertext C).
- Il destinatario riceve il testo cifrato C e usa la **chiave di decifratura (KDEC)** per riottenere il messaggio in chiaro M.
- L'operazione di decifratura è: $M = DEC(K_{DEC}, C)$.

1.1 Proprietà delle Chiavi

La caratteristica fondamentale di questo sistema è che la chiave di decifratura è diversa da quella di cifratura:

$$K_{DEC} \neq K_{ENC}$$

Sebbene le due chiavi siano matematicamente collegate, deve essere computazionalmente impossibile (o "difficile") determinare una chiave partendo dalla conoscenza dell'altra, a meno di non possedere informazioni segrete aggiuntive.

2 Modelli d'Uso Pratici

2.1 Modello d'Uso #1: Stile HTTPS/TLS

Un esempio comune di crittografia asimmetrica è nel protocollo TLS (Transport Layer Security), utilizzato per proteggere le connessioni web (HTTPS).

- **Fase 1: Handshake Iniziale**: Questa è la fase di segnalazione. Utilizza la crittografia asimmetrica per stabilire un segreto condiviso (shared secret) e per autenticare le parti (ad esempio, il server).
- **Fase 2: Trasferimento Dati**: Una volta stabilito il segreto condiviso, entrambe le parti lo utilizzano per derivare (tramite una funzione KDF - Key Derivation Function) chiavi **simmetriche**. Queste chiavi simmetriche vengono poi usate per cifrare e autenticare tutti i dati successivi della sessione.
- **Fasi successive: Rekeying**: Periodicamente, le chiavi simmetriche possono essere "rinfrescate" (rekeying), derivando nuove chiavi sempre dallo shared secret originale stabilito durante l'handshake.

2.2 Modello d'Uso #2: Cifratura Ibrida

Questo modello combina l'efficienza della crittografia simmetrica (veloce e adatta a grandi volumi di dati) con la flessibilità di quella asimmetrica (ottima per lo scambio di chiavi).

Il processo è il seguente:

1. Si genera una chiave **simmetrica** casuale (K) (ad esempio, una chiave AES).
2. Si utilizza questa chiave K e un cifrario simmetrico per cifrare il messaggio lungo (m).
3. Si cifra la chiave K utilizzando un cifrario **asimmetrico** (usando la chiave pubblica del destinatario).
4. Si invia al destinatario sia il messaggio (m) cifrato con K, sia la chiave (K) cifrata con la chiave pubblica.

Il destinatario usa la sua chiave privata asimmetrica per decifrare K, e poi usa K per decifrare il messaggio lungo m. Un esempio di questo approccio è ECIES, utilizzato nel 5G per il Subscriber Identity Concealment (SUCI).

3 I Due Utilizzi della Crittografia a Chiave Pubblica

Assumendo che le operazioni di cifratura (ENC o E) e decifratura (DEC o D) siano inverse ($DEC(ENC(M)) = M$) e commutative ($ENC(DEC(M)) = M$), la crittografia a chiave pubblica può essere usata in due modi principali.

3.1 Cifratura a Chiave Pubblica (Confidenzialità)

In questo scenario, chiunque può cifrare un messaggio, ma solo il possessore della chiave privata può decifrarlo.

- L'utente B genera una coppia di chiavi: una pubblica (E_B) e una privata (D_B).
- B rende pubblica la sua chiave E_B.
- Qualsiasi utente A può inviare un messaggio M a B cifrandolo con la chiave pubblica di B: $C = E_B(M)$.
- Solo B può decifrare il messaggio C usando la sua chiave privata: $M = D_B(C) = D_B(E_B(M))$.

3.2 Firma Digitale (Autenticazione e Non Ripudio)

Questo è un approccio "duale": solo il possessore della chiave privata può "firmare" (cifrare), ma chiunque può "verificare" (decifrare) la firma.

- L'utente A genera una coppia di chiavi: una pubblica (E_A) e una privata (D_A).
- A rende pubblica la sua chiave E_A.
- Solo A può "firmare" un messaggio M applicando la sua chiave privata: $S = D_A(M)$.
- Chiunque (ANY B) può verificare la firma applicando la chiave pubblica di A al messaggio firmato, per riottenere M: $M = E_A(S) = E_A(D_A(M))$.

3.3 La Firma Digitale in Pratica

Nelle applicazioni reali, per efficienza, non si firma l'intero messaggio M, ma un suo **hash** (un'impronta digitale di lunghezza fissa), $H(M)$.

Processo di Firma (es. Alice):

1. Alice calcola l'hash del messaggio: $H(M)$.
2. "Firma" l'hash utilizzando la sua chiave privata (SK - Secret Key): $Tag = DEC(SK, H(M))$.
3. Invia il messaggio M e il Tag.

Processo di Verifica (es. Bob):

1. Bob riceve M e il Tag.
2. Ottiene la chiave pubblica (PK - Public Key) di Alice (che è pubblica).
3. Calcola autonomamente l'hash del messaggio M ricevuto: $H(M')$.
4. Applica la chiave pubblica di Alice al Tag per "invertirlo" (verificare): $H(M) = ENC(PK, Tag)$.
5. Confronta i due hash. Se $H(M') = H(M)$, la firma è valida: il messaggio è integro e proviene autenticamente da Alice.

La firma digitale fornisce quindi **integrità del messaggio** e **autenticazione della sorgente**.

3.3.1 Firma Digitale vs. MAC Simmetrico

- **MAC Simmetrico (es. HMAC)**: La chiave K è simmetrica e condivisa tra sorgente (SRC) e destinazione (DST). Il DST deve conoscere la stessa chiave K per verificare. Non fornisce non ripudio (il DST potrebbe aver generato il MAC).
- **Firma Digitale**: La chiave K usata per firmare è la chiave privata del mittente (SRC). Il DST ha solo bisogno della chiave pubblica (PubKey) del mittente per verificare. Questo fornisce la **non-repudiazione** (source authentication): il mittente non può negare di aver firmato il messaggio, poiché solo lui possiede la chiave privata.

4 Algoritmi Fondamentali e Problemi Difficili

La sicurezza della crittografia asimmetrica si basa su "problemi difficili" (hard problems): problemi computazionalmente facili in una direzione, ma difficili nella direzione opposta.

4.1 I Pionieri

- **Whitfield Diffie e Martin E. Hellman (1976)**: Hanno inventato la crittografia asimmetrica. Il loro primo algoritmo è stato il **Diffie-Hellman KEY AGREEMENT** (accordo di chiave).
- **Rivest, Shamir, e Adleman (1977)**: Hanno inventato il primo sistema completo di cifratura a chiave pubblica e firma digitale: il **sistema crittografico RSA**.

4.2 Problema del Logaritmo Discreto (DLOG)

Utilizzato da Diffie-Hellman (DH).

- **Facile**: Dati un primo grande p, un generatore g e un esponente x, calcolare $y = g^x \pmod{p}$.
- **Difficile**: Dati y, g, e p, calcolare l'esponente $x = \text{DLog}_g(y) \pmod{p}$.

Il calcolo dell'esponenziazione modulare (g^x) è facile e può essere eseguito in tempo polinomiale (complessità $O(n_{bit})$) usando l'algoritmo "square-and-multiply". Non esiste un algoritmo efficiente simile per il problema opposto (il logaritmo discreto).

4.3 Problema della Fattorizzazione

Utilizzato da RSA.

- **Facile**: Dati due primi grandi p e q, calcolare il loro prodotto $N = p \cdot q$.
- **Difficile**: Dato N, trovare i suoi fattori primi p e q.

5 Diffie-Hellman Key Agreement (DH)

Il protocollo Diffie-Hellman (1976) è un protocollo di **accordo di chiave (KEY AGREEMENT)**. Permette a due parti (es. Alice e Bob) di stabilire un segreto condiviso (una chiave simmetrica) scambiando solo valori pubblici, senza un canale sicuro preesistente. Si basa sul problema DLOG.

Protocollo DH:

1. Alice e Bob si accordano pubblicamente su un primo p e un generatore g.
2. Alice sceglie un numero casuale segreto x e calcola $A = g^x \pmod{p}$.
3. Bob sceglie un numero casuale segreto y e calcola $B = g^y \pmod{p}$.
4. Alice invia A a Bob.
5. Bob invia B ad Alice.
6. Alice calcola il segreto condiviso $K = (B)^x \pmod{p} = (g^y)^x \pmod{p} = g^{yx} \pmod{p}$.
7. Bob calcola il segreto condiviso $K = (A)^y \pmod{p} = (g^x)^y \pmod{p} = g^{xy} \pmod{p}$.

Alice e Bob ottengono la stessa chiave $K = g^{xy}$. Un attaccante (Eve) che osserva lo scambio vede solo g, p, A, B. Per calcolare K, dovrebbe risolvere il DLOG per trovare x o y, il che è computazionalmente difficile.

Limitazioni di DH: DH non implementa né un sistema di cifratura a chiave pubblica (non si possono inviare dati cifrati) né una firma digitale.

6 L'Algoritmo RSA

Sviluppato da Rivest, Shamir e Adleman (1977), RSA implementa sia la cifratura che la firma digitale, basandosi sul problema della fattorizzazione.

6.1 Principio Matematico: Teorema di Eulero

Il principio di RSA si basa sulla periodicità dell'esponenziazione modulare. Il **Teorema di Eulero** (o Teorema di Eulero-Fermat) definisce il periodo di $m^x \pmod N$. Questo periodo è dato dalla **funzione toziente di Eulero**, $\Phi(N)$.

Formalmente: se $\text{GCD}(m, N) = 1$, allora:

$$m^{\Phi(N)} \equiv 1 \pmod N$$

Per il caso RSA, dove $N = p \cdot q$ (con p, q primi):

$$\Phi(N) = (p-1)(q-1)$$

La conseguenza diretta è che l'aritmetica sugli esponenti avviene modulo $\Phi(N)$. Questo porta all'identità che fa funzionare RSA:

$$m^k \equiv m \pmod N \quad \text{se} \quad k \equiv 1 \pmod{\Phi(N)}$$

6.2 Costruzione di RSA

1. Si generano due primi grandi p e q (che devono restare **segreti**).
2. Si calcola il modulo RSA: $N = p \cdot q$ (questo è **pubblico**).
3. Si calcola $\Phi(N) = (p-1)(q-1)$ (questo resta **segreto**).
4. Si sceglie un esponente pubblico e (chiave pubblica), tale che $1 < e < \Phi(N)$ e $\text{GCD}(e, \Phi(N)) = 1$.
5. Si calcola l'esponente privato d (chiave privata) tale che $e \cdot d \equiv 1 \pmod{\Phi(N)}$. (d è l'inverso modulare di $e \bmod \Phi(N)$).

La **chiave pubblica** è la coppia (N, e). La **chiave privata** è d.
Operazioni:

- **Cifratura**: $C = M^e \pmod N$
- **Decifratura**: $M = C^d \pmod N$

La decifratura funziona perché: $C^d \equiv (M^e)^d \equiv M^{ed} \pmod N$. Dato che $ed \equiv 1 \pmod{\Phi(N)}$, per la conseguenza del Teorema di Eulero, si ha: $M^{ed} \equiv M^1 \equiv M \pmod N$.

6.2.1 La "Trapdoor" di RSA

La sicurezza di RSA si basa sull'assunzione che, dato N, sia difficile fattorizzarlo per trovare p e q.

- Senza p e q, è difficile trovare $\Phi(N)$.
- Senza $\Phi(N)$, è difficile calcolare d partendo da e.

Conoscere $\Phi(N)$ è la "trapdoor" (botola): rende facile calcolare d (usando l'Algoritmo Euclideo Esteso), mentre è difficile per chiunque altro.

6.3 Esempio "Giocattolo" di RSA

- $p = 11$, $q = 17$ (segreti)
- $N = 11 \cdot 17 = 187$ (pubblico)
- $\Phi(N) = (10) \cdot (16) = 160$ (segreto)
- $e = 7$ (pubblico, coprimo con 160)
- $d = 7^{-1} \pmod{160} = 23$ (segreto) (Nota: $7 \cdot 23 = 161 \equiv 1 \pmod{160}$)

Cifratura: $C = M^7 \pmod{187}$
Decifratura: $M = C^{23} \pmod{187}$
Firma: $\text{TAG} = H(M)^{23} \pmod{187}$
Verifica: $H(M) = \text{TAG}^7 \pmod{187}$

7 Il Problema: Autenticità delle Chiavi Pubbliche

La crittografia asimmetrica ha un punto debole: come facciamo a essere sicuri che la chiave pubblica che stiamo usando appartenga davvero all'entità con cui crediamo di comunicare?

7.1 Attacco "Man-in-the-Middle" (MITM)

Un utente malintenzionato (Attaccante) può interporsi nella comunicazione.

Scenario Firma (Impersonificazione):

1. Flavia ha la sua coppia di chiavi $(flaviaPK, flaviaSK)$.
2. L'Attaccante genera le sue chiavi $(gbPK, gbSK)$.
3. L'Attaccante invia un messaggio M' a Bob, firmandolo con la sua chiave $gbSK$.
4. L'Attaccante intercetta la richiesta di Bob per la chiave di Flavia e gli risponde: "Sono Flavia, ecco la mia PK: $gbPK$".
5. Bob verifica la firma sul messaggio M' usando $gbPK$. La verifica ha successo e Bob è convinto di aver ricevuto un messaggio autentico da Flavia.

Scenario Scambio di Chiavi (RSA Key Transport):

1. Flavia chiede alla Banca la sua chiave pubblica.
2. Il MITM intercetta la richiesta e risponde a Flavia: "Sono la Banca, ecco la mia PK: pk_{badguy}".
3. Flavia genera una chiave simmetrica K e la invia cifrata: $ENC(K, pk_{badguy})$.
4. Il MITM decifra il messaggio, ottiene K, poi ricifra K con la vera chiave della banca $ENC(K, pk_{bankA})$ e lo inoltra alla banca.
5. Il MITM ora possiede la chiave K e può leggere e modificare tutti i dati scambiati.

Questo attacco funziona in modo simile anche con Diffie-Hellman. Il problema è lo stesso: abbiamo bisogno di un modo per "legare crittograficamente" un'identità a una chiave pubblica.

8 La Soluzione: Certificati Digitali e PKI

8.1 Certificati Digitali

La soluzione è il **Certificato Digitale**.

- **Scopo**: Legare (bind) una chiave pubblica a un soggetto (persona, azienda, ecc.).
- **Come**: Tramite un "legame crittografico", che è una **firma digitale** apposta da una terza parte fidata (Trusted Third Party, TTP), chiamata **Certification Authority (CA)**.
- **Contenuto**: Il certificato attesta "Il soggetto con NOME X possiede la seguente CHIAVE PUBBLICA Y".

8.2 Public Key Infrastructure (PKI)

L'infrastruttura a chiave pubblica (PKI) si basa sulla fiducia in una TTP (la CA), che garantisce per gli altri utenti.

A) Emissione di un Certificato (Offline)

1. La Banca genera una coppia di chiavi (PK, SK).
2. Archivia la SK localmente (non deve mai essere divulgata).
3. Invia una richiesta alla CA (tramite un canale sicuro) contenente il suo nome (BankName) e la sua PK (BankPK).
4. La CA, dopo aver verificato l'identità della Banca, emette un certificato: $CERT = (\text{BankName}, \text{BankPK})_{CA\_sign}$. Questo è il certificato firmato con la chiave privata della CA.

B) Verifica della Validità del Certificato (Online)

1. Flavia (l'utente) vuole comunicare con la Banca.
2. La Banca invia a Flavia il suo certificato $CERT$.
3. Flavia esegue due controlli:
   - "Mi fido della CA che ha emesso questo certificato?" (È nella mia lista di CA fidate?).
   - "La firma della CA sul certificato è corretta?" (Verifica la firma usando la chiave pubblica della CA).

Questo però non basta. Un attaccante potrebbe fare un "replay" di un certificato valido.

8.2.1 Prova di Possesso (Proof of Possession)

È necessario un passo aggiuntivo: Flavia deve "sfidare" (challenge) la Banca per **dimostrare che possiede la chiave privata (SK)** associata alla chiave pubblica (PK) contenuta nel certificato.

Ci sono due modi per farlo:

- **Via Firma Digitale**: Flavia genera un numero casuale (NONCE) e lo invia alla Banca. La Banca deve firmare il NONCE con la sua chiave privata. Flavia verifica la firma con la PK del certificato.
- **Via Cifratura**: Flavia genera un NONCE, lo cifra con la PK della Banca e lo invia. Solo la vera Banca, possedendo la SK, può decifrare il NONCE e restituirlo in chiaro.

L'approccio di TLS (con RSA Key Transport) è una variante del secondo: Flavia genera la chiave di sessione K, la cifra con la PK della banca e la invia. Solo la banca autentica può decifrare K.

8.3 Catene di Certificati (Certificate Chains)

Spesso, la CA che emette il certificato del sito (es. "LUNATRUST") non è direttamente nella lista di fiducia dell'utente. In questo caso, LUNATRUST presenta il suo certificato, che è stato firmato da una CA di livello superiore (es. "UNIVERSETRUST"). Se UNIVERSETRUST è nella lista delle CA fidate (Root CA) dell'utente, la catena di fiducia è valida.

8.4 Formato dei Certificati: X.509

Lo standard de-facto per i certificati digitali è X.509. Un certificato X.509 v3 contiene molte informazioni, tra cui:

- **Subject**: L'identità del proprietario (es. CN=www.facebook.com).
- **Subject Public Key Info**: La chiave pubblica del soggetto e l'algoritmo (es. RSA).
- **Issuer**: L'identità della CA che ha emesso il certificato (es. CN=DigiCert).
- **Validity Period**: Le date di inizio e fine validità.
- **Serial Number**: Un numero unico per quel certificato.
- **Estensioni (v3)**: Campi aggiuntivi che specificano gli usi del certificato, come:
  - **Key Usage**: Per cosa può essere usata la chiave (es. Digital Signature, Key Encipherment).
  - **Extended Key Usage**: Usi specifici (es. TLS Web Server Authentication).
  - **Subject Alternative Name (SAN)**: Altri nomi associati (es. facebook.com).
  - **Basic Constraints**: Indica se il certificato appartiene a una CA (CA:TRUE) o a un utente finale (CA:FALSE).
- **Digital Signature**: La firma apposta dalla CA sull'intero certificato.

8.5 Revoca dei Certificati

Una PKI deve avere un meccanismo per gestire i certificati compromessi (es. chiave privata rubata o persa). L'approccio principale (oltre alla data di scadenza) è la revoca esplicita.

Certificate Revocation List (CRL): Ogni CA pubblica regolarmente una lista (firmata digitalmente) contenente i numeri di serie di tutti i certificati che ha revocato. Prima di fidarsi di un certificato, un'applicazione dovrebbe controllare che non sia presente nella CRL della CA che lo ha emesso. In pratica, questo controllo è spesso trascurato da applicazioni non critiche.