Crittografia Asimmetrica

Riccardo Longo

Crittografia Asimmetrica

- Mittente e destinatario operano in maniera diversa (**asimmetria**)
- La chiave si divide in **Pubblica** e **Privata**
- Non è necessario un **canale privato** per lo **scambio di chiavi**
- È comunque necessaria una infrastruttura per **attribuire** correttamente le chiavi agli utenti
- Usata sia per **cifratura** che per **firma digitale**

Schema generale

Uno schema a chiave pubblica è composto da tre algoritmi:

Generazione Chiavi dati un parametro P (dipendente dal livello di sicurezza) viene generata una coppia di chiavi $(\mathrm{PK}, \mathrm{SK})$

Cifratura dato un messaggio m e la chiave pubblica PK viene generato il cifrato c

Decifratura Dato un cifrato c e la chiave privata SK viene ricostruito il messaggio in chiaro m

Generazione Chiavi

- Il parametro in input determina la la **lunghezza** delle chiavi e va a definire gli ambienti matematici che verranno usati nelle operazioni di cifratura e decifratura
- PK è la chiave pubblica, e viene **distribuita pubblicamente**
- SK è la chiave privata, va tenuta **segreta**
- È importante che questo algoritmo abbia a disposizione un'adeguata **sorgente random** affinché le chiavi siano generate in modo sicuro

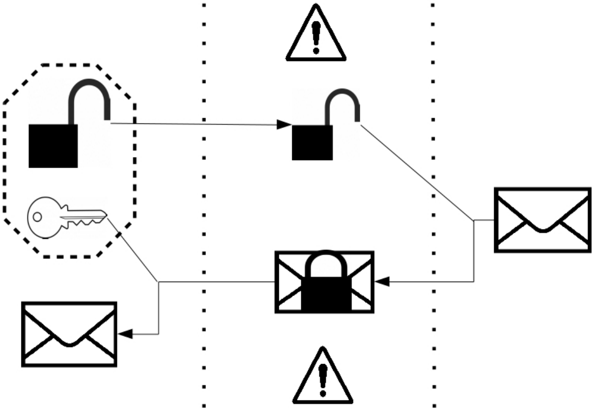
Cifratura

- **Chiunque** in possesso della chiave pubblica può cifrare
  - È importante **autenticare** separatamente il mittente in molte applicazioni
  - Serve una infrastruttura di **distribuzione delle chiavi** pubbliche adeguata per assicurare una corretta identificazione **chiave-utente**
- La cifratura può essere **deterministica** o **probabilistica**
  - deterministica se un messaggio viene sempre cifrato nello **stesso ciphertext**
  - probabilistica se a un messaggio possono corrispondere più cifrati, viene usato un **input random** nella cifratura

Decifratura

- Solo chi è in possesso della **chiave privata** può decifrare
- La decifratura è sempre **deterministica**
- Alcuni schemi includono un controllo di **integrità** del cifrato in questa fase

Schema



Trapdoor function

- facili da calcolare $\quad x \longmapsto y \quad : \quad f(x) = y$
- difficili da invertire $\quad y \longmapsto x \quad : \quad f^{-1}(y) = x$
- a meno di avere la chiave $\quad k, y \longmapsto x \quad : \quad g(k, y) = x$

- la sicurezza degli schemi a chiave pubblica è basata sull'**intrattabilità** di problemi matematici conosciuti e studiati:
  - **fattorizzazione**
  - **residui quadratici**
  - **logaritmo discreto**
  - problemi sui reticoli
  - decodifica
  - isogenie
  - sistemi multivariati

Schemi Ibridi e Key Encapsulation

- Gli schemi a chiave pubblica sono **molto più lenti** di quelli simmetrici
- Spesso vengono usati per **cifrare solo una chiave simmetrica** che poi viene effettivamente usata per cifrare i messaggi (*payload*)
- Similmente il **Key Encapsulation** prevede la generazione e cifratura di un messaggio random, che poi viene usato per generare la chiave simmetrica tramite una **KDF**

RSA

- Il nome deriva dalle iniziali degli autori: Ron **Rivest**, Adi **Shamir**, Leonard **Adleman**
- Basa la sua sicurezza sulla difficoltà della **fattorizzazione** di interi e problemi derivati
- Fa uso di **aritmetica modulare**:
  - dati $n, x, y \in \mathbb{Z}, \quad x \equiv y \mod n \iff \exists k \in \mathbb{Z} : x = y + kn$
  - si considera solo il resto della divisione per n
  - in RSA $n = pq$ con p e q primi molto grandi

Generazione Chiavi

- Genera due numeri primi p e q
  - **random**
  - di grandezza **simile** ma non troppo
  - la **primalità** si può testare velocemente
- Calcola $n = pq$, e esponente **pubblico**, spesso $e = 2^{16} + 1 = 65537$ per questioni di efficienza
- Calcola $d \equiv e^{-1} \mod mcm(p-1, q-1)$:
  - questo assicura che $(m^e)^d \equiv m \mod n \quad \forall m \in \mathbb{Z}$
  - facile da calcolare se si conoscono p e q, praticamente impossibile sapendo solo n ed e
- PK $= (n, e)$, SK $= d$

Cifratura e Decifratura

- Il messaggio M viene convertito in un intero $0 \leq m < n$ tramite una apposita funzione di padding reversibile
- Il cifrato è calcolato come $c \equiv m^e \mod n$
- Per decifrare si usa la chiave privata:

$$c^d \equiv (m^e)^d \equiv m \mod n$$

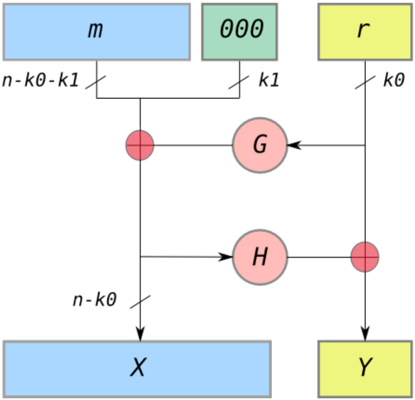
e si inverte il padding

Perché serve il padding

- Se m ed e sono abbastanza piccoli affinché $m^e < n$ l'esponenziazione è facilmente invertibile
- Se e è condiviso tra più chiavi (cosa diffusa in pratica) e vengono cifrati messaggi uguali ma per n diversi, può diventare facile ricavare m
- Senza padding lo schema è deterministico e quindi soggetto ad attacchi di tipo **chosen plaintext**
- Senza padding lo schema è **malleabile**: $(m_1^e)(m_2^e) = (m_1 m_2)^e$, quindi soggetto ad attacchi **chosen ciphertext**: si decifra c tramite la decifratura di $cr^e \mod n$, con r scelto dall'attaccante

OAEP

Optimal Asymmetric Encryption Padding



Chiavi deboli

- Se $p - q < 2n^{\frac{1}{4}}$, si può usare la **fattorizzazione di Fermat**
- Se $p - 1$ o $q - 1$ ha solo fattori piccoli, n si può fattorizzare con l'algoritmo di **Pollard p-1**
- Se $q < p < 2q, \quad d < n^{\frac{1}{4}}/3$, d si può calcolare da n ed e
- Se le chiavi non sono generate con sufficiente random, e si trova una coppia di chiavi che hanno un **fattore in comune** è facilissimo romperle entrambe

Altre considerazioni di sicurezza

- Gli algoritmi di fattorizzazione permettono attacchi pratici a chiavi di 768 bit, usare chiavi di **minimo 1024 bit, meglio 2048 bit**
- molte implementazioni sono suscettibili a **timing attacks**; usare implementazioni a decifratura costante o **blinding**: per ogni messaggio scegliere r random, calcolare $(r^e c)^d \mod n$ e poi moltiplicare per $r^{-1} \mod n$
- Un difetto nel padding **PKCS#1 v1** permetteva un attacco pratico di tipo **chosen ciphertext** contro SSL
- Con attacchi di tipo **side channel** è possibile estrarre la chiave privata da molti dispositivi

Gruppi e logaritmo discreto

- Un **gruppo** è una struttura matematica definita da un insieme $\mathbb{G}$ ed una operazione *binaria associativa*, che ammetta elemento *neutro* ed *inverso*:
  - Gli interi ($\mathbb{Z}$)
  - Gli interi modulo un primo p ($\mathbb{Z}_p$)
  - I punti di una **curva ellittica**
- In un gruppo *finito* il **logaritmo discreto** (DLOG) consiste in:

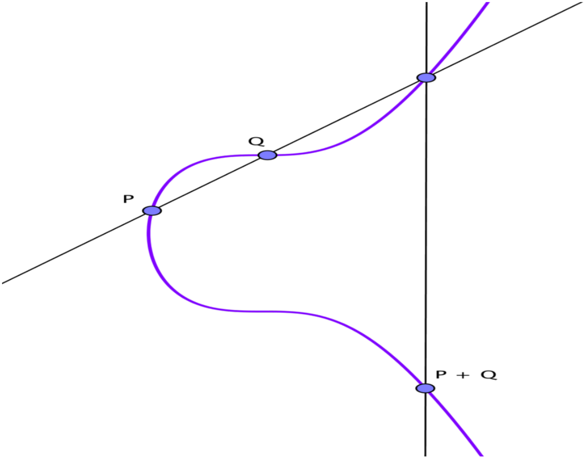
$$g, y \to x : g^x = y$$

- I gruppi in cui DLOG è un problema difficile permettono di costruire schemi a chiave pubblica

Curve Ellittiche

- Curve piane su *campi finiti* di ordine q, definite da equazioni della forma
$$y^2 = x^3 + ax + b$$
- I punti che soddisfano l'equazione, più un punto *all'infinito* $\mathcal{O}$ formano un gruppo *additivo* di ordine n
- Alcune curve con parametri accuratamente selezionati hanno DLOG difficile con **chiavi molto più piccole** rispetto ai gruppi $\mathbb{Z}_p$ (256 bit vs 3072 bit)
- Alcune curve sono particolarmente suscettibili a **side channel**, le curve di *Edwards* invece resistono meglio
- Vari sistemi di coordinate (proiettive, jacobiane, ...) consentono di rappresentare i punti con varie ottimizzazioni, spesso viene usata solo la X, derivando poi le altre

Addizione sulle curve ellittiche



ElGamal

- **Generazione chiavi**: $\mathbb{G}$ gruppo *ciclico* di *ordine* q, g *generatore* (parametri pubblici, condivisibili da altri utenti)
  - Genera $1 < x < q - 1$ random
  - $\text{PK} = h = g^x$
  - $\text{SK} = x$
- **Cifratura**: $1 < y < q - 1$ random, messaggio m mappato in $m' \in \mathbb{G}$
  - $s = h^y = (g^x)^y = g^{xy}$
  - $c = (c_1, c_2) = (g^y, m's)$
- **Decifratura**: per il *teorema di Lagrange* $g^{q-1} = 1$ quindi:
  - $s' = c_1^{q-1-x} = g^{(q-1-x)y}$
  - $m' = c_2 s' = m'ss' = m'g^{xy}g^{(q-1-x)y} = m'g^{(x+q-1-x)y} = m'(g^{q-1})^y = m'1^y$

Cramer-Shoup

- **Generazione chiavi**: $\mathbb{G}$ gruppo ciclico di ordine q, g_1 e g_2 generatori distinti
  - Genera SK $= (x_1, x_2, y_1, y_2, z)$ random, $1 < \cdot < q - 1$
  - PK $= (c, d, h) = (g_1^{x_1} g_2^{x_2}, g_1^{y_1} g_2^{y_2}, g_1^{z})$
- **Cifratura**: $1 < k < q - 1$ random, messaggio m mappato in $m' \in G$, H *one-way* hash
  - $u_1 = g_1^k, u_2 = g_2^k, e = h^k m'$
  - $\alpha = H(u_1, u_2, e), v = c^k d^{k\alpha}$
  - il ciphertext è (u_1, u_2, e, v)
- **Decifratura**:
  - calcola $\alpha = H(u_1, u_2, e)$ e verifica che $v = u_1^{x_1} u_2^{x_2} (u_1^{y_1} u_2^{y_2})^{\alpha}$
  - $m' = e/(u_1^z)$