Principles of Cryptocurrency Design

Appunti ed Esercizi

Francesco Pasquale

15 maggio 2025

1 Elliptic Curve Digital Signature Algorithm (ECDSA)

Sia $(\mathbb{F}_p, +, \cdot)$, con p primo, il campo dei numeri $\{0, 1, \ldots, p-1\}$ con le usuali operazioni di somma e prodotto modulo p. Sia $\mathcal{C} = \{(x, y) \in \mathbb{F}_p^2 : y^2 = x^3 + ax + b\} \cup \{\infty\}$ una curva ellittica. Sia $G \in \mathcal{C}$ un punto della curva tale che l'ordine n del gruppo $\langle G \rangle$ generato da G sia primo. Sia $(\texttt{sk}, \texttt{pk})$, con $\texttt{sk} \in \{1, \ldots, n-1\}$ e $\texttt{pk} = \texttt{sk} \cdot G$, una coppia di chiavi. Sia HASH una funzione hash crittografica.

L'algoritmo di firma digitale ECDSA di un messaggio $\langle msg \rangle$ funziona in questo modo (si vedano le specifiche dettagliate in [1]):

Algorithm 1 SIGN($\langle msg \rangle$, sk)

1: Calcola $h = \text{HASH}(\langle msg \rangle)$ e assumiamo $h < n$ (se non lo è, si prende il numero formato solo dai $\log n$ bit più significativi);
2: Scegli $k < n$ (possibilmente u.a.r. e con uno pseudorandom generator *sicuro*);
3: Calcola $R = k \cdot G$ e indichiamo con (r, y_r) le sue coordinate;
4: Calcola $s = k^{-1}(h + r \cdot \texttt{sk}) \mod n$
5: Restituisci la firma $\sigma = (r, s)$

Si osservi che è cruciale che il valore k scelto alla linea 2 rimanga segreto, altrimenti dal messaggio $\langle msg \rangle$, dalla firma $\sigma = (r, s)$ e da k si può ricavare la chiave segreta sk invertendo l'equazione alla linea 4. E questo non è l'unico fatto a cui bisogna fare attenzione, come si deduce dall'esercizio seguente.

Esercizio 1. Verificare che, se conosciamo due messaggi distinti, m_1 e m_2, e due firme, σ_1 e σ_2 di m_1 e m_2 rispettivamente, ottenute usando lo stesso k alla linea 2 dell'Algoritmo 1, allora possiamo calcolare la chiave segreta sk con cui sono state generate le firme.

Per via di quanto evidenziato con l'esercizio precedente, potrebbe essere utile non generare il k alla linea 2 u.a.r., ma generarlo deterministicamente a partire dal messaggio e dalla chiave segreta, in modo da evitare l'esigenza di avere accesso a uno pseudorandom generator sicuro ogni volta che bisogna firmare un messaggio e per evitare la possibilità che ci siano due messaggi distinti firmati con lo stesso valore di k.

Esercizio 2. Progettare un meccanismo per derivare in modo deterministico e "sicuro" un intero k da usare alla linea 2 dell'Algoritmo 1 a partire dal messaggio $\langle msg \rangle$ da firmare e dalla chiave segreta sk. Poi vedere la proposta descritta qui.

L'algoritmo di verifica della firma funziona in questo modo:

Algorithm 2 VERIFY($\langle msg \rangle$, pk, σ)

1: Calcola $h = \text{HASH}(\langle msg \rangle)$ e assumiamo $h < n$ (se non lo è, si prende il numero formato solo dai $\log n$ bit più significativi);

2: Calcola $\begin{cases} u_1 &= s^{-1}h \mod n \\ u_2 &= s^{-1}r \mod n \end{cases}$, dove r e s sono le componenti della firma $\sigma = (r, s)$

3: Calcola il punto sulla curva $\hat{R} = u_1 G + u_2 \cdot \texttt{pk}$;

4: Restituisci TRUE se la prima coordinata di $\hat{R}$ è uguale a r, altrimenti restituisci FALSE.

Esercizio 3. Verificare che lo schema di firma digitale descritto negli Algoritmi 1 e 2 è *corretto*: per ogni coppia di chiavi $(\texttt{sk}, \texttt{pk})$ e per ogni messaggio $\langle msg \rangle$, si ha che

$$\text{VERIFY}\big(\langle msg \rangle, \texttt{pk}, \text{SIGN}(\langle msg \rangle, \texttt{sk})\big) = \text{TRUE}$$

2 L'upgrade "Segregated Witness" al protocollo Bitcoin

Nell'upgrade al protocollo Bitcoin denominato Segregated Witness (SegWit) sono state apportate numerose modifiche. Una delle modifiche principali è stata una diversa serializzazione delle transazioni, per risolvere il problema della malleabilità.

Esercizio 4. Leggendo le specifiche sulla nuova serializzazione delle transazioni introdotta con l'upgrade SegWit (si veda, per esempio, qui), modificare opportunamente la classe TRANSACTION, implementata nella lezione del 14 aprile (`https://www.mat.uniroma2.it/~pasquale/dida/aa2425/pcd/pcd250414.zip`), in modo che faccia correttamente anche il *parsing* delle transazioni con la nuova serializzazione.

Riferimenti bibliografici

[1] Daniel R.L. Brown. *SEC 1: Elliptic Curve Cryptography.* Certicom Research, 5 2009. Rev. 2.0. `https://www.secg.org/sec1-v2.pdf`.