Principles of Cryptocurrency Design

Appunti ed Esercizi

Francesco Pasquale

10 marzo 2025

1 Digital signatures

Per un background teorico sugli schemi di firme digitali si veda, per esempio, il Capitolo 12 in [1].

Esercizio 1. Il codice Python seguente[1]

Algorithm 1 Hashed RSA

```
from Crypto.PublicKey import RSA
from hashlib import sha256

keyPair = RSA.generate(bits = 1024)

msg = b'Benvenuti al corso di Principles of Cryptocurrency Design - AA 23/24!'
msg_hash = int.from_bytes(sha256(msg).digest(), byteorder = 'big')
msg_sig = pow(msg_hash, keyPair.d, keyPair.n)

print("Firma: ", hex(msg_sig))
```

ha scritto a video questa stringa:

```
Firma: 0x1c46bba62ba574a4b048adc57226ae9f0538a2da7202100d93ff0c41cd3767d8074690b
e996e8f0e9c38f75f222bb15c7dbd48db5211d08f1d072f196357fd047169530d0393f4c6f569eddf
8286f7742a756708615cd995a10f9ee3335db9c79f14daaec77fb12e94bd07435df767d618825591c
7413f29d572e7a320514ba0
```

Scrivere un programma per verificare che si tratta di una firma valida del messaggio *Benvenuti al corso di Principles of Cryptocurrency Design - AA 24/25!* generata con la chiave privata associata alla chiave pubblica

```
n = 0xaded882f418c88459440a37076cc42a9b2f40c56d6308ebd32f1724631f5035cea206d17200
8884c3993670e44362f2624ff6ea3f508f9d085f05d1a2487ea4949578588ff2c551a05a0f98369e2
5dc7441237c785e8ea58379ebe54cf82e9067c02457d3ebde78e7daeaabe4ede6cd561820ed136ed6
89b7e70b79a412ad509

e = 0x10001
```

[1] La libreria `hashlib` è built-in Python. Per `Crypto` usare la libreria `pycryptodome` `https://pypi.org/project/pycryptodome/`

Esercizio 2. Supponiamo che, invece di firmare l'*hash* del messaggio (linee 7 e 8 in Algorithm 1), avessimo firmato direttamente il messaggio:

Algorithm 2 Textbook RSA

```
1. from Crypto.PublicKey import RSA
2.
3. keyPair = RSA.generate(bits = 1024)
4.
5. msg = b'Benvenuti al corso di Principles of Cryptocurrency Design - AA 23/24!'
6. msg_sig = pow(int.from_bytes(msg, byteorder='big'), keyPair.d, keyPair.n)
7.
8. print("Firma: ", hex(msg_sig))
```

1. Scrivere un programma per verificare che la seguente

   `Firma: 0x87106345c3d2bbde2f7a778fc3551c1559761c15f032efbc4351c0f14e453a95e42e419461adf461a76883680b9bfd73f3d032cb1b7996ab371039303a5ef0e9088bfa12c996aededa21cfde926eee625de2ae1a3319ce594b23264da46b7912e3659087ebe58cd890636ad3a7cfdf71a3e6f574d7a1d0452260d2a3b1a22549`

   è una firma valida del messaggio *Benvenuti al corso di Principles of Cryptocurrency Design - AA 24/25!* generata con lo schema in Algorithm 2 con la chiave privata associata alla stessa chiave pubblica dell'esercizio precedente.

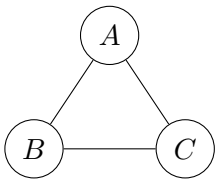
2. Mostrare che lo schema di firma digitale in Algorithm 2 non è sicuro trovando una coppia di numeri interi (`fake_msg`, `fake_sig`) tale che `fake_sig` risulti una firma valida per `fake_msg` relativamente alla chiave pubblica del punto precedente.[2]

3. Notare il motivo per cui l'attacco del punto precedente invece non è attuabile sullo schema di firma digitale in Algorithm 1.

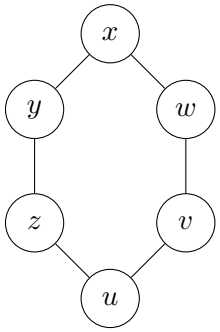
2 Lower bound per Byzantine Broadcast senza PKI

Esercizio 3. Considerate i sistemi G e H qui di seguito.

Sistema G



Sistema H



1. Eseguire il protocollo di Dolev-Strong con $f = 1$ nel sistema G, quando tutti i nodi sono onesti, A è il nodo sorgente e riceve in input 1.

2. Eseguire il protocollo di Dolev-Strong con $f = 1$ nel sistema H in cui x e u sono copie esatte[3] di A e ricevono in input 1, y e v sono copie esatte di B e z e w sono copie esatte di C (tutti i nodi sono onesti e seguono il protocollo).

[2] Hint: Partire da una `fake_sig` arbitraria e trovare un `fake_msg` che ha `fake_sig` come firma.
[3] Ossia entrambe hanno la stessa chiave privata

3. Osservare che le esecuzioni dei due punti precedenti sono equivalenti (A, B e C non possono distinguere se sono nel sistema G o nel sistema H).

4. Eseguire il protocollo di Dolev-Strong con $f = 1$ nel sistema H in cui x e u sono copie esatte di A, ma alla copia di A in x viene dato in input 1 mentre alla copia di A in u viene dato in input 0; y e v sono copie esatte di B e z e w sono copie esatte di C (tutti i nodi sono onesti e seguono il protocollo). Quali sono gli output dei sei nodi?

5. È possibile simulare l'esecuzione in H del punto precedente con una esecuzione equivalente nel sistema G in cui

   (a) B e C sono nodi onesti e A è un nodo corrotto che simula il comportamento dei nodi x, u, v e w?

   (b) A e B sono nodi onesti e C è un nodo corrotto che simula il comportamento dei nodi z, u, v e w?

Esercizio 4. A lezione abbiamo dimostrato che, in assenza di PKI non esistono protocolli per il problema Byzantine Broadcast che soddisfano *validity* e *consistency* per tre nodi, se almeno uno dei tre è corrotto.

Generalizzare la dimostrazione al caso di un numero di nodi arbitrario: dato un n qualunque, in assenza di PKI nessun protocollo per Byzantine Broadcast può soddisfare *validity* e *consistency*, se almeno $n/3$ dei nodi sono corrotti.

Riferimenti bibliografici

[1] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography.* Chapman and hall/CRC, 2007.