Generazione entropia con i dadi

Impara a creare la tua mnemonica

👤 Valerio Vaccaro

Spazio 21

📅 2025-10-24

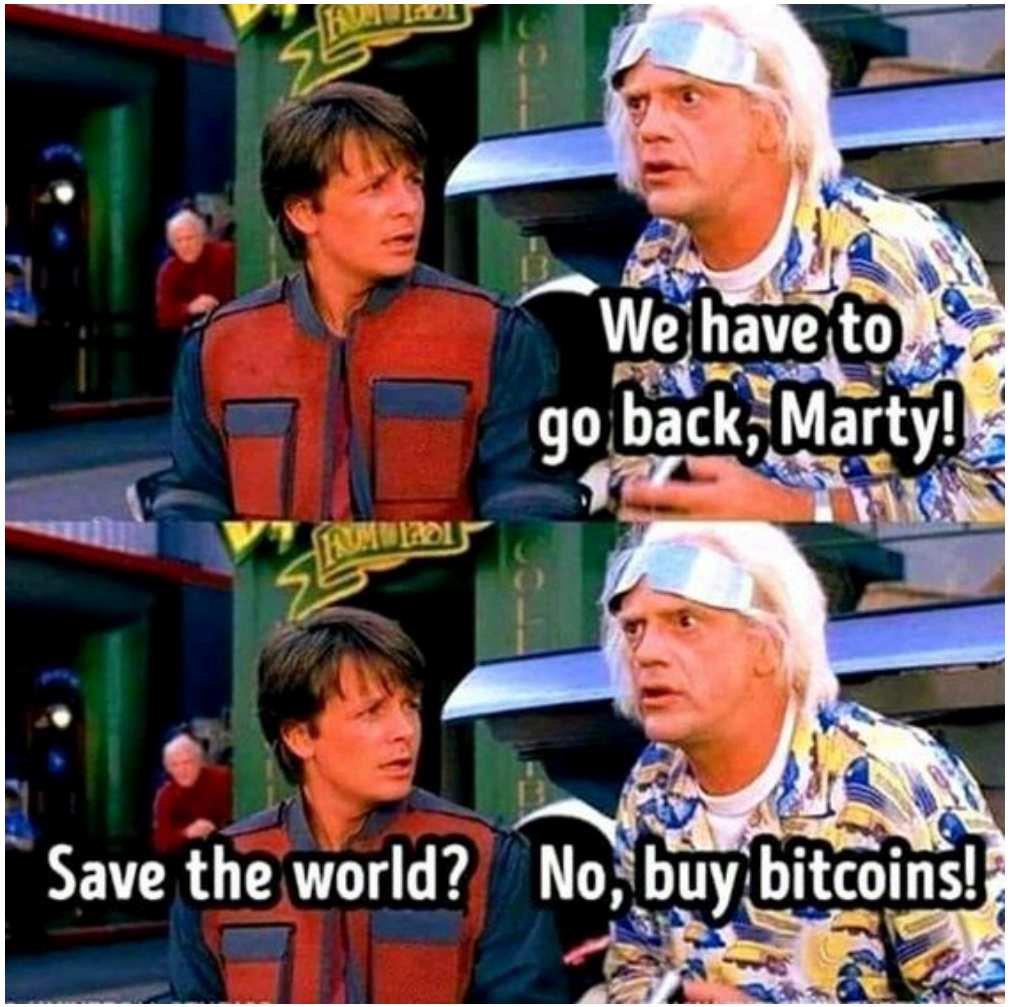
👤 Valerio Vaccaro 👀

- 💻 Sviluppatore Bitcoin ed Esperto Hardware
- 🔥 Contributore a progetti Bitcoin open source
- ⚠️ Appassionato di hardware fai-da-te (DIY)
- ₿ Ingegnere Bitcoin e Liquid presso Blockstream

👤 https://www.linkedin.com/in/valeriovaccaro/

🐙 https://github.com/valerio-vaccaro/

Meme



Licenza

May this presentation inspire you to become more self-sovereign!

Sommario

- 🎯 Introduzione alla generazione di entropia
- 🎲 Metodi TRMG: D8+D16+D16, D6, monete, carte
- 📝 Creazione delle parole mnemoniche
- ✅ Calcolo del checksum
- 🔧 Metodi per trovare la parola finale
- 💾 Backup e sicurezza

🎯 Introduzione

Creare la mnemonica sfruttando una propria fonte di entropia è **ridurre la superficie di attacco** di un wallet Bitcoin.

Tuttavia occorre tenere conto di alcuni fattori:

- 🚀 Il processo deve risultare **semplice e veloce**
- 🔒 L'entropia non va copiata su device non necessari
- ⚠️ Bisogna evitare l'uso di software/script/programmi
- 🔧 Differenti setup possono richiedere processi leggermente differenti

🎯 Introduzione

🔧 Fonti Hardware

- 🖥️ Generatore di numeri casuali hardware (TRNG)
- 🌡️ Sensori di temperatura
- ⚡ Rumore elettronico e radio
- 🕐 Jitter degli orologi

🖱️ Fonti Utente

- 👆 Movimenti nell’interfaccia grafica
- ⌨️ Tempi di arrivo dei messaggi
- 🕐 Intervalli nella pressione dei tasti
- 📷 Input della fotocamera

Ma come facciamo a sapere se l'entropia è veramente casuale?

🎯 Introduzione 🚀

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [1]:
```
import os

with open('random.bin', 'wb') as file:
    for i in range(1,1024*1024):
        seed = os.urandom(32)
        file.write(seed)
```

In [2]:
```
from hashlib import sha256

seed = 'pippo'.encode('utf-8')

with open('random2.bin', 'wb') as file:
    for i in range(1,1024*1024):
        seed = sha256(seed).digest()
        file.write(seed)
```

In [ ]:

🎯 Introduzione 🚀

In [1]:
```
cat bible.txt | ent
```

```
Entropy = 4.596758 bits per byte.

Optimum compression would reduce the size
of this 4451368 byte file by 42 percent.

Chi square distribution for 4451368 samples is 68853960.24, and randomly
would exceed this value less than 0.01 percent of the times.

Arithmetic mean value of data bytes is 86.2069 (127.5 = random).
Monte Carlo value for Pi is 4.000000000 (error 27.32 percent).
Serial correlation coefficient is 0.109292 (totally uncorrelated = 0.0).
```

In [2]:
```
cat random.bin | ent
```

```
Entropy = 7.999995 bits per byte.

Optimum compression would reduce the size
of this 33554400 byte file by 0 percent.

Chi square distribution for 33554400 samples is 249.31, and randomly
would exceed this value 58.87 percent of the times.

Arithmetic mean value of data bytes is 127.4935 (127.5 = random).
Monte Carlo value for Pi is 3.142053501 (error 0.01 percent).
Serial correlation coefficient is 0.000032 (totally uncorrelated = 0.0).
```

In [3]:
```
cat random2.bin | ent
```

```
Entropy = 7.999995 bits per byte.

Optimum compression would reduce the size
of this 33554400 byte file by 0 percent.

Chi square distribution for 33554400 samples is 252.71, and randomly
would exceed this value 52.87 percent of the times.

Arithmetic mean value of data bytes is 127.4897 (127.5 = random).
Monte Carlo value for Pi is 3.142155783 (error 0.02 percent).
Serial correlation coefficient is -0.000041 (totally uncorrelated = 0.0).
```

In [ ]:
```

```

I Metodi TRMG

TRMG = **T**rue **R**andom **M**nemonic **G**enerator

🎯 Metodi Disponibili:

1. 🎲 **D8+D16+D16**: 1 dado a 8 facce + 2 dadi a 16 facce
2. 🎲 **D6**: Dadi a 6 facce standard
3. 🪙 **Monete**: Lancio di monete
4. 🃏 **Carte da gioco**: Estrazione di carte

Attenzione: in tutti i casi dopo l'estrazione server correggere l'ultima parola per essere coerenti con il checksum

🎲 Metodo D8+D16+D16

Metodo principale utilizzato nel TRGM:

- 🎲 **1 dado a 8 facce** (sempre il primo)
- 🎲 **2 dadi a 16 facce**

Vantaggi:

- ✅ Massima entropia per lancio
- ✅ Processo più veloce
- ✅ Meno lanci necessari

Svantaggi:

- ❌ Dadi a 16 facce meno comuni
- ❌ Più costosi da acquistare

📝 Creazione delle Parole

Il primo step è il calcolo delle **12 o 24 parole** della mnemonica.

- 📊 **12 parole** sono normalmente più che sufficienti per la sicurezza
- 🎯 Per trovare la parola basta scorrere la tabella cercando la corrispondenza dei dadi
- 📋 Il **primo dado è SEMPRE** quello da 8 facce
- 📖 La colonna **word** contiene la parola cercata

Ad ogni lancio corrisponde sempre una parola

github.com/valerio-vaccaro/TRMG/blob/main/d8ff.md

Esempio di Lancio

Lancio dei Dadi:

- Dado 8 facce: **1**
- Dado 16 facce: **9**
- Dado 16 facce: **11**

Risultato:

BACON

Tabella di Riferimento:

First	Second	Third	Word
1	9	11	bacon
1	9	12	badge
1	9	13	bag

🎲 Metodo D6 (Dadi a 6 facce)

Metodo alternativo con dadi standard:

- 🎲 **Dadi a 6 facce**
- ✅ Dadi facilmente reperibili
- ❌ Più lanci necessari

Dobbiamo riempire questa griglia

1024	512	256	128	64	32	16	8	4	2	1	Index	Word

Metodo D6 (Dadi a 6 facce)

Considerando questa tabella

Result	Value
1	00
2	01
3	10
4	11
5	0
6	1

Ogni risultato può quindi riempire uno o due caselle della griglia, i valori che superano la dimensione della griglia possono essere ignorati.

Metodo D6 (Dadi a 6 facce)

Riempita la grigla basta ricercare nella tabella il risultato corrispondente alla griglia compilata.

github.com/valerio-vaccaro/TRMG/blob/main/d6.md

🪙 Metodo Monete

Metodo con monete per massima semplicità:

- 🪙 **11 lanci di moneta** per ogni parola
- 📊 Ogni lancio genera 1 bit (testa=1, croce=0)
- ✅ Disponibili ovunque
- ❌ Molti lanci necessari (132 per 12 parole)
- ❌ Processo molto lungo

Si usa la stessa tabella vista al punto precedente.

🃏 Metodo Carte da Gioco

Metodo con carte da poker:

- 🃏 **Mazzo da 52 carte**
- ✅ Carte facilmente reperibili
- ❌ Richiede mescolamento accurato

Si riempie la solita griglia.

1024	512	256	128	64	32	16	8	4	2	1	Index	Word

Metodo Carte da Gioco

Questa volta però la tabella per il riempimento è basata sulle carte da poker

Suit	Rank	Value
Spades	A	00000
Spades	2	00001
Spades	3	00010
Spades	4	00011
Spades	5	00100
Spades	6	00101
...		

github.com/valerio-vaccaro/TRMG/blob/main/poker.md

✅ Calcolo del Checksum

L'**ultima parola** non è completamente decisa da noi ma contiene una **parte di controllo**.

Mnemonica 12 parole:

- 🎯 Possiamo scegliere i primi **7 bit** di entropia
- ✅ Gli ultimi **4 bit** sono il checksum

Mnemonica 24 parole:

- 🎯 Possiamo scegliere i primi **3 bit** di entropia
- ✅ Gli ultimi **8 bit** sono il checksum

🔧 Metodi per Trovare la Parola Finale

🎯 Brute Force

- Si provano tutte le parole in sequenza
- Molto arduo con mnemoniche a 24 parole
- Usato con **Ledger** o **Electrum** (flag BIP39)

🧮 Calcolo Matematico

- Si calcolano tutte le parole possibili
- Con le prime 11 o 23 parole
- Usato con **Jade** e altri hardware wallet

🔧 Metodi per Trovare la Parola Finale

🎨 Inserimento Completo

- Si inserisce la mnemonica completa
- L'hardware wallet la sistema automaticamente
- Usato elegantemente da **Specter-DIY**

💾 Backup e Sicurezza

Fondamentale avere una **buona politica di backup**:

- 📚 **Multipli backup**
- 💿 **Su multipli supporti**
- 🔐 **Eventualmente criptati o splittati** (ma bisogna saperlo fare bene)

⚠️ Regole di Sicurezza

✅ Da Fare:

- 🎲 Usare sempre dadi fisici
- 📝 Scrivere su carta
- 🔒 Conservare in luoghi sicuri
- 🔄 Fare multipli backup

❌ Da Evitare:

- 💻 Non copiare su device digitali
- 🚫 Non usare software non verificati da voi
- ⚠️ Non condividere l'entropia
- 🎯 Non perdere i backup

Bibliografia

- officinebitcoin.it/lezioni/mnedad
- TRMG Repository - True Random Mnemonic Generator
- BIP 39 - Mnemonic code for generating deterministic keys

Domande

💻 Progetto Satoshi Spritz

- 💻 Federazione di gruppi locali di Bitcoiner
- 🎓 Eventi gratuiti e privacy oriented
- 🤖 BITCOIN ONLY
- 🔧 Satoshi Spritz Connect online settimanale
- 📖 Orientato all'apprendimento della self-sovereign
- 🍕 Tutte le settimane un evento online -> Satoshi Spritz Connect

https://satoshispritz.it

https://t.me/SatoshiSpritzConnect

₿ Officine Bitcoin

- 🤝 Comunità Italiana di Bitcoiners, totalmente gratuita
- 🤖 BITCOIN ONLY
- 🎓 Focus su educazione e sviluppo di progetti
- 📋 Progetti:
  - 💼 Sviluppo nodi Bitcoin
  - 👨‍🏫 Uso di Hardware Wallet
  - 💻 Filosofia open source
  - 🤝 Installazione di Debian
  - 🎲 **Mnemoniche & Dadi** (questa lezione!)
  - ... e molto altro

https://officinebitcoin.it