Bitcoin Cryptography Primitives

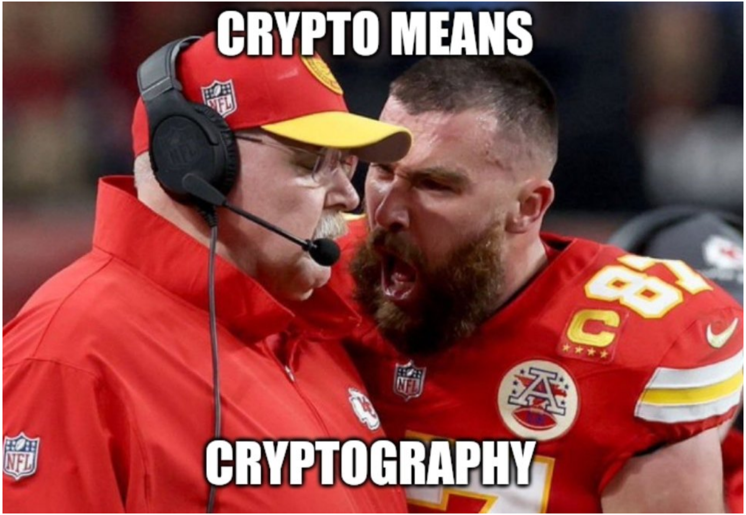
Leonardo Comandini
leonardocomandini@gmail.com

March, 2024

About me

- PoliMi, math engineering, quantitative finance
- Thesis about timestamping with Bitcoin
- Eternity Wall, OpenTimestamps
- Blockstream, Green Wallet, Liquid Network
- Software developer, applied cryptography

Definitions



Contents

- Hash Functions
- Merkle Trees
- Elliptic Curve Cryptography
- Discrete Logarithm Problem
- Signing Algorithms (ECDSA, Schnorr)

Next week: how these building blocks are used in Bitcoin wallets.

Hash functions

$$h : \{0,1\}^* \to \{0,1\}^k \approx 0..2^k - 1$$

Desired properties:

- Pre-image resistance
- Second pre-image resistance
- Collision resistance

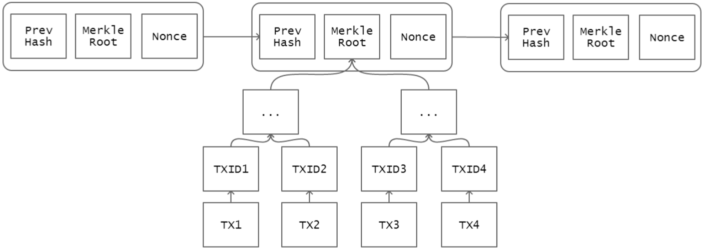
Examples:

- SHA256
- RIPEMD160

Hash functions

- Deterministic: the same input yields to the same output $\rightarrow$ commitment-reveal schemes
- "Random": given a random input, the output is uniformly random $\rightarrow$ Proof of Work

Merkle Trees



- Straight arrows represent hash
- Curved arrows represent concatenation, then hash

Symmetric Cryptography

Encryption:

$$encrypt(key, plaintext) \rightarrow cyphertext$$

Decryption:

$$decrypt(key, cyphertext) \rightarrow plaintext$$

Faster and more efficient. Drawback: single *key*.

(A)Symmetric encryption is not used in Bitcoin.

Elliptic Curve Cryptography

Let p be a prime number.
Let $\mathbb{F}_p$ denote the field of integers modulo p.
Let $a, b \in \mathbb{F}_p : 4a^3 + 27b^2 \neq 0 \; mod \; p$.

$$E(\mathbb{F}_p) := \{(x, y) \in \mathbb{F}_p^2 : y^2 = x^3 + ax + b \; mod \; p\} \cup \{\infty\}$$

It is possible to define an addition operation:

$$+ : E(\mathbb{F}_p) \times E(\mathbb{F}_p) \to E(\mathbb{F}_p)$$

and scalar multiplication:

$$\cdot : \mathbb{Z} \times E(\mathbb{F}_p) \to E(\mathbb{F}_p)$$

E.g. Bitcoin uses *secp256k1*.

Elliptic Curve Discrete Logarithm Problem

Fix $G \in E(\mathbb{F}_p)$ call it *generator*.
Let $n = |E(\mathbb{F}_p)|$ be a prime number, call it *order of the curve*.
Let $x \in \mathbb{Z}_n$ be a private key and $P = x \cdot G$ be the corresponding public key.

$$
\begin{array}{lll}
\mathbb{Z}_n \to E(\mathbb{F}_p) & \textit{private to public} & (\textit{easy}) \\
E(\mathbb{F}_p) \to \mathbb{Z}_n & \textit{public to private} & (\textit{hard})
\end{array}
$$

Elliptic Curve Digital Signature Algorithm

```
def sign(x, m):
    k = rand(n)
    r = (kG).x % n
    e = h(m)
    s = k^-1*(e + rx) % n
    return r, s

def verify(sig, m, P):
    r, s = sig
    e = h(m)
    return sR == eG + rP
```

Elliptic Curve Digital Signature Algorithm

- Prove knowledge of the secret key while committing to a message
- Used in Bitcoin from the beginning

Schnorr Signature Algorithm

```
def sign(x, m):
    k = rand(n)
    r = (kG).x
    e = h(r||P||m)
    s = (k + ex) % n
    return r, s

def verify(sig, m, P):
    r, s = sig
    e = h(r||P||m)
    R = (r, mod_sqrt(r**3 + a*r + b))
    return sG == R + eP
```

Schnorr Signature Algorithm

- Added to Bitcoin in 2021 (via soft-fork)
- Linear: the sum of two signatures for the same message, is a valid signature for the sum of the public keys
- Linearity allows complex protocols, verification remains simple, complexity in setup and off chain (musig, adaptor signatures)

Brute-forcing a 256-bit key

THERMODYNAMIC CONSTRAINTS WHEN BRUTE-FORCING A 256-BIT KEY

ACCORDING TO THE 2^{ND} LAW OF THERMODYNAMICS, THE MINIMUM AMOUNT OF ENERGY REQUIRED TO RECORD A SINGLE BIT BY CHANGING THE STATE OF A SYSTEM IS

$$kT$$

BOLTZMAN CONSTANT

TEMPERATURE OF THE SYSTEM

ASSUMING OUR COMPUTER IS IDEAL AND RUNNING AT 3.2 K (THE TEMPERATURE OF THE COSMIC BACKGROUND RADIATION), A BIT CHANGE WOULD CONSUME 4.4×10^{-16} ERG. SINCE THE ANNUAL OUTPUT OF THE SUN IS 1.21×10^{41} ERG, IF WE USED ALL ITS ENERGY WE COULD POWER 2.7×10^{56} SINGLE BIT CHANGES, WHICH IS ENOUGH TO PUT A 187-BIT COUNTER THROUGH ALL ITS VALUES. TO RUN THROUGH A 256-BIT KEY WE WOULD NEED TO BUILD DYSON SPHERES AND CAPTURE THE ENERGY OF

$$2^{69} \simeq 5.9 \times 10^{20} \text{ SUNS DURING 1 YEAR!}$$

Which tools would use to solve these problems? How?

- 1. Commit to the result of an event, without revealing the chosen result.
- 2. Store private data on a server.
- 3. Issue independently verifiable public statements.

Thank you