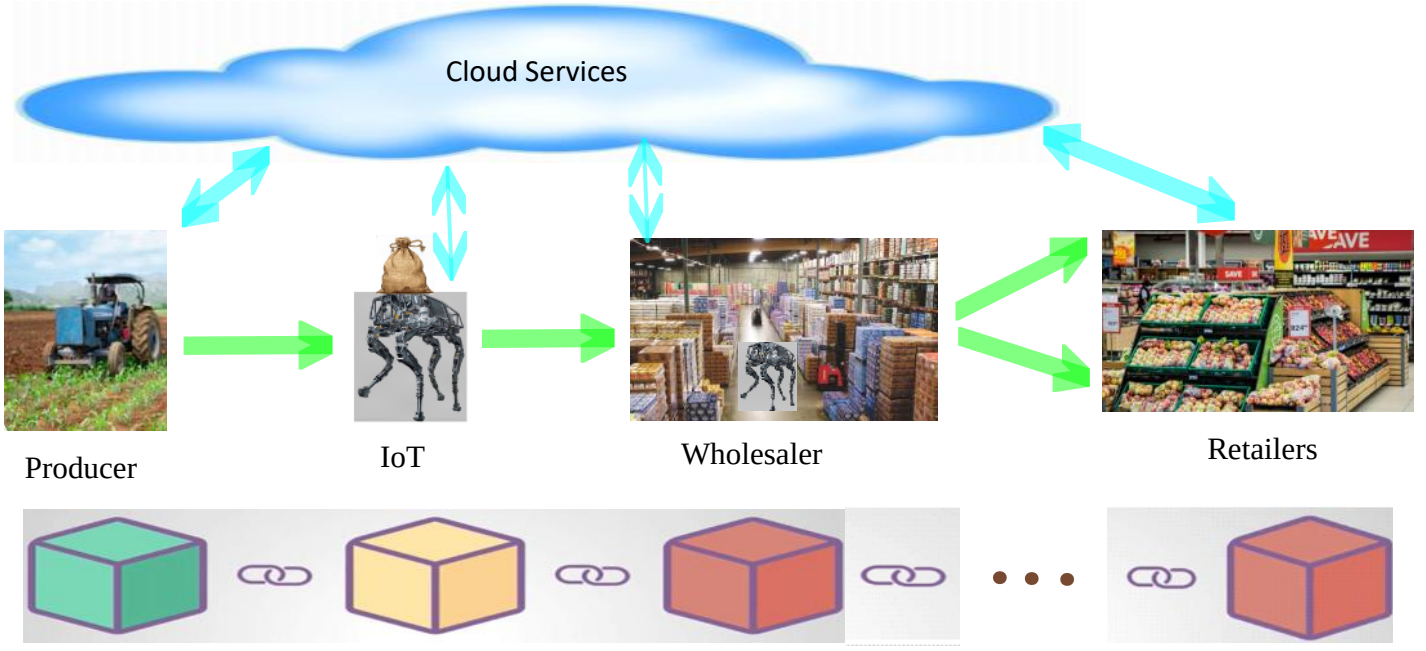
Cryptography provides information security in a wide area of applications, namely in blockchain and IoT technologies. The main blockchain application is for business processes monitoring and control. It is also associating with New Economic Movement and Industry Tokenization allowing to provide transparent Crowd Funding without brokers mediation.
Blockchain is also providing a transparent and trusted Business Value Chain integration with IoT technologies supported by tokens as a way to attract investors. IoT can be realized by robotics controlling physical events for Smart Contracts operation.

## Cryptography, Internet of Things (IoT), Blockchain, Cryptocurrency Business Value Chain




## Blockchain

Events are registeted as Oracles --> Oracles are transmitted to Smart Contracts -->
--> Smart Contracts are executing Transactions: for cryptocurrency transfer and/or payments for tokens.

Transactions can performe a **cryptocurrency transfer**
or **payments for tokens**.

## Bitcoin

Bitcoin blockchain is a **decentralized, (peer-to-peer digital monetary system)** cryptocurrency that uses cryptography to control its creation and management.
Bitcoin allows value to be transferred over the internet **without intermediaries** such as banks.
In 2008 Satoshi Nakamoto published the Bitcoin whitepaper: *"Bitcoin: A Peer-to-Peer Electronic Cash System"*

Bitcoin consists of three tightly connected layers:
1. **The currency (BTC)** – the unit of value, the smales unit of BTC is Satoshi: 1 BTC = 1 000 000 000 Sat.
2. **The network** – thousands-millions of computers (nodes) running Bitcoin software

3. **The protocol** – the rules that govern how Bitcoin works

Unlike traditional payment systems, Bitcoin removes trusted intermediaries such as banks and relies on:

- Blockchain
- Cryptographic verification
- Distributed consensus mechanism based on Proof-of-Work (PoW)
- Incentives for block mining (validation)

**Fundamental Problem Bitcoin Solves**

- Ownership verification
- Prevention of double spending
- Trustless operation

Before Bitcoin, the payment systems relied on:
**User → Bank → Ledger → Payment Confirmation**

This creates problems:

- central point of failure
- Censorship (cenzūra)
- trust dependency

Bitcoin replaces the central authority with cryptographic proof and distributed consensus mechanism.

**Core Problem: Double Spending**
Digital information can be copied two or many times. This creates the double spending problem.
**Example:**
**Alice owns 1 BTC**
Alice could make a copy of this bit string attempt to send the same coin to:
**Bob and Charlie**
Both transactions may appear valid unless a system ensures only one can succeed.

Bitcoin solves this problem using:

- ❖ **Cryptographic signatures and hash functions**
- ❖ **Public transactions ledger**
- ❖ **Timestamped blocks of transactions**
- ❖ **Proof-of-Work consensus mechanism**

**Mathematical Foundations of Bitcoin**
Bitcoin relies heavily on three mathematical domains:

| **Field** | **Role** |
|---|---|
| Number Theory | modular arithmetic in finite fields |
| Cryptography | signatures and hashing |
| Probability Theory | mining security and attacks |

Bitcoin ownership is controlled through public-key cryptography.

When a user spends coins, they must prove ownership of the private key corresponding to a public key
Is able to create a digital signature.

**Bitcoin originally uses: Elliptic Curve Digital Signature Algorithm - ECDSA** of **secp256k1** kindf of signatures and Hashing basedd on **SHA-256**

**A cryptographic hash function H(x) must satisfy the following security conditions:**

- Deterministic
- Preimage resistance
- Second preimage resistance
- ***Collision resistance***

**SHA-256 Function**

h = sha256(m)

Output:

256-bit digest

Bitcoin frequently uses double hashing:

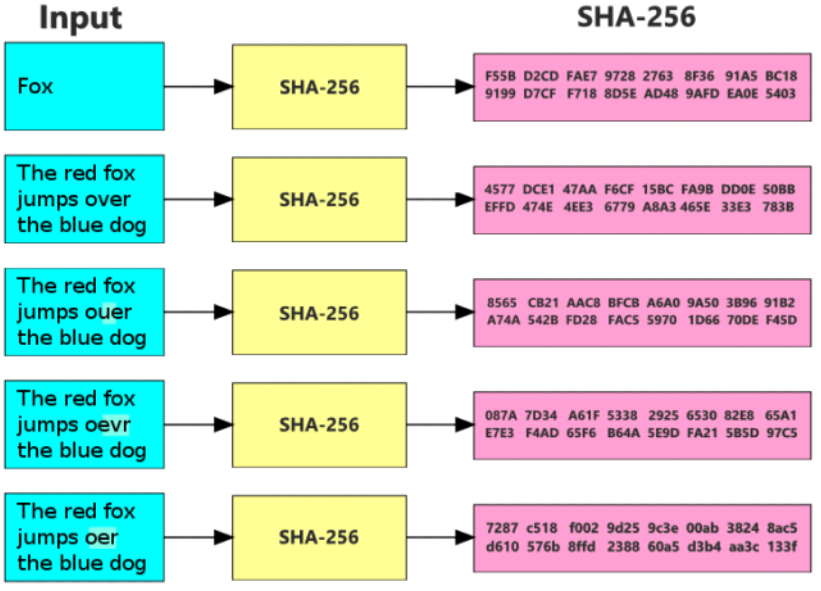
h = sha256(sha256(m))

>> sha256('Hello Bob')
ans = 2D6A12FFC0A952FBD09F8909DE4E0E4B20BA2B906CD12AC22BBEF4EE5BD9003E
>> h=sha256(ans)
h = 476D03360CCD1CF6F25B2B4F8038E9B526BF6F0F7386F2781CA92D31DC362873

Hash function is used for:

- Address computation
- Transaction IDs
- Transactions signing
- Block hashing
- Proof-of-Work (PoW)

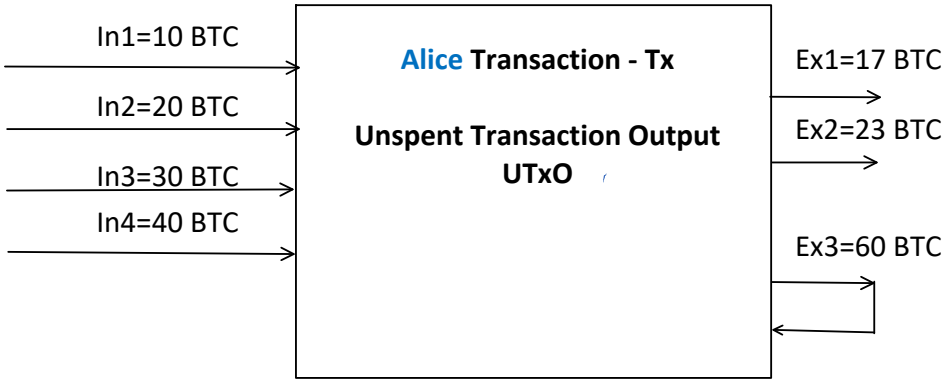
>> sha256('Fox')
ans = F55BD2CDFAE7972827638F3691A5BC189199D7CFF7188D5EAD489AFDEA0E5403




**Bitcoin:** H-function is **sha256**.
**Ethereum**: H-function is **keccak256**.

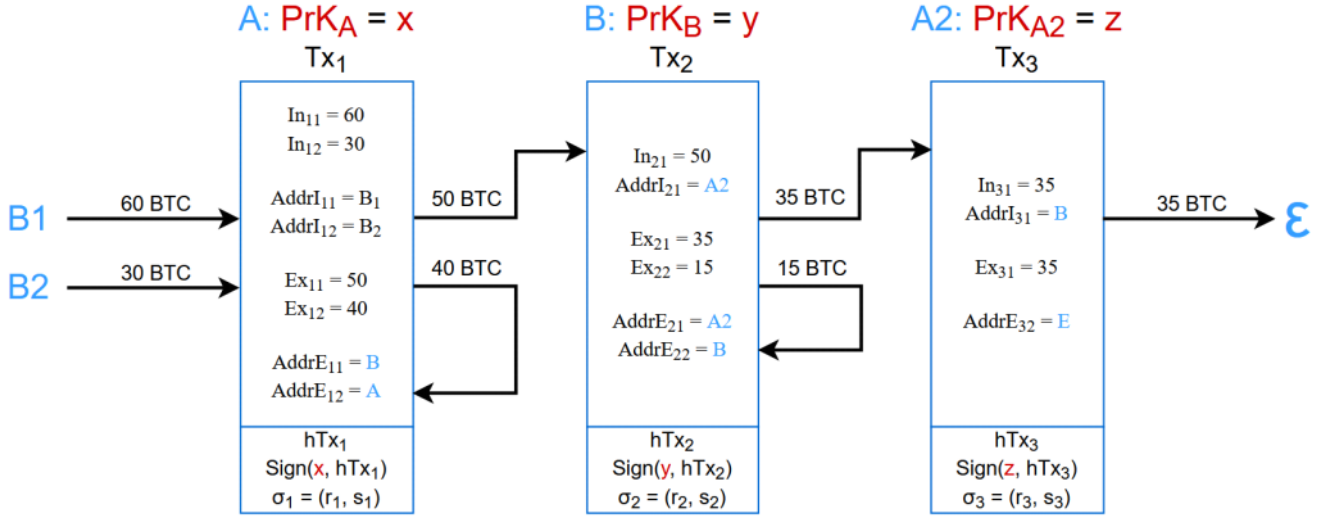
**How does Bitcoin Works:**

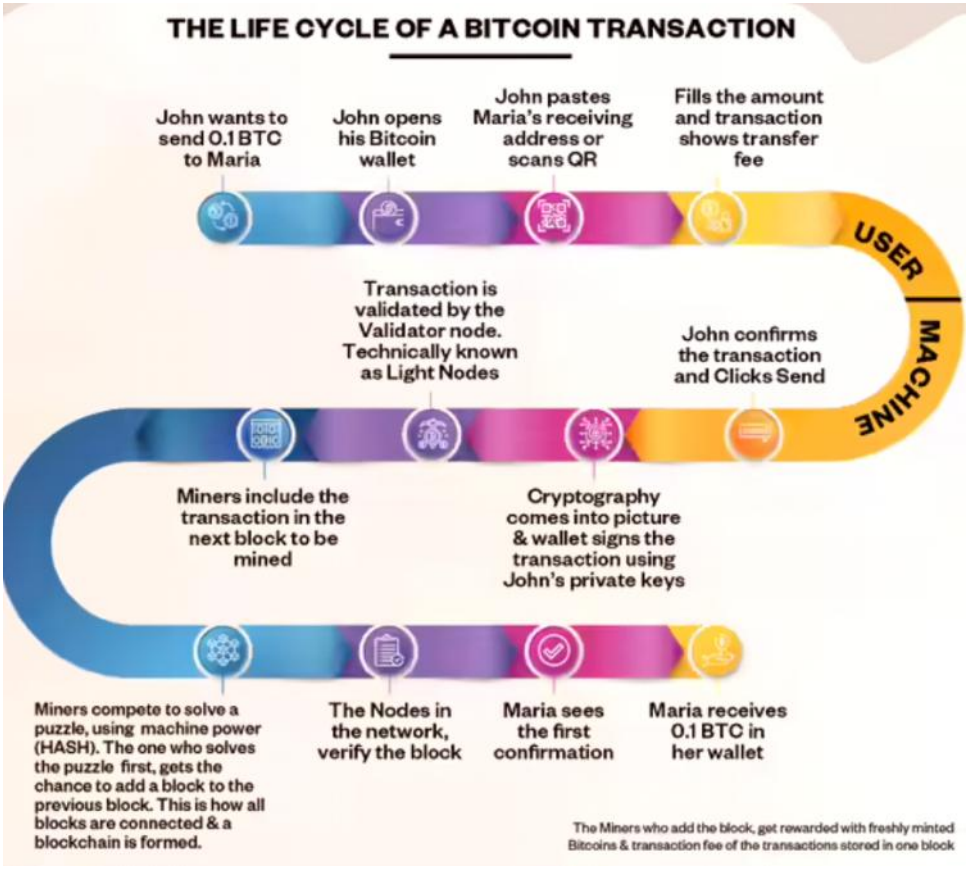
Bitcoin relies on blockchain and works through a sequence of processes combining cryptography, networking, distributed consensus mechanism, and incentives for mining.
Every time a transaction happens, it gets written onto a "block." That block is linked to the previous one, forming a chain. This record is copied onto thousands-millions of computers around the world (called **nodes**).
Also, anyone can participate in the ecosystem by [downloading Bitcoin's open-source software] and create bitcoin based applications.
Among the **nodes** there are **full node** in the Bitcoin network and thes **full nodes** independently stores a block chain containing only blocks validated by that node. When several nodes all have the same blocks in their block chain, they are considered to be in consensus. The validation rules of these nodes follow to maintain consensus are called consensus mechanism.
Bitcoin transactions are operating using so called **Unspent Transactions Output** - **UTxO** mechanism:
The sum of transaction Incomes (**In**) must be equal to the sum of transaction Expenses (**Ex**) or the sum of transaction Inputs must be equal to the sum of transaction Outputs.




Balance verification: In1 + In2 + In3 + In4 = Ex1 + Ex2 + Ex3 =
10 + 20 + 30 + 40 = 17 + 23 + 60 = 100

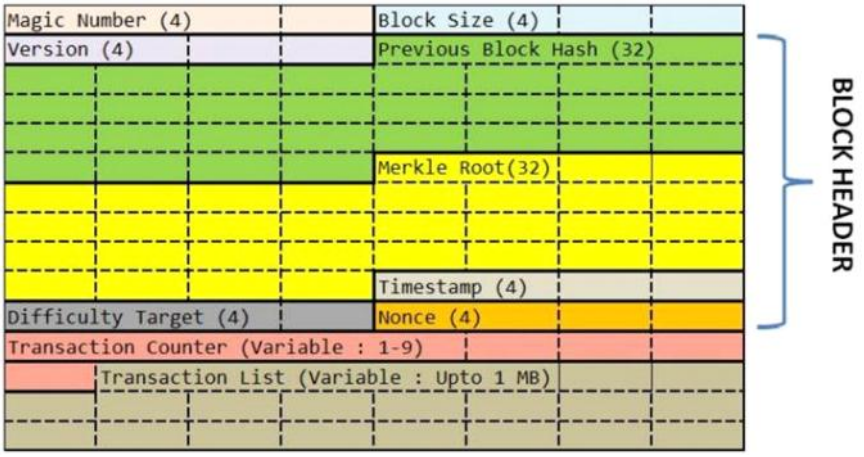
Transactions flow






Transactions are included in blocks

**Bitcoin block structure**

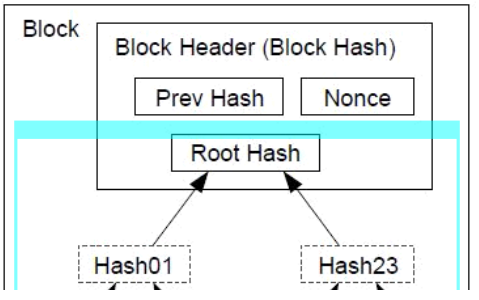



1 rectangular represents 1 Byte = 8 bits
32 Bytes == 64 hex numbers == 256 bits
**Nonce** is a number represented by 4 Bytes

Transactions list is represented by Merkle Tree which is represented by Root Hash.
This Root Hash is placed into the block's header along with the hash of the previous block and a random number called a nonce to be explained later.

Merkle tree: is a graph consisting of leafs representing transactions data, graph vertices

Antraštė

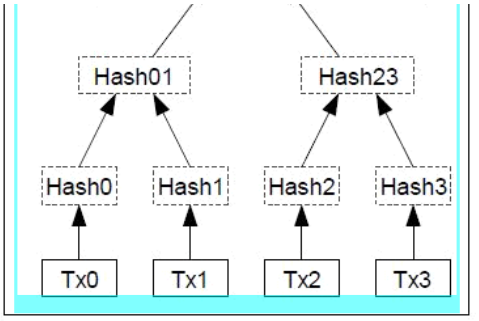



0 Level

1 Level == $2^1$

Merkle tree: is a graph consisting of leafs representing transactions data, graph vertices and edges.

The block's header is then hashed with sha256 producing an output that will serve as the block's identifier.



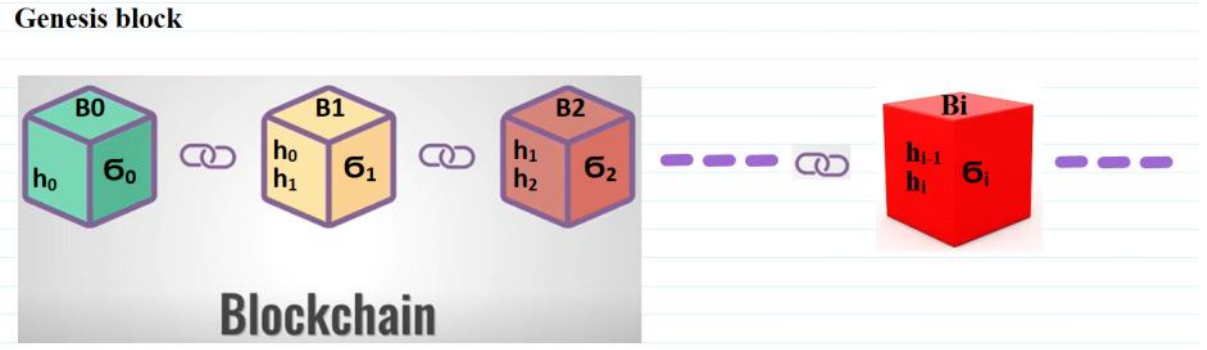

1 Level == $2^1$

2 Level == $2^2$

30 Level == $2^{30}$

Structure of a Bitcoin Transaction Blockheader.

| Field | Purpose | Updated when... | Size (Bytes) |
|---|---|---|---|
| Version | Block version number | When software upgraded | 4 |
| hashPrevBlock | 256-bit hash of the previous block header | A new block comes in | 32 |
| hashMerkleRoot | 256-bit hash based on all of the transactions in the block | A transaction is accepted | 32 |
| Time | Current timestamp as seconds since 1970-01-01T00:00 UTC | Every few seconds | 4 |
| Bits | Current target in compact format | The difficulty is adjusted | 4 |
| Nonce | 32-bit number (starts at 0) | A hash is tried (increments) | 4 |




Bitcoin block mining ir based on **PoW** consensus mechanism referencing to Difficulty Target - **DT**.
Symbolically it can represented in the following way:
>> sha256('Block header data hashed using the arbitrary **Nonce** ...')
ans = 41F1801FC7E4B36D44DA242013CE7C6E8C111D19B5FB8795EF718A1030AD6980
Difficulty Target - **DT** defines the number of most significant hexadecimal digits having zero values required to mine a block.
For example **DT**=18.
This means that block is mined if the **Nonce** value is found such that h-value of block header has 18 leading hexadecimal zeroes.
Let such a **Nonce** is found, then the block is mind if e.g. miner obtains the following h-value:
>> sha256('Block header data hashed using the concrete **Nonce** of 4 bytes')
ans = 000000000000000000E7B8F4C1AAAC8D2D85A326BEE0B9E242E0E360B09C1347

The full lifecycle using Elliptic Curve Cryptography (**ECC**) can be divided into the following stages:

**Key generation:** is based on Elliptic Curve Digital Signature Algorithm - **ECDSA**
**ECDSA** Public Parameters are the following **PP**=(EC **secp256k1**; BasePoint-Generator $G$; prime $p$; param. $a$, $b$);
Prime $p$ is of 256 bit length and defines the finite fielf $F_p = \{0, 1, 2, 3, \ldots, p\text{-}1\}$ where operations are performed **mod** $p$.

Parameters ***a***, ***b*** defines the equation of Elliptic Curve (EC) over the finite fielf $F_p$:
$y^2=x^3+ax+b \mod p$

**Private Key PrK = *z* generation.**

Is generated at random as a hexadecimal number having 64 hexadecimal digits ***z*=randi(*p*-1)**

Where ***p*** - is a large prime having 256 bit length.

**Public Key PuK = *A* generation.**

Derived mathematically from private key ***z***.

Bitcoin uses the elliptic curve: **secp256k1** it means that the number of Elliptic Curve (**EC**) points is equal to ***p***.

The public key is generated using **EC** generator ***G*** multiplication by number (scalar) ***z*** we denote by *:

**PuK = *A* = *z**G**

Public key ***A*** is an **EC** point and hence is represented by coordinates (*x*, *y*)

In **secp256k1** every coordinate ocuupies 256 bits.

Then The representation of ***A*** requires 512 bits.

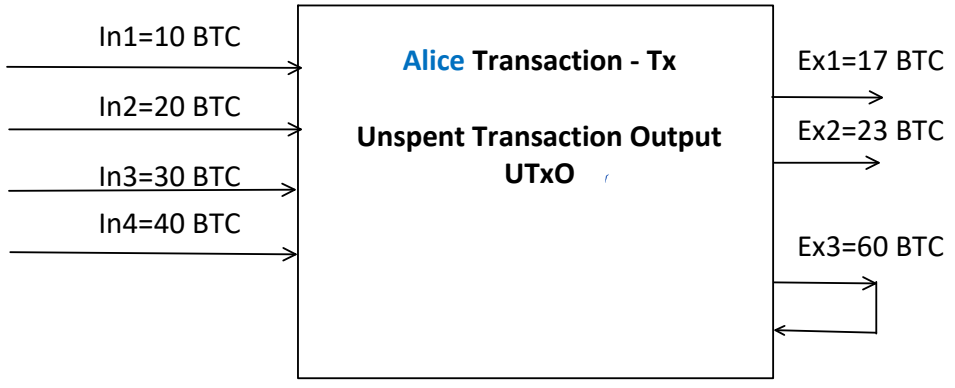
## Address creation

**Private Key z → Public Key *A* (EC multiplication operation *) → sha256(*A*) → Bitcoin Address by taking 34 least significant hexadecimal digits**

Let user's **PuK**=(*x*, *y*) is represented by the following hexadecimal numbers representing coordinates (*x*, *y*):
x = 52A6B8D367854D62EABF9F415A5E685055067270094729364l7C747610270624
y = 3EF4698C496E1475B7ECB6A2E4C85B6C7C1C57AA124DFCA859AE3B6CBB3F4A75

To compute Address **PuK** coordinates *x* and *y* must be hashed
>> h=sha256('52A6B8D367854D62EABF9F415A5E685055067270094729364l7C7476102706243EF4698C496E1475B7ECB6A2E4C85B6C7C1C57AA124DFCA859AE3B6CBB3F4A75')
h = 16A134BC52822E45372C9FE6411939B8B60747EAEA5FAF9AB40137FC97BD05F0

Bitcoin addresss is represented by 34 least significant bits of hased **PuK**
Bitcoin Address Example: B8B60747EAEA5FAF9AB40137FC97BD05F0




Balance verification: In1 + In2 + In3 + In4 = Ex1 + Ex2 + Ex3 =
10 + 20 + 30 + 40 = 17 + 23 + 60 = 100

Incomes are encrypted
n1 = $g^{In1}$ mod p --> Enc(a, i1,n1) = cIn1
n2 = $g^{In1}$ mod p --> Enc(a, i1,n1) = cIn2
n2 = $g^{In1}$ mod p --> Enc(a, i1,n1) = cIn3
n4 = $g^{In1}$ mod p --> Enc(a, i1,n1) = cIn4

Expenses are encrypted

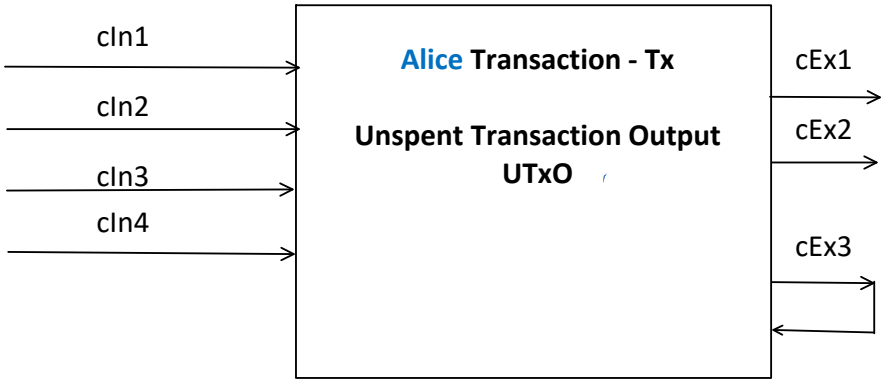
$x1 = g^{In1} \bmod p$ --> Enc(a, i1,x1) = cEx1

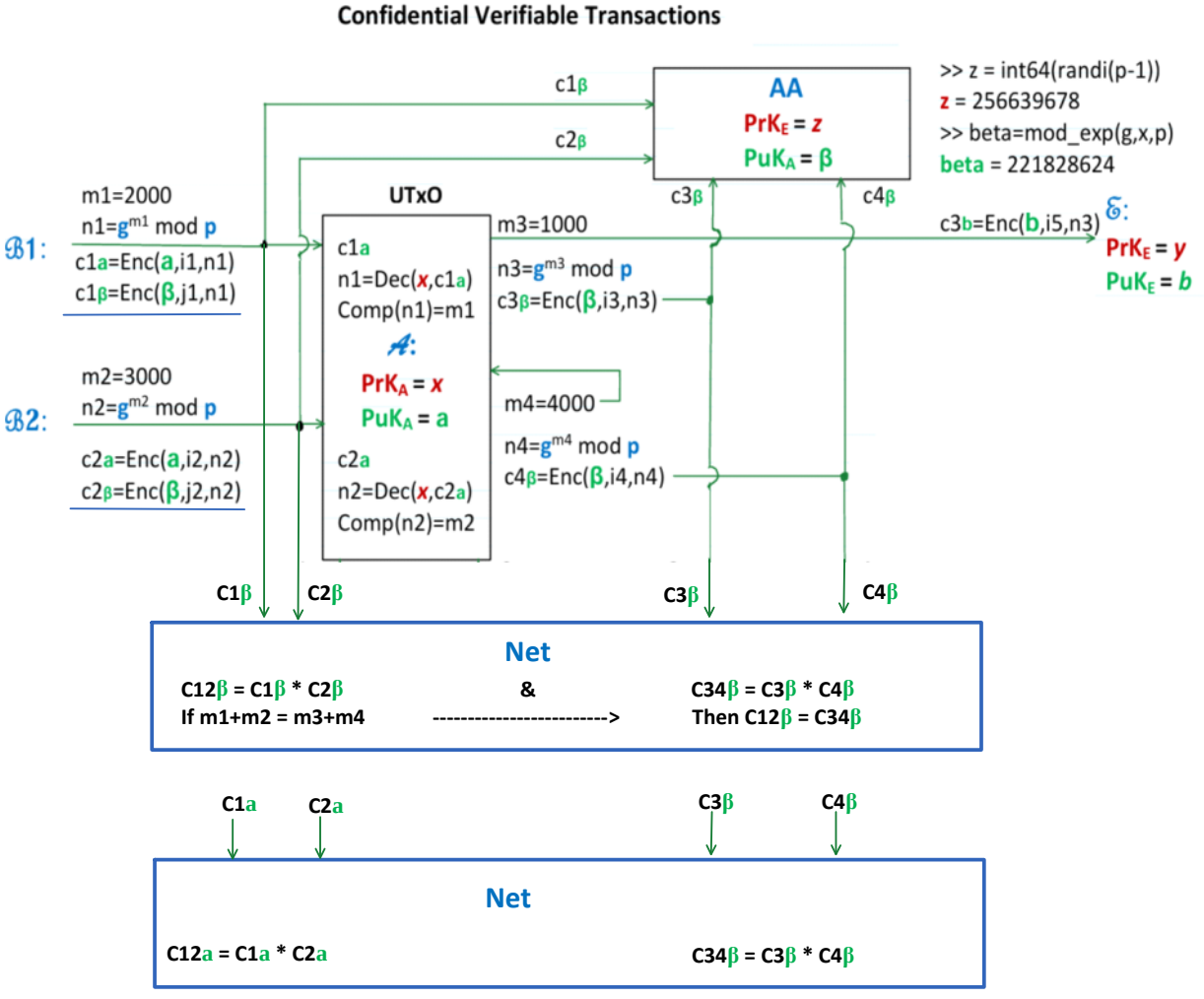
$x2 = g^{In1} \bmod p$ --> Enc(a, i1,x1) = cEx2

$x3 = g^{In1} \bmod p$ --> Enc(a, i1,x1) = cEx3

If balance is is valid, then:

cIn1 * cIn2 * cIn3 * cIn4 = cEx1 * cEx2 * cEx3




How to convince Net, that $C12a$ and $C34\beta$ encrypts the same sum of Incomes and Expenses:

2000 + 3000 = 5000 = 1000 + 4000 = 5000

**Diagrams & Visual References (With Links)**
You can paste the following diagrams directly into OneNote using the links.
**Bitcoin Transaction Flow Diagram**
Shows how a transaction moves from wallet → network → block → blockchain.

**UTXO Model Diagram**
Explains how inputs consume previous outputs and create new ones.
🔗 https://bitcoin.org/img/dev/en-utxo.png

**Block Structure Diagram**
Shows block header, transactions, nonce, and hashes.
🔗 https://learnmeabitcoin.com/static/img/blockchain/block.png

**Blockchain Linked Blocks Diagram**
Visual representation of how blocks are chained together.
🔗 https://bitcoin.org/img/dev/en-blockchain.png

**Merkle Tree Diagram**
Explains how transaction hashes are combined efficiently.
🔗 https://learnmeabitcoin.com/static/img/blockchain/merkle-tree.png

**Mining & Proof of Work Diagram**
Shows hashing, difficulty target, and nonce search.
🔗 https://www.researchgate.net/profile/Ashish-Jain-40/publication/330153005/figure/fig1/AS:711522451030016@1546632781160/Working-of-Bitcoin-Mining.png

**34. Authoritative Learning Links (Highly Recommended)**
**Official & Core Resources**
- Bitcoin Whitepaper (original): https://bitcoin.org/bitcoin.pdf
- Bitcoin.org Developer Guide: https://developer.bitcoin.org
- Bitcoin Core GitHub: https://github.com/bitcoin/bitcoin

**Technical Learning (Beginner → Advanced)**
- Learn Me A Bitcoin (excellent technical explanations): https://learnmeabitcoin.com
- Bitcoin Wiki: https://en.bitcoin.it/wiki/Main_Page
- Mastering Bitcoin (Andreas M. Antonopoulos – free online): https://github.com/bitcoinbook/bitcoinbook

**Blockchain Explorers (Live Data)**
- Blockchain.com Explorer: https://www.blockchain.com/explorer
- Mempool Visualizer: https://mempool.space