CHAPTER 3
BLOCKCHAIN PLATFORMS

To date, Blockchain has gone through an impressive development journey roughly divided into three generations, with each step being a significant improvement in functionality, performance and practical applications. Now let's take a look at the evolution and outstanding differences of each blockchain generation.

First generation: Bitcoin – Blockchain's foundation

Bitcoin, born in 2009, is considered the first generation of blockchain technology. As the first decentralized system, Bitcoin uses a Proof of Work (PoW) consensus mechanism to ensure security and transparency in transactions. However, PoW requires a huge amount of energy to solve complex algorithms, leading to high costs and large resource consumption. Bitcoin is primarily focused on storing value and financial transactions, with limited scalability and no programmability.

Second generation: Ethereum – Power from smart contracts

Ethereum, launched in 2015, marks the second generation of blockchain with programmability and smart contract support. The platform allows developers to create decentralized applications (DApps) on the blockchain, expanding application potential beyond the financial sector. However, Ethereum also inherits many of Bitcoin's limitations, including the use of energy-consuming proof of work and poor scalability. As the number of transactions increased, the Ethereum network became slow, with high transaction (gas) fees and an unstable user experience.

In September 2022, Ethereum switched to a proof of stake (PoS) mechanism as part of the Ethereum 2.0 roadmap. While this is an important step forward, Ethereum still faces many challenges, such as high gas fees and complex staking mechanisms, typically slashing and uncertainty around stake lock-up periods. .

Third generation: Cardano – Built from academic research

Cardano represents the third generation of blockchain, designed to address the limitations of previous platforms. Cardano uses proof of stake from the start, which offers significant energy savings compared to PoW. Cardano's system is built on academic research and mathematical proofs, ensuring security, scalability, and sustainability.

Not only does it overcome the energy weakness, Cardano is also designed with outstanding scalability and interoperability with other blockchains. Furthermore, Cardano has a built-in decentralized governance mechanism and budget, facilitating continuous development and adaptation to new future requirements. This is an important step that helps Cardano not only

fill Ethereum's gaps but also lay a solid foundation for the development of the global blockchain ecosystem.

Compare some characteristics of the platforms

Characteristic	**Bitcoin**	**Ethereum**	**Cardano**
Consensus mechanism	PoW	PoS (Ethereum 2.0)	PoS (Ouroboros)
Token Stake Status	Are not	Token Lock	Do not lock Tokens
Ledger model	UTxO	Accounting	EUTxO
Transaction speed	7 TPS	~20 TPS	250+ TPS
Block Time	10 minutes	12 seconds	20 seconds
Transaction costs	High, change	Higher gas fees, change	Lower fees, predictable
Smart contracts	Do not have	Mature, widely applied	Newly developed, focus on security
Developed ecosystem	Simple	Big, diverse	In development, focusing on practical applications

Developmental approach	Conservative development focuses on security	Dynamic development with continuous upgrades	Peer review and research-based development
On-chain governance	Are not	Are not	Have

3.1. First generation: Bitcoin – Blockchain's foundation

3.1.1. Introduction to Bitcoin

Bitcoin is the world's first decentralized electronic currency, created by an anonymous individual or group of people named Satoshi Nakamoto. Launched in 2009, Bitcoin is not only a digital currency but also marks the beginning of blockchain technology - the foundation for later decentralized applications.

Origin and history of development

Bitcoin was first introduced in a scientific document called "Bitcoin: A Peer-to-Peer Electronic Cash System" published by Satoshi Nakamoto in 2008. This document described a peer-to-peer electronic payment system (peer-to-peer) allows direct transactions between two parties without the need for an intermediary. On January 3, 2009, the first block of Bitcoin, called the Genesis Block, was mined, marking the beginning of a new financial era.

Satoshi Nakamoto created Bitcoin with the goal of solving the problems of the traditional financial system, especially the problem of "double spending" in digital transactions. Unlike traditional payment systems that depend on intermediaries such as banks or financial companies, Bitcoin allows direct transactions without the need for trust in third parties.

Outstanding features of Bitcoin

Bitcoin possesses outstanding features that make its name and influence in the world of finance and technology:

- **Decentralization**:
    - Bitcoin is not managed by any individual or organization.
    - All transactions and mining are processed by nodes in the peer-to-peer network.
- **Transparent**:

    - Every Bitcoin transaction is recorded on the blockchain, allowing anyone to audit it.
    - However, the identities of the transacting parties are kept secure thanks to the cryptographic address system.
- **High security**:
    - Bitcoin uses the SHA-256 encryption algorithm to protect data and transactions.
    - Blocks in the blockchain are linked together using cryptographic hash functions, creating an unmodifiable chain.
- **Limited supply**:
    - Bitcoin has a maximum supply of 21 million units, expected to be fully mined by 2140.
    - This scarcity makes Bitcoin a store of value similar to gold.

The role of Bitcoin in the blockchain ecosystem

Bitcoin has opened a new era in technology and finance, laying the foundation for the development of thousands of other cryptocurrencies and blockchain applications. Some important roles of Bitcoin include:

- **Pioneer in blockchain technology**:
    - Bitcoin was the first application of blockchain technology, introducing the Proof of Work (PoW) consensus mechanism.
    - This mechanism ensures the integrity and security of transactions without the need for a third party.
- **Value storage tool**:
    - With limited supply and scarcity, Bitcoin is likened to "digital gold".
    - Many investors see Bitcoin as an inflation hedge and long-term store of value.
- **Global payment system**:
    - Bitcoin allows for fast cross-border transactions at lower costs than traditional systems.
    - This is especially useful for countries with underdeveloped financial systems.
- **Platform for innovation**:
    - The success of Bitcoin has fueled the development of other blockchain platforms such as Ethereum, Cardano, and Binance Smart Chain.
    - Decentralized applications (dApps) and smart contracts are both inspired by Bitcoin.

Challenges and limitations of Bitcoin

Despite its many advantages, Bitcoin also faces some major challenges:

- **Scalability**:

    - The Bitcoin network can only process about 4-7 transactions per second, much lower than traditional payment systems like Visa. This is due to the block size of 1 MB and block generation time of about 10 minutes.
    - This leads to slow transaction processing times and high transaction fees during times of network congestion.
- **Energy consumption**:
    - The POW(Proof of Work) consensus mechanism requires a large amount of energy to mine, causing environmental concerns.
- **Legality and acceptance**:
    - Bitcoin is still not widely recognized or accepted in many countries.
    - Some governments are concerned about the use of Bitcoin in illegal activities.

3.1.2. Bitcoin's operating principles and transaction mechanism

The Bitcoin system, unlike traditional banking and payment systems, is based on decentralized trust. Instead of having a trusted central authority, in Bitcoin, trust is formed as a property that emerges from the interactions between different components in the Bitcoin system. In this chapter, we will look at Bitcoin from an overview perspective by following a single transaction through the Bitcoin system, observing how that transaction becomes “trusted” and accepted through the mechanism. Bitcoin's distributed consensus is ultimately recorded on the blockchain – a distributed ledger of all transactions. The following chapters will delve into the technology behind trading, networks, and mining.

Overview model of Bitcoin

In the overview diagram illustrated in Figure 3-1, we see that the Bitcoin system consists of users with wallets that hold keys, transactions that are transmitted on the network, and miners that create the blockchain. consensus through competitive computation. This blockchain is the official ledger of all transactions.

Each example in this chapter is based on an actual transaction performed on the Bitcoin network, simulating interactions between users (Joe, Alice, Bob, and Gopesh) transferring funds from one wallet to another. While tracking a transaction through the Bitcoin network to the blockchain, we will use a blockchain explorer site to visualize each step. Blockchain Explorer is a web application that acts as a Bitcoin search engine, allowing you to search for addresses, transactions, and blocks, and view the relationships and flows between them.

Figure 3-1. Bitcoin overview

Popular blockchain explorers include:

- **Bitcoin Block Explorer: https://www.blockexplorer.com/**
- **BlockCypher Explorer: https://live.blockcypher.com/**
- **https://blockchain.info**

Each of these explorers has a search function, allowing you to enter a Bitcoin address, transaction hash, block number or block hash to retrieve corresponding information. response from the Bitcoin network. For each example transaction or block, we will provide a URL so you can look it up and study the details yourself.

Bitcoin Trading

Simply put, a Bitcoin transaction is a notification to the network that one owner has agreed to transfer an amount of his Bitcoin value to another owner. The new owner can then continue to use these Bitcoins by creating a new transaction, allowing the value to be transferred to another person, forming a continuous chain of ownership.

Transaction inputs and outputs

Bitcoin transactions can be visualized as lines in a double-entry ledger. Each transaction consists of one or more "inputs", similar to debits from a Bitcoin account. On the opposite side, transactions have one or more "outputs", like credits to a Bitcoin account. However, the total quantities of input and output are not necessarily equal. Typically, the sum of the outputs

will be slightly less than the sum of the inputs; this difference is considered a transaction fee, a small payment collected by the miner when they add this transaction to the ledger. . A Bitcoin transaction is represented as an entry in the ledger illustrated in Figure 3-2.

In addition, each transaction contains proof of ownership of the amount of Bitcoin used (input) in the form of the owner's digital signature. This signature can be independently verified by anyone. In the context of Bitcoin, "using" means signing a new transaction to transfer value from a previous transaction to a new owner, identified through their Bitcoin address.

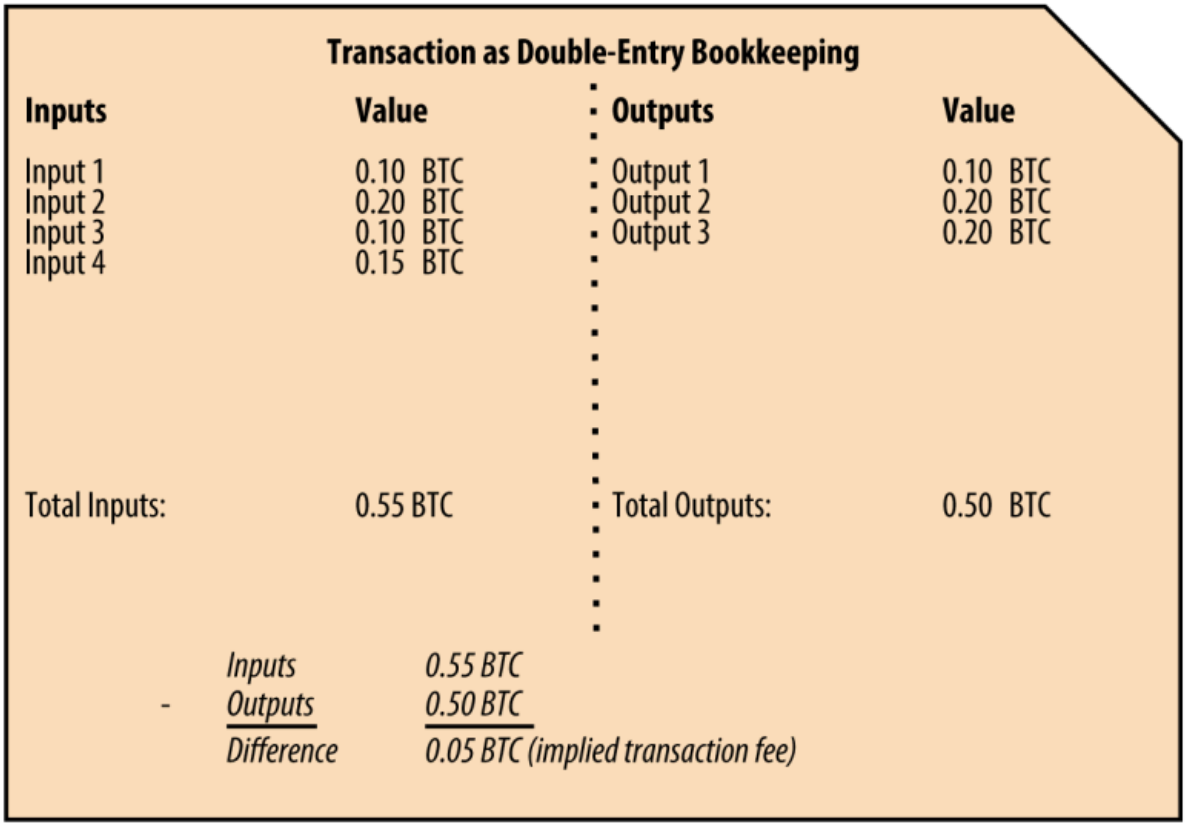


Figure 3-2. Transactions like double-entry bookkeeping

Transaction Chains

For example: When Alice pays Bob's coffee shop, she uses the output from a previous transaction as input. Suppose previously, Alice received Bitcoin from her friend Joe. This transaction created a Bitcoin value locked with Alice's key. In the new transaction, Alice references the previous transaction as input, and creates new outputs to pay for the coffee and receive the change back.

These transactions form a linked chain, in which the inputs of the current transaction are tied to the outputs from previous transactions. Alice's key serves as a digital signature, unlocking the outputs of the previous transaction, thereby proving to the Bitcoin network that she is the rightful owner of the funds. Alice then attaches the payment for the coffee to Bob's address,

"binding" the output with the requirement that Bob provide his digital signature to use the money. This process clearly illustrates the transfer of value from Alice to Bob.

The sequence of transactions, from Joe to Alice and then to Bob, is illustrated in Figure 3-3.

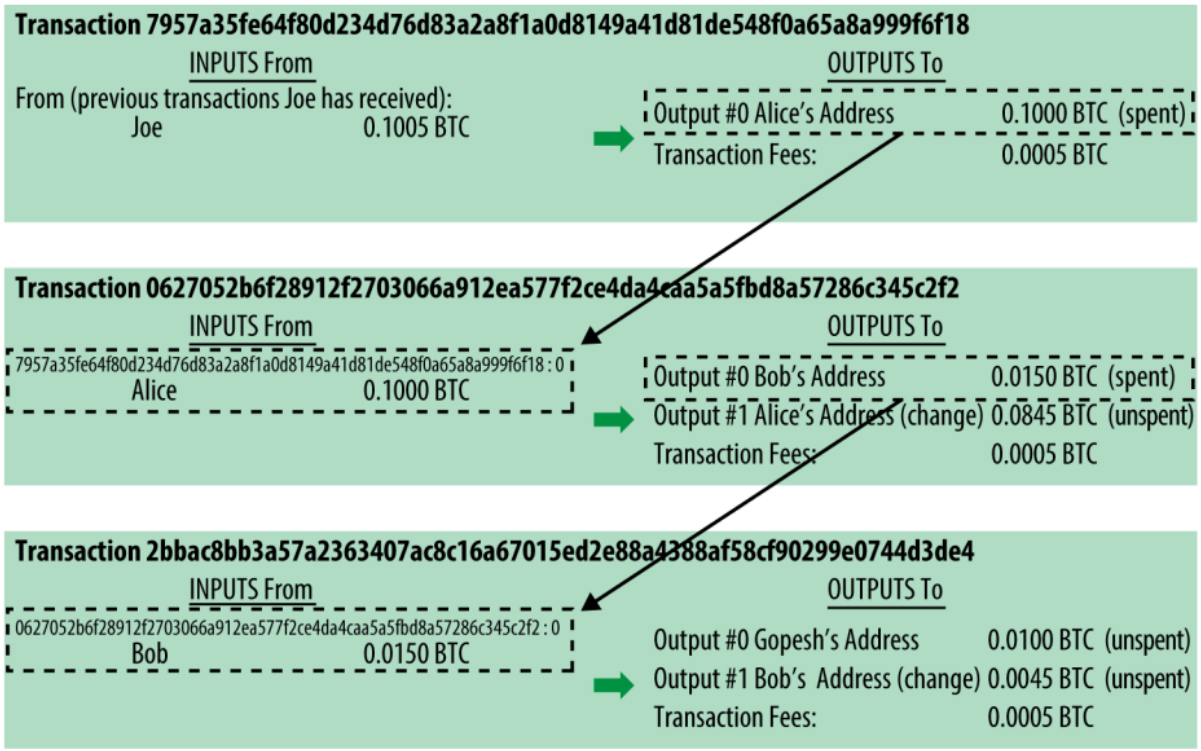


Figure 3-3. Chain of transactions

Add Transaction to Ledger

The transaction generated by Alice's wallet application is 258 bytes in size, containing all the information necessary to confirm ownership of the funds and assign the new owner. Now, this transaction needs to be transmitted to the Bitcoin network to become part of the blockchain.

In the next section, we will learn how a transaction becomes part of a new block and how this block is "mined". Finally, we will see how new blocks, once added to the blockchain, gradually gain increasing trust from the network as more new blocks are added.

Transmit Transaction

Since a transaction contains all the information needed for processing, it does not matter how or from where the transaction is transmitted to the Bitcoin network. The Bitcoin network is a peer-to-peer network, in which each Bitcoin client participates by connecting to several other Bitcoin clients. The purpose of this network is to transmit transactions and blocks to all participating members.

How Trading Goes Viral

Any system, such as a server, desktop application, or wallet, that participates in the Bitcoin network using the Bitcoin protocol is called a **Bitcoin node** (bitcoin node). Alice's wallet app can send new transactions to any Bitcoin node it is connected to, via any type of connection: wired, WiFi, cellular, etc.

Alice's Bitcoin wallet does not need to be directly connected to Bob's Bitcoin wallet, nor does she need to use the coffee shop's internet connection, although both of these methods are possible.

Any Bitcoin node that receives a valid transaction that it has never seen before will immediately forward that transaction to all other nodes it is connected to. This is a so-called transmission technique **flooding** (spread). As a result, transactions quickly spread throughout the peer-to-peer network, reaching a large portion of the nodes within seconds.

Mining Bitcoins

Alice's transaction process is now transmitted on the Bitcoin network. However, this transaction only becomes part of the blockchain when it is verified and included in a block through a process called mining.

Bitcoin's trust system is based on computation. Transactions are bundled into blocks, which require a large amount of computation to prove, but only a small amount of computation to verify their correctness. The mining process serves two main purposes in Bitcoin:

- Mining nodes verify all transactions based on Bitcoin consensus rules. Mining therefore ensures the security of transactions by eliminating invalid or malformed transactions.
- Mining creates new Bitcoins in each block, similar to how central banks print new money. The number of Bitcoins created in each block is limited and decreases over time, following a fixed release schedule.

Mining strikes a delicate balance between costs and rewards. This process consumes electricity to solve a mathematical problem. A successful miner will receive rewards in the form of new Bitcoins and transaction fees. However, rewards are only received if miners properly verify all transactions according to consensus rules. This balance provides security for Bitcoin without the need for a central authority.

An easy way to describe mining is like a giant sudoku game, constantly competing and resetting every time someone finds a solution. The game's difficulty is automatically adjusted so that it takes about 10 minutes to find the solution. Imagine a giant sudoku board, with thousands of rows and columns. If you give a complete table, others can check it very quickly. However, if the table only has a few cells filled in and the rest is blank, it will take a lot of effort to solve! The difficulty of sudoku can be adjusted by changing its size, but is still easy to verify even when the board is very large.

Spending Transactions

Once Alice's transaction is recorded on the blockchain, it becomes part of the Bitcoin distributed ledger and is visible to all Bitcoin applications. Full-nodes can verify transactions are valid by tracking the origin of Bitcoins from their creation in the first block, through each transaction to Bob's address. Meanwhile, lightweight clients use simple payment verification (SPV), transaction confirmations that reside on the blockchain and are supplemented by many other blocks, ensuring that miners have accept it.

Now, Bob can spend the Bitcoins received from Alice or other transactions. For example, Bob can transfer payments from Alice to suppliers or contractors. Bob's Bitcoin software often aggregates many small transactions throughout the day into one large transaction, consolidating into a single output and address.

As Bob spends these Bitcoins, the transaction chain continues to expand. For example, if Bob pays web designer Gopesh, this transaction becomes part of the chain from Alice to Gopesh. A chain of transactions illustrates how value moves from person to person, continuously building on the blockchain.

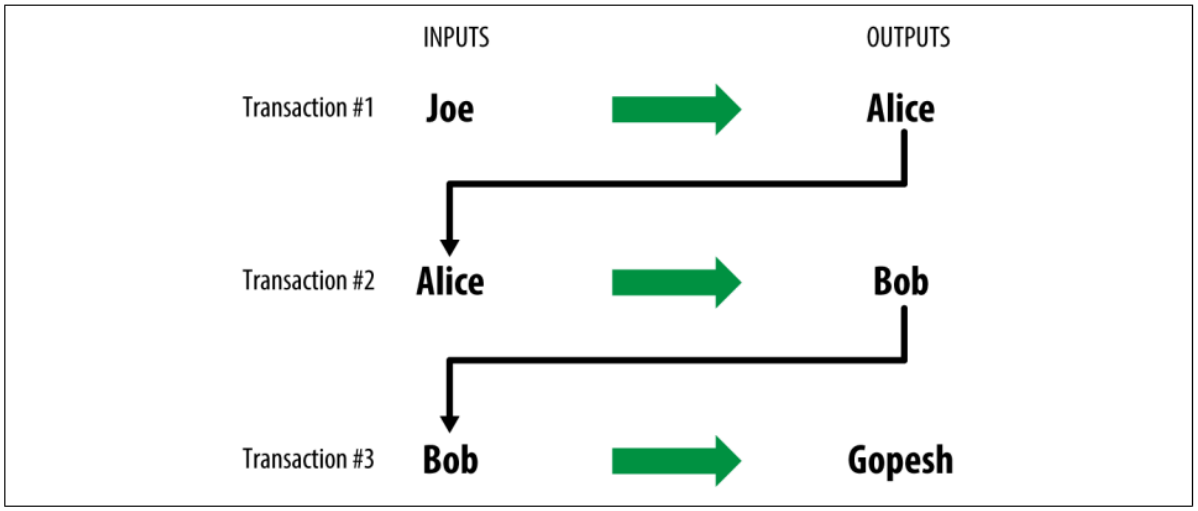


Figure 3-4: Alice's transaction is part of a chain of transactions from Alice to Gopesh.

3.1.3. Keys, addresses and wallets

Bitcoin ownership is determined through digital keys, Bitcoin addresses and digital signatures. Digital keys are generated and stored in the user's wallet, independent of the Bitcoin network. They offer features such as decentralized control, ownership authentication, and cryptographic security.

Every Bitcoin transaction needs a valid digital signature, generated from a private key. Whoever owns the private key can control the Bitcoins involved. The key pair includes a private key (like a PIN) and a public key (like an account number), usually managed by wallet software. Users only need to care about the Bitcoin address – a form of digital fingerprint of the public key – to receive money.

Bitcoin addresses simplify transactions, similar to the recipient name on a check, and can represent complex scripts. We'll learn how to create, store, and manage keys, encryption formats, and advanced applications like multi-signature addresses or paper wallets.

Public Key Cryptography

Public key cryptography was invented in the 1970s, laying the mathematical foundation for computer and information security. Since then, many suitable mathematical functions such as prime number exponentiation and elliptic curve multiplication have been discovered. These functions are virtually invertible, meaning they are easy to calculate in one direction but very difficult to calculate in reverse. Based on these functions, cryptography allows the creation of digital secrets and digital signatures that cannot be forged. Bitcoin uses elliptic curve multiplication as the basis for its cryptography.

In Bitcoin, public key cryptography is used to create a key pair that controls access to Bitcoin. Key pairs include:

- **Private key:** used to sign transactions when spending Bitcoin.
- **Public key:** is generated from a secret key and used to receive Bitcoin.

The mathematical relationship between the public key and the private key allows the private key to be used to create a digital signature. This signature can be verified using the public key without revealing the private key.

When spending Bitcoin, the current owner presents the public key and signature (the signature changes each time but is generated from the same private key) in the transaction. Thanks to this, the entire Bitcoin network can verify the transaction is valid and confirm the sender owned the Bitcoin at the time of the transaction.

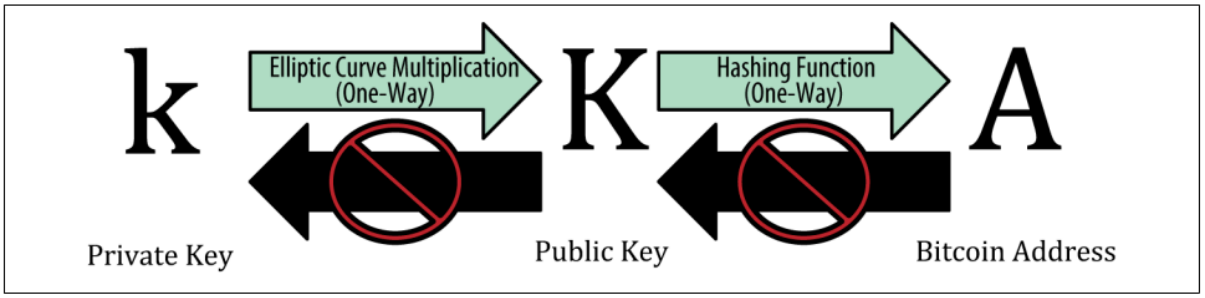
In most Bitcoin wallets, the private key and public key are stored together as a key pair for convenience. However, since the public key can be computed from the private key, it is also possible to simply store the private key.

Secret Key and Public Key

A Bitcoin wallet contains a set of key pairs, each consisting of:

- **Secret key (k):** is a number, usually chosen at random.
- **Public key (K):** generated from the secret key via elliptic curve multiplication, a one-way cryptographic function.

From the public key (K), a one-way cryptographic hash function will be used to generate the Bitcoin address (A). The relationship between the private key, public key, and Bitcoin address is illustrated in Figure 3-5.



Hình 3-5: Private key, public key, and bitcoin address

Bitcoin operates based on two important types of keys: **secret key** and **public key**. These two types of keys help ensure safety, authenticate ownership, and effectively manage Bitcoin assets.

Secret key is a very large random number, generated by the Bitcoin wallet software. You can imagine it like a personal password that only you know. The private key acts as the master key to control and spend Bitcoin.

Secret keyword, the system will generate **public key**. This is a form of information that can be shared publicly, like the bank account number you give others to send you Bitcoin. The public key is generated from the private key through a special mathematical operation, called **elliptic curve multiplication**. The special thing here is that this math operation only works in one direction: the secret key can create the public key, but no one can do the opposite to find the secret key from the public key.

From the public key, the system continues to perform another cryptographic operation, called **hash function**, to create **Bitcoin address**. This address is where you receive Bitcoin from others.

Specifically, the process is as follows: the Bitcoin wallet software will generate a random number (secret key). From there, it applies elliptic curve multiplication to generate the public key. Finally, from the public key, the system applies a hash function to generate a Bitcoin address.

This system is very secure because the operations only work in one direction. That means, even if someone knows your public key or Bitcoin address, they cannot find out your private key. Only the holder of the private key can create a digital signature to confirm ownership and spending of Bitcoin.

In other words, the private key is the only key that gives you control over your Bitcoin assets, while the public key and Bitcoin address are information that can be shared without compromising security. Thanks to this mechanism, Bitcoin ensures transparency and safety for the entire system.

Bitcoin Address

A Bitcoin address is a string of characters used to receive money from others. Generated from public keys, Bitcoin addresses usually start with the number "1", Example of a Bitcoin address: **1J7mdg5rbQyUHENYdx39WVWK7fsLpEoXZy** and can be compared to the "payee" on a paper check. It represents where funds are received in a Bitcoin transaction.

The address generation process starts from the public key. This key is hashed twice: first using the SHA256 algorithm, and second using RIPEMD160, creating a 160-bit string called the Bitcoin address.

For ease of use and to avoid confusion, Bitcoin addresses are often encoded with Base58Check. This system makes addresses easier to read, avoids confusion between similar characters, and adds checksums to detect input errors.

A Bitcoin address is not a public key but a shortened version and is secured through hash functions. It is the only part users need to share to receive money.

The Base58Check encoding and decoding mechanism as well as the resulting representations are depicted in Figure 3-6. It illustrates the process of converting a public key into a Bitcoin address.

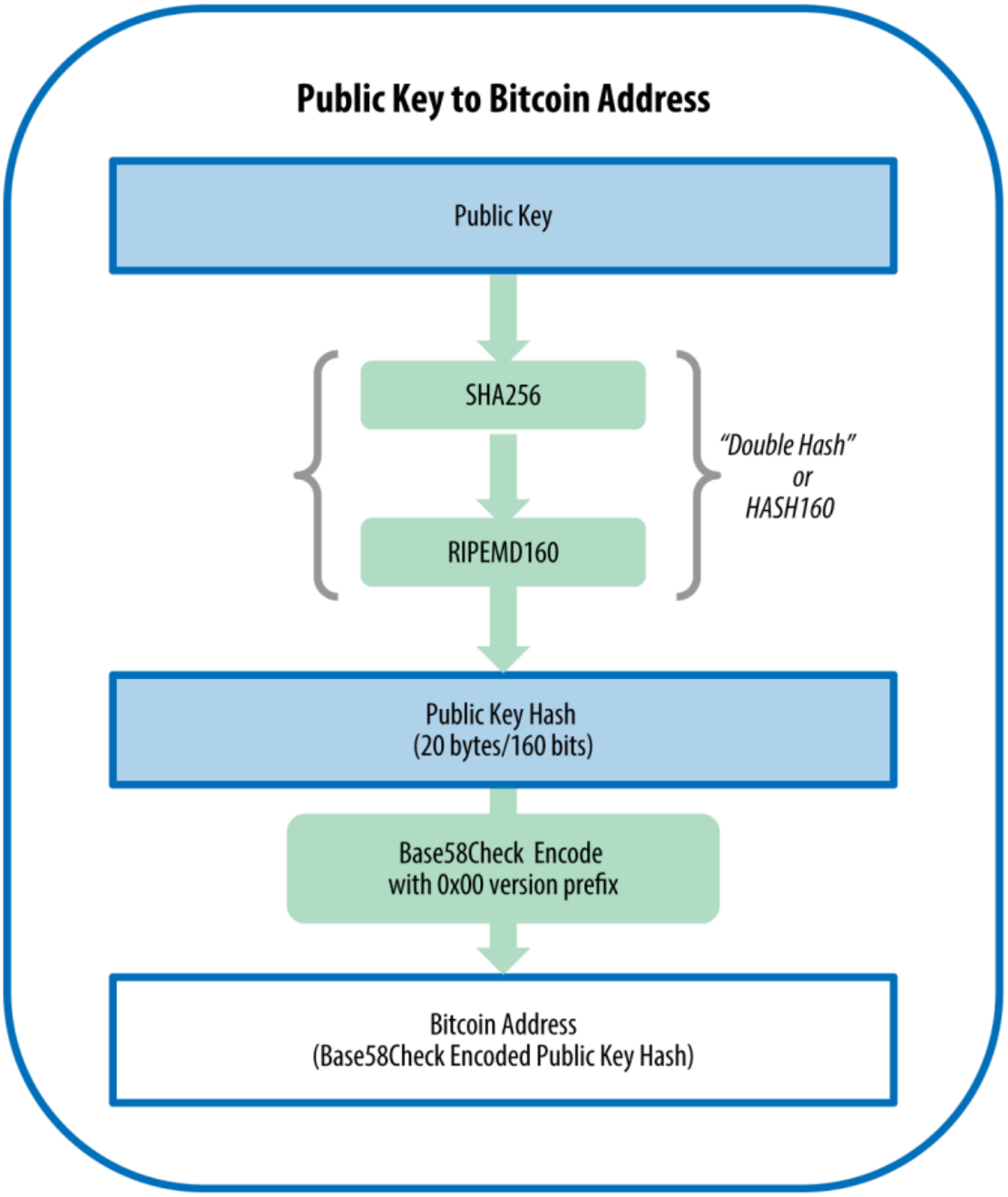


Figure 3-6: Describes the process of converting public keys into Bitcoin addresses

To better understand the relationship between private key, public key and Bitcoin address, you can learn through BIP-38. BIP-38 is a proposed standard (Bitcoin Improvement Proposal) used to encrypt a private key with a password (passphrase), then encrypt the result using Base58Check. The main purpose of BIP-38 is to increase the security of private keys, especially when they are stored in insecure places such as on paper, USB, or when moving between wallets. By encrypting the private key, even if someone gets a copy of it, they cannot use it without the decryption password. This helps protect your Bitcoin assets from the risk of theft due to private key exposure. In short, BIP-38 provides a secure and convenient method of protecting private keys by encrypting them.

3.1.4. Bitcoin Network

Peer-to-Peer (P2P) Network Architecture

Bitcoin is built on a peer-to-peer (P2P) network architecture that operates on the internet platform. The term "peer-to-peer" means that computers participating in the network are equal, there are no "special" nodes, and all nodes share responsibility for providing network services. Nodes in the network are connected in a mesh with a "flat" structure, with no central server, no centralized services, and no hierarchy in the network.

In a P2P network, nodes both provide and consume services, with reciprocity serving as the driving force for participation. P2P networks are sustainable, decentralized and open. A typical example of P2P network architecture is the early internet, where nodes in an IP network operated equally. Although today's internet architecture is more hierarchical, the IP protocol still retains its flat structural nature. Outside of Bitcoin, the most successful application of P2P technology is file sharing, with Napster being the pioneer and BitTorrent being the most recent evolution.

Bitcoin's P2P network architecture is not just a structural choice, but also reflects and supports Bitcoin's core characteristic: a decentralized digital cash system. Bitcoin's key design principle is decentralization of control, which can only be achieved and maintained through a flat, decentralized P2P network based on consensus.

The term "Bitcoin network" refers to the collection of nodes running Bitcoin's P2P protocol. In addition to Bitcoin's P2P protocol, there are other protocols such as Stratum used for mining and lightweight wallets or mobile wallets. These additional protocols are provided by gateway routing servers that connect to the Bitcoin network via the P2P protocol and then extend this network to nodes using other protocols.

For example, Stratum servers connect mining nodes using the Stratum protocol to the main Bitcoin network and serve as a bridge between the Stratum protocol and Bitcoin's P2P protocol. The term "Bitcoin extended network" is used to refer to the entire network, including Bitcoin's P2P protocol, pool-mining protocols, Stratum protocol, and other related protocols that connect components of the Bitcoin system.

Node Types and Roles

Although nodes in Bitcoin's P2P network are considered equal, they can take on different roles based on the functions they support. A Bitcoin node is a combination of functions: routing, blockchain database, mining, and wallet service. A complete button with all four of these functions is shown in Figure 8-1.

- **Routing function**

All nodes include routing functionality to join the network and may have additional functionality. Nodes perform verification, propagate transactions and blocks, as well as find

and maintain connections with peer nodes. In the full node example, the routing function is represented by the orange circle labeled “Network Routing Node” or the symbol “N.”

- **Full Node**

Some nodes, called full nodes, maintain a complete and updated copy of the blockchain. These nodes can automatically and independently verify any transaction without the need for external references. In the illustration, full blockchain database functionality is represented by the blue circle labeled “Full Blockchain” or the symbol “B.”

- **Nút SPV (Simplified Payment Verification)**

Some nodes maintain only part of the blockchain and verify transactions using a method called Simplified Payment Verification (SPV). These nodes are often called SPV nodes or lightweight nodes. In the illustrations, SPV nodes are represented without the blue circle, indicating they do not have a full copy of the blockchain.

- **Mining Node**

Mining nodes compete to produce new blocks by running specialized hardware to solve the Proof-of-Work algorithm. Some mining nodes are also full nodes, maintaining a full copy of the blockchain, while others are light nodes that participate in pool mining and depend on the pool server for maintenance. maintain a full node. The mining function is indicated by the black circle labeled “Mining” or the symbol “M.”

- **User Wallets**

The user wallet can be part of a full node, as is common in desktop Bitcoin clients. However, an increasing number of user wallets, especially on devices with limited resources such as smartphones, are SPV nodes. Wallet functionality is indicated by the green circle labeled “Wallet” or the symbol “W.”

- **Other Nodes and Servers**

In addition to the main node types in Bitcoin's P2P protocol, there are also servers and nodes running other protocols, such as mining pool protocols and lightweight client access protocols. -access).

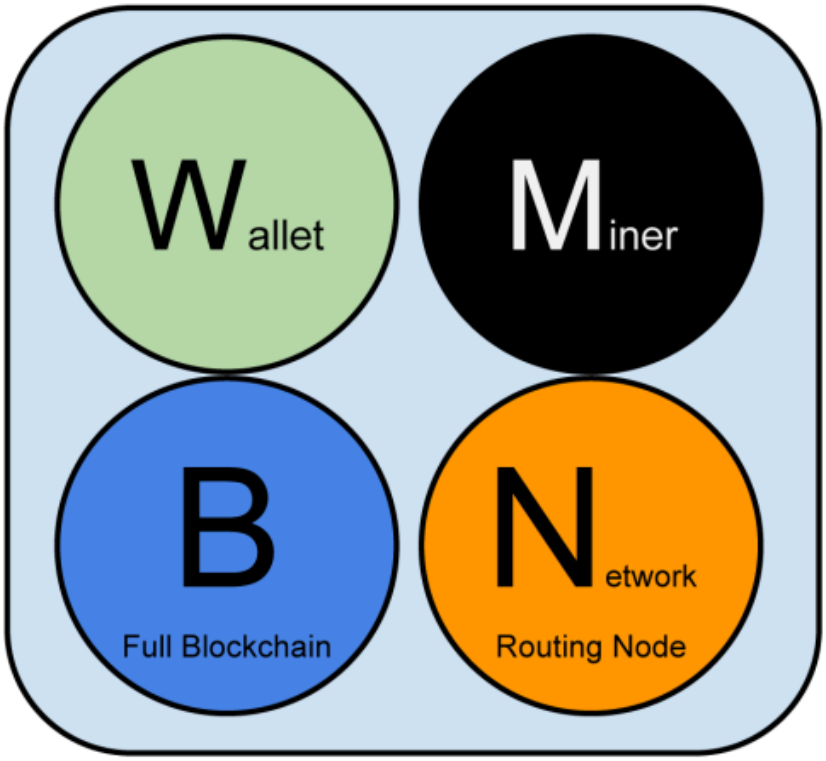


Figure 3-7: A Bitcoin network node with all four functions: wallet, miner, full blockchain database, and network routing.

Expanded Bitcoin Network

The main Bitcoin network, running Bitcoin's P2P protocol, consists of between 5,000 and 8,000 listening nodes running various versions of the Bitcoin reference application (Bitcoin Core) and a few hundred nodes running other versions of the protocol. P2P Bitcoin formats such as Bitcoin Classic, Bitcoin Unlimited, BitcoinJ, Libbitcoin, btcd, and bcoin. A small percentage of nodes in the Bitcoin P2P network are mining nodes, participating in the mining process, validating transactions, and generating new blocks.

Many large companies connect to the Bitcoin network by running full node clients based on the Bitcoin Core application, with a full copy of the blockchain and network node functionality, but no mining or wallet functionality. These nodes act as edge routers of the network, allowing other services (exchanges, wallets, block explorers, trade payment processing) to be built on top of them.

The extended Bitcoin network includes the network running Bitcoin's P2P protocol as described above, as well as nodes running other specialized protocols. Attached to Bitcoin's main P2P network are a number of pool servers and protocol gateways that connect nodes running other protocols. These nodes are mainly pool mining nodes and light wallet clients, which do not store the entire copy of the blockchain.

Figure 3-8 Illustration of the extended Bitcoin network, including different types of nodes, gateway servers, edge routers, and client wallets, as well as the different protocols they use to connect to each other.

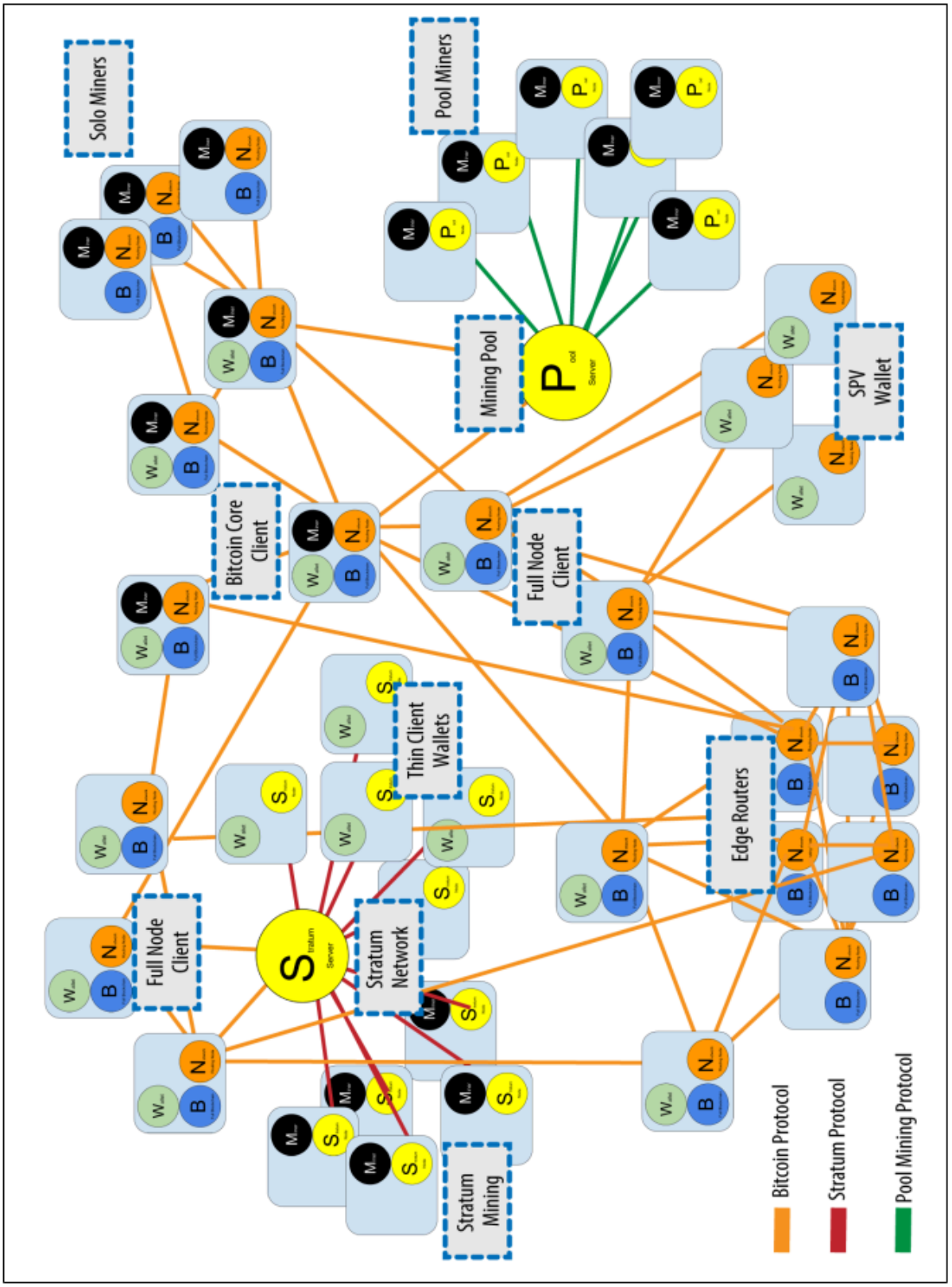


Figure 3-8. *The extended Bitcoin network illustrates different types of nodes, gateways, and protocols.*

3.1.5. Meaning and impact of Bitcoin

Bitcoin is not just a digital currency but has far-reaching implications and impacts on technology, finance, economics and society.

Technologically, Bitcoin is the first application of blockchain technology - a decentralized, transparent and secure ledger system. This technology is not limited to the financial sector but also has the potential for widespread application in other industries such as supply chain, healthcare, and management.

In finance, Bitcoin provides a peer-to-peer transaction method without going through banks or intermediary organizations. This is especially useful for people in areas without access to the traditional banking system. At the same time, Bitcoin is seen as a store of value, similar to "digital gold", helping to protect against inflation and devaluation of fiat currencies.

Economically, Bitcoin has created a completely new economy, attracting the participation of many industries such as Bitcoin mining (mining), cryptocurrency trading, and blockchain technology development. These industries not only drive innovation but also create many employment and investment opportunities.

Socially, Bitcoin brings financial freedom, allowing people to control their assets without depending on governments or banks. This promotes individual freedom and reduces dependence on centralized systems.

However, Bitcoin also faces many challenges and controversies. The Bitcoin mining process consumes large amounts of energy, causing concerns about its environmental impact. The lack of clear regulation makes Bitcoin vulnerable to abuse for illegal activities. In addition, Bitcoin prices often fluctuate strongly, making it difficult to use as a stable payment currency.

Even so, Bitcoin is expected to continue to play a central role in the cryptocurrency ecosystem. Technological developments and growing adoption could turn Bitcoin into a foundation for decentralized finance solutions, while changing the way we think about money and value.

In short, Bitcoin is not only a financial tool but also a symbol of innovation and freedom, contributing to reshaping the way the world operates in the future.

3.2. Ethereum: Smart Contracts and DApps

3.2.1 Ethereum

If you need to answer the question "*Who has the greatest role and influence after Bitcoin's anonymous Satoshi Nakamoto?*", then the answer is **Vitalik Buterin**, a talented programmer born in 1994, is Canadian by nationality and has Russian roots. In 2011, just two years after Bitcoin was created, Vitalik was hard at work writing for the website *Bitcoin Magazine*, with a compensation of 5 Bitcoin per post. Not long after, he became a co-founder of this magazine.

Attachment to *Bitcoin Magazine* and his travels around the world to learn, meet, and interact with Bitcoin developers have helped Buterin become a Bitcoin expert. At the same time, he also soon realized the limitations in Bitcoin's features. From there, Buterin realized that he could build a new version of blockchain with greater potential by improving upon Bitcoin.

In 2013, Buterin first introduced his Ethereum project to the blockchain community via a **white document** (white paper). Buterin's vision for Ethereum is to become "**global computer**" (the World computer) when this decentralized system operates on many nodes around the world and every calculation or transaction on it takes place across the entire giant network as one entity.

The whitepaper lays out the vision and some initial concepts for Ethereum including key points:

- Provide one *programming language* **Turing complete (**Turing complete), the language is capable of performing any calculation or algorithm given enough time and resources. In other words, this language can simulate every computer program imaginable. Ethereum uses this language to build “smart contracts” – self-executing programs that run on the blockchain.
- **Set up peer-to-peer transactions in the blockchain:** This platform allows the creation and deployment of smart contracts and decentralized applications (DApps). Anyone can define, create, and exchange value on Ethereum, including cryptocurrencies, stocks, and many other assets.

Buterin's idea is not entirely new; Like Bitcoin, Ethereum continues to inherit the ideas of blockchain pioneers and develop them in new ecosystems. For example, Nick Szabo

previously proposed the idea of **smart contracts** (smart contract) and asset management on a distributed database. Additionally, with the **Decentralized distributed applications** (DApps), an evolution of previous blockchains (like Namecoin, Colored Coins and Metacoins, which mostly just tried to simulate money) Buterin and his colleagues wanted to merge and improve all of these concepts to make **Blockchain is programmable and widely used**.

Accompany **Vitalik Buterin** in the early years of Ethereum's development were famous names, including **Gavin Wood** (later founded Polkadot), **Charles Hoskinson** (later founder of Cardano), **Anthony DiIorio** (founder of Decentral Inc), and **Joseph Lubin** (founder of ConsenSys)…

Shortly after the white paper was released, Gavin Wood published it **golden documents** (Yellow Paper) – an in-depth technical document with a title *Ethereum: A Secure Decentralised Generalised Transaction Ledger*.

In this document, Wood not only describes Ethereum in detail but also explains it in detail **Ethereum Virtual Machine (EVM)**, along with the operating mechanisms of the system. The yellow document serves as a supplement to the white paper, providing more precise technical descriptions of how the system works based on the ideas introduced in the white paper.

In July and August 2014, the Ethereum public presale event (crowdsale) was held, in which a certain amount of Ether was sold before the project officially launched. Such presale events are often called initial token offerings (ICOs) or initial token sales (ITS) in the blockchain sector.

This pre-sale event was very successful. Vitalik Buterin's team has raised $18.4 million, securing resources for Ethereum development. This was done under the auspices of the non-profit organization Ethereum Foundation, which was founded just before the crowdsale event and is headquartered in Switzerland.

First test version

In May 2015, the first test version of Ethereum went online. Known as "Olympic," this version allows users to test the system's load capacity and security. Participants also receive incentives if serious errors or problems are discovered.

Official version: Frontier

On July 30, 2015, the first official version of Ethereum was released under the name "Frontier." This is the official launch of the Ethereum network, opening up the possibility of building and deploying decentralized applications (DApps) through the use of smart contracts.

From there, Ethereum continued to evolve through several major upgrade phases, including:

- **Homestead (2016):** Enhanced stability and improved security features.
- **Metropolis (Byzantium và Constantinople, 2017-2019):** Provides new features such as zk-SNARKs and improved network performance.
- **Serenity (Ethereum 2.0, 2022):** Completely change from PoW to PoS mechanism, reducing energy consumption and improving scalability.

Ethereum, after successfully transitioning to POS, continues to aim for improved performance, scalability, and sustainability with the next stages of development such as:

- **Sharding:** Helps fragment data for more efficient processing and increased network capacity.
- **Layer 2 Solutions:** Scaling solutions like Optimism and zk-Rollups help reduce the load on the Ethereum mainnet.

We will explore Ethereum extensions and future development directions in the next chapters. Before moving on to the next part, let's continue to get acquainted with some basic terms:

State Machine

Ethereum views itself as a transaction-based state machine, which starts with an initial state (genesis state) and is converted to a final state through transactions. This final state is not the state in which the system ends, but is always the most up-to-date state of the platform (see more in yellow paper)

Bitcoin can also be described as a state machine, with state represented by the global set of all unspent transaction outputs (UTXOs). The state of Bitcoin is also changed by transactions on the network.

To initiate these transactions, participants must use their keys to access one or more UTXOs and convert them into new UTXOs. As stated in the above sections about Bitcoin, with Bitcoin, users do not have an account balance associated with their address. They only manage the keys in their wallets so they can unlock the UTXOs assigned to them.

So while Bitcoin's state is quite abstract, Ethereum views state as a fundamental concept upon which its entire project is built. Unlike Bitcoin, accounts are an important underlying structure in the Ethereum network. These accounts represent the addresses of participants in the network, but may contain more information.

Ethereum and Turing completeness

As soon as you start learning about Ethereum, you will immediately come across the term "Turing complete" (**Turing complete**). It is often said that Ethereum, unlike Bitcoin, is Turing complete. So what does this really mean?

This term is related to the British mathematician **Alan Turing**, who is considered the father of computer science. In 1936, he created a mathematical model of a computer, consisting of a state machine that manipulated symbols by reading and writing them to memory sequentially (like a endless rolls of paper tape). With this model, Turing provided the mathematical basis for answering (in the negative) questions about universal computability, i.e. whether all problems are solvable. He proved that there are classes of problems that cannot be computed.

Specifically, Turing proved that **Halting problem** - whether it is possible to determine, given any program and any input, whether that program will terminate or continue to run forever - is unsolvable.

Alan Turing also defined a system as **Turing complete** if it can be used to simulate any Turing machine. Such a system is called **Universal Turing Machine (UTM)**.

Ethereum's ability to execute a stored program, in a state machine is called **Ethereum Virtual Machine (EVM)**, which simultaneously reads and writes data to memory, making it a Turing complete system and thus a UTM. Ethereum can compute any algorithm that any Turing machine can perform, within the limits of finite memory.

Ethereum's breakthrough lies in combining the general-purpose computing architecture of a stored-program computer with a decentralized blockchain, thereby creating a distributed "world computer" with a single state (singleton). Ethereum programs run "everywhere" but create a common state protected by consensus rules.

Turing complete as a “Feature”

When you hear that Ethereum is Turing complete, you might conclude that this is a feature that Turing incomplete systems lack. But actually, this is quite the opposite. Turing

completeness is easy to achieve; in fact, the simplest known complete Turing state machine has just that **4 states and uses 6 symbols**, with a long-only state definition **22 instructions**.

Sometimes, there are even systems that are discovered to be **"Turing complete by accident"**. An interesting list of such systems can be found at: http://beza1e1.tuxen.de/articles/accidentally_turing_complete.html

However, Turing completeness is dangerous, especially in open access systems like public blockchains, because of the problem **halting problem** was mentioned in the previous section.

For example, modern printers are Turing complete and can be put into a "freeze" state by complex print files.

The fact that Ethereum is Turing complete means that any program of any complexity can be computed by Ethereum. But this flexibility brings difficult problems of security and resource management. An unresponsive printer can be turned off and on again. But that is not possible with a public blockchain.

Consequences of Turing completeness

Turing proved that you cannot predict whether a program will terminate or not by simulating it on a computer. Simply put, we cannot predict a program's journey without actually running it. Turing complete systems can run in "infinite loops" (**infinite loops**) – a term used (in a simplified way) to describe a never-ending program.

It's very easy to create a program that runs forever. However, unwanted infinite loops can appear without warning, due to complex interactions between initial conditions and program code. In Ethereum, this poses a major challenge: each node (client) participating in the network must verify every transaction, including running the smart contracts that transaction invokes.

However, as Turing demonstrated, Ethereum cannot predict in advance whether a smart contract will end, or how long it will run, without actually running it (and possibly running forever).

Whether by accident or intention, a smart contract can be made to run forever as a node attempts to verify it. This is essentially a formality **Denial of service (DoS) attack**. Additionally, between a program that takes just a few milliseconds to verify and a program that runs forever, there exists a wide range of programs that consume heavy resources, increase memory, overload CPU, and waste resources. resources.

In a "world computer," a resource-abusing program is abusing the entire world's resources. So how can Ethereum limit the resources a smart contract uses if it cannot predict resource usage in advance?

Resource limitation mechanism: Gas

To solve this problem, Ethereum introduced a resource measurement mechanism called **gas**. When EVM executes a smart contract, it carefully calculates each command (including calculations, data access, etc.).

Each order has a fixed cost in gas units. When a transaction triggers the execution of a smart contract, it must be accompanied by a gas amount, which sets a maximum limit on the resources that smart contract can consume.

EVM will stop execution if the amount of gas consumed exceeds the amount of gas available in the transaction. **Gas** is the mechanism that Ethereum uses to enable Turing complete computation while still limiting the resources any program can use.

How to get Gas?

The next question is, how do users get gas to pay for computation on the Ethereum "world computer"? You will not find gas on any exchange. Gas can only be purchased as part of a transaction and can only be purchased with **ether**.

Ether needs to be included with the transaction, and needs to be clearly designated for gas purchases along with an acceptable gas price. Just like fuel prices at gas stations, gas prices are not fixed.

Gas is purchased to make the transaction, the calculation is performed, and any unused gas is returned to the sender of the transaction.

3.2.2. Smart Contract

Right in the title of the Ethereum white paper, Vitalik Buterin mentioned new generation Smart contracts. The document described the Bitcoin protocol as one **weak version** of the smart contract concept that Nick Szabo defined. Accordingly, Buterin proposed a version *new generation* More powerful programming language based **Solidity**, which is a Turing complete language (*Turing Complete*). Since then, many other cryptocurrencies have supported programming languages that allow the development of more complex smart contracts between mutually untrusting parties.

What is a smart contract?

First of all, let's look again at the concept of "contract." We can easily see that in everyday life, there are many types of contracts such as house rental contracts, loan contracts, car rental contracts, or professional leasing contracts... In contracts, there will often be contents and terms for the parties to implement. An effective contract will detail the formal requirements, the responsibilities of each party, when and how the terms will be enforced, and the consequences if these rules are not followed. Therefore, the contract is a reliable document, ensuring that the parties involved perform as planned.

Smart contracts have many similarities to traditional contracts, but are implemented through computer code. Once created on a decentralized blockchain network, smart contracts cannot be changed. Once the conditions are met, the smart contract will automatically execute without third-party intervention. In the Ethereum documents, the authors state that, in the context of Ethereum, the term is actually a bit misleading. Because smart contracts on Ethereum are not truly "smart" nor are they legal contracts. However, the term is still widely used to refer to immutable computer programs, operating deterministically in the context of the Ethereum Virtual Machine (EVM), as part of the Ethereum network protocol, i.e. on the decentralized global computer Ethereum.

Because blockchain is decentralized, immutable, and transparent, everyone in the network can publicly verify the transaction results of smart contracts.

Nick Szabo was the first to describe the concept of smart contracts. In 1997, he published the article **"The Idea of Smart Contracts"** (The idea of smart contracts). He envisioned the job *convert contracts into programming code to create self-executing contracts without the need for trust between parties.*

To illustrate his concept, Nick Szabo uses a vending machine as an example of how smart contracts work. Once you put the correct amount into the machine, you will get the desired product. Programming instructions inside the vending machine ensure that the contract will be executed as intended.

Differences between Bitcoin and Ethereum about Smart Contracts

- **Bitcoin**: Only supports some basic smart contract functions (such as conditional transactions), with limited programming and scalability.

- **Ethereum**: Taking the concept of smart contracts to the next level with complete Turing programmability, allowing for the creation of decentralized applications (**DApps**) and decentralized financial ecosystems (**DeFi**) abundant. Anyone can create smart contracts on the Ethereum blockchain. The source code of smart contracts is transparent and can be publicly verified. Everyone can see how the execution logic of smart contracts is built.

Structure of Smart Contract

Smart contracts in Ethereum are computer programs written in source code and deployed on the Ethereum blockchain. Each smart contract has three main components:

- **Functions:** Functions are sections of code that perform specific actions, such as changing the state of a contract, transferring money from one person to another, or checking certain conditions. Functions can be called directly from outside the contract or triggered by other events in the system. For example, a smart contract might have a "transfer" function to transfer money between two accounts.
- **State:** States are variables that store information about the contract's data. This state can be changed by functions in the smart contract. For example, if the smart contract is a financial contract, the state could be the user's account balance or the number of tokens a person owns.
- **Events:** Events are notifications or signals emitted from smart contracts when a certain action or condition occurs. Events help users or other applications monitor contract activity. For example, when a transaction is successful, the smart contract can emit an event to notify that the transaction has been completed.

Ethereum Virtual Machine (EVM)

Ethereum Virtual Machine (EVM) is the execution environment for smart contracts on Ethereum. EVM is like a virtual computer that can execute programs written in bytecode in Ethereum. When a smart contract is deployed on the blockchain, its bytecode is uploaded to the EVM for execution. EVM is responsible for running smart contracts, processing transactions, and updating the state of the blockchain.

EVM is capable of interacting with other data and contracts on the Ethereum blockchain, while ensuring the security and transparency of transactions. Each time a smart contract is

executed, EVM calculates the costs of system resource usage, including gas, and confirms that the contract works as expected.

Smart Contract implementation process

When a user wants to interact with a smart contract in Ethereum, they send a transaction to the Ethereum network. This transaction could be a request to execute a function in a smart contract, such as transferring an amount of ether from one person to another. Once the transaction is issued, execution will take place according to the following steps:

1. **Send transaction:** Users create and send transactions to the Ethereum network through their digital wallets (e.g. MetaMask, MyEtherWallet).
2. **Sign and approve the transaction:** The transaction will be signed by the user with their private key to confirm the legitimacy of the transaction.
3. **Broadcast transactions on the network:** Transactions are broadcast on the Ethereum network and nodes in the network will begin processing transactions.
4. **Run the contract in EVM:** When a transaction involves a smart contract, EVM will find the corresponding smart contract, read its bytecode, and execute the functions requested by the user. State changes (such as money transfers) will be recorded in a new block of the blockchain.
5. **Gas cost:** To execute functions in smart contracts, users must pay gas costs, equivalent to the system resources consumed. If gas is insufficient, execution will stop and the transaction will fail.
6. **Update status and create new block:** When the smart contract performs successful actions, such as transferring money, EVM will update the state of the blockchain and create a new block containing information about that transaction.
7. **Event replay:** If the smart contract generates an event, other applications can listen and process information from that event. For example, a DApp can update its user interface based on events emitted by the smart contract.

Relationship between Smart Contract and Gas

Every time a smart contract is called and executed, gas costs are calculated. Gas is a unit of measurement for resources that a transaction consumes when performed on the Ethereum

blockchain. Each operation in a smart contract, from calculation to reading and writing data, requires a certain amount of gas.

Gas has two main elements:

- **Gas Limit:** Is the maximum amount of gas that a user is willing to pay for a transaction. If a smart contract requests more than this gas limit, the transaction will be canceled.
- **Gas Price:** Is the value that users are willing to pay for each unit of gas. Gas price is usually calculated in gwei (1 gwei = 10^-9 ether).

The transaction cost formula would be:
Cost = Gas Limit × Gas Price
For example, if the gas limit is 50,000 and the gas price is 20 gwei, the transaction cost will be 1,000,000 gwei (or 0.001 ether).

As stated above, Gas helps protect the Ethereum network from unoptimized or endlessly running smart contracts. If a smart contract is not optimized or does not have reasonable stopping conditions, the gas limit will stop executing, avoiding taking up system resources.

Smart Contract "Hello World"

Here is a basic example of "Hello World" in Solidity:

// SPDX-License-Identifier: MIT

```
pragma solidity ^0.8.18;
contract chao{
    string public chaomung="Hello World!!!!";
}
```

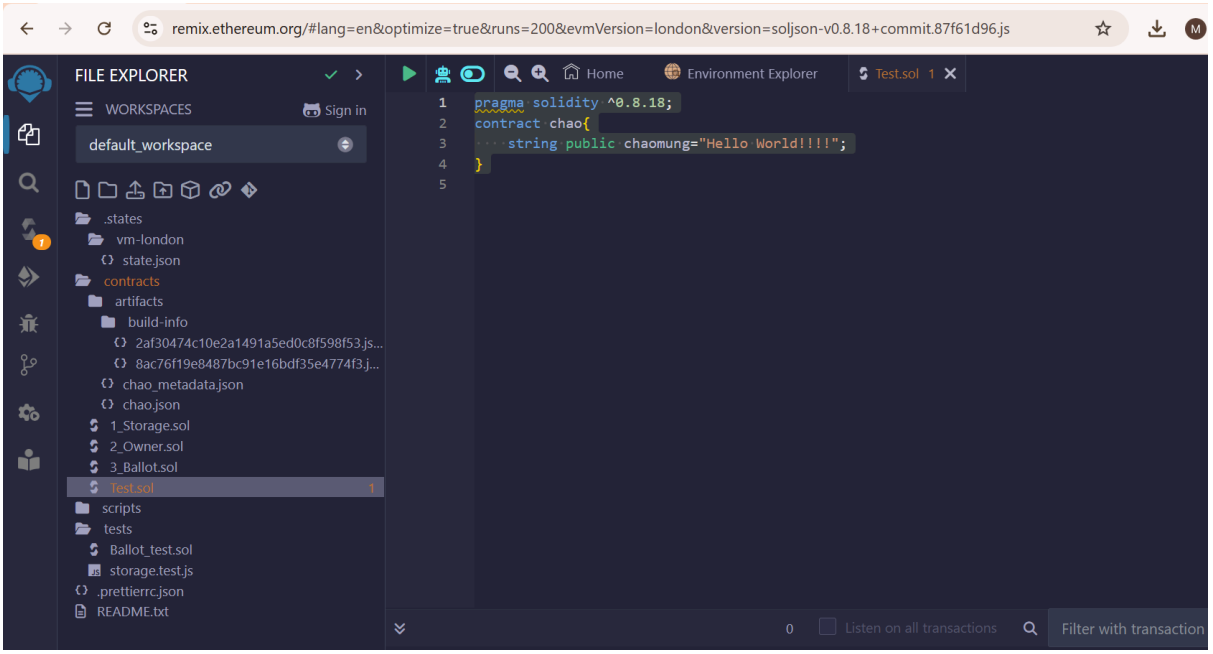
Details of each part of the contract:

1. **pragma solidity**: Defines the Solidity version to be used. For example, ^0.8.0 ensures the code runs on versions 0.8.0 and above.
2. **string public chaomung**: Message variable stores the message "Hello World!!!!" and can be accessed publicly.

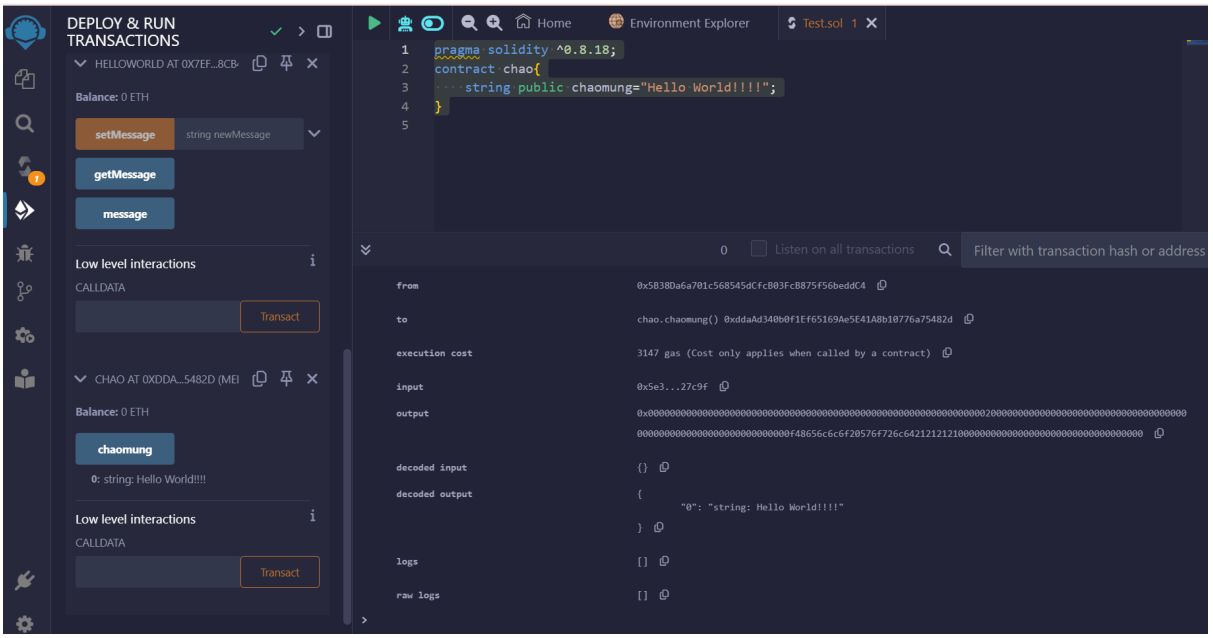
Deployment and testing process on Remix

1. **Open Remix IDE**:

- Access Remix IDE at https://remix.ethereum.org/
- Create a new file (for example: Test.sol).
- Paste the above code into the file.



2. **Compile**:
   - Select the "Compiler" tab in Remix.
   - Select the appropriate Solidity version and click "Compile Test.sol".
3. **Deploy**:
   - Switch to the "Deploy & Run Transactions" tab.
   - Select the environment as "JavaScript VM (London)" to test on the emulated network.
   - Click the "Deploy" button to deploy the contract.
4. **Interact**:
   - Once deployed, you will see contract-related buttons and fields in Remix.
   - Press Chaomung to read the default message ("Hello World!!!").



3.2.3. Decentralized applications (DApps)

In this section we will explore the world of decentralized applications, also known as DApps. From the early days of Ethereum, the vision of its founders went far beyond the concept of "smart contracts": it was to reinvent the web and create a new world of DApps, appropriately called web3. Smart contracts are a way to decentralize the control logic and payment functions of an application. DApps in web3 go further by decentralizing all other aspects of an application: storage, messaging, naming, etc.

What is DApp?

A DApp is an application built primarily or completely in a decentralized manner. Let's look at the aspects of an application that can be decentralized:

- **Backend software (application logic)**
- **Frontend software**
- **Data storage**
- **Message communication**
- **Resolve domain names**

Each of these aspects can be developed in a centralized or decentralized manner. For example:

- **Frontend**: Can be a web application running on a centralized server or a mobile application running on the user device.
- **Backend and storage**: Can be hosted on private servers and proprietary databases or using smart contracts and P2P hosting.

Benefits of building a DApp

Compared to conventional centralized architecture, DApps bring many outstanding benefits:

1. **Error resistance**
    - Business logic is controlled by smart contracts, the backend of the DApp is completely distributed and managed on the blockchain platform.
    - The DApp has no downtime and will continue to be available as long as the platform remains operational.
2. **Transparency**
    - The nature of DApps on the blockchain allows anyone to examine the source code to better understand how it works.
    - Every interaction with the DApp is permanently stored on the blockchain.
3. **Censorship resistance**
    - As long as users have access to an Ethereum node (or run the node themselves if needed), they can always interact with the DApp without being hindered by any centralized authority.
    - Even smart contract owners cannot change the code once it is deployed to the network.

Current status and future of DApps

Currently, in the Ethereum ecosystem, there are very few applications **truly decentralized** — most still rely on somewhat centralized services and servers. In the future, we hope every component of the DApp can operate completely decentralized.

Backend (Smart Contract)

- In a DApp, smart contracts are used to store business logic (program code) and state related to the application.

- Smart contracts can be replaced **backend server-side** in traditional applications. However, it should be noted:
    1. Every calculation in a smart contract is expensive, so it is necessary to keep it minimal.
    2. The parts of the application that need a decentralized and trusted execution platform must be clearly identified.

Smart contract architecture design

1. **Limit code editing**
    - Once deployed, the smart contract code cannot be changed. It can only be cleared if programmed with a command **SELF DESTRUCT**, but other than removing it completely, there's no way to edit the code.
2. **DApp Size**
    - A large smart contract can consume a lot of gas to deploy and use. Therefore, some applications may choose to compute off-chain and use an external data source.
    - However, if the DApp's core business logic depends on external data (e.g. from a centralized server), users will have to trust these resources.

In short, building and deploying a DApp requires attention to decentralization, gas efficiency, and data integrity. Despite their challenges, DApps are ushering in a new era of decentralized and transparent web applications.

3.3. Third generation: Cardano – Built from academic research

3.3.1. Introducing Cardano

Cardano is considered one of the most advanced and unique blockchains today, featuring an approach based on rigorous academic research before implementing the technology. Designed to overcome the limitations of previous generation blockchains, Cardano places an emphasis on sustainability, transparency, and scientific innovation. With a combination of theoretical research and practical application, this project has reshaped the way blockchain is developed systematically and reliably. Now, we will learn about the history and development of the Cardano blockchain.

3.3.1.1. Background of Birth

In 2015, Charles Hoskinson – one of the co-founders of Ethereum – decided to launch a new blockchain. After leaving Ethereum due to some disagreements about the direction of development, Hoskinson saw an opportunity to create a blockchain platform built by science and careful research.

Cardano is designed as a third generation blockchain, aiming to overcome the problems of Bitcoin (first generation) and Ethereum (second generation). The main goal is to achieve sustainability, scalability, and decentralization while ensuring security and transparency.

3.3.1.2. Startup Phase (2015-2017)

Cardano is developed and promoted by three main organizations:

- **IOG (Input Output Global):** Input Output is a research and engineering company and venture studio building blockchain and Web3 products to empower everyone, everywhere.

  Founded by Charles Hoskinson and Jeremy Wood, Input Output is one of three pioneers behind Cardano, initially contracted to design, build and help maintain the Cardano platform. As a completely decentralized company, Input Output is comprised of dynamic, innovative teams - based around the world, together committed to innovation through the delivery of the highest standards of software engineering. software based on rigorous, peer-reviewed science.

  Input Output is a leader in building distributed computing systems and decentralized technology solutions. The company continues to research and build new models and products in the field of distributed ledger technology and Web3 architecture. Input Output is committed to open source principles and ethical, purposeful business, creating technology that benefits the many, not the few. Like the Cardano Foundation and EMURGO, promoting blockchain education is core to Input Output's philosophy. IO Research is focused on advancing academic research in blockchain, supported by a team of educators, academic partners, and specially developed courses.

- **The Cardano Foundation:** Cardano Foundation is an independent non-profit organization based in Switzerland. The Foundation is tasked with promoting the Cardano public digital infrastructure and working to anchor it as a utility for financial and social systems, thereby empowering digital architects of the future.

  The Foundation facilitates the advancement of Cardano worldwide in enterprise applications. The fund develops infrastructure tools—including where there may not be an immediate commercial use case—plus strengthens operational resilience and promotes campus diversity suitable for use on infrastructure as well as developing healthy and representative governance.

Another important part of the Cardano Foundation's mission is to engage and support the Cardano community. The Foundation supports the development of tools that the community can use to leverage Cardano to solve problems in new ways.

- **Emurgo:** EMURGO is a blockchain technology company and founder of the Cardano blockchain, providing products and services to drive adoption of Cardano's Web3 ecosystem. Founded in 2015 in Japan, EMURGO's mission is to facilitate commercial adoption through dynamic partnerships with existing ecosystem members and seamless integration of those just joined.

  By prioritizing investments, providing ongoing education, and providing infrastructure services, EMURGO aims to unlock the full potential of the Cardano ecosystem.

In 2017, Cardano officially launched with the release of ADA, the platform's official cryptocurrency. ADA is named after Ada Lovelace, who is considered the world's first computer programmer.

The Cardano blockchain has been developed with the Ouroboros consensus protocol, the world's first mathematically proven protocol with an emphasis on scalability and security.

3.3.1.3. Cardano's development roadmap

Cardano's development roadmap has been divided into five eras, each focusing on a different set of features:

- Byron focuses on establishing a foundation.
- Shelley focuses on decentralization and decentralization.
- Goguen is all about smart contracts.
- Basho is the driving force to achieve true scalability.
- Voltaire is based on implementing decentralized governance.

Each era is built around a set of features that are implemented and improved over multiple code releases. Although the work for each of these development streams is delivered sequentially, it is often performed concurrently, with research, prototyping, and development occurring simultaneously at the same time.

Figure 3-xx: Cardano development eras

- **Byron (Laying the Foundation)**

The Byron phase, launched in 2017, marked the beginning of Cardano with basic features. Users can trade ADA through the Daedalus wallet (full wallet version) or Yoroi (light wallet version).

This phase focuses on building the blockchain infrastructure and introducing the Ouroboros protocol, ensuring security and sustainability. Byron also laid the foundation for the network's growth by creating a global community of ADA users.

- **Shelley (Decentralization)**

The Shelley phase, launched in 2020, marks the transition from a centralized to a decentralized system. One of the biggest innovations is the introduction of staking mechanisms and stake pools. Cardano users can participate in the network by delegating their stakes to pools or operating their own stake pools.

Shelley has demonstrated Cardano's strong performance with thousands of staked groups established, helping to increase the decentralization and security of the network. This phase also improves system performance and stability.

- **Goguen (Smart Contract)**

The Goguen phase, launched in 2021, ushers in a new era for Cardano by introducing a smart contract feature. With the introduction of Plutus, a powerful smart contract programming

platform, Cardano has become a suitable platform for building decentralized applications (DApps).

Goguen also introduced Marlowe, a programming language specifically for financial contracts, helping users without programming skills create and execute complex contracts. Thanks to Goguen, Cardano is closer to competing with other major blockchain platforms like Ethereum.

- **Basho (Optimization)**

The Basho phase focuses on optimizing Cardano's performance and scalability. Enhancements include the introduction of Hydra, a layer 2 solution that speeds up transaction processing and reduces costs.

Hydra allows the Cardano network to process millions of transactions per second using state channels. This makes Cardano one of the most scalable blockchains, serving the growing needs of users and businesses.

- **Voltaire (Manager)**

The Voltaire phase, the final phase in the development roadmap, focuses on building a complete decentralized governance system. ADA users will have the right to participate in important network decisions through a voting mechanism.

Voltaire introduces a treasury system, where a portion of transaction fees are allocated to fund community development projects and initiatives. This helps Cardano maintain sustainable and flexible development in the future.

3.3.1.4. Impact and Future Vision

Cardano is not just a blockchain, but also an ecosystem with technological and social foresight. With a combination of scientific research, modular design, and a thriving developer community, Cardano is reshaping the way we think about blockchain.

- **Current Impact**

Cardano has made impressive achievements in delivering sustainable and secure blockchain solutions. Thousands of stake groups around the world have participated in the network, proving the decentralization and scalability of the platform.

Additionally, Cardano has attracted attention from non-profit organizations and governments in many countries, especially developing countries. For example, Cardano's Atala Prism project was used to deploy a digital identity system in Ethiopia, helping to improve access to education and public services.

- **Future Vision**

In the future, Cardano aims to become a comprehensive blockchain platform, supporting practical applications in many fields such as finance, education, healthcare, and administration. Some main development directions include:

DApps and DeFi support: With its powerful smart contract feature, Cardano will continue to support the development of decentralized applications and decentralized finance, competing directly with platforms like Ethereum.

Enhance scalability: Hydra and other layer 2 solutions will continue to be developed to ensure fast and low-cost transaction processing.

Community development: Cardano will continue to expand its user and developer community, driving participation from institutions and individuals worldwide.

Practical applications: Cardano will focus on creating practical blockchain solutions that help solve global problems such as digital identity, resource management, and transparency in governance.

With its commitment to scientific research and sustainable development, Cardano is expected to play an important role in shaping the future of blockchain technology and driving innovation globally.

3.3.2. Cardano's mechanism of action

Cardano is built on a consensus protocol **Ouroboros**, a pioneering Proof of Stake (PoS) system developed through academic research evaluated by the scientific community. At the heart of Ouroboros is the staking pool mechanism, where trusted server nodes are managed by dedicated operators. ADA holders can delegate their stakes to these pools, helping to ensure that anyone can participate in the protocol without requiring highly technical knowledge or the ability to maintain a node online . Stake groups play an important role in maintaining the network and managing the combined stakes of multiple stakeholders in a unified entity.

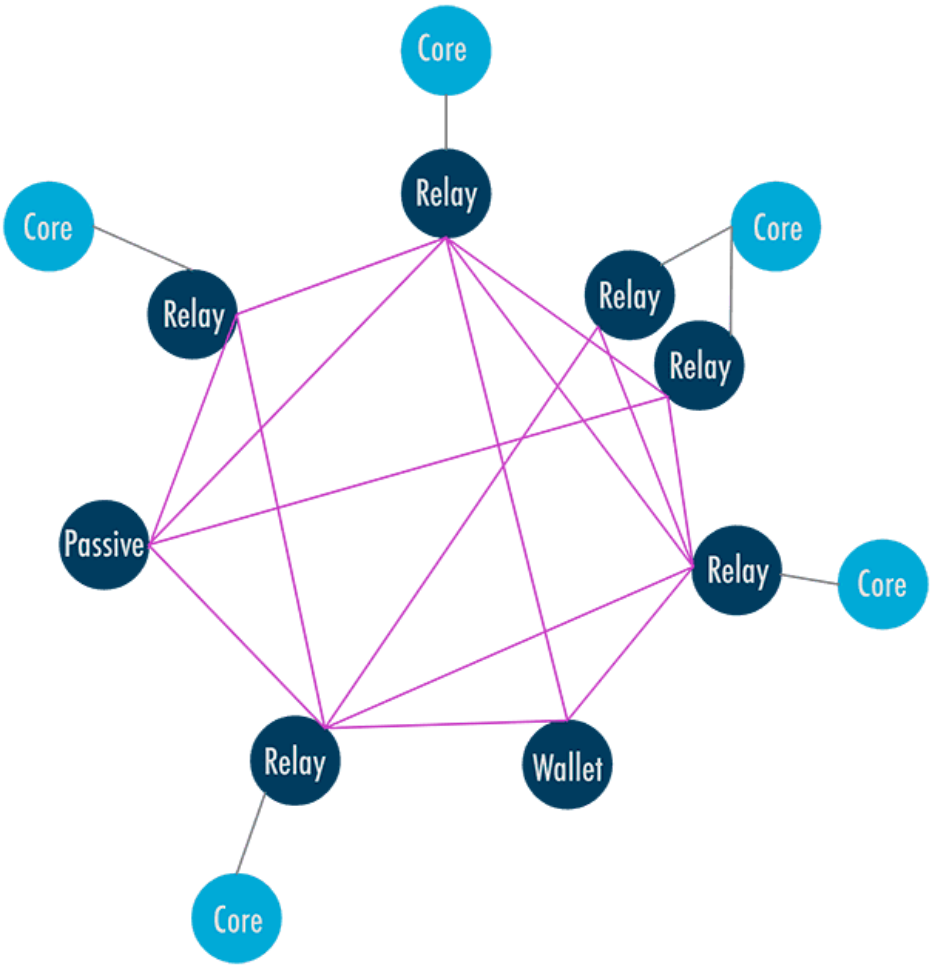


Figure 3-xx: Overview image of Cardano Blockchain network

Cardano Network

Cardano is a public blockchain ledger so every transaction, block details, and epoch data can be easily tracked using a variety of tools.

Cardano Explorer is a user-facing tool that takes data from the main database and reflects it on a simple and convenient web interface.

The explorer shows the latest epoch details. You can view the following information:

- Blocks are produced.
- Epoch time begins.
- The time the block was created.
- Number of transactions processed
- …

By selecting a specific block, you can explore that block in more detail to see its ID, size, epoch and block details, transaction count.

You can also search for specific epochs, transactions or blocks by pasting their IDs into the search field.

Here are some Explorers you can access:

- await
- Cardanoscan
- Cexplorer
- Cardano Assets
- Pool.pm

Cardano's operations are based on a detailed architecture that describes how it works at a high level. This architecture provides insight into core components and how they interact with each other.

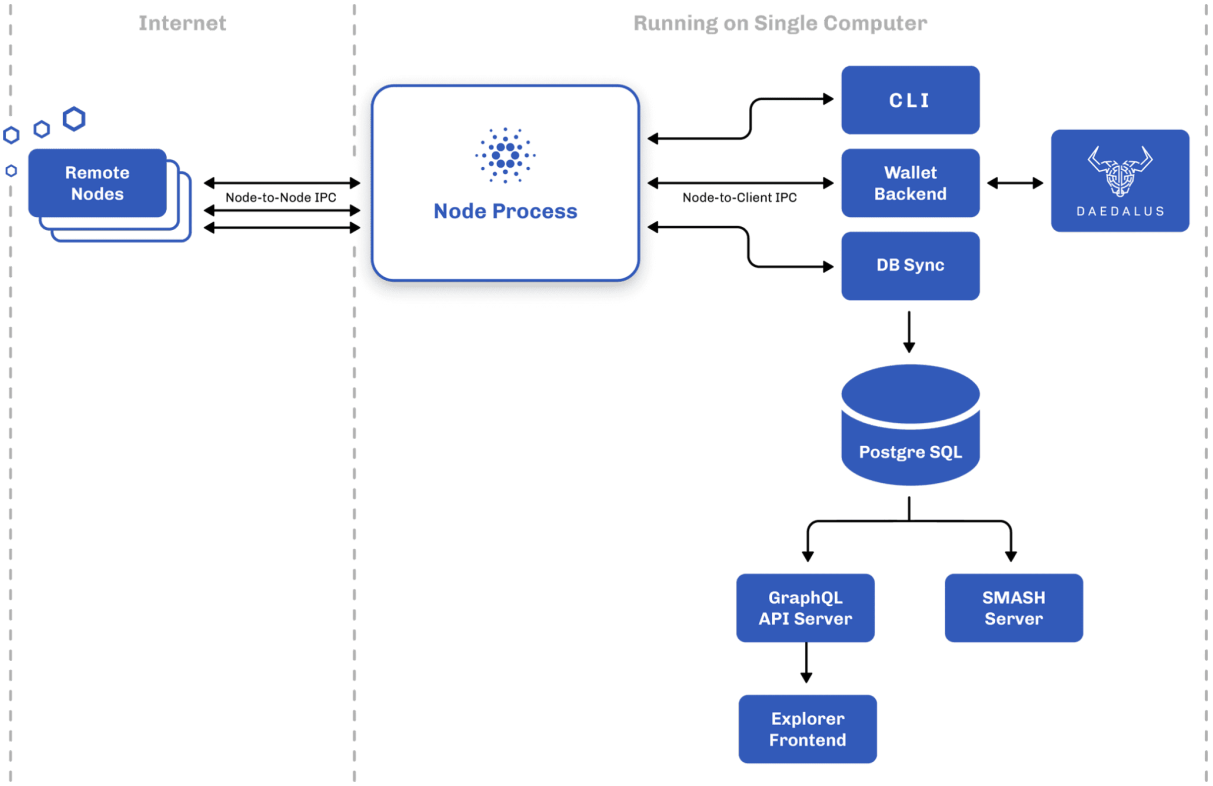


Figure 3-xx: Architecture of Cardano Blockchain.

Currently, Cardano implementations are designed with a highly modular architecture, consisting of many different components. Depending on specific use cases, these components can be flexibly combined to meet deployment needs.

- **Nodes and remote nodes**

A blockchain system consists of a set of nodes distributed across the network, communicating with each other to reach consensus on the state of the system.

Nodes take on the following tasks:

- Implements the Ouroboros protocol
- Validate and transmit blocks
- Production of blocks (some nodes take on this task)
- Provide information about the state of the blockchain to other local clients.

- **Node Process**

cardano-node is the top-level component of the Cardano system, which includes other subsystems, of which the most important components are consensus, ledger, and network. In addition, it also integrates support components such as configuration, command line interface (CLI), logging, and monitoring.

- **Node-to-Node IPC Protocol**

The purpose of the inter-node (IPC) protocol is to exchange blocks and transactions between nodes, which plays a role in the Ouroboros consensus algorithm.

This protocol includes three mini-protocols:

- **chain-sync:** Used to keep track of the chain and get block headers.
- **block-fetch:** Used to get block bodies.
- **tx-submission:** Used to forward transactions.

These mini-protocols are multiplexed over a TCP (Transmission Control Protocol) connection running continuously between nodes. They can run bidirectionally over the same TCP connection to support peer-to-peer (P2P) setup.

The overall protocol and each mini-protocol are designed for a trustless environment, where both sides need to protect against denial of service (DoS) attacks. For example, each mini-protocol uses a consumer-driven control flow, meaning that a node only requests more work when it is ready, instead of being forced to accept more work. .

The protocol's modular and scalable design allows new mini-protocols to be added or updated over time without causing compatibility issues.

- **IPC protocol between node and Client (Node-to-Client IPC Protocol)**

The purpose of the IPC protocol between node and **Client** is to allow local applications to interact with the blockchain through the node. This includes applications such as wallet backends or blockchain exploration tools. This protocol allows applications to access the raw data of the chain and query the current state of the ledger. Additionally, it also provides the ability to submit new transactions to the system.

This protocol uses the same design as the inter-node protocol, but with a different set of mini-protocols and uses local pipes instead of TCP connections. So this is a narrow, low-level interface that only provides what the node can provide. For example, the node provides access to all raw data of the chain, but does not provide a way to query the data on the chain. The task of providing data services and higher-level APIs is left to dedicated clients, like cardano-db-sync and the wallet backend.

This protocol includes three mini-protocols:

- **chain-sync:** Used to track the chain and receive blocks.
- **local-tx-submission**: Used to send transactions.
- **local-state-query:** Used to query the ledger status.

Version ***chain-sync*** in protocol **node-client** uses full blocks, instead of just block headers, so there is no need for a separate block-fetch protocol. Protocol ***local-tx-submission*** Similar to the tx-submission protocol between nodes but simpler and returns detailed information about transaction validation errors. Protocol ***local-state-query*** provides the ability to query the current ledger state, which contains a lot of interesting data that is not directly reflected on-chain.

- **Command line interface (CLI)**

Node's CLI tool is likened to "swiss army knife" system. It can perform almost any task, but it is low-level and not very convenient because it is text-based and does not have a graphical interface (GUI).

CLI tools can:

- Query the node to get information.
- Build and sign transactions.
- Send transaction.
- Manage encryption keys.

- **Daedalus knows**

Daedalus is a full-node wallet that helps users manage ADA and send and receive payments on the Cardano blockchain. Daedalus consists of two parts: the wallet interface (frontend) and the backend.

- Frontend: Is a graphical application that users can see and interact with directly.
- Backend: Is a service process that monitors the user's wallet state and performs complex tasks such as coin selection, transaction construction, and submission.

The backend interacts with a local node via the node-to-client IPC protocol and connects to the frontend via an HTTP API. The backend also provides a CLI tool that allows interaction with the wallet. Additionally, the wallet's backend can be used separately – without Daedalus – through its API. This is a convenient way for software developers to integrate Cardano into other applications and systems.

In addition to Daedalus wallet, which is a full-node wallet, there are many other wallets such as

Knows Hardware

Here is a list of hardware wallets to consider for storing and transacting ada:

- Trezor Model T
- Ledger Nano S Plus
- Ledger Nano X

Light wallet

In addition to Daedalus wallet and hardware wallets, Blockchain Cardano also has many light wallets developed by the community, operating on browsers or mobile applications, including:

- Lace
- Us
- Eternl
- GeroWallet
- Typhon
- Ellipal
- AdaLite
- Infinity Wallet
- Atomic Wallet
- Guard
- Tango

- SimpleHold Wallet
- Coin Wallet
- Purpose
- NOW Wallet

- **DB Sync**

Cardano's node only stores the blockchain along with the relevant information needed to validate the blockchain. This design principle aims to minimize code complexity, reduce computational costs and resource usage, keep the node's local interfaces as simple as possible, and use separate clients. externally to provide more convenient interfaces and additional functionality.

In particular, the node does not provide a convenient query interface for historical information on the blockchain. Data services **DB-Sync** This is provided by a separate component that uses a SQL (Structured Query Language) database. This data has one-way properties. It only has the direction of writing from Node Cardano and reading from User.

3.3.3 EUTxO model of Cardano Blockchain

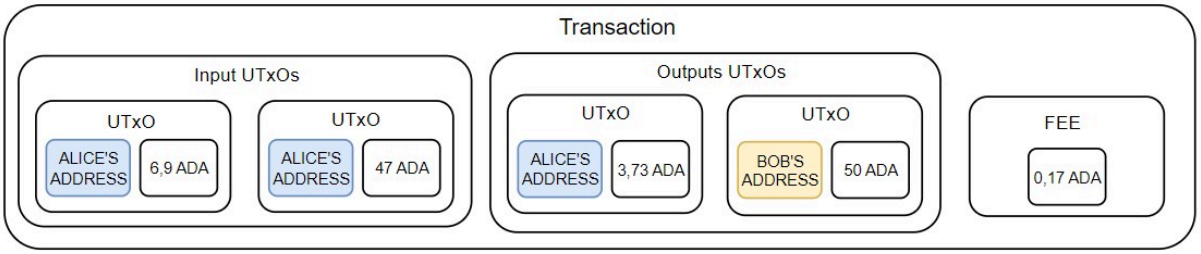
Basic information about UTXO

UTXO (Unspent Transaction Output) is the output from a previous transaction, which can be used as input for future transactions.

The UTXO model is not based on the concept of accounts or balances, but operates like cash, with each UTXO carrying a specific value, for example: 6.9 ADA, 47 ADA, or 459.7 ADA. The Cardano wallet manages UTXOs and displays the total as a balance, for example 513.6 ADA from three UTXOs.

When sending 50 ADA, the wallet will select UTXOs of sufficient value to cover, for example, use two UTXOs (6.9 ADA and 47 ADA) to create a total transaction of 53.9 ADA. The transaction generates two output UTXOs: 50 ADA to the recipient and 3.73 ADA returned to the sender after deducting the transaction fee.

The input UTXO is fully consumed, and a new output UTXO is created from the transaction, reflecting the basic principle of the UTXO model.



Difference between UTxO and Account Based Model

The account-based model works on the concept of accounts and balances, similar to how the banking system works. Ethereum applies this model, where each user has an account that holds his or her token balance. Transactions in this model perform by updating the balances of the sender and receiver accounts, which are one in the same **atomic operation** depends on the global state.

- ➢ **Atomic operation** is a concept in computer science that describes operations or actions that are performed without interruption or intervention. This ensures data integrity and system consistency, especially in environments with multiple threads or processes operating on the same resource.

In contrast, the eUTxO (Extended Unspent Transaction Output) model is an extension of the UTxO model used in Bitcoin. In this model, assets are stored as UTxO instead of account balances. Each UTxO represents a specific amount of value, which can be used as input for new transactions. Transactions use UTxOs from previous transactions and create new UTxOs for future use.

These two models represent distinct approaches to blockchain asset management, each tailored to the specific goals and applications of each platform.

Cardano's extended UTXO model

Cardano uses the EUTxO accounting model improved from the Bitcoin Blockchain's UTxO, to support multiple assets and smart contracts. It is different from the account-based model used by banks or Ethereum. In this section, the difference between the account-based model and the UTxO model will be briefly explained. The goal is to explain in detail how users spend UTxO.

The EUTXO model extends the UTXO model in two ways:

- *Generalize the concept of "address" using the lock and key metaphor.*
  Instead of limiting the key to just the public key and the key to just the signature, addresses in the EUTXO model can contain arbitrary logic in the form of scripts. For example, when a node validates a transaction, the node determines whether the transaction is allowed to use a particular output as input. The transaction will look up the instruction set given by the output's address and execute the instruction if the transaction can use the output as input.
- *Output can carry (almost) arbitrary data, in addition to address and value.*
  This makes scripts more powerful thanks to their ability to carry state information.

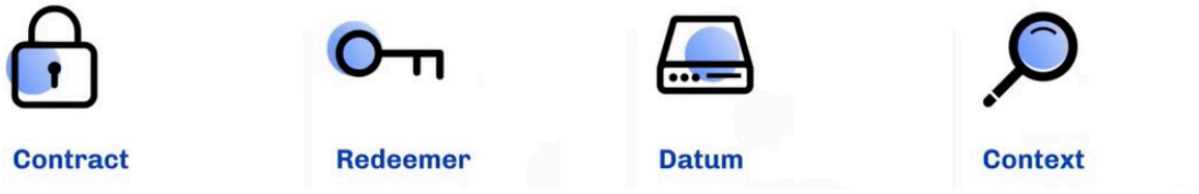
Furthermore, the EUTxO model extends the UTxO model by allowing output addresses to contain complex logic to decide which transactions can unlock them and by adding custom data to all outputs. When validating an address, the script accesses the data carried by the output, the transaction being validated, and some additional data called "redeemer," which is provided by the transaction for each input . By looking up all this information, the script has enough context to give a "yes" or "no" answer even in complex situations and use cases.

EUTxO allows arbitrary logic to be executed as scripts. This logic examines the transaction and data to decide whether the transaction is allowed to use an input or not.

The UTxO model with its graph structure is fundamentally different from the account-based model used by some current smart contract-enabled blockchains. Therefore, the design patterns that work for DApps on account-based blockchains are not directly applicable to Cardano. New design patterns are needed because the underlying data representation is different.

EUTxO inherits the branching design of the UTxO (Bitcoin) model, where a branch is defined as a chain of transactions that requires a chain of validation. To separate logic across different branches and enhance parallelism, it is important to build DApps and other solutions using multiple UTxOs. This provides scalability benefits, similar to the development of Bitcoin services that require dividing a wallet into multiple sub-wallets.

Basic components of EUTxO



- **Smart Contract (Contract)**: Used to lock UTxO, ADA, native assets and NFTs.
- **Redeemer**: Data provided by users to unlock locked assets and spend them.

- **Datum**: Data such as scores, user information or information related to your application. It is the information attached to UTxO that if you want to use that UTxO, you need to satisfy the conditions in the Scripts where Datum is a parameter.
- **Context**: Information such as metadata about the transaction being validated.

Benefits of the EUTxO model

Cardano's EUTxO model provides a secure and flexible environment to handle multiple operations without system failure. This model offers greater scalability and security, along with simpler transaction logic, as each UTxO can only be consumed once and completely, making transaction verification easier .

The EUTxO model has unique advantages over other accounting models. The success or failure of transaction validation depends only on the transaction itself and its inputs, not on any other factors on the blockchain. Therefore, the validity of a transaction can be checked off-chain before posting to the blockchain. A transaction can still fail if another transaction simultaneously consumes an input for which the transaction is waiting, but if all inputs remain, the transaction is guaranteed to succeed.

This is in contrast to an account-based model (like Ethereum), where a transaction can fail during script execution. This never happens in the EUTxO model.

Thanks to the "local" nature of transaction validation, a high degree of parallelism can be achieved. In principle, a node can validate transactions in parallel, as long as those transactions do not attempt to consume the same input. This provides greater efficiency and simplifies the analysis of possible outcomes, while also demonstrating that no unexpected events have occurred. You can learn more in the blog post about the EUTxO model.

A powerful feature of the EUTxO model is that the fee required for a valid transaction can be accurately predicted before posting the transaction. This is a unique feature not found in account-based models. Account-based blockchains, like Ethereum, are non-deterministic, meaning they cannot guarantee the efficiency of on-chain transactions. This uncertainty causes the risk of losing money, unwanted high fees, and opportunities for countervailing behavior.

In short, EUTxO offers higher levels of security, predictability of smart contract execution costs (no nasty surprises), and stronger parallelism.

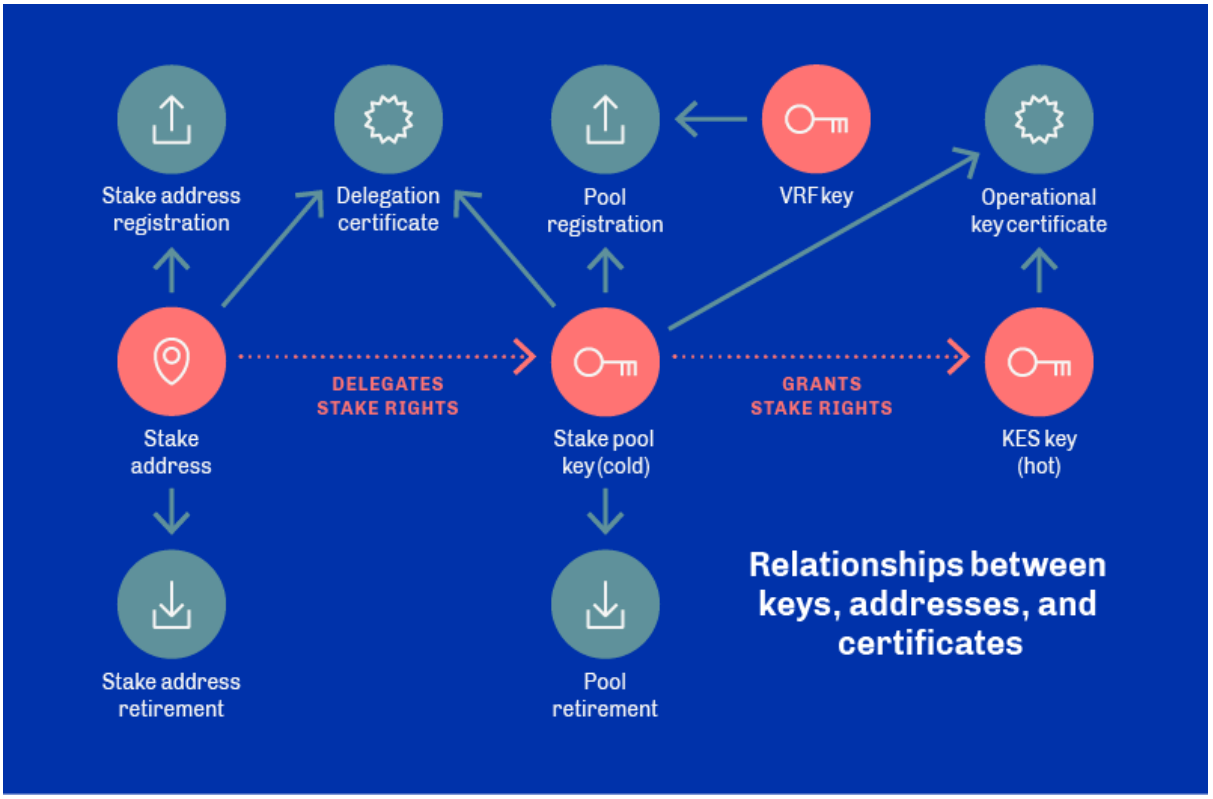
3.3.4. Key, address of Blockchain cardano.

Cardano's key

Cardano keys are asymmetric cryptographic key pairs used for many purposes on the Cardano blockchain, including:

- Sign and verify payment transactions and staking certificates.
- Identify and define addresses on the Cardano blockchain.

Below is a diagram illustrating the relationship between keys, addresses, and certificates:



Types of keys in Cardano

In the Cardano ecosystem, there are two main types of keys:

- **Lock node**
- **Address lock**

Lock node represents the security of the blockchain and includes the following keys:

Operator/Operational Key

- Operator keys are offline key pairs of the node operator, including a certificate counter used to generate new certificates.
- The operator's responsibility is to manage hot (online) and cold (offline) locks for the pool.

- Cold keys must be kept as secure as possible and must not be stored on an Internet-connected device. It is recommended to create multiple backups of the cold key.

KES key pair (Key Evolving Signature)

- KES is used to create an operating certificate for the block producer node, verifying the identity of the user.
- The KES key can evolve for a certain number of cycles, after which it becomes useless.
- This prevents re-attacking the history, even if the key is exposed. After the period expires, the node operator must generate a new KES key, issue a new operating node certificate, and restart the node.

Khóa VRF (Verifiable Random Function)

- VRF keys are used in the Ouroboros Praos protocol to enhance the security of block production.
- Unlike other protocols such as Ouroboros Classic, the slot leader schedule in Praos is kept secret. When the slot leader is selected, the VRF key proves the right to create the block.
- The VRF key is stored in the operating certificate and verifies that the node has the authority to create blocks in that slot.

Address lock

The address key represents the address functions derived from the key, used to identify assets on the blockchain. Keys include:

- **Payment key:** A single address key pair is commonly used to generate a UTXO address.
- **Khóa staking (Staking key):** The stake/reward address key pair is typically used to generate account/reward addresses.

Payment and staking address

You can have multiple addresses with ADA coins in your wallet. If you create a staking certificate and submit it to the Cardano blockchain, all tokens will be delegated to the pool you have chosen. This also applies to newly created addresses to which you send ADA from the exchange, for example. As soon as the next snapshot occurs in the Cardano network, the newly received ADA tokens will also be activated for staking.

To achieve the capabilities described above, *need to be separated* the tracking of transactions of ADA coins and their authorization. Within just a single wallet address, Cardano has an

address structure that differentiates between payment addresses and staking addresses (sometimes called reward addresses). The billing address is intended to hold funds that can be spent. The stake address determines if and how funds from the payment address are used in the stake.

In the image below, you can see the Shelley payment address, which includes a section for funds (payment credentials) and a reference to the stake address (stake key).

ADA coins always belong to the payment address (never the staking address). Each payment address can optionally refer to a stake address. Staking rights of all ADA tokens at the payment address are associated with the staking address.

The money at the payment address represents staking rights. The stake address determines how this permission is handled. Delegating ADA coins to a pool is done in two steps. First, the payment address must refer to the stake address. The stake address must then be delegated to the pool.

In the wallet, the user selects the pool he wants to authorize and confirms the transaction, which is sent to the blockchain. A stake certificate is generated implicitly, which authorizes funds to the selected pool via the stake address. During authorization, a rewards account is created in which the system accumulates staking rewards.

Note that the staking address is registered, not the payment address(es). Therefore, it is possible to perform a single registration for all future created payment addresses. Also, note that the funds remain on the payment address (fully controlled by the owner) and can be spent.

You can easily distinguish addresses from each other by prefix. The billing address has the prefix “addr”. The stake address has the prefix “stake”. Let's add that Byron addresses have no prefix and are encoded by Base58. The Shelley payment address and stake address are both encrypted by bech32.

Byron address:

37ctjaVyb4KDXBNC4haBVPCvro8AQPHwvCMp3RFhiSVWwfFmZ6iazSK6JK1hY6wHNmtrp1f1kdbva8TCneM2YsiXT7mrVr21EacHntXz5YyUdj84pe

Payment address:

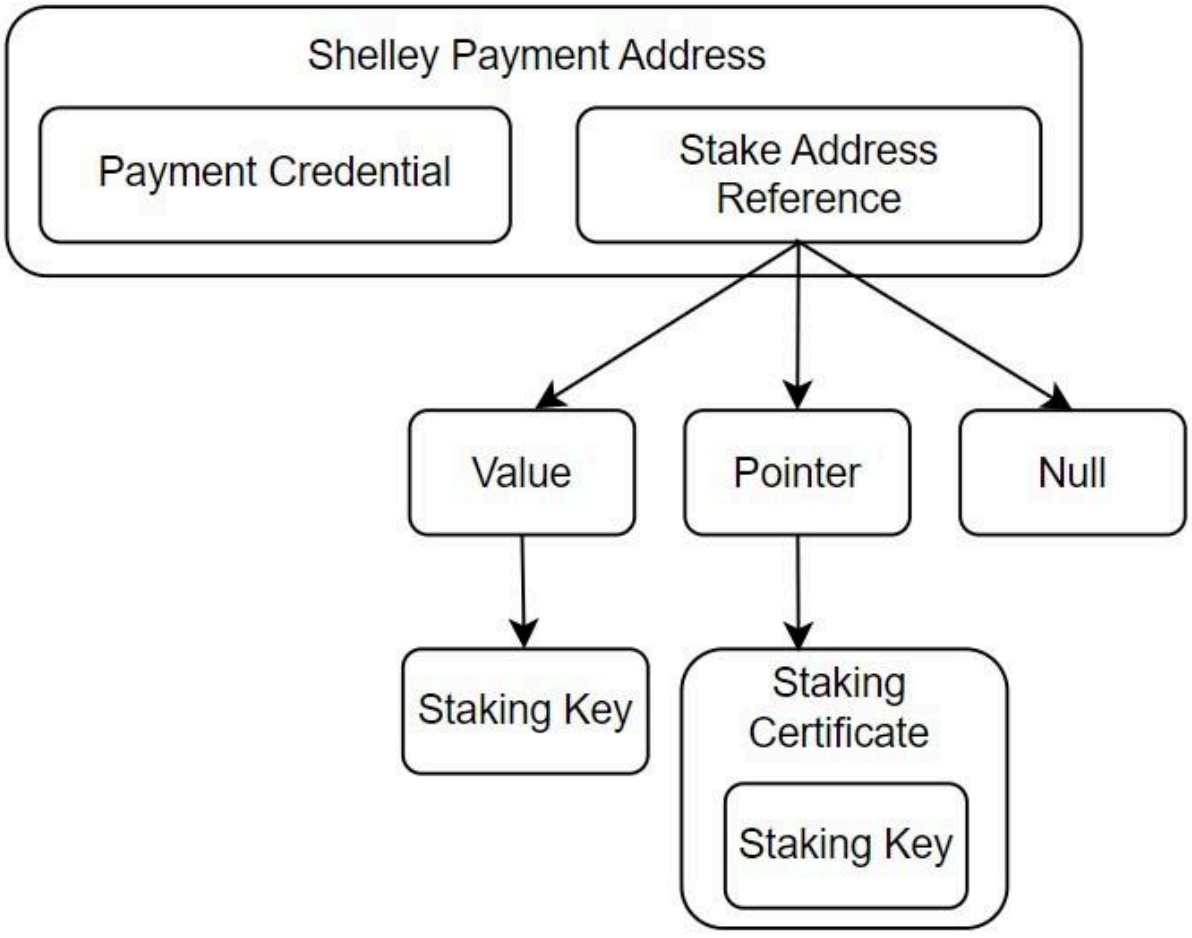
addr1vpu5vlrf4xkxv2qpwngf6cjhtw542ayty80v8dyr49rf5eg0yu80w

Stake address:

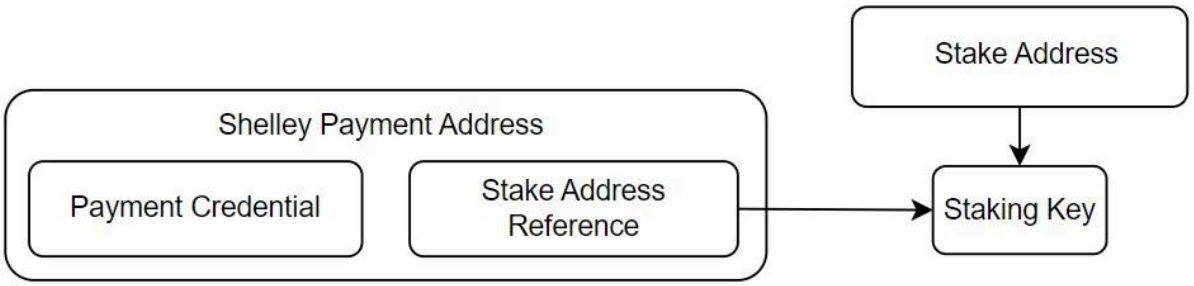
stake1vpu5vlrf4xkxv2qpwngf6cjhtw542ayty80v8dyr49rf5egfu2p0u

Stake Address Reference (Stake Address Reference)

There are three options for assets that can appear in the staking address reference of the Shelley payment address. Based on the content of the reference, we can divide the Shelley payment address into several types.



The Stake Address Reference can contain a so-called Value, which refers to the hash of the staking key or validation script. These addresses are called base addresses.

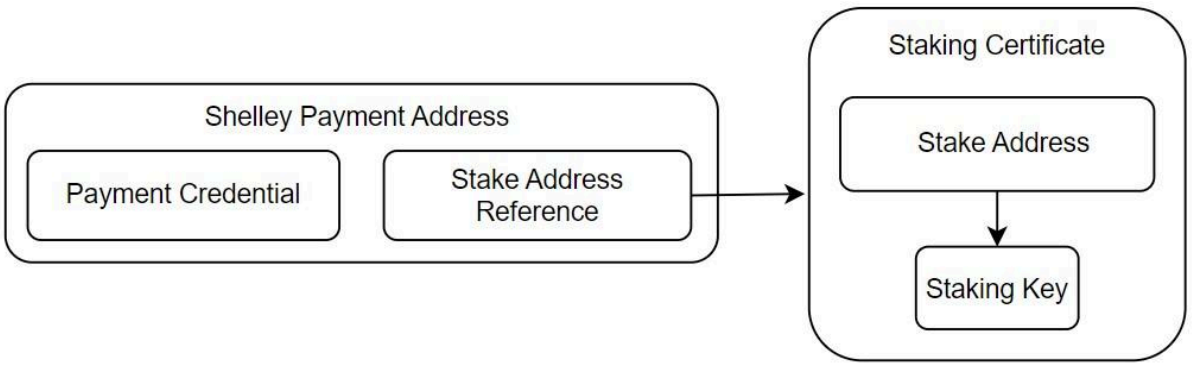


Let's add that instead of the stake key, the reference could refer to the hash of the script (i.e. the hash of the script that will be used for spending).

The stake key is used to control staking rights for all linked payment addresses. Staking keys are often owned by the same entity that owns the payment addresses, but this is not always the case. It is possible that someone other than the owner of the payment address has control over staking rights (e.g. smart contract). These addresses are called hybrid addresses. Note that another key is needed to spend money on payment addresses.

Furthermore, references can contain so-called Pointers. These addresses are called Pointer addresses.

In this case, the stake key is referenced indirectly through a Pointer. The reference points to the location in the blockchain where the stake certificate is stored. The stake key is stored in the certificate. Pointer occupies a smaller size than stake key. It only takes 3 numbers to find it: the slot index, the transaction index in the block, and the certificate index in the transaction.



Finally, the reference may not contain anything, only a value called Null. These addresses are called enterprise addresses.

In this case, the funds in the payment addresses cannot be linked to staking rights. In other words, ADA coins cannot be used for staking. This option is suitable, for example, for exchanges or other organizations that want to make it clear that they will not stake ADA.

There is one more type of address that you should know. This is not based on the content in the reference. It is a rewards account address.

The reward address is the hash value of the address's public stake key. They are used to distribute staking rewards. Unlike billing addresses that are based on the UTxO model, rewards accounts are based on an account-based model. Paying rewards regularly will only increase the account balance. As soon as a user withdraws rewards through a transaction, a new UTxO will be generated from the balance.

3.3.5. Smart contracts on Cardano

What is a smart contract?
Smart contracts are digital agreements defined by code that automate and enforce terms without the need for intermediaries. This allows for transparent and secure transactions on the blockchain. Based on pre-established conditions in the code, the state of the contract can only change in a way that complies with predetermined rules.

On the Cardano blockchain, the compiled code of smart contracts is stored and distributed on a decentralized network. Once deployed, the rules of the contract cannot be changed, nor can the compiled code be decompiled back into the original source code.

Introducing smart contracts on Cardano

Smart contracts on Cardano work differently compared to other blockchains. To understand smart contracts, it is necessary to first master the eUTxO model.

A smart contract is essentially a piece of code written to validate the movement of a locked UTXO at the contract address. When you lock a UTXO at the address of a script, that UTXO can only be spent or moved if the script confirms and authorizes the transaction to do so.

Overview of smart contract structure

Smart contracts include two main components:

1. **On-Chain Components (on-chain):**
   This is a validator script used to ensure that any transactions involving value locked at the contract's address comply with the contract's rules. Creating authentication tokens requires specialized tools and programming languages.
2. **Off-Chain Components (off-chain):**
   This is the code or application used to create transactions in accordance with the rules

of the contract. This component can be written in most popular programming languages.

Smart contracts often use data **datum** attached to the UTXO to maintain the "state" of the contract. If a UTXO has no data **datum**, it can be permanently locked on the contract's address.

On-Chain Component (Authentication Token)

The validator script is automatically executed when a UTXO at the contract address is moved through a transaction. This code will take information from the transaction as input and return the value **true** (correct) or **false** (false), decides whether the transaction is valid or not according to the specified rules.

- Each migrated UTXO activates its own authentication token.
- Execution of the validation token takes place on the Cardano node that is validating the transaction.

To activate the validation code, the transaction first needs to transfer a UTXO to the contract's address, which is mathematically generated from the contract code. This process is considered to initialize a contract version.

Off-Chain Components

The off-chain component is responsible for:

- Find locked UTXOs in the contract.
- Create valid transactions to move these UTXOs.

In contracts that require multiple execution steps, the state of the contract is encoded in **datum** and is attached to each transaction. Each step in the contract will change **datum** to reflect the new state.

Input structure of authentication code:

Smart contracts on Cardano are technically very simple, mainly based on validator scripts. These codes allow you to create rules or logic for the Cardano node to execute when confirming a transaction.

- **Datum:** The data attached to the UTXO is locked, often used to store state.

- **Redeemer:** Data from the transaction, providing information for the authentication token.
- **Context:** Information about the transaction, such as signature list, transaction value, and validity period.

The authentication code uses the above information to decide whether the transaction is valid or not.

Table xx: Information contained in Context

Parameter	**Describe**
inputs	Inputs for spending.
reference inputs	Use the inputs as reference.
outputs	Create new outlets using transactions.
fees	Transaction fees.
minted value	Minted or Burned value.
certificates	The certificates are included in the transaction.
withdrawals	Users can withdraw rewards upon delegation
valid range	Effective time zone of the transaction.
signatories	List of signatures.

redeemers	The data is used to provide input to the script from the spender.
info data	Hash code of datum.
id	Transaction ID.

Basic contract process

Note: This is just an example! The validator doesn't have to rely on a hash function - you can implement any logic you want.

1. **Components on string:**
   You create a validator-script that compares the datum value in the UTXO moved from the contract's address with the hash value of the redeemer used in that transaction.
2. **Off-chain components:**
   You create a script, using the programming language of your choice, to create a transaction that transfers an amount of ADA or other asset to the address of the validation script. When you create a transaction, you specify the datum value Hash("secret"), which ensures that only the hash value of the word "secret" is stored on the chain.
3. **Sign and send transaction:**
   You sign and send this transaction to a Cardano node, either directly or through one of the many APIs available such as Blockfrost or Dandelion. At this point, the amount of ADA you deposited into the contract will be locked by the validation script.
4. **UTxO spending rules above:**
   The only way to spend the UTxO just created in step 3 (move this locked amount of ADA) is to create a transaction with the word "secret" as the redeemer, because the UTxO is locked in the script, which will execute the process. The rule you create requires the redeemer's hash value to match Hash("secret").

Typically, the datum value will be much more complex, and the contract user may not know how it specifically works. So they will rely on your off-chain component to create transactions - this is typically provided by you as an API.

Smart contract programming language on Cardano

Cardano introduced smart contracts in 2021 and now supports smart contract development and deployment in a variety of languages including:

1. **Marlowe:**
   A domain-specific language (DSL) focused on financial contracts, making it easy for users to create complex financial contracts without the need for deep programming skills.
2. **Aiken:**
   Language optimized for on-chain authentication token generation, focusing on developer experience.
3. **Choose:**
   Python-based smart contract programming language, friendly to developers already familiar with Python.
4. **Rather:**
   Powerful platform that allows creating applications that interact with the Cardano blockchain.
5. **More:**
   A programming language embedded in TypeScript, as well as a library that supports creating transactions.

To write a well-designed smart contract, you first need to understand how Cardano works in general. You can then learn how to implement smart contracts in the supported languages as mentioned.

3.3.6 On-Chain Governance on Cardano

Cardano is moving towards a decentralized governance model that encourages community participation and ensures efficient decision-making. This model is detailed in CIP-1694, based on a tripartite structure and seven different types of governance actions.

Main roles in administration:

1. **ADA Holders:**

   - **Authorization of voting rights:** Voting rights can be delegated to Authorized Representatives (DReps).
   - **Register as DRep:** It is possible to register as a DRep by locking a deposit (dRepDeposit).
   - **Recommended administrative action:** A deposit (govActionDeposit) is required to propose administrative actions.
2. **Authorized Representative (DReps):**

   - Propose, discuss, and vote on protocol changes.

    - Voting rights are based on the amount of authorized stakes.
3. **Stake Pool Operators (SPOs):**
    - Maintain network infrastructure and participate in discussions and votes on changes.
    - Voting rights are based on active stakes.
4. **Constitutional Committee (CC):**
    - Ensure governance actions comply with the Cardano Constitution.
    - Provide balance of power and monitor transparency and fairness.

Decision making process:

- **Off-thread discussion:** Build consensus before submitting governance actions to the blockchain.
- **Voting:** Governance actions are decided through voting by DReps, SPOs, and CCs.
- **Execution:** Actions are executed at the epoch boundary after approval.

This governance model ensures transparency, fairness and high representation, while maintaining stability and sustainable development for the Cardano ecosystem.

Types of Governance Actions on Cardano

1. **Distrust:**
    - Proposal to create a state of no confidence in the current Constitutional Committee.
2. **New Constitutional Commission and/or Thresholds and/or Provisions:**
    - Change Constitutional Committee membership, signature threshold, or related provisions.
3. **Constitutional Update or Proposed Policy:**
    - Constitutional edits or proposed policies, stored as hashes on the chain.
4. **Initialize Hard Fork:**
    - Enable network backward incompatibility upgrades, requiring previous software upgrade.
5. **Change Protocol Parameters:**
    - Changes to updatable protocol parameters, excluding major protocol version changes ("hard forks").
6. **Withdrawal of Treasury Funds:**
    - Withdraw funds from the on-chain Cardano treasury.
7. **Information:**
    - Record information on the chain without causing any direct impact on the chain.

These types of actions ensure flexibility and efficiency in the governance of the Cardano ecosystem.

Questions and exercises

1. Who was created by Bitcoin, and what was its original goal?
2. How are Bitcoin transactions validated and added to the blockchain?
3. What ensures that Bitcoin transactions are irreversible and transparent?
4. What is the difference between public key and private key? What role do they play in Bitcoin transaction security?
5. How are Bitcoin addresses generated from public keys?
6. What is a Bitcoin wallet, and what types of wallets are there? Advantages and disadvantages of each type?
7. What is the role of nodes and miners in the Bitcoin network?
8. How has Bitcoin changed the way we view money and financial transactions?
9. What are the main challenges facing Bitcoin? (e.g. power consumption, scalability)
10. How is Bitcoin different from traditional financial systems in terms of transparency, security, and decentralization?
11. In what context was Cardano born, and what problems of previous generation blockchains does it aim to solve?
12. What important milestones did Cardano's launch phase (2015-2017) achieve?
13. What stages is Cardano's development roadmap divided into, and what are the main goals of each stage?
14. What is Cardano's future vision, and how will it impact the blockchain ecosystem?
15. What layers is the architecture of the Cardano Blockchain divided into, and what is the role of each layer?
16. How does Cardano's Ouroboros consensus mechanism work to ensure security and decentralization?
17. How is the UTxO model different from the account-based model?
18. What are the main benefits of Cardano's EUTxO model, and why is it suitable for smart contracts?
19. What are the basic components of EUTxO, and what role do they play in trading?
20. How are keys in Cardano used to ensure transaction security and authentication?
21. How are payment addresses and staking addresses different, and what role do they play in the Cardano ecosystem?
22. How do smart contracts on Cardano work in the EUTxO model?
23. What are the main differences between smart contracts on Cardano and other blockchains?
24. What are the main roles in Cardano's On-Chain governance, and what responsibilities do each role have?
25. What is the decision-making process in Cardano's On-Chain governance?
26. What are the types of governance actions on Cardano, and what is the purpose of each?

References

Bitcoin
https://learnmeabitcoin.com/
https://drive.google.com/file/d/1kG_oKReuoXi-yqhAda9rXPbL5T4KeEWJ/view?usp=drive_link
https://dl.ebooksworld.ir/motoman/Oreilly.Mastering.Bitcoin.Unlocking.Digital.Cryptocurrencies.www.EBooksWorld.ir.pdf

ETH
https://drive.google.com/file/d/1S9P5K4aeSNnvIIetKVgPNmSfrSGwqN_y/view?usp=drive_link

Cardano
https://docs.cardano.org/
https://drive.google.com/file/d/1Ci38r_r9MoAiELKiwjhtLb0txsRZpL_F/view?usp=drive_link
https://drive.google.com/file/d/1_XcsxjDhRH1kNDoV8dBjQTIMdSGFvTaQ/view?usp=drive_link

https://cexplorer.io/article/understanding-cardano-addresses
https://cexplorer.io/article/understanding-utxo-spending-through-a-script

https://www.ledger.com/academy/library/topic/blockchain

https://aft.acm.org/wp-content/uploads/2019/10/Ouroboros_AFT19_Tutorial.pdf