

UNIVERSITÀ  
DEGLI STUDI  
DI PADOVA



UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA DELL'INFORMAZIONE

# Exploring Recent Technological Advancements in the Bitcoin Network

*Relatore: Prof. Mauro Migliardi*

Laureando: *Edoardo Zanella*

Anno Accademico 2024/2025

Data di Laurea: 13-03-2025



## Abstract

During my degree in Information Engineering, I cultivated a deep interest in blockchain technologies, which I explored independently. This thesis examines recent innovations on Bitcoin such as Ordinals (2022) and Runes protocol (2024), I chose this topics driven by personal curiosity and professional aspirations. Academically, it enhances my technical understanding of Bitcoin's evolving ecosystem, while professionally, it positions me within the expanding field of decentralized systems.

Due to the recent emergence of these technologies, relevant academic literature is hard to find, so this study draws on web-based sources (white papers, developer documentation, and direct posts from interested people when available) and practical experiments, such as inscribing satoshis and issuing Runes tokens.

Ordinals and Runes leverage Bitcoin's security and decentralization to expand its functionality, aligning with trends in modern Distributed Ledger Technologies (DLTs) while raising concerns about scalability and network congestion.

The study concludes an outlook on their future development, highlighting their potential to enhance Bitcoin's ecosystem and the challenges that must be addressed for sustainable adoption.



# Contents

|   |           |
|---|-----------|
| <b>Introduction</b>   | <b>1</b>  |
| <b>1 Core Mechanisms of the Bitcoin Network</b>                   | <b>3</b>  |
| 1.1 Satoshi as the atomic value . . . . .                         | 3         |
| 1.1.1 Who is Satoshi? . . . . .                                   | 3         |
| 1.2 The UTXO model . . . . .                                      | 4         |
| 1.2.1 UTXO Model Vs Accounts Model . . . . .                      | 6         |
| 1.3 Addresses . . . . .   | 7         |
| 1.3.1 Legacy Address . . . . .                                    | 8         |
| 1.3.2 Native SegWit Address . . . . .                             | 9         |
| 1.3.3 Nested SegWit Address . . . . .                             | 10        |
| 1.3.4 Taproot Address . . . . .                                   | 11        |
| 1.4 Transactions . . . . .  | 13        |
| 1.4.1 Bitcoin’s Scripting language . . . . .                      | 14        |
| <b>2 The ordinal theory</b>                                       | <b>21</b> |
| 2.1 Ordinal Numbering System . . . . .                            | 22        |
| 2.2 Inscriptions and Digital Artifacts . . . . .                  | 24        |
| 2.2.1 How Data Is Inscribed Onto Satoshis . . . . .               | 25        |
| 2.3 Tools and Resources . . . . .                                 | 27        |
| 2.4 Controversies and Debates . . . . .                           | 28        |
| 2.4.1 How do ordinals make the Bitcoin network worse? . . . . .   | 28        |
| 2.4.2 How do ordinals add value to the bitcoin network? . . . . . | 30        |
| 2.5 Future Outlook . . . . .                                      | 31        |
| <b>3 The Runes Protocol</b>                                       | <b>35</b> |
| 3.1 Token standards on bitcoin . . . . .                          | 35        |
| 3.2 How do runes work . . . . .                                   | 37        |
| 3.3 Debates on Runes . . . . .                                    | 38        |

|                              |           |
|------------------------------|-----------|
| 3.4 Future Outlook . . . . . | 39        |
| <b>Conclusion</b>            | <b>43</b> |
| <b>Bibliography</b>          | <b>45</b> |

# Introduction

Since its inception in 2008 with Satoshi Nakamoto's white paper, Bitcoin has proven to be a relatively stable network (*stability here referring not to its price, but rather to its internal operational parameters*) solidifying its value as a peer-to-peer electronic cash system.

Although Bitcoin's primary goal has been, and continues to be, serving as a form of decentralized digital cash, recent technological advancements have enabled it to extend its possible applications. Among these developments, the introduction of the Ordinal Theory stands out as perhaps the most significant milestone by assigning identities to individual satoshis, Bitcoin's smallest unit. This theory has indeed paved the way for the creation of meta-protocols such as Inscriptions and Runes.

Inscriptions, launched in 2022, are unique digital artifacts similar to non-fungible tokens (NFTs) and their content is tied to one specific satoshi. Runes is instead a fungible token standard introduced in 2024; it offers an efficient framework to issue fungible tokens directly on Bitcoin's blockchain, addressing limitations of earlier standards like BRC-20.

These advancements reflect a broader trend within distributed ledger technologies (*DLTs*), where blockchain networks are increasingly leveraged for diverse use cases, from digital art to decentralized finance (DeFi). Unlike other DLTs such as Ethereum or Solana, which prioritize flexibility and smart contract functionalities, Bitcoin's design emphasizes security, decentralization, and immutability. Ordinals and Runes embrace the same philosophy, allowing for new use cases within Bitcoin's existing infrastructure, without requiring fundamental changes to its protocol. Their adoption has, however, sparked debates about scalability, network congestion, and fidelity to Bitcoin's original vision.

This thesis aims to explore these recent technological advancements, providing a technical overview of Bitcoin's foundational mechanisms (*such as the UTXO model and transaction scripting*), before delving into the mechanics, implications, and future prospects of Ordinals and Runes. By examining their role within Bitcoin and the wider DLT landscape, this work seeks to assess their potential to enhance Bitcoin's utility while addressing the challenges they pose to its ecosystem.

# Chapter 1

## Core Mechanisms of the Bitcoin Network

Bitcoin's ability to support recent innovations, such as the Ordinal Theory and the Runes Protocol, rests on its robust technical foundation, built over time by contributors with Bitcoin Improvement Proposals (BIPs).

This chapter explores the core components of Bitcoin's architecture, including the concept of satoshis as its smallest unit, the Unspent Transaction Output (UTXO) model for tracking ownership, the evolution of address formats, and the scripting language that governs transactions.

These elements serve as building blocks for advanced applications discussed in later chapters. While diving into these concepts, this section lays the foundations to analyze how new meta-protocols leverage the UTXO infrastructure, enabling it to transcend its original purpose while preserving its foundational principles.

### 1.1 Satoshi as the atomic value

Satoshis, not bitcoins, are the atomic native currency of the Bitcoin Network. One bitcoin is divisible into 100,000,000 satoshis, with each satoshi equivalent to 0.00000001 bitcoins.

#### 1.1.1 Who is Satoshi?

"Satoshis" usually abbreviated to "sats" are named after the anonymous inventor/s of Bitcoin who operated under the pseudonym Satoshi Nakamoto. He released the Bitcoin white paper titled "Bitcoin: A Peer-to-Peer Electronic Cash

System” [1] on October 31, 2008. He was the founder of the Bitcoin talk forum and has been active from November 2009 to December 2010. During this period, Nakamoto developed Bitcoin’s initial software and engaged in community discussions.

*Link to [bitcointalk.org](https://bitcointalk.org) Satoshi’s profile (last active in 2010)*  
*<https://bitcointalk.org/index.php?action=profile;u=3>*

## 1.2 The UTXO model

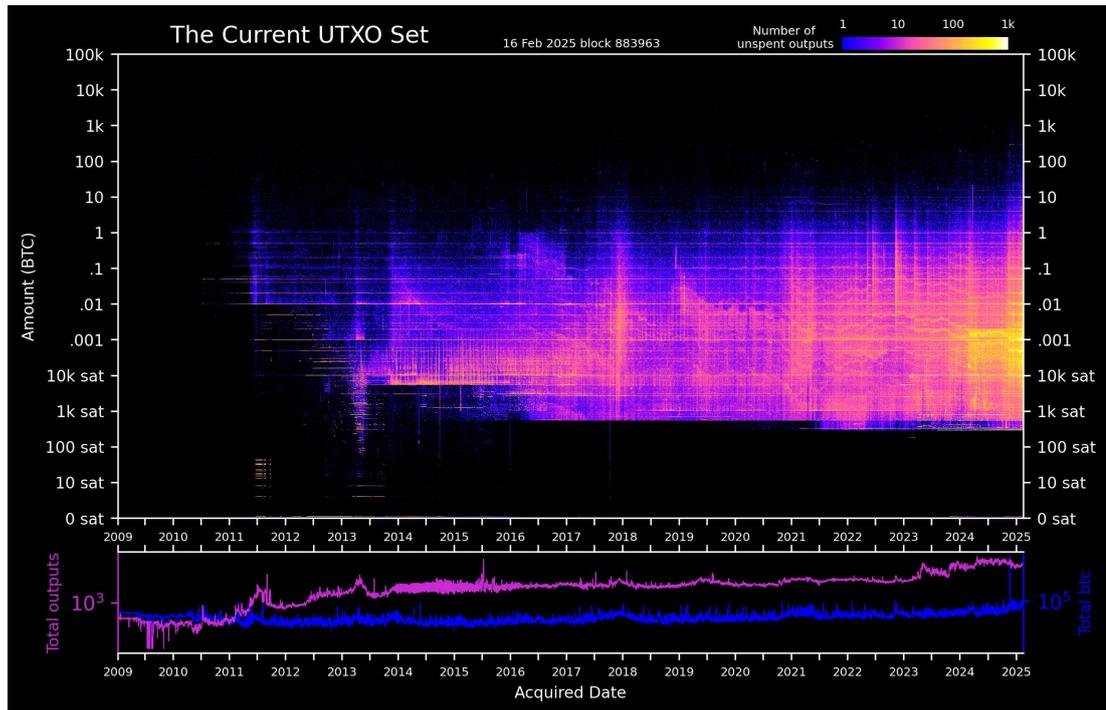
Bitcoin operates on a unique model known as the UTXO (Unspent Transaction Output) model. This model plays a critical role in how transactions are processed and verified on the Bitcoin network. Understanding this model and how different UTXOs are managed fundamental to exploring more advanced concepts.

Bitcoin transactions must reference specific UTXOs, making it nearly impossible to add an invalid transaction to the blockchain. The UTXO model makes Bitcoin more auditable and transparent than traditional financial systems, which rely on 1.2.1 accounts, balances, and third parties.

While the public ledger is transparent, there are ways to maintain a certain level of privacy: the standard practice in Bitcoin is to use one-time addresses, using a new one for each UTXO received. This approach obscures links between the various transaction of a user, reducing the ability to track funds across multiple payments without advanced analysis.

Unspent Transaction Outputs are specific amounts of bitcoin that a wallet received but hasn’t yet spent. A UTXO can be conceptualized as analogous to a physical bill: an object representing a discrete amount of bitcoin. Unlike currencies issued by a government like Euros or American Dollars, UTXOs can hold any arbitrary value of bitcoin.

The UTXO set is the collection of all the UTXOs on the Bitcoin blockchain at any moment, Bitcoin nodes use this set to agree on what coins exist and who has ownership over them.



**Figure 1.1:** UTXO set - block height 883963 - 16 Feb 2025

UTXOs are created in two primary contexts:

1. as outputs of transactions that spend existing UTXOs 1.4
2. as new outputs introduced via coinbase transactions when a block is mined.  
The first transaction in each block is called "Coinbase Transaction" and represents the payout from the network to the miner that successfully resolved the mathematical problem of finding the right hash that matches the hash of the block.

*Note how a Coinbase transaction is the only type of bitcoin transaction that is considered valid even if it has no inputs.*

Example Coinbase transaction of block 884035:

`mempool.space/tx/`

`280c8af21ba7e0e3ae3821886b113146cd42861d86192494fd4e76ee320752c2`

Usually UTXO logic is handled by the wallet's software, but what happens under the hood is that the wallet will select enough UTXOs to cover the output amount, and since UTXOs cannot be divided if the actual output is greater than the intended one all the excess bitcoin will be given back to the user as a new "change" UTXO. Similar to cash payments, when paying \$50 with \$100 bill, you can't rip

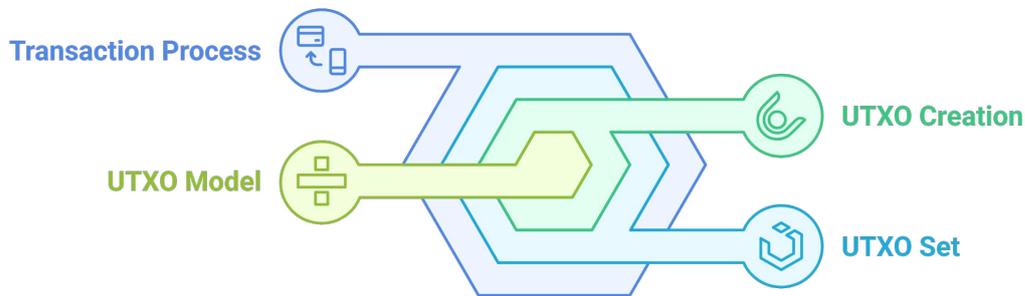


Figure 1.2: UTXO diagram

it in half, but instead you have to use the full note and the cashier will pay you your change back.

### 1.2.1 UTXO Model Vs Accounts Model

The UTXO model and accounts model are two distinct systems used to record and manage transactions. [5]

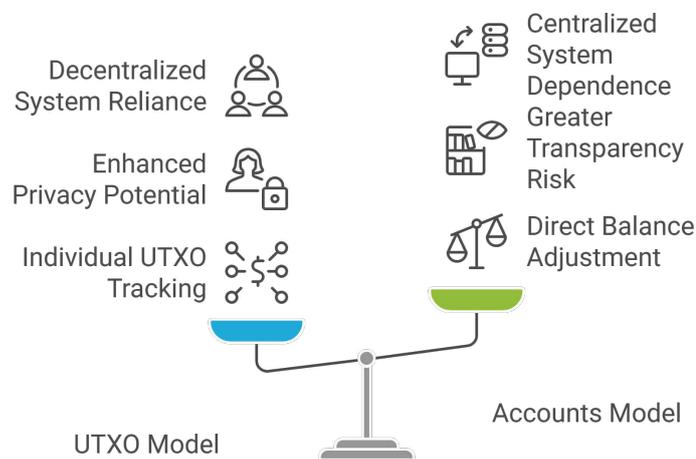
**UTXO Model** the balance in a user’s wallet is not a single figure. Instead, it’s the total of all “unspent” outputs from previous transactions. The UTXO model doesn’t adjust a stored balance figure with each transaction. Instead, it keeps track of individual pieces of bitcoin (UTXOs) that have been received and not yet spent. Please note that it is most likely impossible for an external observer to link 2 addresses owned by the same person, if privacy best practices are applied correctly.

**Accounts Model** it is a balance-based approach similar to traditional banking. When a transaction occurs, the system simply deducts the specified amount from one account and adds it to another, directly adjusting the balances. Most traditional financial systems, like any “FIAT” bank, use the accounts model.

Not all cryptocurrencies follow the UTXO model like Bitcoin. For instance, Ethereum employs the accounts model, which sacrifices some of the transparency and auditability inherent in the UTXO model:

If a malicious actor wanted to determine how much money Alice has in her Ethereum wallet, they would only need to locate her account and check its balance.

The concept is also valid with fiat currencies, the account balance is not stored on a public blockchain but centralized servers. For bitcoin however, the issue is different, because to find the balance the malicious actor would have to add every one of the UTXOs owned by Alice up, and assuming that Alice is using one-off addresses, this doesn't become impossible but rather complicated. Users in account models also risk chargebacks and overdrawn accounts.



**Figure 1.3:** Comparing Transaction Models in Cryptocurrencies

## 1.3 Addresses

Bitcoin addresses are derived from the public portion of an ECDSA (Elliptic Curve Digital Signature Algorithm) keypair and correspond to specific locking scripts that secure bitcoins.

At a lower level, sending bitcoins to an address involves creating a transaction output that locks the funds to a script pattern defined by the address, typically using a public key hash or script hash.

There are multiple address standards supported, and each one needs a different procedure to be generated. [6]

### 1.3.1 Legacy Address

**Examples:**

**P2PKH:** 1FnfdZmSrQ7N1ud3DpeFEjEb6bhRpmu2zT

**P2SH:** 37jPCtUg6roL4SmuUrQSbingQ48baavTCc

Legacy addresses for direct payments utilize the P2PKH (*Pay-to-Public-Key-Hash*) script and incorporate a version byte prefix of 00, ensuring the resulting Base58 encoded address begins with 1, which helps distinguish it from a similar looking P2SH address.

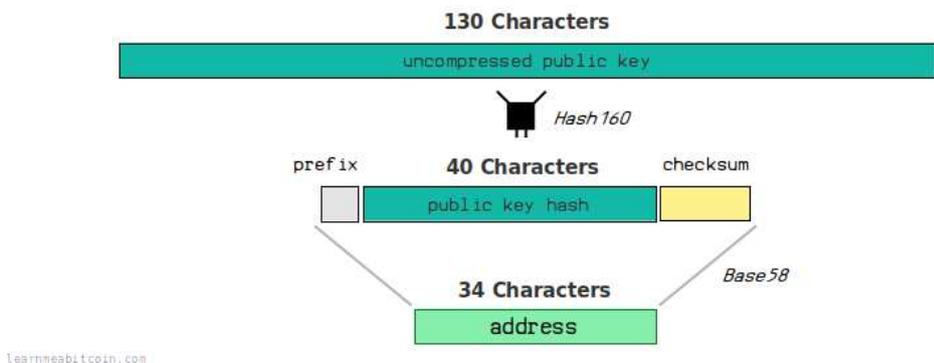


Figure 1.4: Legacy P2PKH (pay to public key hash) address

The transactions involving only Legacy locking and unlocking scripts are executed by Bitcoin's native scripting language, simply called Script 1.4.1.

The funds can be spent if the owner of the hashed public key provides the original public key, along with a valid signature in the ScriptSig field 1.4.1.

An other legacy format is **Pay-to-Script-Hash (P2SH)**, which extends P2PKH capabilities by allowing the execution of more complex scripts, for example multisig transactions. These transactions are useful for allowing multiple parties to sign and verify the content of the same transaction. The creation process is almost the same as P2PKH addresses, but we use the script hash instead of the public key hash, and the version byte prefix is 05 to force the first character of the address to be a 3

When Satoshi (*see Section 1.1.1*) originally published the first versions of Bitcoin Core, legacy addresses used the P2PK (Pay-to-Public-Key) script; This approach has been supplanted by P2PKH. In practice P2PKH improves usability, because it has an integrated checksum (to detect input errors) and the address is shorter

(34 instead of 130 characters) because of the conversion to base58 encoding; this all came at the cost of a small tweak to the unlock logic.

### 1.3.2 Native SegWit Address

**Examples: P2WPKH:** bc1q49katt6nj3neqsk76u5ejzkmu0rvrwmqj5xvzs

**P2WSH:** bc1qkuavpqr8zjg54n0m7sr2gmf9s08c0t8ts77qd3txhe68w22cpd2sdwzkwg

Native SegWit addresses employ the P2WPKH (Pay-to-Witness-Public-Key-Hash), they usually start with **bc1q**, use **bech32** encoding, and allow for un-malleable transactions, meaning that they prevent third party unconfirmed transaction alteration, ergo you can confidently spend the outputs of P2WPKH transactions while they are still in the memory pool.

The screenshot shows a 'ScriptPubKey' form. Under 'Version', the 'OP\_0 (P2WPKH or P2WSH)' radio button is selected. The 'Data (public key hash)' field contains the hexadecimal string '302ae54d7ea5f03be4ccf8e7e969f8bfff28c996', with a note '20 bytes' below it. The 'Hex' field displays '0014302ae54d7ea5f03be4ccf8e7e969f8bfff28c996' with a note '22 bytes' and 'Type: P2WPKH'. Under 'Network', the 'Mainnet' radio button is selected. The 'Address' field shows the bech32 encoding 'bc1qxq4w2nt75chrhexv1rn7j60chl1j3jvk3hluv9' with a note '42 characters'.

**Figure 1.5:** Example of a Native SegWit Address stemmed from a public key hash

Segwit transactions execution logic is similar to Legacy transactions, with the key difference being that relocation of signature data to the Witness field, separate from the Script execution (see Section ??).

Native Segwit is now the most common and efficient address type because transactions involving SegWit addresses have a lower weight multiplier and are subject to lower fees.

The funds can be spent if the owner of the hashed public key provides the original public key, along with a valid signature in the Witness field ??.

SegWit also added support for another script type Pay-to-Witness-Script-Hash (P2WSH), which allowed for un-malleable multisig transactions that are cheaper compared to P2SH because they store data in the witness field instead of the ScriptSig 1.4.1 field.

The screenshot displays a 'ScriptPubKey' configuration window. It features a 'Version' section with two radio buttons: 'OP\_0 (P2WPKH or P2WSH)' which is selected, and 'OP\_1 (P2TR)'. To the right, the 'Data (script hash)' field contains the hexadecimal string 'e595dd4d0eb8c7eaf264d68d264b8e49b32a6019659b3029e4cb69bc3b244f6f', with a note indicating it is '32 bytes' long. Below this, the 'Hex' field shows the full 34-byte hexadecimal representation: '0020e595dd4d0eb8c7eaf264d68d264b8e49b32a6019659b3029e4cb69bc3b244f6f', labeled as '34 bytes' and 'Type: P2WSH'. The 'Network' section has 'Mainnet' selected over 'Testnet'. Finally, the 'Address' field shows the Bech32 encoding: 'bc1quk2a6ngwhrr74uny66xjvjuwfxej5cqvkdq20yed5mcweyfahsuyjccd', noted as '62 characters' long.

**Figure 1.6:** Example of a Native SegWit Address stemmed from a script hash

P2WPKH and P2WSH addresses are distinguishable, addresses stemmed from a public key hash (20 bytes) have a length of 42 characters, as opposed to addresses stemmed from a script hash (32 bytes), that are 62 characters long.

### 1.3.3 Nested SegWit Address

Nested SegWit addresses can use a few script types: P2SH, P2SH-P2WPKH, P2SH - P2WSH, instead of sending to a public key hash, bitcoins are instead sent to a hash of that script, hence Pay-To-Script-Hash (P2SH).

These address formats allow for backwards compatibility of SegWit, one of the ways that made SegWit backward compatible with older wallets was by “nesting” a Native Segwit (P2WPKH) script inside a P2SH script. Hence the name “Nested” SegWit (P2SH-P2WPKH), since the Native SegWit script is nested or used in place of a P2SH RedeemScript to lock the utxo.

A Nested Segwit address is a normal P2SH address (starting with 3), but the RedeemScript is the ScriptPubKey of P2WPKH. Although this address format is presented after Native Segwit, it has been introduced earlier to allow for backwards compatibility with non-SegWit nodes while supporting SegWit multi-signature transactions. Addresses starting with a 3, could now be a SegWit compatible address or a multi-signature non-SegWit address. distinguishable only by examining the spending conditions. As a general rule of thumb today though, an address starting with a 3 is usually a Nested Segwit (P2SH-P2WPKH) address.

### 1.3.4 Taproot Address

**Example P2WSH:** bc1pq4y8vpn2n5gnzxxz...en3w0m25t8ysnkny07

All Taproot addresses start with `bc1p` and it's the most convenient address standard for interacting with Bitcoin Ordinals 2 and Runes 3; Ordinals are similar to NFTs on other blockchains, instead BRC-20 and Runes are two fungible tokens standard on Bitcoin, similar to other token standards like ERC20 on Ethereum, but with a new set of limitations imposed by Bitcoin's UTXO model 1.2

The screenshot shows a 'ScriptPubKey' configuration window. It has two main sections: 'Version' and 'Data (tweaked public key)'. In the 'Version' section, there are two radio buttons: 'OP\_0 (P2MPKH or P2MSH)' and 'OP\_1 (P2TR)'. The 'OP\_1 (P2TR)' option is selected. The 'Data (tweaked public key)' section contains a text box with the hexadecimal value '02c33e122953cf42a177c896819e950cf88948e422eba6da77e5abaab0f663af', with '32 bytes' indicated below it. Below these sections is a 'Hex' field containing the full hexadecimal string '512002c33e122953cf42a177c896819e950cf88948e422eba6da77e5abaab0f663af', with '34 bytes' and 'Type: P2TR' indicated below it. At the bottom, there is a 'Network' section with 'Mainnet' selected and 'Testnet' unselected. Finally, an 'Address' section shows the Bech32 encoding 'bc1pqtpnuy3f20859gtheztgr854pnugjj8yyt46dknhuk464v8kvwhsfc7y65' with '62 characters' indicated below it.

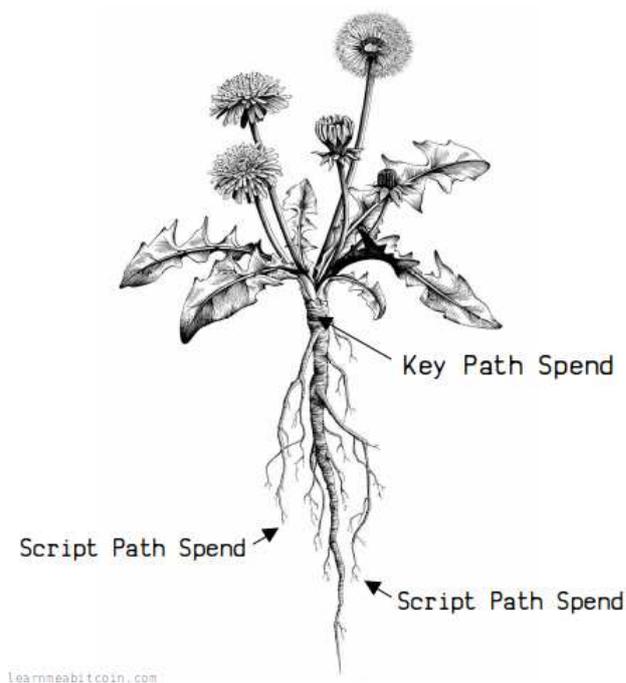
**Figure 1.7:** Example of a Taproot Address stemmed from a tweaked public key

In short, the **Taproot upgrade provides the flexibility to place a variety of different locking conditions on an output.**

It would be reasonable to think that spending such UTXOs is expensive, because of the large locking script dimensions

$$\text{fee} = \text{transaction weight} \cdot \text{fee rate}$$

, instead all of the possible spending conditions can be embedded within a single 32-byte "tweaked public key" (aka the taproot) thanks the merkle tree structure of the locking script.



**Figure 1.8:** Dandelion illustration by Janine Wiget

”The name originated in a visualization of a tree with a thick central trunk like a dandelion taproot – the technique is mostly useful because of the assumption that there is one high probability path and the rest is fuzzy stragglers, and I thought it was a good one because of the punny fact that it verifies script-path spends by tapping into the hidden commitment in the root.”

*Gregory Maxwell*, Bitcoin Optech Github (Pull Request 667)

An other advantage of this implementation, is that only the executed spending condition is revealed, hiding alternative conditions and enhancing privacy, and since the merkle tree structure is arbitrary, we are incentivized to create a binary tree and to fill the shortest paths in the tree with the scripts that we are more likely to use.

P2TR scripts allow for better script flexibility (more than 1 unlocking option) potential lower fees (if we chose an unlocking option near to the root of the merkle tree) in a way that is both private and efficient.

## 1.4 Transactions

Bitcoin Transactions are the fundamental mechanism to transfer the ownership of bitcoins.

”A digital coin contains the public key of its owner. To transfer it, the owner signs the coin together with the public key of the next owner. Anyone can check the signatures to verify the change of ownership.”

*Satoshi Nakamoto*, p2pfoundation.ning.com

This quote from Satoshi provides a useful insight to grasp how transactions work, where the ”digital coin” refers to a UTXO (Unspent Transaction Output). Even though transactions can be executed in more than one way, they all have something in common:

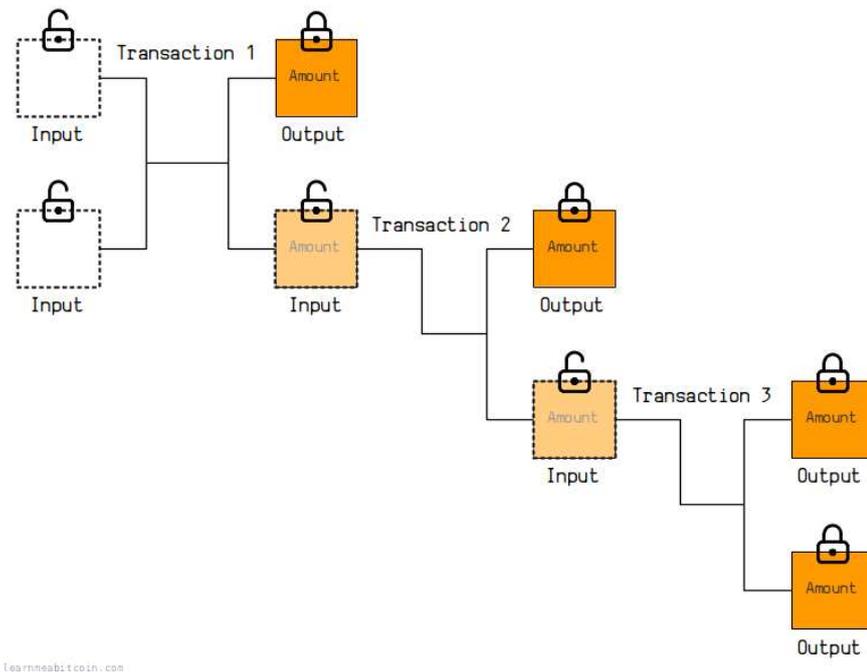
Every transaction carries the necessary data to

1. Cryptographically prove that you are the owner of the UTXO(s) you are trying to spend.
2. Communicate to other network participants how that UTXO(s) may be spent in the future and by whom.

So each transaction consists in unlocking input UTXO(s), arbitrarily redistributing the bitcoins in the old UTXO(s) into the new ones, and locking the new UTXO(s).

The logic is recursive (*see figure 1.9*): every input UTXO in a transaction has been taken from the output of another transaction. Where does this recursion stop? The base case is when a UTXO is created the first time in a ”coinbase transaction” 1.2, in other words when a miner successfully mines a new block, a new UTXO containing the block reward is created from scratch and given to him.

Every transaction is broadcasted from a network participant and validated by nodes before miners include them in the blockchain, there is only one rule: the script containing the cryptographic proof of ownership must be resolved with no errors, otherwise the transaction will be dropped.



**Figure 1.9:** Example of a Transaction chain

Transactions can be identified through their own unique identifier, known as a Transaction ID (TXID). A TXID is a 32-byte hash, created by double-SHA256ing the serialized raw transaction data; For legacy transactions, this includes all fields, whereas SegWit transactions exclude the Witness data:

*this is a simplified example*

`TXID = HASH256([version] [inputs] [outputs] [locktime])`

This creates a completely unique hash result for each transaction. The TXID is extremely important information, as it's needed when searching for a transaction in a blockchain explorer, or when you want to spend the output of a transaction in a new transaction.

### 1.4.1 Bitcoin's Scripting language

Bitcoin transactions are validated by its native scripting language (called Script), or some variations of it (e.g., SegWit or Taproot).

Script is intentionally not Turing complete, lacks several logical functions including loops. This design was chosen not to waste computing power and to protect

nodes across the network.

Script defines the cryptographic conditions required to lock and unlock bitcoin, not to build applications or run programs, in other words, the script of a Bitcoin transaction determines to whom the bitcoin was sent.

Bitcoin Script is based on a stack-based architecture, where data is organized and processed using a Last-In-First-Out (LIFO) structure.

The stack serves as a temporary storage mechanism for operands and results of operations, allowing the script to manipulate and operate on these values efficiently. As transactions are processed, the stack dynamically changes, reflecting the state of the script's execution.

[7] One unique feature about Script is its use of Reverse-Polish Notation (RPN) to express operations. In RPN, operators are placed after their operands, eliminating the need for parentheses and order of operations rules (e.g., `3 4 +` instead of `3 + 4`).

Two fundamental components in the Bitcoin Script framework are the locking script and the unlocking script. These two scripts work together across every Bitcoin transaction, cryptographically enabling the transfer of bitcoins.

### Locking Scripts

The **scriptPubKey** is the locking script. Placed onto an UTXO, this script specifies the conditions that must be met to spend the bitcoins in the future. It effectively acts as a lock, ensuring that only the intended recipient can unlock and claim the bitcoins. This script can encode conditions such as requiring a correct digital signature or even more complex criteria like multisig requirements, time locks, or even the resolution of a cryptographic puzzle.

Here are the most common locking code patterns:

1. P2PKH/P2SH (legacy) - P2PKH locks outputs to a public key hash, requiring the public key and a signature; P2SH locks to a script hash, requiring the script and its conditions to be met.

Example: `76a914{publickeyhash}88ac/a914{scripthash}87`

2. P2WPKH/P2WSH (SegWit)- Lock the output to the *hash of a public key/hash of custom script*. Works the same as a P2PKH/P2SH, but

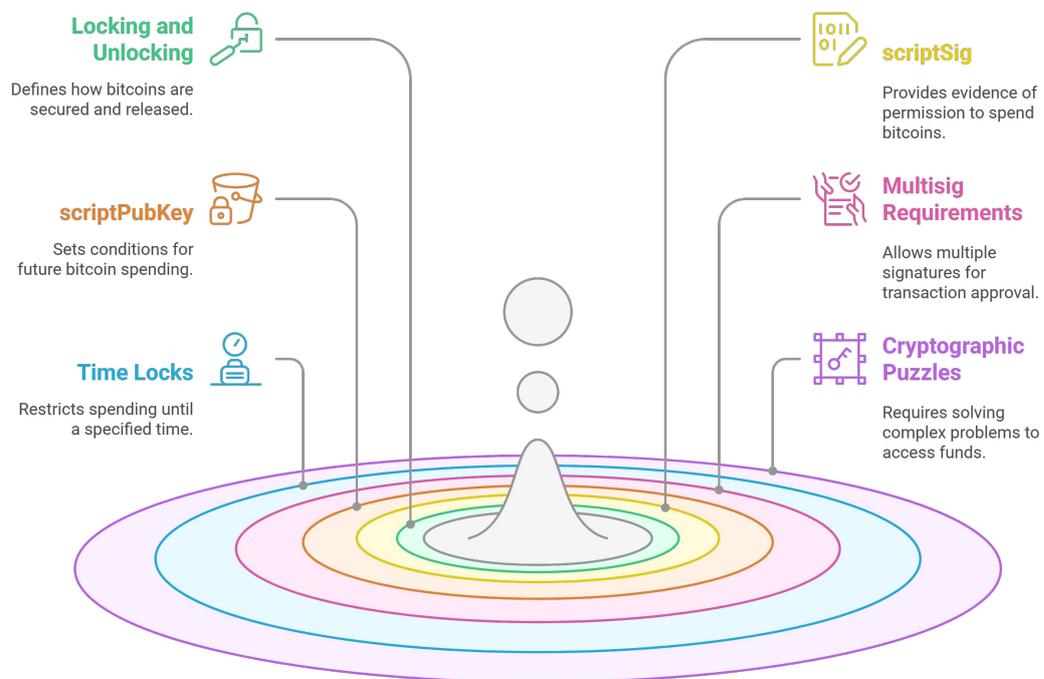


Figure 1.10: Bitcoin Script Functions

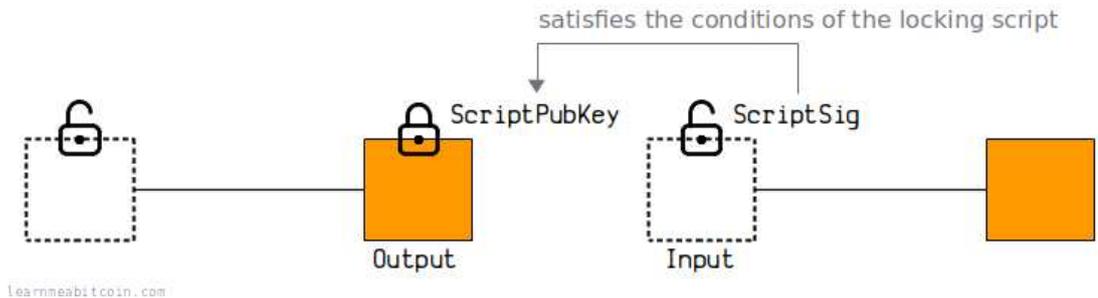


Figure 1.11: Visualization of the ScriptSig unlocking mechanism

the **unlocking code goes in the witness field instead of the scriptsig field.**

Example: `0014{publickeyhash}/0020{scripthash}`

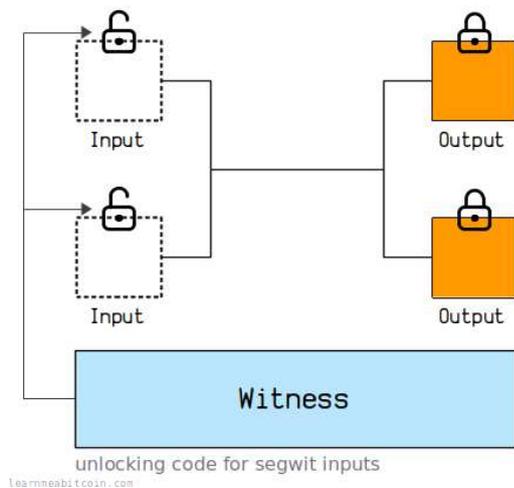
### Unlocking Scripts

There are now two primary methods to unlock UTXOs.

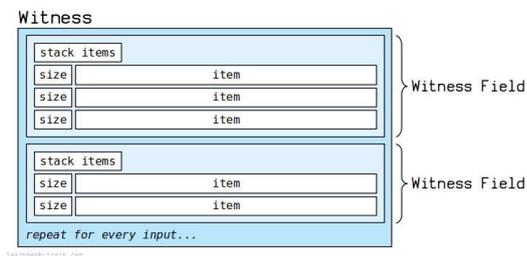
1. Legacy locking scripts such as P2PK, P2PKH, P2MS, and P2SH get unlocked by the ScriptSig field within a transaction input. The scriptPubKey lays out the conditions under which the funds can be claimed, and the **scriptSig provides the solution to this cryptographic challenge**. During the verification process, the scriptSig is executed first, followed by the scriptPubKey. If the final result is true, the transaction is valid, and the funds can be spent; otherwise, the transaction is rejected.
2. Segwit transactions (inputs that were locked by newer locking scripts such as P2WPKH, P2WSH, and P2TR) get unlocked within the **Witness** section of a transaction. The witness section does not override the scriptSig section which is normally used for legacy transactions.

Before the Segregated Witness update in 2016, the TXIDs for legacy transactions were created from the entire transaction data, including the signatures, allowing for the exploit of transaction IDs: when someone pushed a chain of linked transactions into the mempool, the hash of the first transaction could be modified, and the rest of the chain would be dropped because considered invalid by the network.

The Witness field was introduced to fix this possible exploit, in fact in SegWit transactions signatures are uploaded in this new field which is



**Figure 1.12:** Visualization of the Witness unlocking mechanism



**Figure 1.13:** Structure of the witness object

not included as part of the TXID calculation.

*Legacy transactions are still vulnerable to this kind of exploit, since they rely on scriptSig*

A witness field works in the same way as the ScriptSig field, but instead of using `OP_CODES` to push items on to the stack, you use compact size fields before each item that's the reason why SegWit locks do not actually use the Script language, because they're just hard coded patterns of bytes that get executed in a specific way. Therefore, you can't use any `OP_CODES` inside it to perform logical operations; all you can do is push data on to the stack, which is all that is required for unlocking the most common types of locking scripts anyway. So witness data is a simplified alternative to Script that only contains data instead of opcodes.

Bitcoin scripts are the foundation of trust in the Bitcoin network, as they allow the decentralized network to work while trust is not placed in a central authority, but in the cryptographic proofs that the scripts provide.

| Aspect               | Witness (SegWit)                                   | ScriptSig (Legacy)                      |
|----------------------|--|---|
| Data Placement       | Witness section (separate)                         | ScriptSig field (within input)          |
| Malleability         | Reduced (Witness excluded from TXID)               | Vulnerable (ScriptSig included in TXID) |
| Efficiency           | Higher (block weight, lower fees)                  | Lower (larger size, higher fees)        |
| Security/Flexibility | Enhanced (e.g., P2TR for privacy, smart contracts) | Basic (limited to simpler scripts)      |
| Compatibility        | Requires modern support                            | Universally supported                   |

**Table 1.1:** Comparison between Witness Standard (SegWit) and ScriptSig Standard (Legacy)



## Chapter 2

# The ordinal theory

Bitcoin has historically been viewed as a conservative network, prioritizing security and decentralization over the programmability seen in DLTs like Ethereum or Solana. The introduction of the Ordinal theory [2] by Casey Rodarmor, in November 2022, marks a significant shift, enabling Bitcoin to support non-fungible digital artifacts without altering its core protocol. Unlike Ethereum's ERC-721 standard for NFTs, which relies on smart contracts and external data storage, inscriptions embed data directly onto Bitcoin's blockchain using the taproot upgrade, offering unmatched immutability and censorship resistance.

Bitcoin has no notion of stable, public accounts or identities: addresses are single-use, and wallet accounts are private. Roadmor's proposal is motivated by the desire to provide stable identifiers within the network, that may be used by Bitcoin applications.

The Ordinal Theory is a method to give each satoshi (See chapter 1.1) a unique identity based on the order it was mined. This allows satoshis to be tracked and transferred individually, without needing changes to the Bitcoin network. Ordinals enable the creation of digital artifacts called inscriptions, similar to non-fungible tokens (NFTs), directly on Bitcoin, enhancing its use beyond simple currency transactions.

This chapter explores the mechanics of Ordinals, their implications for Bitcoin's ecosystem, and their role in bridging Bitcoin with the growing domain of digital collectibles.

## 2.1 Ordinal Numbering System

The foundation of the ordinal theory is being able to isolate and identify every single satoshi, but how is this impressive result achieved?

As discussed in the Ordinal Numbers Bitcoin Improvement Proposal there are several notations to express the identity of a satoshi, in the table below the same satoshi is expressed in 5 different notations

See the example satoshi on the explorer: <https://ordinals.com/sat/2099994106992659>

| Notation            | Description  | Example                |
|---------------------|--|------------------------|
| Integer notation    | The ordinal number assigned according to mining order  | 2099994106992659       |
| Decimal notation    | Block height and offset within the block   | 3891094.16797          |
| Degree notation     | Represents cycle, block in halving epoch, block in difficulty adjustment, and sat index: A°B‘ C“ D““ | 3°111094‘214‘‘16797‘‘‘ |
| Percentile notation | Satoshi’s position in Bitcoin supply as a percentage   | 99.99971949060254%     |
| Name                | Encoding using characters a to z, e.g., <code>satoshi</code> for the first satoshi                   | satoshi                |

**Table 2.1:** Different notation formats used in Ordinals

Bitcoin has periodic events, some frequent, some more uncommon, and these naturally lend themselves to a system of rarity. These periodic events are:

**Blocks:** A new block is mined approximately every 10 minutes, from now until the end of time.

**Difficulty adjustments:** Every 2016 blocks, or approximately every two weeks, the Bitcoin network responds to changes in hashrate

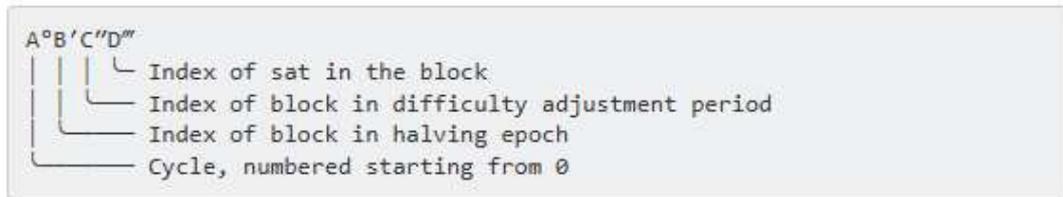
by adjusting the difficulty target which blocks must meet in order to be accepted.

**Halvings:** Every 210,000 blocks, or roughly every four years, the amount of new sats created in every block is cut in half.

**Cycles:** Every six halvings, something magical happens: the halving and the difficulty adjustment coincide. This is called a conjunction, and the time period between conjunctions a cycle. A conjunction occurs roughly every 24 years. The first conjunction should happen some time in 2032.

*From: <https://rodarmor.com/blog/ordinal-theory/>*

That's why, from a collector's point of view, the Degree notation is so important, because it expresses the sat's position in a Bitcoin-native time measurement.



**Figure 2.1:** Degree notation

There is no standard for how rare a sat can be, even though this bitcoin native way of measuring time suggests that some satoasis are different from the others, for example:

**uncommon** The first sat of each block

**rare** The first sat of each difficulty adjustment period

**epic** The first sat of each halving epoch

and so on.

Collectors then started trading satoasis based on a wide variety of factors, there are hundreds of different categories but here are some interesting examples:

**Palindrome sats** palindrome numbers in general appreciated by bitcoin collectors: you can find Palindrome sats (sats whose number reads the same backward or forward), but also other types of combinations like **Paliblock**

**Palindrome** (a palindrome sat in a palindrome block) and even more complex palindrome combinations, of course the lower the chance to find such a combination, the more bitcoins collectors are willing to spend to acquire that satoshi.

**Sats traded in historic transactions** are collected as well, for example **Pizza sats** are the satoshis that Laszlo Hanyecz used in the first physical purchase made with bitcoin in history: two Papa Jhon's pizzas on May 22 2010 paid 10,000 Bitcoins (worth around \$40 at the time).

**Hitman sats** are the satoshis allegedly involved in the transaction made by Ross Ulbricht (the founder of the dark-web marketplace "Silk Road") to hire a hitman.

**Sats owned by "X"** for example the **Trump sats** These are the first satoshis publicly owned by a US President (Donald Trump) on Bitcoin L1, received via a Casascius coin at the 2024 Bitcoin Conference.

Of course sats owned by Satoshi Nakamoto 1.1.1 are collected as well.



**Figure 2.2:** Laszlo Hanyecz and his family, moments after first physical purchase made with bitcoin was successfully completed



**Figure 2.3:** Casascius Bitcoin that was handed to Trump at the Bitcoin 2024 conference, the pubkey was accidentally doxxed by the New York Post

## 2.2 Inscriptions and Digital Artifacts

Inscribing data on the Bitcoin blockchain is a technical process that leverages the taproot technology 1.3.4.

Inscriptions allow arbitrary data to be attached to satoshis, creating unique Bitcoin-native digital artifacts. These are held in Bitcoin wallets and transferred using standard transactions, offering durability, immutability, security, and decentralization akin to Bitcoin itself.

Note that inscriptions can't fetch any data from outside of the blockchain by design! The industry standard on other chains is to upload the desired `text/photo/html/...` file onto a centralized or decentralized server, and inscribe on the blockchain of choice a pointer to such file. The pointer is immutable but the other server could stop serving the content at any moment!

The real strength of ordinals is that there is no server that can crash, no firewall can be broken; the data is downloaded in every node that runs the Bitcoin Core software (*21978 nodes reachable [9] Updated: Feb 28 2025*). **Breaking a bitcoin inscription is as difficult as breaking Bitcoin itself**

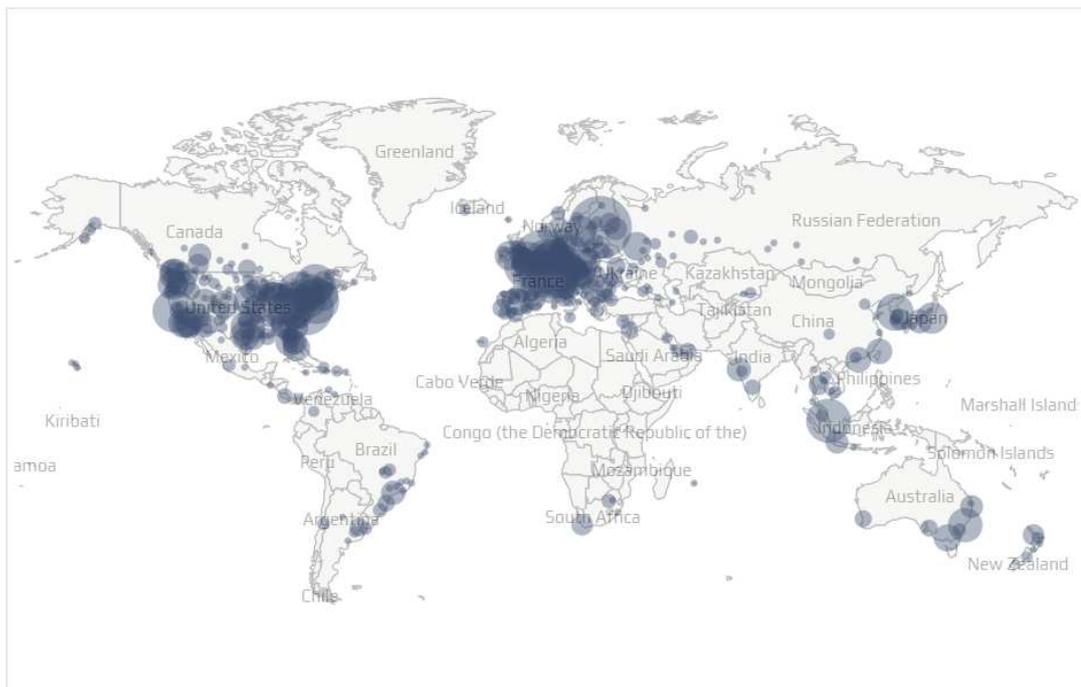


Figure 2.4: Reachable Bitcoin Nodes on Feb 28 2025 (21978 nodes)

### 2.2.1 How Data Is Inscribed Onto Satoshis

Data inscription uses taproot outputs, a Bitcoin feature since 2021, to store information in script paths. The data is serialized using opcodes:

1. Start with `OP_FALSE OP_IF ... OP_ENDIF` to frame the inscription.
2. Include `OP_PUSH "ord"` to identify it as an inscription.
3. Use `OP_PUSH 1` for the content type, it follows HTTP response format, consisting of a MIME type and byte string content, stored economically with the witness 1.1 discount (e.g.: `text/plain;charset=utf-8`), followed by `OP_PUSH 0` and the content itself, limited to 520 bytes per push for larger data.

For example, inscribing "Hello, world!" looks like:

```
OP_FALSE
OP_IF
  OP_PUSH "ord"
  OP_PUSH 1
5  OP_PUSH "text/plain;charset=utf-8"
  OP_PUSH 0
  OP_PUSH "Hello, world!"
OP_ENDIF
```

The process involves a Two-Phase Procedure:

1. Commit transaction creates a taproot output with inscription content.
2. Reveal transaction spends this output, revealing content on-chain, inscribed on the first sat of the input without a pointer field.

Inscriptions are permanently linked to the first satoshi of the output in the reveal transaction. To inscribe on a specific satoshi, ensure it's the first in the output, managed by tools like the "ord" software, which uses satpoints or sat indices for precision.

Even though an ordinal could be transferred in a 1 sat UTXO, in practice there is always some other satoshis acting as padding due to constraints like dust limits and fees. Most of the inscriptions are now in bundles of 546 or 333 sats, custom amounts are also possible. Analysis of early inscriptions reveals that they have a larger padding, on the order of tens of thousands of satoshis, and this was because the padding satoshis were meant to be spent for fees to move the inscription between addresses; that standard was soon abandoned as bitcoin price rose and large padding became expensive and impractical.

## 2.3 Tools and Resources

The ordinal protocol is built on bitcoin, extending it's open source and permission-less design.

There are still two ways you can explore and interact with the network, based on your needs:

**For developers:** Contrary to common assumptions, downloading the whole bitcoin blockchain is not enough to fully support ordinals interactions, a key piece is missing. While Bitcoin has permitted data embedding through the OP\_RETURN opcode since its very first version, and people have started experimenting such things from 2010, the "vanilla" Bitcoin core client doesn't know how to read ordinals metadata, even if it has direct access to it. This necessitates an external indexer: an off-chain tool that interprets serialized blockchain data.

The predominant indexer (the one that most people in the community use) is written in Rust for its performance and memory safety

The setup to index a node is rather simple and it requires:

1. Running a Bitcoin Core client to include new blocks. [bitcoin.org/en/download](https://bitcoin.org/en/download)
2. Executing an open source indexer [3], until it aligns with the node's block height [github.com/ordinals/ord](https://github.com/ordinals/ord)

This configuration enables a fully indexed node that you can fetch information about every satoshi with, without relying on any third party!

**For collectors:** non-technical users can leverage web-based interfaces, you can explore the ecosystem from your browser.

Websites like [ord.io](https://ord.io) and [ordiscan.com](https://ordiscan.com) let you explore the whole blockchain through their servers.

Other tools like [ordinalsbot.com](https://ordinalsbot.com) let's you inscribe files on chain without an indexer. You can buy assets directly from other people through marketplaces that support ordinals, the marketplace with the highest on-chain volume has been [magiceden.io](https://magiceden.io).

Transactions happening on these kind of platforms are partially trustless even though multiple participants are involved in the same transaction.

Why partially trustless? Because both the seller and the buyer are sent a verifiable PSBT (*partially signed bitcoin transaction*) from the marketplace, they sign it but it's ultimately the marketplace that handles the creation of the transaction and has to broadcast it to the network, introducing a dependency on its reliability.

## 2.4 Controversies and Debates

The introduction of Bitcoin Ordinals has sparked significant debate within the Bitcoin community, reflecting differing views on their role:

### 2.4.1 How do ordinals make the Bitcoin network worse?

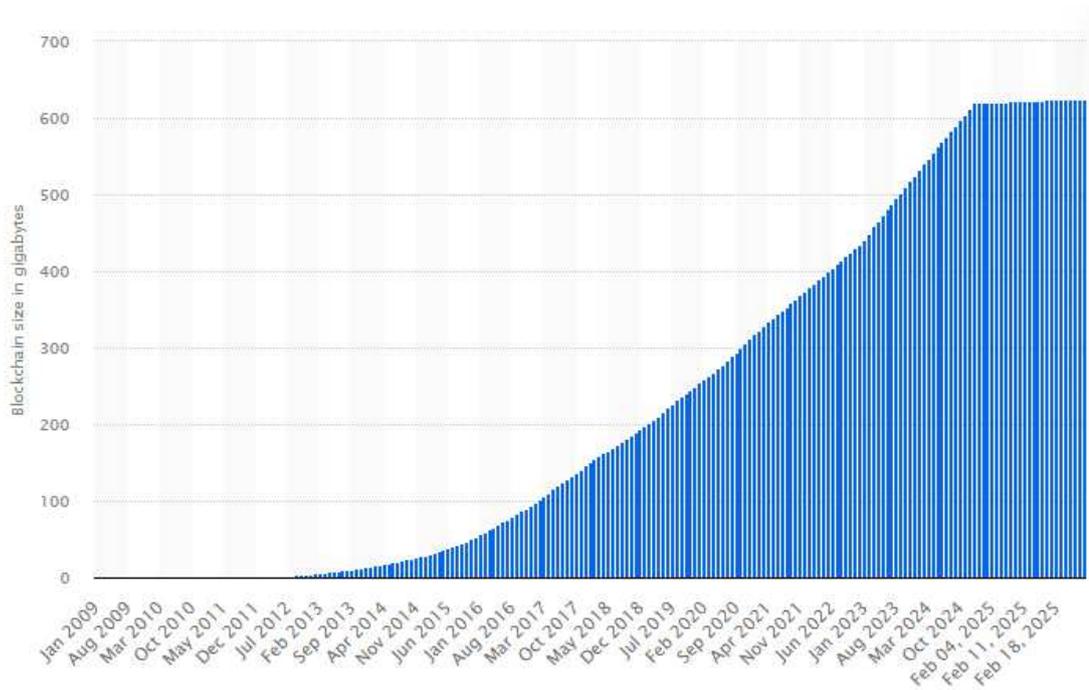
While ordinal numbers are just a computation happening completely off-chain (in the indexer [3]) they cannot harm the network, but their associated inscriptions append data to the blockchain, raising legitimate concerns. A more relevant question could be: **How do Bitcoin Inscriptions harm the network?**

Bitcoin Inscriptions have a couple of effects on the blockchain:

**Blockchain size implications** Inscriptions append data to the blockchain, the current size of the blockchain is just over 600 gigabytes. Since the first ordinal (See the first inscription on [ord.io/0](https://ord.io/0) was inscribed on December 14 2022, approximately 88 million inscription transactions have been broadcasted, adding 25Gb of data to the blockchain and earning the miners a total of 4373 Bitcoins. While the size gain over time trend remains consistent (see figure 2.5) the fact that this much data has been added raises concerns for the future as it will now be in the blockchain for ever.

Bitcoin inscriptions have added of data to the blockchain in a little more than 2 years trough 88 million Inscriptions; a total of 4373 Bitcoins were paid to the miners in the process.

The long term effects on the scalability of this protocol remain uncertain and beyond immediate control, keeping the blockchain size small is a key aspect for the network resilience, as it allows machines with less powerful hardware do download a full node and contribute to the network safety.



**Figure 2.5:** Size of the Bitcoin blockchain from January 2009 to February 2025 (gigabytes)

Data from: <https://www.statista.com/statistics/647523/worldwide-bitcoin-blockchain-size/>

**Network Congestion Effects** Critics contend Ordinals misuse the network, leading to higher transaction fees and congestion, potentially undermining Bitcoin’s primary function as a peer-to-peer cash system. They worry about using scarce resources for non-financial purposes, potentially undermining Bitcoin’s core value proposition and deviation from Satoshi Nakamoto’s vision. For instance, since February 2023, inscriptions have occupied significant block space, with some reports noting up to 50% utilization.

The debate has historical parallels, reminiscent of the 2016 block size wars, which led to forks like Bitcoin Cash. Some, like Luke Dashjr (*community-paid Bitcoin core developer*), argue Ordinals exploit vulnerabilities, with proposed changes in Bitcoin Core v27 aiming to limit new inscriptions, potentially affecting future adoption. This has sparked discussions about censorship, with some seeing Ordinals as a stress test for Bitcoin’s decentralized nature, while others fear it could lead to a “hash war” over intellectual property storage.

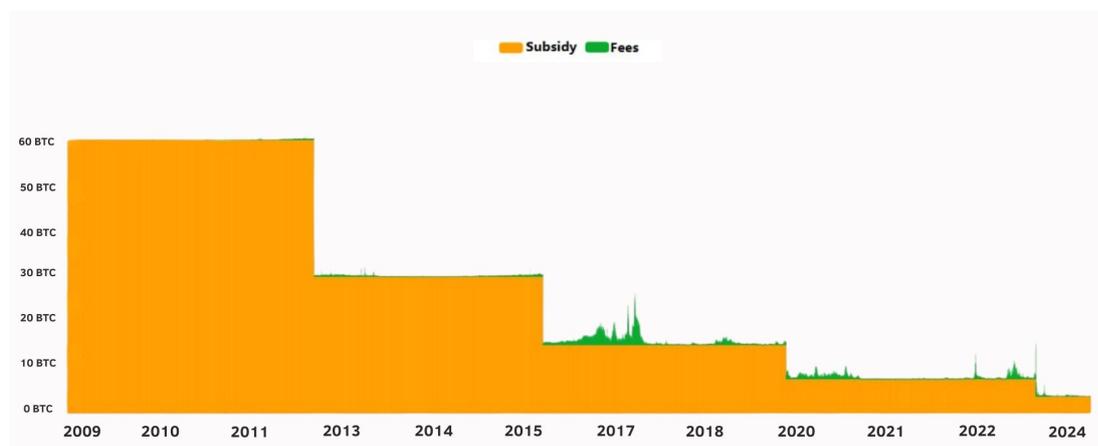
*Luke Dashjr original tweet:*

<https://x.com/LukeDashjr/status/1732204937466032285>

## 2.4.2 How do ordinals add value to the bitcoin network?

Advocates argue that the Ordinal theory, can enhance Bitcoin ecosystem in key areas

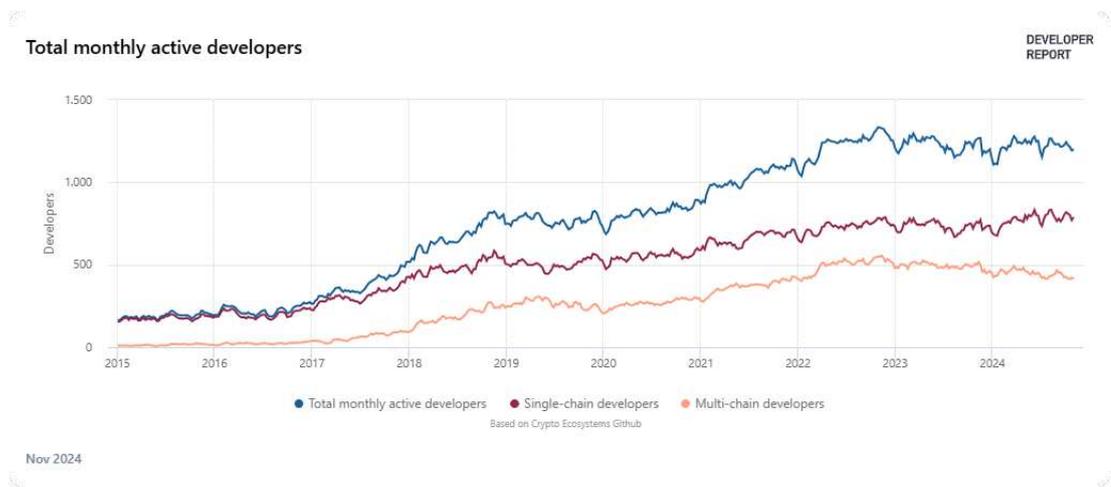
**Miners Revenue:** Ordinals increase demand for block space, ensure miners revenue through transaction fees, crucial as block rewards halve over time. Even though this "donation" is not needed, right now, for miners to be profitable, block rewards are going to diminish significantly in the future, and they will eventually disappear ( 2124), miners will therefore be completely dependent on the transaction fees (See Block Fees Vs Subsidy 2009-2024 Figure 2.6); testing new applications like Ordinals may reveal opportunities for Bitcoin's evolution with new meta-protocols (BRC-20 for example have been the first iteration of a fungible token, ordinal based meta-protocol), even if it's not the best solution now, it will get polished as time passes.



**Figure 2.6:** Block Fees Vs Subsidy (2009-2024)

**Ecosystem Expansion** Ordinals (and other metaprotocols) attract new developers and users, particularly from the NFT and DeFi spaces, revitalizing Bitcoin's developer community.

The number of active newcomer developers peaked around the end of 2022, suggesting interest in the Ordinals protocol and the number established developers kept a slow growth rate.



**Figure 2.7:** Total monthly active developers (Bitcoin) 2015-2024

*Data available:*

<https://www.developerreport.com/ecosystems/bitcoin>

**Censorship Resistance** Ordinals leverage Bitcoin’s robust, decentralized blockchain, making them more censorship-resistant than Ethereum-based NFTs. This perspective sees Ordinals as a natural evolution, expanding Bitcoin’s utility beyond peer-to-peer payments.

The real strength of inscriptions lies in their permanence: they remain accessible as long as at least one node worldwide runs Bitcoin Core software. The Bitcoin blockchain can potentially turn into a digital time capsule for not only technical knowledge, but also human history; it could become the new digital Library of Alexandria, immune to physical destruction.

*Link to a digital copy of The Bible, for ever available on-chain*

<https://www.ord.io/86474782>

## 2.5 Future Outlook

The future for ordinals is bright for collectors and enthusiasts, with developers working to reduce the friction that comes from the UTXO model and new infrastructure being developed daily.

While Ordinals offer unique opportunities as digital assets, most of them are inherently speculative, residing on Bitcoin, a cryptocurrency already known for its

volatility and high-risk profile. This dual layer of risk underscores the need for careful consideration before spending bitcoins to buy inscriptions.

The block time (average time for a block to be confirmed) on Bitcoin is slow ( 10 minutes) compared to Solana ( 395 milliseconds) and Ethereum ( 12 seconds), the mempool and the UTXO model are not simple concept to understand; the user experience is definitely not as simple as on other chains. This disparity highlights Bitcoin’s limitations for applications demanding rapid transaction finality, and incentivize holding ordinals as collectibles rather than a speculative asset, bought and flipped for a profit.

On the other hand there are projects that are working to provide valuable upgrades to the whole community, for example Xverse wallet is an independently audited and open source Bitcoin wallet that supports ordinals, runes and BRC-20 tokens, their team kept delivering useful features like indexing services, automated UTXO selection and fee estimation, MagicEden marketplace enhanced ordinals trading with full mempool protection [10], an innovating mechanism leveraging Tapscript, Schnorr signatures, and Partially Signed Bitcoin Transactions (PS-BTs). (See figure 2.8)



**Figure 2.8:** Magic Eden’s Total Mempool Protection

The rising popularity of Ordinals is evident in market data. The cumulative number of inscriptions has grown steadily since their introduction, as shown in Figure 2.9 below, showing interest from users. Similarly, Figure 2.10 tracks the total fees paid for Ordinals transactions, underscoring their growing economic footprint on the Bitcoin network and the demand for block-space.

These trends are overall positive because adoption persists despite the technical difficulties, proving that the market is willing to pay a premium in user experience to own assets on the world’s most secure blockchain.

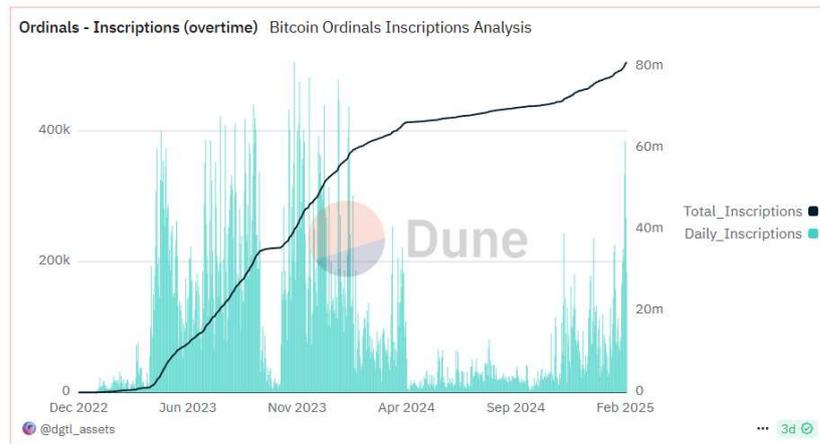


Figure 2.9: Ordinals - Incriptions (overtime)

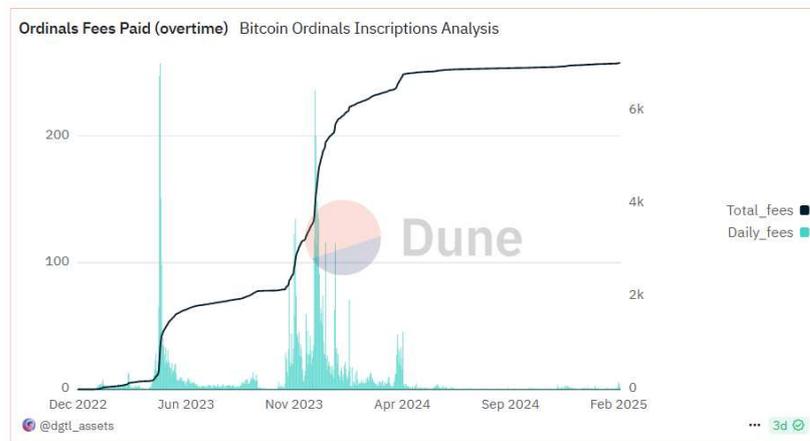


Figure 2.10: Ordinals Fees Paid (overtime)

*Data from:*  
[https://dune.com/dgtl\\_assets/bitcoin-ordinals-analysis](https://dune.com/dgtl_assets/bitcoin-ordinals-analysis)



# Chapter 3

## The Runes Protocol

Trading fungible tokens has become the main scope of modern DLTs, with standards like Ethereum's ERC-20 and Solana's SPL tokens enabling decentralized finance (DeFi) and tokenized economies.

Bitcoin, constrained by its UTXO model and lack of native smart contract support, lagged in this area until the introduction of token standards like BRC-20 and, more recently, Runes in April 2024.

The Runes protocol was introduced by Casey Rodarmor, the same developer that proposed the Ordinals numbering scheme. They leverage Bitcoin's `OP_RETURN` opcode to create efficient, on-chain fungible tokens.

The scope of this chapter is to examine the Runes protocol, its advantages over predecessors, and its potential to integrate Bitcoin into the broader tokenized ecosystem while staying true to its foundational principles.

### 3.1 Token standards on bitcoin

Runes are the last iteration of an evolution process (See figure 3.1) ongoing from 2012. There have been three attempts of creating a fungible token standard before the Ordinal protocol, and three after.

The first fungible token created was Colored Coins (2012), it was overcomplicated and not easily accessible. Mastercoin (2013) and Counterparty (2014) were the first standards to use `OP_RETURN` to embed tokens data on chain; they never gained any real traction because of the lack of interest at the time and technical



**Figure 3.1:** Visual of Layer 1 Bitcoin fungible tokens protocols

limitations.

The Ordinals protocol (Chapter 2) was introduced in 2022, and it was used to create the Atomicals (2022), which was introduced as a standard for fungible and non fungible tokens and BRC-20 (2023), which was a popular standard for fungible tokens inspired by Ethereum’s ERC-20. BRC-20 uses text Inscriptions to inscribe token data on-chain, it gained traction and was considered a good token standard on bitcoin, but problems arose quickly: the most important is definitely the ”junk” UTXOs proliferation, you would often be left with a lot of unspendable UTXOs because the amount of such UTXOs was so low that they were not worth moving as transactions fees to do so were higher than the UTXO amount itself.

Fast forward to April 2024, when the Runes protocol went live.

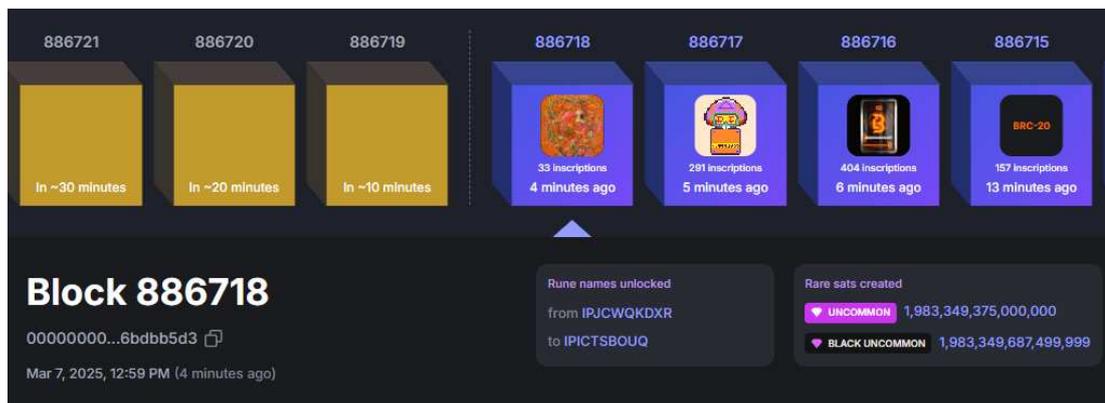
Runes were proposed as a solution to the problems of BRC-20, aiming to be more efficient and user friendly than the former;

Runes use the OP\_RETURN opcode to save data directly on the blockchain, as opposed to BRC-20 that uses text ordinals to save information, with useless headers that bloat the network. Runes are structured in such a way that no excess output are produced (there are no ”junk” UTXOs). Since runes transactions are pretty simple, they are compatible with most of the already developed infrastructure, wallets and nodes, without the need for new infrastructure to be built.

## 3.2 How do runes work

Runes are identified by IDs, which consist of the block in which a rune was etched and the index of the etching transaction within that block, represented in text as `BLOCK:TX`. For example, the ID of the rune etched in the 20th transaction of the 500th block is `500:20`; but an easier way to look for them through explorers is by their name. Rune names are unlocked in a phased way based on length and block height to prevent short-name squatting and to let a fair competition develop for desirable names. New names are unlocked block by block, for example names from `IPJCWQKDXR` to `IPICTSBOUQ` were unlocked in block 886718 (See Figure 3.2), this is a good way to create an "auction" every time more people want to etch (We will dive into rune-specific terms later) a rune with a specific name.

The process of creating a new Rune, termed etching, occurs via a transaction broadcast to the network; Etching creates a rune, sets its properties and once set, these properties are immutable, even to its etcher. When more etching transactions for the same rune name are broadcasted to the network, the transaction with the highest fee rate (measured in sats/vByte) succeeds, all the other are still mined but have no effects (no new runes are created).



**Figure 3.2:** View of runes names unlocked in block 886718 - ordiscan

<https://ordiscan.com/block/886718>

The person who manages to win the "auction" gets to choose all the parameters of the Rune in the Etching process. The most important parameters are the number of batches/mints, the amount of runes for mint/batch (which multiplied by each other give back the full supply), and the amount of "premined" runes (runes that the etcher keeps for himself).

After 6 block confirmations from the etching transaction, an other transaction called the "confirm" transaction is revealed and the minting process goes live: everyone can mint one or more batches of tokens by paying bitcoins, the bitcoins are not paid to the etcher of the rune, but they are paid in transaction fees to the miners. Runes are mintable until the hard cap for mints is reached, all mint transactions done later will be mined but won't have any effect (bitcoins will be burned).

### Runestones

Rune protocol messages, called "runestones", are stored in Bitcoin transaction outputs. A runestone output's script pubkey begins with an `OP_RETURN`, followed by `OP_13`, followed by optional data pushes. These data pushes are concatenated and decoded into a sequence of 128-bit integers, and finally parsed into a runestone. A transaction may have at most one runestone and it may etch a new rune, mint an existing one, or transfer runes from a transaction's inputs to its outputs.

### Cenotaphs

Cenotaphs are an upgrade mechanism, allowing runestones to be given new semantics that change how runes are created and transferred, while not misleading unupgraded clients as to the location of those runes, as unupgraded clients will see those runes as having been burned. Runestones may be malformed for a number of reasons, including non-pushdata opcodes in the runestone `OP_RETURN`, invalid varints, or unrecognized runestone fields. Malformed runestones are termed cenotaphs. Runes input to a transaction with a cenotaph are burned and runes etched in a transaction with a cenotaph are set as unmintable. Mints in a transaction with a cenotaph count towards the mint cap, but the minted runes are burned.

## 3.3 Debates on Runes

Even if better than the previous standard, critics still pointed out some valid questions about the runes protocol.

A primary limitation of Runes at the moment is their lack of liquidity, and this is a consequence of Bitcoin's UTXO model. This design implies that if no buyer emerges for a specific Rune, the holder remains unable to offload it, effectively rendering it illiquid. This mechanism is not new and usually applicable to the context of non fungible tokens.

Given the current state Bitcoin's technologies advancements it is impossible to have a decentralized liquidity pool (as we have on other DLT like ethereum or solana); marketplaces have come up with bitcoin native solutions. The process to buy and sell in a trustless swap looks something like this: Alice, who wants to sell Runes, selects a UTXO containing the desired quantity and signs a Partially Signed Bitcoin Transaction (PSBT). This PSBT commits her UTXO in exchange for a specified amount of bitcoin. Bob, on the other end, is trying to purchase Runes, browses the marketplace's listed UTXOs, selects Alice's, signs the corresponding PSBT, and broadcasts it to the network, completing the exchange.

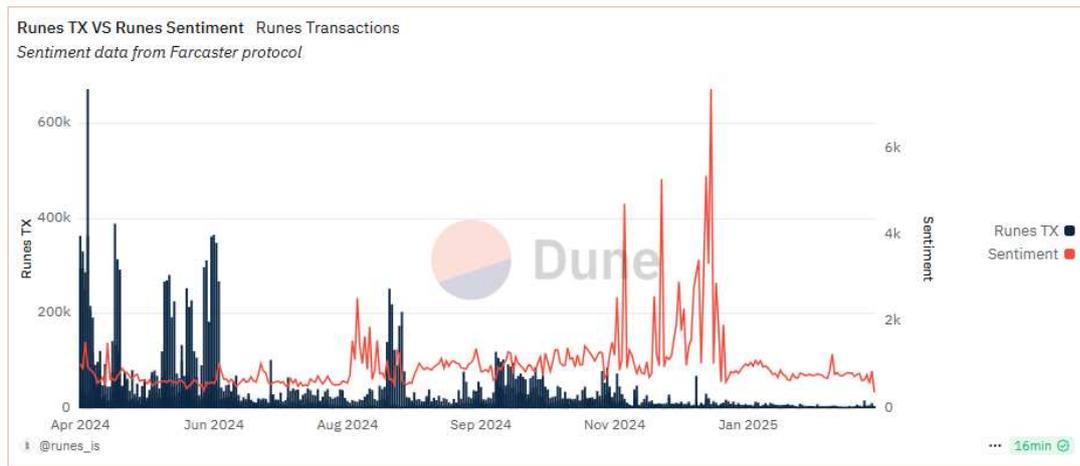
Marketplaces like `dotswap.app` are offering solutions with centralized liquidity pools, allowing users to instantly buy and sell Runes. While this is a good upgrade, it's not ideal for bitcoiners, as there is a trust component to a third party. Runes bloat the network in a different way, because they don't add a lot of data to the blockchain, but they impact the experience of everyday users by having sudden spikes in interest, and making users pay more for their normal transactions.

On the other hand runes have been an interesting experiment: as of March 2025, a total of 205,000 Runes have been etched, and about 78 bitcoins ( \$5 million) was paid to the miners in the process, underlining a discrete interest from the community. tokens such as `DOG•GO•TO•THE•MOON` and `BILLION•DOLLAR•CAT`, were completely community driven, meaning that there was no team allocation or an entity that had more power between holders such tokens reached almost one billion \$ of market capitalization underlining a strong narrative behind them.

*Source [https://dune.com/runes\\_is/runes](https://dune.com/runes_is/runes)*

## 3.4 Future Outlook

While Runes may not be essential for the Bitcoin network right now, I believe that the development of technologies like fungible tokens will become increasingly



**Figure 3.3:** Runes Transactions VS Runes Sentiment (Sentiment data from Farcaster protocol)

important over time: a good token standard will probably be achieved over many iterations of trial and error.

Transactions involving runestones, showed in figure 3.3, are clearly in a down-trend, and this might prove that we are not technically ready for implementing this kind of token standard on Bitcoin quite yet.

However the future of runes is definitely something to pay attention to, especially with potential protocol upgrades such as the re-enabling of `OP_CAT` in Bitcoin’s scripting language. `OP_CAT` was enabled when Bitcoin founder Satoshi Nakamoto was still an active participant in the ecosystem but was disabled due to a lack of use cases. It was revisited in recent years to enable the types of applications being built on Ethereum and Solana directly on Bitcoin.

Some tangible examples affecting Runes could be:

**Efficient Data Handling in Runestones** Currently, Runestones use `OP_RETURN` for static data, limiting dynamic operations. Being able to concatenate multiple data elements (e.g., Rune ID and mint number) into a single script could enable more efficient verification and parsing, reducing friction in the user experience allowing for a lower on-chain footprint as well.

**Enabling Complex Token Operations** `OP_CAT` would likely allow more complex operations, like conditional transfers or minting rules; facilitating complex smart contracts, new security features like multi-signature setups, and integration with decentralized finance (DeFi) platforms.

But the general applications could impact not only the Runes ecosystem, but all

of bitcoin protocols.

Even if the OP\_CAT protocol upgrade is not certain, there are some indications that developers are actively working on it, as Reported by The Block: Bitcoin Ordinals project Taproot Wizards is raised \$30 million in a round led by Standard Crypto.

*Source* [https://www.theblock.co/post/338805/taproot-wizards-raises-30-million-to-expand-op\\_cat-functionality-on-bitcoin](https://www.theblock.co/post/338805/taproot-wizards-raises-30-million-to-expand-op_cat-functionality-on-bitcoin)



# Conclusion

This thesis has explored how Bitcoin's ecosystem has evolved over time, each upgrade has been proposed by individuals and refined by developers. Miners are then incentivized to vote on small and safe upgrades, keeping the security of the network in first place rather acting to secure quick profits. We have seen this in the address evolution, from older address types like legacy to newer ones like SegWit, which fixed issues like unconfirmed transaction malleability, and then to taproot, which enabled more complex scripting solutions. These updates showed how the Script language was adapted to meet new needs while keeping the network secure and flexible.

Looking at the wider scene of DLTs, it's clear that the adoption of both fungible and non fungible digital it's spreading, and these technology will eventually be integrated in everyday things. Although Bitcoin was not initially conceived to support such functionalities, the advent of protocols such as Ordinals and Runes has extended its capabilities in this direction. The ordinal numbering system is innovative because it allows to track how individual satoshis move across the address space over time. Data shows an increasing interest the current implementation of non fungible tokens based on the ordinals theory, called inscriptions. Nevertheless, the Runes framework, despite its operational success, exhibits notable limitations when compared to fungible token standards on alternative blockchains, and the issues arise primarily because of the slow block time of bitcoin, and because of the lack of advanced programmability in the framework. Even though the latter problem could be resolved by the re-introduction of an old operational code into the scripting language, the bitcoin blockchain still wouldn't be the best market fit for fungible tokens.

The Ordinal protocol, enhances Bitcoin by adding a layer of provenance to the blockchain, enthusiasts appreciate it because every image and sentence appended

to the blockchain will stay there indefinitely. By paying a fee for the block space used, you are able to store data, (potentially) for ever. Even though we could consider these technologies battle tested, inscriptions are almost 3 years old and runes almost 1 year old, their relative novelty precludes definitive predictions regarding sustained market demand. Nonetheless, they are innovations in the bitcoin ecosystem, and mark a significant step forward in the evolution of Bitcoin and the wider DLT landscape.

In conclusion, this thesis highlights Bitcoin's remarkable resilience and adaptability as a decentralized system. By embracing technological advancements such as SegWit, Taproot, Ordinals, and Runes, Bitcoin not only reinforces its status as a foundational pillar of blockchain technology but also demonstrates its potential to support a diverse array of applications. While challenges like maintaining scalability with expanded functionality persist, Bitcoin's ability to evolve while adhering to its principles of security and decentralization affirms its relevance in an evolving digital economy.

# Bibliography

- [1] *The Bitcoin Whitepaper* Available: <https://bitcoin.org/bitcoin.pdf>  
Written by Satoshi Nakamoto
- [2] *Ordinal Theory Handbook*. Available: <https://docs.ordinals.com/>. Written by Casey Rodarmor and Raph Japh
- [3] *Ordinal Indexer Repository*. Available: <https://github.com/ordinals/ord>.
- [4] *Technical explanations of how Bitcoin works for programmers*. Available: <https://learnmeabitcoin.com/technical/>. Written by Greg Walker
- [5] *Bitcoin's UTXO Model: What Is It and How To Manage UTXOs*. Available: <https://river.com/learn/bitcoins-utxo-model/>.
- [6] *Understanding the Differences Between Bitcoin Address Formats When Developing Your App*. Available: <https://www.hiro.so/blog/understanding-the-differences-between-bitcoin-address-formats-when-developing-your-app>
- [7] *What is Bitcoin Script? Unveiling Its Role in Bitcoin*. Available: <https://komodoplatfrom.com/en/academy/bitcoin-script/#what-is-bitcoin-script>. Written by Delton Rhodes
- [8] *Electric Capital 2024 Developer Report*. Available: <https://www.developerreport.com/ecosystems/bitcoin>.
- [9] *Estimates of the relative size of the Bitcoin peer-to-peer network by finding all of its reachable nodes*. Available: <https://bitnodes.io/>
- [10] *Introducing: Full Mempool Protection*. Available: <https://help.magiceden.io/en/articles/10075361-introducing-full-mempool-protection>