NFTs in Bitcoin?

From ordinals to inscriptions

Context

What are Non-fungible Tokens?

- A unique digital identifier that cannot be copied, substituted, or subdivided;
- Recorded in a blockchain;
- Used to certify authenticity and ownership (digital asset or rights relating);

Colored Tokens based protocols

> By carefully tracking the origin of a given bitcoin, it is possible to "color" a set of bitcoins to distinguish it from the rest.

- Represents real-world assets on the Bitcoin blockchain;
- Couldn't hold large digital assets in blockchain;

Etherium ERC-721

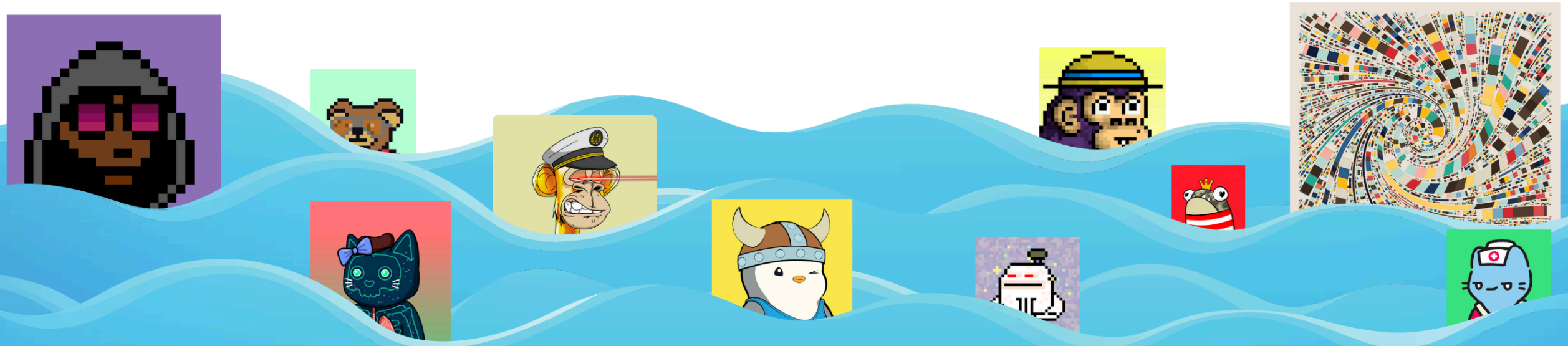
> A standard interface for non-fungible tokens, also known as deeds.

https://eips.ethereum.org/EIPS/eip-721

Every ERC-721 compliant contract must implement the `ERC721` and `ERC165` interfaces

OpenSea

NFTs feber in OpenSeas platform



Neymar desembolsa R$ 6 milhões e entra no mundo dos NFTs

O atleta adquiriu dois tokens da coleção Bored Ape Yacht Club por meio da plataforma Ethereum

Why not create a NFT ecosystem in Bitcoin?

Problems to address in Bitcoin:

- Bitcoin has no notion of stable, public accounts or identities;
- Addresses are single-use, and wallet accounts are private;
- Additionally, the use of addresses or public keys as stable identifiers precludes transfer of ownership or key rotation.

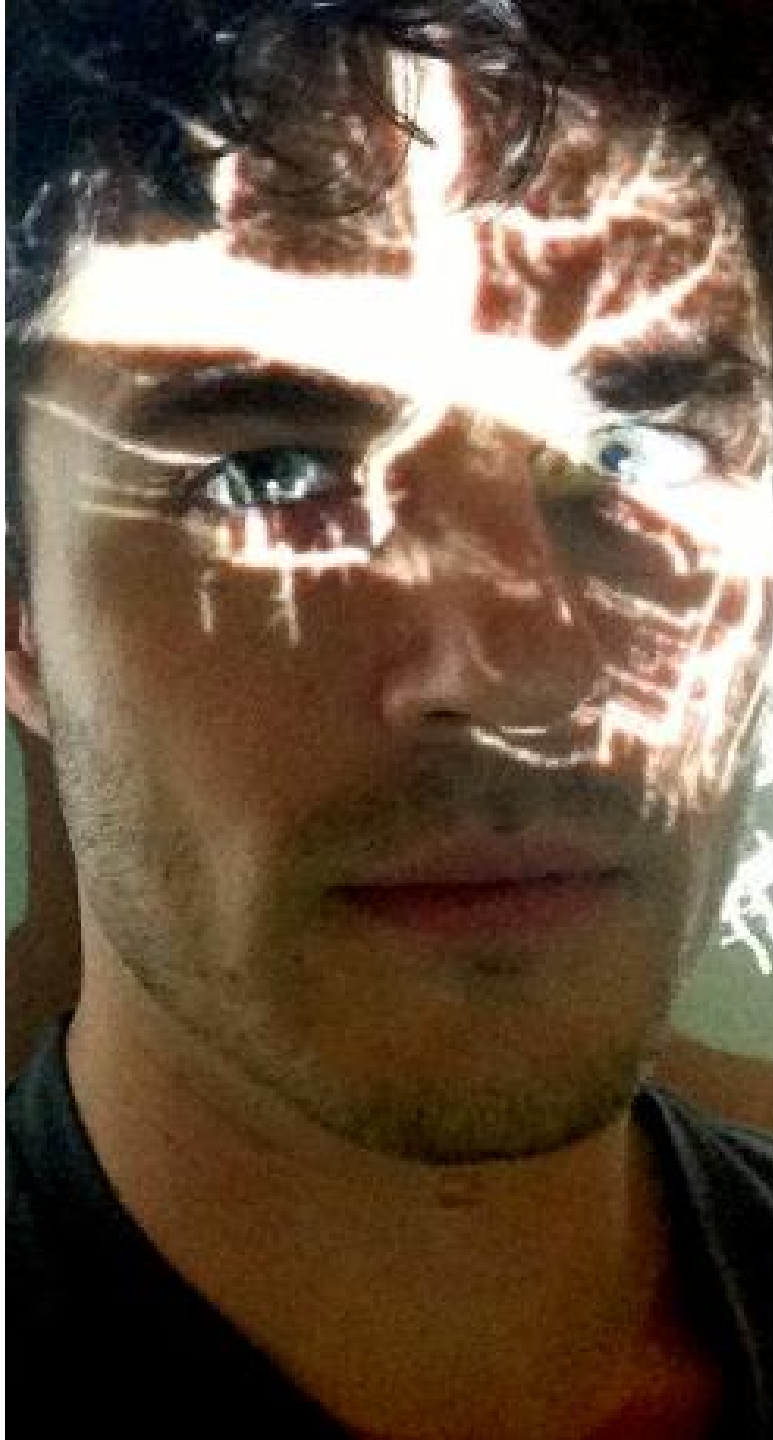
Ordinal Numbers

Ord BIP Proposal

Author

Casey Rodarmor

- Used to make some generative digital art as hobbie and wanted to allow artists to sell cool art;
- Insatisfaction with Etherium building blocks (misses multisig, everything is mutable...)



Design

- Every sat is numbered, starting at 0, in the order in which it is mined.
- The ordinal numbers of sats in inputs are transferred to output sats in first-in-first-out order.
- The coinbase transaction have an implicit first input with the subsidy value and has an input for every fee-paying transaction in the block, in order.
- Underpaying the subsidy does not change the ordinal numbers of sats mined in subsequent blocks.

Example: Coinbase Transaction

```
# INPUTS

## mining input
[5000000000..10000000000]

# OUTPUTS

## output 0
[5000000000..10000000000]
```

Example: 1 Input and 2 Outputs

```
# INPUTS

## input 0
[50..100]

# OUTPUTS

## output 0
[50..75]

## output 1
[75..100]
```

Example: Paying Fees

```
# INPUTS

## input 0
[50..100]

# OUTPUTS

## output 0
[50..75]

## output 1
[75..90]

# FEE = 10
```

Example: Receiving Fees

```
# INPUTS

## mining input
[100..200]

## fee 1
[90..100]

# OUTPUTS

## output 0
[100..200]
[90..100]
```

Satoshis Representation

Notation type	Representation
integer	`209999410699265 9`
decimal	`3891094.16797`
degree	`3°111094'214"16797"'`
percentile notation	`99.999719490602 54%`
name	`satoshi`

Integer notation

`2099994106992659`

The ordinal number, assigned according to the order in which the satoshi was mined.

Decimal notation

`3891094.16797`

The first number is the block height in which the satoshi was mined, the second the offset of the satoshi within the block.

Percentile notation

`99.99971949060254%`

The satoshi's position in Bitcoin's supply, expressed as a percentage.

Name Notation

`satoshi` .

An encoding of the ordinal number using the characters `a` through `z` .

Degree Notation

```
3°111094'214"16797"
```

```
A°B'C"D"
│  │  │  └─ Index of sat in the block
│  │  └──── Index of block in difficulty adjustment period
│  └─────── Index of block in halving epoch
└────────── Cycle, numbered starting from 0
```

Rarity

- *Blocks*: A new block is mined approximately every 10 minutes.
- *Difficulty adjustments*: Every 2016 blocks (2 weeks), the Bitcoin network adjusts the difficulty target.
- *Halvings*: Every 210,000 blocks (4 years), the amount of new sats created in every block is cut in half.
- *Cycles*: Every 6 halvings (24 years), the halving and the difficulty adjustment coincide.

Rarity

- **common**: not the first sat of its block
- **uncommon**: first sat of each block
- **rare**: first sat of each difficulty adjustment period
- **epic**: first sat of each halving epoch
- **legendary**: first sat of each cycle
- **mythic**: The first sat of the genesis block

Inscriptions

- Inscriptions inscribe arbitrary content in sats.
- Sats can then be transferred using bitcoin transactions.
- An inscription is a MIME type and the content.

Inscription content is serialized using data pushes within unexecuted conditionals, called "envelopes".

```
# Taproot Script

OP_FALSE
OP_IF
  OP_PUSH "ord" # indicate ordinals envolope
  OP_PUSH 1 # indicates that the next push is content-type
  OP_PUSH "text/plain;charset=utf-8"
  OP_PUSH 0 # indicates that subsequent pushes are content
  OP_PUSH "Hello, world!"
OP_ENDIF
```

Multiple data pushes must be used if content is larger than 520 bytes.

Inscriptions Ids

`521f8eccffa4c41a3a7728dd012ea5a4a02feed81f41159231251ecf1e5c79dai0`

The part in front of the `i` is the transaction ID (`txid`) of the reveal transaction. The number after the `i` defines the index (starting at 0) of new inscriptions being inscribed in the reveal transaction.

Inscription Numbers

- Inscriptions are assigned inscription numbers by the order reveal transactions appear in blocks and envelopes appear in those transactions.
- note: Due to a historical bug in `ord`, inscriptions which are revealed and then immediately spent to fees are numbered as if they appear last in the block.

Delegate Field (#11)

Inscriptions may nominate a delegate inscription. This can be used to cheaply create copies of an inscription.

```
OP_FALSE
OP_IF
  OP_PUSH "ord"
  OP_PUSH 11 # deletage (32bytes id) (4 byte index)
  OP_PUSH 0x1f1e1d1c1b1a191817161514131211100f0e0d0c0b0a09080706050403020100
OP_ENDIF
```

Metadata Field (#5)

Inscriptions may include CBOR metadata.

```
OP_FALSE
OP_IF
    ...
    OP_PUSH 0x05 # include metadata
    OP_PUSH '{"foo":"bar","baz":[null,true,false,0]}'
    ...
OP_ENDIF
```

Pointer Field (#2)

Causes the inscription to be made on the sat at the given position in the outputs.

```
OP_FALSE
OP_IF
  OP_PUSH "ord"
  OP_PUSH 1
  OP_PUSH "text/plain;charset=utf-8"
  OP_PUSH 2 # pointer to
  OP_PUSH 0x0001 # satoshi 256 in transaction (little endian)
  OP_PUSH 0
  OP_PUSH "Hello, world!"
OP_ENDIF
```

Provenance Field (#3)

- The owner of an inscription can create child inscriptions;
- This can be used for collections;
- Children can themselves have children, allowing for complex hierarchies.

Provenance Field Steps

1. Create an inscribe transaction T as usual for C.
2. Spend the parent P in one of the inputs of T.
3. Include tag `3`, in C, with the inscription ID of P.

```
OP_FALSE
OP_IF
  …
  OP_PUSH 3 # provonance of (32bytes id) (4 byte index)
  OP_PUSH 0x1f1e1d1c1b1a191817161514131211100f0e0d0c0b0a09080706050403020100ff
  …
OP_ENDIF
```

Tools

Ord Wallet

Ordinals Wallet

Curiosities

On October 8th, 2012, jl2012 posted a scheme to the same forum which uses decimal notation and has all the important properties of ordinals. The scheme was discussed but never implemented.

On August 21st, 2012, Charlie Lee posted a proposal to add proof-of-stake to Bitcoin to the Bitcoin Talk forum. This wasn't an asset scheme, but did use the ordinal algorithm, and was implemented but never deployed.