# Enveloppe's protocol Bitcoin based

Thomas Suau
Université Libre de Bruxelles (ULB)

Brussels
May 24th, 2024

1. Why enveloppes ?

2. Enveloppes

3. Mechanisms & Goals

# Pushing data on Bitcoin

The idea to push and retrieve data on Bitcoin is not young we call them **metaprotocols** :

- **Colored Coins** : A protocol to create and trade tokens on Bitcoin created in 2012 ;
- **Mastercoin** : Same idea as Colored Coins but with one specific currency associated MSC which made the first *Initial Coin Offering* (ICO) in 2013 ;
- **Counterparty** : A more general protocol created in January 2014 to allow complex contracts resolution on Bitcoin.

# Codes for Metaprotocols

Previously the most used `OP_CODE` to push objects on Bitcoin was `OP_RETURN`.

Introduced natively on Bitcoin in 2014, we previously used mainly `OP_DROP` or storing it in output of `P2PKH` scripts [1]

`OP_RETURN` can store 83 Bytes **maximum** !

1. For more information : *L'élegance de Bitcoin*, Ludovic Lars, Konsensus Network, p.333 (2023).

# Ordinals core components

The Ordinals protocol contains three core components (including cli) :

- a **Theory** : which is counting satoshis (smallest Bitcoin unit) and link protocol objects (inscriptions) with satoshis ;
- an **Enveloppe** : based on Bitcoin Script which can be viewed as a Script template 6 ;
- an **Indexer** with command line : fully written in Rust it's required a Bitcoin node with `txindex=1`.

# Ordinals script enveloppe

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

```
OP_IF
OP_PUSHBYTES_3 6f7264
OP_PUSHBYTES_1 01
OP_PUSHBYTES_24 746578742f706c61696e3b636861727
365743d7574662d38
OP_0
OP_PUSHBYTES_10 57652061726520757 30a
OP_ENDIF
```

**Figure** – Ordinals Enveloppe (left) Example of text file inscription (right)

# Atomicals Core components

The Atomicals protocol contains three core components :

- An **ElectrumX implementation** : a fork of ElectrumX named `atomicals-electrumx`. It creates custom electrumx RPC endpoint ;
- An **Enveloppe** : built in Bitcoin Script on the same principle of Ordinals 8 ;
- A **Command Line** : built in JavaScript to interact directly with Atomicals protocol. The CLI doesn't manage the indexer.

# Atomicals script enveloppe

```
OP_FALSE
OP_IF
 0x0461746F6D // Push "atom" 4 bytes
 <Operation>  // Followed by a single push to denote the operation type
 <Payload>    // Payload (CBOR encoded) for the operation
OP_ENDIF
```

```
OP_IF
OP_PUSHBYTES_4 61746f6d
OP_PUSHBYTES_3 6e6674
OP_PUSHBYTES_66 a16461726773a46d726571756573745
f7265616c6d6961746f6d6963616c7368626974776f726b
636435363830656e6f6e63651a004a05a36474696d651a6
506915f
OP_ENDIF
```

**Figure** – Atomicals Enveloppe (left) Example of a Realm transaction (right)

# Commit/Reveal

Both enveloppe's protocol are using a mechanism of Commit/Reveal transaction.

The first commit transaction is made as a `P2TR` script. The second (reveal) transaction is signing the locking script and adding the enveloppe as shown above.

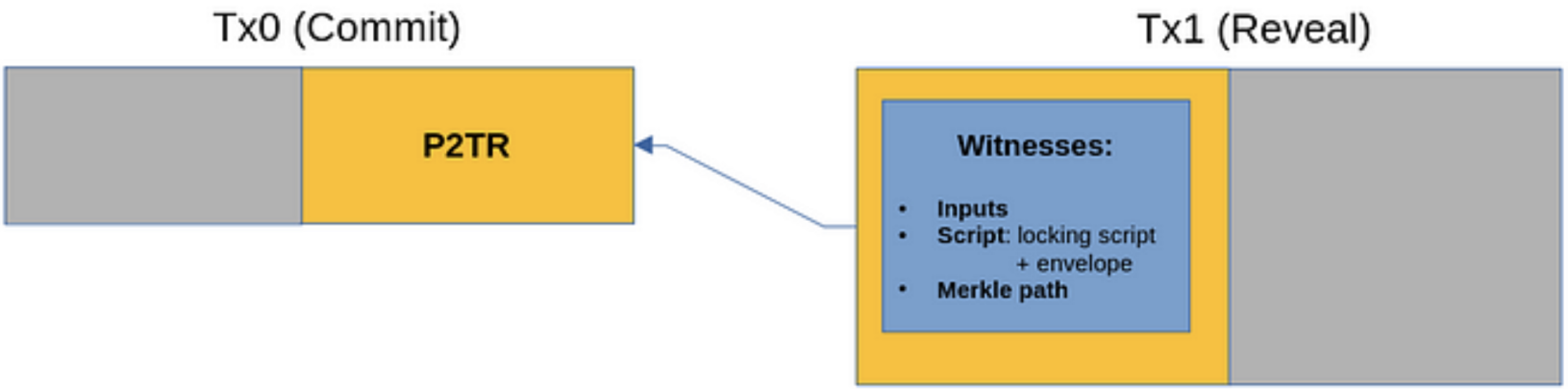


**Figure** – Commit Reveal Scheme.

*source : Incorporate smart contracts into Ordinals on BTC - CoinGeek*

# Goals

I purpose a list of current research problems about these protocols :

- **How can we deploy BLOBs and make calls of them through Enveloppe's protocol ?** A first way to achieve it could be to create specific indexer which can integrate specific Virtual Machines for BLOB management as RGB ;
- **Is it possible to build native bridges between UTXO-based blockchains ?** With ordinals protocol on Litecoin and on Dogecoin we can wonder if native bridges are possible in better way than nowadays bridges ;

# Goals

- **Create a classification and academic studies of Enveloppes based Protocols.** This will be the first academic work about this kind of Protocols based on Bitcoin. We can easily extend those research to every UTXO-based blockchains. An extension of such classification could be done for Extended-UTXO model used by Cardano.
- **Show optimality of such protocols regarding last Schnorr-Taproot update (BIP 341).** Some actual considerations tend to show that these protocols aren't optimal. The Taproot upgrade is implementing MAST (Merkelized Alternative Script Trees). Are we able to use MAST to implement and optimise Enveloppe's protocol ?