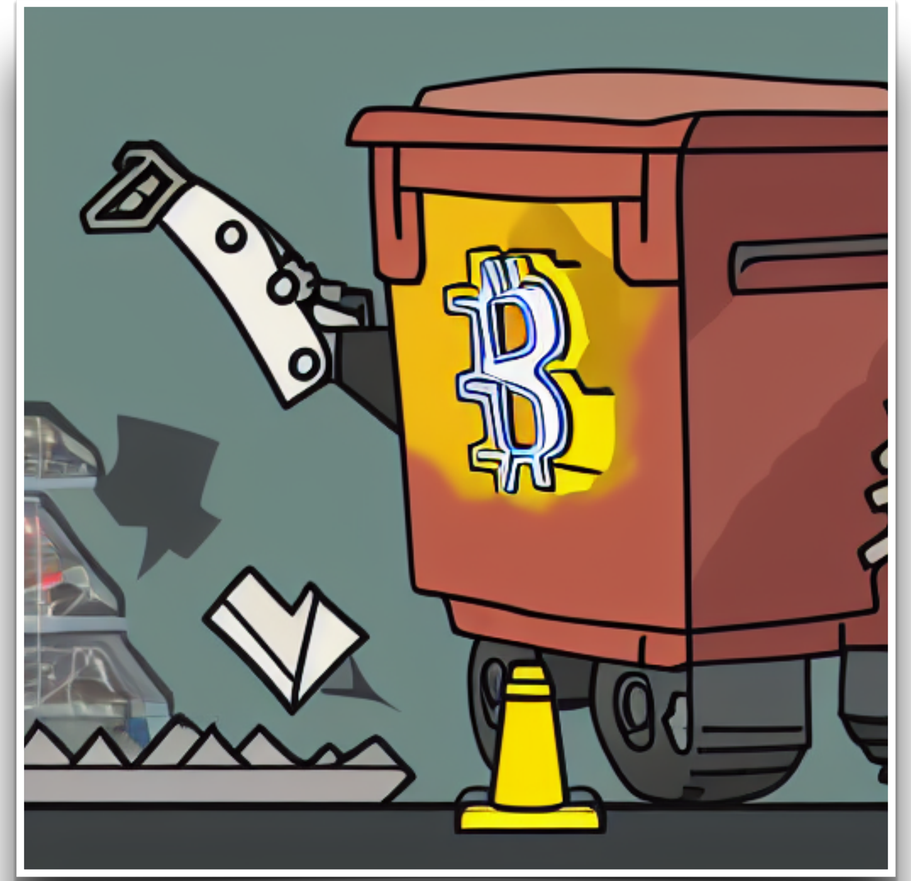
Deep dive into BIP324

Improving P2P network privacy

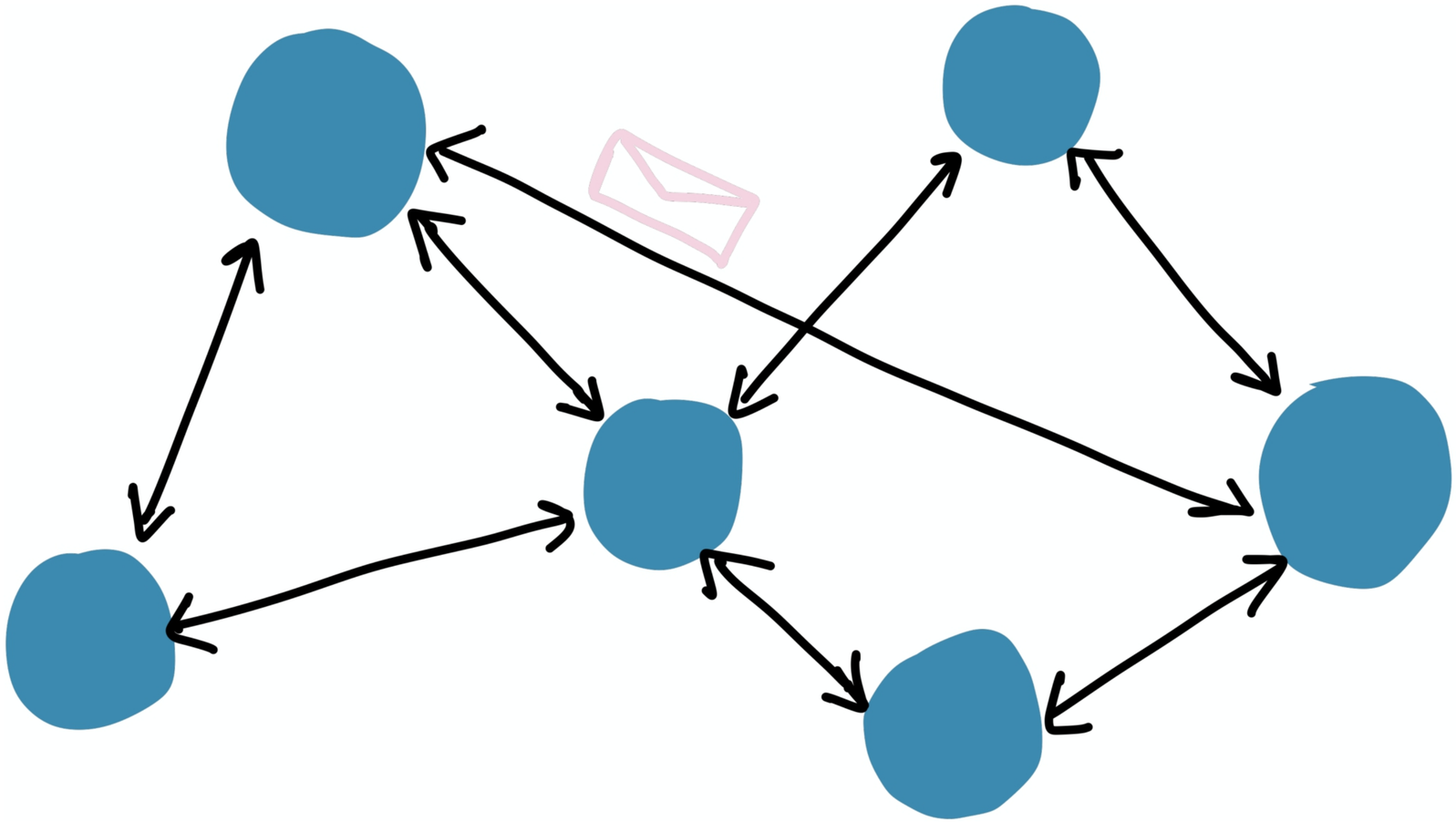


Hi! I'm stratospher

https://github.com/stratospher

- bitcoin core contributor : P2P network, libsecp
- potions at bitshala
  (a community to onboard bitcoin devs)

P2P Network

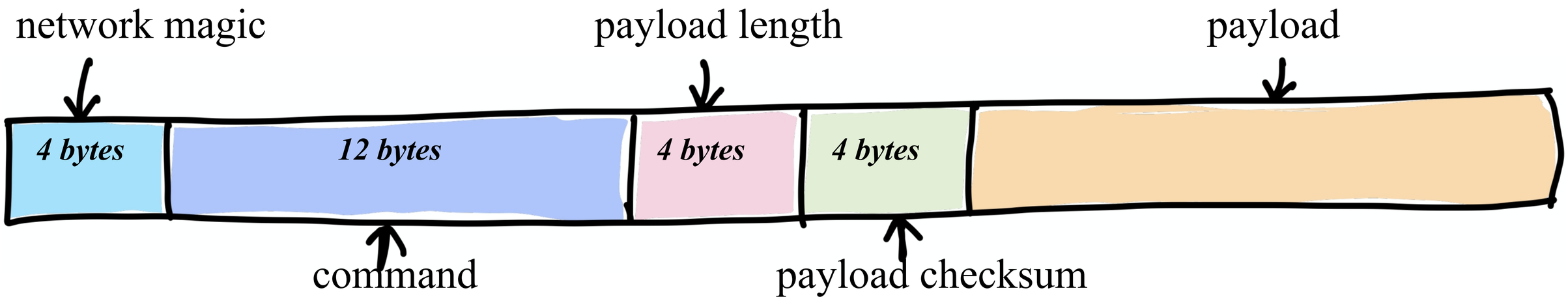


version

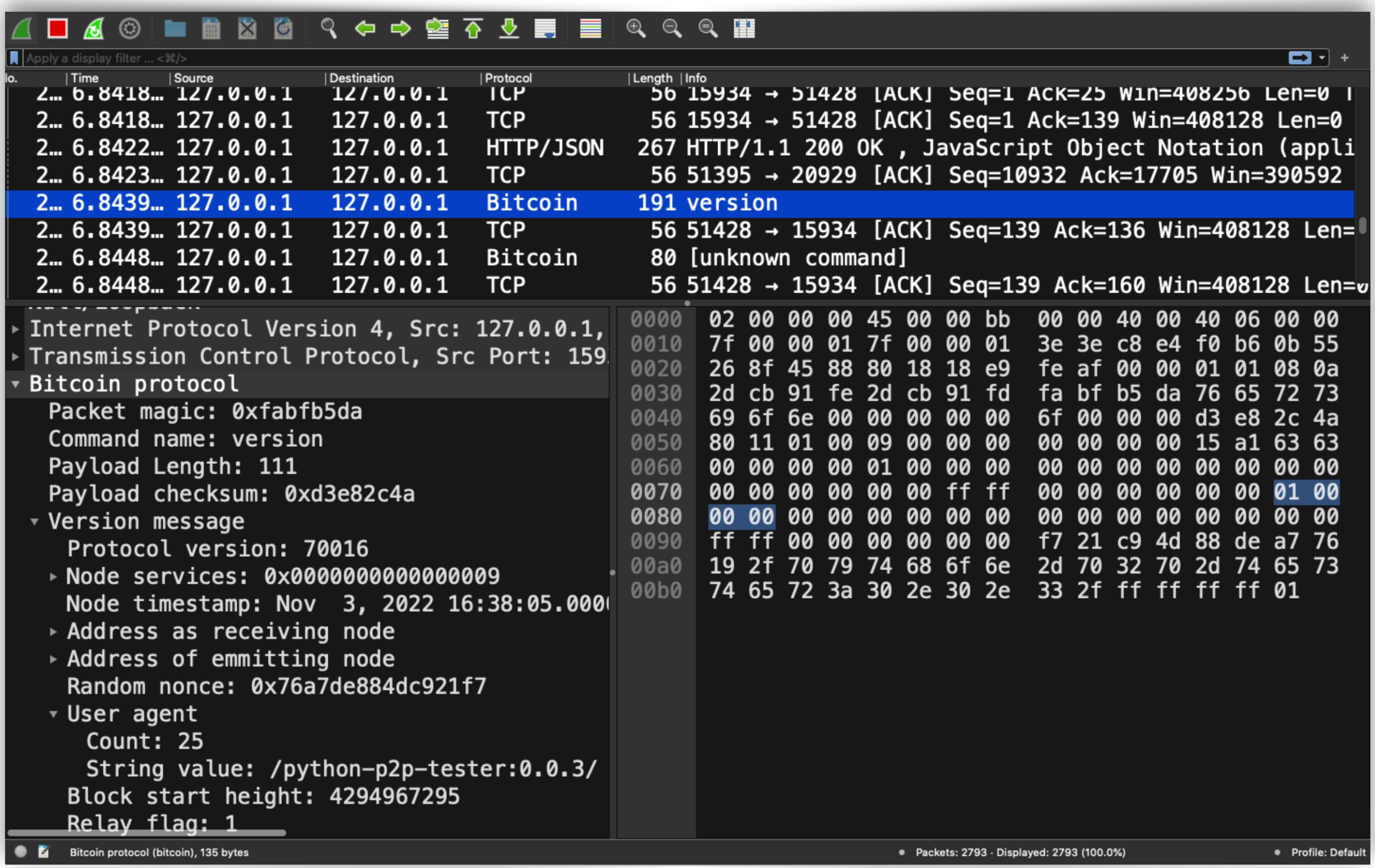
ping

getaddr

1. Self-revealing

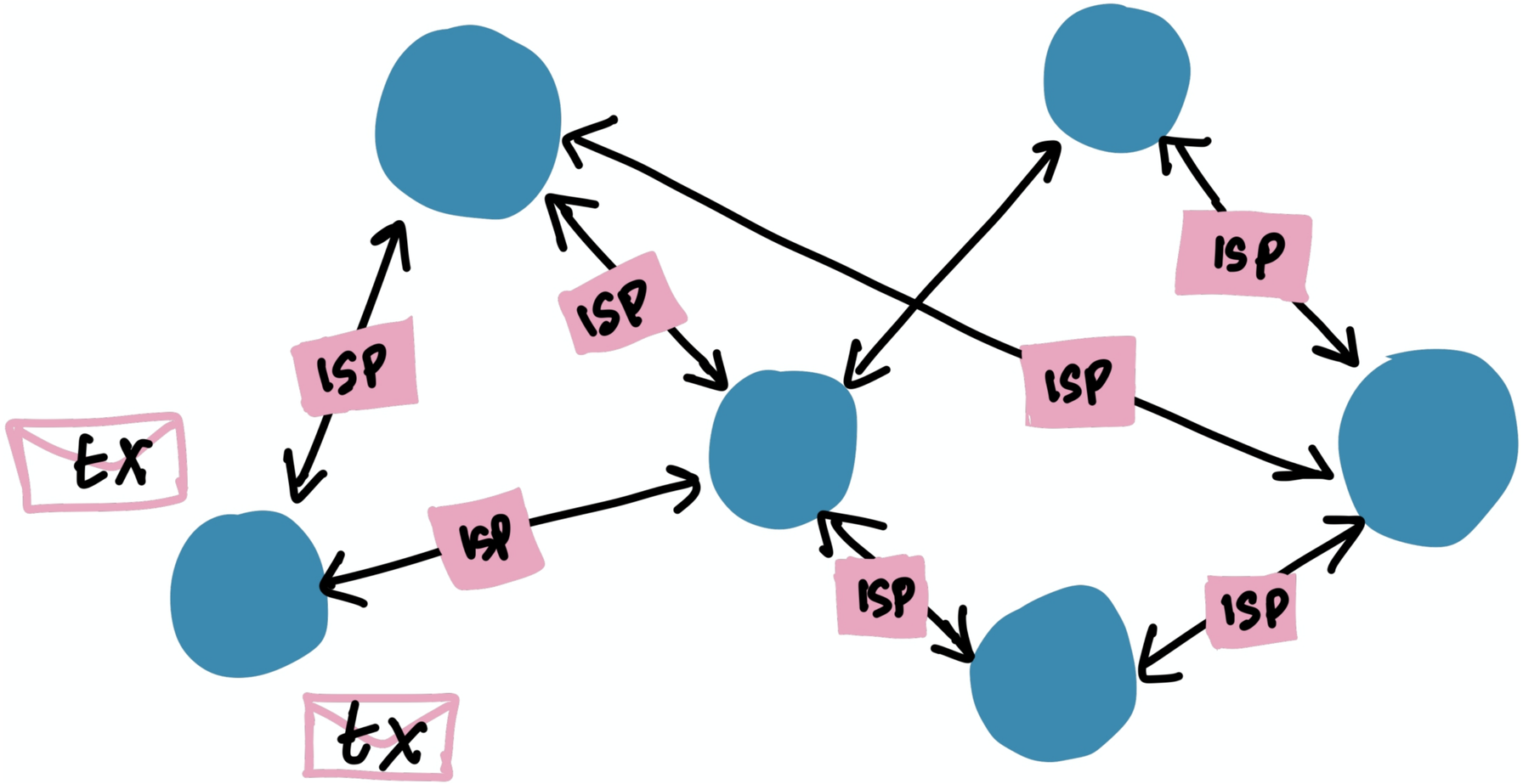


Network Message


Apply a display filter ... <⌘/>
lo. | Time | Source | Destination | Protocol | Length | Info
2… 6.8418… 127.0.0.1 127.0.0.1 TCP 56 15934 → 51428 [ACK] Seq=1 Ack=25 Win=408256 Len=0
2… 6.8418… 127.0.0.1 127.0.0.1 TCP 56 15934 → 51428 [ACK] Seq=1 Ack=139 Win=408128 Len=0
2… 6.8422… 127.0.0.1 127.0.0.1 HTTP/JSON 267 HTTP/1.1 200 OK , JavaScript Object Notation (appli
2… 6.8423… 127.0.0.1 127.0.0.1 TCP 56 51395 → 20929 [ACK] Seq=10932 Ack=17705 Win=390592
2… 6.8439… 127.0.0.1 127.0.0.1 Bitcoin 191 version
2… 6.8439… 127.0.0.1 127.0.0.1 TCP 56 51428 → 15934 [ACK] Seq=139 Ack=136 Win=408128 Len=
2… 6.8448… 127.0.0.1 127.0.0.1 Bitcoin 80 [unknown command]
2… 6.8448… 127.0.0.1 127.0.0.1 TCP 56 51428 → 15934 [ACK] Seq=139 Ack=160 Win=408128 Len=
Internet Protocol Version 4, Src: 127.0.0.1,
Transmission Control Protocol, Src Port: 159
Bitcoin protocol
Packet magic: 0xfabfb5da
Command name: version
Payload Length: 111
Payload checksum: 0xd3e82c4a
Version message
Protocol version: 70016
Node services: 0x0000000000000009
Node timestamp: Nov 3, 2022 16:38:05.000
Address as receiving node
Address of emmitting node
Random nonce: 0x76a7de884dc921f7
User agent
Count: 25
String value: /python-p2p-tester:0.0.3/
Block start height: 4294967295
Relay flag: 1
0000 02 00 00 00 45 00 00 bb 00 00 40 00 40 06 00 00
0010 7f 00 00 01 7f 00 00 01 3e 3e c8 e4 f0 b6 0b 55
0020 26 8f 45 88 80 18 18 e9 fe af 00 00 01 01 08 0a
0030 2d cb 91 fe 2d cb 91 fd fa bf b5 da 76 65 72 73
0040 69 6f 6e 00 00 00 00 00 6f 00 00 00 d3 e8 2c 4a
0050 80 11 01 00 09 00 00 00 00 00 00 00 15 a1 63 63
0060 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00
0070 00 00 00 00 00 00 ff ff 00 00 00 00 00 00 01 00
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0090 ff ff 00 00 00 00 00 00 f7 21 c9 4d 88 de a7 76
00a0 19 2f 70 79 74 68 6f 6e 2d 70 32 70 2d 74 65 73
00b0 74 65 72 3a 30 2e 30 2e 33 2f ff ff ff ff 01
Bitcoin protocol (bitcoin), 135 bytes
Packets: 2793 · Displayed: 2793 (100.0%)
Profile: Default

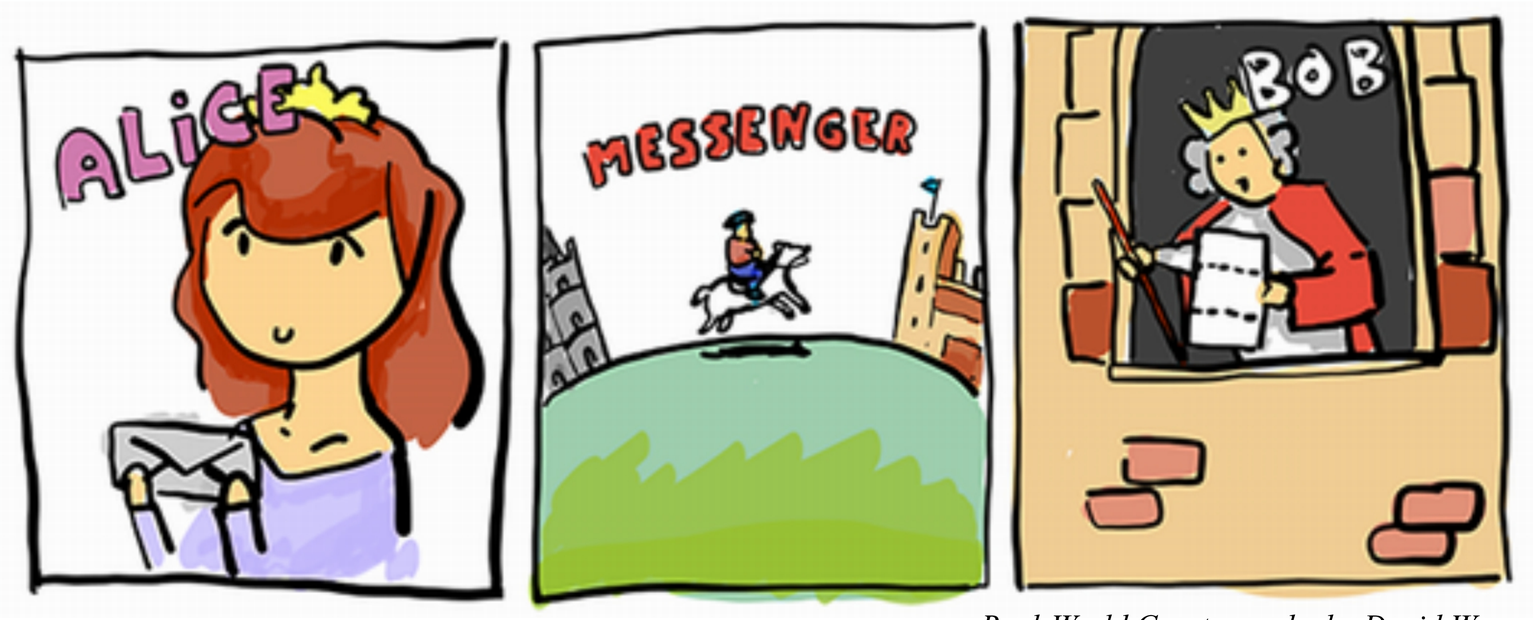
2. Unencrypted

Global observers can detect source/timing of new transactions/blocks



3. Unauthenticated

- Identity : “Does Bob know the packet is from Alice?” ❌



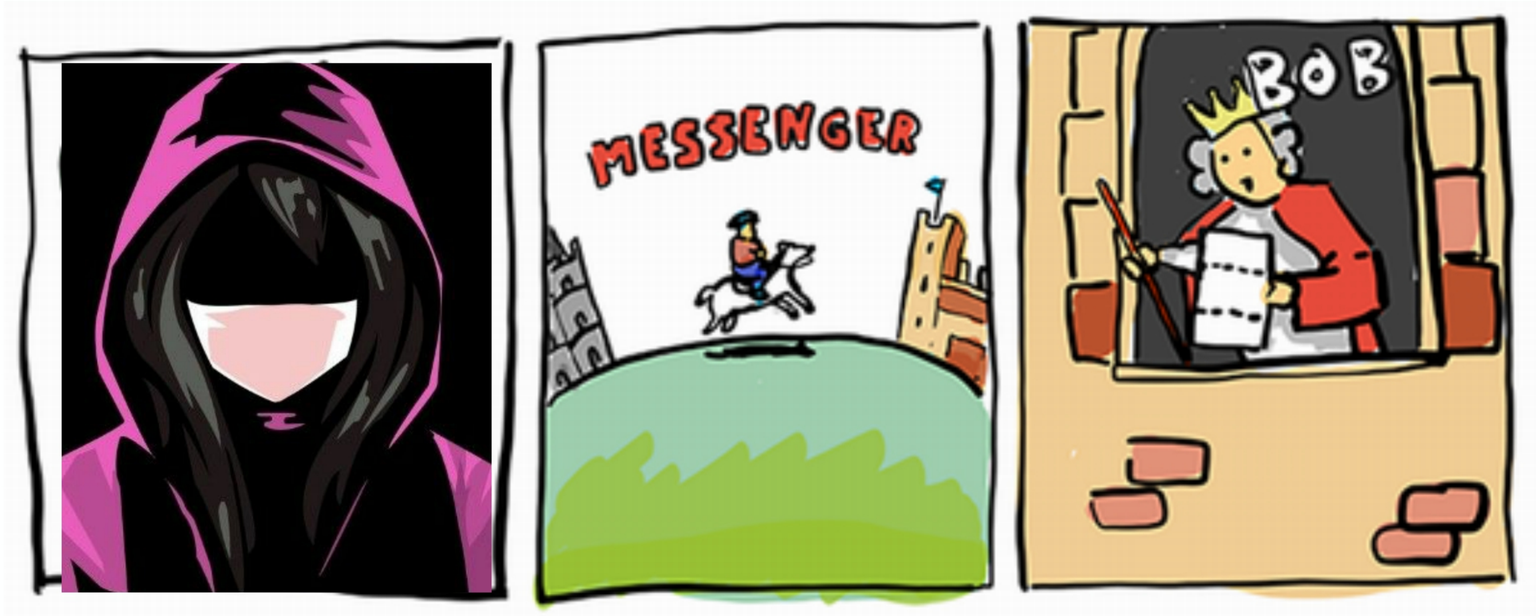
source: Real-World Cryptography by David Wong

3. Unauthenticated

- Encryption scheme used by packets: ✅

“Bob has no idea who the packet is from”

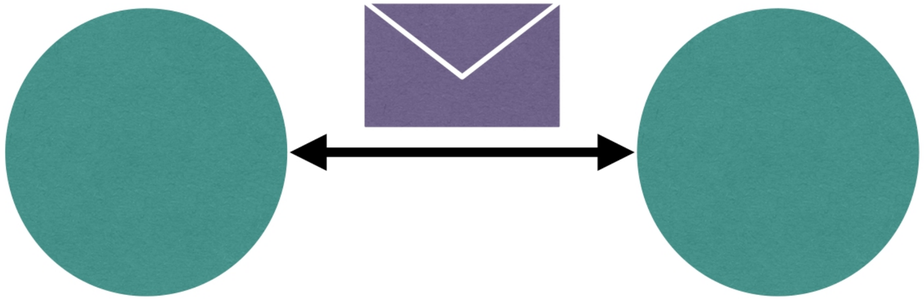
“packet hasn’t been tampered with”



BIP324

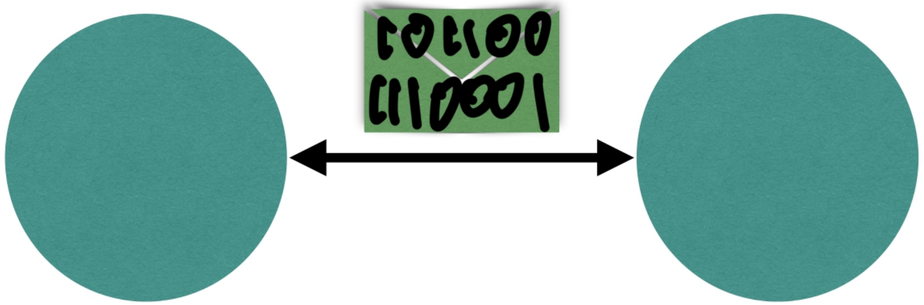
**new bitcoin P2P protocol with opportunistic encryption,
mild bandwidth reduction and upgradeability**

Before BIP324



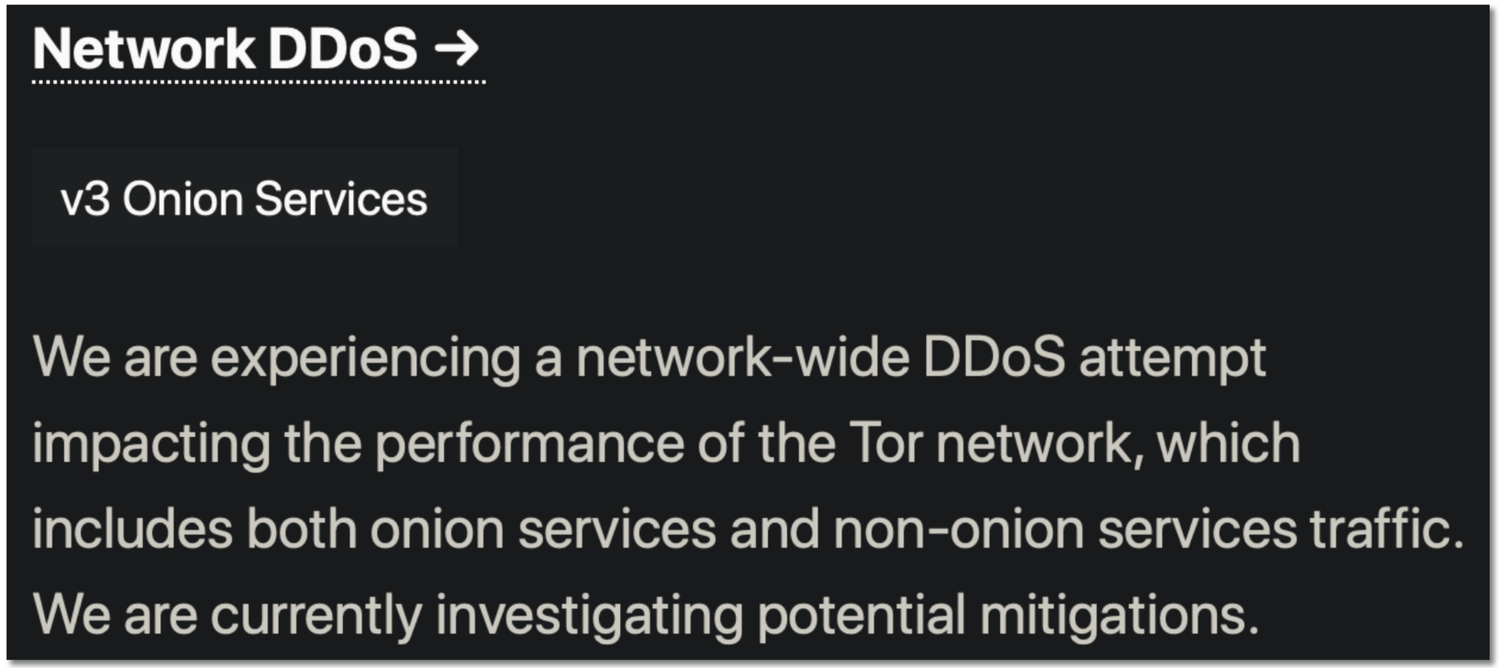
- Plaintext bytes
- Version handshake first

After BIP324



- Pseudorandom bytes
- Initial v2 handshake first

Why not just use tor ?



1. A solution which works everywhere
2. Tor has high latency
3. Cheaper network partition attacks

Goals

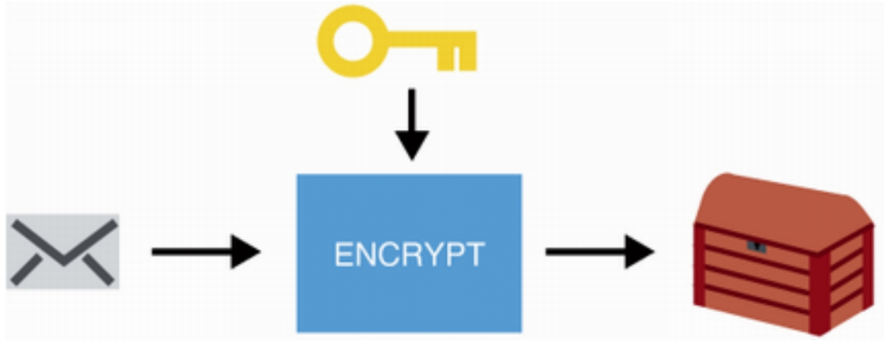
- Confidentiality against passive attacks
- Observability of active attacks
- Pseudorandom bytestream
- Shapable bytestream

Goals

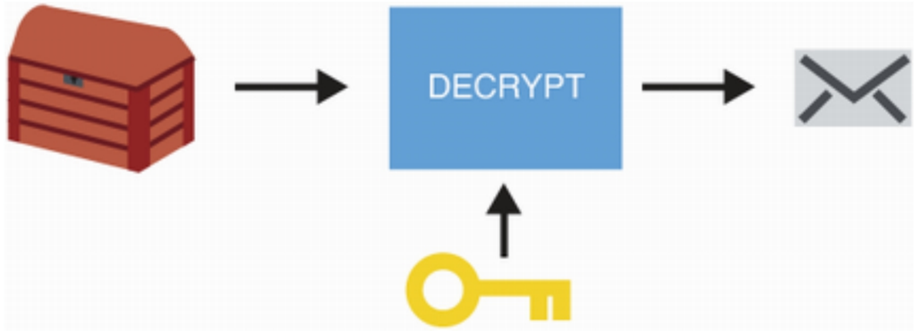
- **Forward secrecy**
- **Upgradability**
- **Compatability**
- **Low overhead**

[recap]: Cryptography

Encryption

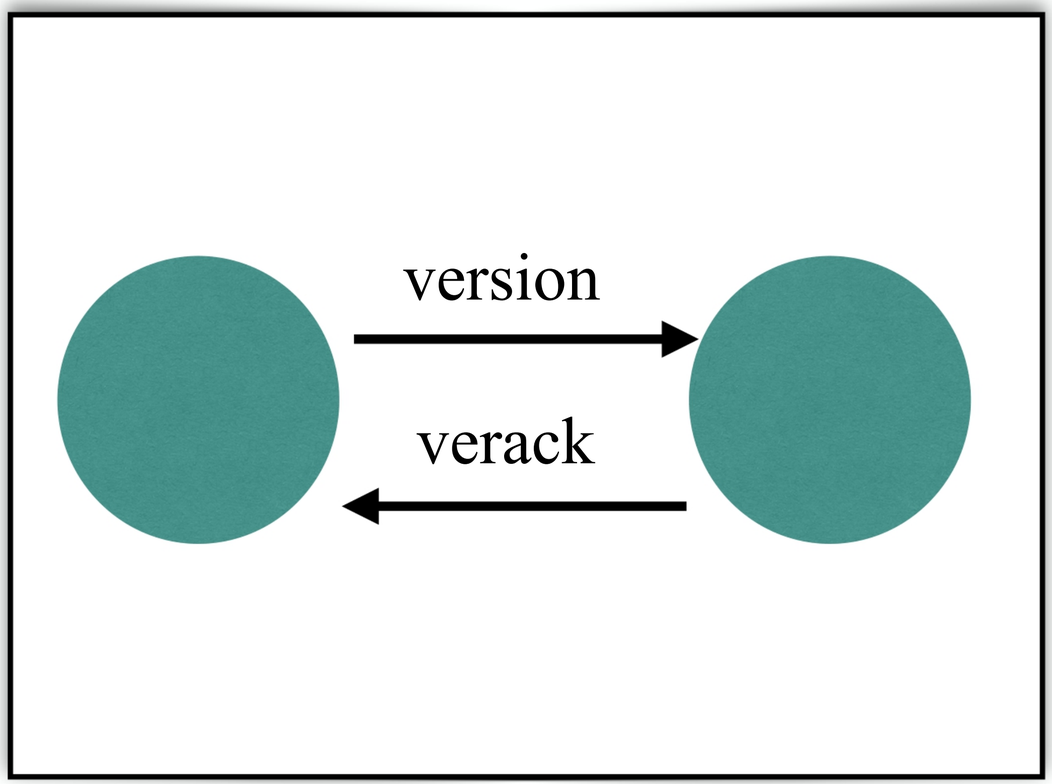


Decryption



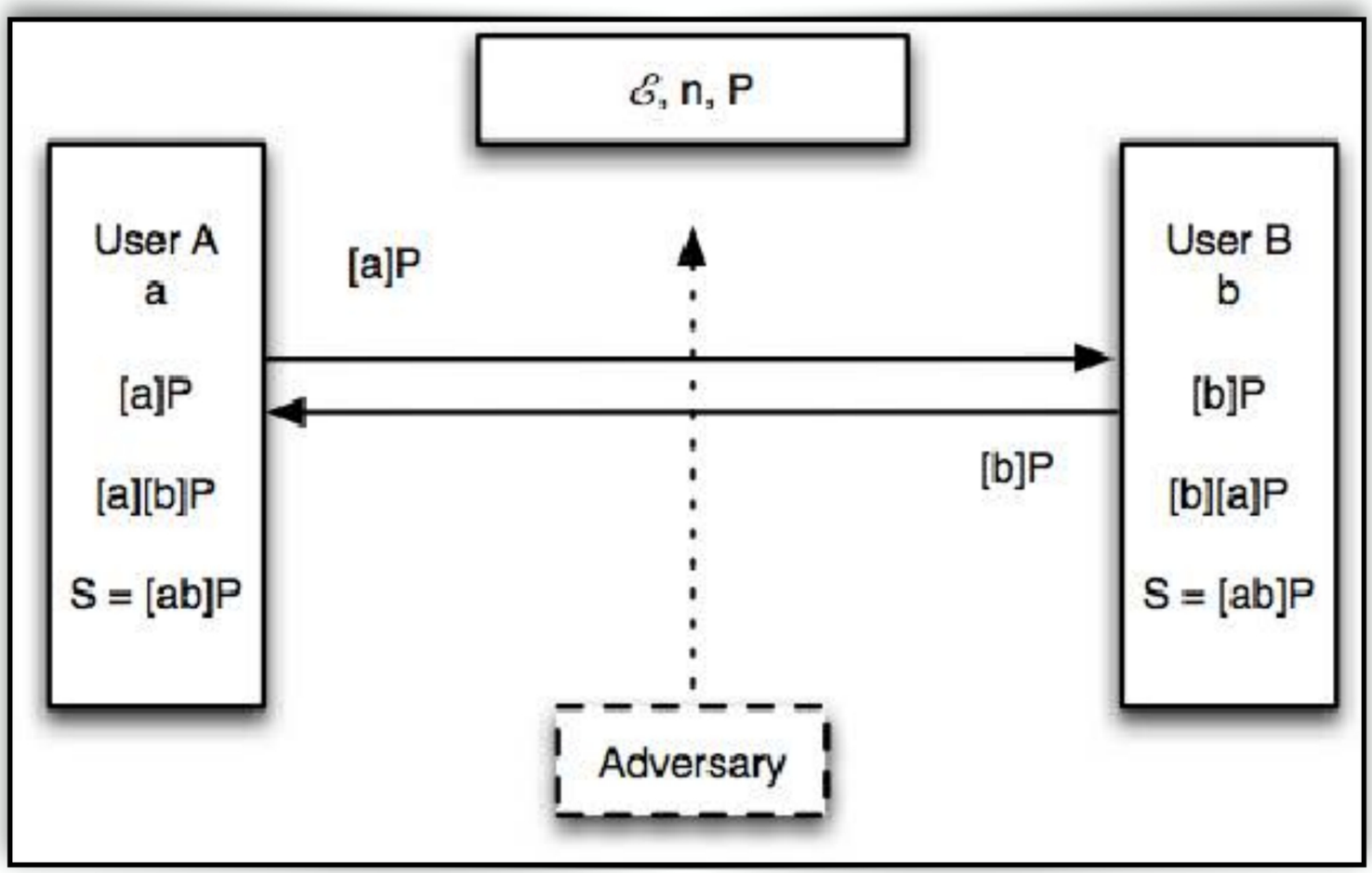
Initiator and Responder need same keys

[recap]: Version handshake



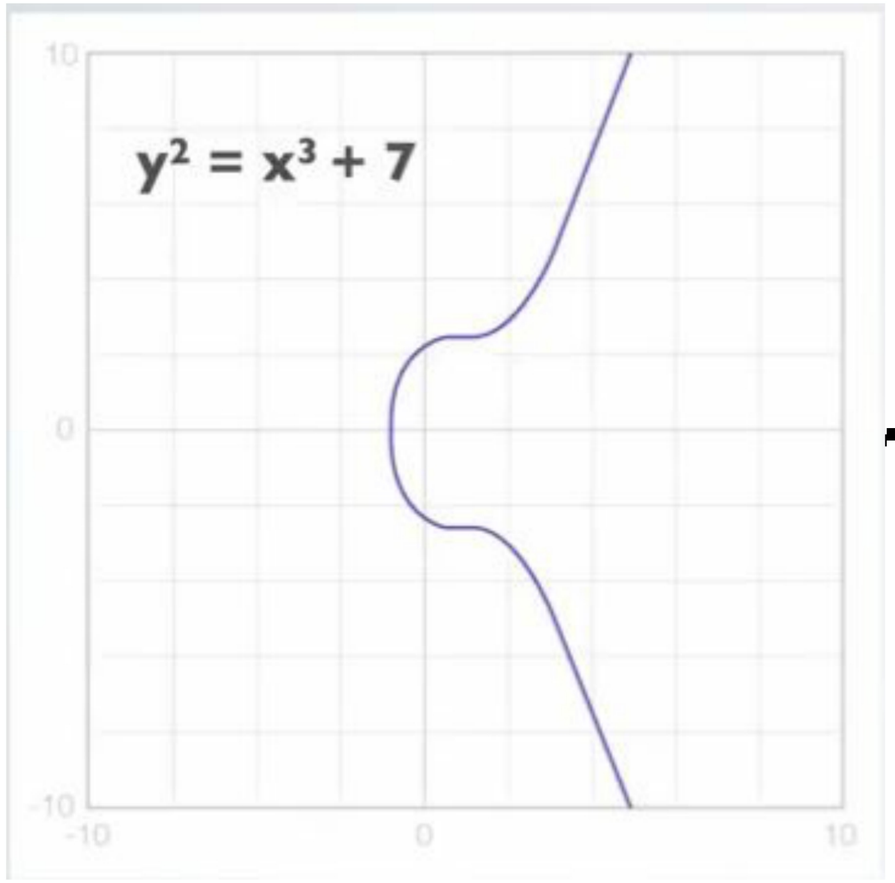
Initiator and Responder need keys before version handshake

[recap]: Elliptic Curve Diffie-Hellman(ECDH)



Establishes a shared secret over an insecure channel

[recap]: Elligator swift



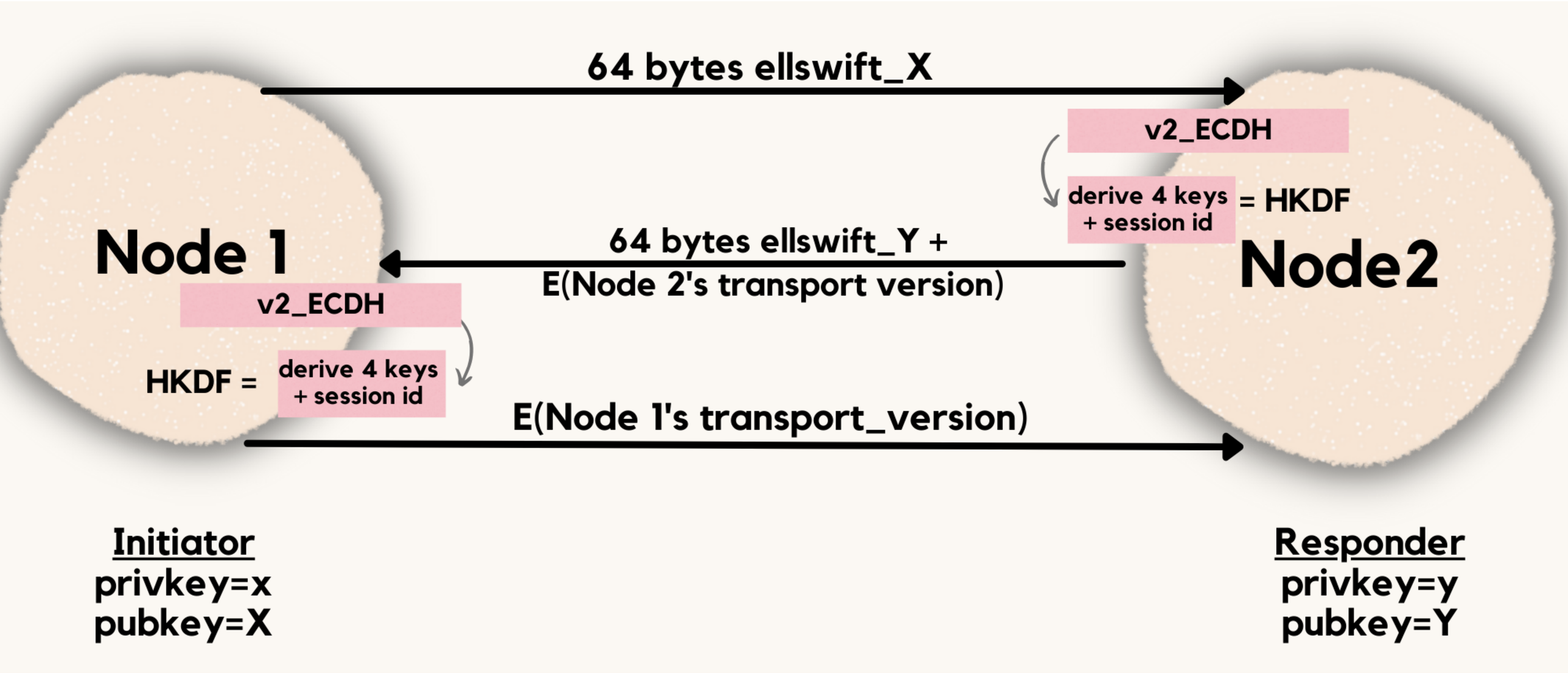
1010100110011

Elliptic curve point

Uniform byte string

Design

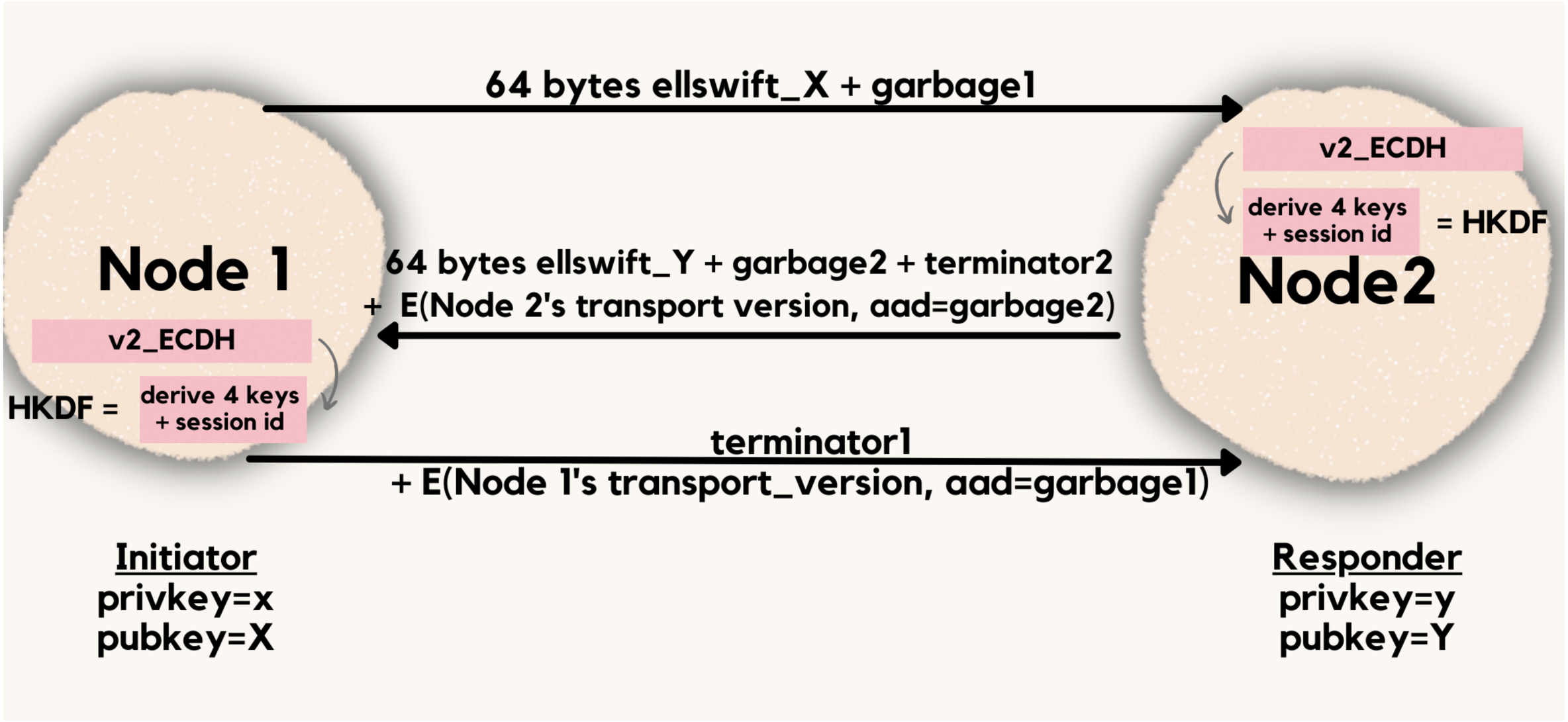
Naive approach - 1



Warning ⚠️ : detectable pattern (not used)

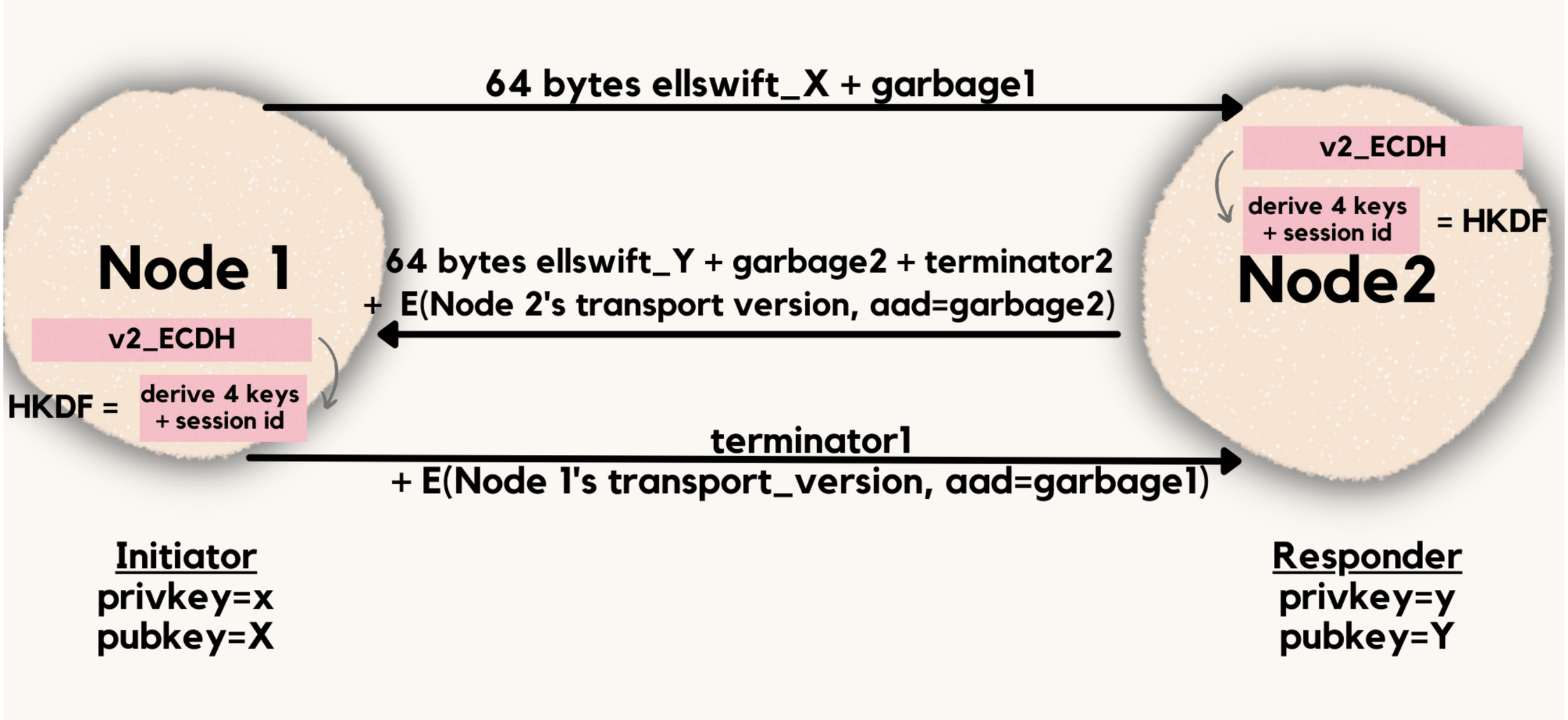
Design

Naive approach - 2



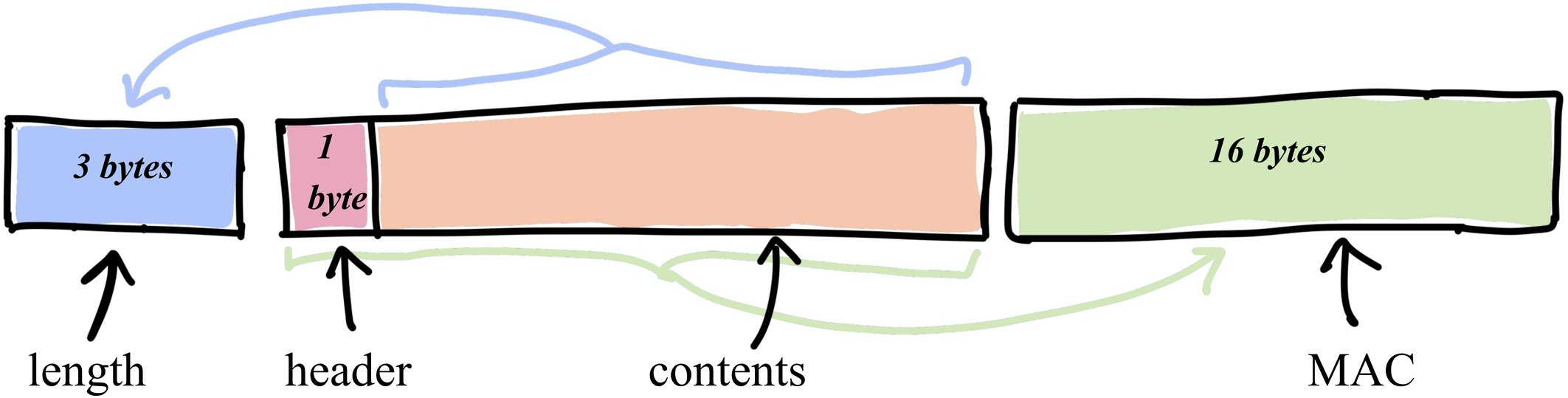
Warning ⚠️ : detectable pattern (not used)

Design



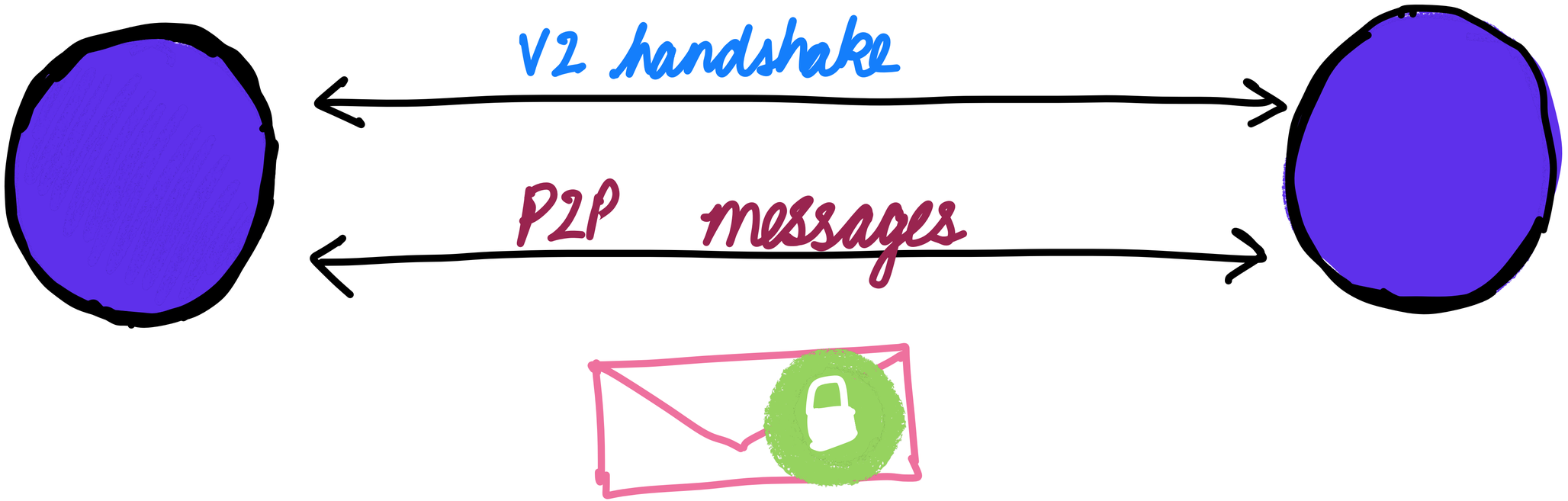
The responder does not wait for all 64 bytes to be received.
It responds back with its 64 bytes when 1st mismatch occurs.

Packet Encryption Cipher

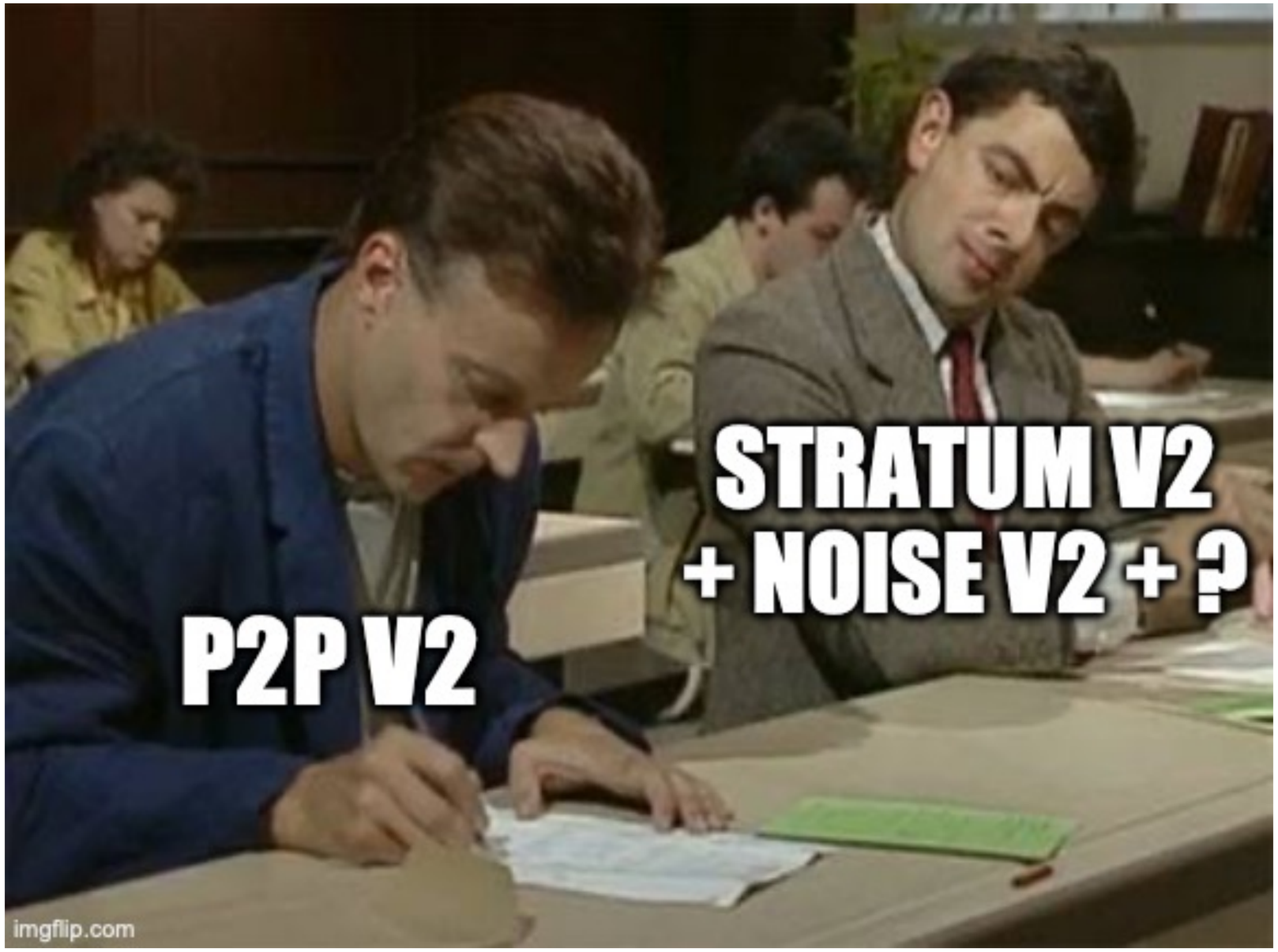


- *authenticated encryption scheme*
- *pseudorandom bytestream*

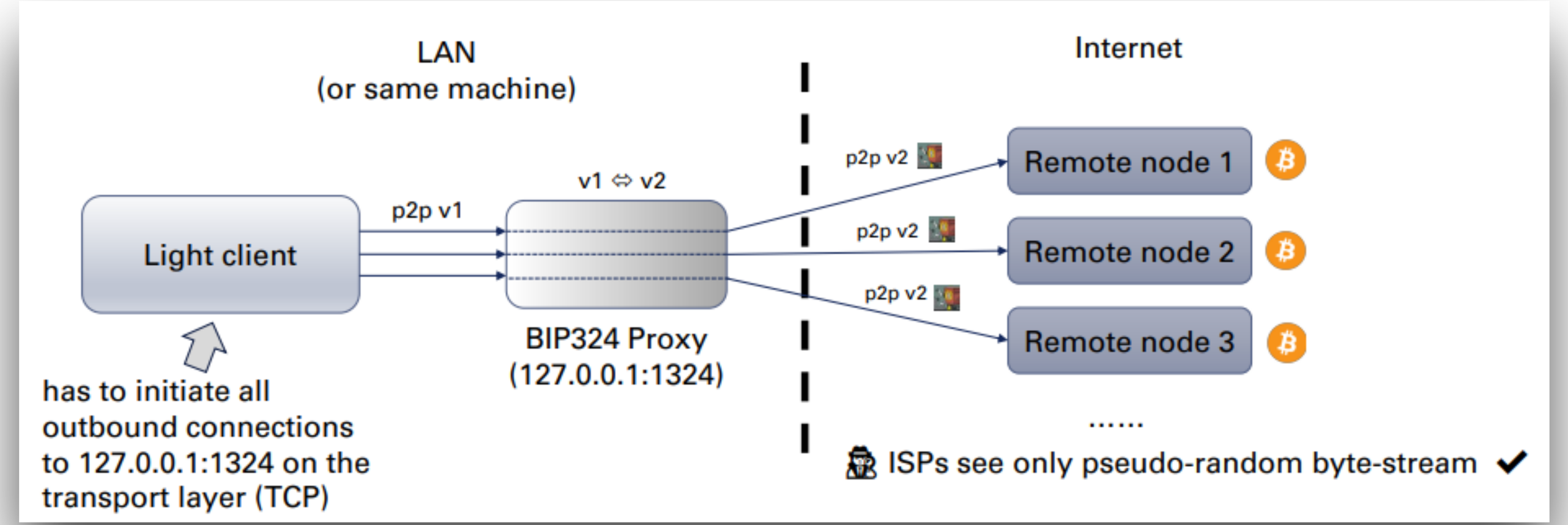
Recap!



we can do more!



BIP324 proxy



https://github.com/theStack/bip324-proxy

https://replit.com/@stratospher/proxymoto

Build ideas!

Trick wireshark into believing bitcoin protocol is something else!

Tasks:

1. Understand the code for how ellswift based ECDH work
2. Find out protocols which start with a fixed bytestring.
3. Modify the ellswift based ECDH code to start with the same fixed bytestring!
4. Were you able to trick wireshark?
5. Can you come up with more tricks?

Build ideas!

A monitoring dashboard for v2 connections.

1. What are traffic patterns you can can observe?
2. What are statistics you can collect for your v2 connections?

related examples:

- bitcoin-cli -netinfo 4
- Grafana dashboard for P2P messages

```
$ bitcoin-cli getpeerinfo
[
  {
    "addr": "ip address",
    # ...
    "servicesnames": [
      "WITNESS",
      "NETWORK_LIMITED",
      "P2P_V2"
    ],
    # ...
    # ...
    "connection_type": "manual",
    "transport_protocol_type": "v2",
    "session_id": "psst.. it's a secret"
  },
  # ...
]
```

Thank you!

https://github.com/stratospher/blogosphere/blob/main/btcpp_atx24.pdf