phd Positive Hack Days

Mining threats in Namecoin

Alexey Goncharov,
Positive Technologies

BREAKING THE CONSTANT

Namecoin

Alternate DNS registrar based on blockchain

- censor-proof, domains cannot be seized or blackholed

Maintains unofficial `.bit` TLD

Resolvers:

- nodes of OpenNIC project
- plugins for Chrome, Firefox and Opera
- ncdns - opensource project for full-featured authoritative server

«Bitcoin frees money – Namecoin frees DNS»

Used by:

- Chthronics
- Dimnie
- RTM
- GandCrab
- Smoke Loader
- Neutrino

Agenda

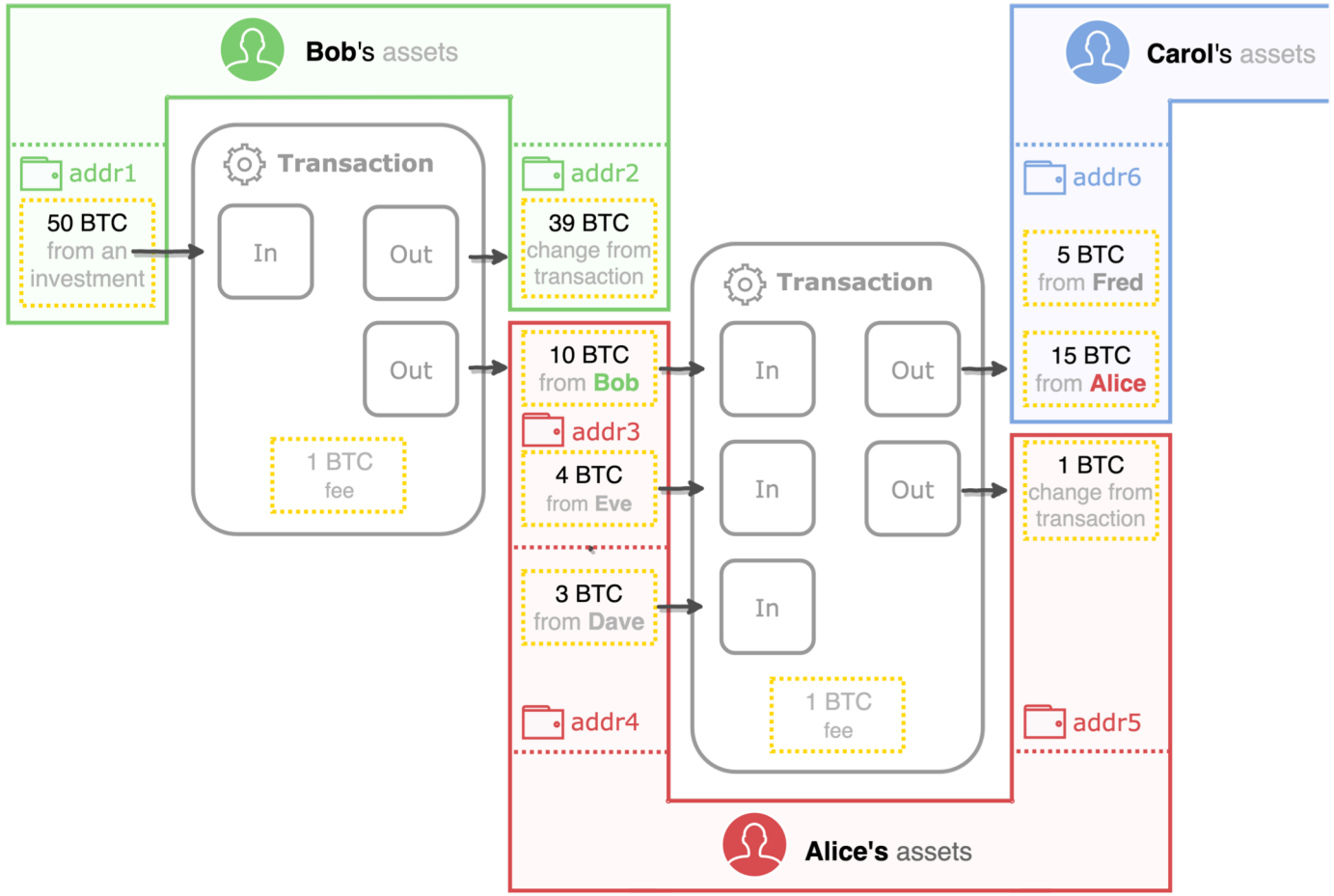
- Bitcoin 201
- How Namecoin works
- Mapping assets in Namecoin
- Takeaways

science ends here

- Bonus track #1
- Bonus track #2

Bitcoin 201

Bitcoin flow



Transaction structure

```
{ "hash": "90b18aa54288ec610d83ff1abe90f10d8ca87fb6411a72b2e56a169fdc9b0219",
  "ver": 1,
  "lock_time": 0,
  "size": 226,
  "vin_sz": 1,
  "vout_sz": 2,
```

Header

```
  "in": [
    { "prev_out": {
        "hash": "18798f8795ded46c3086f48d5bdabe10e1755524b43912320b81ef547b2f939a",
        "n": 0 },
      "scriptSig": "3045022100c1efcad5cdcc0dcf7c2a79d9e1566523af9c7229c78..." }
  ],
```

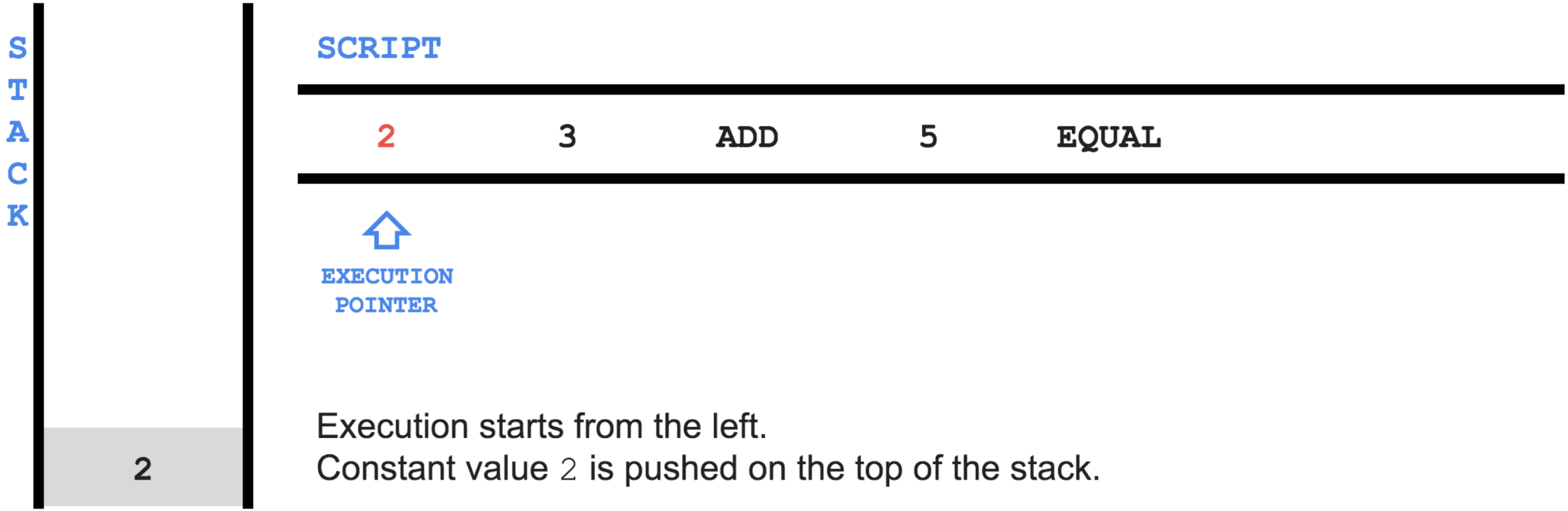
Inputs

```
  "out": [
    { "value": 4000000000,
      "scriptPubKey": "OP_DUP OP_HASH160 4b358739fc... OP_EQUALVERIFY OP_CHECKSIG"},
    { "value": 1000000000,
      "scriptPubkey": "OP_DUP OP_HASH160 55368b388c... OP_EQUALVERIFY OP_CHECKSIG"}
  ]
}
```

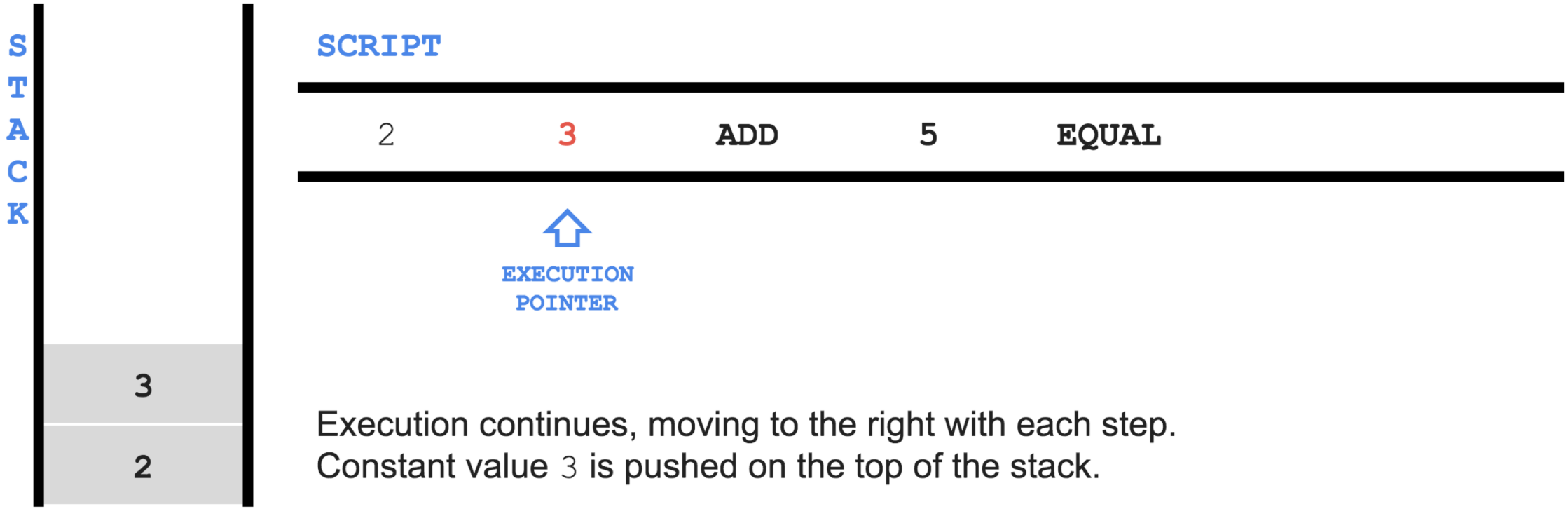
Outputs

2 3 OP_ADD 5 OP_EQUAL

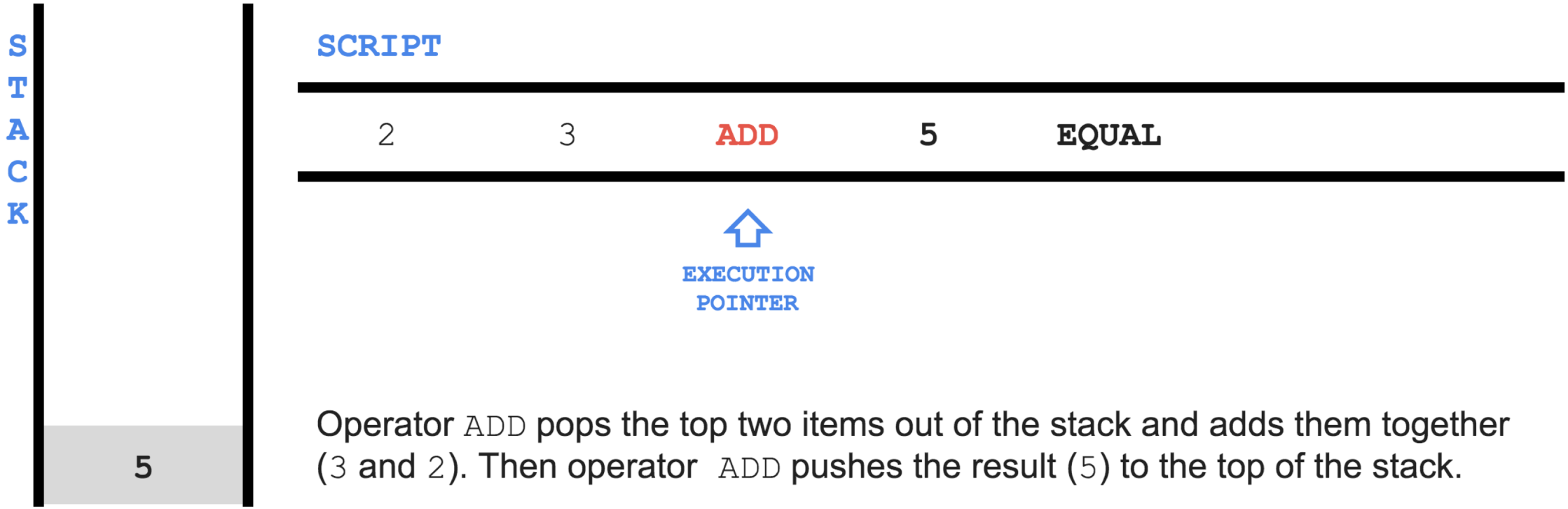
How Script works



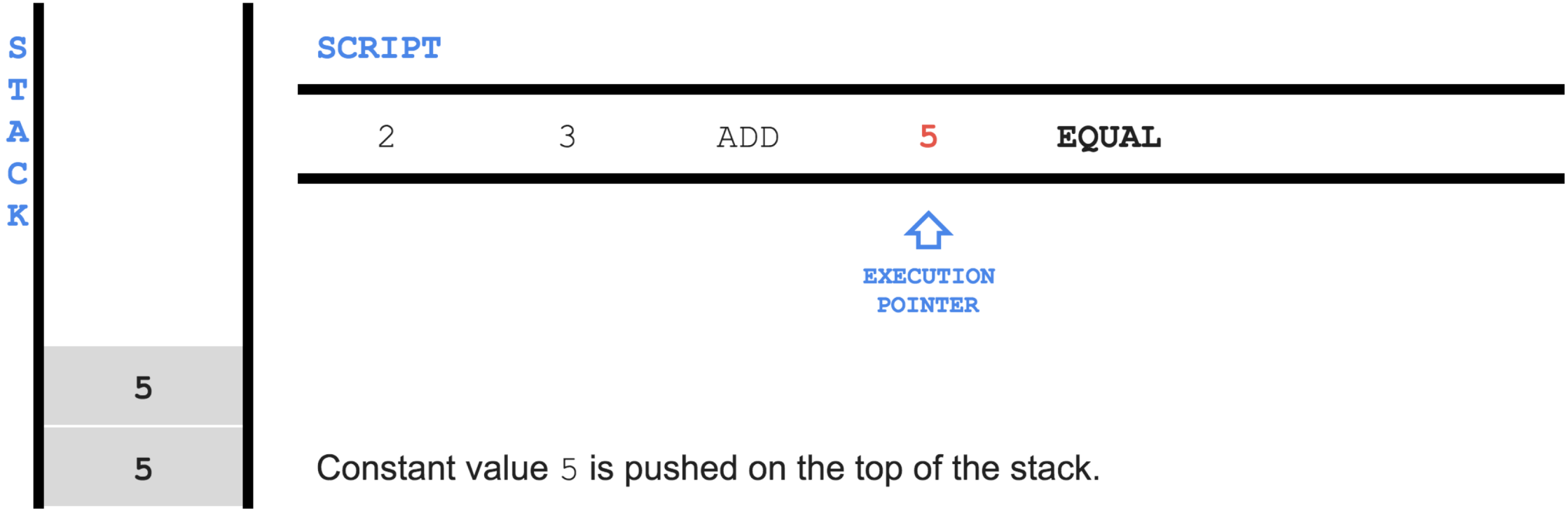
Execution starts from the left.
Constant value 2 is pushed on the top of the stack.



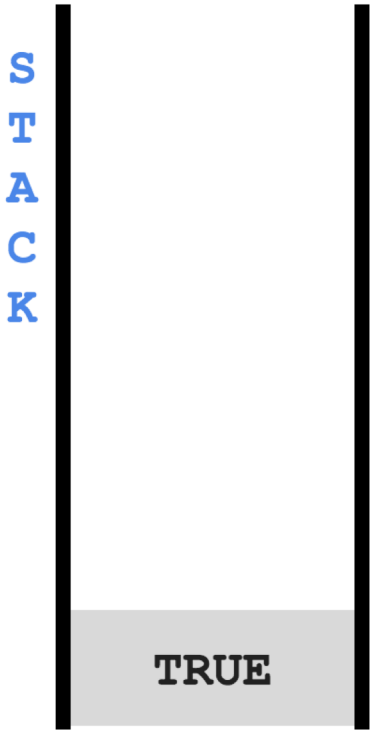
Execution continues, moving to the right with each step.
Constant value 3 is pushed on the top of the stack.



Operator `ADD` pops the top two items out of the stack and adds them together (`3` and `2`). Then operator `ADD` pushes the result (`5`) to the top of the stack.



Constant value 5 is pushed on the top of the stack.



SCRIPT

2	3	ADD	5	**EQUAL**

EXECUTION POINTER

Operator `EQUAL` pops the two top items out of the stack and compares the values (`5` and `5`) and if they are equal, `EQUAL` pushes `TRUE` of the top of the stack.

2 3 OP_ADD 5 OP_EQUAL


```
{ "hash": "90b18aa54288ec610d83ff1abe90f10d8ca87fb6411a72b2e56a169fdc9b0219",
  "ver": 1,
  "lock_time": 0,
  "size": 226,
  "vin_sz": 1,
  "vout_sz": 2,
```


Header


```
  "in": [
    { "prev_out": {
        "hash": "18798f8795ded46c3086f48d5bdabe10e1755524b43912320b81ef547b2f939a",
        "n": 0 },
      "scriptSig": "3045022100c1efcad5cdcc0dcf7c2a79d9e1566523af9c7229c78..." }
  ],
```


Inputs


```
  "out": [
    { "value": 4000000000,
      "scriptPubKey": "OP_DUP OP_HASH160 4b358739fc... OP_EQUALVERIFY OP_CHECKSIG"},
    { "value": 1000000000,
      "scriptPubkey": "OP_DUP OP_HASH160 55368b388c... OP_EQUALVERIFY OP_CHECKSIG"}
  ]
}
```


Outputs

P2PKH

Unlocking Script
(scriptSig)

+

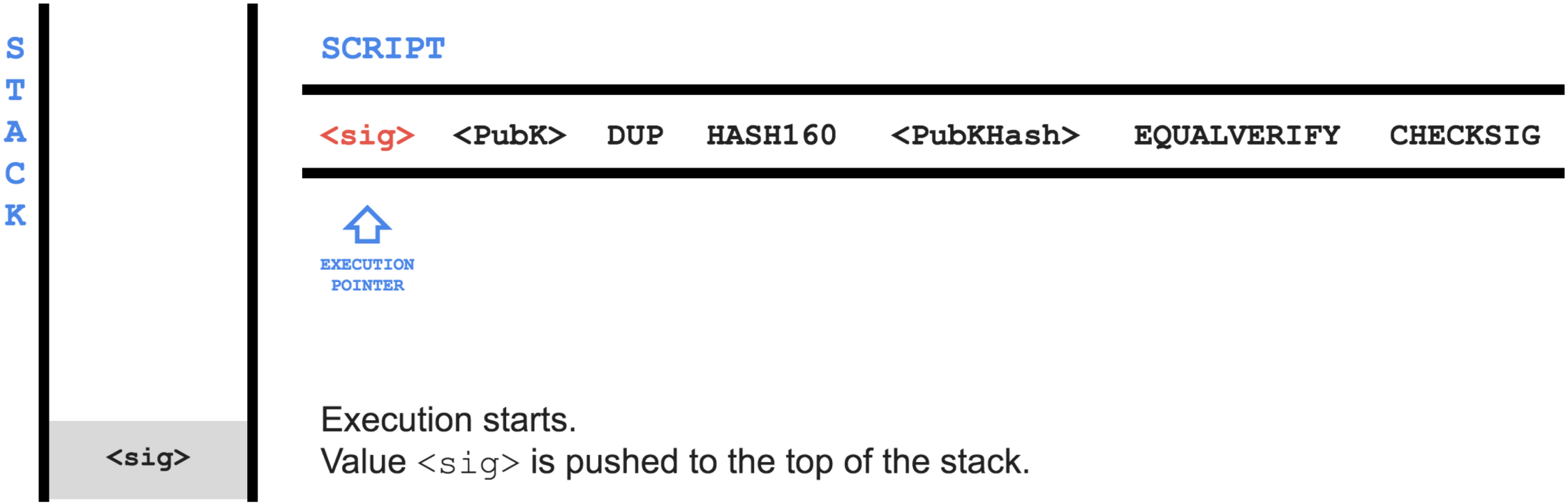
Locking Script
(scriptPubKey)

```
<sig> <PubK>
```

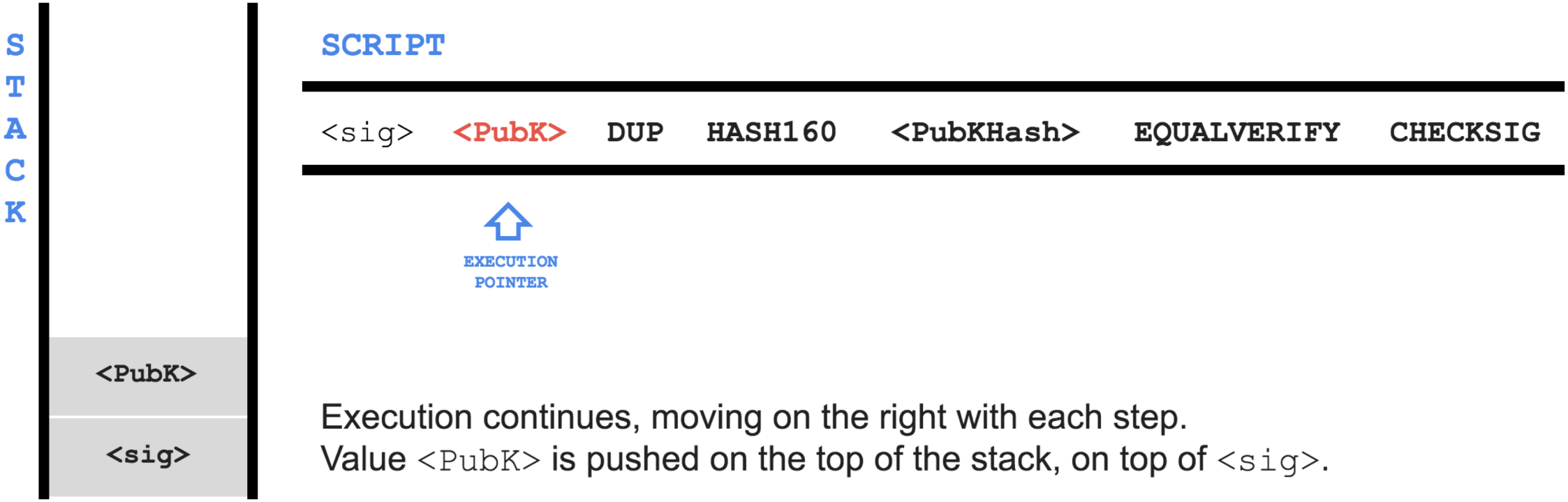
```
DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
```

Unlock Script (scriptSig) is provided by the user to resolve the encumbrance

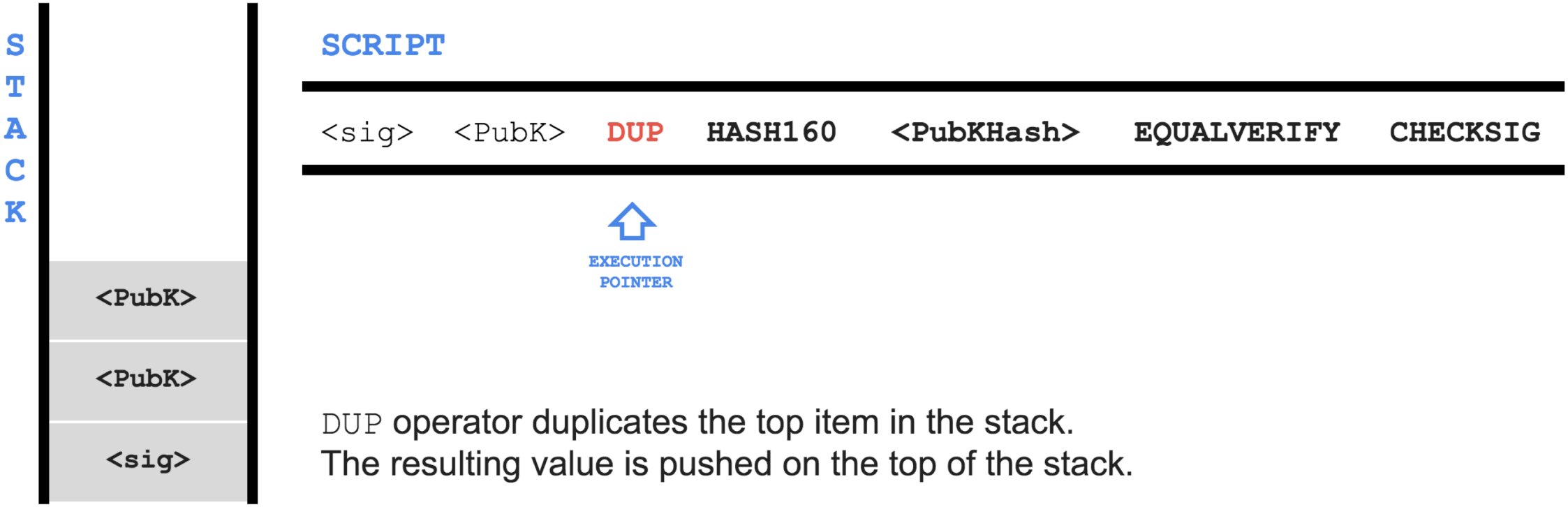
Lock Script (scriptPubKey) is found in a transaction output and is the encumbrance that must be fulfilled to spend the output



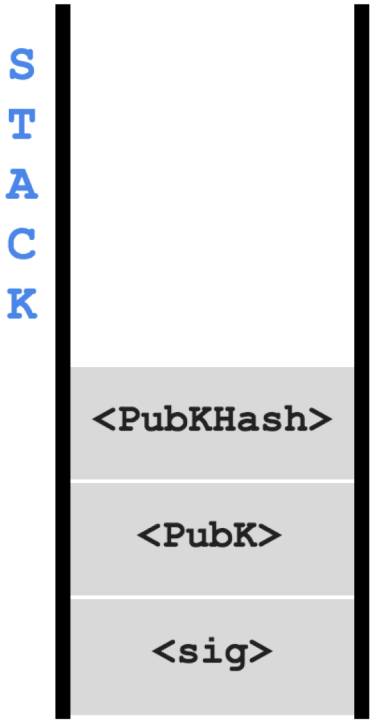
Execution starts.
Value `<sig>` is pushed to the top of the stack.



Execution continues, moving on the right with each step.
Value <PubK> is pushed on the top of the stack, on top of <sig>.



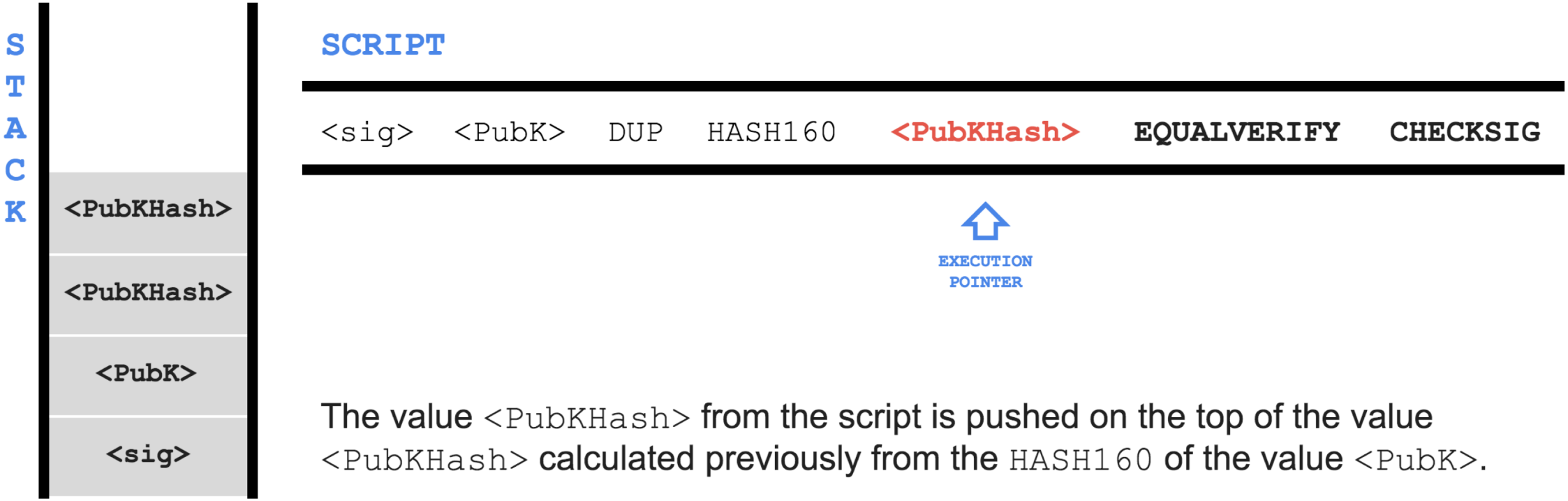
DUP operator duplicates the top item in the stack.
The resulting value is pushed on the top of the stack.



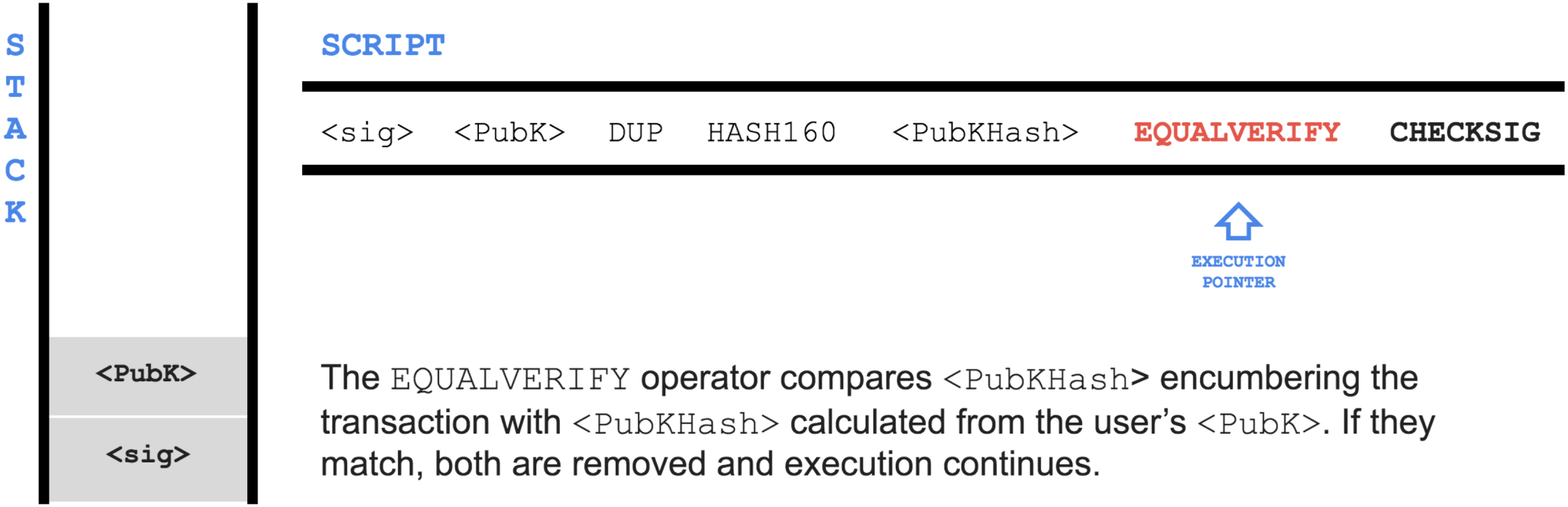
SCRIPT

`<sig>` `<PubK>` `DUP` **`HASH160`** **`<PubKHash>`** **`EQUALVERIFY`** **`CHECKSIG`**

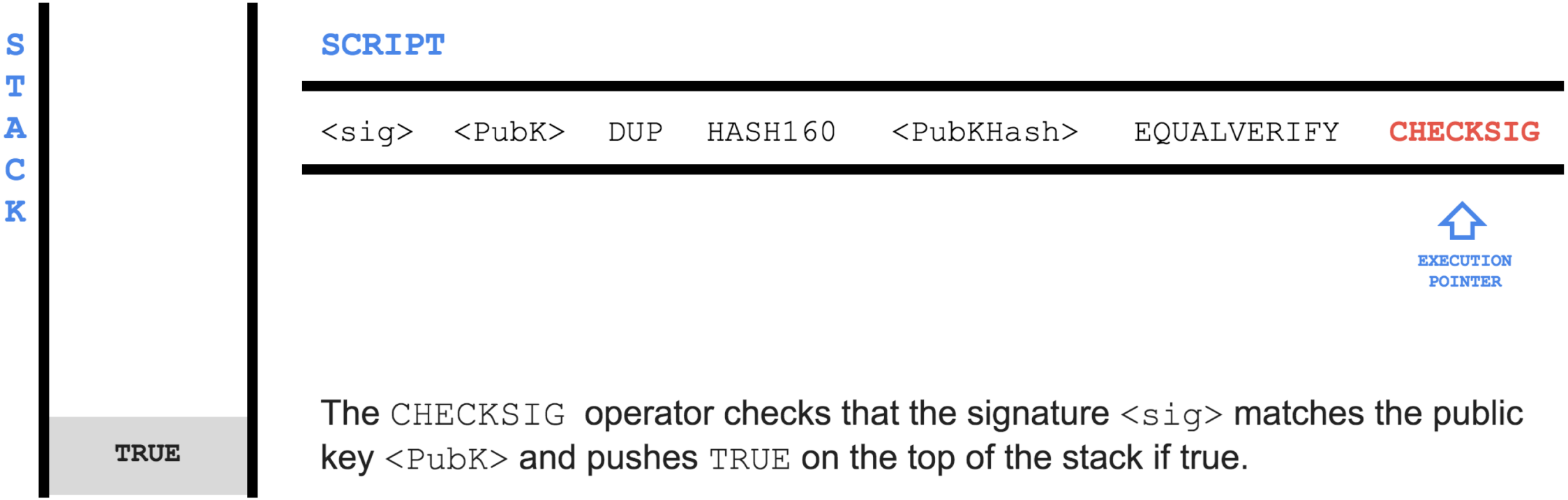
`HASH160` operator hashes the top item in the stack with `RIPEMD160(SHA256(<PubK>))`. The resulting value `<PubKHash>` is pushed on the top of the stack.



The value `<PubKHash>` from the script is pushed on the top of the value `<PubKHash>` calculated previously from the `HASH160` of the value `<PubK>`.



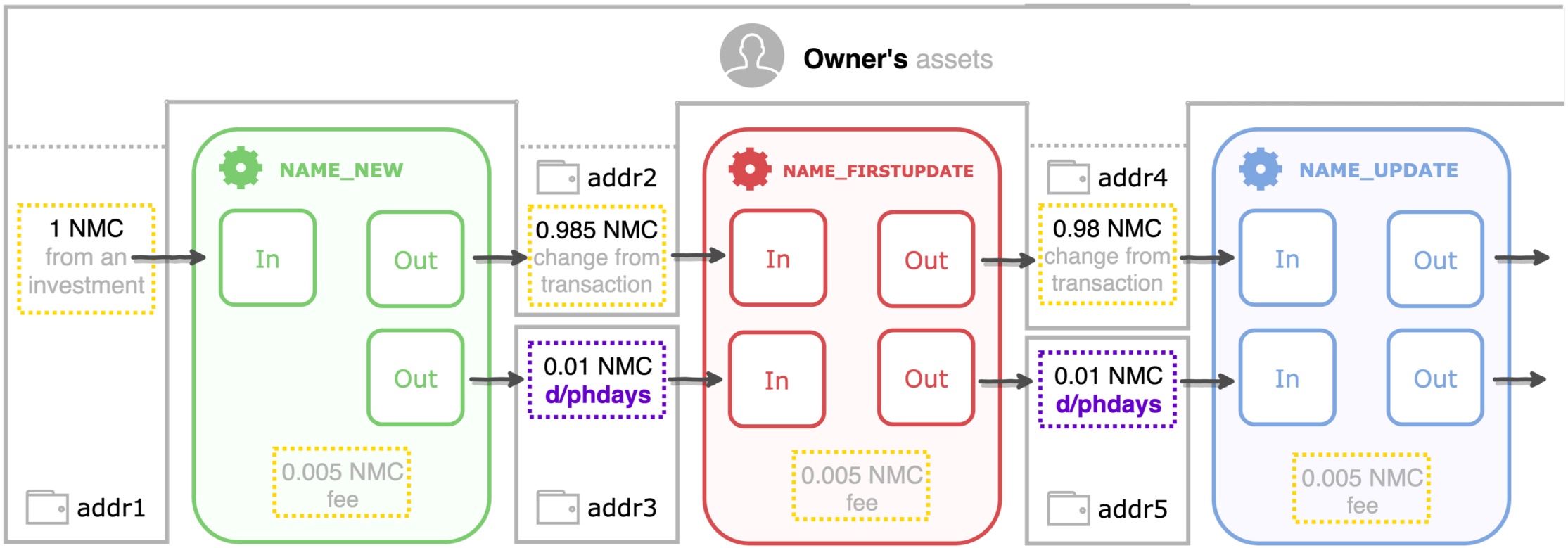
The EQUALVERIFY operator compares <PubKHash> encumbering the transaction with <PubKHash> calculated from the user's <PubK>. If they match, both are removed and execution continues.



The `CHECKSIG` operator checks that the signature `<sig>` matches the public key `<PubK>` and pushes `TRUE` on the top of the stack if true.

How Namecoin works

Namecoin flow



NAME_NEW

`OP_NAMENEW <20 byte hash> OP_2DROP <P2PKH>`

Pick a random salt	`d5 ee b2 2e e8 11 7f 57`
Convert `d/phdays` to ASCII	`64 2f 70 68 64 61 79 73`
Concatenate the salt with `d/phdays` in ASCII	`d5 ee b2 2e e8 11 7f 57 64 2f 70 68 64 61 79 73`
Hash the result with HASH160 (little endian)	`75 46 fa a7 ee d7 b4 9f a0 c7 dd 58 1b f2 a8 4f ff a6 74 36`

NAME_FIRSTUPDATE

```
OP_NAME_FIRSTUPDATE <Name> <Salt> <Value> OP_2DROP OP_2DROP <P2PKH>
```

d/phdays

d5 ee b2 2e e8 11 7f 57

{"ip":["1.2.3.4"]}

NAME_UPDATE

```
OP_NAME_UPDATE <Name> <Value> OP_2DROP OP_DROP <P2PKH>
```

d/phdays

{"ip":["1.1.1.1"]}

Namecoin Economy

- Chthronic C&C: registered in **2016**, still alive

Name d/pationare (pationare.bit)

Summary

Status	Active
Expires after block	464604 (22089 blocks to go)
Last update	2018-12-07 13:30:11 (block 428604)
Registered since	2016-12-25 13:43:35 (block 319753)

Current value

```
{
    "ns": [
        "a.dnspod.com",
        "b.dnspod.com"
    ]
}
```

Operations

Date/time	Block	Transaction	Operation	Value
2018-12-07 13:30:11	428604	09381e8d8c...	OP_NAME_UPDATE	{"ns":["a.dnspod.com","b.dnspod.com"]}
2018-05-02 21:18:06	396669	fbe8fedb09...	OP_NAME_UPDATE	{"ns":["ANIRBAN.NS.CLOUDFLARE.COM","REZA.NS.CLOUDFLARE.COM"]}
2018-01-11 20:08:35	379573	6067d7a10e...	OP_NAME_UPDATE	{"ns":["ns1.dnscontrolfff.to","ns2.dnscontrolfff.to"]}

- from **0.0109 NMC** per year
  - 0.00763 USD
  - ~0.5 RUR
- **0.08 USD** per year for daily updates
- compare with **1 USD/year** for **.info**
- or **10 USD/year** for **.com**
- **OPEX** instead of **CAPEX**

Threat mining

Blockchain crawler

Upstream movement

Collects:

- domain names
- IP addresses
- Namecoin addresses managed by threat actor
- unspent coins (UTXO)

Downstream movement looks similar

Heuristics will be discussed later

```
def upstream_movement(tx):
    global names
    global IPs
    global utxo
    global known_addresses

    heuristic_result = upstream_heuristic_test(tx)

    if heuristic_result and heuristic_result.guiding_outs:
        if tx.has_name_op():
            names.add(tx.name_op.name)
            for ip_address in tx.name_op.get_ip():
                IPs.add(ip_address)
        for guiding_out in heuristic_result.guiding_outs:
            known_addresses.add(guiding_out.address)
            tx = namecoin.transactions.find_one({"in.id": guiding_out.id})
            if tx:
                upstream_movement(tx)
            else:
                utxo.add(guiding_out)
```

«Common Change»

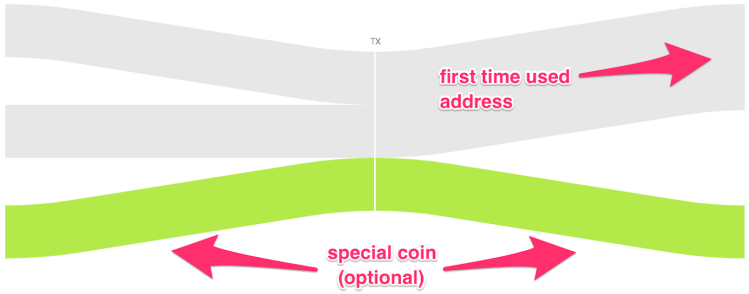
«If the output of the transaction is ***only one ordinary coin***, then this coin belongs to the person who owns the input coins.

If at the same time there is a ***special coin*** at the output, then it also belongs to the person who owns the input coins.

All addresses used in such a transaction are managed by the same person»

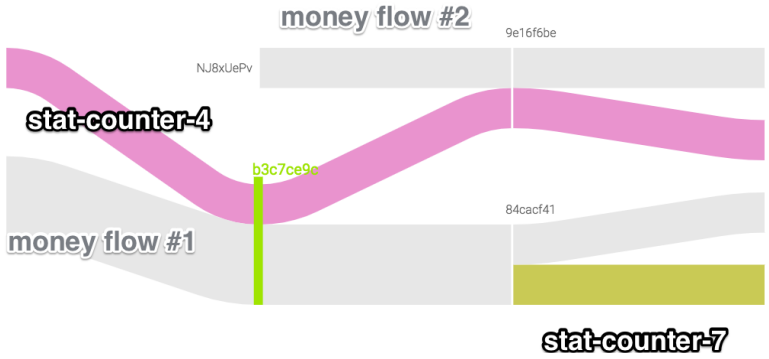
Case #1: First time used address

- Produced by domain creation and update
- Default behavior of native clients – *namecoind*, *namecoin-qt*
- The most common transaction pattern in Namecoin



Case #2: Re-used address

- Corresponds to domain transfer
- In common case **doesn't mean** that owners are the same, but
- It makes **no sense in acquiring a malicious domain**, so it is considered as a transfer between the accounts of the same person



«Common Spending»

«If it is known that ***at least one of the addresses*** at the input of the transaction is managed by a certain person, then ***all other addresses*** at the inputs of this transaction are managed by the same person.

The coins at these inputs belong to the same person»

- Can be used for *downstream movement* only
- Requires data from *Common Change* heuristic (addresses are managed by threat actor)

```
def common_spending(tx):
    result = { "guiding_ins": [] }

    for input in tx.get_ins():
        if input.address in known_addresses:
            return {"guiding_ins": tx.ins.all}

  return {}
```

«Known Address»

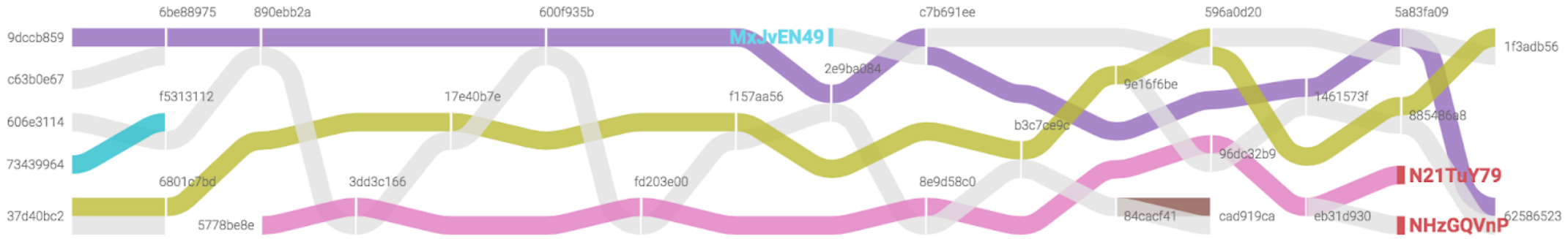
«If it is known that ***the address*** at the input (output) of the transaction is managed by a certain person, then ***the coins*** that put at this address (spent from this address) belong to the same person»

- Can be used for both upstream and downstream movement
- Also requires addresses from *Common Change*

```
def known_address(tx):
    result = { "guiding_outs": [], "guiding_ins": [] }

    for output in tx.get_outs():
        if output.address in known_addresses:
            result["guiding_outs"].append(output)
    for input in tx.get_ins():
        if input.address in known_addresses:
            result["guiding_ins"].append(input)
    return result
```

Dangling inputs and outputs



Takeaways

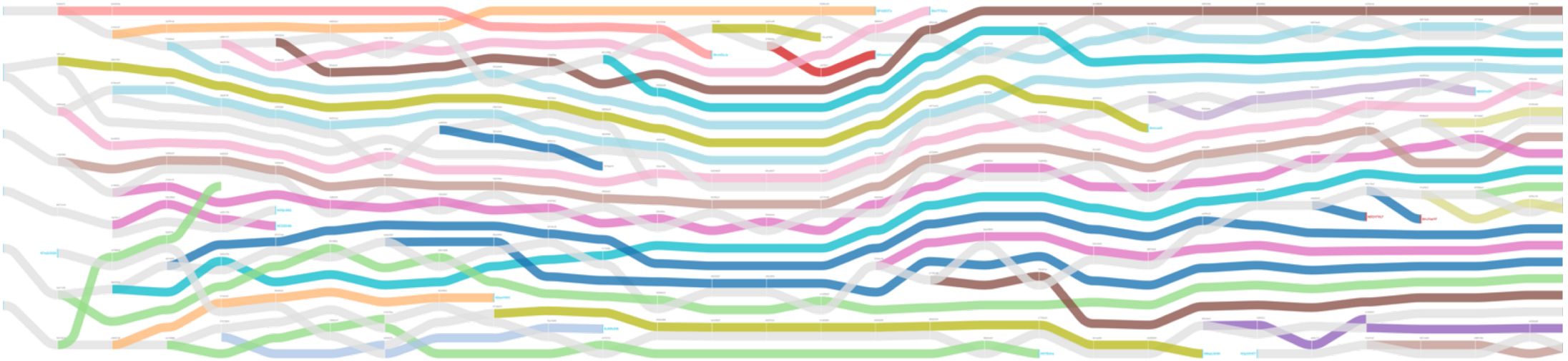
Takeaways

https://github.com/b4bay/rusty-blockparser

https://github.com/b4bay/threat_mining_in_namecoin

- RTM, Shifu, Dimnie and GandCrab
- 164 domains
- 277 IP addresses
- 39 UTXO

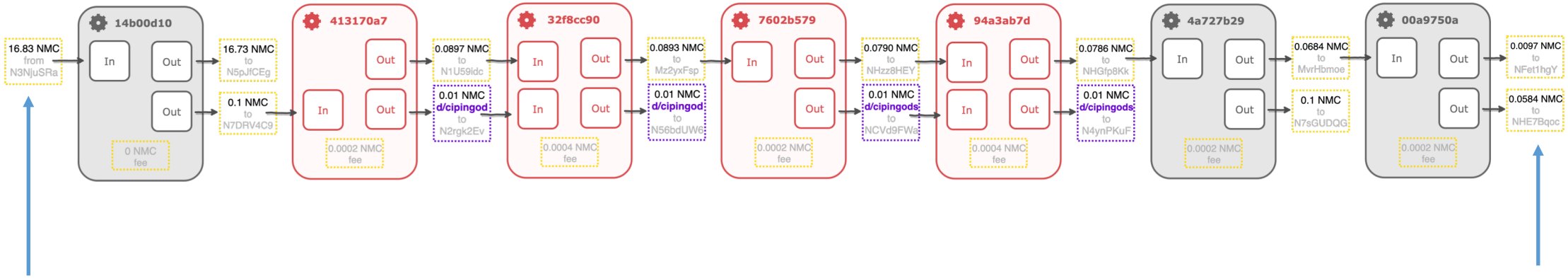
b4baysky



Bonus track

May contains assumptions with no strong proof

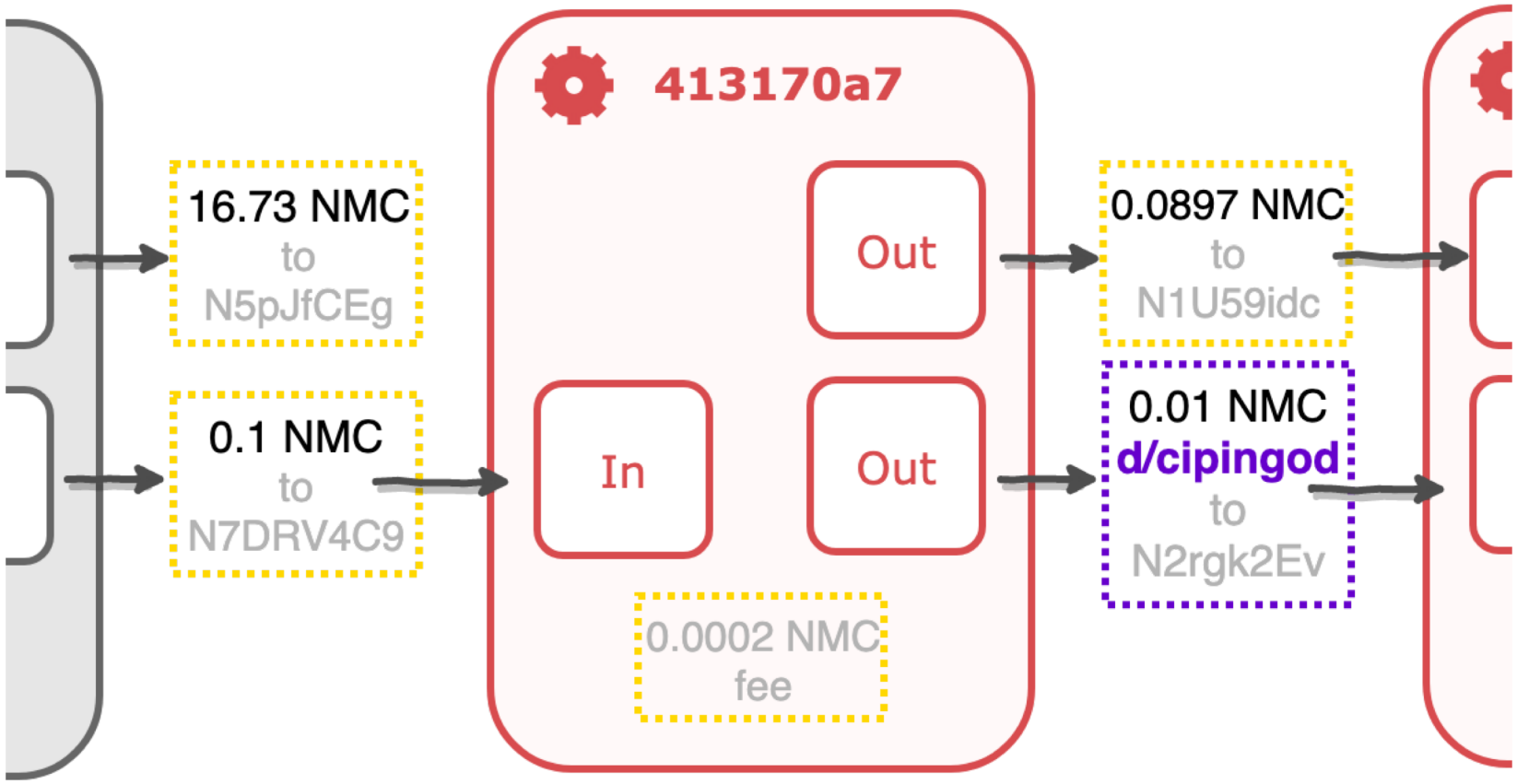
Dimnie ♥ RTM



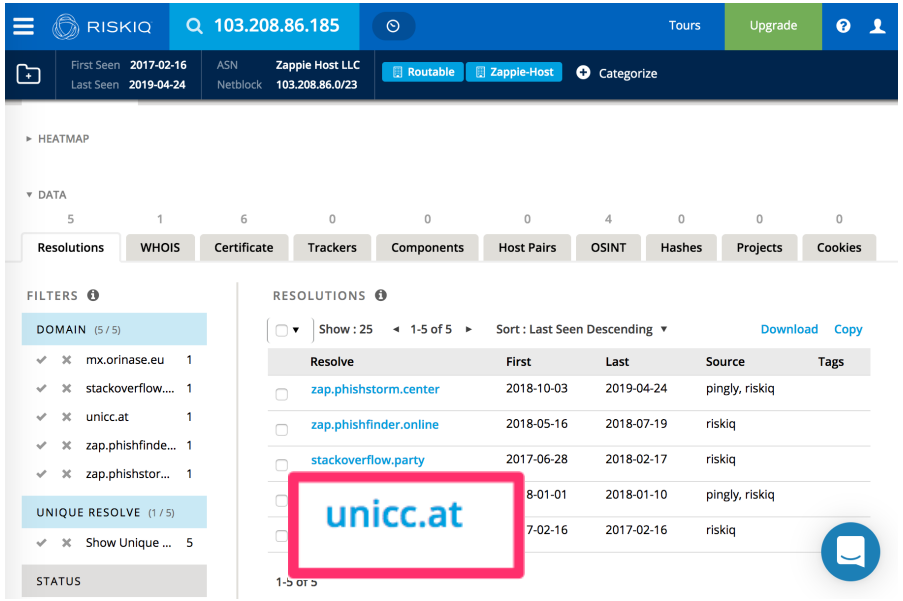
managed by Dimnie
used to create gosmos.bit

managed by RTM
used to create baa1tldbpi.bit

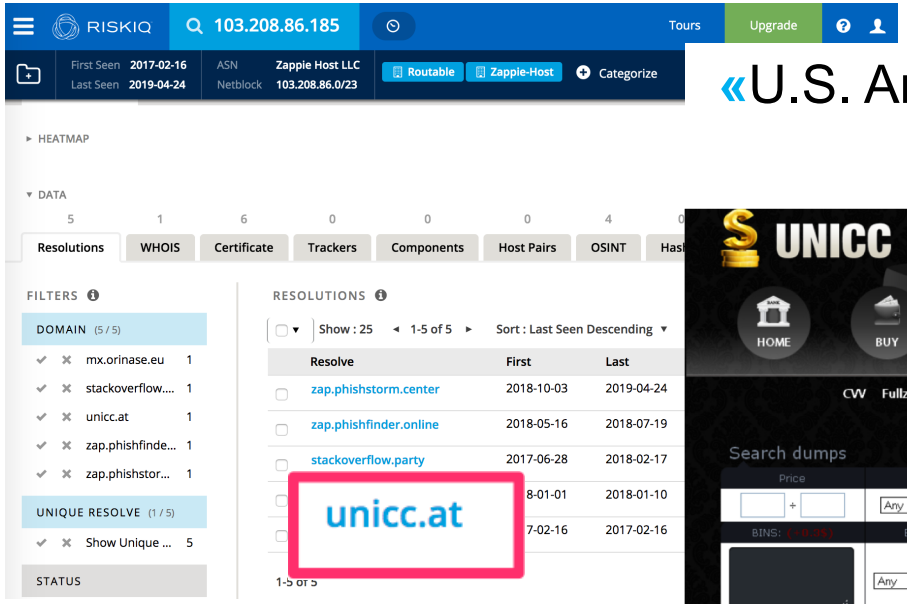
cipingod.bit ➔ 103.208.86.185



cipingod.bit ➔ 103.208.86.185

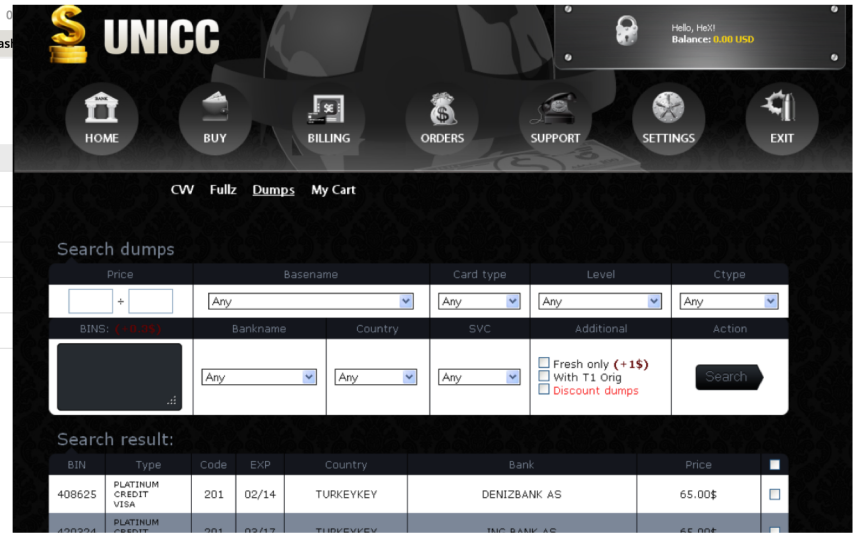


cipingod.bit ➔ 103.208.86.185

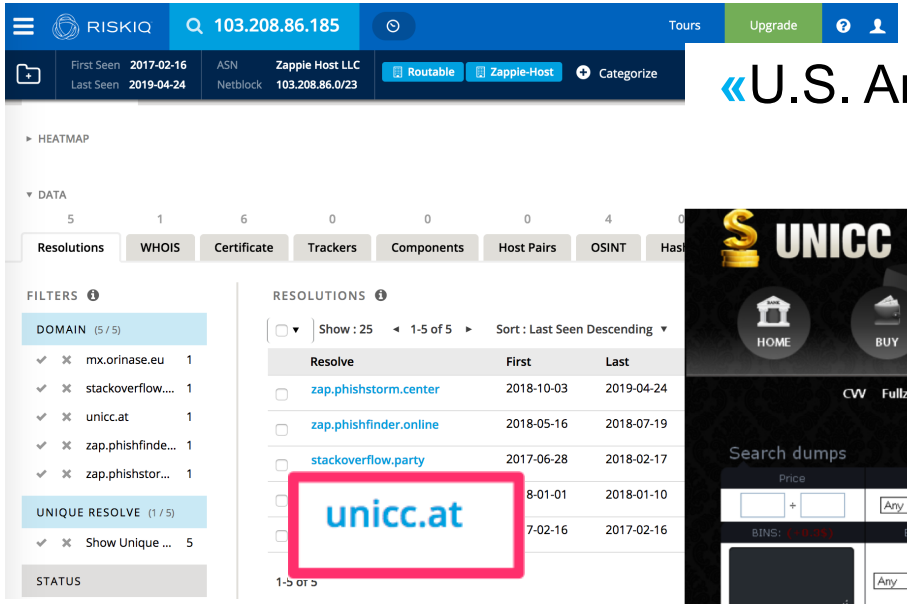


«U.S. Arrests 13, Charges 36 in ‘Infraud’ Cybercrime Forum Bust»

// Brian Krebs, 08 Feb 2018



cipingod.bit ➔ 103.208.86.185



«U.S. Arrests 13, Charges 36 in ‘Infraud’ Cybercrime Forum Bust»

// Brian Krebs, 08 Feb 2018

«Thirty-six Defendants Indicted for Alleged Roles in Transnational Criminal Organization Responsible for More than $530 Million in Losses from Cybercrimes»

// The U.S Department of Justice, 07 Feb 2018

- Svyatoslav Bondarkeno of Ukraine;
- Amjad Ali aka “Amjad Ali Chaudary,” aka “RedruMZ,” aka “Amjad Chaudary,” 35, of Pakistan;
- Roland Patrick N’Djimbi Tchikaya aka “Darker,” aka “dark3r.cvv,” 37, of France;
- Miroslav Kovacevic aka “Goldjunge,” 32, of Serbia;
- Frederick Thomas aka “Mosto,” aka “istunna,” aka “Bestssn,” 37, of Alabama;
- Andrey Sergeevich Novak aka “Unicc,”
- Gennaro Fioretti aka “DannyLogort,” aka “Genny Fioretti,” 56, of Italy;
- Edgar Rojas aka “Edgar Andres Viloria Rojas,” aka “Guapo,” aka “Guap01988,” aka “Onlyshop,” 27, of Australia;
- John Telusma aka “John Westley Telusma,” aka “Peterelliot,” aka “Pete,” aka “Pette,” 33, of Brooklyn, New York;
- Rami Fawaz aka “Rami Imad Fawaz,” aka “Validshop,” aka “Th3d,” aka “Zatcher,” aka “Darkeyes,” 26, of Ivory Coast;
- Muhammad Shiraz aka “Moviestar,” aka “Leslie” of Pakistan;
- Jose Gamboa aka “Jose Gamboa-Soto,” aka “Rafael Garcia,” aka “Rafael101,” aka “Memberplex2006” aka “Knowledge,” 29, of Los Angeles, California;
- Alexey Klimenko aka “Grandhost,” 34, of Ukraine;
- Edward Lavoile aka “Eddie Lavoie,” aka “Skizo,” aka “Eddy Lavoile,” 29, of Canada;
- Anthony Nnamdi Okeakpu aka “Aslike1,” aka “Aslike,” aka “Moneymafia,” aka “Shilonng,” 29, of the United Kingdom;
- Pius Sushil Wilson aka “FDIC,” aka “TheRealGuru,” aka “TheRealGuruNYC,” aka “RealGuru,” aka “P01son,” aka “1nfection,” aka “1nfected,” 31, of Flushing, New York;
- Muhammad Khan aka “CoolJ2,” aka “CoolJ,” aka “Secureroot,” aka “Secureroot1,” aka “Secureroot2,” aka