# Transactions and Smart Contracts

# Timelocks 1


```
{
    "txid": "7cf0883a4a24664a1c77fdd465782f0d7c962b7942daa4408f3083ea659aef6a",
    "version": 2,
    "locktime": 500114,
    "vin": [
        {
            "txid": "42a3fdd7d7baea12221f259f38549930b47cec288b55e4a8facc3c899f4775da",
            "vout": 0,
            "scriptSig": "473044022048d1468895910edafe53d4ec4209192cc3a8f0f21e7b9811f83b5e419bfb57
            e002203fef249b56682dbbb1528d4338969abb14583858488a3a766f609185efe68bca0121031a455dab5e
            1f614e574a2f4f12f22990717e93899695fb0d81e4ac2dcfd25d00",
            "sequence": 4294967295
        }
    ],
    "vout": [
        {
            "value": 0.00990000,
            "n": 0,
            "scriptPubKey": "OP_HASH160 e9c3dd0c07aac76179ebc76a6c78d4d67c6c160a OP_EQUAL",

        }
    ]
}
```

# Timelocks 1

**locktime:**

- 0 propagate the message immediately
- 0 < x < 500 000 000 – transaction not valid until block x
- x > 500 000 000 – transaction not valid until unix epoch time x

- A lock for the *entire* transaction!!!

# Timelocks 1

Here is my transaction. You can spend it in 50 days ☺

Alice

Bob


```
{
    "txid": "7cf0883a4a24664a1c77fdd465782f0d7c962b7942daa4408f3083ea659aef6a",
    "version": 2,
    "locktime": 500114,
    "vin" : [
        {
            "txid": "42a3fdd7d7baea12221f259f38549930b47cec288b55e4a8facc3c899f4775da",
            "vout": 0,
            "scriptSig": "473044022048d1468895910edafe53d4ec4209192cc3a8f0f21e7b9811f83b5e419bfb57
            e002203fef249b56682dbbb1528d4338969abb14583858488a3a766f609185efe68bca0121031a455dab5e
            1f614e574a2f4f12f22990717e93899695fb0d81e4ac2dcfd25d00",
            "sequence": 4294967295
        }
    ],
    "vout": [
        {
            "value": 0.00990000,
            "n": 0,
            "scriptPubKey": "OP_HASH160 e9c3dd0c07aac76179ebc76a6c78d4d67c6c160a OP_EQUAL",

        }
    ]
}
```


Bob

# Timelocks 1

You know what, I'll just spend this money.

Alice

Bob


```
{
    "txid": "7cf0883a4a24664a1c77fdd465782f0d7c962b7942daa4408f3083ea659aef6a",
    "version": 2,
    "locktime": 500114,
    "vin": [
        {
            "txid": "42a3fdd7d7baea12221f259f38549930b47cec288b55e4a8facc3c899f4775da",
            "vout": 0,
            "scriptSig": "473044022048d1468895910edafe53d4ec4209192cc3a8f0f21e7b9811f83b5e419bfb57
            e002203fef249b56682dbbb1528d433
            1f614e574a2f4f12f22990717e93899
            "sequence": 4294967295
        }
    ],
    "vout": [
        {
            "value": 0.00990000,
            "n": 0,
            "scriptPubKey": "OP_HASH160 e9c
        }
    ]
}
```

```
{
    "version": 1,
    "locktime": 0,
    "vin": [
        {
            "txid": "42a3fdd7d7baea12221f259f38549930b47cec288b55e4a8facc3c899f4775d
            "vout": 0,
            "scriptSig": "304402207
            0f45e2590602201b06ecbbbf6fae0a3455d98ac2924cd6dc3022425a2f08c60ffb4
            e9[ALL] 04a84d304aa8963fbd36287e674f109827b6d6ea60d57a7d9357df03be
            475ca128b1a50408b7f584041ffd52d6b19aba5256e99dcdbbe2ed7373775d"
        },
        "sequence": 4294967295
    ],
    "vout": [
        {
            "value": 0.10000000,
            "n": 0,
            "scriptPubKey": "OP_DUP OP_HASH160 7f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a8 OP_EQUALVERIFY OP_CHECKSIG"
        }
    ]
}
```


Alice

# Timelocks 2 (CLTV)

**To avoid this BIP 65 introduces a new instruction:**

- OP_CHECKLOCKTIMEVERIFY (OP_CLTV)
- Now we can use locktime at the level of UTXOs

# CLTV


| Inputs (UTXO referenced) | | | |
|---|---|---|---|
| num | hash | Nr_output | Unlocking script |
| 0 | 0xff… | 4 | 012123… |
| **Outputs** | | | |
| num | value | Locking script | |
| 0 | 3.1 | OP_DUP OP_HASH160 <Bob> OP_EQUALVERIFY OP_CHECKSIG | |

# CLTV


| Inputs (UTXO referenced) | | | |
|---|---|---|---|
| num | hash | Nr_output | Unlocking script |
| 0 | 0xff… | 4 | 012123… |

| Outputs | | |
|---|---|---|
| num | value | Locking script |
| 0 | 3.1 | |

**<now + 30 days> OP_CHECKLOCKTIMEVERIFY OP_DROP** OP_DUP OP_HASH160 <Bob> OP_EQUALVERIFY OP_CHECKSIG

# CLTV

UTXO(Tx: 0xff, Nr_out: 0):

**<now + 30 days> OP_CHECKLOCKTIMEVERIFY OP_DROP** OP_DUP OP_HASH160 <Bob> OP_EQUALVERIFY OP_CHECKSIG

Bob

# CLTV

UTXO(Tx: 0xff, Nr_out: 0):

<now + 30 days> OP_CHECKLOCKTIMEVERIFY OP_DROP OP_DUP OP_HASH160 <Bob> OP_EQUALVERIFY OP_CHECKSIG

Bob

| Inputs (UTXO referenced) | | | |
|---|---|---|---|
| num | hash | Nr_output | Unlocking script |
| 0 | 0xff… | 0 | |

| Outputs | | |
|---|---|---|
| num | value | Locking script |
| 0 | 3.1 | OP_DUP OP_HASH... |

# CLTV

UTXO(Tx: 0xff, Nr_out: 0):

<now + 30 days> OP_CHECKLOCKTIMEVERIFY OP_DROP OP_DUP OP_HASH160 <Bob> OP_EQUALVERIFY OP_CHECKSIG

Bob

| Inputs (UTXO referenced) | | | |
|---|---|---|---|
| num | hash | Nr_output | Unlocking script |
| 0 | 0xff… | 0 | <sig Bob> <PK Bob> |
| **Outputs** | | | |
| num | value | Locking script | |
| 0 | 3.1 | OP_DUP OP_HASH... | |

# CLTV

UTXO(Tx: 0xff, Nr_out: 0):

<now + 30 days> OP_CHECKLOCKTIMEVERIFY OP_DROP OP_DUP OP_HASH160 <Bob> OP_EQUALVERIFY OP_CHECKSIG

Bob

| Locktime: <now + 100 days> | | | |
|---|---|---|---|
| Inputs (UTXO referenced) | | | |
| num | hash | Nr_output | Unlocking script |
| 0 | 0xff… | 0 | <sig Bob> <PK Bob> |
| Outputs | | | |
| num | value | Locking script | |
| 0 | 3.1 | OP_DUP OP_HASH... | |

# CLTV validation

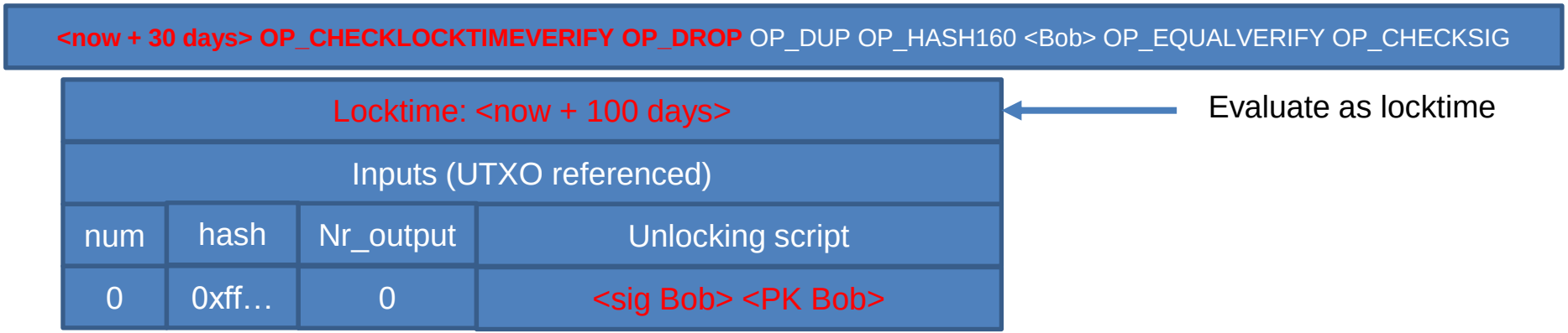
UTXO(Tx: 0xff, Nr_out: 0):

| **<now + 30 days> OP_CHECKLOCKTIMEVERIFY OP_DROP** OP_DUP OP_HASH160 <Bob> OP_EQUALVERIFY OP_CHECKSIG |
|---|

| Locktime: <now + 100 days> | | | |
|---|---|---|---|
| Inputs (UTXO referenced) | | | |
| num | hash | Nr_output | Unlocking script |
| 0 | 0xff… | 0 | <sig Bob> <PK Bob> |

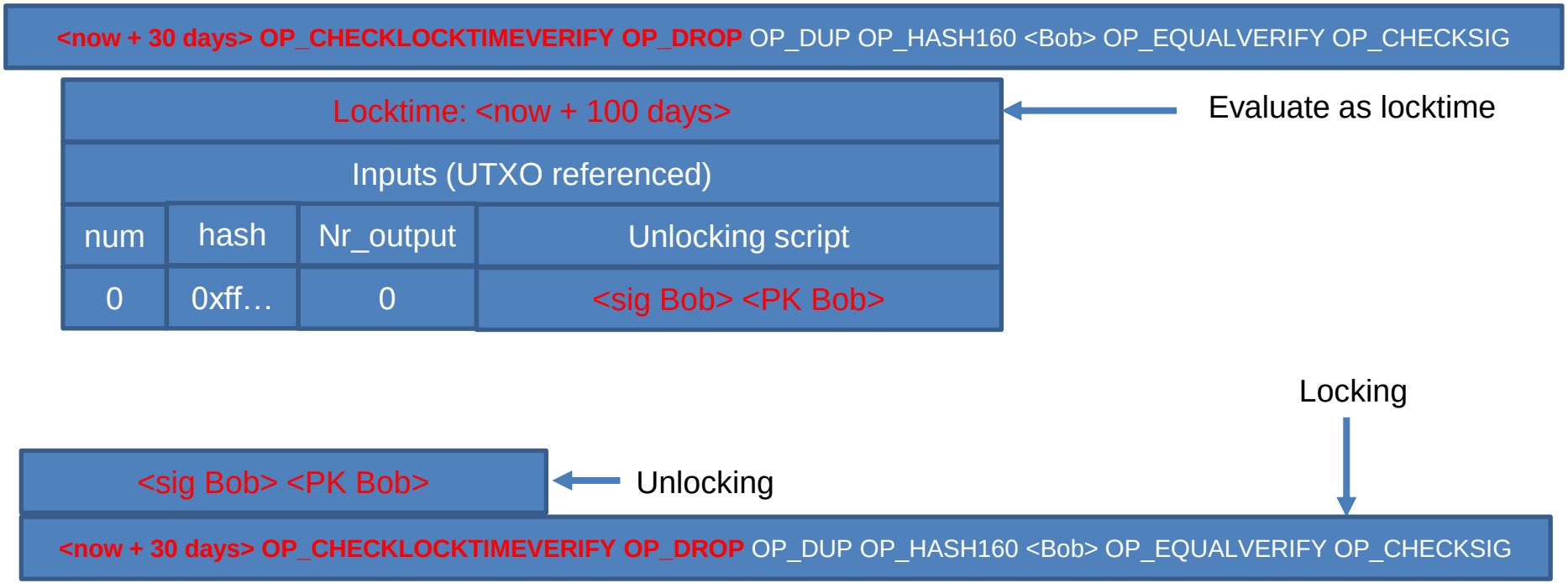
# CLTV validation

UTXO(Tx: 0xff, Nr_out: 0):

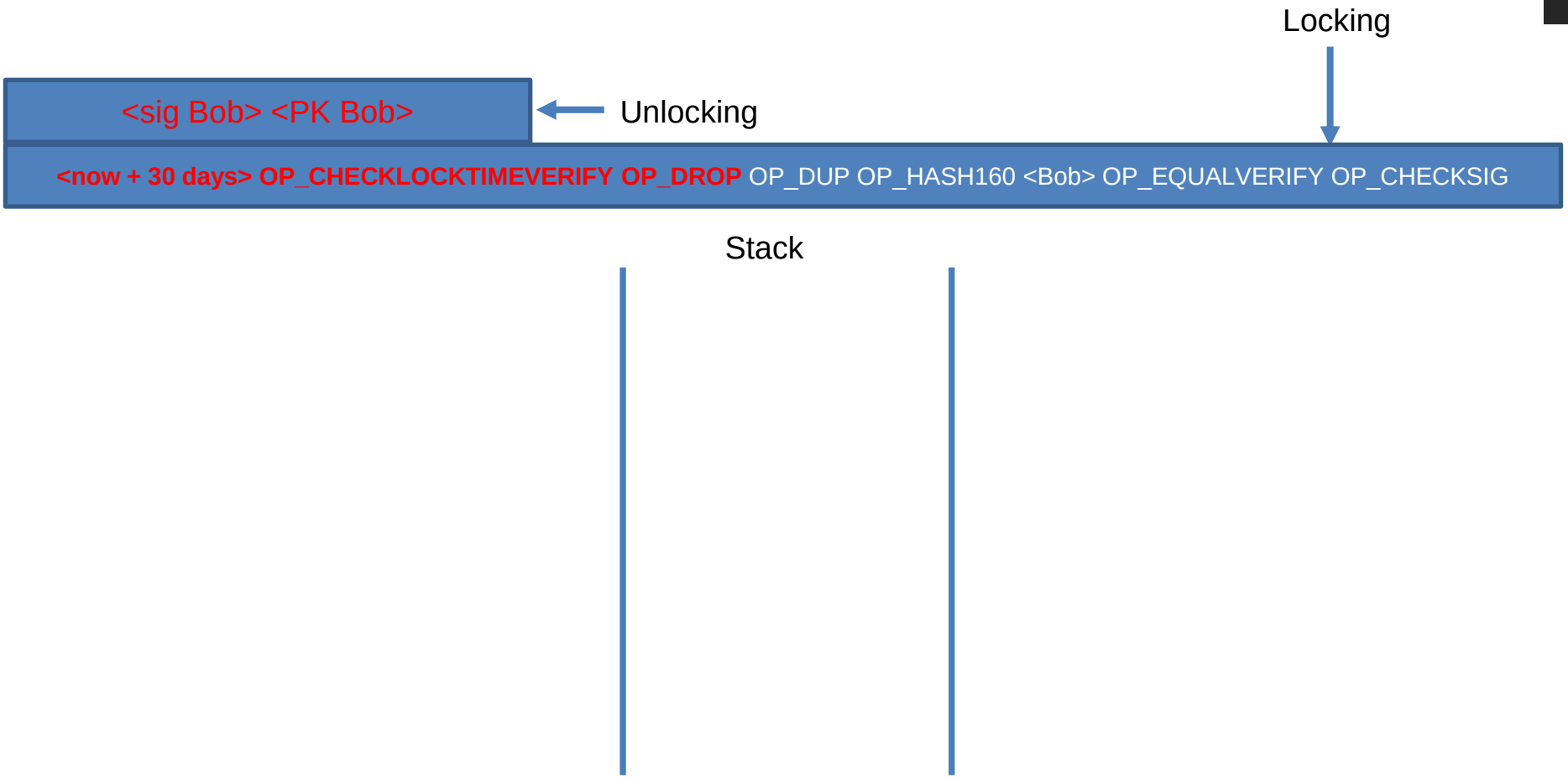


# CLTV validation

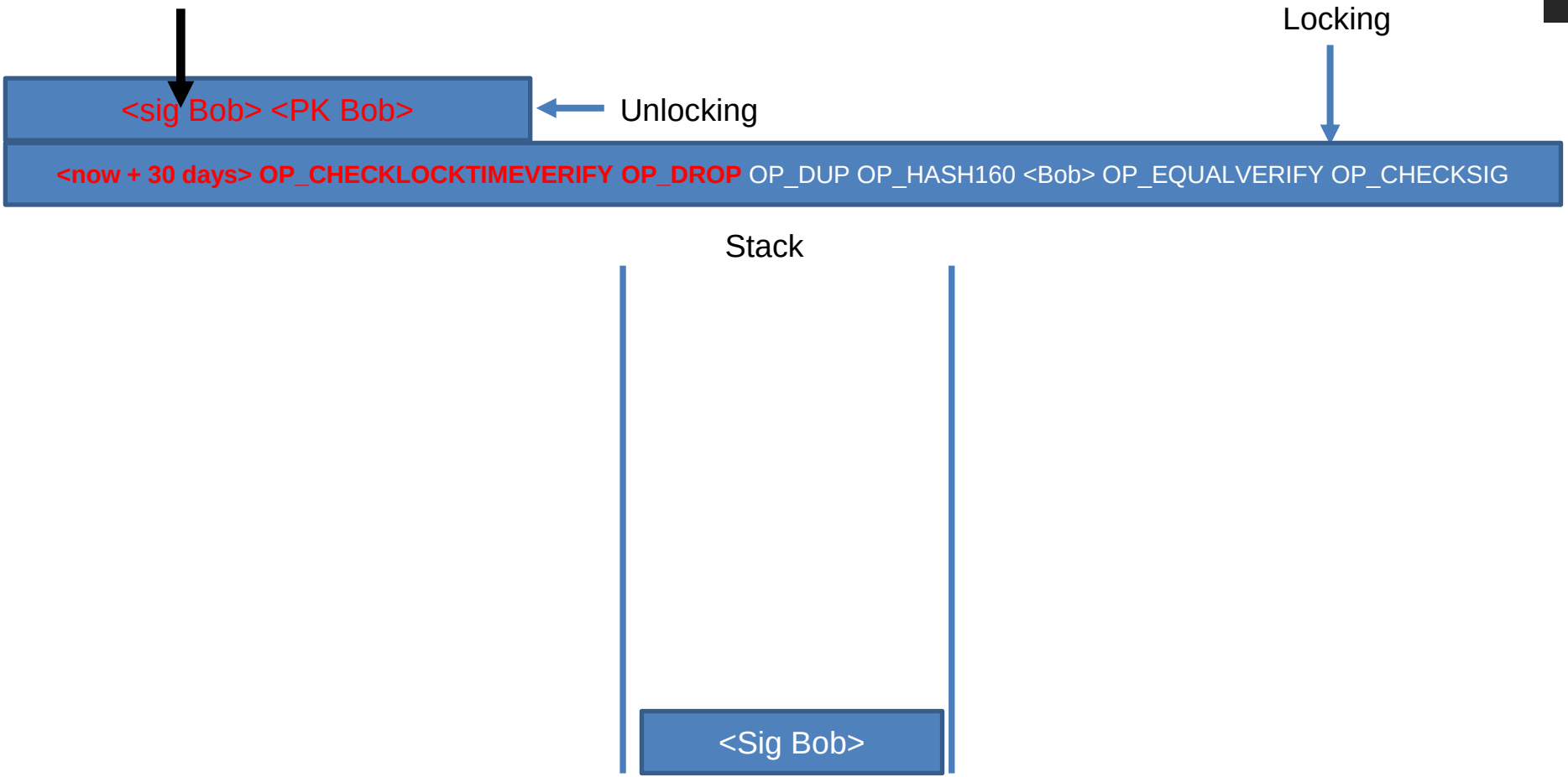
UTXO(Tx: 0xff, Nr_out: 0):



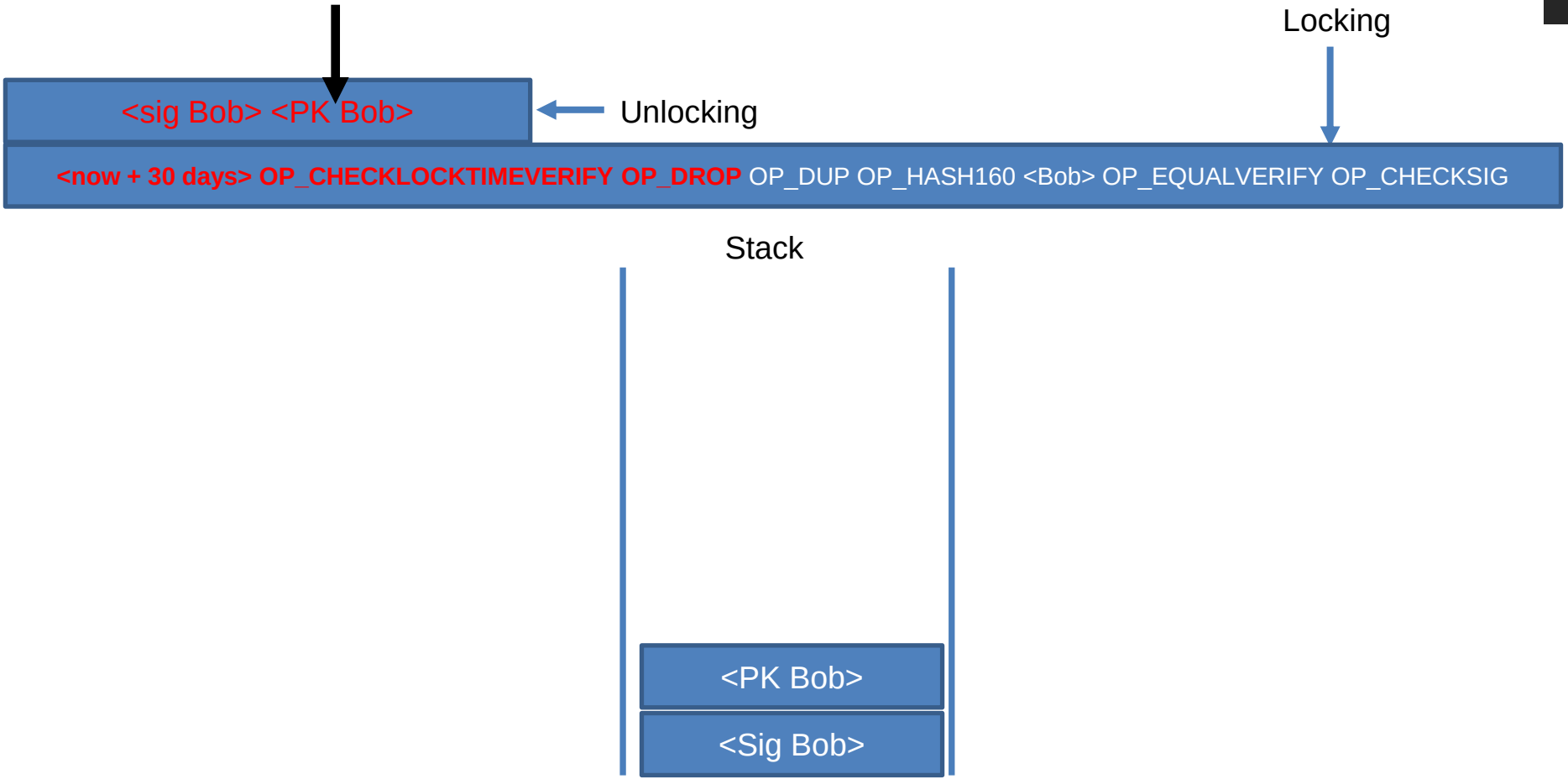
# CLTV validation



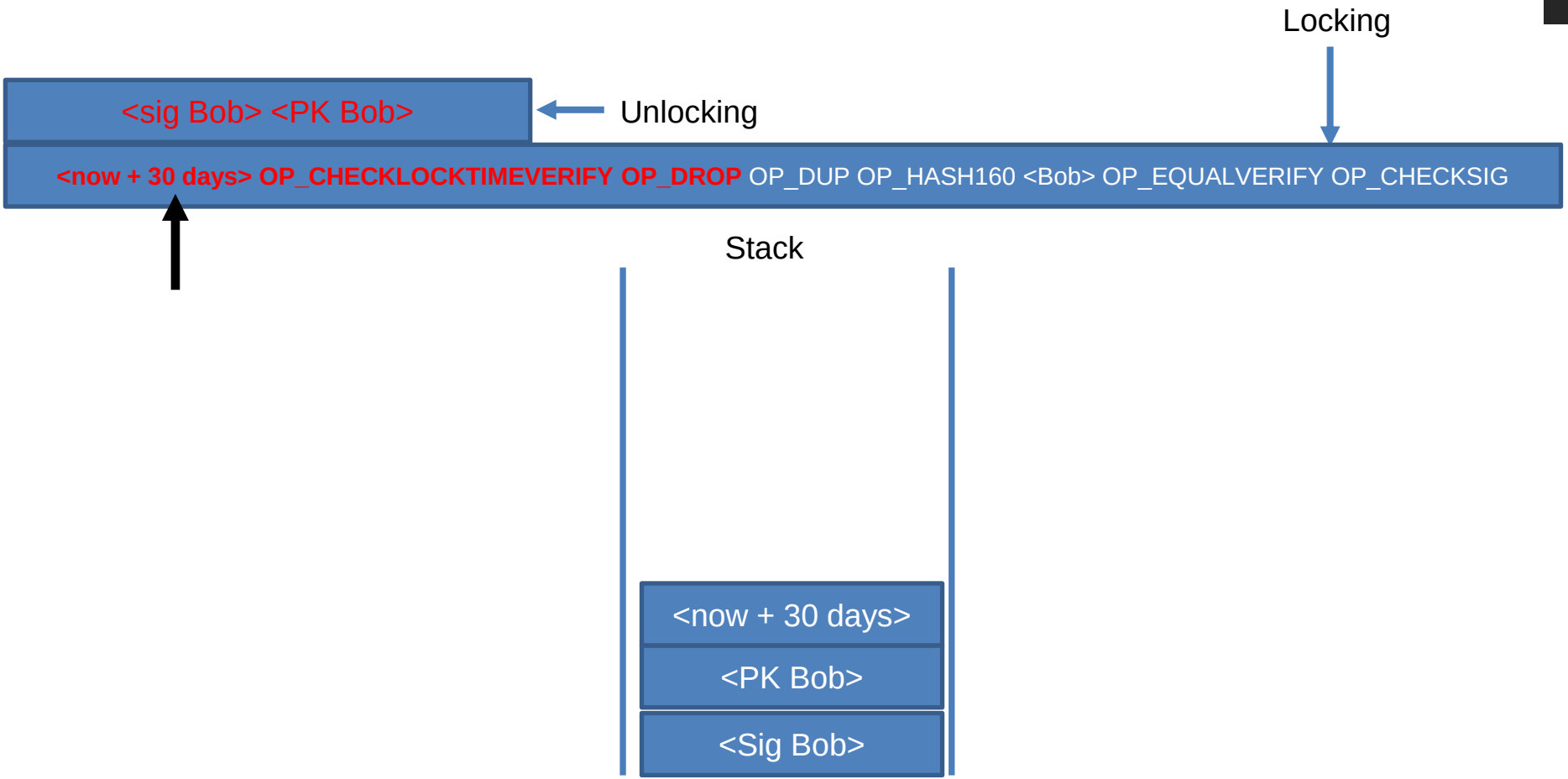
# CLTV validation



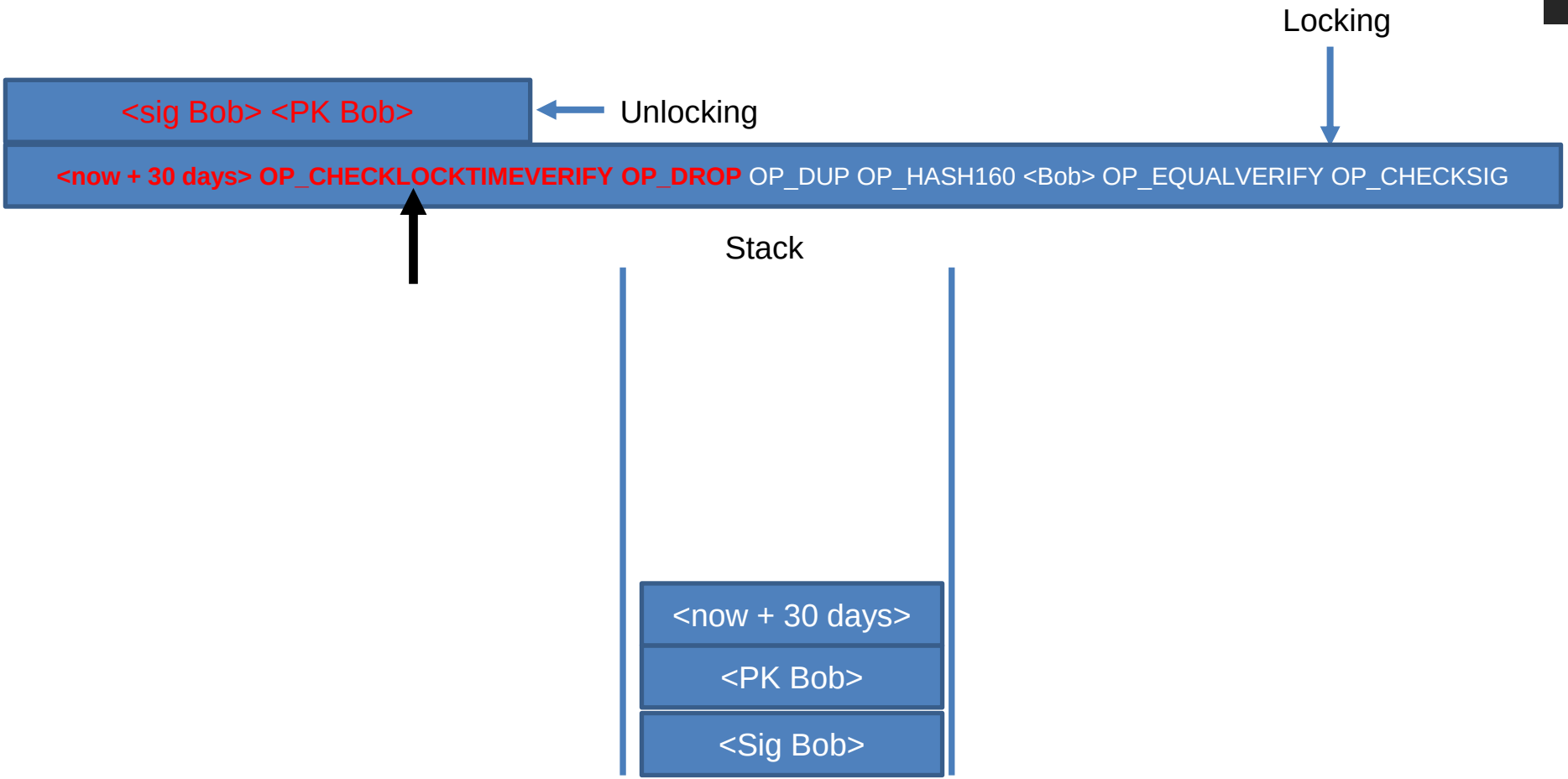
# CLTV validation



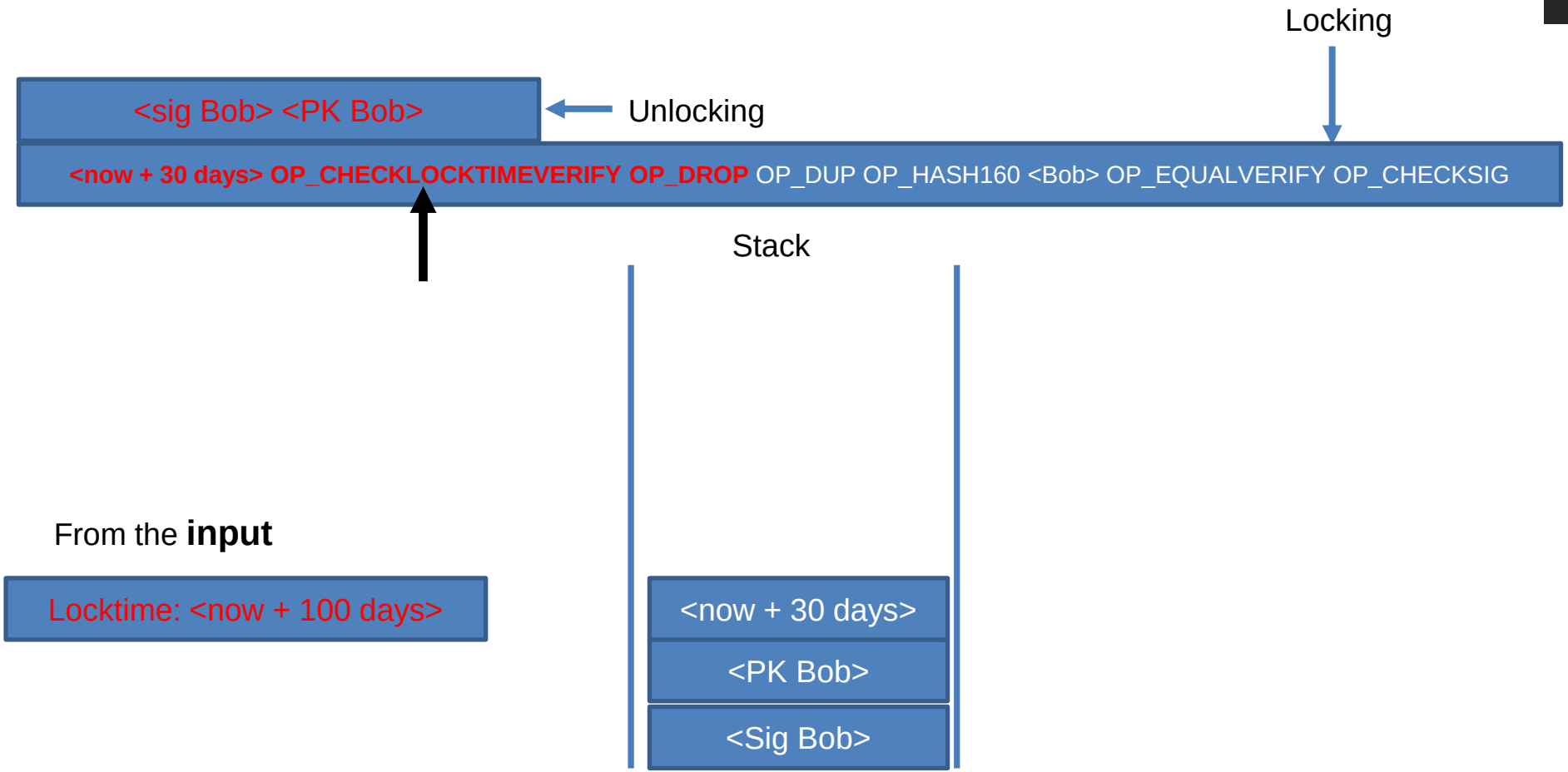
# CLTV validation



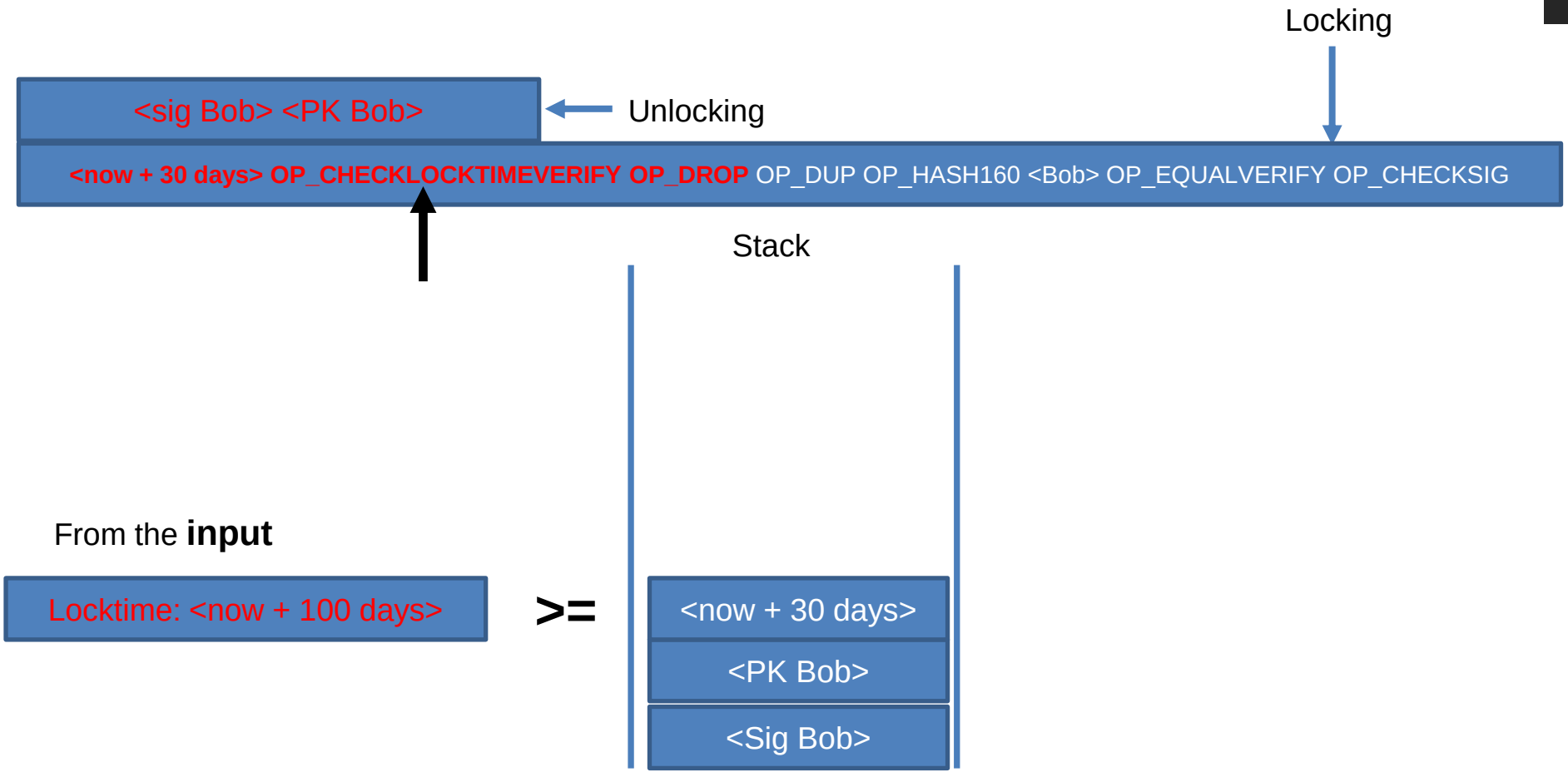
# CLTV validation



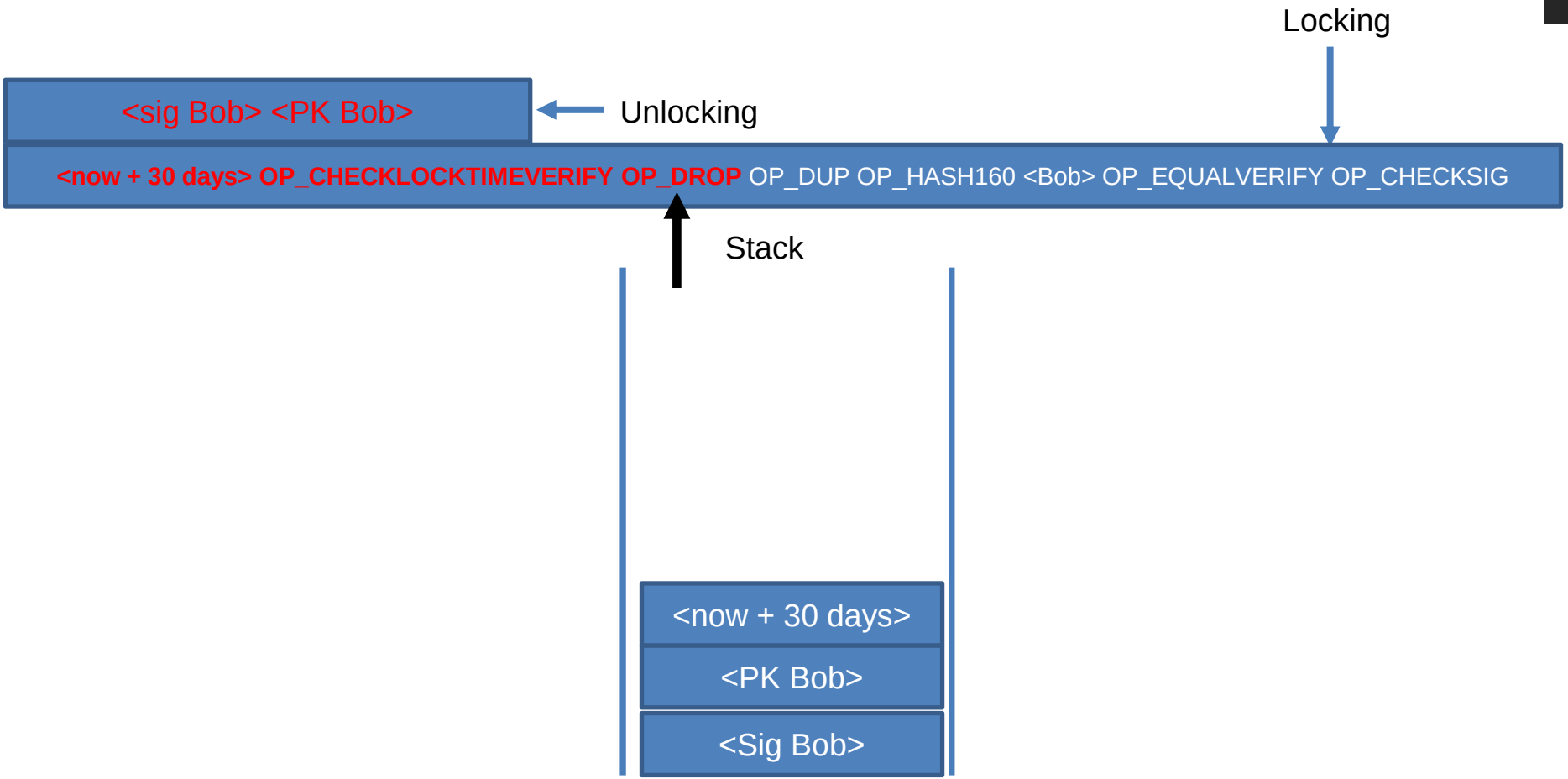
# CLTV validation



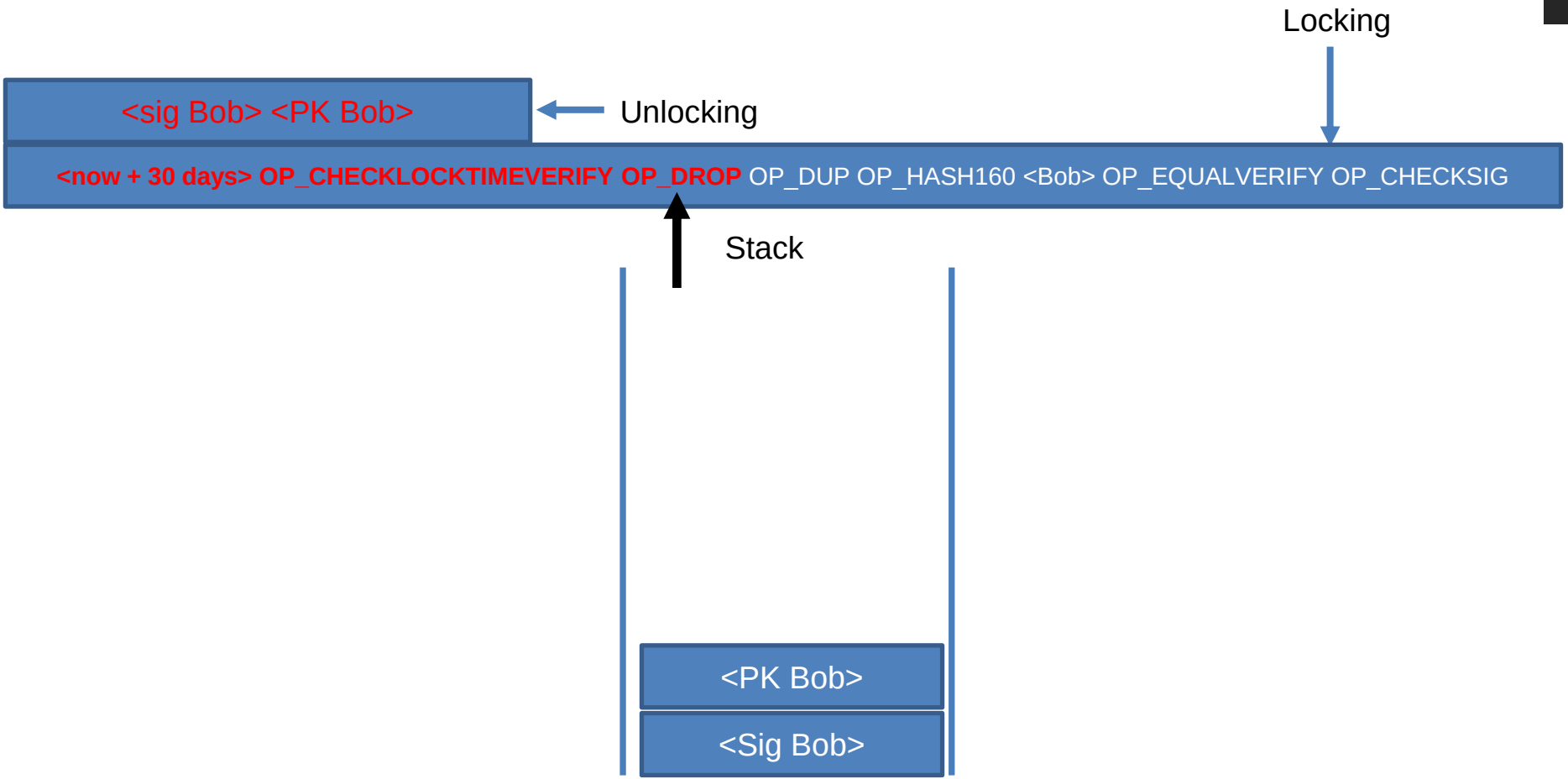
# CLTV validation



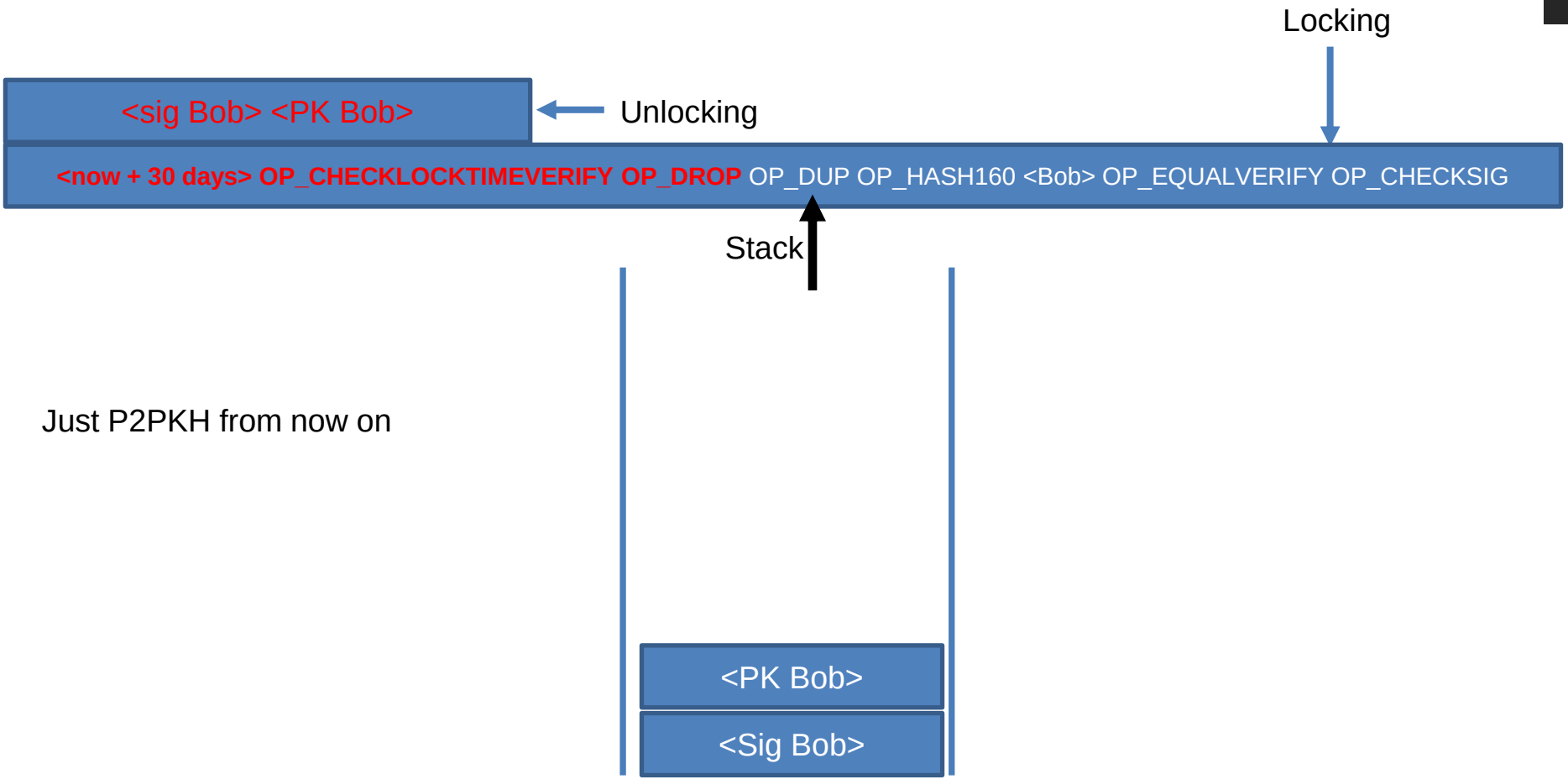
# CLTV validation



# CLTV validation



# CLTV validation




Just P2PKH from now on

# Timelocks 3


```
{
    "txid": "7cf0883a4a24664a1c77fdd465782f0d7c962b7942daa4408f3083ea659aef6a",
    "version": 2,
    "locktime": 500114,
    "vin": [
        {
            "txid": "42a3fdd7d7baea12221f259f38549930b47cec288b55e4a8facc3c899f4775da",
            "vout": 0,
            "scriptSig": "473044022048d1468895910edafe53d4ec4209192cc3a8f0f21e7b9811f83b5e419bfb57
            e002203fef249b56682dbbb1528d4338969abb14583858488a3a766f609185efe68bca0121031a455dab5e
            1f614e574a2f4f12f22990717e93899695fb0d81e4ac2dcfd25d00",
            "sequence": 4
        }
    ],
    "vout": [
        {
            "value": 0.00990000,
            "n": 0,
            "scriptPubKey": "OP_HASH160 e9c3dd0c07aac76179ebc76a6c78d4d67c6c160a OP_EQUAL",

        }
    ]
}
```

# Timelocks 3 (CSV)

**To have a relative timelock BIP112 introduces:**

- OP_CHECKSEQUENCEVERIFY (OP_CSV)
- We can use locktime at the level of UTXOs/Input
- By using sequence

- One for each input

# Timelocks 3


```
{
    "txid": "7cf0883a4a24664a1c77fdd465782f0d7c962b7942daa4408f3083ea659aef6a",
    "version": 2,
    "locktime": 500114,
    "vin": [
        {
            "txid": "42a3fdd7d7baea12221f259f38549930b47cec288b55e4a8facc3c899f4775da",
            "vout": 0,
            "scriptSig": "473044022048d1468895910edafe53d4ec4209192cc3a8f0f21e7b9811f83b5e419bfb57
            e002203fef249b56682dbbb1528d4338969abb14583858488a3a766f609185efe68bca0121031a455dab5e
            1f614e574a2f4f12f22990717e93899695fb0d81e4ac2dcfd25d00",
            "sequence": 4
        }
    ],
    "vout": [
        {
            "value": 0.00990000,
            "n": 0,
            "scriptPubKey": "OP_HASH160 e9c3dd0c07aac76179ebc76a6c78d4d67c6c160a OP_EQUAL",

        }
    ]
}
```


Can be spent only if the UTXO has 4 confirmations!

# Timelocks 3

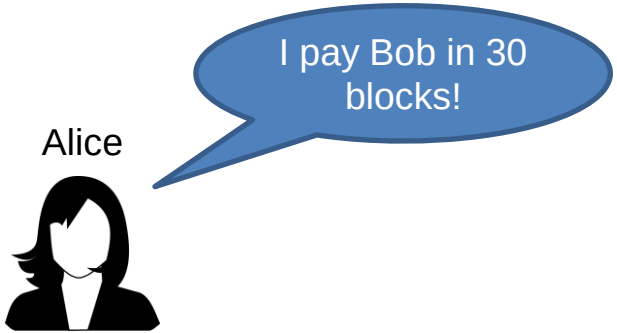

```
{
    "txid": "7cf0883a4a24664a1c77fdd465782f0d7c962b7942daa4408f3083ea659aef6a",
    "version": 2,
    "locktime": 500114,
    "vin": [
        {
            "txid": "42a3fdd7d7baea12221f259f38549930b47cec288b55e4a8facc3c899f4775da",
            "vout": 0,
            "scriptSig": "473044022048d1468895910edafe53d4ec4209192cc3a8f0f21e7b9811f83b5e419bfb57
            e002203fef249b56682dbbb1528d4338969abb14583858488a3a766f609185efe68bca0121031a455dab5e
            1f614e574a2f4f12f22990717e93899695fb0d81e4ac2dcfd25d00",
            "sequence": 4
        }
    ],
    "vout": [
        {
            "value": 0.00990000,
            "n": 0,
            "scriptPubKey": "OP_HASH160 e9c3dd0c07aac76179ebc76a6c78d4d67c6c160a OP_EQUAL",

        }
    ]
}
```


If not, the transaction is not valid

Can be spent only if the UTXO has 4 confirmations!

# CSV




| Inputs (UTXO referenced) | | | |
|---|---|---|---|
| num | hash | Nr_output | Unlocking script |
| 0 | 0xff… | 4 | 012123… |

| Outputs | | |
|---|---|---|
| num | value | Locking script |
| 0 | 3.1 | |

**<30 blocks> OP_CHECKSEQUENCEVERIFY OP_DROP** OP_DUP OP_HASH160 <Bob> OP_EQUALVERIFY OP_CHECKSIG

# CSV

UTXO(Tx: 0xff, Nr_out: 0):

<30 blocks> OP_CHECKSEQUENCEVERIFY OP_DROP OP_DUP OP_HASH160 <Bob> OP_EQUALVERIFY OP_CHECKSIG

Bob

| Inputs (UTXO referenced) | | | |
|---|---|---|---|
| num | hash | Nr_output | Unlocking script |
| 0 | 0xff… | 0 | <sig Bob> <PK Bob> |
| **Outputs** | | | |
| num | value | Locking script | |
| 0 | 3.1 | OP_DUP OP_HASH... | |

# CSV

**<30 blocks> OP_CHECKSEQUENCEVERIFY OP_DROP** OP_DUP OP_HASH160 <Bob> OP_EQUALVERIFY OP_CHECKSIG

| Inputs (UTXO referenced) | | | | |
|---|---|---|---|---|
| num | hash | Nr_output | Unlocking script | sequence |
| 0 | 0xff… | 0 | <sig Bob> <PK Bob> | 41 |

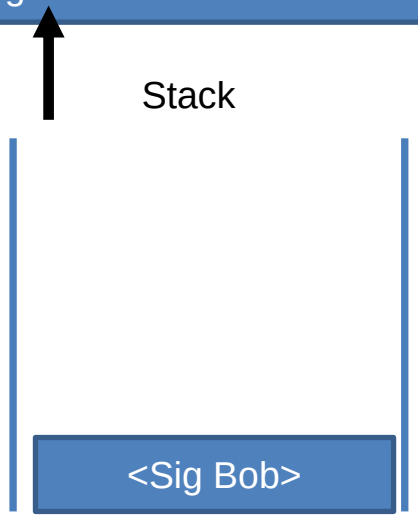
Valid?

Stack

# CSV

**<30 blocks> OP_CHECKSEQUENCEVERIFY OP_DROP** OP_DUP OP_HASH160 <Bob> OP_EQUALVERIFY OP_CHECKSIG

| Inputs (UTXO referenced) | | | | |
|---|---|---|---|---|
| num | hash | Nr_output | Unlocking script | sequence |
| 0 | 0xff… | 0 | <sig Bob> <PK Bob> | 41 |

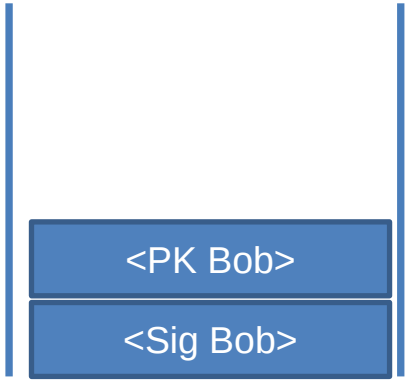


# CSV

**<30 blocks> OP_CHECKSEQUENCEVERIFY OP_DROP** OP_DUP OP_HASH160 <Bob> OP_EQUALVERIFY OP_CHECKSIG

| Inputs (UTXO referenced) | | | | |
|---|---|---|---|---|
| num | hash | Nr_output | Unlocking script | sequence |
| 0 | 0xff… | 0 | <sig Bob> <PK Bob> | 41 |

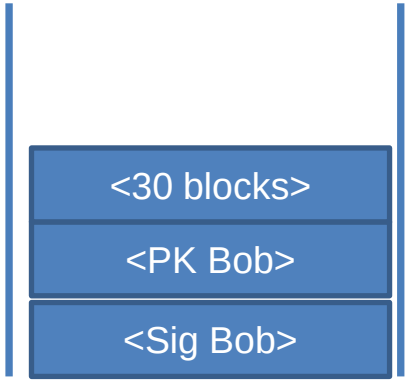
Stack



# CSV


| Inputs (UTXO referenced) | | | | |
|---|---|---|---|---|
| num | hash | Nr_output | Unlocking script | sequence |
| 0 | 0xff… | 0 | <sig Bob> <PK Bob> | 41 |

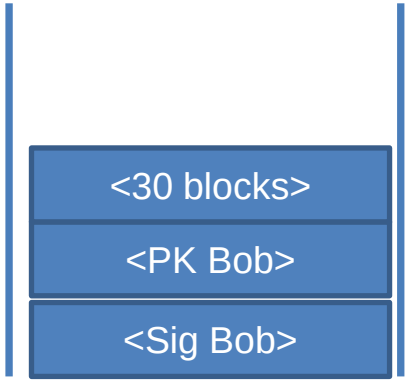
Stack



# CSV

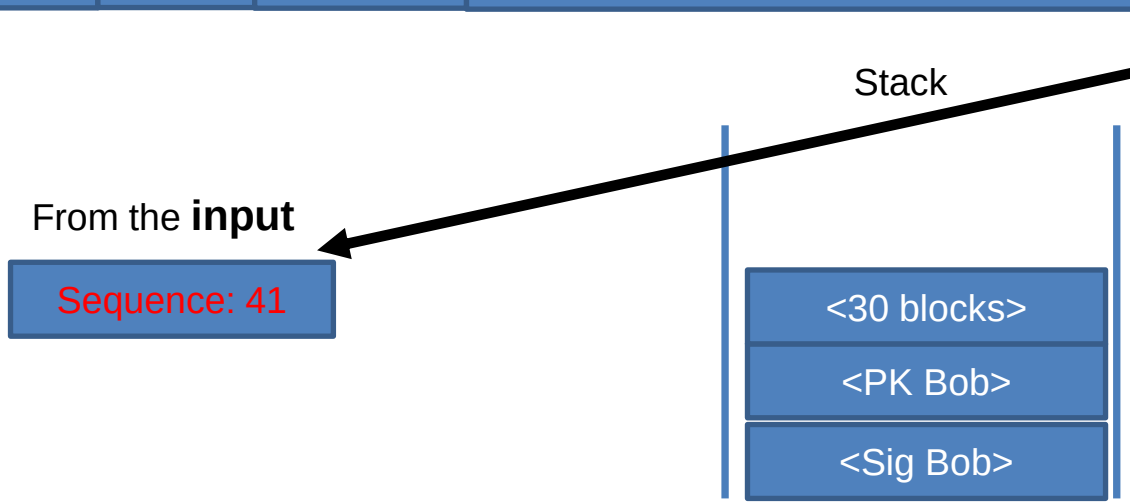

| Inputs (UTXO referenced) | | | | |
|---|---|---|---|---|
| num | hash | Nr_output | Unlocking script | sequence |
| 0 | 0xff… | 0 | <sig Bob> <PK Bob> | 41 |

Stack



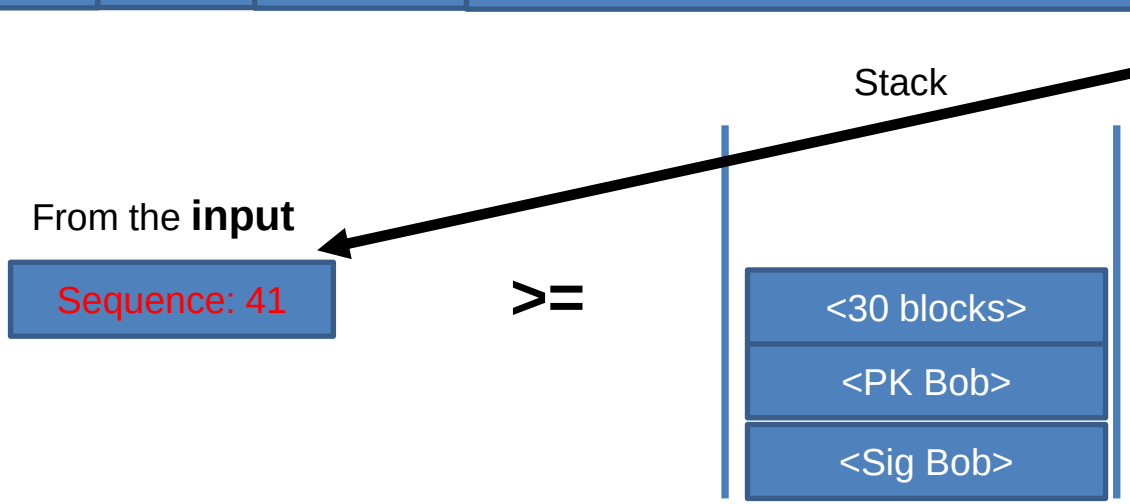
# CSV


| Inputs (UTXO referenced) | | | | |
|---|---|---|---|---|
| num | hash | Nr_output | Unlocking script | sequence |
| 0 | 0xff… | 0 | <sig Bob> <PK Bob> | 41 |



# CSV

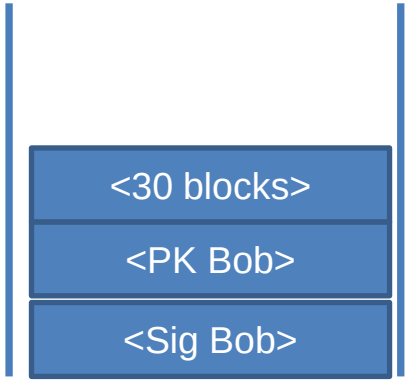

| Inputs (UTXO referenced) | | | | |
|---|---|---|---|---|
| num | hash | Nr_output | Unlocking script | sequence |
| 0 | 0xff… | 0 | <sig Bob> <PK Bob> | 41 |



# CSV


| Inputs (UTXO referenced) | | | | |
|---|---|---|---|---|
| num | hash | Nr_output | Unlocking script | sequence |
| 0 | 0xff… | 0 | <sig Bob> <PK Bob> | 41 |

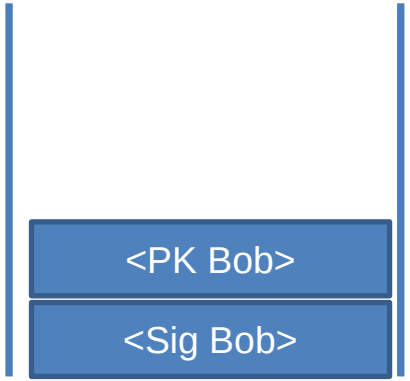
Stack



# CSV


| Inputs (UTXO referenced) | | | | |
|---|---|---|---|---|
| num | hash | Nr_output | Unlocking script | sequence |
| 0 | 0xff… | 0 | <sig Bob> <PK Bob> | 41 |

Stack



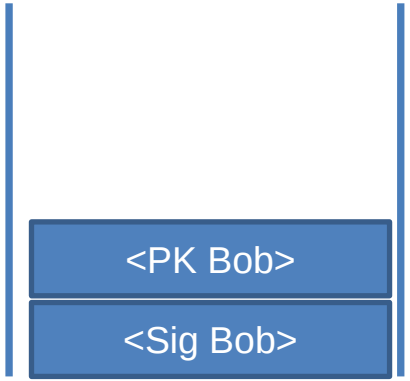
# CSV

<30 blocks> OP_CHECKSEQUENCEVERIFY OP_DROP OP_DUP OP_HASH160 <Bob> OP_EQUALVERIFY OP_CHECKSIG

| Inputs (UTXO referenced) | | | | |
|---|---|---|---|---|
| num | hash | Nr_output | Unlocking script | sequence |
| 0 | 0xff… | 0 | <sig Bob> <PK Bob> | 41 |

P2PKH standard

Stack



# IF ELSE

## For interesting contracts

**Usual code:**

```
if (condition):
    code to run when true
else:
    code to run when false
code to run always
```

**With stack:**

```
(condition)
IF
    code to run when true
ELSE
    code to run when false
ENDIF
code to run always
```

# IF ELSE

## 1-2 MULTISIG

**Locking script:**

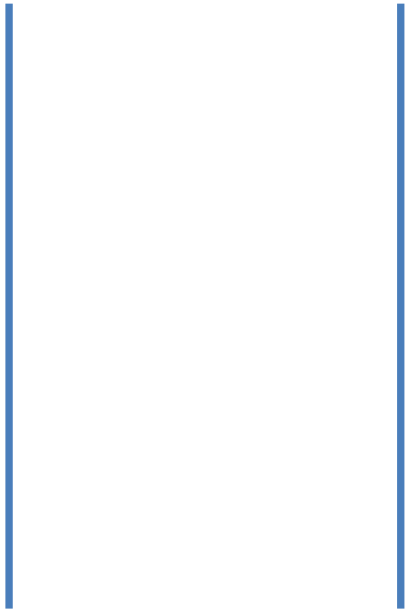
```
IF
    <PK Alice> OP_CHECKSIG
ELSE
    <PK Bob> OP_CHECKSIG
ENDIF
```

## How to activate IF/ELSE?

# Multisig

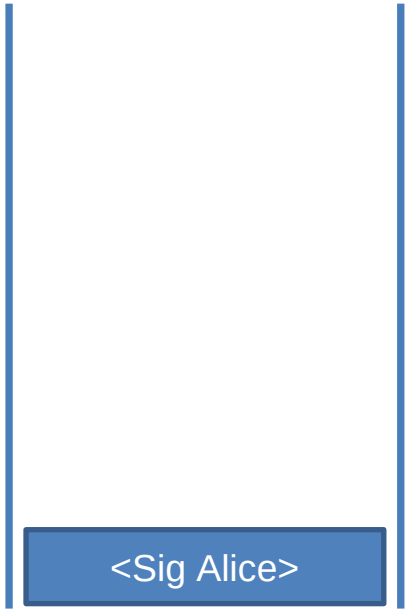
| <sig Alice> 1 | IF <PK Alice> OP_CHECKSIG ELSE <PK Bob> OP_CHECKSIG |
| --- | --- |

Stack



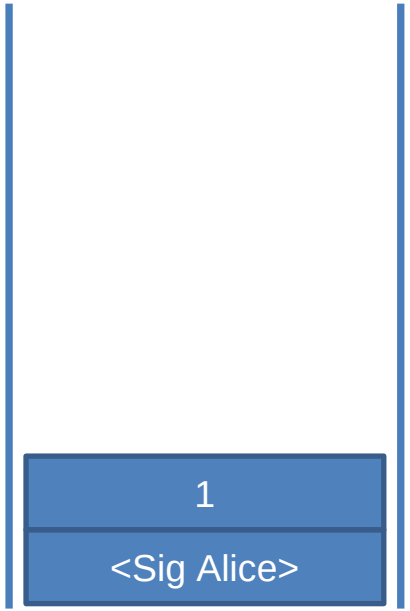
# Multisig


Stack



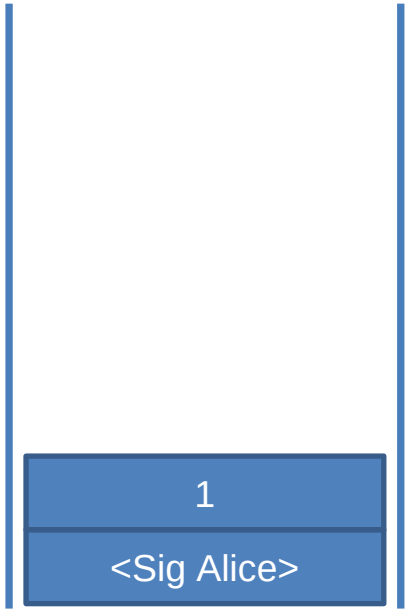
# Multisig


Stack

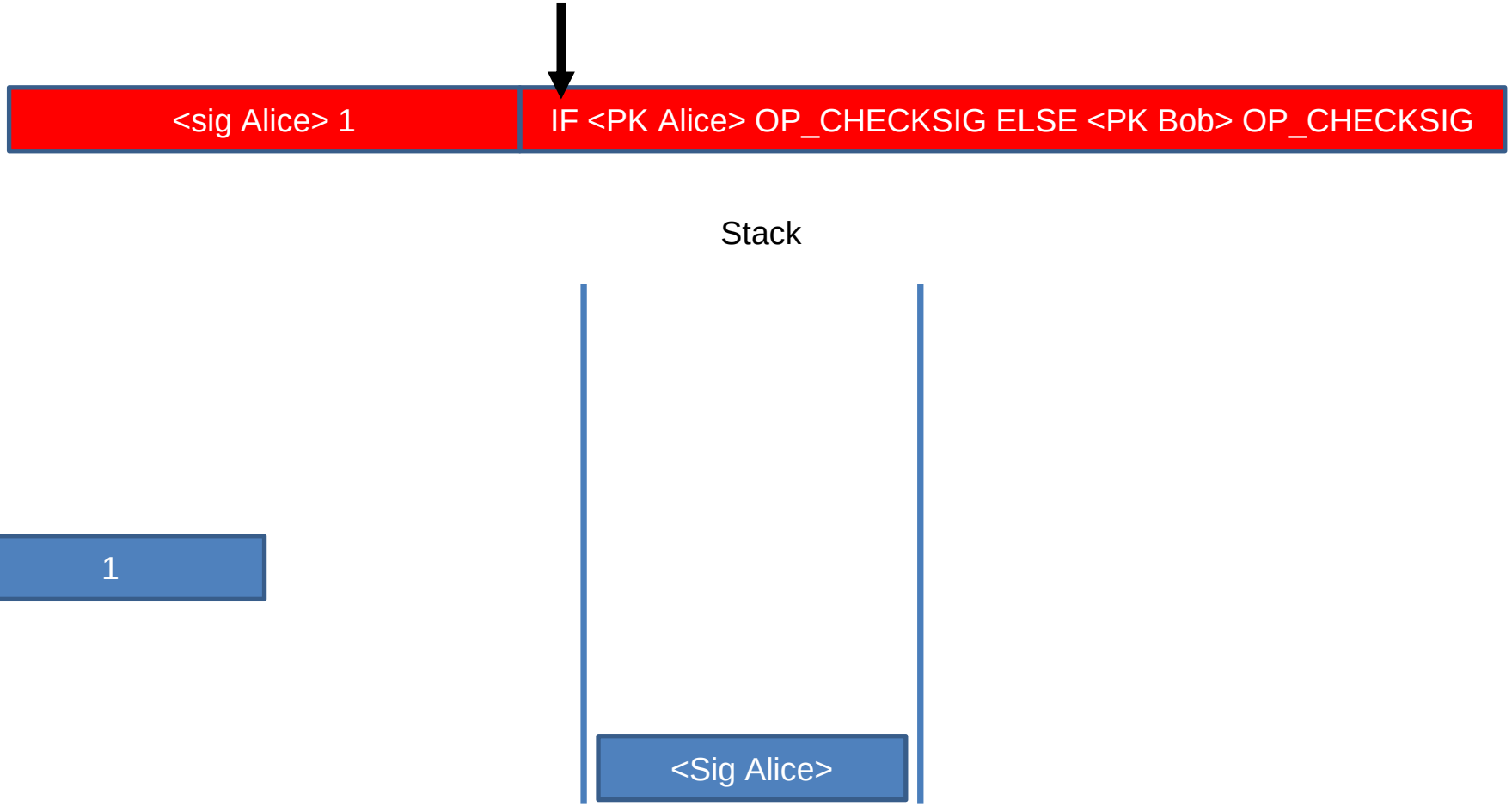


# Multisig


Stack



# Multisig


<sig Alice> 1
IF <PK Alice> OP_CHECKSIG ELSE <PK Bob> OP_CHECKSIG
Stack
1
<Sig Alice>

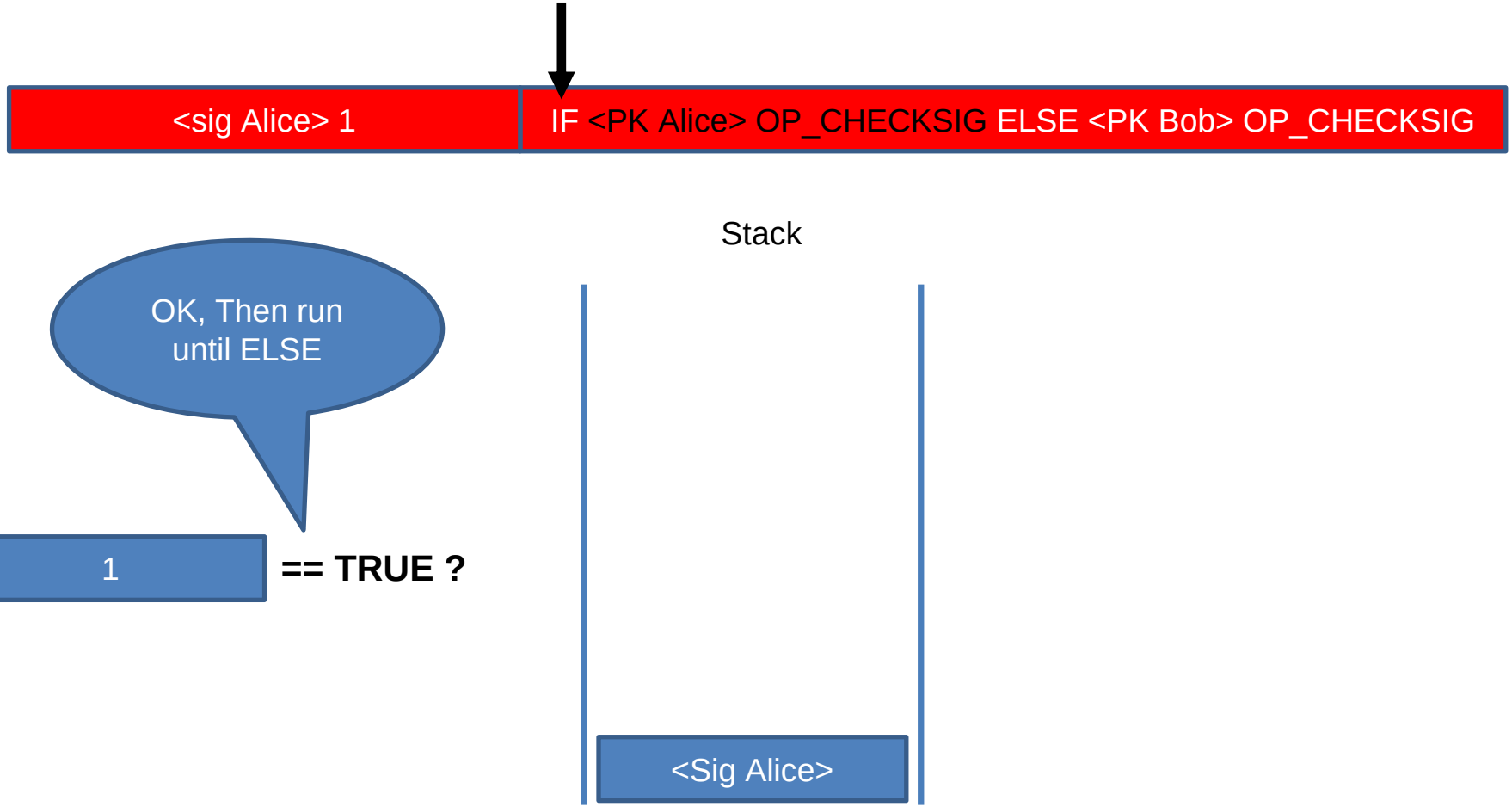
# Multisig


Stack

1 == TRUE ?

<Sig Alice>

# Multisig


<sig Alice> 1
IF <PK Alice> OP_CHECKSIG ELSE <PK Bob> OP_CHECKSIG
Stack
OK, Then run until ELSE
1
== TRUE ?
<Sig Alice>

# Multisig

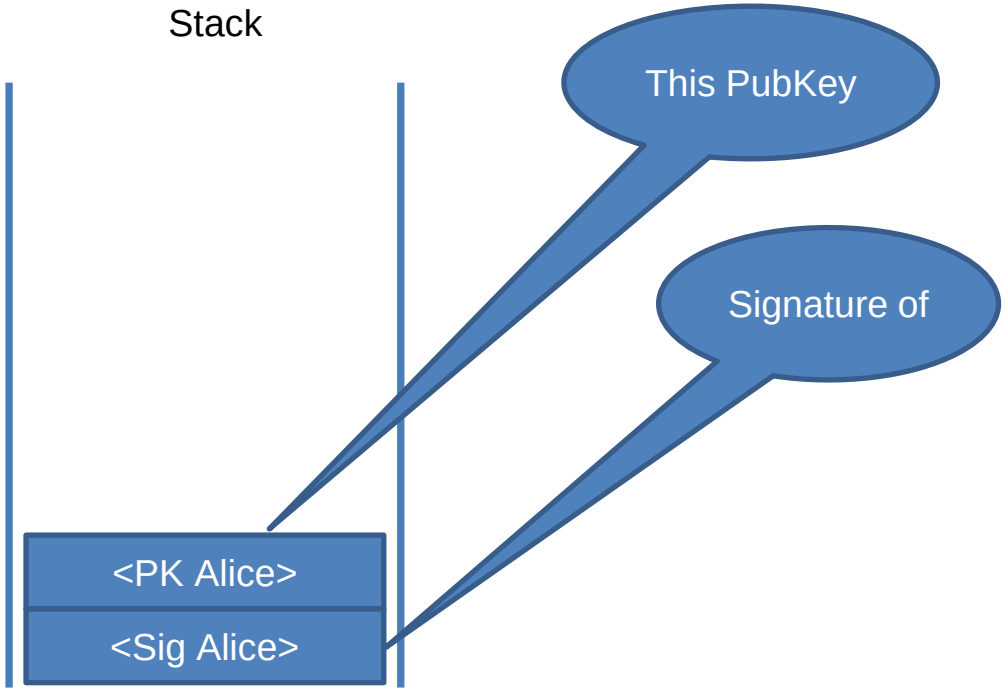

Stack

# Multisig


Stack

# Multisig

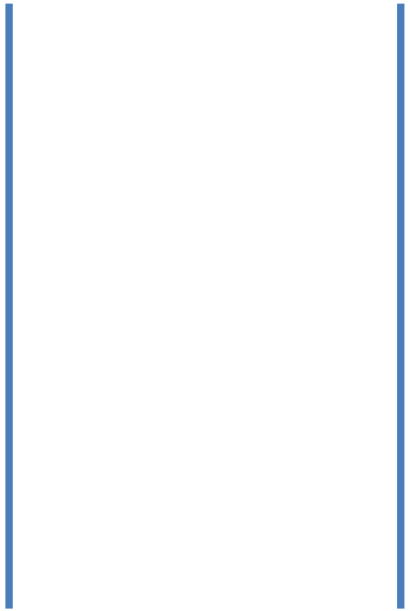

<sig Alice> 1
IF <PK Alice> OP_CHECKSIG ELSE <PK Bob> OP_CHECKSIG


Stack
This PubKey
Signature of
<PK Alice>
<Sig Alice>

# Multisig

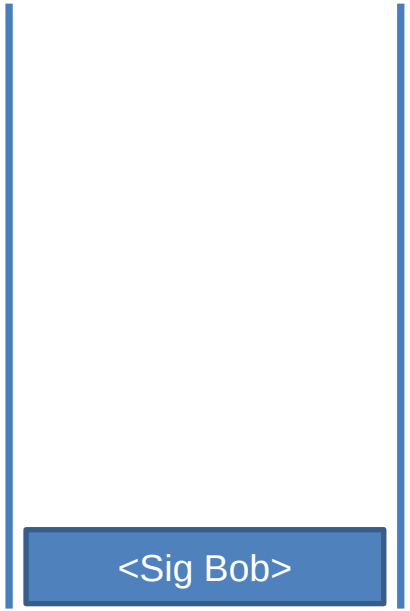
| <sig Bob> 0 | IF <PK Alice> OP_CHECKSIG ELSE <PK Bob> OP_CHECKSIG |
| --- | --- |

Stack



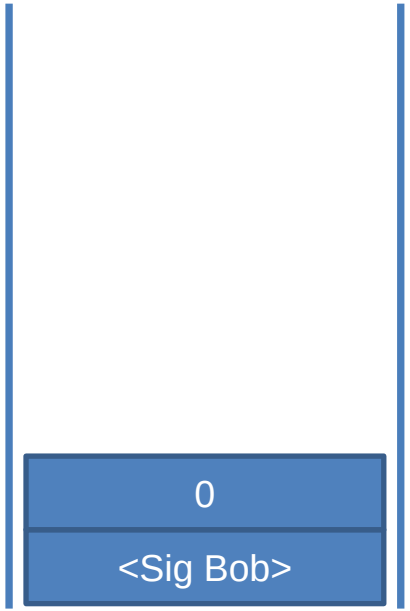
# Multisig


Stack



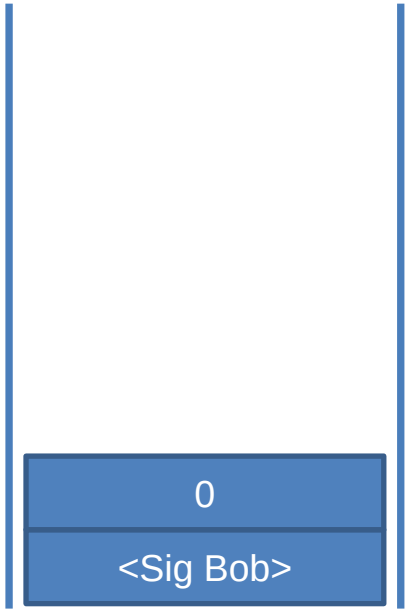
# Multisig


Stack

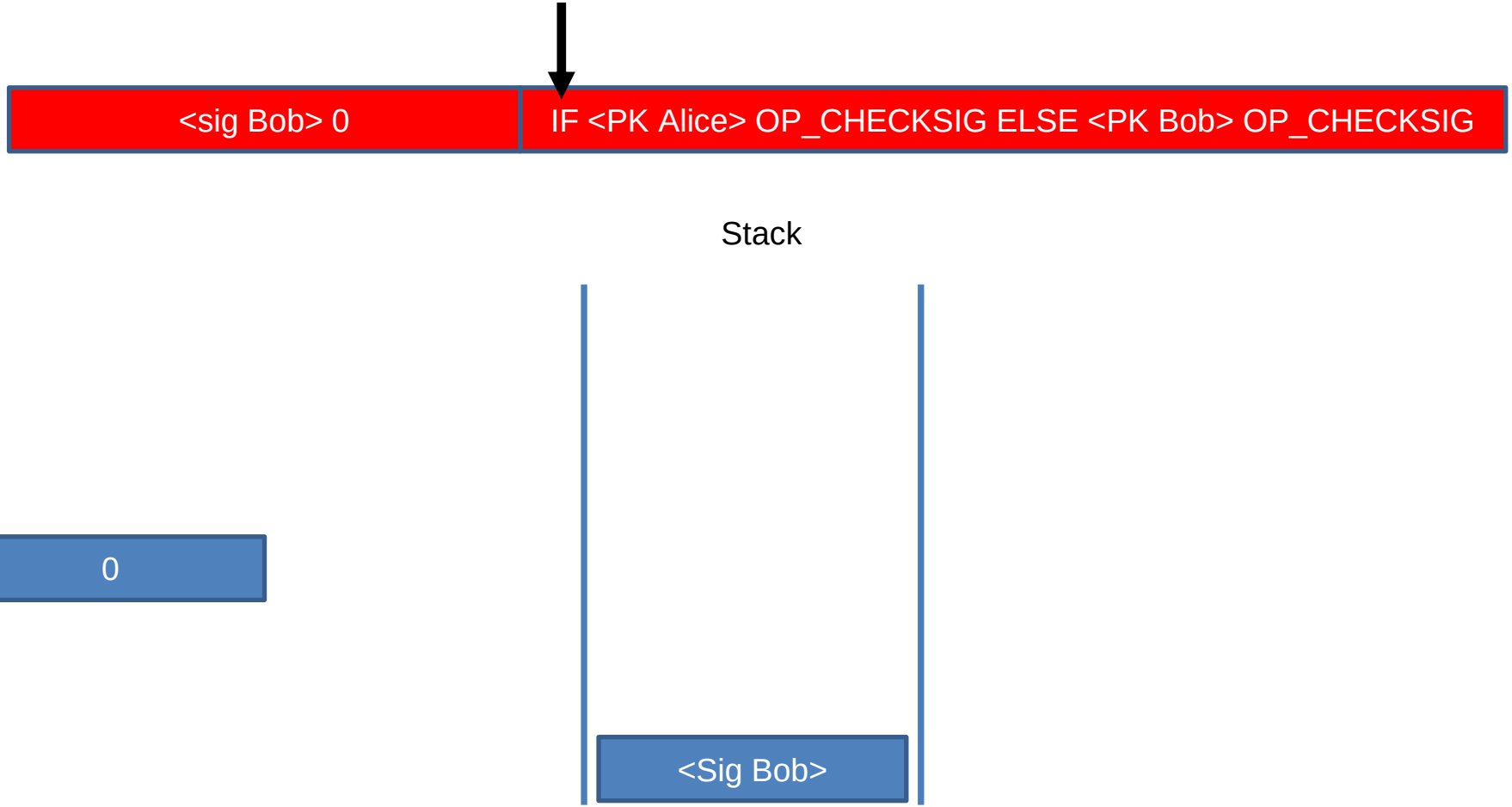


# Multisig


Stack

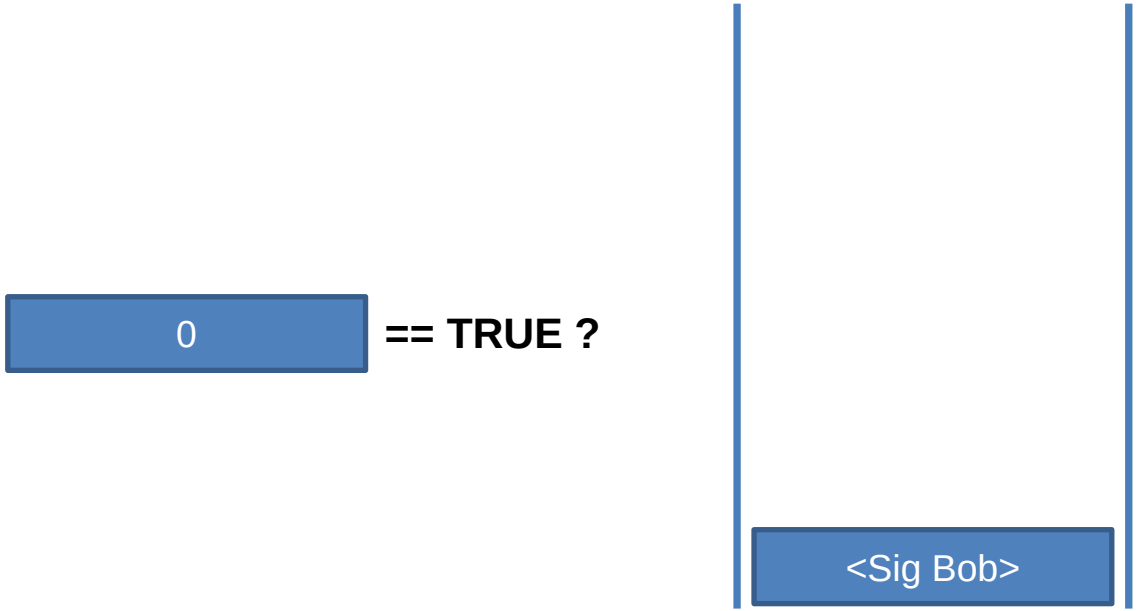


# Multisig

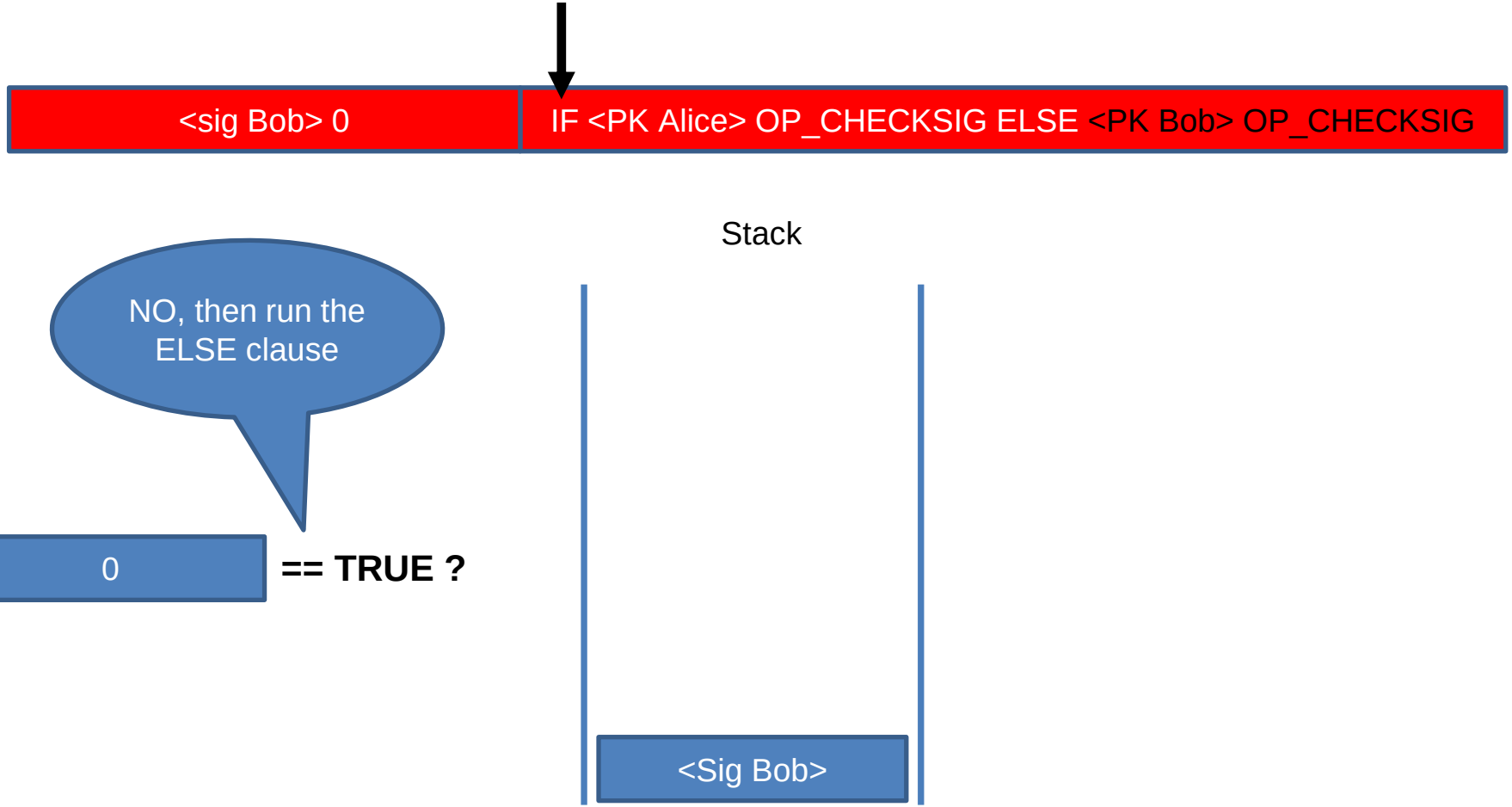

<sig Bob> 0
IF <PK Alice> OP_CHECKSIG ELSE <PK Bob> OP_CHECKSIG
Stack
0
<Sig Bob>

# Multisig


Stack

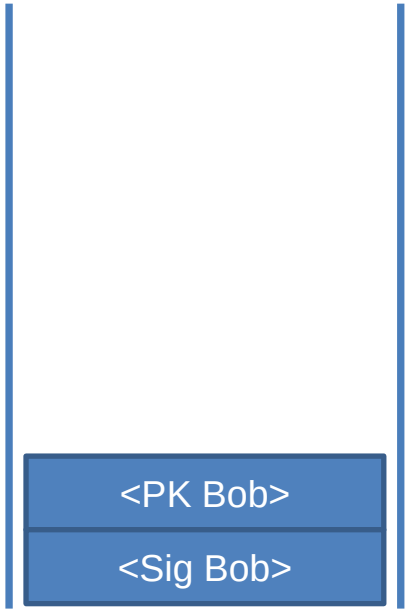


# Multisig


<sig Bob> 0
IF <PK Alice> OP_CHECKSIG ELSE <PK Bob> OP_CHECKSIG
Stack
NO, then run the ELSE clause
0
== TRUE ?
<Sig Bob>

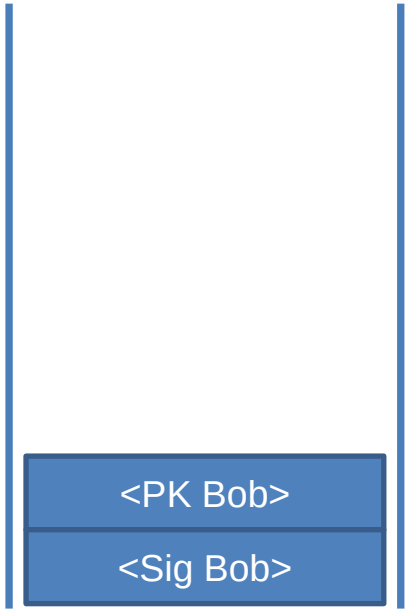
# Multisig


Stack



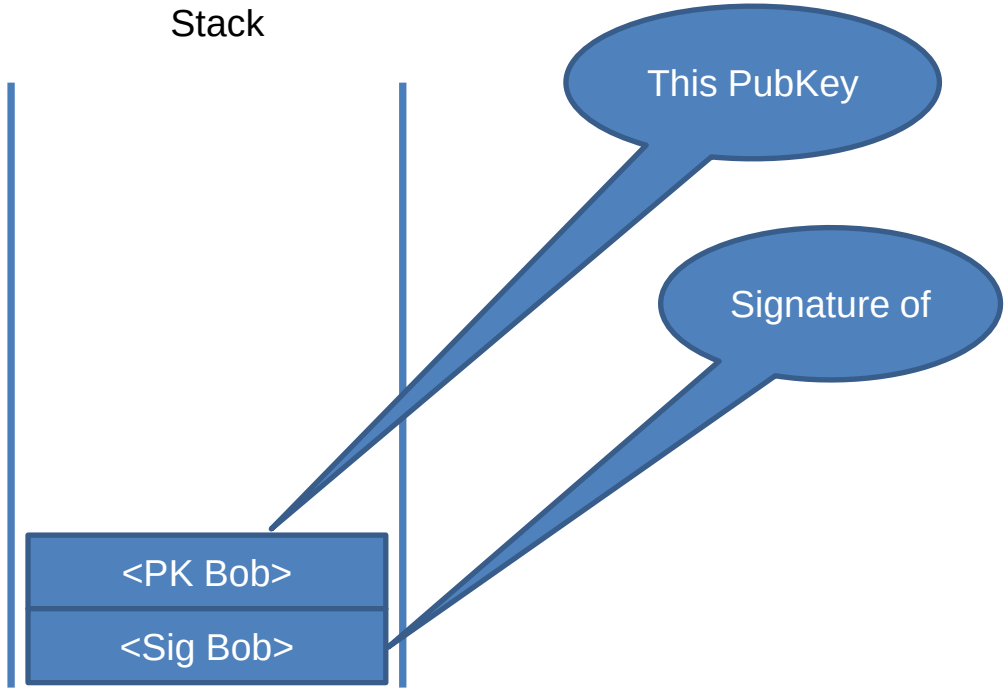
# Multisig


Stack



# Multisig


<sig Bob> 0
IF <PK Alice> OP_CHECKSIG ELSE <PK Bob> OP_CHECKSIG


Stack
This PubKey
Signature of
<PK Bob>
<Sig Bob>

# IF ELSE

**Company with 3 owners, and a lawyer:**

- Alice, Bob y Charlie are partners
- Mohamed is the lawyer
- To spend the funds 2 of 3 partners should be in agreement

If they cannot reach agreemin in 30 days, enters Mohamed:

- To spend the funds we need the signature of Mohamed + one of A,B,C

If Alice, Bob and Charlie lose their keys:

- Mohamed can recuperate the funds in 3 months

# IF ELSE

## Company with 3 owners, and a lawyer

**Locking script:**

```
IF
  IF
    2
  ELSE
    <30 days> OP_CHECKSEQUENCEVERIFY OP_DROP
    <Lawyer's PK> OP_CHECKSIGVERIFY
    1
  ENDIF
  <PK Alice> <PK Bob> <PK Charlie> 3 OP_CHECKMULTISIG
ELSE
    <90 days> OP_CHECKSIGVERIFY OP_DROP
    <Lawyer's PK> OP_CHECKSIG
ENDIF
```

# IF ELSE

**3 paths of execution:**

```
IF
   IF
      2
   ELSE
      <30 days> OP_CHECKSEQUENCEVERIFY OP_DROP
      <Lawyer's PK> OP_CHECKSIGVERIFY
      1
   ENDIF
   <PK Alice> <PK Bob> <PK Charlie> 3 OP_CHECKMULTISIG
ELSE
      <90 days> OP_CHECKSIGVERIFY OP_DROP
      <Lawyer's PK> OP_CHECKSIG
ENDIF
```

# IF ELSE

**3 paths of execution:**

**TRUE TRUE**
(e.g. 0 <SIG A> <SIG C> TRUE TRUE)

```
IF
  IF
    2
  ELSE
    <30 days> OP_CHECKSEQUENCEVERIFY OP_DROP
    <Lawyer's PK> OP_CHECKSIGVERIFY
    1
  ENDIF
  <PK Alice> <PK Bob> <PK Charlie> 3 OP_CHECKMULTISIG
ELSE
    <90 days> OP_CHECKSIGVERIFY OP_DROP
    <Lawyer's PK> OP_CHECKSIG
ENDIF
```

# IF ELSE

## 3 paths of execution:

**FALSE TRUE**
(e.g. 0 <SIG Lawyer> <SIG C> FALSE TRUE)

```
IF
  IF
    2
  ELSE
    <30 days> OP_CHECKSEQUENCEVERIFY OP_DROP
    <Lawyer's PK> OP_CHECKSIGVERIFY
    1
  ENDIF
  <PK Alice> <PK Bob> <PK Charlie> 3 OP_CHECKMULTISIG
ELSE
    <90 days> OP_CHECKSIGVERIFY OP_DROP
    <Lawyer's PK> OP_CHECKSIG
ENDIF
```

# IF ELSE

**Some questions:**

- The lawyer cannot steal the funds at any moment (FALSE)?
- Which execution paths can be executed at 5, 35, 105 days after thecontract transaction had been confirmed?
- Are the funds lost if the lawyer loses the private key?
- How can the partners do a reset each 29 and 89 days so that the lawyer cannot steal teh funds?

# Payment channels

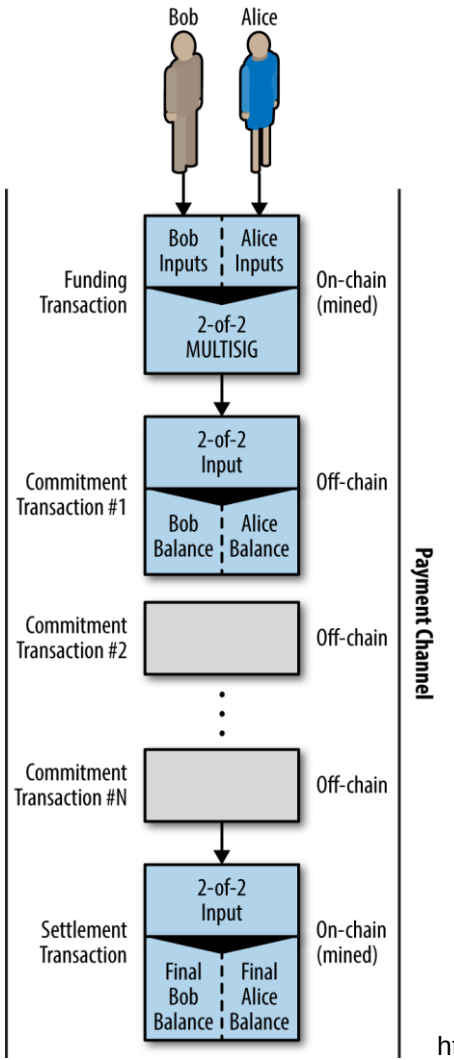
**A mechanisms for doing transactions off-chain:**

- Instant (zero confirm) transactions

**Two parties wich to know how much money each has in the interaction:**

- Funding transaction (initial balance)
- A series fo commitment transactions (off-chain)
- Settlement transaction (final balance)

# Payment channels




https://github.com/bitcoinbook/bitcoinbook/blob/develop/ch12.asciidoc

# A simple channel

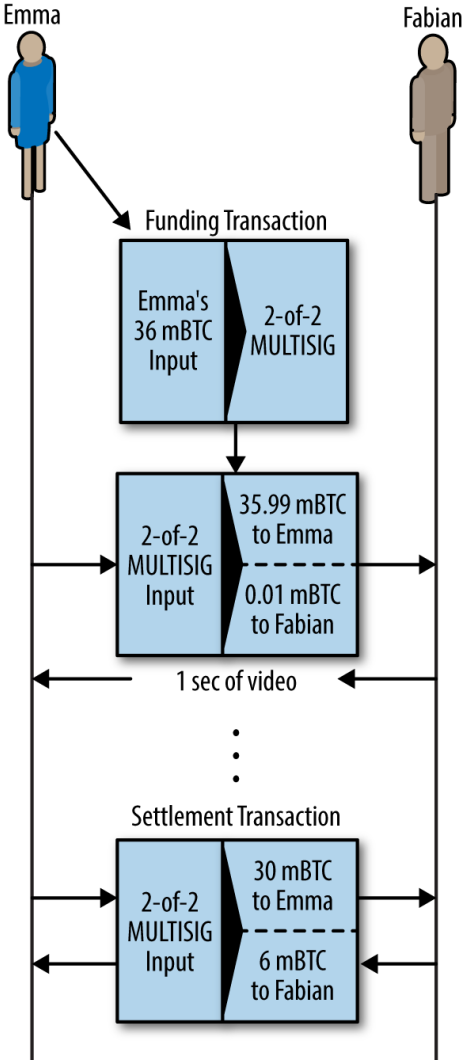
## Streaming

**A streaming application:**

- Fabian offers a streaming service and charges per second of stream
- Each second costs 0.01mBTC = 0.00001 BTC
- Emma wishes to see X minutes of a video
- They have a specialized software that allows this using payment channels

# A simple channel

## Streaming




https://github.com/bitcoinbook/bitcoinbook/blob/develop/ch12.asciidoc

# A simple channel

## Streaming

- Emma sends commit#k with her signature (Fabian's sig is missing to spend)
- Fabian streams 1 sec of the video + commit#k with his sig (can spend now)

**Points:**

- Only one commit can be spent (all commits use the same output as input)
- Emma receives transaction n-1 signed; if she spends it: Fabian stops stream!
- For him, it is best to publish the latest one

# A simple channel

Streaming

**Problems:**

- What happens if Fabian escapes before publishing the first commitment?
- What happens if Emma transmits commitment #1?

**We can solve these two problems with timelocks**

# A simple channel

## Timelocked

**Funding transaction:**

- Emma constructs **funding** transaction, signs it, but doesn't send it to Fabian
- Emma constructs the **refund** transaction with timelock X (e.g. now + 30 days)
- Emma sends the refund to Fabian to sign

- Once she has the refund transaction, Emma transmits the funding one
- Say that funding has timelock T + 4320 blocks (30 days)

- If Fabian dissappears before Commit #1, Emma can recover the funds using refund

Idea: funding can actually be an IF/ELSE with a timelock refund to Emma

# A simple channel

## Timelocked

**Commitment transactions:**

- (Funding: Emma pays 36mBTC a MULTISIG output of Emma and Fabian)
- Commit #1 pays 0.01mBTC to Fabian, and 35.99mBTC to Emma
- Commit #1 has timelock T + 4320 blocks

- Commit #2: F->0.02; E->35.98; T + 4319 blocks
- Commit #3: F->0.03; E->35.97; T + 4318 blocks
- Commit #4: F->0.04; E->35.96; T + 4317 blocks

# A simple channel

## Timelocked

**Commitment transactions:**

- Commit #2: F->0.02; E->35.98; T + 4319 blocks
- Commit #3: F->0.03; E->35.97; T + 4318 blocks
- Commit #4: F->0.04; E->35.96; T + 4317 blocks

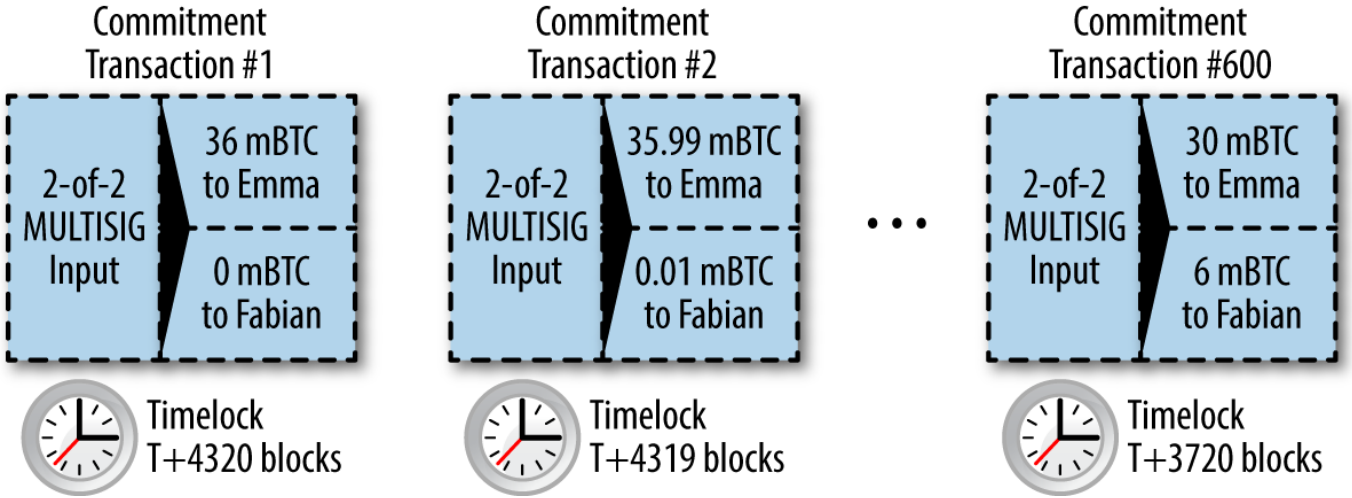
**Can Emma transmit Commit #2 and continue viewing the video?**

- If she doesn't sign Commit #3 NO, and Commit #3 will propagate before #2

**If the two parties are fair, they just sign the last commit without the timelock. If someone is unfair, they can just use the last commit and recover their balance!**

# A simple channel

## Timelocked




https://github.com/bitcoinbook/bitcoinbook/blob/develop/ch12.asciidoc

# Timelocked channels

**The good part:**

- The two parties need not confy in each other
- There is no intermediary
- Each new commit invalidates the previous one

**The bad part:**

- There is a max timelock = the channel has a finte lifespan (e.g. 4320 blocks)
- There is a limit on the number of transactions/commits (4320)
- There is a channel max capacity (but this is OK)

# Bidirectional channels

## Solution for the two probelms

**Can I revoke a previous commit?**

- Once valid, a Bitcoin transaction is good forever
- The only way it is invalidated, is if its inputs have already been spent
- With timelocks we do precisely that, the last commit does a "double spend" of the inputs of the previous commit

- But one needs to order everything perfectly
- Also, the money goes only in one direction (unidirectional channel)
- And some other issues

# Bidirectional channels

## Solution for the two probelms

**The best solution:**

- Bidirectional channels
- With revokation keys
- That never expire (if the participants do not close the channel)

Starts with a funding transaction to a multisig:

- Transaction where Alice pays 5BTC, Bob pays 5BTC
- To an output that is a 2-2 MULTISIG of Alice + Bob
- (We can do a saftey/refund for the deposit as before)

# Bidirectional channels

## Asymmetric commit

Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <5BTC>
Lock 1: <1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Alice PK>
CHECKSIG

Sigs:

# Bidirectional channels

## Asymmetric commit


Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <5BTC>
Lock 1: <1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Alice PK>
CHECKSIG

Sigs:

# Bidirectional channels

## Asymmetric commit


Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <5BTC>
Lock 1: <1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Alice PK>
CHECKSIG

Sigs:

# Bidirectional channels

## Asymmetric commit


Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <5BTC>
Lock 1: <1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Alice PK>
CHECKSIG

Sigs:

Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <5BTC>
Lock 1: <1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Alice PK>
CHECKSIG

Sigs:

# Bidirectional channels

tric commit


Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <5BTC>
Lock 1: <1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Alice PK>
CHECKSIG

Sigs:

Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <5BTC>
Lock 1: <1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Alice PK>
CHECKSIG

Sigs: Bob

# Bidirectional channels

## Asymmetric commit


Commit #1 A

Input: 2-2 MULTISIG

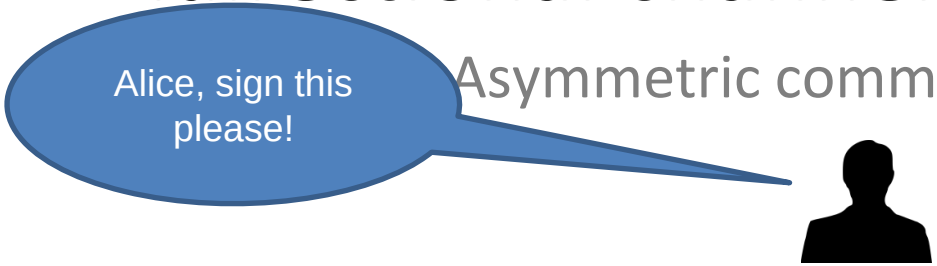
Output 0: <5BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <5BTC>
Lock 1: <1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Alice PK>
CHECKSIG

Sigs: Bob

# Bidirectional channels

## Asymmetric commit




Input: 2-2 MULTISIG

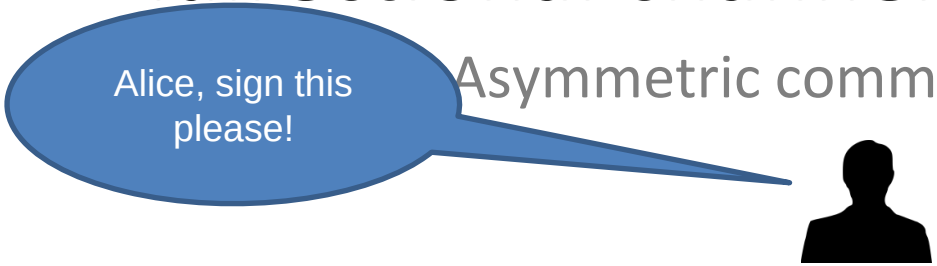
Output 0: <5BTC>
Lock 0: <Alice PK> CHECKSIG

Output 1: <5BTC>
Lock 1: <1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Bob PK>
CHECKSIG

Sigs:

# Bidirectional channels

## Asymmetric commit




Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Alice PK> CHECKSIG

Output 1: <5BTC>
Lock 1: <1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Bob PK>
CHECKSIG

Sigs:

Input: 2-2 MULTISIG

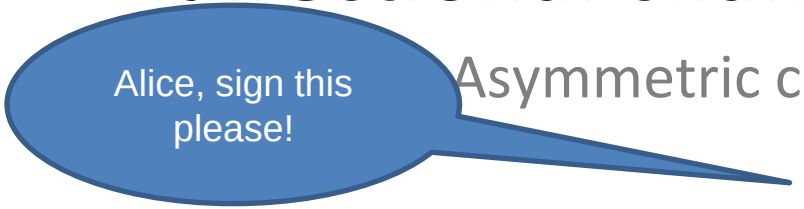
Output 0: <5BTC>
Lock 0: <Alice PK> CHECKSIG

Output 1: <5BTC>
Lock 1: <1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Bob PK>
CHECKSIG

Sigs:

# Bidirectional channels

## Asymmetric commit




Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Alice PK> CHECKSIG

Output 1: <5BTC>
Lock 1: <1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Bob PK>
CHECKSIG

Sigs: Alice

Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Alice PK> CHECKSIG

Output 1: <5BTC>
Lock 1: <1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Bob PK>
CHECKSIG

Sigs:

# Bidirectional channels

## Asymmetric commit


Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Alice PK> CHECKSIG

Output 1: <5BTC>
Lock 1: <1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Bob PK>
CHECKSIG

Sigs: Alice

# Bidirectional channels

## Asymmetric commit

Commit #1 A

Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <5BTC>
Lock 1: <1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Alice PK>
CHECKSIG

Sigs: Bob

Commit #1 B

Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Alice PK> CHECKSIG

Output 1: <5BTC>
Lock 1: <1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Bob PK>
CHECKSIG

Sigs: Alice

# Bidirectional channels

## Asymmetric commit

Commit #1 A

Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <5BTC>
Lock 1: <1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Alice PK>
CHECKSIG

Sigs: Bob, Alice

Commit #1 B

Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Alice PK> CHECKSIG

Output 1: <5BTC>
Lock 1: <1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Bob PK>
CHECKSIG

Sigs: Alice, Bob

# Bidirectional channels

## Revocable assymetric commit

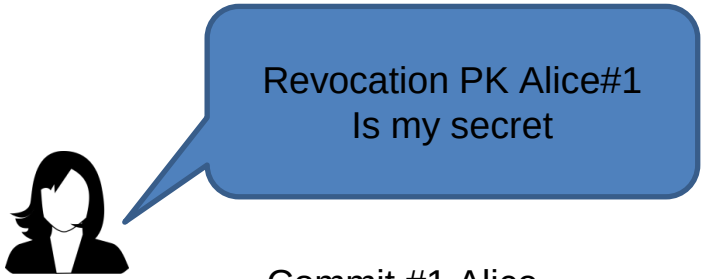
**Important**: Slides 96 – 118 are not really how revocable commits work!

I am just using this simplified version in order to illustrate revocation keys!

- The original solution assumes that the secret key for a revocation thing is split!
- But the same logic applies!

# Bidirectional channels

## Revocable assymetric commit




Commit #1 Alice

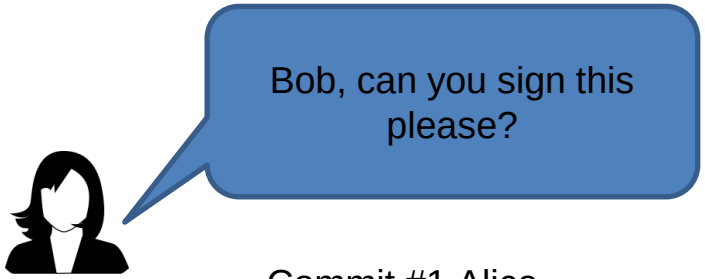
```
Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <5BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Alice#1>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Alice PK>
            CHECKSIG
Sigs:
```

# Bidirectional channels

Revocable etric commit





Commit #1 Alice

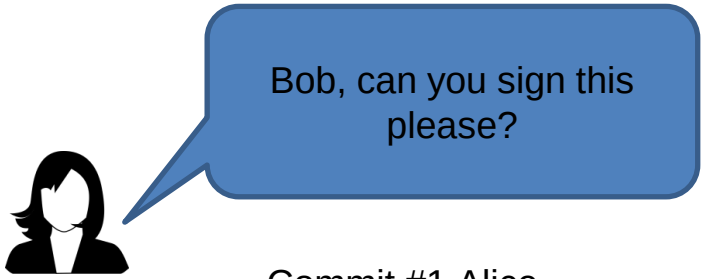
```
Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <5BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Alice#1>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Alice PK>
            CHECKSIG
Sigs:
```

# Bidirectional channels

Revocable etric commit





Commit #1 Alice

```
Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <5BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Alice#1>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Alice PK>
            CHECKSIG
Sigs: Bob
```

# Bidirectional channels

## Revocable assymetric commit

Revocation PK Alice#1
Is my secret

Commit #1 Alice

Revocation PK Bob#1
Is my secret

Commit #1 Bob

```
Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <5BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Alice#1>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Alice PK>
            CHECKSIG
Sigs: Bob
```

```
Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Alice PK> CHECKSIG

Output 1: <5BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Bob#1>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Bob PK>
            CHECKSIG
Sigs:
```

# Bidirectional channels

## Revocable assymetric commit

Revocation PK Alice#1
Is my secret

Commit #1 Alice

Alice, I need your
signature please ☺

Commit #1 Bob

```
Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <5BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Alice#1>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Alice PK>
            CHECKSIG
Sigs: Bob
```

```
Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Alice PK> CHECKSIG

Output 1: <5BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Bob#1>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Bob PK>
            CHECKSIG
Sigs:
```

# Bidirectional channels

## Revocable assymetric commit

Revocation PK Alice#1
Is my secret

Commit #1 Alice

Revocation PK Bob#1
Is my secret

Commit #1 Bob

Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <5BTC>
Lock 1: IF
<999 blocks> CHECKSEQUENCEVERIFY
<Revocation PK Alice#1>
ELSE
<1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Alice PK>
CHECKSIG
Sigs: Bob

Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Alice PK> CHECKSIG

Output 1: <5BTC>
Lock 1: IF
<999 blocks> CHECKSEQUENCEVERIFY
<Revocation PK Bob#1>
ELSE
<1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Bob PK>
CHECKSIG
Sigs: Alice

# Bidirectional channels

ssymetric commit

If I send the tx to BTC, I need to wait 1000 blocks

Any I get paid right away; haha ☺

Commit #1 Alice

Commit #1 Bob

```
Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <5BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Alice#1>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Alice PK>
            CHECKSIG
Sigs: Bob, Alice
```

# Bidirectional channels

## Revocable asymmetric commit

I pay 2 BTC to Bob

Commit #2 Alice

```
Input: 2-2 MULTISIG

Output 0: <7BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <3BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Alice#2>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Alice PK>
            CHECKSIG
Sigs:
```

# Bidirecti channels

Revocab ric commit

I pay 2 BTC to Bob

OK, I'll sign

Commit #2 Alice

```
Input: 2-2 MULTISIG

Output 0: <7BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <3BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Alice#2>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Alice PK>
            CHECKSIG
Sigs:
```

# Bidirecti hannels

Revocab ric commit

I pay 2 BTC to Bob

OK, I'll sign

Commit #2 Alice

```
Input: 2-2 MULTISIG

Output 0: <7BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <3BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Alice#2>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Alice PK>
            CHECKSIG
Sigs: Bob
```

# Bidirecti hannels

Revocab ric commit

Haha, I'll steal your money

Nooooo ☹

Commit #1 Alice

```
Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <5BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Alice#1>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Alice PK>
            CHECKSIG
Sigs: Bob, Alice
```

But what is this for?

# channels

tric commit

I pay 2 BTC to Bob

OK, if you revoke the previous commit I'll sign

Commit #2 Alice

```
Input: 2-2 MULTISIG

Output 0: <7BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <3BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Alice#2>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Alice PK>
            CHECKSIG
Sigs:
```

# channels

tric commit

I pay 2 BTC to Bob

OK, if you revoke the previous commit I'll sign

<Revocation PK Alice#1>

Commit #2 Alice

```
Input: 2-2 MULTISIG

Output 0: <7BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <3BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Alice#2>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Alice PK>
            CHECKSIG
Sigs:
```

# Bidirectional channels

Revocable asymmetric commit

I pay 2 BTC to Bob

Perfect

<Revocation PK Alice#1>

Commit #2 Alice

```
Input: 2-2 MULTISIG

Output 0: <7BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <3BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Alice#2>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Alice PK>
            CHECKSIG
Sigs: Bob
```

# Bidirectional channels

## Revocable asymmetric commit

Haha, I'll steal your money

<Revocation PK Alice#1>

Commit #1 Alice

```
Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <5BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Alice#1>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Alice PK>
            CHECKSIG
Sigs: Bob, Alice
```

# Bidirectional channels

## Revocable asymmetric commit

Haha, I'll steal your money

<Revocation PK Alice#1>

Commit #1 Alice

```
Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <5BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Alice#1>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Alice PK>
            CHECKSIG
Sigs: Bob, Alice
```

# Bidirectional channels

## Revocable asymmetric commit

Commit #1 Alice

Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <5BTC>
Lock 1: IF
<999 blocks> CHECKSEQUENCEVERIFY
<Revocation PK Alice#1>
ELSE
<1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Alice PK>
CHECKSIG
Sigs: Bob, Alice

<Revocation PK Alice#1>

I get everything!

# Bidirectional channels

## Revocable asymmetric commit

I pay 2 BTC to Bob

<Revocation PK Alice#1>

Commit #2 Alice

```
Input: 2-2 MULTISIG

Output 0: <7BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <3BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Alice#2>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Alice PK>
            CHECKSIG
Sigs: Bob
```

# Bidirectional channels

## Revocable asymmetric commit

I pay 2 BTC to Bob

<Revocation PK Alice#1>

Commit #2 Alice

```
Input: 2-2 MULTISIG

Output 0: <7BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <3BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Alice#2>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Alice PK>
            CHECKSIG
Sigs: Bob
```

Commit #2 Bob

```
Input: 2-2 MULTISIG

Output 0: <3BTC>
Lock 0: <Alice PK> CHECKSIG

Output 1: <7BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Bob#2>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Bob PK>
            CHECKSIG
Sigs:
```

# Bidirectional channels

## Revocable asymmetric commit

Revocation key#1 please!

<Revocation PK Alice#1>

Commit #2 Alice

```
Input: 2-2 MULTISIG

Output 0: <7BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <3BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Alice#2>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Alice PK>
            CHECKSIG
Sigs: Bob
```

Commit #2 Bob

```
Input: 2-2 MULTISIG

Output 0: <3BTC>
Lock 0: <Alice PK> CHECKSIG

Output 1: <7BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Bob#2>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Bob PK>
            CHECKSIG
Sigs:
```

# Bidirectional channels

## Revocable asymmetric commit

Revocation key#1 please!

<Revocation PK Alice#1>

<Revocation PK Bob#1>

Commit #2 Alice

Commit #2 Bob

Input: 2-2 MULTISIG

Output 0: <7BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <3BTC>
Lock 1: IF
<999 blocks> CHECKSEQUENCEVERIFY
<Revocation PK Alice#2>
ELSE
<1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Alice PK>
CHECKSIG
Sigs: Bob

Input: 2-2 MULTISIG

Output 0: <3BTC>
Lock 0: <Alice PK> CHECKSIG

Output 1: <7BTC>
Lock 1: IF
<999 blocks> CHECKSEQUENCEVERIFY
<Revocation PK Bob#2>
ELSE
<1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Bob PK>
CHECKSIG
Sigs:

# Bidirectional channels

## Revocable asymmetric commit

Perfect

<Revocation PK Alice#1>

<Revocation PK Bob#1>

Commit #2 Alice

Commit #2 Bob

Input: 2-2 MULTISIG

Output 0: <7BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <3BTC>
Lock 1: IF
<999 blocks> CHECKSEQUENCEVERIFY
<Revocation PK Alice#2>
ELSE
<1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Alice PK>
CHECKSIG
Sigs: Bob

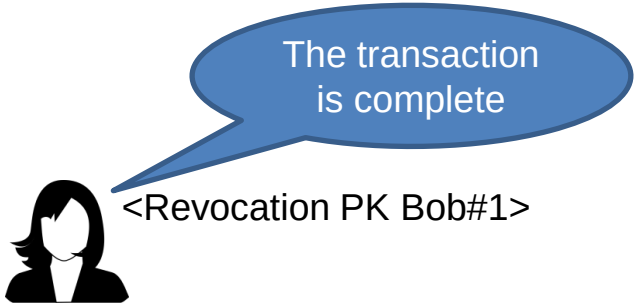
Input: 2-2 MULTISIG

Output 0: <3BTC>
Lock 0: <Alice PK> CHECKSIG

Output 1: <7BTC>
Lock 1: IF
<999 blocks> CHECKSEQUENCEVERIFY
<Revocation PK Bob#2>
ELSE
<1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Bob PK>
CHECKSIG
Sigs: Alice

# Bidirectional channels

## Revocable asymmetric commit




<Revocation PK Alice#1>

Commit #2 Alice

```
Input: 2-2 MULTISIG

Output 0: <7BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <3BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Alice#2>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Alice PK>
            CHECKSIG
Sigs: Bob
```

Commit #2 Bob

```
Input: 2-2 MULTISIG

Output 0: <3BTC>
Lock 0: <Alice PK> CHECKSIG

Output 1: <7BTC>
Lock 1: IF
            <999 blocks> CHECKSEQUENCEVERIFY
            <Revocation PK Bob#2>
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Bob PK>
            CHECKSIG
Sigs: Alice
```

# Bidirectional channels

## Solution to the two problems

**Questions**

- **Can Alice publish Commit #1? (What does Bob have?)**

**Idea:**

- **With the revocation key, Bob/Alice the previous state is worse that current**

**A detail:**

- **The pary that is getting paid needs to receive the revocation key before signing**
- **To her, the last commit is always better!**

# Bidirectional channels

## Revocable asymmetric commit

Not so good:

Input: 2-2 MULTISIG

Output 0: <7BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <3BTC>
Lock 1: IF
<Revocation PK Alice#2> EQUAL DROP
<Bob PK> CHECKSIG
ELSE
<1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Alice PK>
ENDIF
CHECKSIG

Sigs: Bob

Input: 2-2 MULTISIG

Output 0: <7BTC>
Lock 0: <Alice PK> CHECKSIG

Output 1: <3BTC>
Lock 1: IF
<Revocation PK Alice#2> EQUAL DROP
<Alice PK> CHECKSIG
ELSE
<1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Bob PK>
ENDIF
CHECKSIG

Sigs: Alice

# Bidirectional channels

## Solution to the two problems

**In reality**

- **Alice has one half of the revocation key**
- **Bob has the other half**

- **The interchange the halves when doing the revoke**

- **This assures that if one party leaves the channel, the other party can spend the previous commit with revoke1 + revoke2**

# Bidirectional channels

## Revocable asymmetric commit

Alice has half of secret keys A and B

Commit Alice

Input: 2-2 MULTISIG

Output 0: <7BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <3BTC>
Lock 1: IF
<Revocation PK A> CHECKSIG
ELSE
<1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Alice PK>
CHECKSIG

Sigs: Bob

Commit Bob

ut: 2-2 MULTISIG

Outpu 3BTC>
Lock 0: <Ali PK> CHECKSIG

Output 1: <7BTC>
Lock 1: IF
<Revocation PK B> CHECKSIG
ELSE
<1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Bob PK>
CHECKSIG

Sigs:

# Bidirectional channels

## Revocable asymmetric commit

Alice has half of secret keys A and B

Commit Alice

Input: 2-2 MULTISIG

Output 0: <7BTC>
Lock 0: <Bob PK> CHECKSI

Output 1: <3BTC>
Lock 1: IF
<Revocation PK A> CHECKSIG
ELSE
<1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Alice PK>
CHECKSIG

Sigs: Bob

Commit Bob

Input: 2-2 MULTISIG

Output 0: <3BTC>
Lock 0: <Alice PK> CHECKSIG

Output 1: <7BTC>
Lock 1: IF
<Revocation PK B> CHECKSIG
ELSE
<1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Bob PK>
CHECKSIG

Sigs:

# Bidirectional channels

## Revocable asymmetric commit

Alice has half of secret keys A and B

Commit Alice

Input: 2-2 MULTISIG

Output 0: <7BTC>
Lock 0: <Bob PK> CHECKSI

Output 1: <3BTC>
Lock 1: IF
<Revocation PK A> CHECKSIG
ELSE
<1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Alice PK>
CHECKSIG

Sigs: Bob

Bob has the other half

Commit Bob

Input: 2-

Output 0: <3BTC>
Lock 0: <Alice PK> CHECKSIG

Output 1: <7BTC>
Lock 1: IF
<Revocation PK B> CHECKSIG
ELSE
<1000 blocks>
CHECKSEQUENCEVERIFY
DROP
<Bob PK>
CHECKSIG

Sigs:

# Bidirectional channels

## Revocable asymmetric commit

Revoke is just my half of the key!

Commit Alice

```
Input: 2-2 MULTISIG

Output 0: <7BTC>
Lock 0: <Bob PK> CHECKSIG

Output 1: <3BTC>
Lock 1: IF
            <Revocation PK A> CHECKSIG
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Alice PK>
            CHECKSIG

Sigs: Bob
```

Commit Bob

```
Input: 2-2 MULTISIG

Output 0: <3BTC>
Lock 0: <Alice PK> CHECKSIG

Output 1: <7BTC>
Lock 1: IF
            <Revocation PK B> CHECKSIG
        ELSE
            <1000 blocks>
            CHECKSEQUENCEVERIFY
            DROP
            <Bob PK>
            CHECKSIG

Sigs:
```

# Bidirectional channels

## Solution to the two problems

**Important:**

- **The recovation is not automatic**
- **The two parties need to monitor network for transactions**
- **Basically, they need to run a node (can be SPV)**

**Problem with all this:**

- **How to combine two payment channels?**
- **HTLC (Hashed Timelock Contract; BIP 112)**

**Lightning Network!!!**

# HTLC

**Hashed TimeLock Contract (HTLC):**

- **Alice and Bob have their payment channel**
- **Alice pays 1 BTC to Bob, if Bob reveals her the secret in 1 day**
- **If not, Alice can recover her 1 BTC**

**"HTLC Output"**

# HTCL

Channel A<->B, A & B have 5 BTC each

A sends 1 BTC to B with HTCL of 1 day

secret s.t. H(secret) = secretHash

Commit Alice:

Input: 2-2 MULTISIG

Output 0: <5BTC> → <PK Bob>
Output 1: <4BTC> → <PK Alice> (+1000 block delay)

Output 2: <1BTC>
HASH160 <secretHash> EQUAL
IF
<PK Bob>
ELSE
<HTLC timeout = T+1day> CHECKLOCKTIMEVERIFY
<PK Alice>
ENDIF
CHECKSIG

Sigs: Bob

Commit Bob:

Input: 2-2 MULTISIG

Output 0: <4BTC> → <PK Alice>
Output 1: <5BTC> → <PK Bob> (+1000 block delay)

Output 2: <1BTC>
HASH160 <secretHash> EQUAL
IF
<PK Bob>
ELSE
<HTLC timeout = T+1day> CHECKLOCKTIMEVERIFY
<PK Alice>
ENDIF
CHECKSIG

Sigs: Alice

# HTCL

Case #1: Bob publishes the transaction

He can charge Output #2 with the secret (before 1 day passes)

secret

Commit Alice:

Input: 2-2 MULTISIG

Output 0: <5BTC> → <PK Bob>
Output 1: <4BTC> → <PK Alice> (+1000 block delay)

Output 2: <1BTC>
HASH160 <secretHash> EQUAL
IF
<PK Bob>
ELSE
<HTLC timeout = T+1day> CHECKLOCKTIMEVERIFY
<PK Alice>
ENDIF
CHECKSIG

Sigs: Bob

Commit Bob:

Input: 2-2 MULTISIG

Output 0: <4BTC> → <PK Alice>
Output 1: <5BTC> → <PK Bob> (+1000 block delay)

Output 2: <1BTC>
HASH160 <secretHash> EQUAL
IF
<PK Bob>
ELSE
<HTLC timeout = T+1day> CHECKLOCKTIMEVERIFY
<PK Alice>
ENDIF
CHECKSIG

Sigs: Alice

# HTCL

Case #2: Alice publishes the transaction

She can recover Output #2 after a day

Commit Alice:

Input: 2-2 MULTISIG

Output 0: <5BTC> → <PK Bob>
Output 1: <4BTC> → <PK Alice> (+1000 block delay)

Output 2: <1BTC>
HASH160 <secretHash> EQUAL
IF
<PK Bob>
ELSE
<HTLC timeout = T+1day> CHECKLOCKTIMEVERIFY
<PK Alice>
ENDIF
CHECKSIG

Sigs: Bob

secret

Commit Bob:

Input: 2-2 MULTISIG

Output 0: <4BTC> → <PK Alice>
Output 1: <5BTC> → <PK Bob> (+1000 block delay)

Output 2: <1BTC>
HASH160 <secretHash> EQUAL
IF
<PK Bob>
ELSE
<HTLC timeout = T+1day> CHECKLOCKTIMEVERIFY
<PK Alice>
ENDIF
CHECKSIG

Sigs: Alice

# HTCL

Channel A<->B, A has 10 BTC, B 8BTC

Channel B <-> C, B has 5BTC, C 3BTC

# HTCL

Channel A<->B, A has 10 BTC, B 8BTC

Channel B <-> C, B has 5BTC, C 3BTC

I want to send 1BTC
to Charlie

# HTCL

Channel A<->B, A has 10 BTC, B 8BTC

Channel B <-> C, B has 5BTC, C 3BTC

secret

I want to send 1BTC
to Charlie

# HTCL

Channel A<->B, A has 10 BTC, B 8BTC

Channel B <-> C, B has 5BTC, C 3BTC

secret

secretHash = h(secret)

I want to send 1BTC
to Charlie

# HTCL

Channel A<->B, A has 10 BTC, B 8BTC

Channel B <-> C, B has 5BTC, C 3BTC

secret

secretHash = h(secret)

I want to send 1BTC to Charlie

Pagale Bob 1BTC to HTLC output if he shows you secret

# HTCL

Channel A<->B, A has 10 BTC, B 8BTC

Channel B <-> C, B has 5BTC, C 3BTC

secret

Commit Alice (A-B)

secretHash = h(secret)

Input: 2-2 MULTISIG (A,B)

Output 0: <8BTC> → <PK Bob>
Output 1: <8.9 BTC> → <PK Alice> (+1000 block delay)

Output 2: <1.1 BTC>
HASH160 <secretHash> EQUAL
IF
<PK Bob>
ELSE
<HTLC timeout = T+5h> CHECKLOCKTIMEVERIFY
<PK Alice>
ENDIF
CHECKSIG

Sigs: Bob

# HTCL

Charlie will tell you secret if you pay him 1 BTC

Channel A<

Channel B <-> C, B has 5BTC, C 3BTC

secret

Commit Alice (A-B)

secretHash = h(secret)

Input: 2-2 MULTISIG (A,B)

Output 0: <8BTC> → <PK Bob>
Output 1: <8.9 BTC> → <PK Alice> (+1000 block delay)

Output 2: <1.1 BTC>
HASH160 <secretHash> EQUAL
IF
<PK Bob>
ELSE
<HTLC timeout = T+5h> CHECKLOCKTIMEVERIFY
<PK Alice>
ENDIF
CHECKSIG

Sigs: Bob

# HTCL

Charlie will tell you secret if you pay him 1 BTC

Channel A<

Cool ☺

Ch[illegible]el B <-> C, B has 5BTC, C 3BTC

secret

Commit Alice (A-B)

Input: 2-2 MULTISIG (A,B)

Output 0: <8BTC> → <PK Bob>
Output 1: <8.9 BTC> → <PK Alice> (+1000 block delay)

Output 2: <1.1 BTC>
HASH160 <secretHash> EQUAL
IF
<PK Bob>
ELSE
<HTLC timeout = T+5h> CHECKLOCKTIMEVERIFY
<PK Alice>
ENDIF
CHECKSIG

Sigs: Bob

secretHash = h(secret)

# HTCL

Charlie will tell you secret if you pay him 1 BTC

Channel A<

Channel B <-> C, B has 5BTC, C 3BTC

secret

Commit Alice (A-B)

Input: 2-2 MULTISIG (A,B)

Output 0: <8BTC> → <PK Bob>
Output 1: <8.9 BTC> → <PK Alice> (+1000 block delay)

Output 2: <1.1 BTC>
HASH160 <secretHash> EQUAL
IF
<PK Bob>
ELSE
<HTLC timeout = T+5h> CHECKLOCKTIMEVERIFY
<PK Alice>
ENDIF
CHECKSIG

Sigs: Bob

Commit Bob (B-C)

Input: 2-2 MULTISIG (C,B)

Output 0: <3BTC> → <PK Charlie>
Output 1: <3.9 BTC> → <PK Bob> (+1000 block delay)

Output 2: <1.1 BTC>
HASH160 <secretHash> EQUAL
IF
<PK Charlie>
ELSE
<HTLC timeout = T+4h> CHECKLOCKTIMEVERIFY
<PK Bob>
ENDIF
CHECKSIG

Sigs: Charlie

# HTCL

Channel A<->B, A has 10 BTC, B 8BTC

Channel B <-> C, B has 5BTC, C 3BTC

secret

Commit Alice (A-B)

Commit Bob (B-C)

Input: 2-2 MULTISIG (A,B)

Output 0: <8BTC> → <PK Bob>
Output 1: <8.9 BTC> → <PK Alice> (+1000 block delay)

Output 2: <1.1 BTC>
HASH160 <secretHash> EQUAL
IF
<PK Bob>
ELSE
<HTLC timeout = T+5h> CHECKLOCKTIMEVERIFY
<PK Alice>
ENDIF
CHECKSIG

Sigs: Bob

Input: 2-2 MULTISIG (C,B)

Output 0: <3BTC> → <PK Charlie>
Output 1: <3.9 BTC> → <PK Bob> (+1000 block delay)

Output 2: <1.1 BTC>
HASH160 <secretHash> EQUAL
IF
<PK Charlie>
ELSE
<HTLC timeout = T+4h> CHECKLOCKTIMEVERIFY
<PK Bob>
ENDIF
CHECKSIG

Sigs: Charlie

# HTCL

Bob, Secret is "secret"

Channel A<->B, A has 10 BTC, B 8BTC

Channel B <-> C, B has 5BTC, C 3BTC

Commit Alice (A-B)

Commit Bob (B-C)

secret

Input: 2-2 MULTISIG (A,B)

Output 0: <8BTC> → <PK Bob>
Output 1: <8.9 BTC> → <PK Alice> (+1000 block delay)

Output 2: <1.1 BTC>
HASH160 <secretHash> EQUAL
IF
<PK Bob>
ELSE
<HTLC timeout = T+5h> CHECKLOCKTIMEVERIFY
<PK Alice>
ENDIF
CHECKSIG

Sigs: Bob

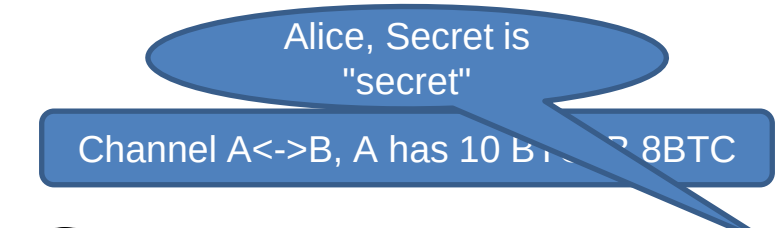
Input: 2-2 MULTISIG (C,B)

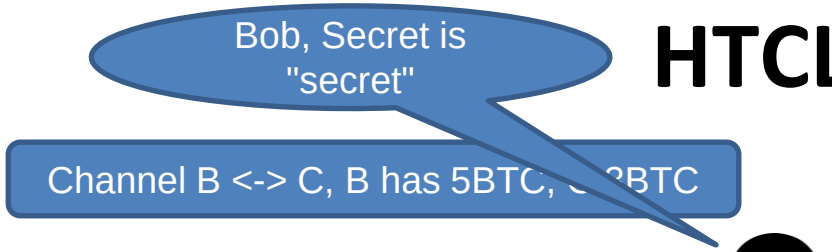
Output 0: <3BTC> → <PK Charlie>
Output 1: <3.9 BTC> → <PK Bob> (+1000 block delay)

Output 2: <1.1 BTC>
HASH160 <secretHash> EQUAL
IF
<PK Charlie>
ELSE
<HTLC timeout = T+4h> CHECKLOCKTIMEVERIFY
<PK Bob>
ENDIF
CHECKSIG

Sigs: Charlie

# HTCL




secret

## Commit Alice (A-B)

Input: 2-2 MULTISIG (A,B)

Output 0: <8BTC> → <PK Bob>
Output 1: <8.9 BTC> → <PK Alice> (+1000 block delay)

Output 2: <1.1 BTC>
HASH160 <secretHash> EQUAL
IF
<PK Bob>
ELSE
<HTLC timeout = T+5h> CHECKLOCKTIMEVERIFY
<PK Alice>
ENDIF
CHECKSIG

Sigs: Bob

## Commit Bob (B-C)

Input: 2-2 MULTISIG (C,B)

Output 0: <3BTC> → <PK Charlie>
Output 1: <3.9 BTC> → <PK Bob> (+1000 block delay)

Output 2: <1.1 BTC>
HASH160 <secretHash> EQUAL
IF
<PK Charlie>
ELSE
<HTLC timeout = T+4h> CHECKLOCKTIMEVERIFY
<PK Bob>
ENDIF
CHECKSIG

Sigs: Charlie

# HTLC

**We are missing one thing**

- **Implement HTLC with revocation keys**
- **The same logic as previously applies**
- **But the transactions are a bit different**

# HTLC

**HTLC can be spent if:**

- **The receiver knows the secret – the funds go to the receiver**
- **The HTLC timed-out (Locktime) – the funds go to the sender**
- **The commit was revoked – the „bad” actor is punished**

# HTCL

Channel A<->B, each has 5 BTC

A sends 1 BTC to B with HTCL

## Commit sender:

Input: 2-2 MULTISIG

Output 0: <5BTC>
Lock 0: <Bob PK> CHECKSIG
Output 1: <4BTC>
Lock 1: IF
<Revocation PK Alice>
ELSE
<1000 blocks> CHECKSEQUENCEVERIFY DROP
<Alice PK> CHECKSIG

Output 2: <1BTC>
H160 <secretHash> EQUAL SWAP
<Revocation PK Alice> EQUAL ADD (that is, OR)
IF
<PK Bob>
ELSE
<HTLC timeout = T+1day> CHECKLOCKTIMEVERIFY
<1000 blocks> CHECKSEQUENCEVERIFY DROP
<PK Alice>
ENDIF
CHECKSIG

Sigs: Bob

## Commit receiver:

Input: 2-2 MULTISIG

Output 0: <4BTC>
Lock 0: <Alice PK> CHECKSIG
Output 1: <5BTC>
Lock 1: IF
<Revocation PK Bob>
ELSE
<1000 blocks> CHECKSEQUENCEVERIFY DROP
<Bob PK> CHECKSIG

Output 2: <1BTC>
H160 <secretHash> EQUAL
IF
<1000 blocks> CHECKSEQUENCEVERIFY DROP
<PK Bob>
ELSE
<Revocation PK Bob> EQUAL
NOTIF
<HTLC timeout = T+1day> CHECKLOCKTIMEVERIFY
ENDIF
<PK Alice>
ENDIF
CHECKSIG

Sigs: Alice

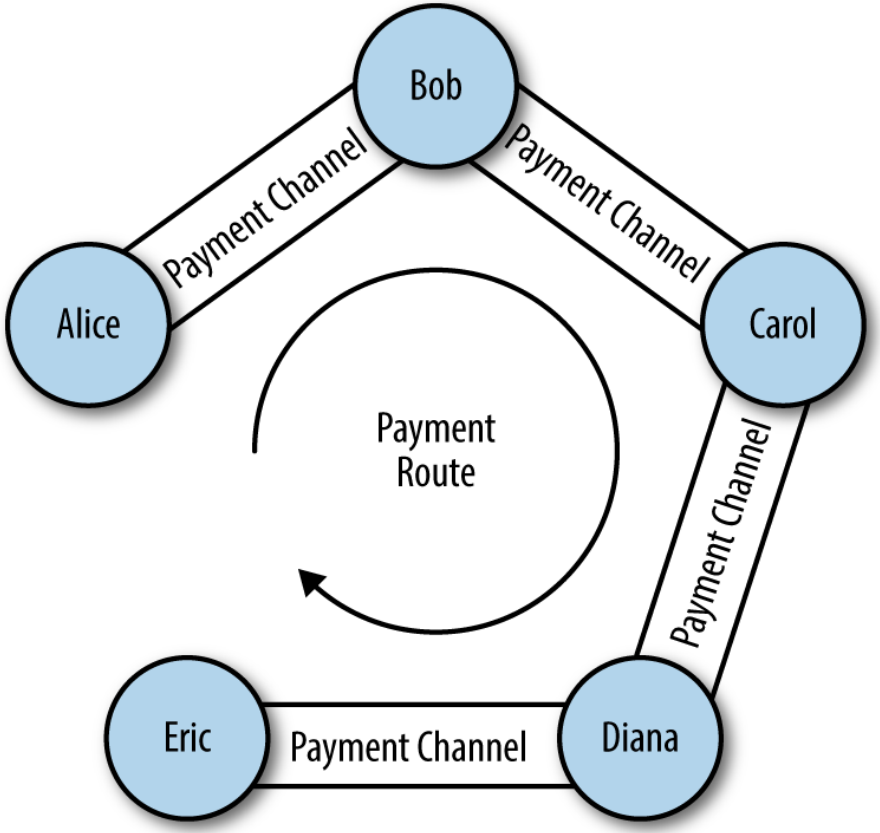
# Lightning network

**Lightning network:**

- Nodes running multiple payment channels
- A pays B by discovering a route with sufficient capacity on the network
- And construct the commit for its first payment channel, with the info on how to proceed

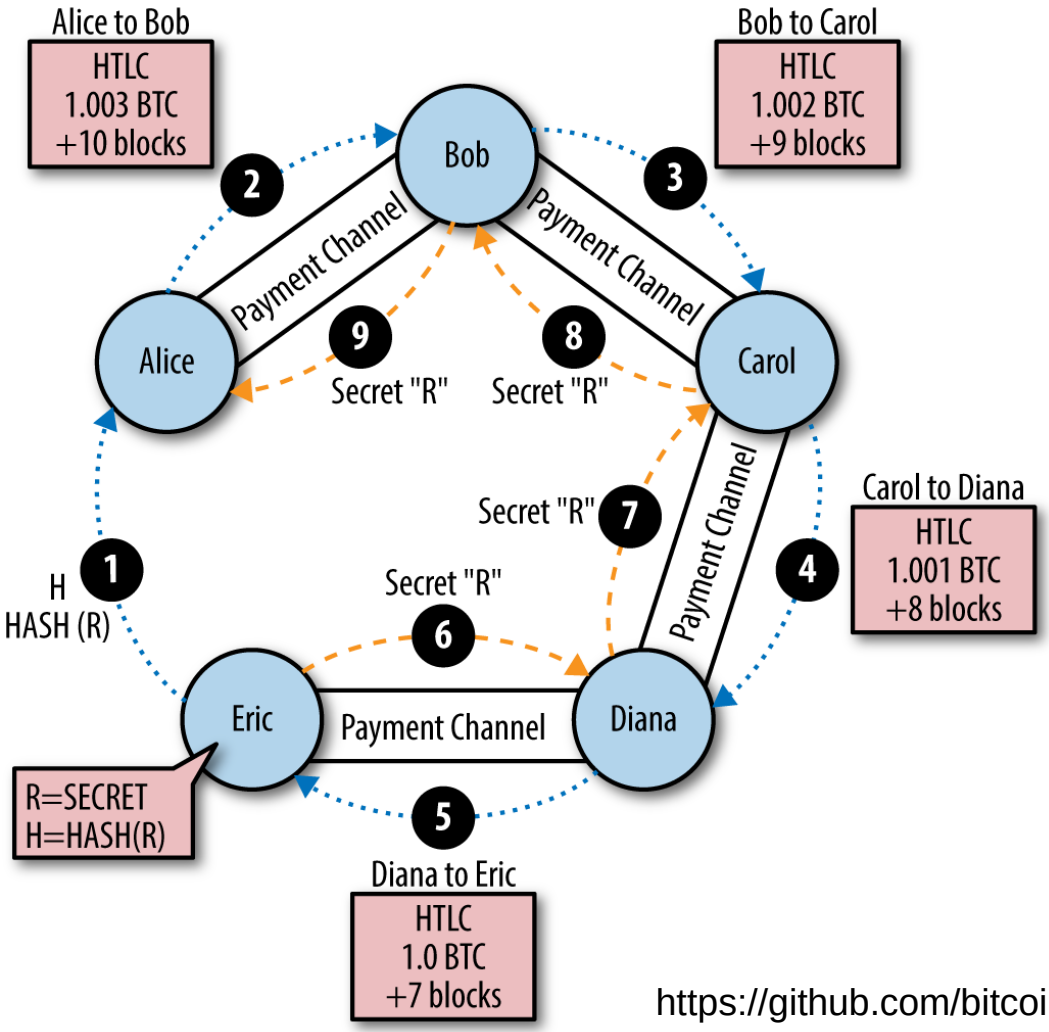
# Lightning network

**Lightning network**




https://github.com/bitcoinbook/bitcoinbook/blob/develop/ch12.asciidoc

# Lightning network




**Alice pays 1 BTC to Eric**

**Eric generates the secret R**

**The remainder: HTLC**

https://github.com/bitcoinbook/bitcoinbook/blob/develop/ch12.asciidoc

# Lightning network

**Lightning network benefits:**

- Speed
- Privacy
- Fungibility
- Granularity (I can pay 1 Satoshi)
- Capacity
- Still trustless (as Bitcoin)

# Lightning network

## Resources for Lightning:

- https://github.com/bitcoinbook/bitcoinbook/blob/develop/ch12.asciidoc
- https://github.com/ElementsProject/lightning/blob/master/doc/deployable-lightning.pdf
- https://github.com/bitcoin/bips/blob/master/bip-0112.mediawiki#Hash_TimeLocked_Contracts
- http://lightning.network/lightning-network-paper.pdf
- https://github.com/ChristopherA/Learning-Bitcoin-from-the-CommandLine/blob/master/11_3_Empowering_Bitcoin_with_Scripts.md