Bitcoin Transaction Scripts

Sameh El-Ansary, PhD

Copyright – Novelari 2018 - Sameh El-Ansary, PhD

The "Script" Language

- Simple language
- Can't do a lot of things
- Used for transaction validation
- Instead of static validation, a script is executed
- Turing incomplete:
  - no loops to avoid a script being a logic bomb targeted to every node in the network

Script Construction (Lock + Unlock)

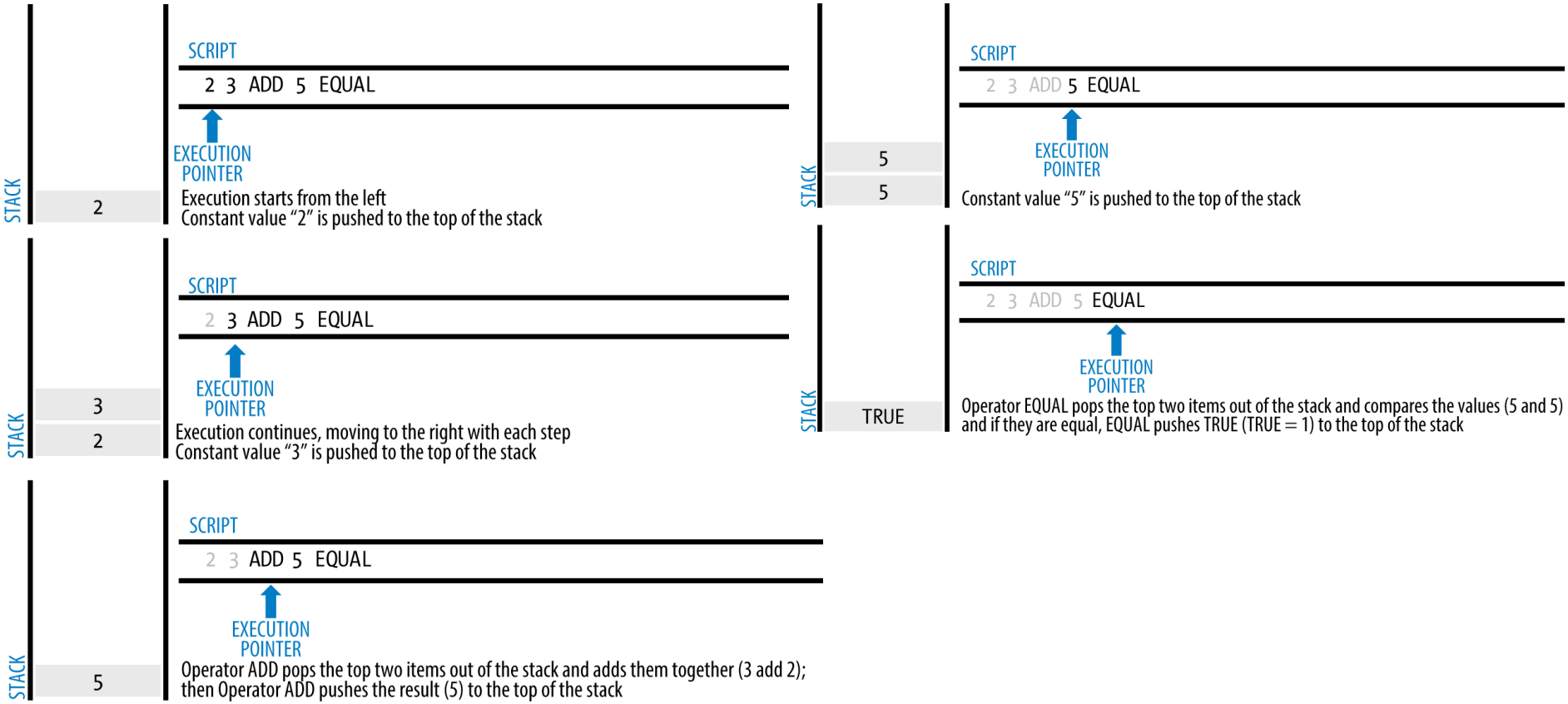
- Every vout has a locking script
- Every vin spending that vout has an unlocking script
- Verification is:
  - Combining both as one script
  - Executing the script, if the output give a value of "OP_TRUE", then the transaction is valid

Copyright – Novelari 2018 - Sameh El-Ansary, PhD

Script Construction (Lock + Unlock)

Locking Script: **3 OP_ADD 5 OP_EQUAL**

Unlocking Script: **2**



Copyright – Novelari 2018 - Sameh El-Ansary, PhD

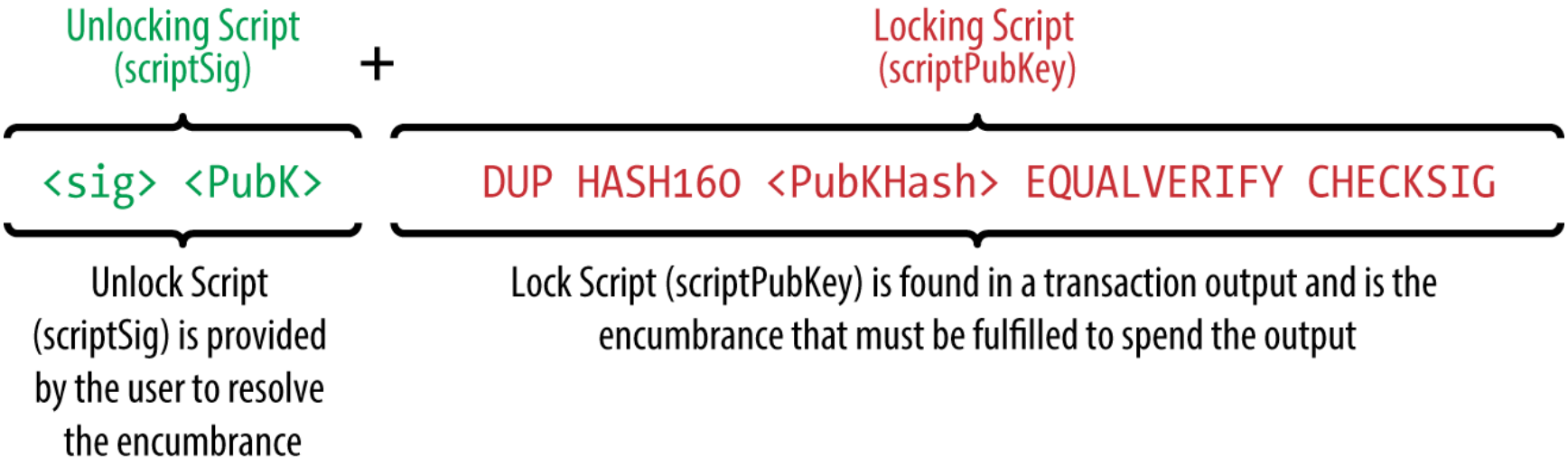
Exercise

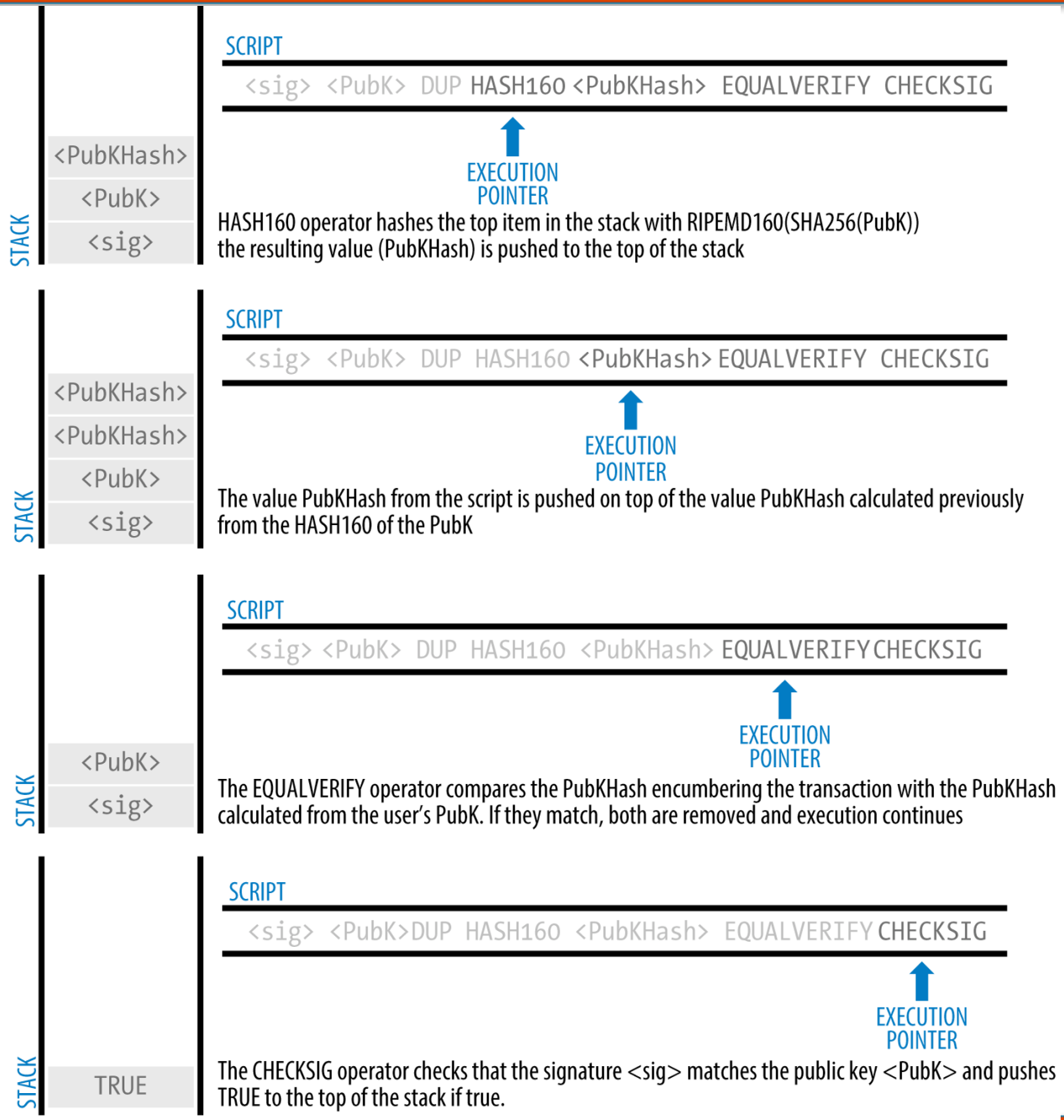
- What is the unlocking script of the following locking script:
  - 7 OP_ADD 3 OP_SUB 1 OP_ADD 7 OP_EQUAL

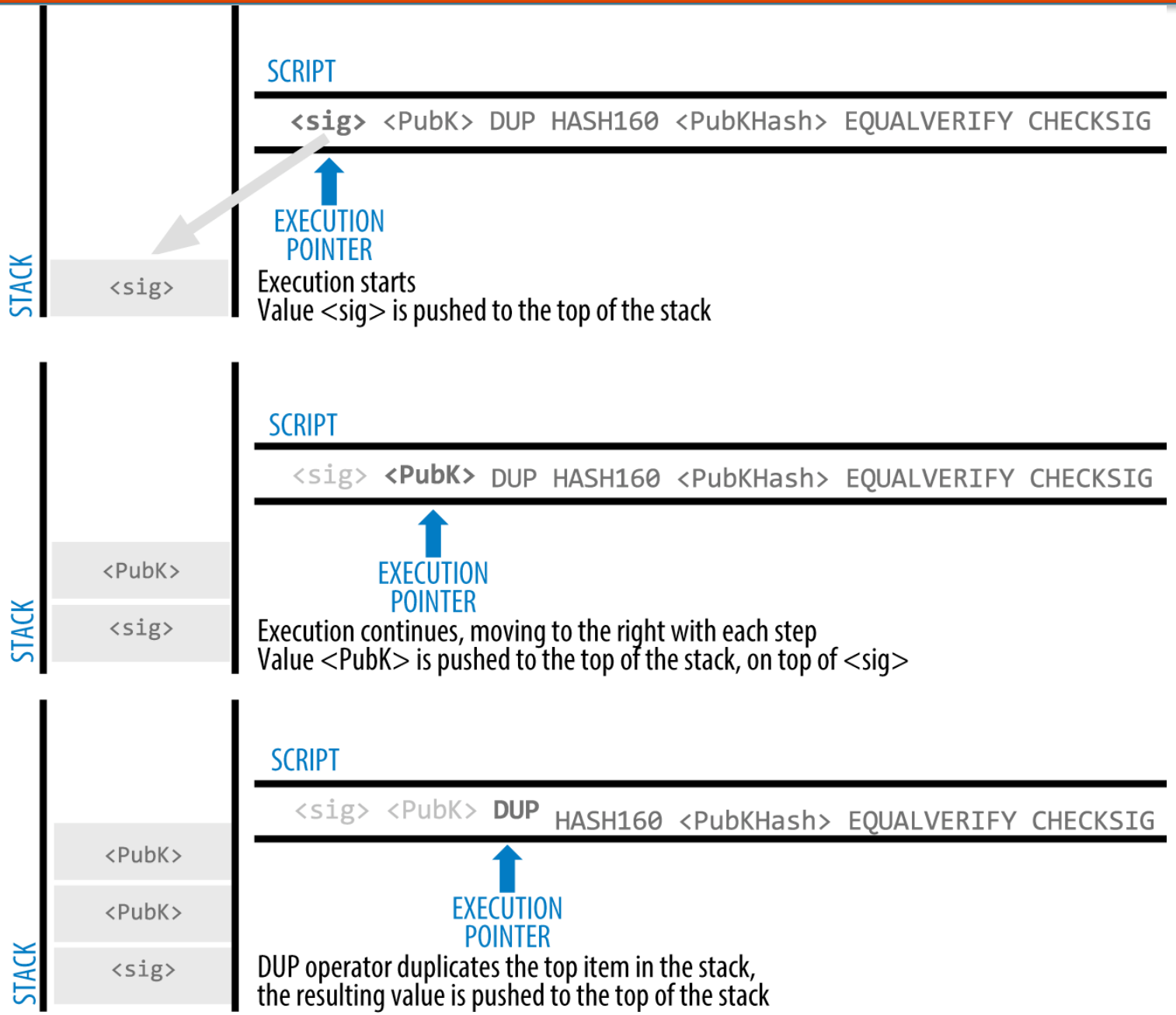
Exercise

- What is the unlocking script of the following locking script:
  - 7 OP_ADD 3 OP_SUB 1 OP_ADD 7 OP_EQUAL
- Answer: 2

P2PKH Script




STACK
<PubKHash>
<PubK>
<sig>
SCRIPT
<sig> <PubK> DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
EXECUTION POINTER
HASH160 operator hashes the top item in the stack with RIPEMD160(SHA256(PubK)) the resulting value (PubKHash) is pushed to the top of the stack
STACK
<PubKHash>
<PubKHash>
<PubK>
<sig>
SCRIPT
<sig> <PubK> DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
EXECUTION POINTER
The value PubKHash from the script is pushed on top of the value PubKHash calculated previously from the HASH160 of the PubK
STACK
<PubK>
<sig>
SCRIPT
<sig> <PubK> DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
EXECUTION POINTER
The EQUALVERIFY operator compares the PubKHash encumbering the transaction with the PubKHash calculated from the user's PubK. If they match, both are removed and execution continues
STACK
TRUE
SCRIPT
<sig> <PubK>DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
EXECUTION POINTER
The CHECKSIG operator checks that the signature <sig> matches the public key <PubK> and pushes TRUE to the top of the stack if true.


SCRIPT
<sig> <PubK> DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
EXECUTION POINTER
STACK
<sig>
Execution starts
Value <sig> is pushed to the top of the stack
SCRIPT
<sig> <PubK> DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
EXECUTION POINTER
STACK
<PubK>
<sig>
Execution continues, moving to the right with each step
Value <PubK> is pushed to the top of the stack, on top of <sig>
SCRIPT
<sig> <PubK> DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
EXECUTION POINTER
STACK
<PubK>
<PubK>
<sig>
DUP operator duplicates the top item in the stack,
the resulting value is pushed to the top of the stack

SIGHASH

Additional Flag with the signature, to say which parts of the whole transaction is the spender signing

SIGHASH flag	Value	Description
ALL	0x01	Signature applies to all inputs and outputs **(Normal)**
NONE	0x02	Signature applies to all inputs, none of the outputs **(Bearer Cheque)**
SINGLE	0x03	Signature applies to all inputs but only the one output with the same index number as the signed input

SIGHASH flag	Value	Description
ALL|ANYONECANPAY	0x81	Signature applies to one input and all outputs **(CrowdFund)**
NONE|ANYONECANPAY	0x82	Signature applies to one input, none of the outputs
SINGLE|ANYONECANPAY	0x83	Signature applies to one input and the output with the same index number

Copyright – Novelari 2018 - Sameh El-Ansary, PhD

Multi-Signature

Lock	2 <Public Key A> <Public Key B> <Public Key C> 3 CHECKMULTISIG
Unlock	<Signature B> <Signature C>

OR

Lock	2 <Public Key A> <Public Key B> <Public Key C> 3 CHECKMULTISIG
Unlock	<Signature A> <Signature B>

Using the Script language we can call create complex scenarios like M out of N signatures for spending

Copyright – Novelari 2018 - Sameh El-Ansary, PhD

Multisig Issues

- Multi-signature are powerful but cumbersome.
- Each sender would have to:
  - Use special wallet with custom transaction scripts.
  - Understand how to create custom scripts.
- Many publics keys → Bigger Tx size in bytes → bigger fees
- These issues resulted that using complex locking scripts is difficult in practice.

Pay-to-Script-Hash (P2SH)

Locking Script	2 PubKey1 PubKey2 PubKey3 PubKey4 PubKey5 5 CHECKMULTISIG
Unlocking Script	Sig1 Sig2

Redeem Script	2 PubKey1 PubKey2 PubKey3 PubKey4 PubKey5 5 CHECKMULTISIG
Locking Script	HASH160 <20-byte hash of redeem script> EQUAL
Unlocking Script	Sig1 Sig2 <redeem script>

The whole locking script is hashed, base58check encoded and used as address e.g. 39RF6JqABiHdYHkfChV6USGMe6Nsr66Gzw

Copyright – Novelari 2018 - Sameh El-Ansary, PhD

Benefits of P2SH

- Multi-signatures are easy to use because senders can send to an address without paying extra fees or extra complexities
- Receivers will still need to deal with the complexities of creation and spending

More Advanced Topics

- Security issues with P2SH
- Timelocks & nSequence (relative time)
- Scripts with flow control (if/then/else)
- Segregated Witness, P2WPKH, **P2WSH**

```
IF
  IF
    2
  ELSE
    <30 days> CHECKSEQUENCEVERIFY DROP
    <Abdul the Lawyer's Pubkey> CHECKSIGVERIFY
    1
  ENDIF
  <Mohammed's Pubkey> <Saeed's Pubkey> <Zaira's Pubkey> 3 CHECKMULTISIG
ELSE
  <90 days> CHECKSEQUENCEVERIFY DROP
  <Abdul the Lawyer's Pubkey> CHECKSIG
ENDIF
```