CSE 3550/5000: Blockchain Technology

Lecture 4

Bitcoin - Part II

Ghada Almashaqbeh

UConn - Spring 2026

Outline

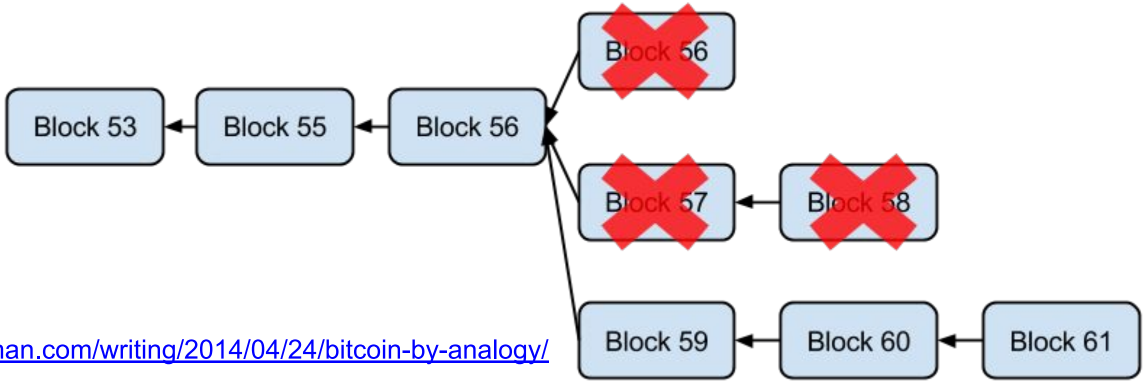
- More about Bitcoin:
  - Consensus.
  - Blockchain forking.
  - Properties like transparency, public verifiability, immutability, etc.
  - Bitcoin scripting language and transaction processing.

Consensus

- Miners hold, hopefully, consistent copies of the blockchain.
  - Only differ in the most recent unconfirmed blocks.
  - A block is confirmed when it is buried under at least 6 blocks.
- A miner votes for a block implicitly:
  - Accept it by including it in the chain and start mining on top of it.
  - Reject it by ignoring the new block and continue mining based on the older blockchain or another newly announced block.
- Remember: Bitcoin network is not perfect!
  - propagation delays, not all nodes hear all announced transactions, nodes may crash at any point of time, etc.
- Result: the blockchain may have multiple branches, i.e., forks.

Blockchain Forking

- Miners work on different branches.
- Resolved by adopting the longest branch since it means more work effort and larger history record.
  - That is, when a miner hears about another branch that is longer than the current one it has, it will switch to the new branch.
- Older history is consistent; all honest miners will agree on a common prefix of the blockchain. They may differ in recent (unconfirmed) history.

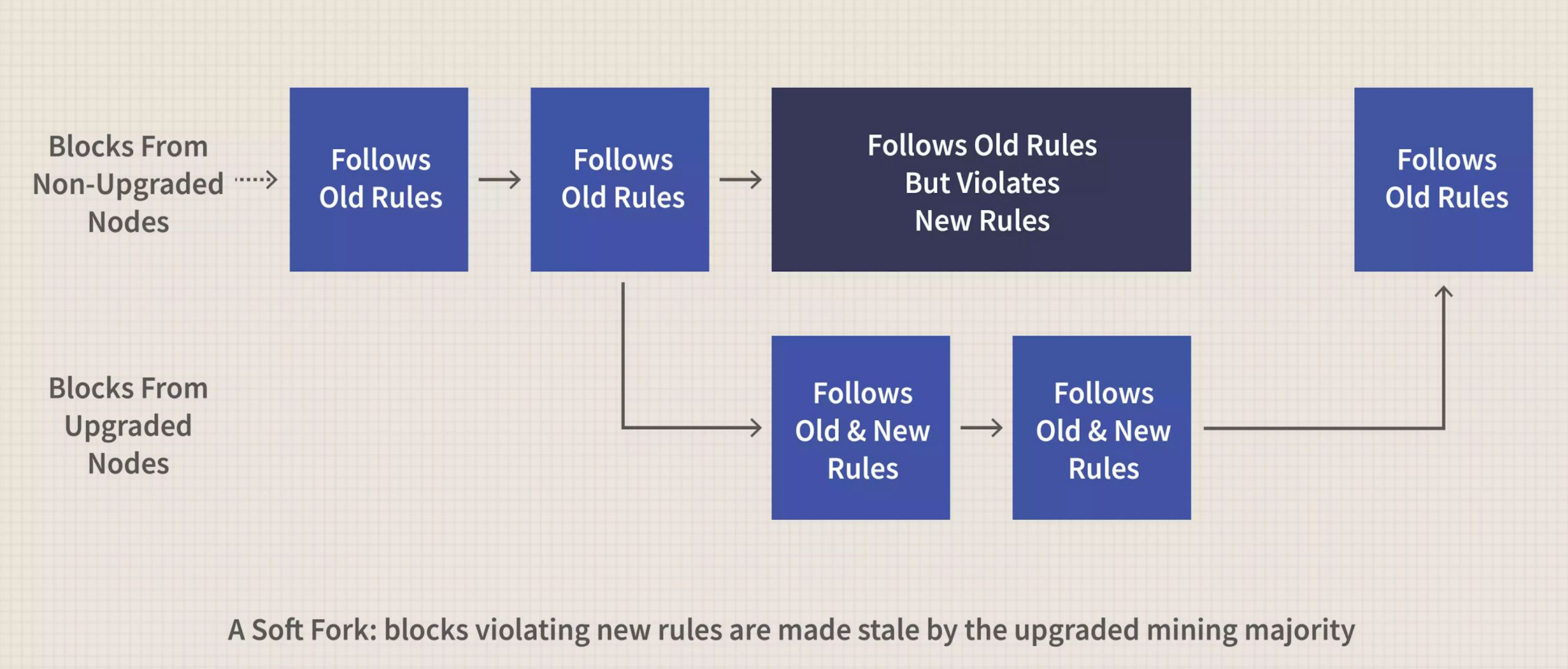


Source: http://www.ybrikman.com/writing/2014/04/24/bitcoin-by-analogy/

Forking Types - Soft Fork

- Temporary fork in the blockchain due to updating the consensus protocol to include additional rules on validating the blocks.
    - Generally, soft forks are related to adopting stricter rules to validate blocks/transactions.
- Why is it called soft?
    - Blocks considered valid by an old version of the protocol are not all valid by the new version.
    - But blocks considered valid by the new version are all valid based on the old version.
- If the majority of the nodes switch to the new version of the protocol, the old nodes will switch eventually since many of their mined blocks will be dropped.
    - Remember we assume honest majority.

Soft Fork - Pictorially

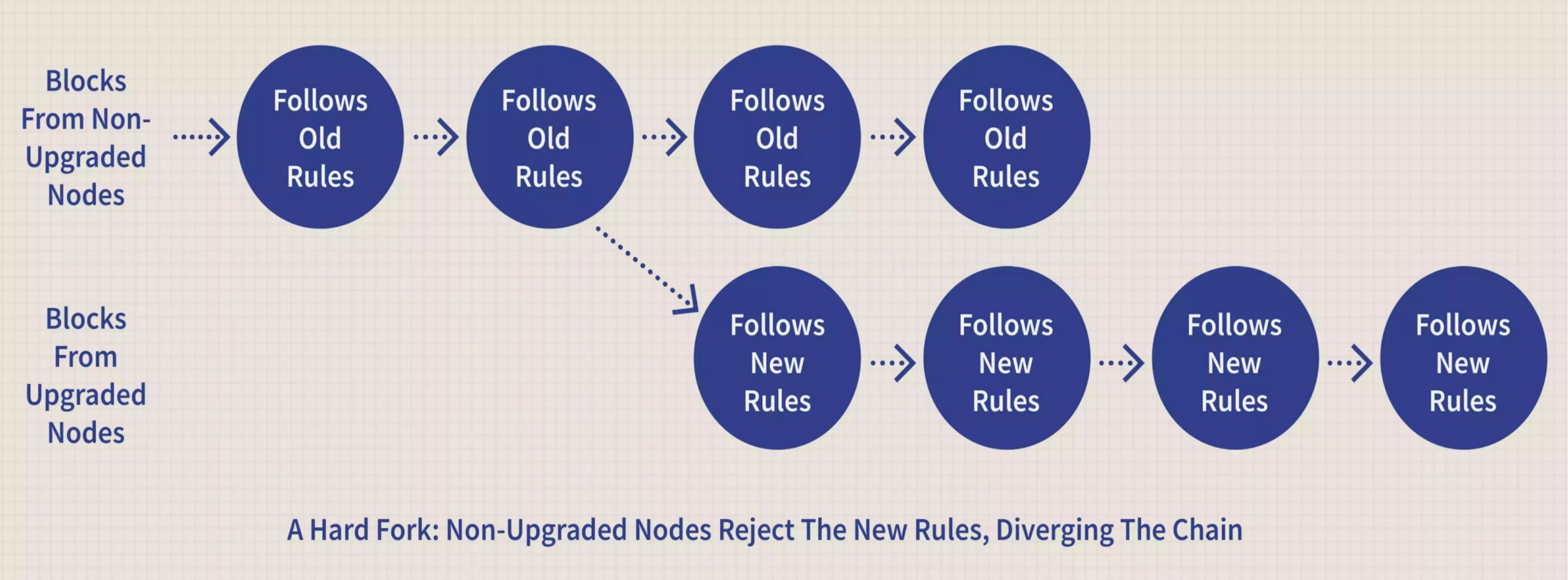

Blocks From Non-Upgraded Nodes
Follows Old Rules
Follows Old Rules
Follows Old Rules But Violates New Rules
Follows Old Rules
Blocks From Upgraded Nodes
Follows Old & New Rules
Follows Old & New Rules
A Soft Fork: blocks violating new rules are made stale by the upgraded mining majority

From https://www.investopedia.com/terms/s/soft-fork.asp

Forking Types - Hard Fork

- Permanent fork in the blockchain due to core changes in the consensus protocol.
- Why is it called hard?
  - All blocks that are valid according to the new version are considered invalid by the old protocol version.
  - Thus, the two branches will not have any blocks/transactions in common.
  - Results in two different blockchains.
- So, a miner can be on one branch (or basically a blockchain) but not on both.
- Historically, Bitcoin experienced several hard forks, e.g., Bitcoin Cash, Bitcoin Gold, Bitcoin Satoshi Version (SV).

Hard Fork - Pictorially



From https://www.investopedia.com/terms/h/hard-fork.asp

Forking Types - Velvet Fork

- A conditional soft fork.
  - Additional validity rules for transactions and blocks are applied when certain conditions are met.
  - If such conditions are not met, then the new additional rules are ignored.
- Usually used for improving protocol design but without producing two different blockchains or forcing miners to upgrade their versions of the network protocol.
  - Called backward-compatible since again blocks/transactions under the new version are also valid under the old version of the protocol.

Blockchain (Bitcoin) Attractive Properties

Apart from decentralization and the open-access work model that we already covered

Take notes ;)

Transparency

- What does that mean?
- How does Bitcoin achieve it?

Public Verifiability

- What does that mean?
- How does Bitcoin achieve it?
- Can public verifiability and transparency achieve a sense of accountability in distributed systems?

Immutability

- What does that mean?
- How does Bitcoin achieve it?

Sybil-attack Resistance

- What are Sybil attacks?
- Why do they matter in Bitcoin?
- How is Bitcoin resistant to Sybil attacks?

Asynchronous Broadcast Channel

- What does that mean?
- How does Bitcoin achieve it?

Bitcoin Scripting Language

Validating Transactions

- Involves validating/checking:
    - The format of a transaction (including that total output value does not exceed the total input value),
    - and that the inputs can be spent to the outputs, i.e., the transaction issuer owns the inputs.
- The latter is done in a programmable way using Bitcoin scripting language.
    - This allows for greater flexibility and introduces the notion of ***programmable money***.

Bitcoin Scripting Language

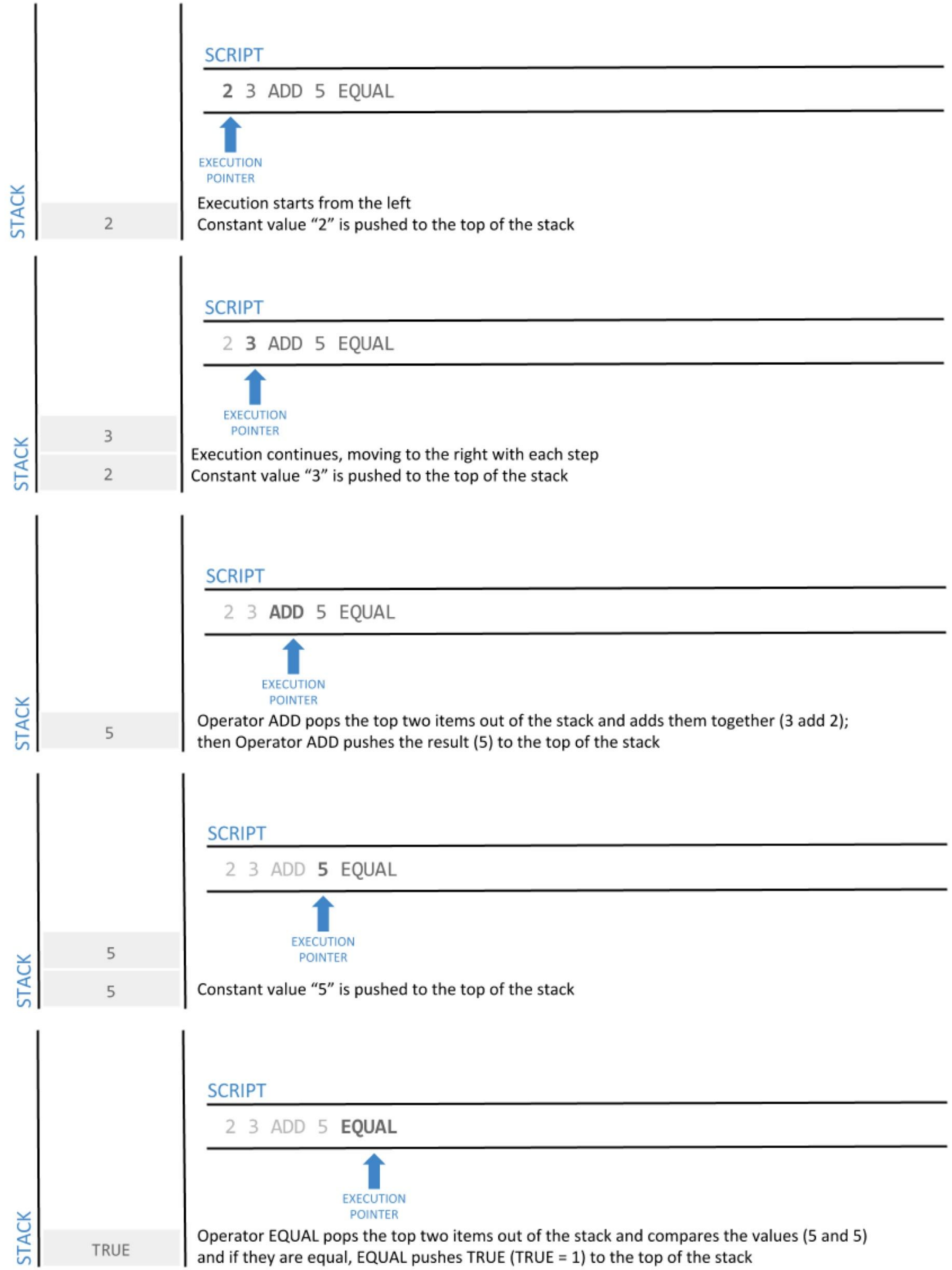
- Non Turing-complete, does not support loops.
  - Limited complexity and it has a predictable execution time.
  - Stack based.
- Kept simple for security reasons.
  - More complex scripting languages, or better saying Turing-complete, provide greater flexibility for the programmer to build complicated functionalities.
  - It is hard to get it right!! Writing fully secure scripts or programs is not easy.
- Attackers are financially motivated to dig into these programs and find security bugs.

Script Construction

- Two parts: unlocking and locking scripts.
  - **Locking:** specify conditions that when met a given input (aka coins) can be spent.
  - **Unlocking:** a proof that the conditions have been met (i.e., provide inputs for the locking script to unlock it).
- Thus, a transaction has an unlocking script for each of its inputs that is processed alongside a locking script for the output of the referenced input transaction.
  - Recall that an input for a (new) transaction is an unspent output from a previous transaction.
  - The concatenated unlocking and locking scripts have to evaluate to **TRUE** in order to allow spending the coins.

Stack-based Scripting

- A clarifying example from “Mastering Bitcoin” book, Chapter 7.
- Locking and Unlocking scripts will be written similarly.



Execution starts from the left
Constant value “2” is pushed to the top of the stack

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

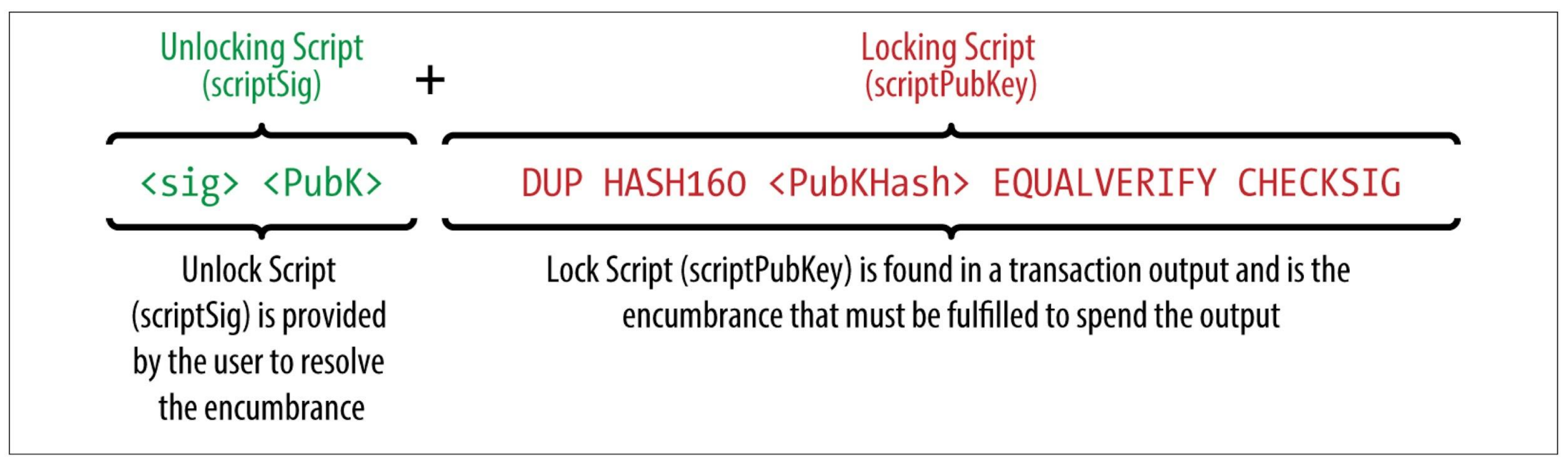
Operator ADD pops the top two items out of the stack and adds them together (3 add 2);
then Operator ADD pushes the result (5) to the top of the stack

Constant value “5” is pushed to the top of the stack

Operator EQUAL pops the top two items out of the stack and compares the values (5 and 5)
and if they are equal, EQUAL pushes TRUE (TRUE = 1) to the top of the stack

Script Construction - P2PKH

- Most popular transaction type in Bitcoin is pay to public key hash.
  - It means sending coins to some public key.



Unlocking Script (scriptSig) + Locking Script (scriptPubKey)

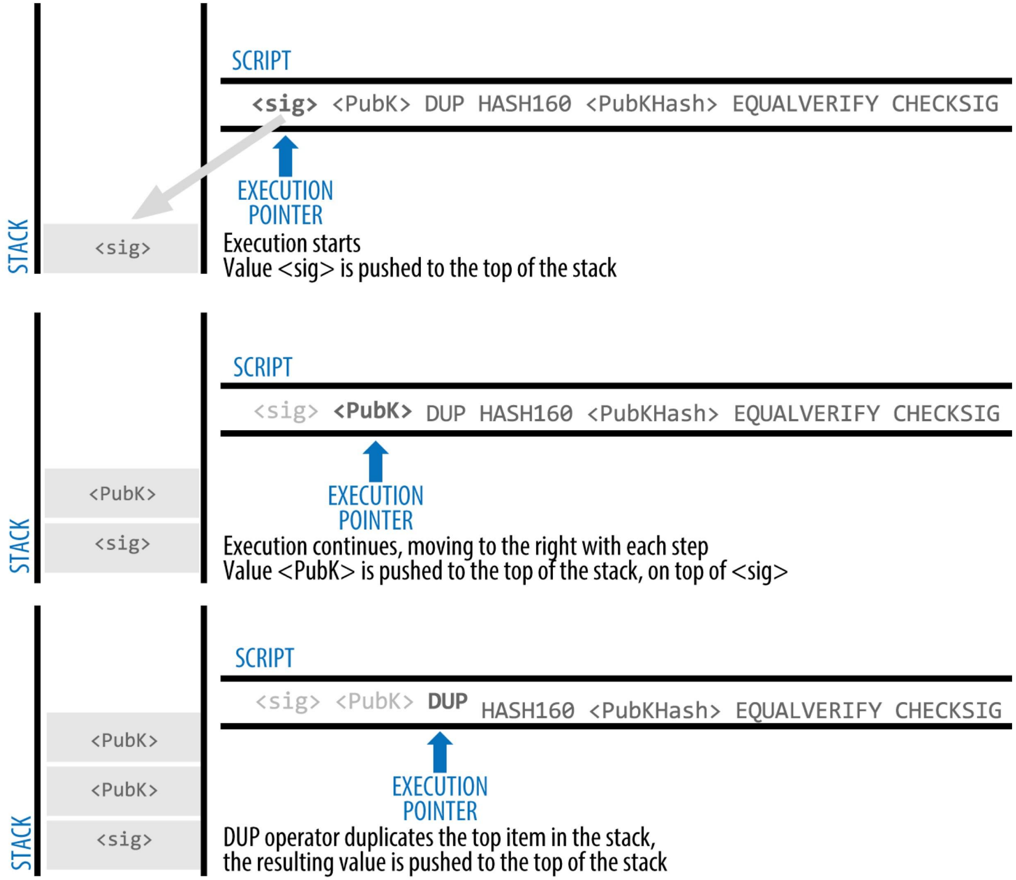
<sig> <PubK> DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG

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

Figure from "Mastering Bitcoin" book, Chapter 7.

P2PKH Script Evaluation I


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>

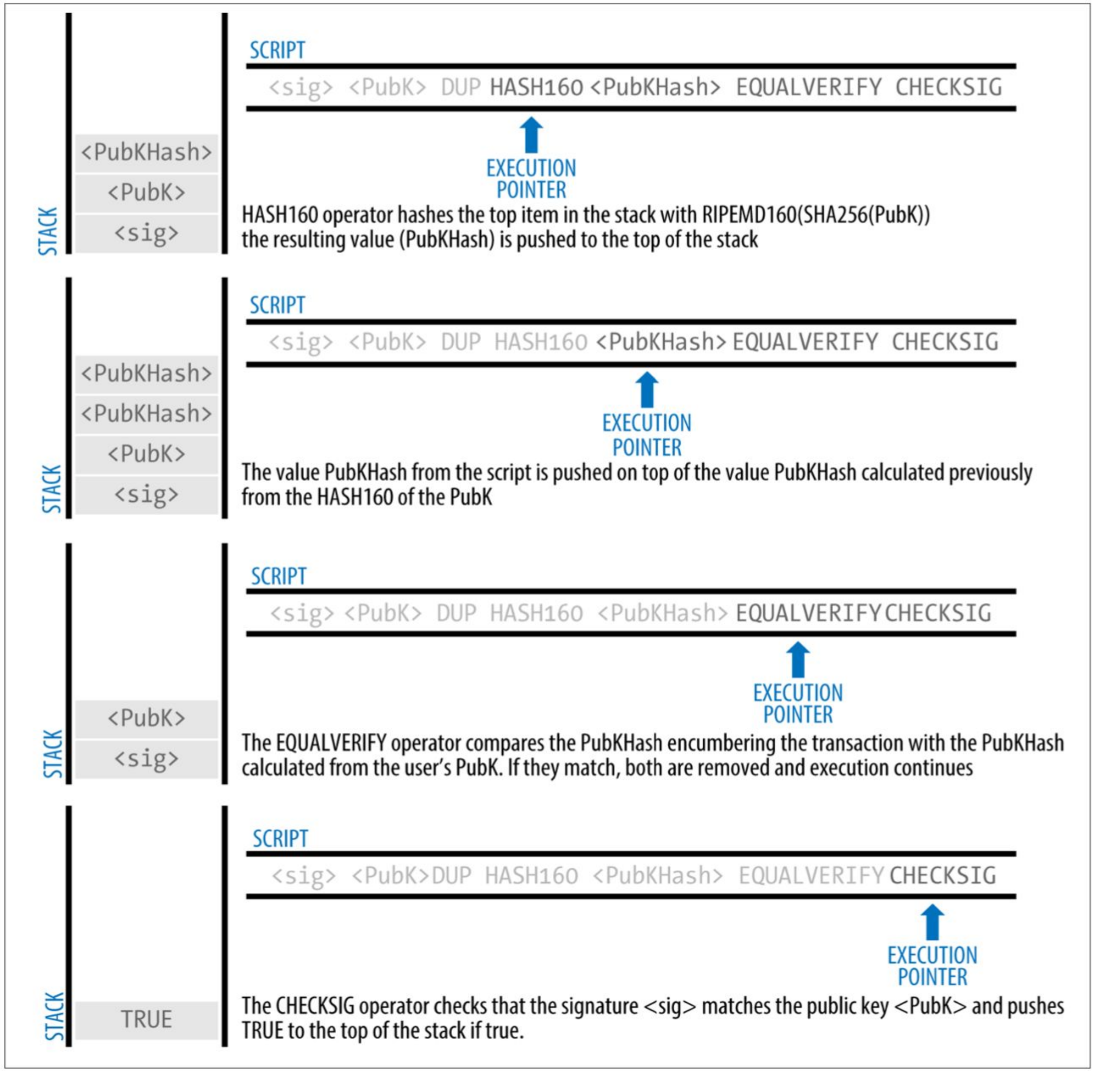
Execution continues, moving to the right with each step
Value <PubK> is pushed to the top of the stack, on top of <sig>

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

Figure from “Mastering Bitcoin” book, Chapter 7.

P2PKH Script Evaluation II


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

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

Bitcoin Standard Transactions

- **Pay to public key hash (P2PKH).**
  - Vast majority of Bitcoin transactions are of this type.
  - X pays Y a Z value of Bitcoins.
- **Pay to public key.**
  - Same as above but instead of using addresses (hashed public keys), use the public key itself.
  - Hashed public keys are more efficient as they are shorter.
- **Data output.**
  - Use OP_RETURN to store up to 40 byte data on the blockchain (e.g., document timestamping).
- **Pay to script hash.**
- **Pay to multi-signature.**
  - More about the above two in the next slides.

Pay to Multi-signature (P2MS)

- One of the very useful and widely implemented scripts in P2SH.
- The script requires signatures from multiple users to unlock the currency instead of one signature from one user.
- Can be built also in a threshold based way, like 2 out of 3 signatures are enough to spend the currency.
  - Up to 3 signatories are allowed, however, if P2SH (will be studied shortly) is used instead, then up to 15 signatories are allowed.
- Mostly used to create escrows.

P2MS - An Example

- Locking, unlocking, and concatenated scripts for a 2 out of 3 multisig transaction (from “Mastering Bitcoin”, Chapter 5).

```
2 <Public Key A> <Public Key B> <Public Key C> 3 CHECKMULTISIG
```

```
<Signature B> <Signature C>
```

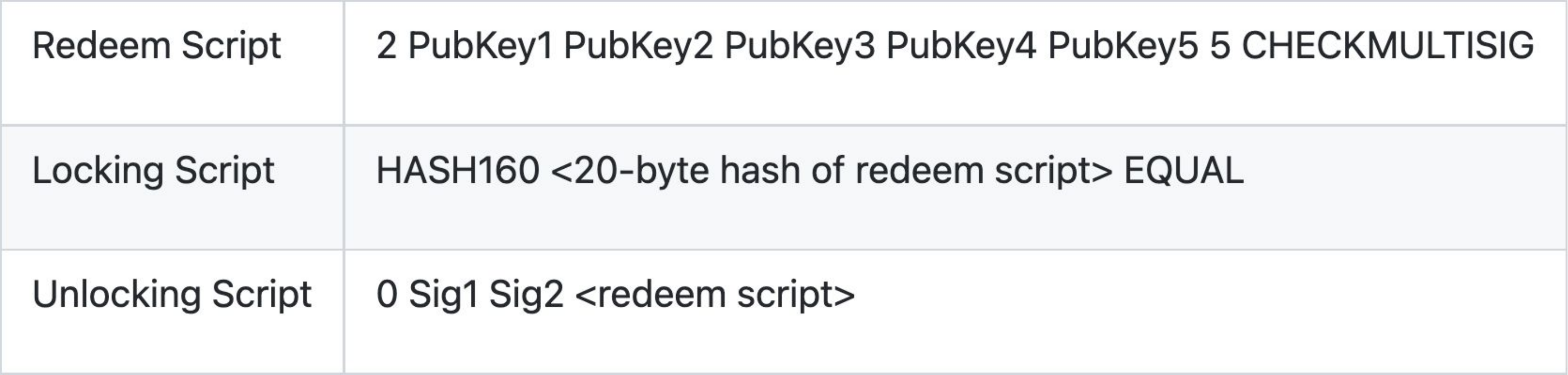
```
<Signature B> <Signature C> 2 <Public Key A> <Public Key B> <Public Key C> 3 CHECKMULTISIG
```

Note: Due to a bug in the implementation of the CHECKMULTISIG opcode, an extra dummy input is needed in front of the unlocking script. Usually the value 0 or OP_0 is placed there. We will ignore that during the course.

Pay to Script Hash (P2SH)

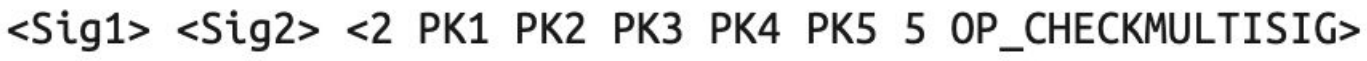
- Provides ways to implement advanced operations in Bitcoin beyond the standard currency transfer transactions.
- The address is the hash of some script, thus, these addresses start with 3 to differentiate them from normal addresses.
- To spend the currency locked under the script hash address you must present an unlocking script that makes this locking script evaluate to TRUE.
    - If the result is indeed true the currency is transferred to the destination address you specify.
- The scripts that you can code are limited by the primitives/opcodes supported in Bitcoin Scripting language (check https://en.bitcoin.it/wiki/Script).

Pay to Script Hash (P2SH) - Example



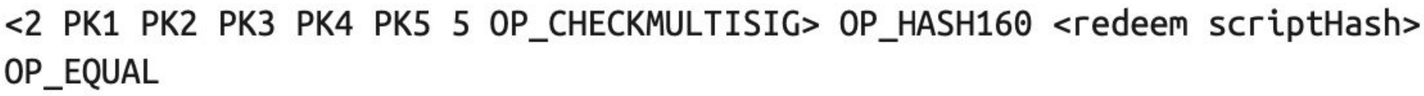
Redeem Script	2 PubKey1 PubKey2 PubKey3 PubKey4 PubKey5 5 CHECKMULTISIG
Locking Script	HASH160 <20-byte hash of redeem script> EQUAL
Unlocking Script	0 Sig1 Sig2 <redeem script>

- To spend it, one presents:



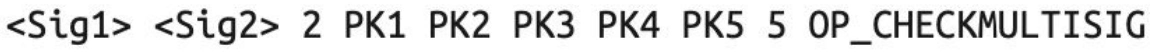
```
<Sig1> <Sig2> <2 PK1 PK2 PK3 PK4 PK5 5 OP_CHECKMULTISIG>
```

- The transaction is executed in two stages: First the script hash is



```
<2 PK1 PK2 PK3 PK4 PK5 5 OP_CHECKMULTISIG> OP_HASH160 <redeem scriptHash>
OP_EQUAL
```

- Then, second, the script is checked to evaluate to TRUE (I am ignoring the 0



```
<Sig1> <Sig2> 2 PK1 PK2 PK3 PK4 PK5 5 OP_CHECKMULTISIG
```

Pay to Script Hash (P2SH) - Note

- Note that verifying the redeem script hash will remove the redeem script itself from the stack.
    - To allow evaluating the script on the input in the second phase, the miner will create a copy of the script to be used for later.
    - In particular, it will create a copy of the stack state, and if script hash verification outputs 1, the stack is reloaded with the older state containing the redeem script.