Goal of this lecture note

- Bitcoin Script
- Tables of OP Codes
- Easy Script
- Pay-to-Public Key Hash (P2PKH) Script
- Multisignature and Smart Contracts Scripts

1 Bitcoin Script

- Bitcoin Script
  - Bitcoin uses a scripting language for transactions.
  - A script is simple, stack-based, and processed from left to right.
  - It is intentionally not Turing-complete, with no loops.
  - A script is a list of instructions.
  - The payer locks the vout value to a payee's public address.
  - The payee unlocks the lock by providing the signature.

1 Bitcoin Script

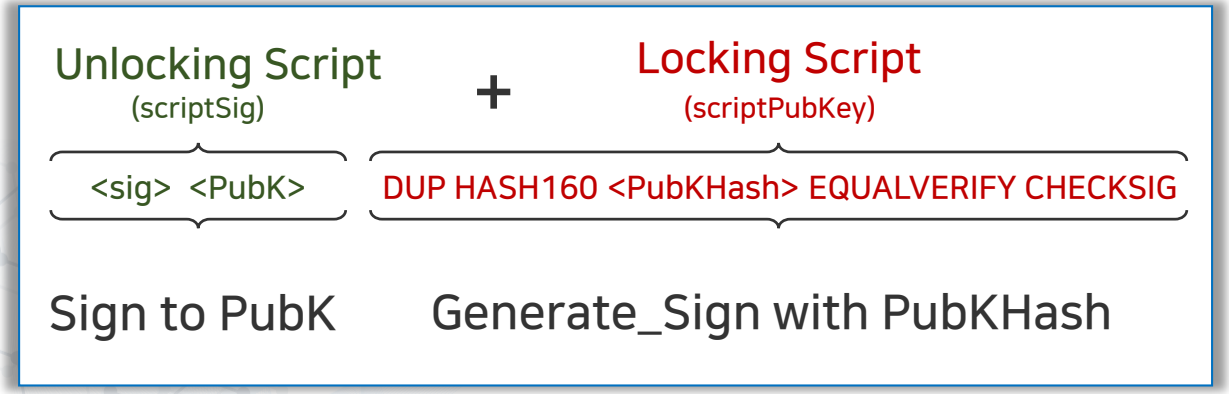
- Bitcoin Script
  - Payer uses a lock script to lock the `vout` value to a destination Bitcoin address and payee uses an unlock script to spend it.
    1. The `vout` value transferred to a destination address mapped from a public key is locked into the locking script, and
    2. A signature is embedded in the unlocking script which proves the ownership of the private key corresponding to the locked value.
  - Further reading from https://en.bitcoin.it/wiki/Script

1 Bitcoin Script

- See if scriptSig unlocks scriptPubKey!
  - *Script Construction (Unlock+Lock)*
    - The locking script is called a *scriptPubKey*, because it contains a public key or a Bitcoin address.
    - The unlocking script is called *scriptSig* because it contains a digital signature.
    - When a correct unlocking script is provided to the locking script, the execution of the complete script comes out TRUE.
    - Then, the provider of scriptSig can spend the value.

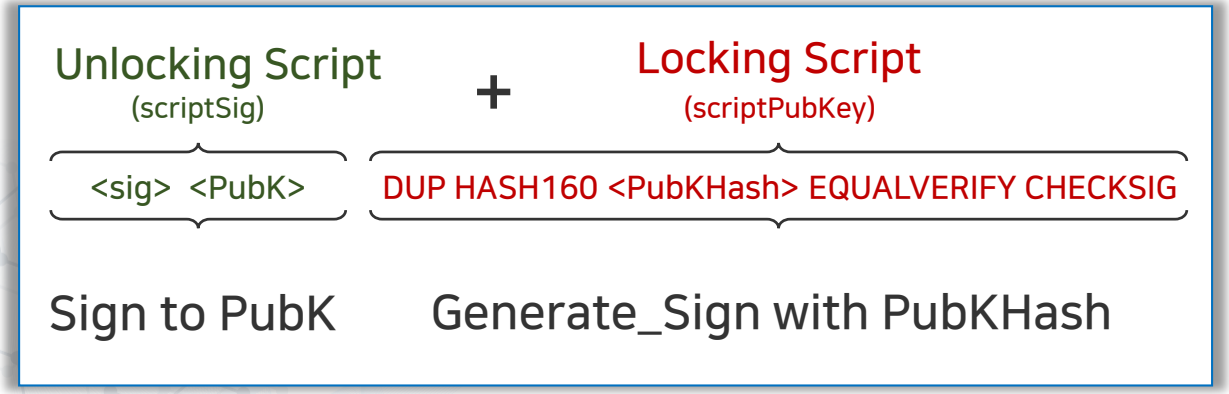
1 Bitcoin Script

- Pay to Public Key Hash
  - 시간 1: A's Sign (Priv. Key)→ Lock to Pub. Key of B 2.0BTC.
  - 시간 2: B's Sign (Priv. Key)→ Lock to Pub. Key of C 1.0BTC.
  - 시간 3: C's Sign (Priv. Key)→ Lock to Pub. Key of D 0.5BTC.



1 Bitcoin Script

- Values provided by users are given in < >.
- DUP, HASH160, EQUALVERIFY, CHECKSIG are Operations.



2 Tables of OP Codes

- Table C-7. Cryptographic and Hashing Operations

Symbol	Value(hex)	Description
OP_RIPEMD160	0xa6	Return RIPEMD160 hash of top item
OP_SHA1	0xa7	Return SHA1 hash of top item
OP_SHA256	0xa8	Return SHA256 hash of top item
OP_HASH160	0xa9	Return RIPEMD160(SHA256(x)) hash of top item
OP_HASH256	0xaa	Return SHA256(SHA256(x)) hash of top item
OP_CODESEPARATOR	0xab	Mark the beginning of signature-checked data

Appendix C of Mastering Bitcoin

2 Tables of OP Codes

- Table C-7. Cryptographic and Hashing Operations

Symbol	Value(hex)	Description
OP_CHECKSIG	0xac	Pop a public key and signature and validate the signature for the transaction's hashed data, return TRUE if matching
OP_CHECKSIGVERIFY	0xad	Same as CHECKSIG, then OP_VERIFY to halt if not TRUE
OP_CHECKMULTISIG	0xae	Run CHECKSIG for each pair of signature and public key provided. All must match. Bug in implementation pops an extra value, prefix with OP_NOP as workaround
OP_CHECKMULTISIGVERIFY	0xaf	Same as CHECKMULTISIG, then OP_VERIFY to halt if not TRUE

Appendix C of Mastering Bitcoin

2 Tables of OP Codes

- Table C-3. Stack Operations

Symbol	Value(hex)	Description
OP_TOALTSTACK	0x6b	Pop top item from stack and push to alternative stack
OP_FROMALTSTACK	0x6c	Pop top item from alternative stack and push to stack
OP_2DROP	0x6d	Pop top two stack items
OP_2DUP	0x6e	Duplicate top two shack items
OP_3DUP	0x6f	Duplicate top three shack items
OP_2OVER	0x70	Copies the third and fourth items in the stack to the top
OP_2ROT	0x71	Moves the fifth and sixth items in the stack to the top
OP_2SWAP	0x72	Swap the two top pairs of items in the stack
OP_IFDUP	0x73	Duplicate the top item in the stack if it is not 0
OP_DEPTH	0x74	Count the items on the stack and push the resulting count

Appendix C of Mastering Bitcoin

2 Tables of OP Codes

- Table C-3. Stack Operations

Symbol	Value(hex)	Description
OP_DROP	0x75	Pop the top item in the stack
OP_DUP	0x76	Duplicate the top item in the stack
OP_NIP	0x77	Pop the second item in the stack
OP_OVER	0x78	Copy the second item in the stack and push it on to the top
OP_PICK	0x73	Pop value N from top, then copy the Nth item to the top of the stack
OP_ROLL	0x7a	Pop value N from top, then move the Nth item to the top of the stack
OP_ROT	0x7b	Rotate the top three items in the stack
OP_SWAP	0x7c	Swap the top three items in the stack
OP_TUCK	0x7d	Copy the top item and insert it between the top and second item

Appendix C of Mastering Bitcoin

2 Tables of OP Codes

- Table C-6. Numeric Operators

Symbol	Value(hex)	Description
OP_1ADD	0x8b	Add 1 to the top item
OP_1SUB	0x8c	Subtract 1 from the top item
OP_2MUL	0x8d	Disabled (Multiply top item by 2)
OP_2DIV	0x8e	Disabled (Divide top item by 2)
OP_MEGATE	0x8f	Flip the sign of top item
OP_ABS	0x90	Change the sign of the top item to positive
OP_NOT	0x91	If top item is 0 or 1 boolean flip it, otherwise return 0
OP_ONOTEQUAL	0x92	If top item is 0 return 0, otherwise return 1
OP_ADD	0x93	Pop top two items, add them and push result

Appendix C of Mastering Bitcoin

2 Tables of OP Codes

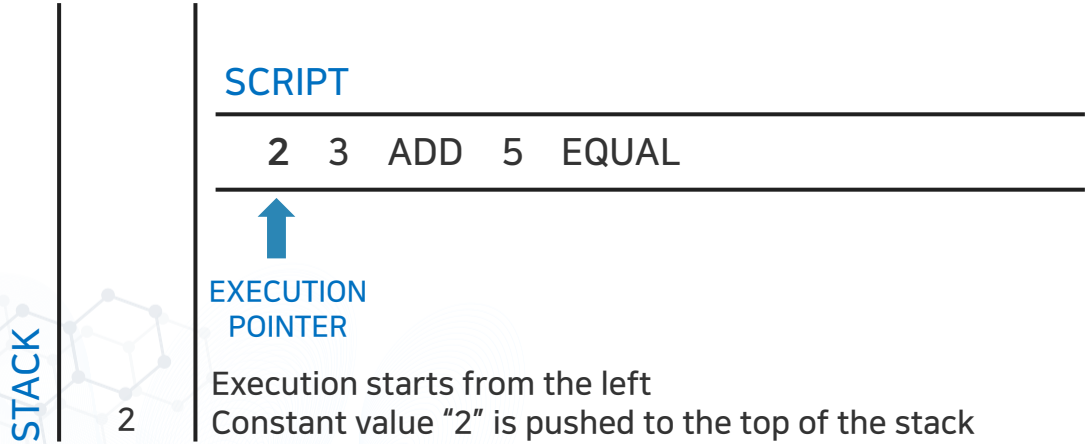
- Table C-2. Conditional Flow Control

Symbol	Value(hex)	Description
OP_NOP	0x61	Do nothing
OP_VER	0x62	Halt – Invalid transaction unless found in an unexecuted OP-IF clause
OP_IF	0x63	Execute the statements following if top of stack is not 0
OP_NOTIF	0x64	Execute the statements following if top of stack is 0
OP_VERIF	0x65	Halt – Invalid transaction
OP_VERMPTIF	0x66	Halt – Invalid transaction
OP_ELSE	0x67	Execute only if the previous statements were not executed
OP_ENDIF	0x68	Ends the OP_IF, OP_NOTIF, OP_ELSE block
OP_VERIFY	0x69	Check the top of the stack, Halt and Invalidate transaction if not TRUE

Appendix C of Mastering Bitcoin

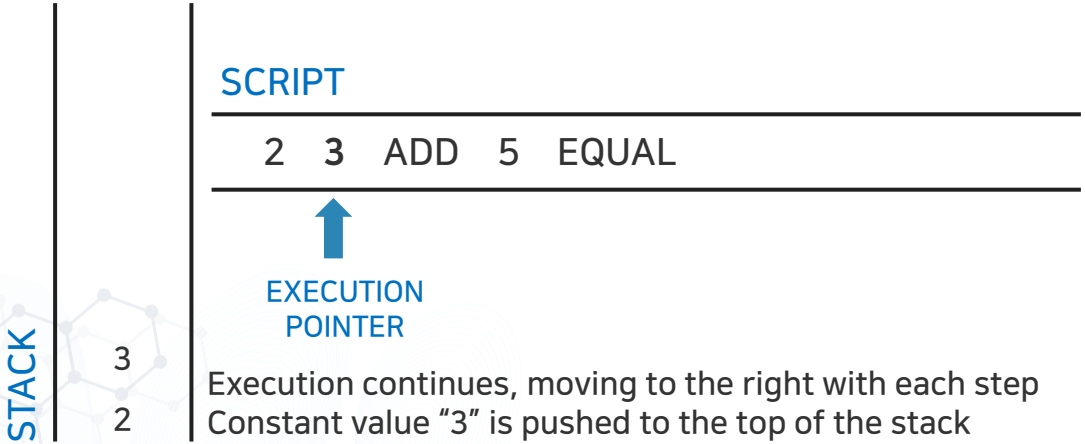
3 Easy Script

- Example script: 2 + 3 = 5



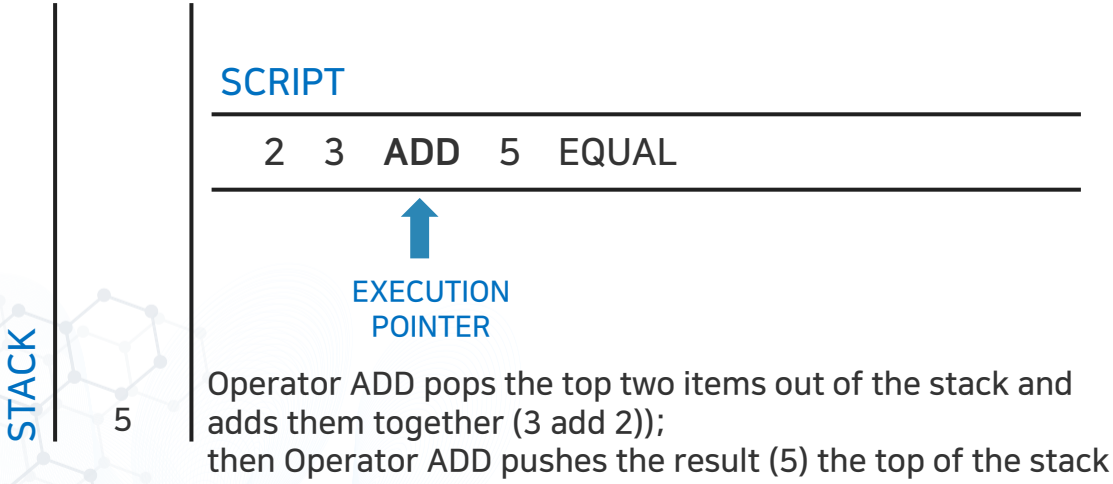
3 Easy Script

- Example script: 2 + 3 = 5



3 Easy Script

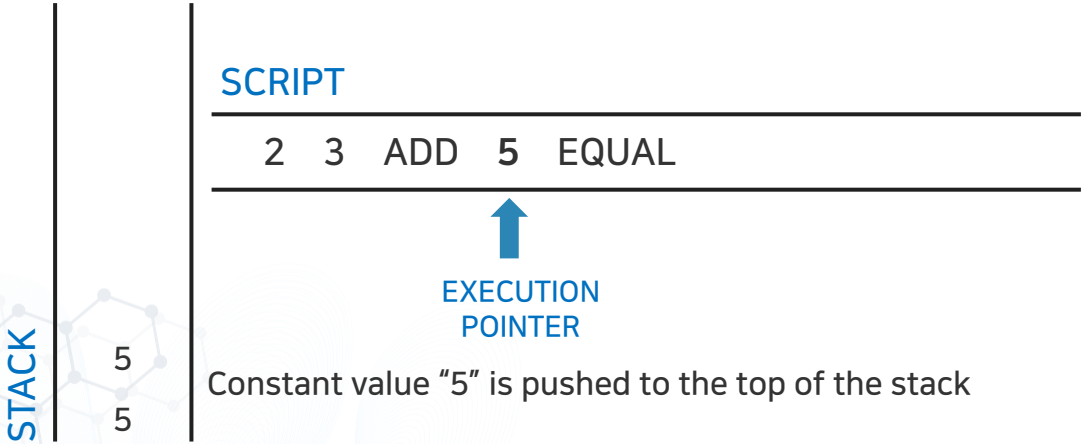
- Example script: 2 + 3 = 5



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) the top of the stack

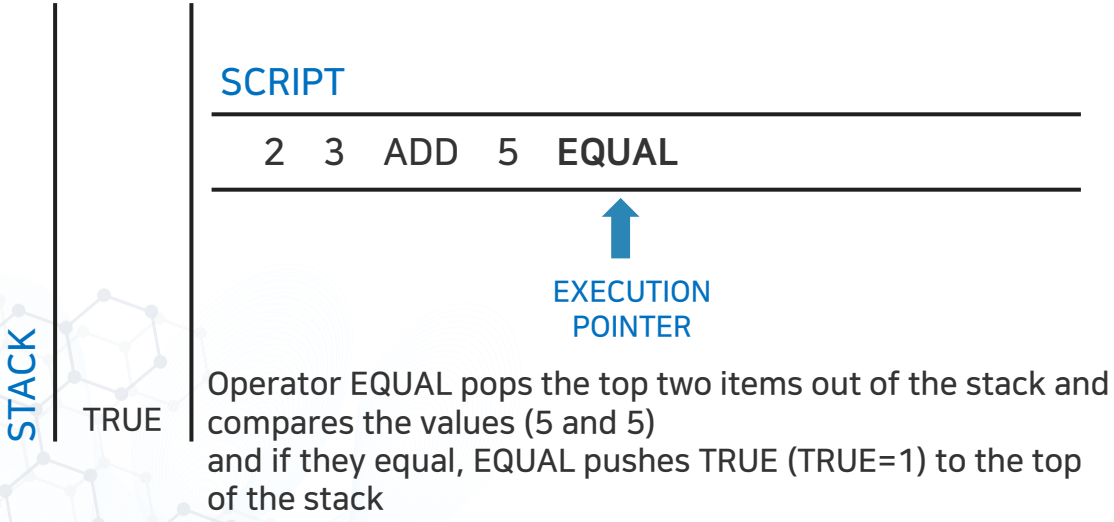
3 Easy Script

- Example script: 2 + 3 = 5



3 Easy Script

- Example script: 2 + 3 = 5



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

3 Easy Script

- Unlock + Lock Pair, shows a proof of ownership
  - Use a part of the arithmetic example script as the locking script:

    ```
    3 OP_ADD 5 OP_EQUAL
    ```

  - Which can be satisfied by a transaction containing an input with the unlocking script:

    ```
    2
    ```

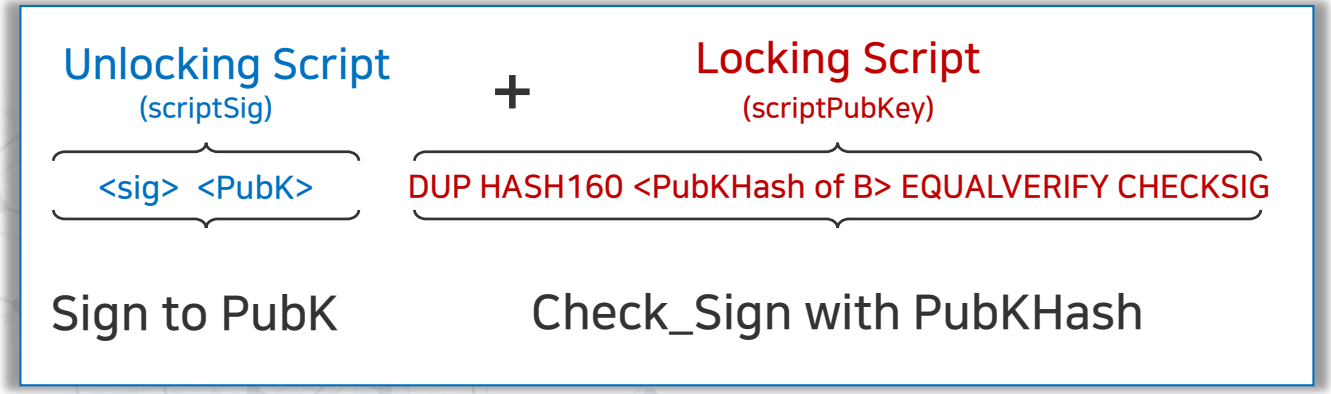
  - Put them together, we have the complete script.

    ```
    2 3 OP_ADD 5 OP_EQUAL
    ```

  - This pair will produce an outcome of TRUE.

4 P2PKH Script

- Now let us make a more realistic pair focusing on B.
  - 시간 1: A's Sign (Priv. Key)→ Lock to Pub. Key of B 2.0BTC.
  - 시간 2: B's Sign (Priv. Key)→ Lock to Pub. Key of C 1.0BTC.
  - the signature.



4 P2PKH Script

- P2PKH of B
  - Unspent value belongs to Pay to Public Key Hash(P2PKH) script.

```
OP_DUP OP_HASH160 <Public Key Hash of B> OP_EQUAL OP_CHECKSIG
```

  - Unlocking script is a digital sign created by corresponding private key.

```
<sig of B>  <PubK of B>
```

4 P2PKH Script

- Locking script with a single <input>
  - One input, four operations
  - `OP_DUP`: duplicate
  - `OP_Hash160`(x)= `RIPEMD`(`SHA256`(x))
  - `<Public Key Hash of B>`
  - `OP_EQUAL`: return TRUE if the two top most values are equal
  - `OP_CHECKSIG`: checks to see if the provided `sign` and `pubkey` are valid

4 P2PKH Script

- Locking script with <input>

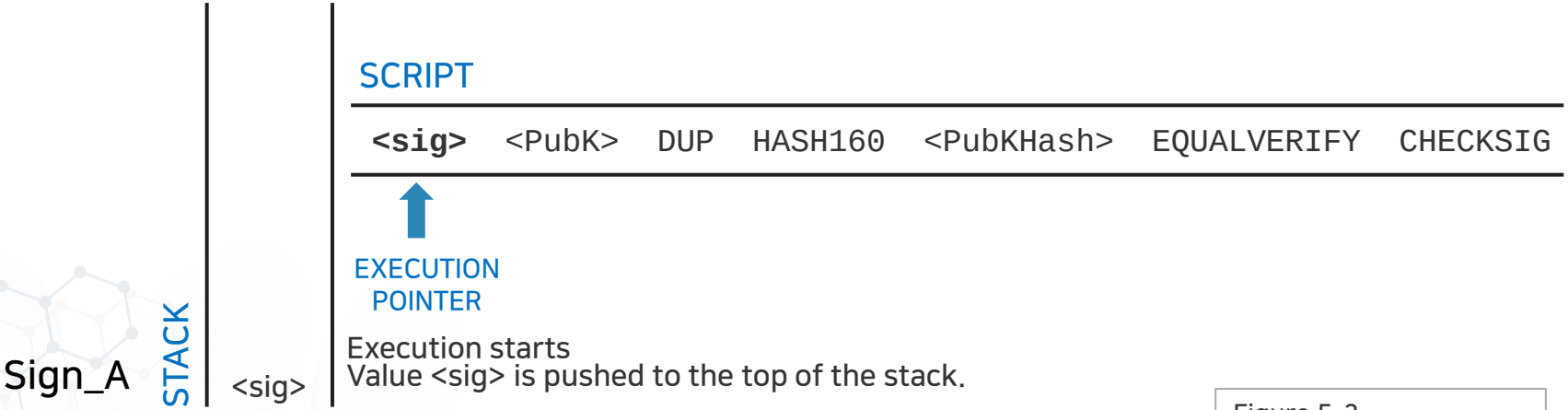
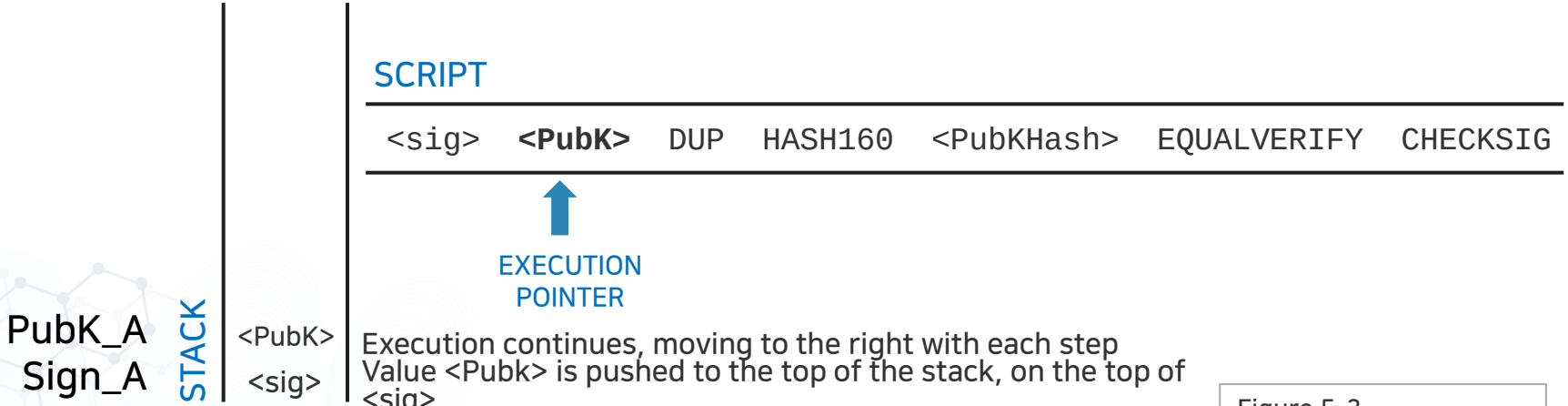


Figure 5-3.
Evaluating a script for a Pay-to-Public-Key-Hash transaction (Part 1 of 2)

4 P2PKH Script

- Locking script with <input>



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

Figure 5-3.
Evaluating a script for a Pay-to-Public-Key-Hash transaction (Part 1 of 2)

4 P2PKH Script

- Locking script with <input>

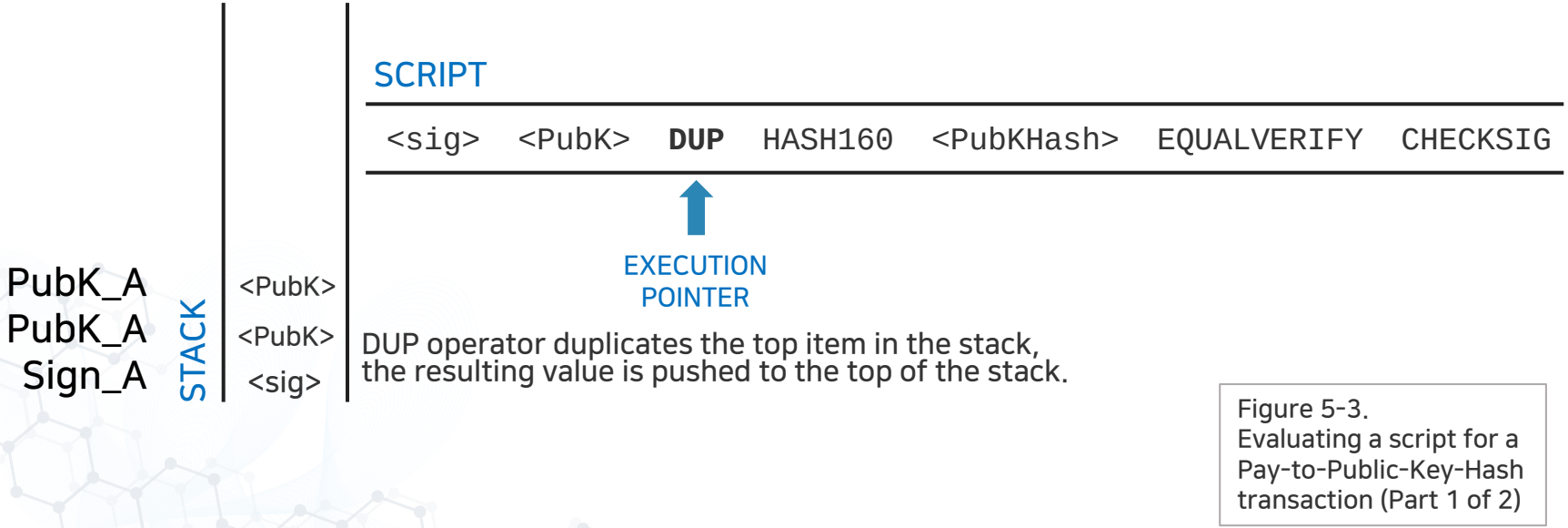


Figure 5-3.
Evaluating a script for a Pay-to-Public-Key-Hash transaction (Part 1 of 2)

4 P2PKH Script

- See if two PubKH_As match

PubKH_A
PubK_A
Sign_A

STACK

<PubK Hash>
<PubK>
<sig>

SCRIPT

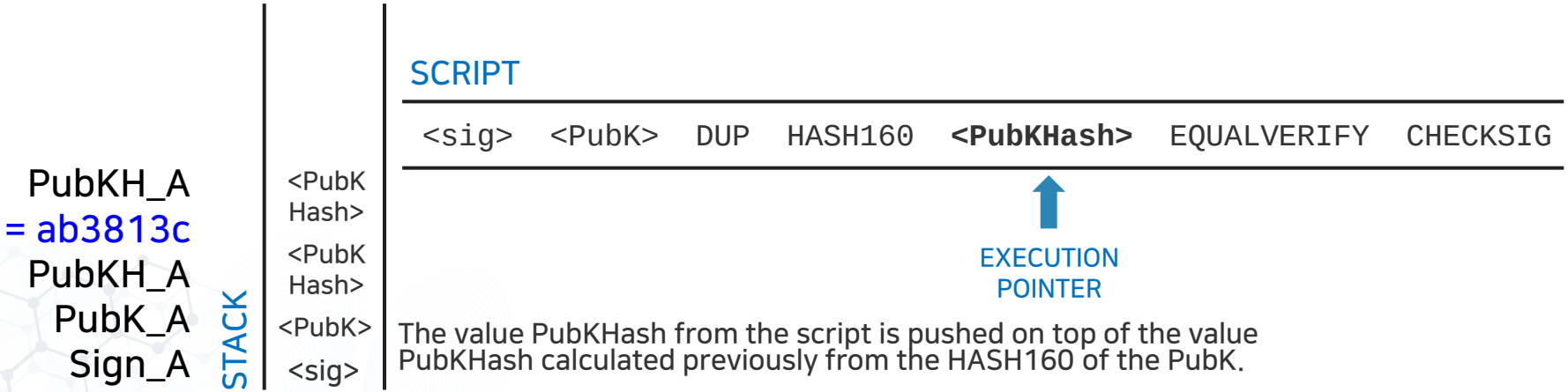
<sig> <PubK> DUP **HASH160** <PubKHash> EQUALVERIFY CHECKSIG

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.

4 P2PKH Script

- See if the two PubKH_As match



4 P2PKH Script

- Check Signature

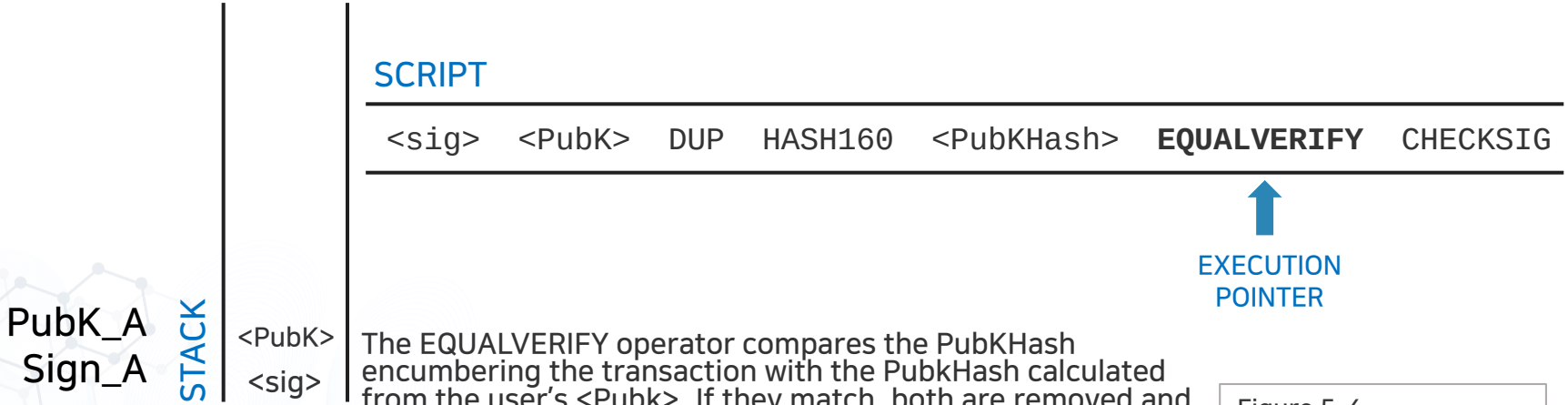


Figure 5-4.
Evaluating a script for a Pay-to-Public-Key-Hash transaction (Part 2 of 2)

4 P2PKH Script

- Recall `SignGenerate` and `isSignatureValid` routines
  - m = {`TXID`, output [n] = {value, a locking script with `PKH_A`}}
  - `Sign_A = SignGenerate (`m`, k_A);`
  - `isSignatureValid(`m`, Sign_A, PK_A) = TRUE/False`

4 P2PKH Script

- Check Signature

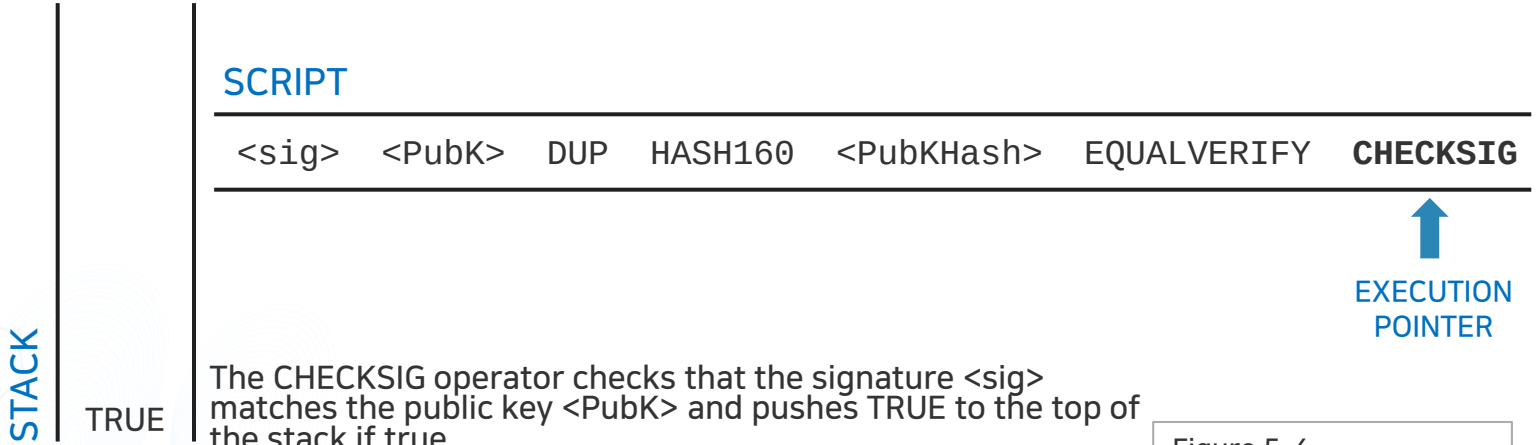


Figure 5-4.
Evaluating a script for a Pay-to-Public-Key-Hash transaction (Part 2 of 2)

5 Multisignature and Smart Contracts Scripts

- Other Scripts
  - Pay to Public Key (P2PK), introduced in the Bitcoin white paper.
  - Pay to Public Key Hash (P2PKH), used in the code by Satoshi Nakamoto.
  - Pay to Script Hash (P2SH), introduced winter of 2012.
    - These Bitcoin addresses are beginning with 3.
    - Hash of a script is the beneficiary.
    - It can be used for a `multisignature` script.
      - M out of N keys are needed to spend the value.
      - Useful for joint accounts

5 Multisignature and Smart Contracts Scripts

- Bitcoin uses scripts for Smart Contracts
  - There are many different possibilities that can be expressed with this scripting language.
  - Smart contracts can be programmed in to code which expresses more complex conditions for spending and how these conditions can be satisfied by unlocking scripts.
  - *This language allows for a nearly infinite variety of conditions to be expressed.*
  - *This is how bitcoin gets the power of "programmable money." (Mastering Bitcoin)*

5 Multisignature and Smart Contracts Scripts

- Bitcoin does not allow any loop for stable operations.
- Ethereum does.
  - `Jump` and `JumpTo` are used in the list of OP codes.
  - https://github.com/crytic/evm-opcodes.
- Bitcoin is more prudent and focuses on safety.