# Bitcoin Script

## Technical overview

ndhy@fit.hcmus.edu.vn

Knowledge Engineering Department

December 19, 2024

# Introduction to Bitcoin Script

- Bitcoin Script is a stack-based programming language used by Bitcoin.
- It is not Turing complete, which increases security by preventing infinite loops.
- Used for defining transaction processing rules.

# Structure of a Bitcoin Transaction

- **Transaction Input:** References an existing unspent transaction output (UTXO).
- **Transaction Output:** Contains the amount and a locking script (`scriptPubKey`) defining spending conditions.
- **Unlocking Script (`scriptSig`):** Provided by the spender to satisfy the conditions of the locking script.

# Bitcoin Script Basics

- **Stack Operations:** Operations manipulate data on a stack.
- **Opcodes:** Instructions that perform various operations.
- Examples:
  - `OP_DUP`
  - `OP_HASH160`
  - `OP_EQUALVERIFY`
  - `OP_CHECKSIG`

# Example Script Execution

## Funding Transaction (scriptPubKey)

```
OP_DUP OP_HASH160 <PubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
```

- OP_DUP: Duplicates the top stack item.
- OP_HASH160: Hashes the top stack item twice.
- <PubKeyHash>: Public key hash from the recipient.
- OP_EQUALVERIFY: Verifies that the top two stack items are equal, removes them if true.
- OP_CHECKSIG: Verifies the transaction signature.

# Example Script Execution

## Spending Transaction (`scriptSig`)

```
<Signature> <PublicKey>
```

- `<Signature>`: Signature created by the sender's private key.
- `<PublicKey>`: Sender's public key.

# Step-by-Step Verification Flow

1. **Initialization:**
   - Combine the `scriptSig` and `scriptPubKey`.
   - Resulting script: `<Signature> <PublicKey> OP_DUP OP_HASH160 <PubKeyHash> OP_EQUALVERIFY OP_CHECKSIG`.
2. **Execution:**
   - Push `<Signature>` onto the stack.
   - Push `<PublicKey>` onto the stack.
   - `OP_DUP`: Duplicate the public key.
   - `OP_HASH160`: Hash the duplicated public key.
   - Push `<PubKeyHash>` (from script) onto the stack.
   - `OP_EQUALVERIFY`: Check if hashes are equal.
   - `OP_CHECKSIG`: Verify the signature.

# Script Execution Example

## Detailed Steps

- **scriptPubKey:**
  `OP_DUP OP_HASH160`
  `ab6809ae7b19d5f785e899017cfa6dd1e30f8c3f OP_EQUALVERIFY`
  `OP_CHECKSIG`
- **scriptSig:**
  `3045022100d0a8b046f7e465fa02b9cba8f0-`
  `90b1f8b76b0b6f93eacff33b8bcf84d5767e-`
  `1d02202e5044313fa76ebcaddf49e7a50947-`
  `2abf1d6b21f53101d5f243a2ff7a0d2688`
  `04b0bd634234abbb1ba1e986e884185c6b38-`
  `90bc5f40e6db70a0ad34a4f5a02c276de0b0-`
  `b91a0da7e6d23243adfc6cdd99d0d96a50a6-`
  `1138a334ce37e3a4d1b8b`

# Script Execution Example

## Execution Steps

1. Push `3045022100d0a8b046...` (signature) onto the stack.
2. Push `04b0bd6342...` (public key) onto the stack.
3. `OP_DUP`: Duplicate the public key.
   - Stack: `[signature, public key, public key]`
4. `OP_HASH160`: Hash the duplicated public key.
   - Stack: `[signature, public key, hash160(public key)]`

# Script Execution Example

## Execution Steps

1. Push `ab6809ae7b19d5f785e899017cfa6dd1e30f8c3f` (expected hash) onto the stack.
   - Stack: `[signature, public key, hash160(public key), ab6809ae7b19d5f785e899017cfa6dd1e30f8c3f]`
2. `OP_EQUALVERIFY`: Compare the two topmost items.
   - If equal, continue. Stack: `[signature, public key]`
   - If not, script execution fails.
3. `OP_CHECKSIG`: Verify the signature using the public key.
   - If valid, the script returns true, and the transaction is accepted.
   - If invalid, the script returns false, and the transaction is rejected.

# Advanced Bitcoin Scripts

- Multi-signature (`multisig`): Require multiple signatures to spend.
- Timelocks (`CheckLockTimeVerify` and `CheckSequenceVerify`): Restrict spending until a certain time or condition.
- Pay-to-Script-Hash (P2SH): Allows complex scripts to be hidden behind a single address.

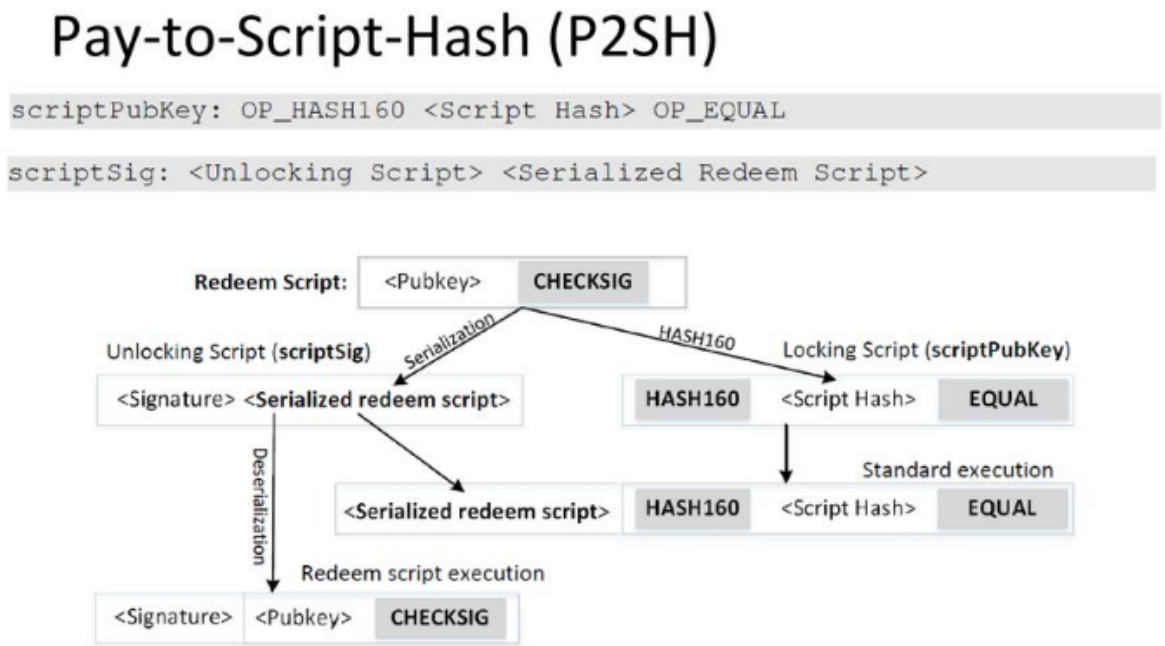


Figure: P2SH (*Magnus Hensley*)

# Advanced Bitcoin Scripts

- Multi-signature (`multisig`): Require multiple signatures to spend.
- Timelocks (`CheckLockTimeVerify` and `CheckSequenceVerify`): Restrict spending until a certain time or condition.
- Pay-to-Script-Hash (P2SH): Allows complex scripts to be hidden behind a single address.

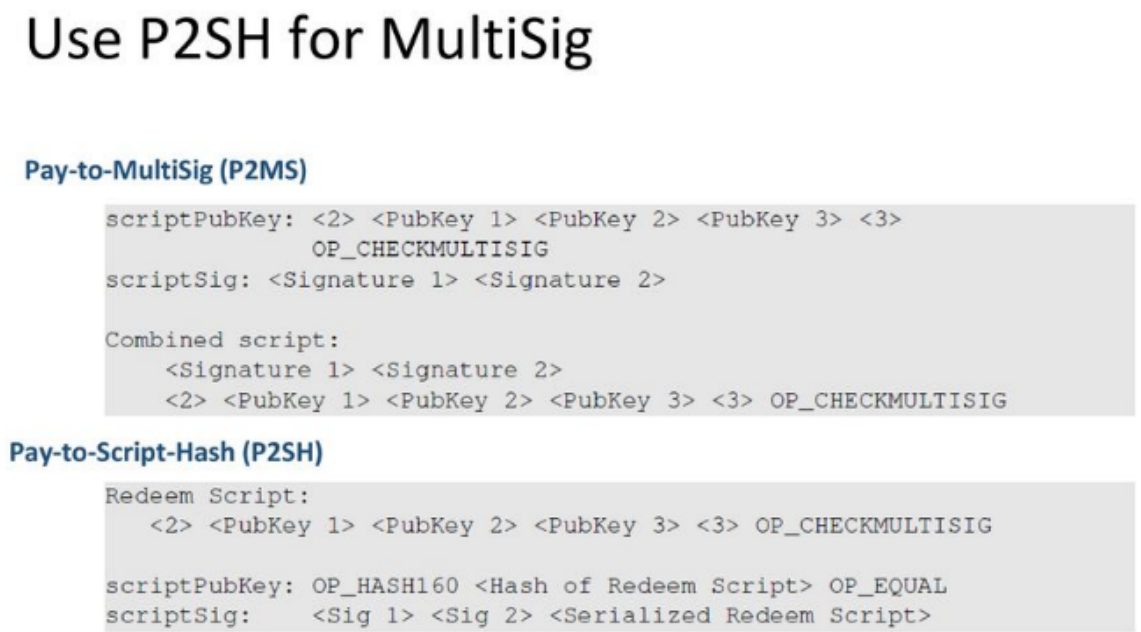


Figure: P2SH for MultiSig (*Magnus Hensley*)

# Comparison with Ethereum's Solidity

- Bitcoin Script is simpler and more secure due to non-Turing completeness.
- Solidity allows for more complex smart contracts but with higher risk of vulnerabilities.

# Submission guidelines

- **Contents:**
  - Source code in `<Code>` folder.
  - Project report (`Report.pdf`).
- **Format:** Group's ID.zip

# Q&A

**Questions?**