Class 15: Script

Schedule

- Project 2 Part 2 is due next **Thursday, 5 March.** Submission is by email, send a PDF with your answers to questions 5-9.
- Keep thinking about ideas for your project. The first deliverable for the project will be a preliminary proposal due on March 19.
- Enjoy your Spring Break!

Bitcoin Script

Transaction outputs in bitcoin are protected by *locking scripts*, and must be unlocked by *unlocking scripts*. The scripts are written in a simple (compared to, say, the Java Virtual Machine language, but quite complex and poorly specified for what one might expect would be needed for bitcoin transactions) stack-based language. A transaction output is not unlocked unless an unlocking script is provided such that the result of executing the unlocking script, followed by executing the locking script, is a stack with value True on top (and no invalid transaction results during the execution).

Some script instructions:

Opcode	Input	Output	Description
`OP_1`	-	**1**	Pushes a **1** (True) on the stack
`OP_DUP`	*a*	*a a*	Duplicates the top element of the stack
`OP_ADD`	*a b*	(*a*+*b*)	Pushes the sum of the top two elements.
`OP_EQUAL`	*a b*	**0** or **1**	Pushes **1** if the top two elements are exactly equal, otherwise **0**.
`OP_VERIFY`	*a*	-	If *a* is not **True** (**1**), terminates as Invalid.
`OP_RETURN`	-	-	Terminates as Invalid.
`OP_EQUALVERIFY`	*a b*	-	If *a* and *b* are not equal, terminates as Invalid.
`OP_HASH160`	*a*	H(*a*)	Pushes bitcoin address, RIPEMD(SHA256(*a*)).

Some more complex instructions:

`OP_IF` [statements] `OP_ENDIF` - If the top of the stack is **1**, executes [statements]. Otherwise does nothing.

`OP_CHECKSIG` - Pops two items from the stack, *publickey* and *sig*. Verifies the entire transaction (known from node state, not the stack) using the *publickey* and *sig*. If the signature is valid, push **1**; otherwise, **0**.

```
OP_1 OP_DUP OP_ADD OP_DUP OP_SUB OP_VERIFY
```

The most common locking script (send to public address):
`OP_DUP`
`OP_HASH160`
`OP_DATA20` *(bitcoin address)*
`OP_EQUALVERIFY`
`OP_CHECKSIG`

What must be on the stack for the locking script to succeed (end with **1** on top of stack)?

According to Most Popular Transaction Scripts (analysis of all transactions in first 290,000 blocks), the ninth most popular script is: `OP_RETURN OP_DATA_40`

What must be on the stack for the `OP_RETURN OP_DATA_40` locking script to succeed (end with **1** on top of stack)? (Trick question: what happens to the coin protected by this locking script?)

Is the bitcoin scripting language Turing-complete?

(See online version for links.)

 Creative Commons Attribution-Noncommercial-Share Alike 3.0