pay-to-sudoku

Sean Bowe
Zcash
Live demo

- Live demos always fail without exception
  - Network will go offline
  - Laptop will start on fire
  - SHA256 collisions destroy Bitcoin network
  - Miners switch to dogecoin
Paying for the solution to a sudoku puzzle

- Alice wants the solution to a puzzle, $P$. 

Paying for the solution to a sudoku puzzle

• Alice wants the solution to a puzzle, $P$.

Redeem script:

OP_???
OP_???
OP_???
OP_???
OP_???
OP_???
Paying for the solution to a sudoku puzzle

- Alice wants the solution to a puzzle, \( P \).

**Problems**

- The script (and the solution) could be gigantic for larger puzzles.
- Bitcoin's scripting system isn't expressive enough.
- Everyone else discovers the solution.
- If somebody tries to spend the script, someone else can spend it using their solution first.

Redeem script:

```plaintext
OP_???
OP_???
OP_???
OP_???
OP_???
```
Paying for the solution to a sudoku puzzle

• Alice wants to pay Bob to solve a puzzle.

```
OP_IF
  bob_pubkey
  OP_CHECKSIGVERIFY
  OP_SUDOKU...
OP_ELSE
  400000
  OP_CHECKLOCKTIMEVERIFY
  alice_pubkey
  OP_CHECKSIGVERIFY
OP_ENDIF
```

Problems

• The script (and the solution) could be gigantic for larger puzzles.
• Bitcoin's scripting system isn't expressive enough.
• Everyone else discovers the solution.
• If somebody tries to spend the script, someone else can spend it using their solution first.
Zero-knowledge contingent payments

• Gregory Maxwell described them in 2011
• Relies on two processes:
  - An interactive zero-knowledge proving scheme
  - An atomic swap over the blockchain
• Achieves
  - Privacy of the solution (and the problem)
  - Small transaction size
HTLC (Hashed Timelock Contract)

Alice

SHA256(K)

Bob

K
HTLC (Hashed Timelock Contract)

Alice

SHA256(K)

OP_SHA256
h_key
OP_EQUAL
OP_IF
   bob_pubkey
   OP_CHECKSIG
OP_ELSE
   future_block_height
   OP_CHECKLOCKTIMEVERIFY
   OP_DROP
   alice_pubkey
   OP_CHECKSIGVERIFY
OP_ENDIF

Bob

K
HTLC (Hashed Timelock Contract)

**Alice**

**SHA256(K)**

- \texttt{OP\_SHA256}
- \texttt{h\_key}
- \texttt{OP\_EQUAL}
- \texttt{OP\_IF}
  - \texttt{bob\_pubkey}
  - \texttt{OP\_CHECKSIG}
- \texttt{OP\_ELSE}
  - \texttt{future\_block\_height}
  - \texttt{OP\_CHECKLOCKTIMEVERIFY}
  - \texttt{OP\_DROP}
  - \texttt{alice\_pubkey}
  - \texttt{OP\_CHECKSIGVERIFY}
- \texttt{OP\_ENDIF}

**Bob**

- \texttt{K}
- \texttt{OP\_ENDIF}

- Bob must disclose K to get the money
- Alice gets her money back if Bob doesn't provide K
- The transaction is not that big.
Alice \hspace{1cm} Bob

$Q \rightarrow$ pk

$pk \rightarrow vk$

$vk \rightarrow A$
Alice

Q
pk
vk

Bob

A
K

Encrypt(A, K)
SHA256(K)

Proof^pk
Zero-knowledge proof

• Given a question $Q$, a hash $H$, and an encrypted answer $E$
• I know answer $A$ and key $K$
• Such that
  – $A$ answers $Q$
  – $E$ is $\text{Encrypt}(A, K)$
  – $H$ is $\text{SHA256}(K)$
Alice uses a HTLC to pay Bob in exchange for $K$. Alice decrypts with $K$ to get the solution.
Pros and cons

• Pro: The transaction is *atomic*, *trustless*, and *private*.

• Pro: The transaction is small and completely prunable.

• Pro: We can do it on Bitcoin today!
Pros and cons

- Con: The transaction is interactive.
- Con: Constructing the zero-knowledge proof can take seconds to minutes depending on the complexity of the circuit.
- Con: The proving key can be tens to hundreds of megabytes in size depending on the complexity of the circuit.
Circuit Statistics

• 16x16 sudoku:
  - Proving key: 68MB
    • Only needs to be computed once, so cost can be amortized.
  - Proving time: 10 to 20 seconds
  - Proof: 288 bytes (sent off chain)
  - Verification time: 40ms
  - Circuit cost:
    • Encrypt(A, K) (81.86%)
      ▪ ChaCha20 would be a 3x improvement over the current cipher.
    • SHA256(K) (10.23%)
      ▪ Could use RIPEMD-160?
    • Solution validity (4.42%)
      ▪ Mostly unoptimizable
Wrapping up

- Code: https://github.com/zcash/pay-to-sudoku

- Thanks:
  - Gregory Maxwell
  - Pieter Wuille
  - Madars Virza
  - Andrew Poelstra
  - Zcash Company