Carleton UNIVERSITY

CrySP

WATERLOO
CHERITON SCHOOL OF
COMPUTER SCIENCE

CommitCoin:

Carbon Dating Commitments with Bitcoin

By Jeremy Clark & Aleksander Essex

Overview

- We propose a method for creating commitments that can later be carbon dated to the approximate time of creation
- A general method uses moderately hard functions but has limitations that make it impractical for deployment
- CommitCoin resolves these drawbacks by using the Bitcoin block-chain

Proof of Work / Puzzles

- Cryptographic Puzzles:
  - Generate puzzle p with difficulty d from randomness r

    p=Gen(d,r)
  - Compute solution s to puzzle p

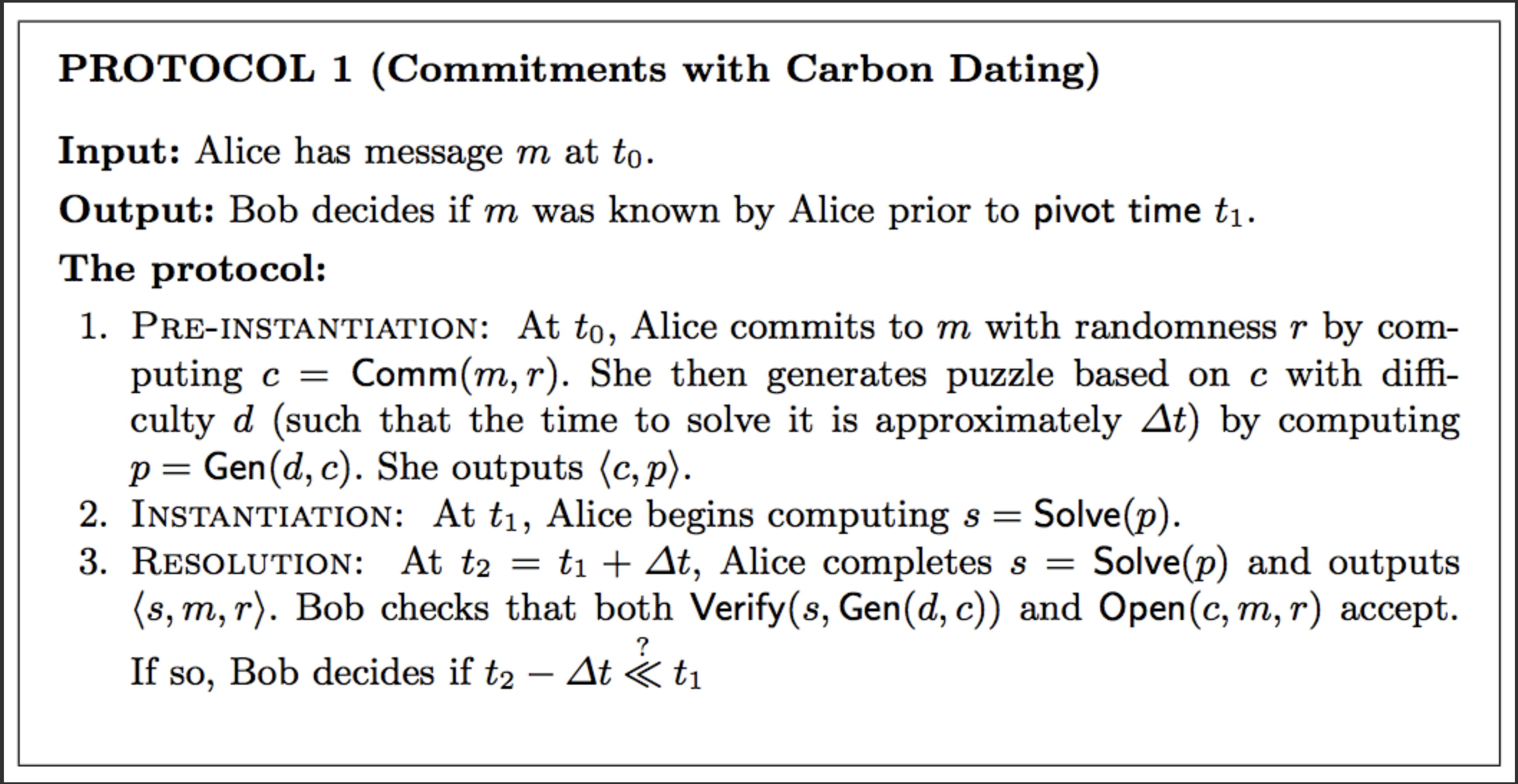
    s=Solve(p)
  - Verify solution s to puzzle p

    Verify(p,s)
- Gen and Verify are efficient; Solve is moderately hard

Related Work on Puzzles

- Moderately hard function:
  - processing time
  - memory access time
  - storage
- Applications:
  - time-release encryption & commitments
  - metering access to prevent email spam or DOS
  - minting coins in digital cash

Carbon Dating



PROTOCOL 1 (Commitments with Carbon Dating)

Input: Alice has message m at t_0.

Output: Bob decides if m was known by Alice prior to **pivot time** t_1.

The protocol:

1. Pre-instantiation: At t_0, Alice commits to m with randomness r by computing $c = \mathsf{Comm}(m, r)$. She then generates puzzle based on c with difficulty d (such that the time to solve it is approximately Δt) by computing $p = \mathsf{Gen}(d, c)$. She outputs $\langle c, p \rangle$.
2. Instantiation: At t_1, Alice begins computing $s = \mathsf{Solve}(p)$.
3. Resolution: At $t_2 = t_1 + \Delta t$, Alice completes $s = \mathsf{Solve}(p)$ and outputs $\langle s, m, r \rangle$. Bob checks that both $\mathsf{Verify}(s, \mathsf{Gen}(d, c))$ and $\mathsf{Open}(c, m, r)$ accept. If so, Bob decides if $t_2 - \Delta t \overset{?}{\ll} t_1$

Ideal Puzzle

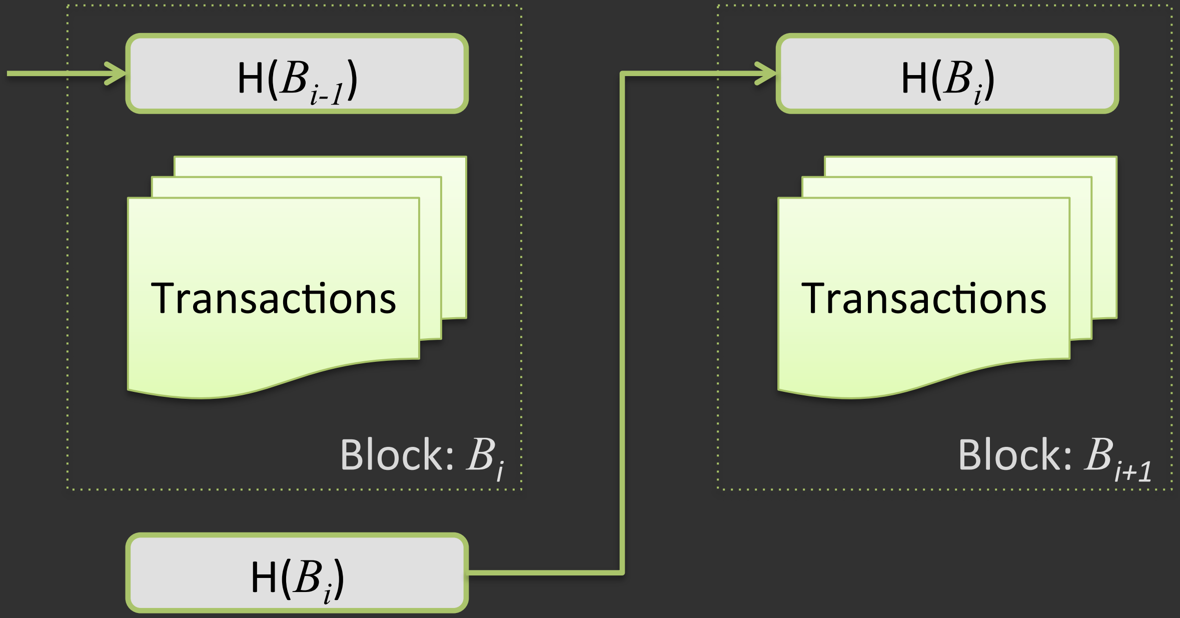
- Two main puzzles: repeated squaring and hash-based
- Repeated squaring:
  - Inherently sequential
  - Verifiable by only creator (and easy to solve by creator)
- Hash-based
  - Creator can also solve it while anyone can verify (non-interactive)
  - Trivially parallelizable

Carbon Dating

- Drawback 1: no ideal proof of work protocol
- Drawback 2: must devote CPU
- Drawback 3: consider predicating an election outcome, nothing stops you from carbon dating commitments to each possible outcome
- Drawback 4: carbon dating is very fuzzy: too fuzzy to be useful?

Bitcoin

- Bitcoin is a digital currency
- A public transcript of every transaction is maintained by a group of nodes
- Sufficient to only understand this transcript ("block chain") to understand CommitCoin



H(B_{i-1})

Transactions

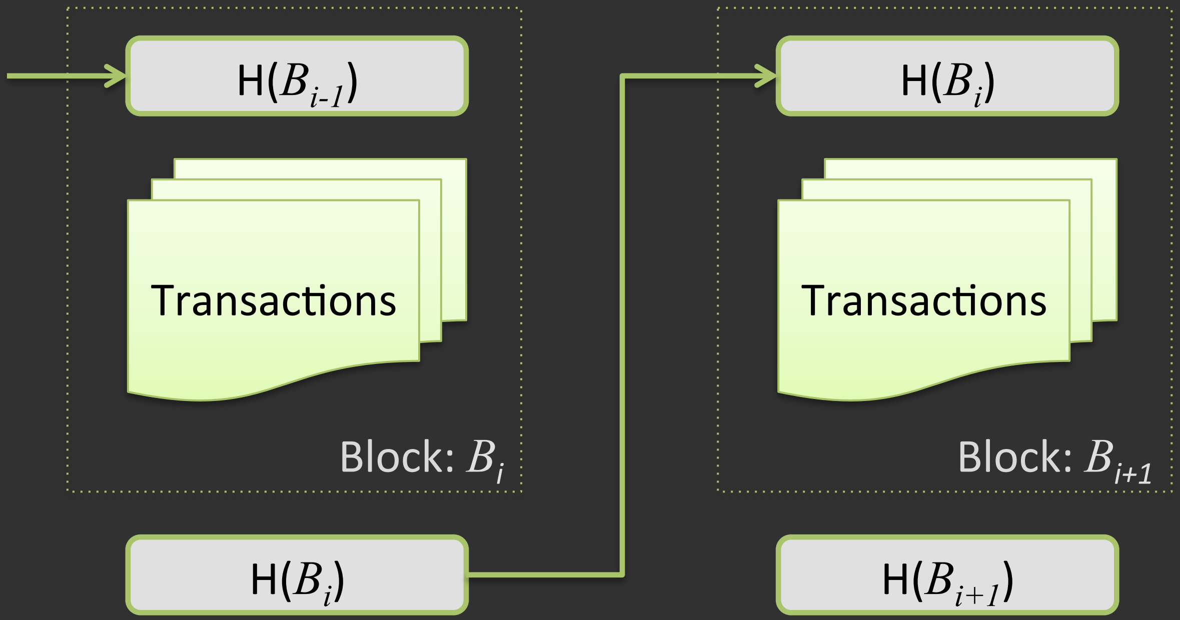
Block: B_i

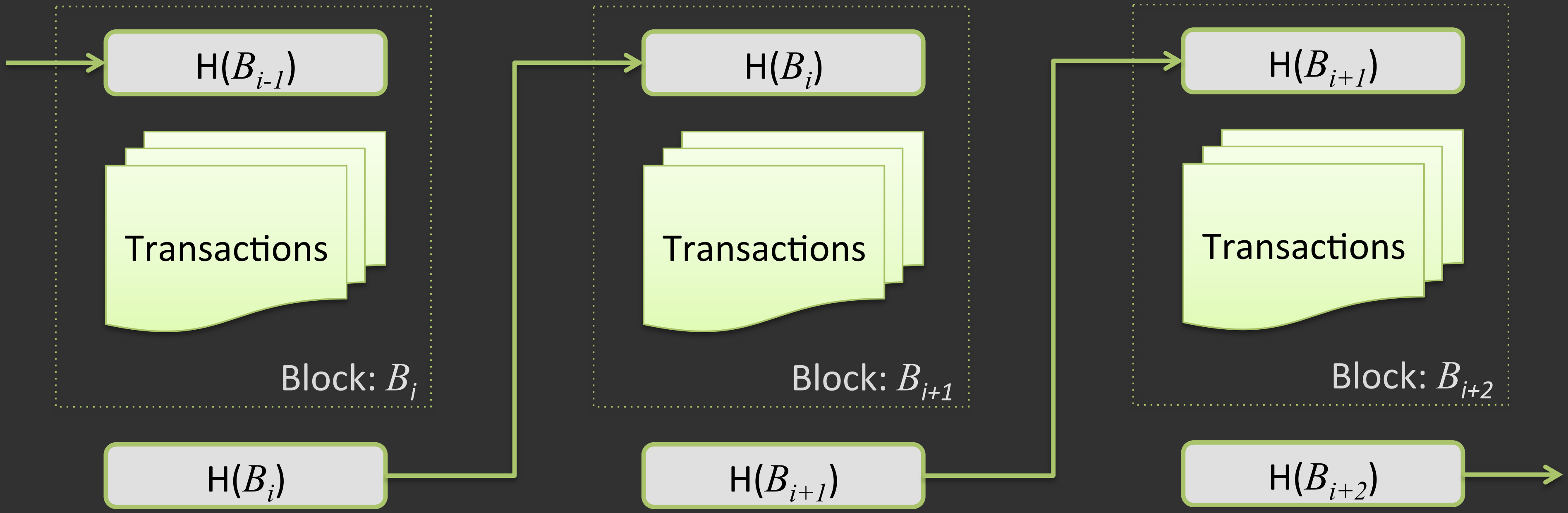
H(B_i)

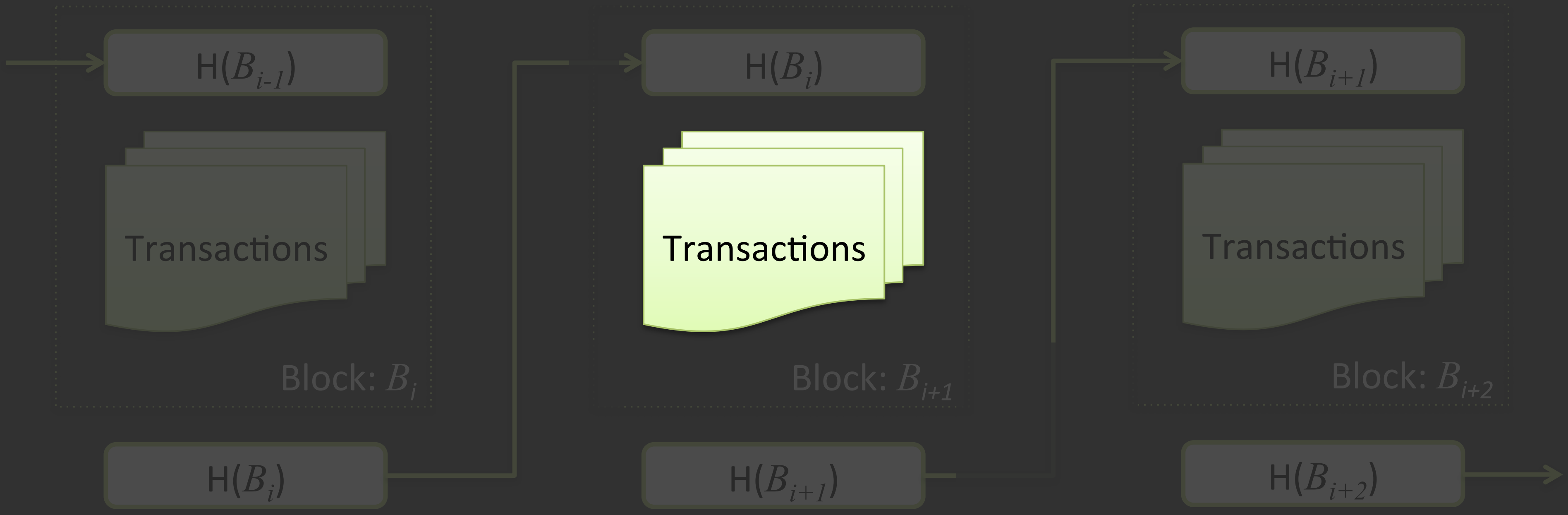
Transactions

Block: B_{i+1}

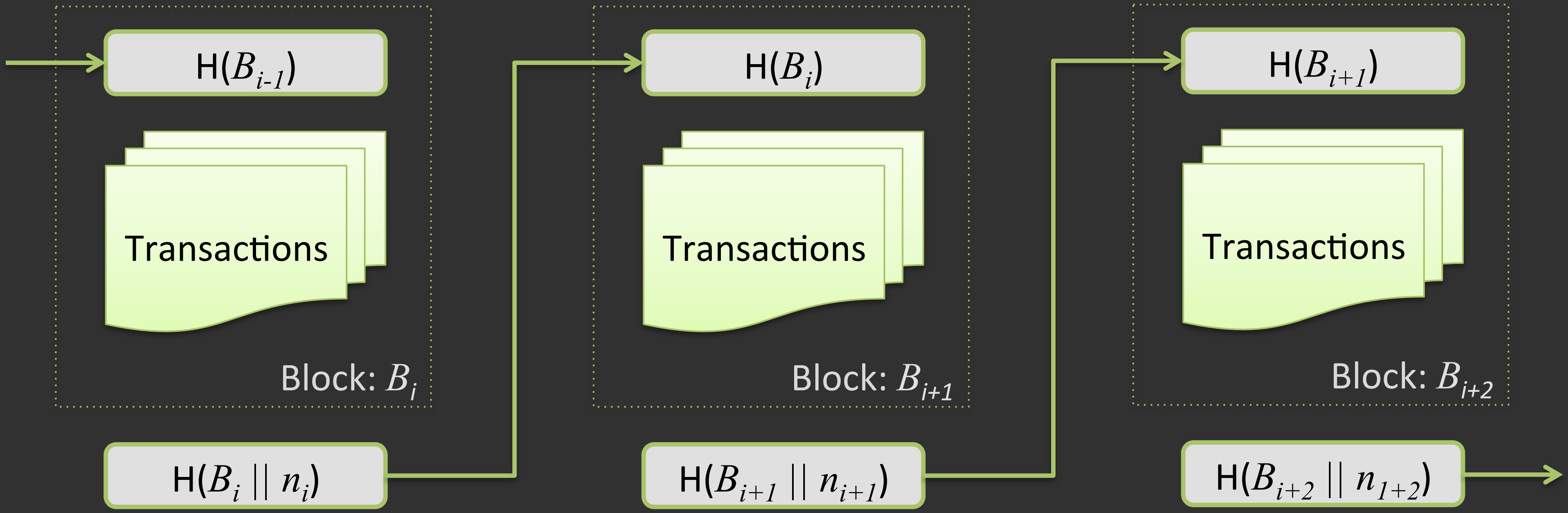
H(B_i)


H(B_{i-1})
Transactions
Block: B_i
H(B_i)
H(B_i)
Transactions
Block: B_{i+1}
H(B_{i+1})


H(B_{i-1})
Transactions
Block: B_i
H(B_i)
H(B_i)
Transactions
Block: B_{i+1}
H(B_{i+1})
H(B_{i+1})
Transactions
Block: B_{i+2}
H(B_{i+2})



Amount: 100 BTC
To: [PubKey Fingerprint]$_B$
From: [PubKey]$_A$
Signed: By A



Each hash is a proof of work.Find an n_i such that:

$$H(B_i \,||\, n_i) = \{0\}^{d} \,||\, \{0,1\}^{n-d}$$

Takes 2^{d-1} hash evaluations on average

Can be parallelized (without storage: suitable for GPU)

CommitCoin

- Idea: insert commitment into the block chain, and the chain of proof of works will provide carbon dating
- Resolves the need to devote a CPU
- While parallelizable, variance in computational power across network is smaller than a singe individual
- Largest pool reports 2^{42} hashes/s

CommitCoin

- Question: how to insert?
- Solution 1:
  - Find a unchecked field in the transaction spec
  - Drawback: could be patched
- Solution 2:
  - Set commitment value to public key fingerprint
  - Drawback: “burns” money

CommitCoin

- Set commitment value to ECDSA private key
- Commitment is randomized; functions as key
- Send 2 units of BTC to corresponding public key (fingerprint added to transcript)
- Send 1 unit back to originating account (public key added to transcript)
- Send 1 unit back using same randomness (private key/commitment computable from transcript)

Application

- Scantegrity is a verifiable voting system
- It uses pre-election commitments that are used after the election to prove the tally is correct
- Simple attack: change pre-election commitments after the election
- Detectable: by verifiers who obtain commitments before the election (but is this really *universally verifiable*?)
- In 2011 Takoma Park election, we used CommitCoin
- Known pivot and negligible probability that an unsound pre-election commitment will verify

Drawbacks Revisted

- Drawback 1: no ideal proof of work protocol
  - Sidestep parallelization issue
- Drawback 2: must devote CPU
  - Use Bitcoin
- Drawback 3: can carbon date commitments to linearly many messages
  - Scantegrity pre-election commitments is large space
- Drawback 4: carbon dating is very fuzzy: too fuzzy to be useful?
  - Can pre-commitment months before election day