Carleton UNIVERSITY

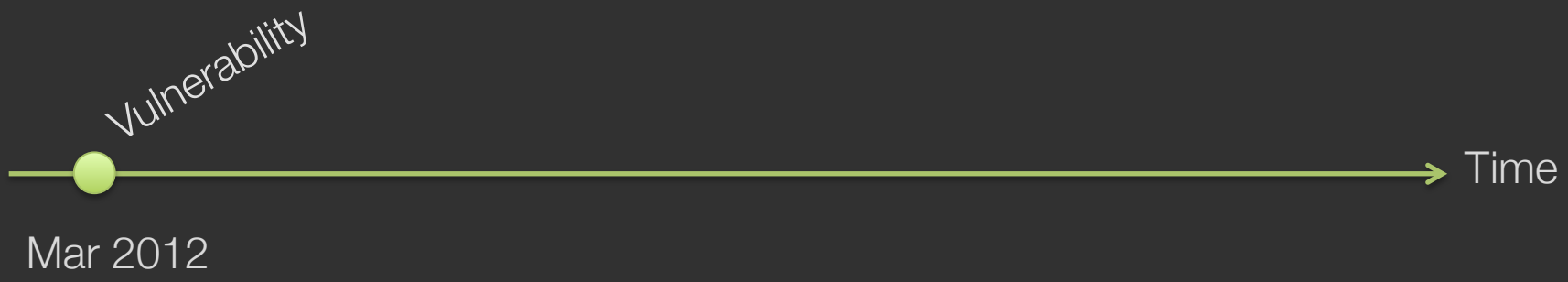
CrySP

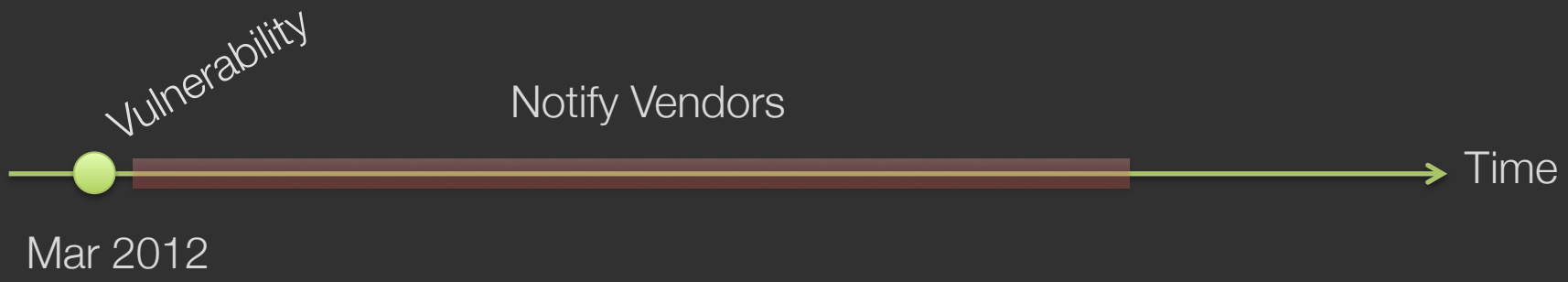
WATERLOO
CHERITON SCHOOL OF
COMPUTER SCIENCE

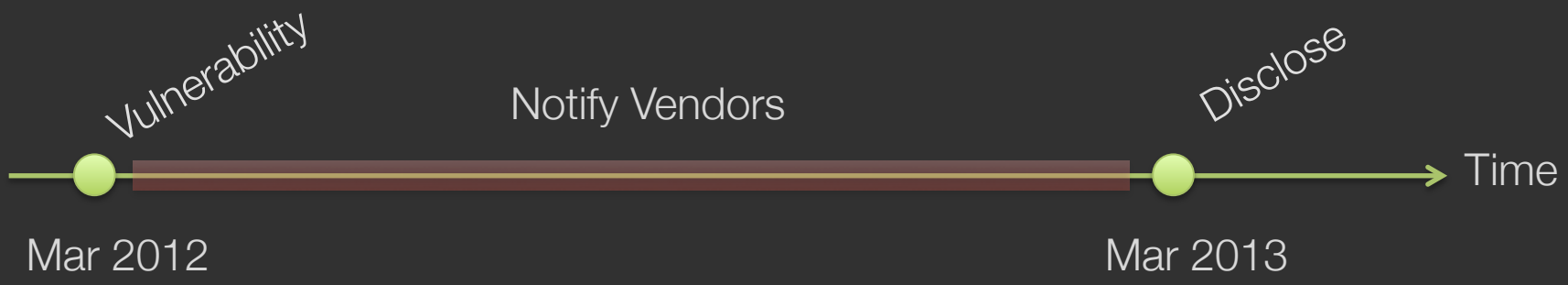
NSERC
CRSNG

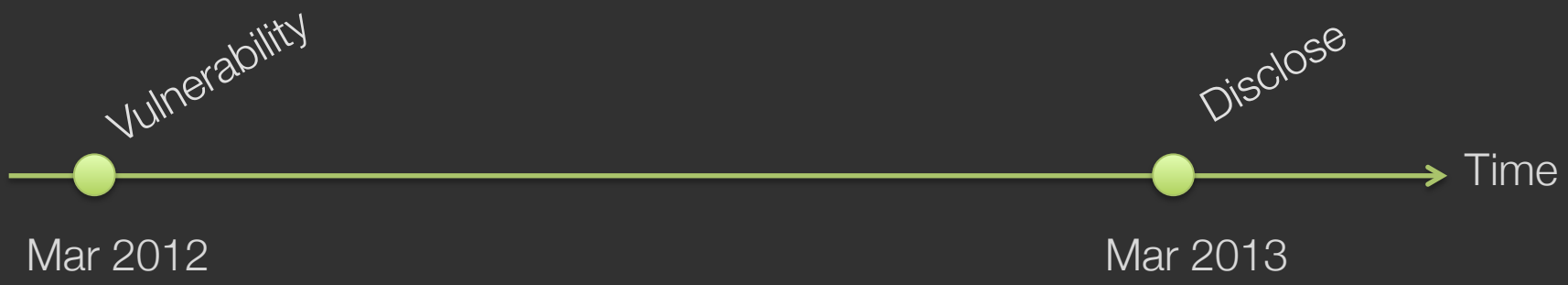
How to “carbon date” digital information

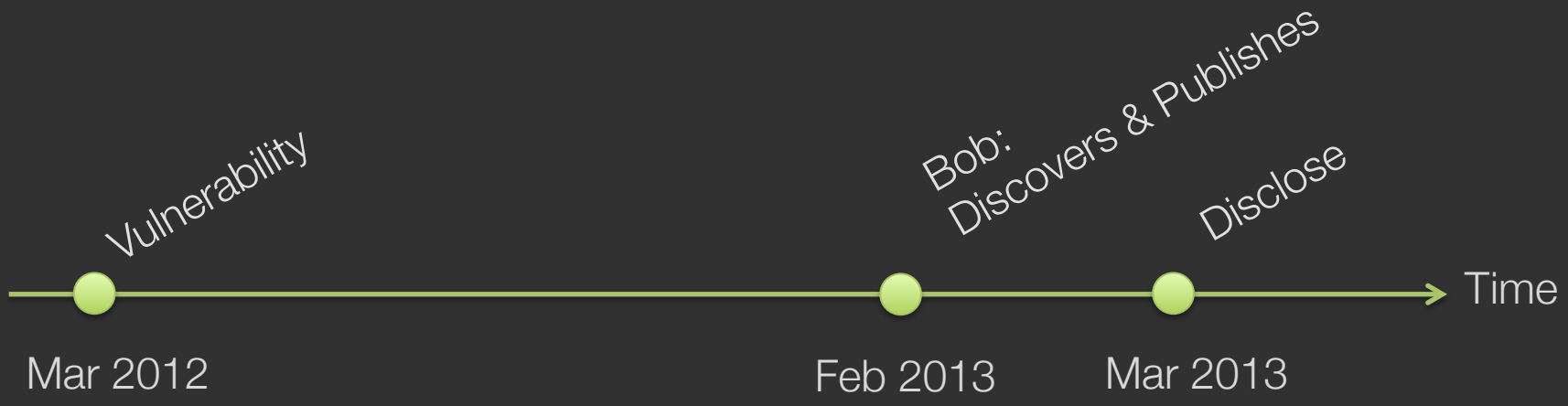
Jeremy Clark

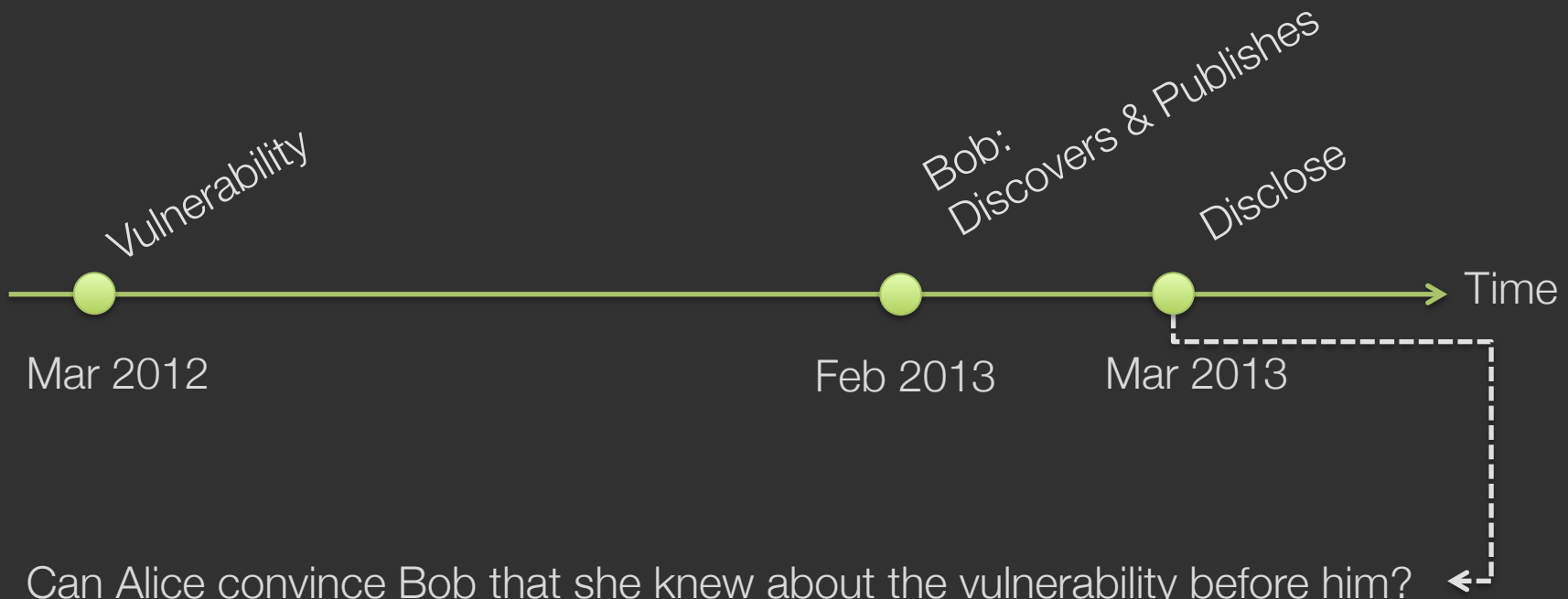

Vulnerability
Time
Mar 2012


Vulnerability
Notify Vendors
Time
Mar 2012

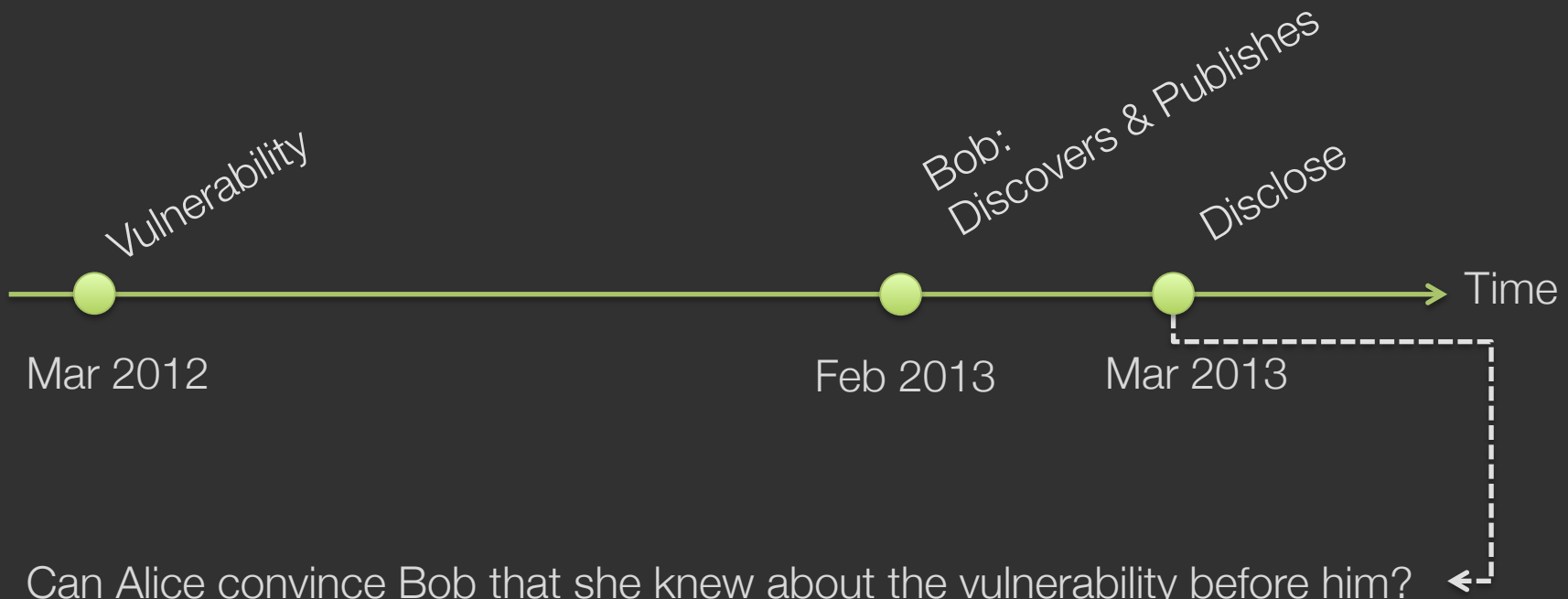

Vulnerability
Notify Vendors
Disclose
Time
Mar 2012
Mar 2013


Vulnerability
Disclose
Time
Mar 2012
Mar 2013


Vulnerability
Bob:
Discovers & Publishes
Disclose
Time
Mar 2012
Feb 2013
Mar 2013

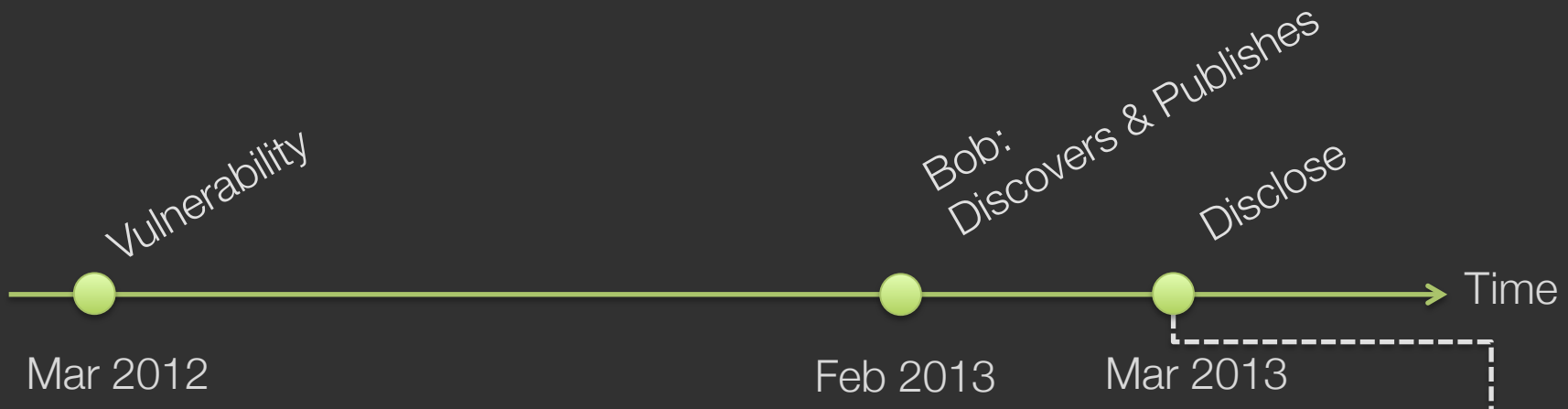


Can Alice convince Bob that she knew about the vulnerability before him?



Can Alice convince Bob that she knew about the vulnerability before him?

- Broadcast a commitment: Bob must be listening
- Time-stamping service: Bob must trust service



Can Alice convince Bob that she knew about the vulnerability before him?

- Broadcast a commitment: Bob must be listening
- Time-stamping service: Bob must trust service
- Carbon-dating: No TTPs and no prior interaction

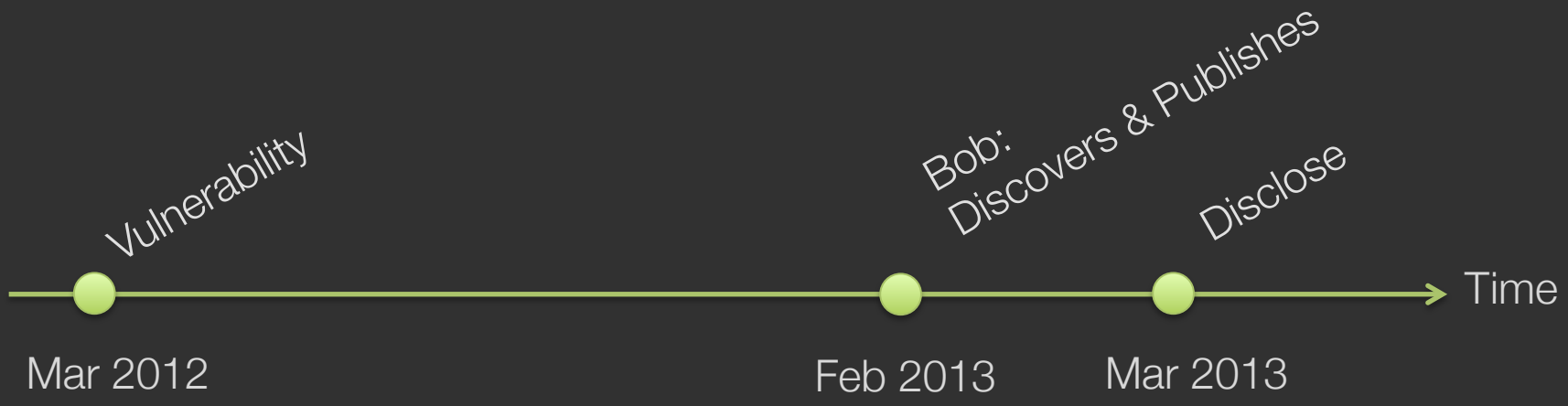
Carbon Dating with Puzzles

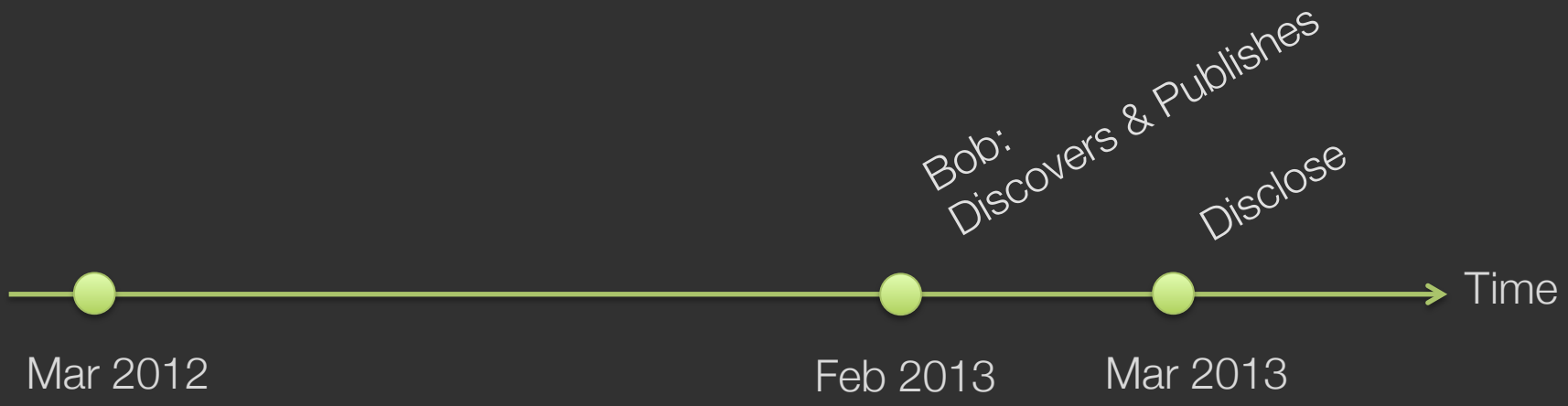
A Cryptographic Puzzle

- I generate a random number r
- I ask you to find any number n such that the output of Hash(r||n) has d leading zeros
  - Hash(r||00000000) = 00100101…
  - Hash(r||00000001) = 01110100…
  - Hash(r||10001011) = 00000000…
- How much work is this? 2^{d-1} hash evaluations on average

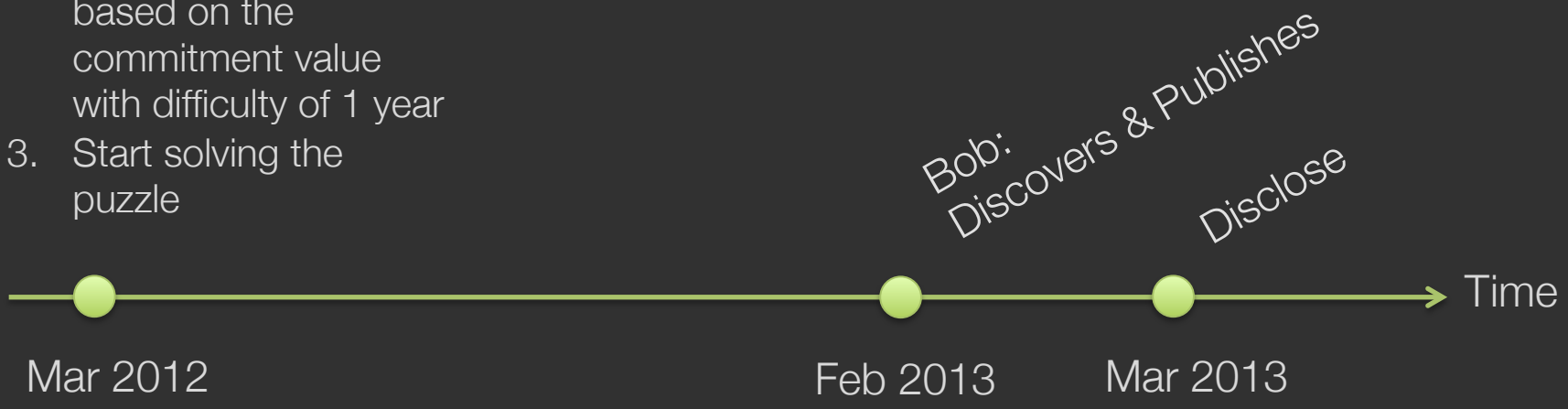
Moderately Hard Functions

- Lots of names: puzzles, proof of work, delaying functions, …
- Difficulty based on:
  - processing time
  - memory access time
  - storage
- Applications:
  - time-release encryption & commitments
  - metering access to prevent email spam or DOS
  - minting coins in digital cash


Vulnerability
Bob:
Discovers & Publishes
Disclose
Time
Mar 2012
Feb 2013
Mar 2013


Bob:
Discovers & Publishes
Disclose
Time
Mar 2012
Feb 2013
Mar 2013

1. Commit to vulnerability
2. Generate a puzzle based on the commitment value with difficulty of 1 year
3. Start solving the puzzle



1. Commit to vulnerability
2. Generate a puzzle based on the commitment value with difficulty of 1 year
3. Start solving the puzzle

1. Commit to vulnerability
2. Generate a puzzle based on the commitment value with difficulty of 1 year
3. Start solving the puzzle

4. Produce solution to puzzle and give to Bob
5. Bob can verify solution is correct, based on commitment, and commitment opens to the vulnerability

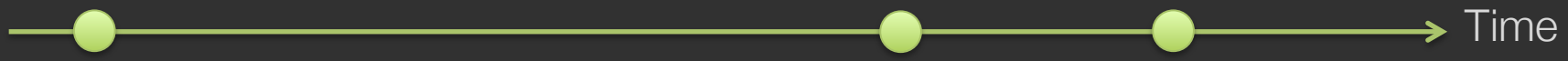
1. Commit to vulnerability
2. Generate a puzzle based on the commitment value with difficulty of 1 year
3. Start solving the puzzle

4. Produce solution to puzzle and give to Bob
5. Bob can verify solution is correct, based on commitment, and commitment opens to the vulnerability

6. Bob concludes that to solve a problem of this difficulty, Alice must have started solving it before Feb 2012

1. Commit to vulnerability
2. Generate a puzzle based on the commitment value with difficulty of 1 year
3. Start solving the puzzle

4. Produce solution to puzzle and give to Bob
5. Bob can verify solution is correct, based on commitment, and commitment opens to the vulnerability



Mar 2012

Feb 2013

Mar 2013

6. Bob concludes that to solve a problem of this difficulty, Alice must have started solving it before Feb 2012

Vulnerability <--- Commitment <--- Puzzle <--- Solution

1 year

You may be wondering…

- In the paper we give further considerations:
  - What about parallel computing? (inherently sequential puzzles)
  - Does the puzzle creator know the solution? (non-interactive puzzles)
  - Does producing one solution help find other solutions? (amortized cost)
  - Is a puzzle binding to a commitment value?

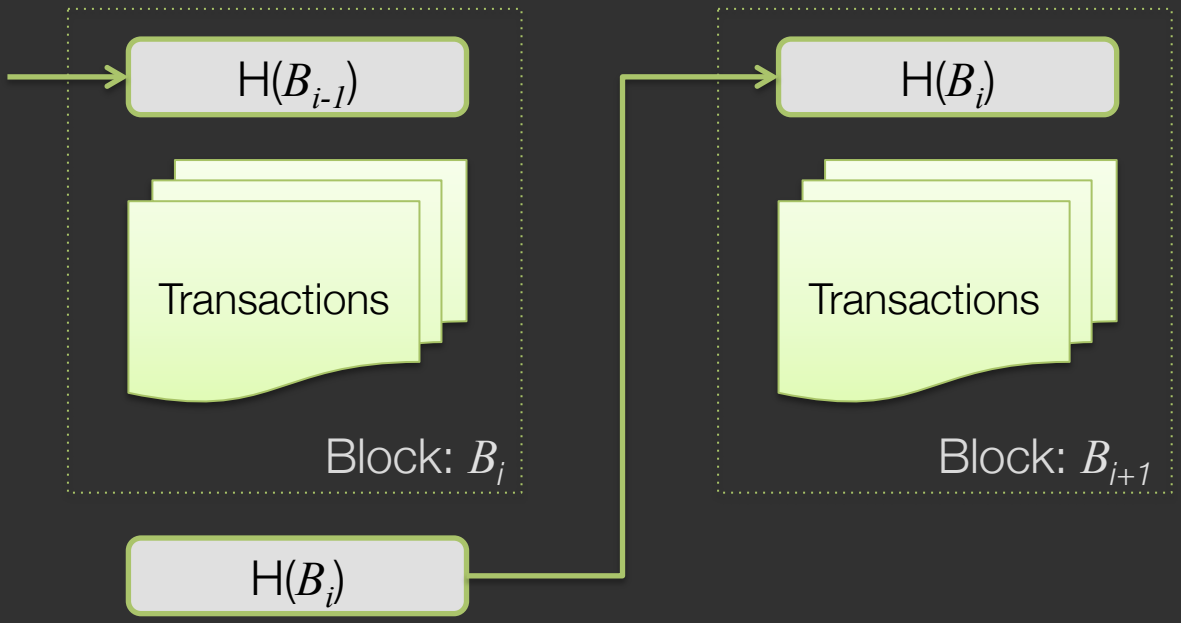
Carbon Dating

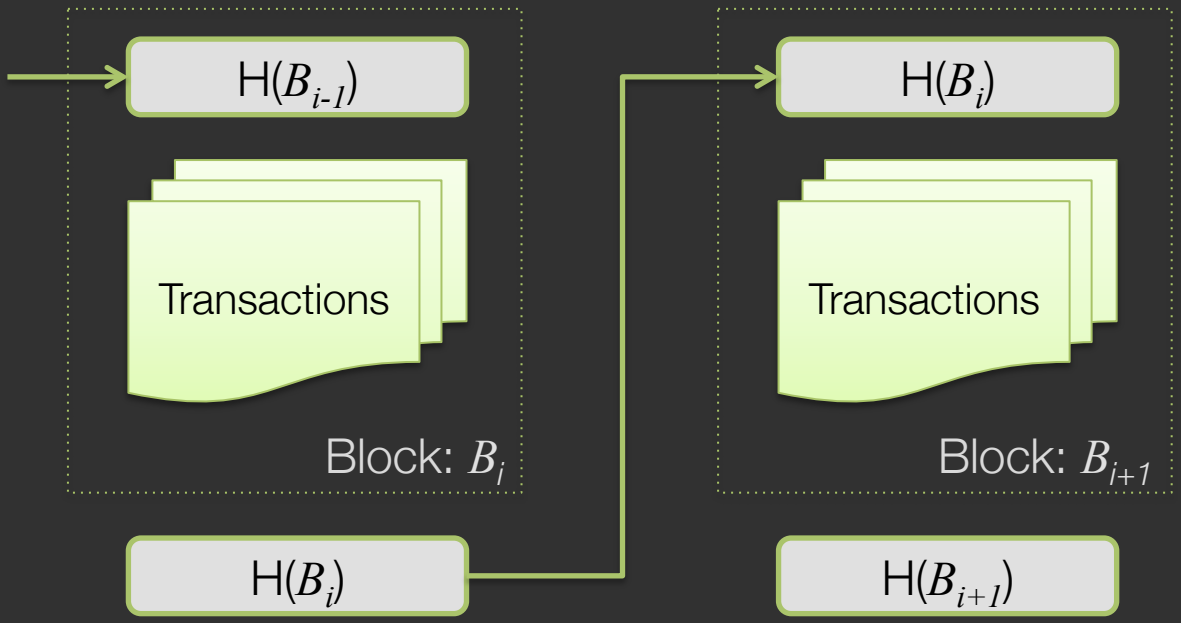
- Drawback 1: no inherently sequential puzzle
- Drawback 2: must devote CPU
- Drawback 3: consider predicating an election outcome, nothing stops you from carbon dating commitments to each possible outcome
- Drawback 4: carbon dating is very fuzzy: too fuzzy to be useful?

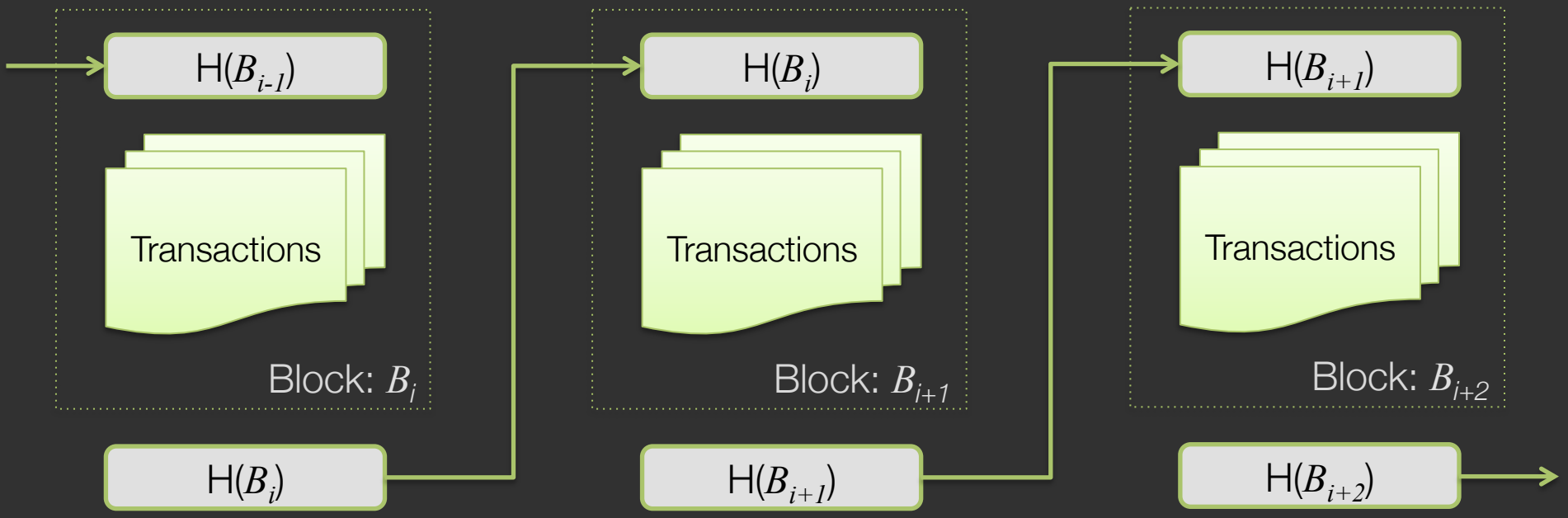
A Diversion: Bitcoin

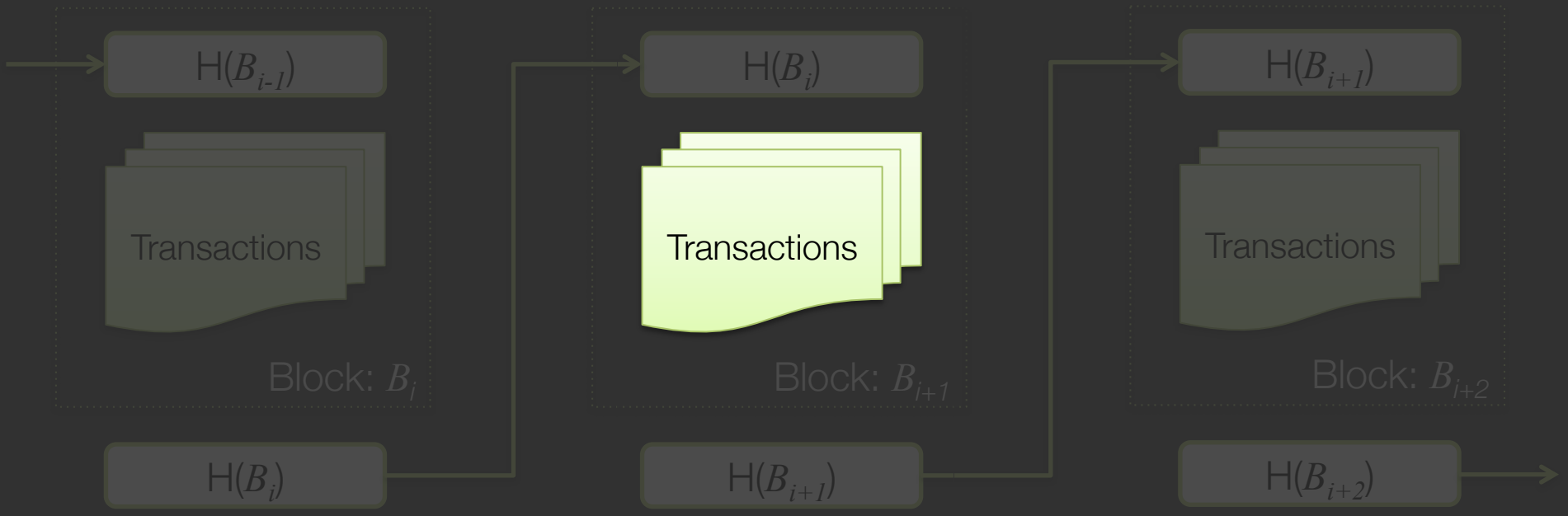
Bitcoin

- 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 how to carbon date with Bitcoin

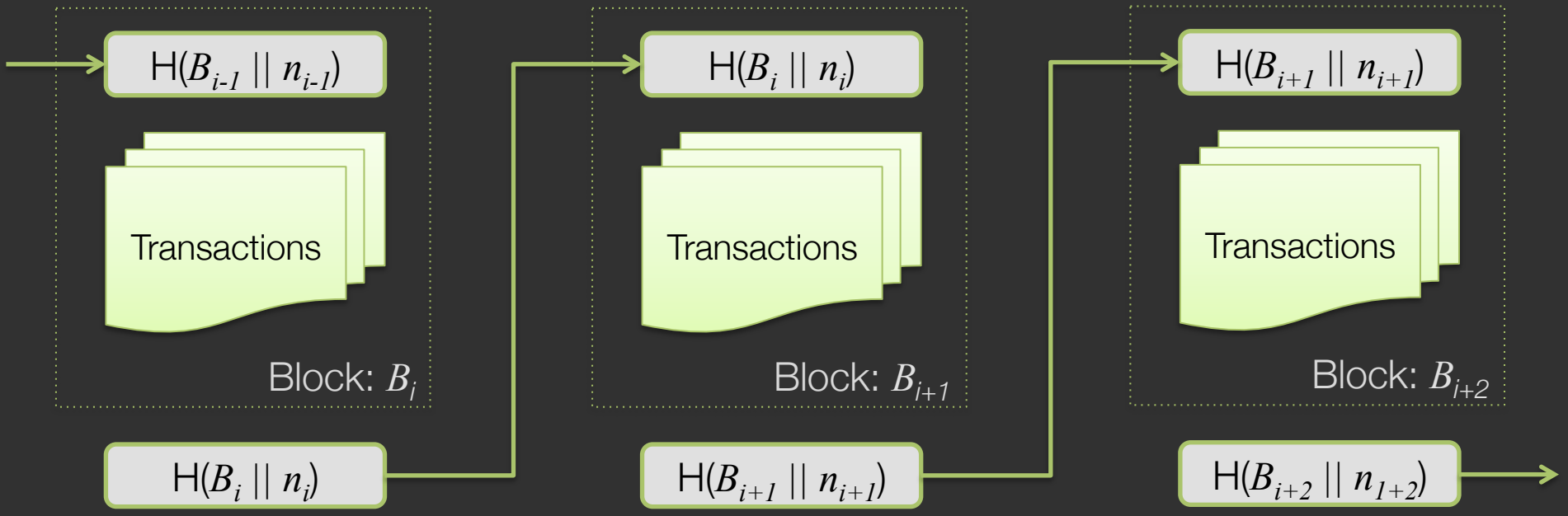

H(B_{i-1})
Transactions
Block: B_i
H(B_i)
H(B_i)
Transactions
Block: B_{i+1}


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
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: $[PubKey]_B$
From: $[PubKey]_A$
Signed: By A



- Each hash is a proof of work
- Takes 2^{d-1} hash evaluations on average (d=53 currently)
- Can be parallelized (without storage: suitable for GPU)
- First node to find solution is awarded newly minted coins

CommitCoin: Carbon Dating with Bitcoin

CommitCoin

- Computational power across network is large: solves puzzle in ~10 min, one pool reports 2^{42} hashes/s
- Idea: insert commitment into the block chain, and the chain of proof of works will provide carbon dating

Drawbacks Revisted

- Drawback 1: no inherently sequential puzzle
  - Sidestep parallelization issue
- Drawback 2: must devote CPU
  - Use Bitcoin network
- Drawback 3: can carbon date commitments to linearly many messages
- Drawback 4: carbon dating is very fuzzy: too fuzzy to be useful?

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

1. Set *randomized* commitment value to ECDSA private key
2. Compute corresponding public key
3. Send 2 units of BTC to public key
4. Send 1 unit back to originating account, signing with private key
5. Again send 1 unit back, singing with private key and the same randomness
6. Leaks private key: commitment computable from transcript

Applying Carbon Dating

Application of Carbon Dating

Scantegrity

- Scantegrity is a verifiable voting system
- It uses pre-election commitments to what should be printed on each ballot
- During the election, voters can request a ballot to audit
- Simple attack: change pre-election commitments after you know which ballots were audited
- Detectable: by verifiers who obtain commitments before the election (but is this really *universally verifiable*?)
- In 2011 Takoma Park election, we used CommitCoin so commitments can be carbon dated to before the election

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?


Vulnerability
(Commitments)
Bob:
Discovers & Publishes
(Election Day)
Disclose
(Verify Election)
Time
Mar 2012
Feb 2013
Mar 2013

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

That’s It. Questions?

See the paper for more…

Carbon dating:

Clark & Essex. “CommitCoin: Carbon Dating Commitments with Bitcoin.” *Financial Cryptography* 2012.

Random beacons:

Clark & Hengartner. “On the Use of Financial Data as a Random Beacon.” *USENIX EVT/WOTE* 2010.

Scantegrity:

Carback, Chaum, Clark, et al. “Scantegrity II Municipal Election at Takoma Park.” *USENIX Security* 2010.

Chaum, Carback, Clark, et al. “Scantegrity II: End-to-End Verifiability for Optical Scan Election Systems using Invisible Ink Confirmation Codes.” *USENIX EVT* 2008.

Short-lived signatures:

Under preparation