How to “carbon date” digital information

Jeremy Clark
Vulnerability

Mar 2012
Vulnerability

Mar 2012

Notify Vendors

Time
Vulnerability

Notify Vendors

Disclose

Mar 2012

Mar 2013

Time
Vulnerability
Mar 2012

Bob: Discovers & Publishes
Feb 2013

Disclose
Mar 2013

Time
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 $\text{Hash}(r||n)$ has d leading zeros
  – $\text{Hash}(r||00000000) = 00100101…$
  – $\text{Hash}(r||00000001) = 01110100…$
  – $\text{Hash}(r||10001011) = 00000000…$
• 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
1. Commit to vulnerability
2. Generate a puzzle based on the commitment value with difficulty of 1 year
3. Start solving the puzzle

Bob: Discovers & Publishes

Mar 2012
Feb 2013
Mar 2013

Time
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
6. Bob concludes that to solve a problem of this difficulty, Alice must have started solving it before Feb 2012
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
$H(B_{i-1})$  \rightarrow  Transactions \rightarrow  Block: B_i \rightarrow  H(B_i) \\

$H(B_i)$  \rightarrow  Transactions \rightarrow  Block: B_{i+1}
Transactions $H(B_i)$

Block: $B_i$

Transactions $H(B_{i-1})$

Block: $B_{i-1}$

Transactions $H(B_i)$

Block: $B_{i+1}$

Transactions $H(B_{i+1})$
Transactions $H(B_{i-1})$

Block: $B_i$

Transactions $H(B_i)$

Block: $B_{i+1}$

Transactions $H(B_{i+1})$

Block: $B_{i+2}$

Transactions $H(B_{i+2})$
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)

Mar 2012

Bob: Discovers & Publishes (Election Day)

Feb 2013

Disclose (Verify Election)

Mar 2013

Time
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:

Random beacons:

Scantegrity:

Short-lived signatures:
Under preparation