Authenticated Data Structures for Privacy-Preserving Monero Light Clients

Kevin Lee, Andrew Miller
University of Illinois at Urbana-Champaign
Monero hides your activity with mixins

- Sender and receiver identities are kept confidential from peers
- Transactions use mixins to mask the actual spent coin
- RingCT hides denominations in transactions
Lightweight configuration - “remote node”

- Clients can connect to peers running a full node in order to send and receive transactions
- The lightweight client only stores minimal data about the blockchain
- Client queries remote node for outputs, selects the mixins, then transmits its transaction

1. The client sends global output requests for the chaff and the real coin.
2. The server sends back a response.
3. The client verifies the response by only checking for its own coin.
4. If the check passes, the client proceeds to construct the transaction.

Server
Runs a full Monero node

Client
Has private keys and its own coins (coins it knows about)

JSON-RPC over HTTP
But responses are not authenticated!

- What if the remote node returns an invalid response with modified outputs?
  - If the modified outputs correspond to those belonging to a transaction the client holds, then an exception will be triggered by the client.
  - However, lack of preemptive protocol can lead to opportunities for attack.
  - *Retry-and-Intersect*
    - Malicious node modifies all of the output keys requested.
    - Mitigations: caching mixin choices, using TLS for communication.

[Diagram showing outputs requested 1st time, outputs requested 2nd time, and real output.]
Another attack: **Guess-and-Check**

- Even if the previously mentioned mitigations are put into effect, a remote node can still amount an attack which reveals partial information about the client’s funds.
- 2 scenarios:
  - Case 1: corrupt outkey is not the client’s real one
  - Case 2: corrupt outkey is the client’s real one
- As part of The Monero Project Vulnerability Response Process, we have disclosed these vulnerabilities to core developers.
Goal: lightweight client peace-of-mind

- The client should be able to construct a transaction without revealing which is the real coin it’s spending
- Assumption: at least one server is honest
- Needs to support:
  - Output retrieval
  - Proof generation
  - Updates to the structure
  - Conflict resolution
- And it has to be fast(er)!
Build an ADS over the Monero blockchain

The top root contains the greatest index of all the outputs, and is stored at the client.

Sibling nodes are concatenated and hashed to form the parent node. The maximum index between the siblings is passed onto the parent.
A “Refereed Delegation” approach

- The client needs to quickly figure out which server is lying
- The earliest point of disagreement can be found in $O(\log N)$
- Once the source of conflict has been found
  - It is now easy to find the lying server
- Client stores limited information
  - PoW headers
  - Root hash
  - Transactions it owns
- Avoids the need to hardfork
The Refereed Delegation model in practice

- **VERSUM**
  - Supports quick incremental computation and conflict resolution
  - Uses SEQHASH

- **TrueBit**
  - Makes use of *smart contracts* in Ethereum to resolve conflicts
  - Uses judges to justify challenges

- **Our implementation**
  - Emphasis on correctness of transaction outputs
    - There are consequences of including incorrect outputs
Benchmarks w/ initial performance

- Implementation done in Python
- A snapshot of the blockchain at October 2017
- Comparison with current protocol in query response
- Proposed change takes 2.17 GB

<table>
<thead>
<tr>
<th></th>
<th>Average Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build</td>
<td>1785.935888 seconds</td>
</tr>
<tr>
<td>Query response</td>
<td>0.007219 seconds</td>
</tr>
<tr>
<td>Proof verification</td>
<td>0.000132 seconds</td>
</tr>
<tr>
<td>Update top tree</td>
<td>0.003195 seconds</td>
</tr>
<tr>
<td>Conflict resolution</td>
<td>0.171619 seconds</td>
</tr>
</tbody>
</table>
More work can be done!

- Design choices
  - Serialization library
  - Choice of ADS
- Investigate the trend of remote node usage
- Other privacy-preserving cryptocurrencies
Questions?
Thank you!

Email: klee160@illinois.edu
Lab: http://decentralize.ece.illinois.edu/
Backup Slides

Email: klee160@illinois.edu
Lab: http://decentralize.ece.illinois.edu/
Cryptocurrencies are assets on distributed ledgers

- A decentralized peer-to-peer network that uses a blockchain to keep track of user account balances
  - Users broadcast spends to the network

⚠️ Blockchain ≠ Bitcoin!
But, responses are not authenticated!

- Full nodes are expensive to run
  - Fully secure, but at the cost of...
    - Storage
    - Computation power
- Current thin clients for Bitcoin
  - Electrum
- Dishonest servers?
  - How can we tell if a server is lying?
- Authenticated data structures (ADS)

A Merkle tree, along with proof
Build an ADS over the Monero blockchain

• Goal: build an authenticated data structure over the Monero blockchain
  – Using Merkle trees, the client only needs to store the root
• Needs to support:
  – Output retrieval
  – Proof generation
  – Updates to the structure
  – Conflict resolution
• And it has to be fast(er)!

Adding a new node to the tree
No one who speaks Python can be an evil man?

- Implementation done in Python
- C++ scraper to collect information from the Monero blockchain:
  - (block hash, transaction hash, output public key, global index)
- Deployed on LinuxONE medium virtual servers running SLES12 SP2
  - 2 servers as full node servers, 1 as client