Lightning Network
Formal verification of a payment protocol

Léo Louistisserand
Joint work with Simon Jeanteur and Matteo Maffei

(Now PhD student at LORIA in e-voting)

03/04/2024
Bitcoin's lack of scalability → off-chain protocols

Protocol not proven → attacks

Our goal: prove the security of a fix
Core idea

- Lock coins on the chain
- Exchange this money off-chain
- Use the chain to cash in
Opening a channel

Published on-chain

Saved off-chain

Funding transaction

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice: 5</td>
<td>A&amp;B: 10</td>
</tr>
<tr>
<td>Bob: 5</td>
<td></td>
</tr>
</tbody>
</table>

Closing transaction

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A&amp;B: 10</td>
<td>Alice: 5</td>
</tr>
<tr>
<td></td>
<td>Bob: 5</td>
</tr>
</tbody>
</table>
Overview of the protocol

Updating a channel

<table>
<thead>
<tr>
<th>Old transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>A&amp;B: 10</td>
</tr>
<tr>
<td>Alice: 5</td>
</tr>
<tr>
<td>Bob: 5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>New transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>A&amp;B: 10</td>
</tr>
<tr>
<td>Alice: 4</td>
</tr>
<tr>
<td>Bob: 6</td>
</tr>
</tbody>
</table>
### Revocation mechanism

#### Signed by Alice, held by Bob

<table>
<thead>
<tr>
<th>A&amp;B: 10</th>
<th>( \Delta t &gt; 1 \text{ hour} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Alice: 5</td>
</tr>
<tr>
<td></td>
<td>Bob: 5</td>
</tr>
<tr>
<td></td>
<td>( r_{kB} )</td>
</tr>
<tr>
<td></td>
<td>Alice: 10</td>
</tr>
<tr>
<td></td>
<td>( h(r_{kB}) )</td>
</tr>
</tbody>
</table>

#### Signed by Bob, held by Alice

<table>
<thead>
<tr>
<th>A&amp;B: 10</th>
<th>( \Delta t &gt; 1 \text{ hour} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Alice: 5</td>
</tr>
<tr>
<td></td>
<td>Bob: 5</td>
</tr>
<tr>
<td></td>
<td>( r_{kA} )</td>
</tr>
<tr>
<td></td>
<td>Bob: 10</td>
</tr>
<tr>
<td></td>
<td>( h(r_{kA}) )</td>
</tr>
</tbody>
</table>
Revocation mechanism

Overview of the protocol

Revocation mechanism

Alice

Bob

h(rk_{A,n+1})

h(rk_{B,n+1})

\text{sig}(tx_{n+1}^B, sk_A)

\text{sig}(tx_{n+1}^A, sk_B)

rk_{A,n}

rk_{B,n}
What if Alice and Bob dont share a channel?

Alice  Indiana  Bob
### Hashed Timelock Contract

**Signed by Alice, held by Indiana**

| A&I: 10 | \( \Delta t > 1 \text{ hour} \)  
| | Alice: 5  
| | Indiana: 5  
| \( s; t < 18\text{h00} \) | Alice: 4  
| | Indiana: 6  
| \( r_k_l \) | Alice: 10  
| \( h(s), h(r_k_l) \) | 

**Signed by Indiana, held by Bob**

| I&B: 17 | \( \Delta t > 1 \text{ hour} \)  
| | Indiana: 12  
| | Bob: 5  
| \( s; t < 17\text{h00} \) | Indiana: 11  
| | Bob: 6  
| \( r_k_B \) | Indiana: 17  
| \( h(s), h(r_k_B) \) |
# Hashed Timelock Contract

## Signed by Alice, held by Indiana

| A&I: 10 | Δt > 1 hour  
 |         | Alice: 5  
 |         | Indiana: 5  
 | s; t < 18h00  
 |         | Alice: 4 - fee  
 |         | Indiana: 6 + fee  
 | rk_{I}  
 |         | Alice: 10  
 | h(s), h(rk_{I}) |

## Signed by Indiana, held by Bob

| I&B: 17 | Δt > 1 hour  
 |         | Indiana: 12  
 |         | Bob: 5  
 | s; t < 17h00  
 |         | Indiana: 11  
 |         | Bob: 6  
 | rk_{B}  
 |         | Indiana: 17  
 | h(s), h(rk_{B}) |
Wormhole attack

Overview of the protocol

Alice ➔ Attacker1 ➔ Indiana ➔ Attacker2 ➔ Bob

$ if s

h(s)

$ if s

s

s

$ if s

$ if s

$ if s

s
Desired properties

Honest participants cannot lose money

Honest participants get their fees
The Proverif tool

```
new sk: private_key;
in(public, tx: transaction);
out(public, rk: revocation_key);
event Agreed(tx).
```

```
query attacker(sign(tr, skAlice))
==> event(Agreed(tr)).
```

```
true
false
cannot be proved
timeout
```
Modelling a transaction

Transaction is represented by a quadruplet $\rightarrow tx = (pk_1, pk_2, b, h)$
Modelling a payment channel

4 processes

Open -> Update
Close -> Update_LN

Challenge: passing the state from one process to the other.
Modelling the whole network

Threat model:
- Honest agents communicate via authenticated and secret channels
- All agents can be compromised
Modelling the properties

- Indiana cannot lose money
  - **No money blocked**: Indiana can always close the channel
  - **No punishment possible**: Attacker cannot punish Indiana
  - **Defense against old states**: Indiana can punish old transactions
  - **Unforgeability**: attacker cannot forge transaction
- Indiana gets the fee
  - **Atomicity**: when Alice has paid, Indiana is able to debit
<table>
<thead>
<tr>
<th>Obstacle encountered</th>
<th>Solution adopted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>Not modeling it</td>
</tr>
<tr>
<td>Passing the state</td>
<td>Using events</td>
</tr>
<tr>
<td>Liveness property</td>
<td>Tweak it into a correspondance property</td>
</tr>
<tr>
<td>Unbounded number of agents</td>
<td>Reduction to a bounded model</td>
</tr>
</tbody>
</table>
Liveness property: Indiana always holds a non-revoked transaction

Correspondance property: if transaction n Indiana holds is revoked, Indiana holds transaction n+1
Reduced model

Theorem

Attack on the full network $\iff$ attack on a 4-agent chain + oracles
Reduced model

Theorem

Attack on the full network \implies attack on a 4-agent chain + oracles

The attacker can simulate processes thanks to oracles.

let signing_oracle(sk: private_key) in(public, tx: transaction);
    event oracle_signs(tx, sk);
    out(public, sign(tx, sk)).
Conclusion

Modeling the LN protocol

Expressing all properties as correspondance property

Using a reduced model and a pen-and-paper proof

Next step: take time into account