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

Context

Bitcoin's lack of scalability → off-chain protocols

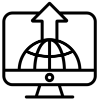
Protocol not proven → attacks

Our goal : prove the security of a fix

Core idea

Lock coins on the chain

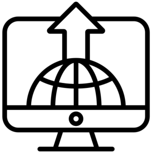
Exchange this money off-chain



Use the chain to cash in

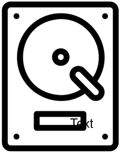
Opening a channel

Published on-chain



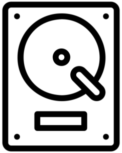
Funding transaction	
Input	Output
Alice: 5 Bob: 5	A&B: 10

Saved off-chain

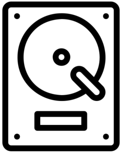


Closing transaction	
Input	Output
A&B: 10	Alice: 5 Bob: 5

Updating a channel



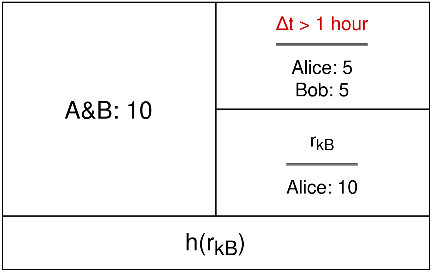
Old transaction	
A&B: 10	Alice: 5 Bob: 5



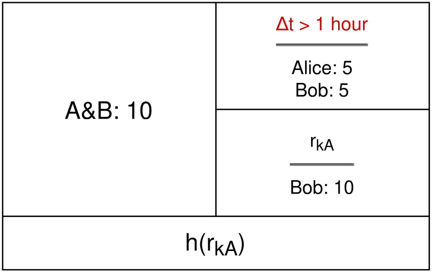
New transaction	
A&B: 10	Alice: 4 Bob: 6

Revocation mechanism

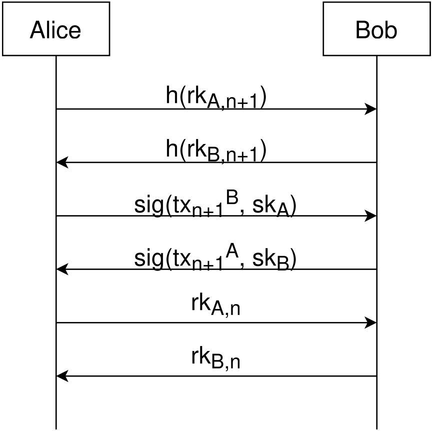
Signed by Alice, held by Bob



Signed by Bob, held by Alice

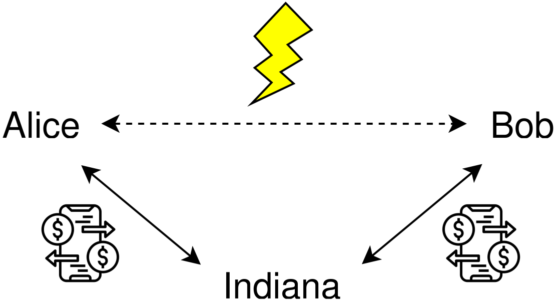


Revocation mechanism

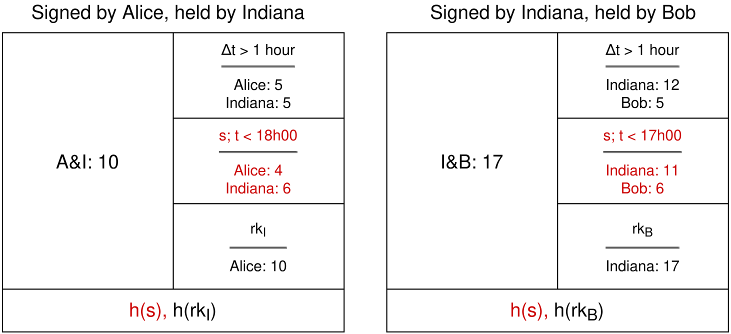


Lightning Network

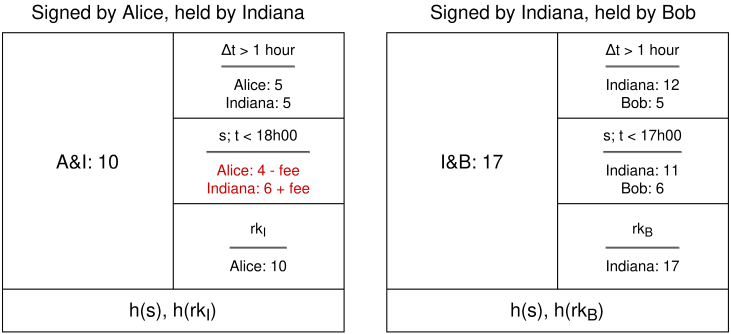
What if Alice and Bob dont share a channel ?



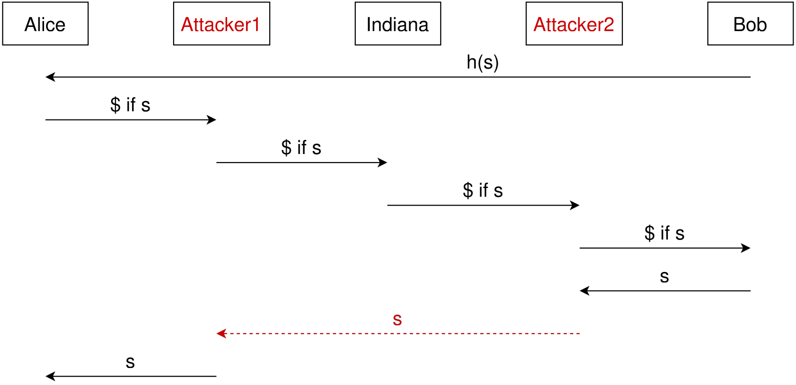
Hashed Timelock Contract



Hashed Timelock Contract



Wormhole attack



Desired properties

Honest participants cannot lose money

Honest participants get their fees

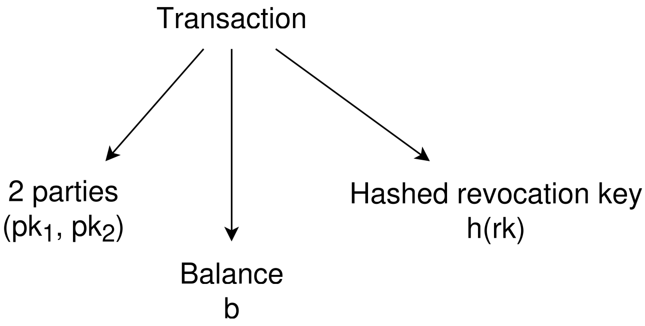
The Proverif tool

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

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

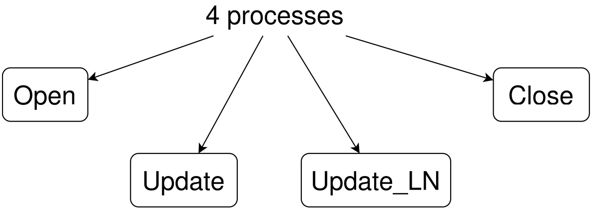
- true
- false
- cannot be proved
- timeout

Modelling a transaction



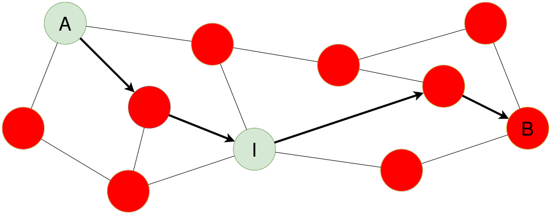
Transaction is represented by a quadruplet $\rightarrow$ tx = (pk1, pk2, b, h)

Modelling a payment channel



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

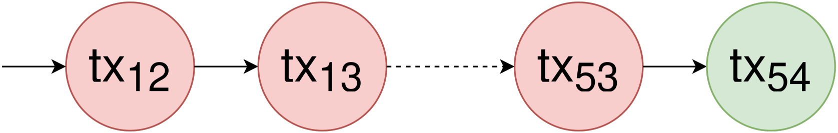
Difficulties

Obstacle encountered	Solution adopted
Time	Not modeling it
Passing the state	Using events
Liveness property	Tweak it into a correspondance property
Unbounded number of agents	Reduction to a bounded model

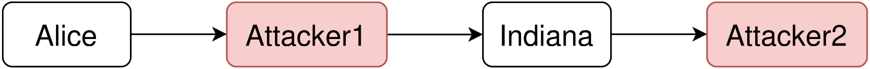
Trick

Liveness property: Indiana always holds a non-revocated transaction

Correspondance property: if transaction n Indiana holds is revocated, Indiana holds transaction n+1



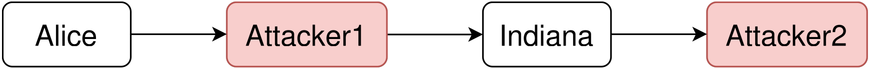
Reduced model



Theorem

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

Reduced model



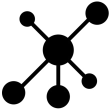
Theorem

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