2023

Bitcoin Bridges: Cure or Curse?

Alexei Zamyatin

Advancing Bitcoin 2023, London

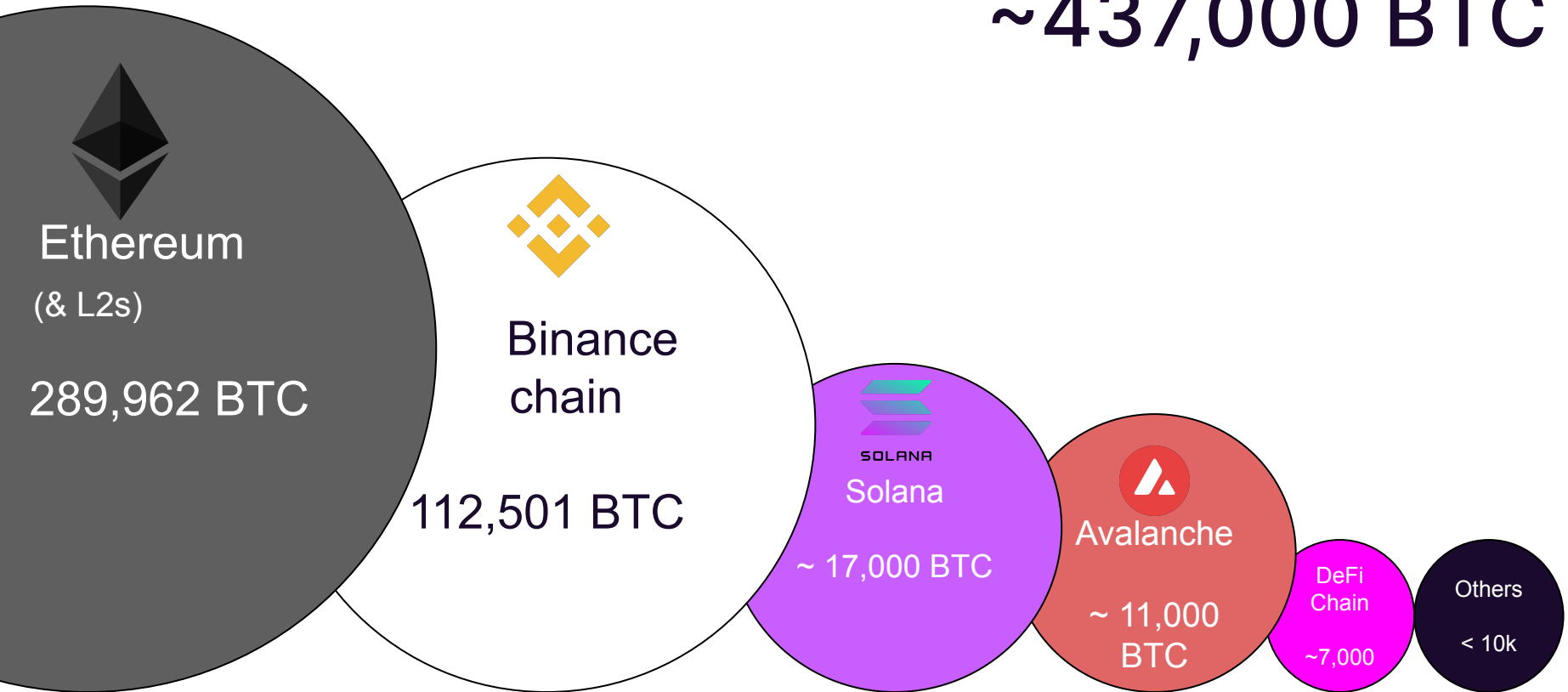
Agenda

- Wrapping: How to move BTC to other chains?
- Why is bridging so hard?
- How to build a decentralized BTC bridge?

Why should we care about Bitcoin bridges?

Bitcoin on other chains

~437,000 BTC



How much is decentralized?

How much is decentralized?

< 0.5 % (~2000 BTC)

Bitcoin Bridges 101

What do we want?

What?

1. **Deposit** BTC into an appchain (“application chain”)
2. **Use** BTC like a native asset on the appchain
3. **Withdraw** BTC back to Bitcoin

UX → Same as using BTC on a centralized exchange

Security → Always be able to get my BTC back

Reminder: Trust Models

	On Bitcoin	**BTC on other chains**
What do I need?	Bitcoin wallet	Bitcoin wallet Wallet on other chain; A way to bridge BTC
What do I trust?	Bitcoin network is secure; Wallet not corrupted;	Bitcoin network is secure; Other network is secure; Wallets not corrupted; Bridge is not corrupted (might be centralized).
How can I check?	Open source code	Open source code; Reputation of bridge if centralized.

Wrapping

BTC only exists on Bitcoin.

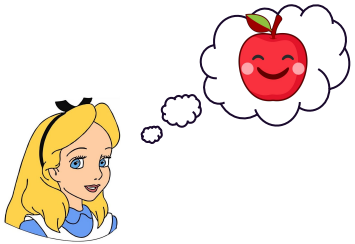
Wrapping = creating a 1:1 representation of BTC on another chain, i.e., as a native token.

In computer science terms:

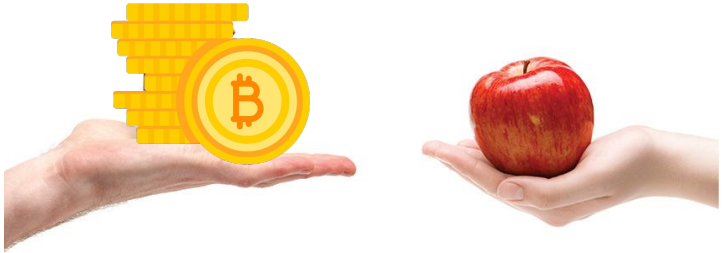
*“Obtain a **write lock** on the state of a UTXO and ensure **updates** made on the other chain are **applied** before the write lock is released”*

Why is bridging so difficult?

The good old Fair Exchange problem



Alice



How to make sure the exchange is always fair?

Bob

BTC

Apple

(In the digital world) someone **must make the first move**.

To ensure fairness in 100% of cases:

Needs a Trusted Third Party

Alice

Bob

BTC

Apple

Formal proof (computer science), 1999

How does this relate to bridges??

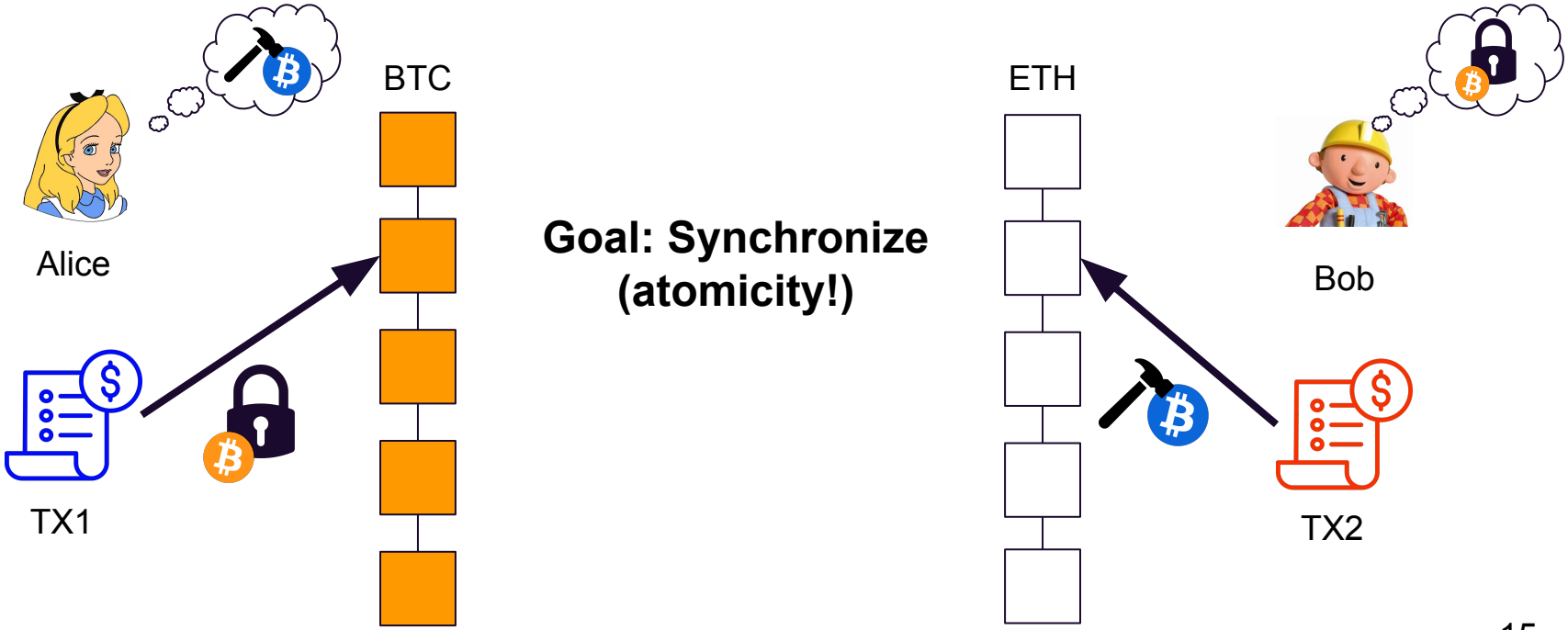
Wrapping = swapping BTC for wrapped BTC

Unwrapping = swapping wrapped BTC for BTC

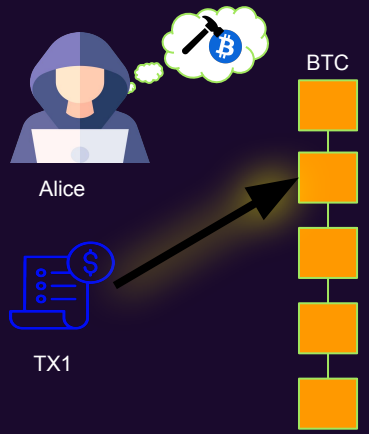
→ **Someone** needs to do the locking and unlocking of BTC on Bitcoin

Also: coin-swaps (if we have time)

The Cross-Chain Problem



Challenge of Bridging = Selecting a suitable 3rd Party

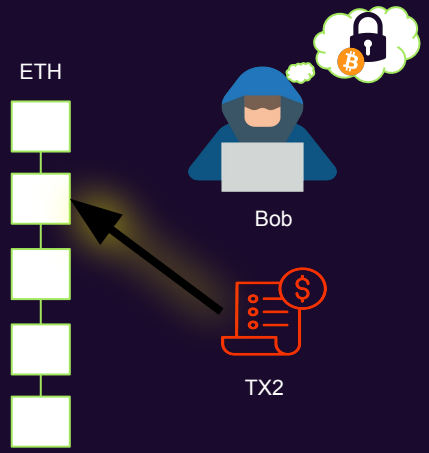


Centralized entity

Committee/Federation

Consensus of 3rd party network

Consensus of involved chains



Ideal case: Consensus of Chains

- Ethereum verifies Bitcoin SPV proofs
- Bitcoin verifies Ethereum SPV proofs
- 1 online party needed to relay proofs

Two models

- **Equal**: independent chains (e.g. Cosmos)
- **Hierarchical**: merged mining, sharding (e.g. Polkadot)

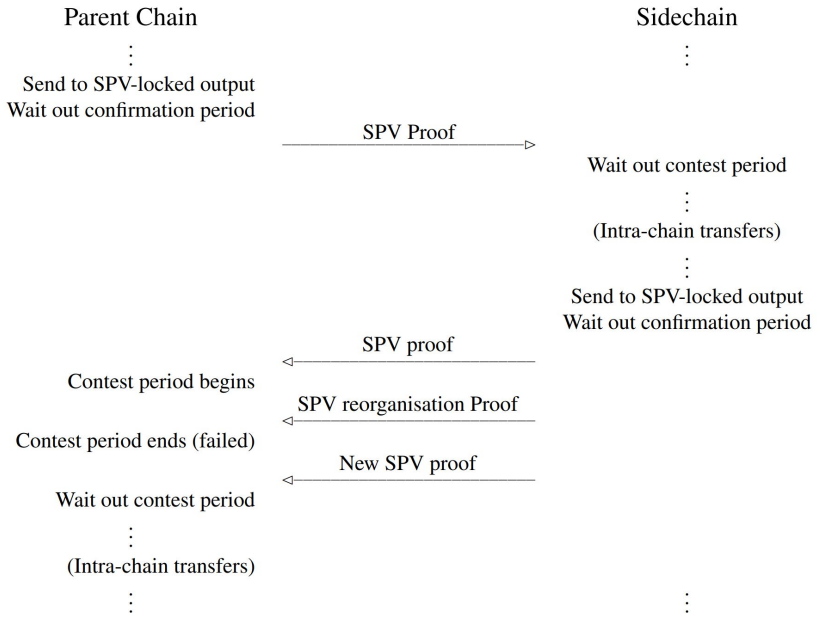


Figure 1: Example two-way peg protocol.

***2014** Sidechains paper by Back et. al.*

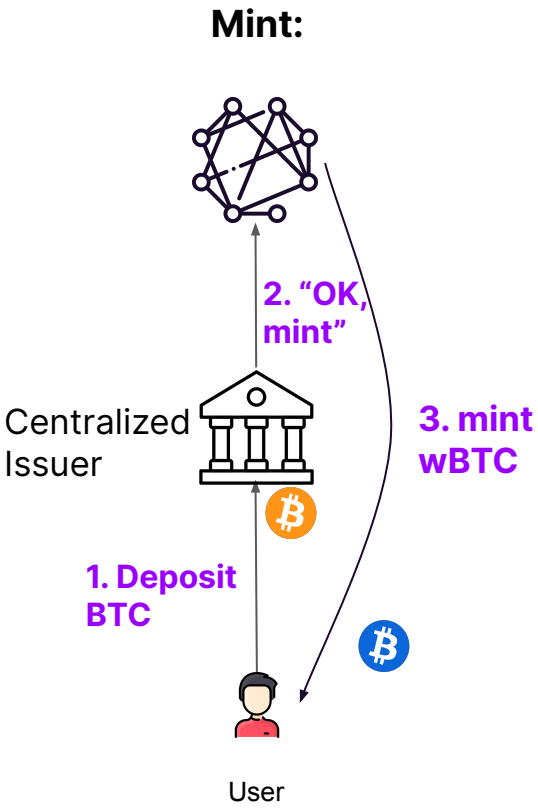
Bridging Bitcoin = Hard Mode

Goal: Lock / unlock Bitcoin based on events on other chains.

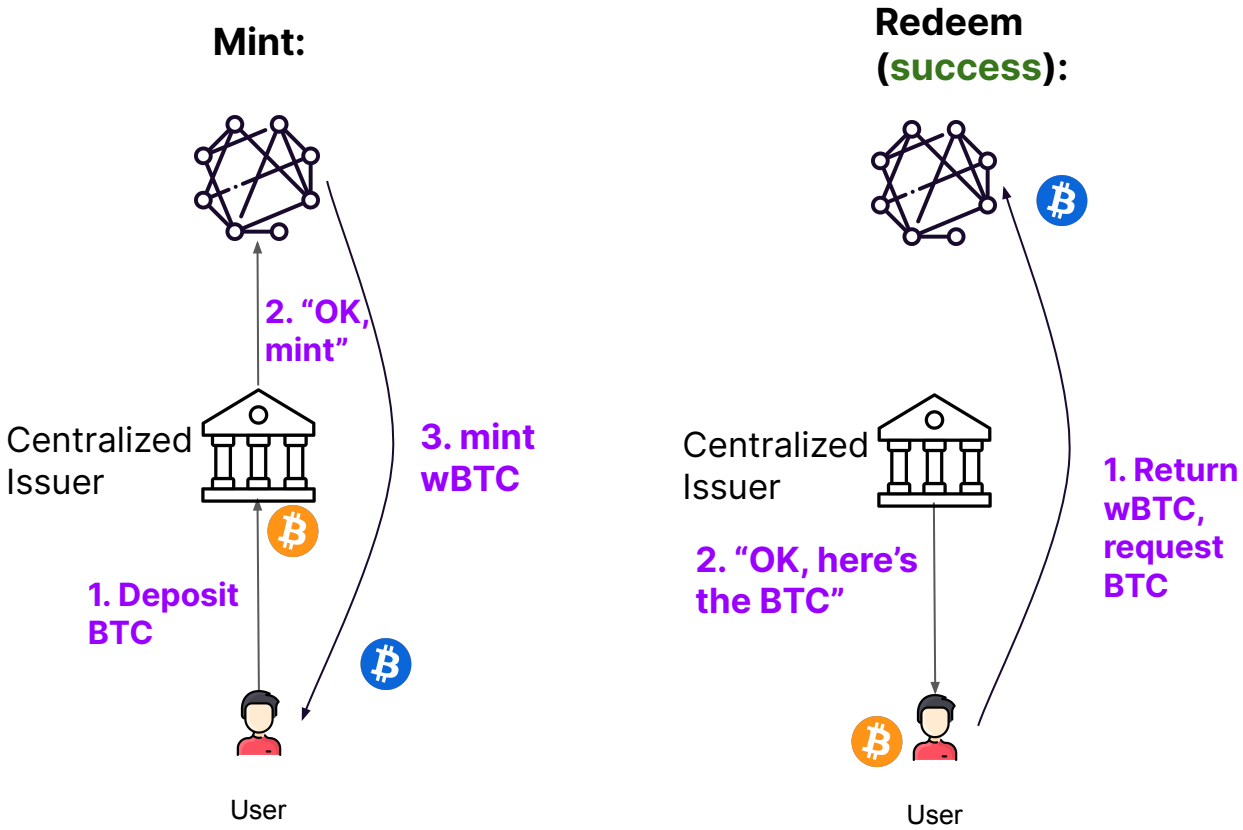
Problem: **Bitcoin does not know about other chains**

→ **Someone** needs to handle locking/unlocking of BTC manually

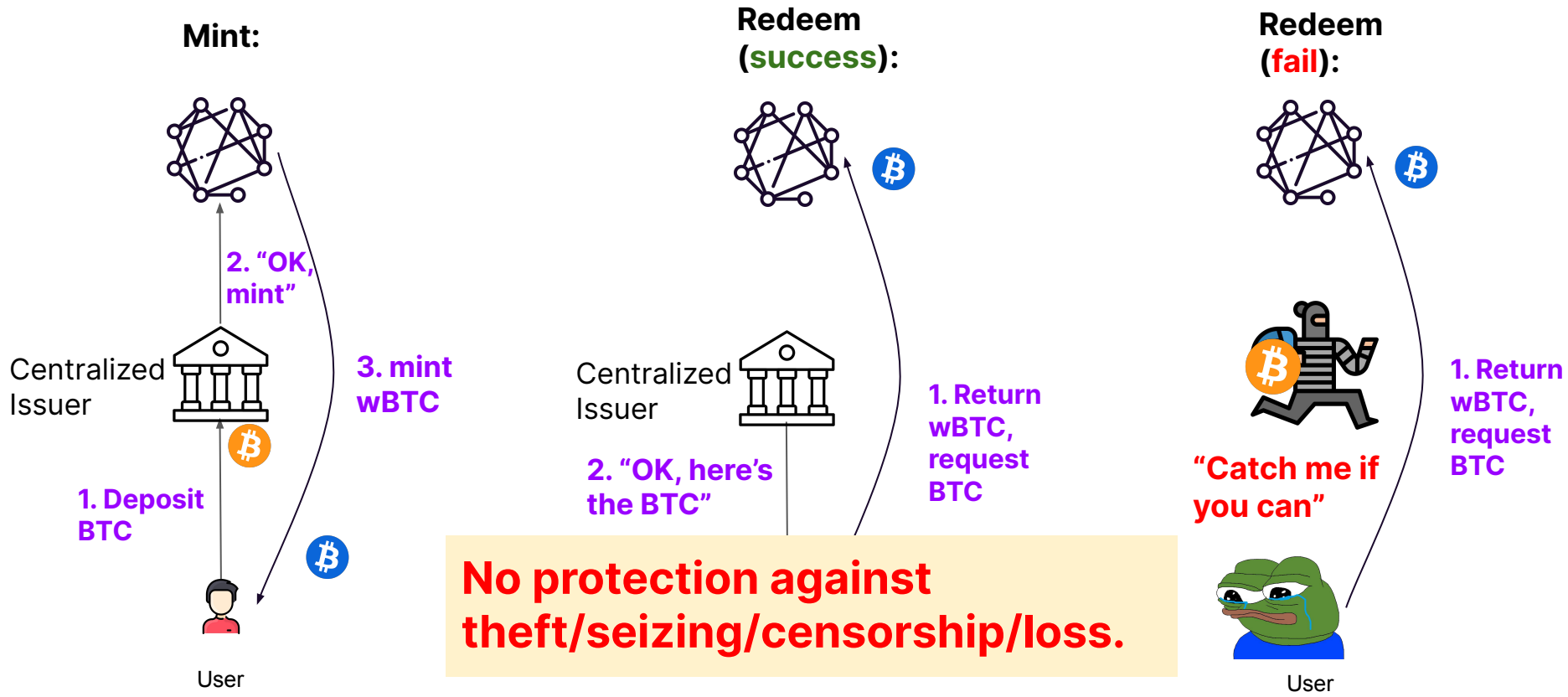
Most (centralized) bridges:



Most (centralized) bridges:



Most (centralized) bridges:



How to build a decentralized BTC bridge?

(without changing Bitcoin)

How to build a decentralized bridge?

1) Allow **anyone** to become an issuer/operator

How to build a decentralized bridge?

1) Allow **anyone** to become a operator/custodian

2) Realize this is even worse... now we're **sending BTC to random people on the internet**

How to build a decentralized bridge?

1) Allow **anyone** to become a operator/custodian

2) Realize this is even worse... now we're sending BTC to random people on the internet

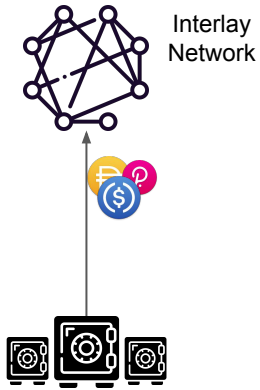
3) Use same tools as Bitcoin to fix:

- **Incentives:** operators lock collateral
- **Punishment:** if operator misbehaves, slash collateral (& reimburse victims)

Example: interBTC

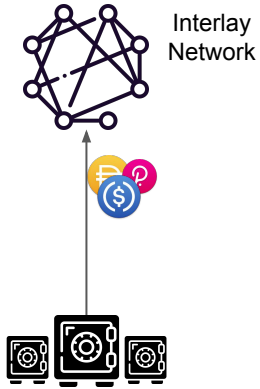
0. Vaults Register

Vaults deposit collateral

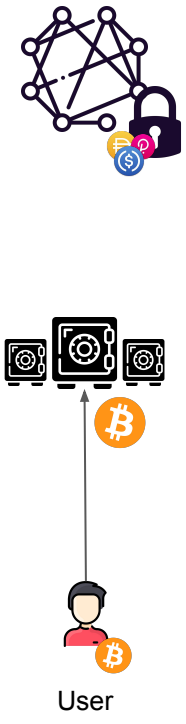


Example: interBTC

0. Vaults Register
Vaults deposit collateral



1. Lock BTC
User: Lock BTC

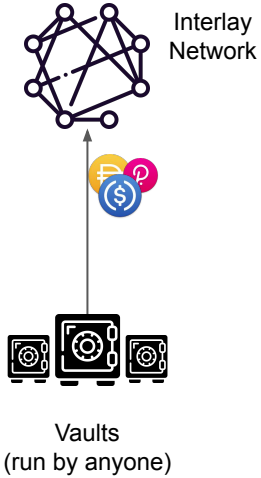


value(BTC) < value(collateral)

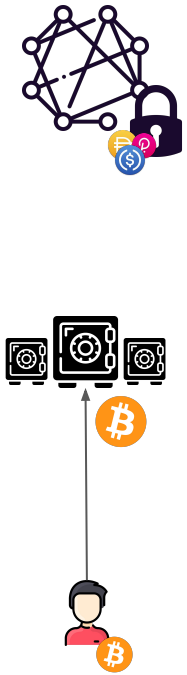
e.g. 150% collateralization rate for USDT

Example: interBTC

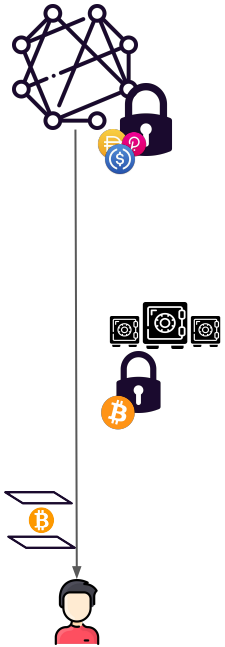
0. Vaults Register
Vaults deposit collateral



1. Lock BTC
User: Lock BTC

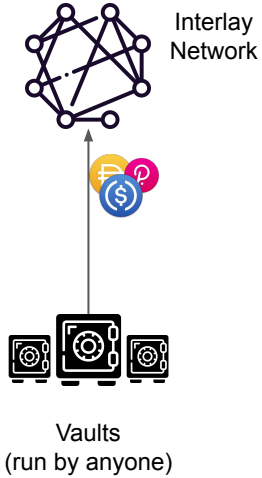


2. Mint iBTC
Chain: Mint iBTC to User

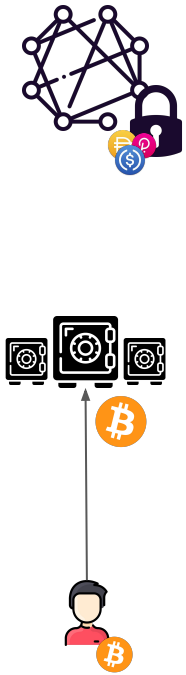


Example: interBTC

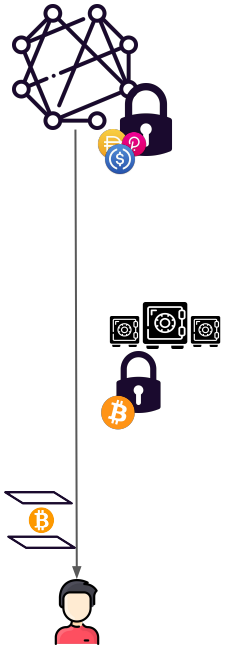
0. Vaults Register
Vaults deposit collateral



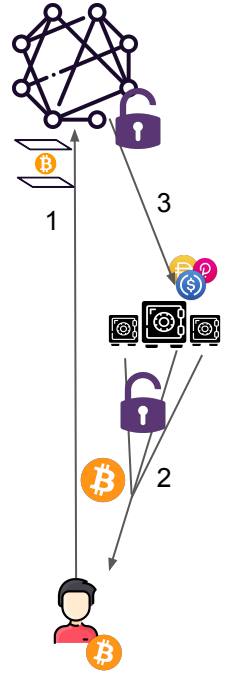
1. Lock BTC
User: Lock BTC



2. Mint iBTC
Chain: Mint iBTC to User

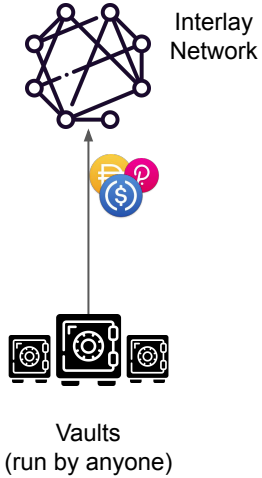


3a. Redeem (Success)
1. User returns iBTC,
2. Vault returns BTC to user,
3. Vault collateral unlocked

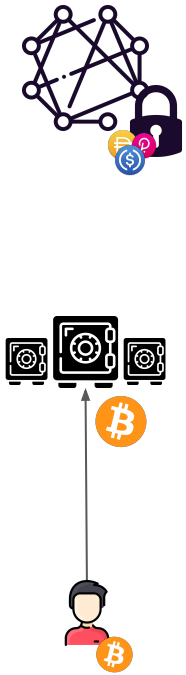


Example: interBTC

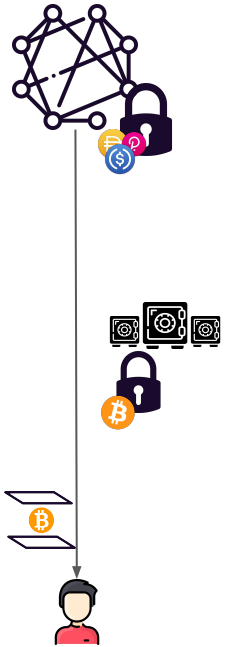
0. Vaults Register
Vaults deposit collateral



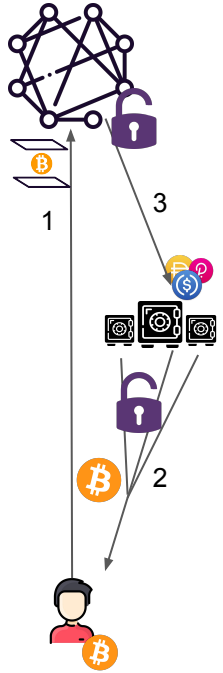
1. Lock BTC
User: Lock BTC



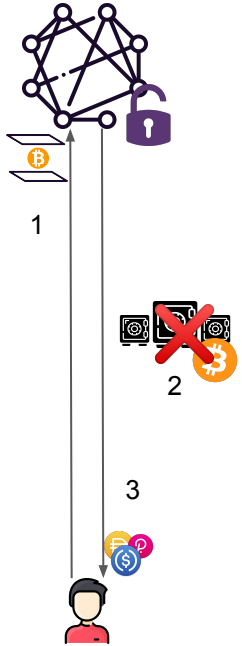
2. Mint iBTC
Chain: Mint iBTC to User



3a. Redeem (Good Vault)
1. User returns iBTC,
2. Vault returns BTC to user,
3. Vault collateral unlocked



3b. Reimburse (Bad Vault)
1. User returns iBTC,
2. Vault fails,
3. User is reimbursed (or tries different Vault)

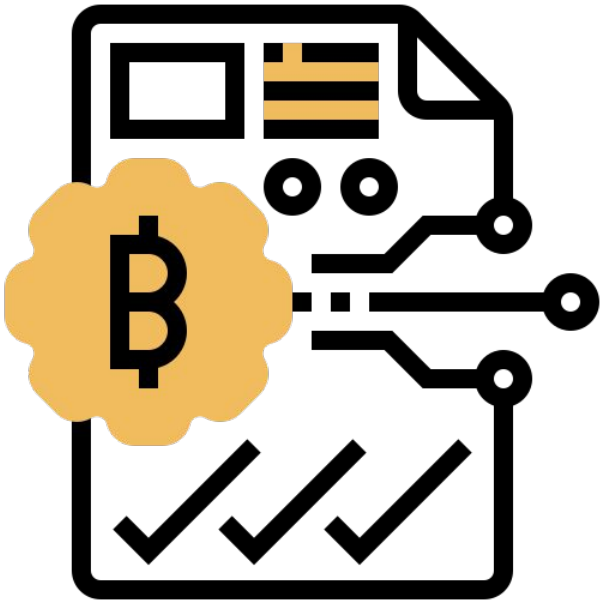


How to verify BTC payments?

Bitcoin light client (SPV) deployed as a smart contract

→ **Track** all Bitcoin block headers

→ **Verify** Bitcoin transactions



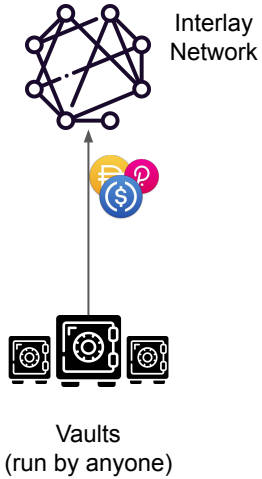
Security model: if in Bitcoin main chain → must be valid

(same as any mobile wallet)

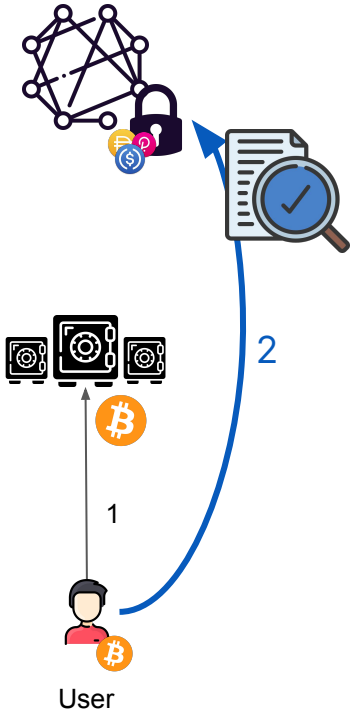
Someone needs to keep the light client up to date

Example: interBTC

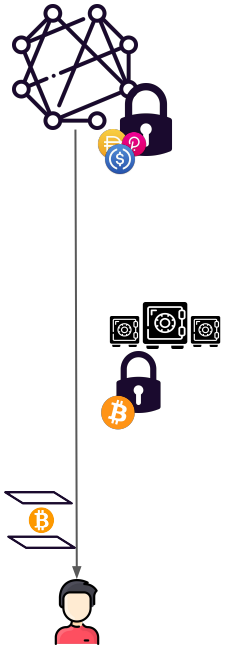
0. Vaults Register
Vaults deposit collateral



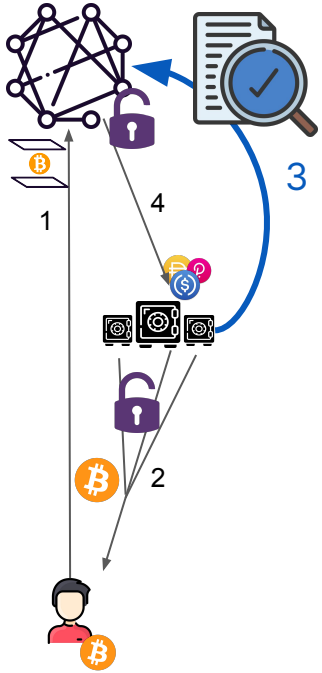
1. Lock BTC
User: Lock BTC



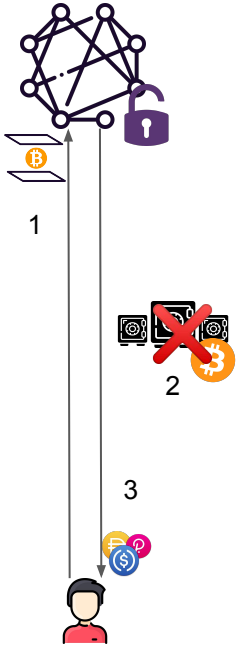
2. Mint iBTC
Chain: Mint iBTC to User



3a. Redeem (Good Vault)
1. User returns iBTC,
2. Vault returns BTC to user,
3. Vault collateral unlocked



3b. Reimburse (Bad Vault)
1. User returns iBTC,
2. Vault fails,
3. User is reimbursed (or tries different Vault)



Summary

1. **Issuer = smart contract**
2. **Permissionless network of Vaults (anyone can join)**
3. **Vaults are over-collateralized**
4. **Verification: Cross-chain light client**

What we skipped

- **Liquidations**
  - Vault collateral may decrease in value
  - Top or or get liquidated
- **Mapping accounts/deposits**
  - OP_RETURN (bad wallet support)
  - Better: ECDSA key derivation on the target chain
- **How do we know the price of BTC?**
  - Yes, needs oracles (can be mix of CEX and DEX)

Extensions / Flavors

- **Vault Committees**
  - Multisig / TSS
  - One big / multiple small
- **Collateral type vs amount**
  - Full / partial
  - Diversified (USDC, ETH,...) / native token
- **Security assumption**
  - Pessimistic / optimistic
- **Verification type**
  - Light client / 3rd party oracle / coinvote

How to build a decentralized BTC bridge?

Changing Bitcoin edition

Miner-enforced bridges

Bitcoin miners verify bridges, ensuring lock/unlock handled correctly.

For example: BIP300 + BIP 301

- BIP300: miners vote on peg-in & peg-out transactions over (long) periods of time
- BIP301: flavor of merged mining

Check out BIPs or layertwolabs.com for more details

Conclusion

Conclusion

Cure:

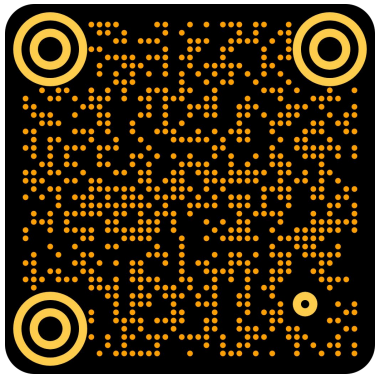
- More use cases & adoption of BTC without security risk to Bitcoin
- High demand for BTC across all other chains
- Objectively more secure than centralized exchanges (if using secure bridge)

Curse:

- 99% of BTC bridges are centralized
- Many BTC bridges wrongly market themselves as decentralized or non-custodial
- Decentralization is hard and comes at a cost (capital efficiency)

Side note:
There are **no**
non-custodial bridges...

... yet?



Thanks!

Feel to reach out at:

Twitter: @alexeiZamyatin
Email: alexei@interlay.io
Website: alexeizamyatin.me

Check out what we are doing at Interlay:

Twitter: @interlayHQ
Website: Interlay.io
Community: linktr.ee/interlay
interBTC bridge: app.interlay.io

SimpleBitcoinBridge()

Mint

1. **Minter** **locks** BTC on Bitcoin
2. **Issuer** on target network **verifies** the lock
3. **Issuer** **mints** a native “wrapped BTC” token at a 1:1 rate (minus fees)

Redeem

1. **Redeemer** **returns** wrapped BTC to **Issuer** on the target network
2. **Issuer** **sends** BTC to **Redeemer**‘s Bitcoin wallet at a 1:1 rate (minus fees)
3. Wrapped BTC is deleted (“**burned**”)

The Role of Issuers

2 responsibilities:

- **Custodian**: hold assets on source chain
- **Issuer**: handle mint/burn logic on target chain

Often, bridges will use fancy terms, obfuscating the trust model:

- Multi-party computation **= multsig**
- Threshold signatures **= multsig**
- Trusted hardware **= trust that there is no new Intel SGX hack**

These are all nice “additions”, and work in practice... until they don’t

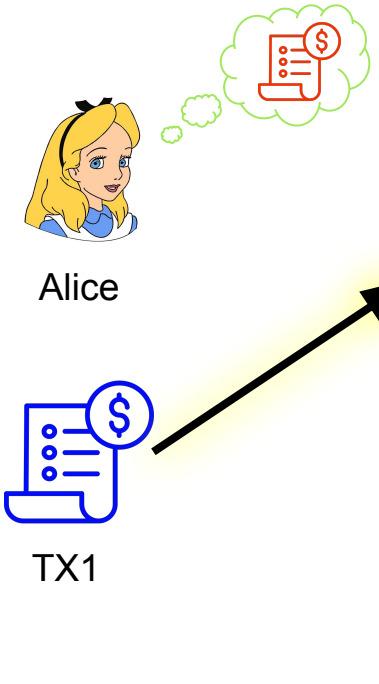
→ In the end, you trust that group of people will not steal your BTC

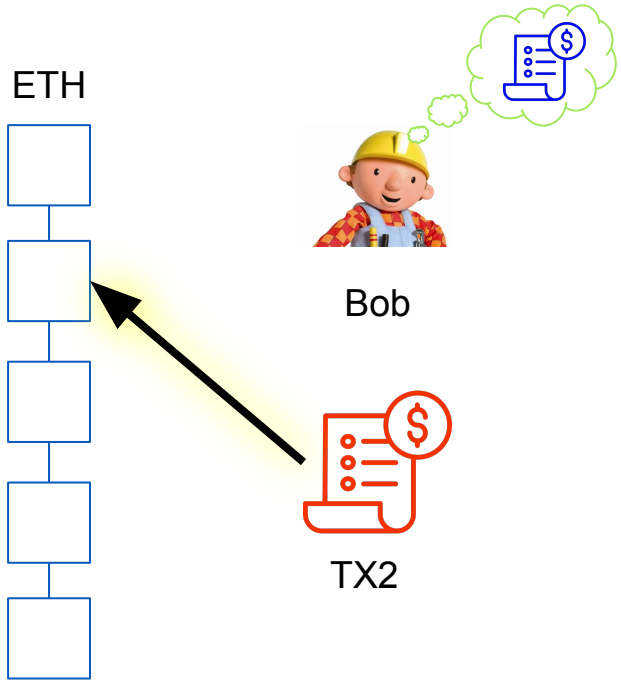
Atomic Swaps

Fair Exchange

TX1 gives BTC to Bob
TX2 gives ETH to Alice

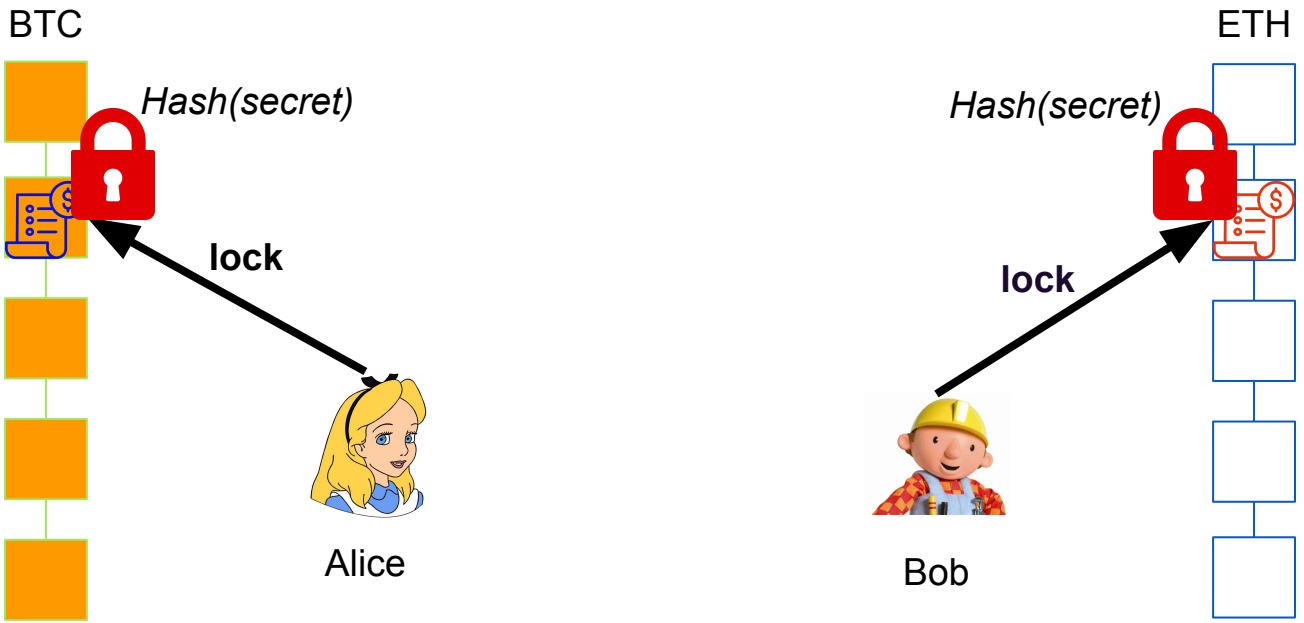
Fair Exchange of assets





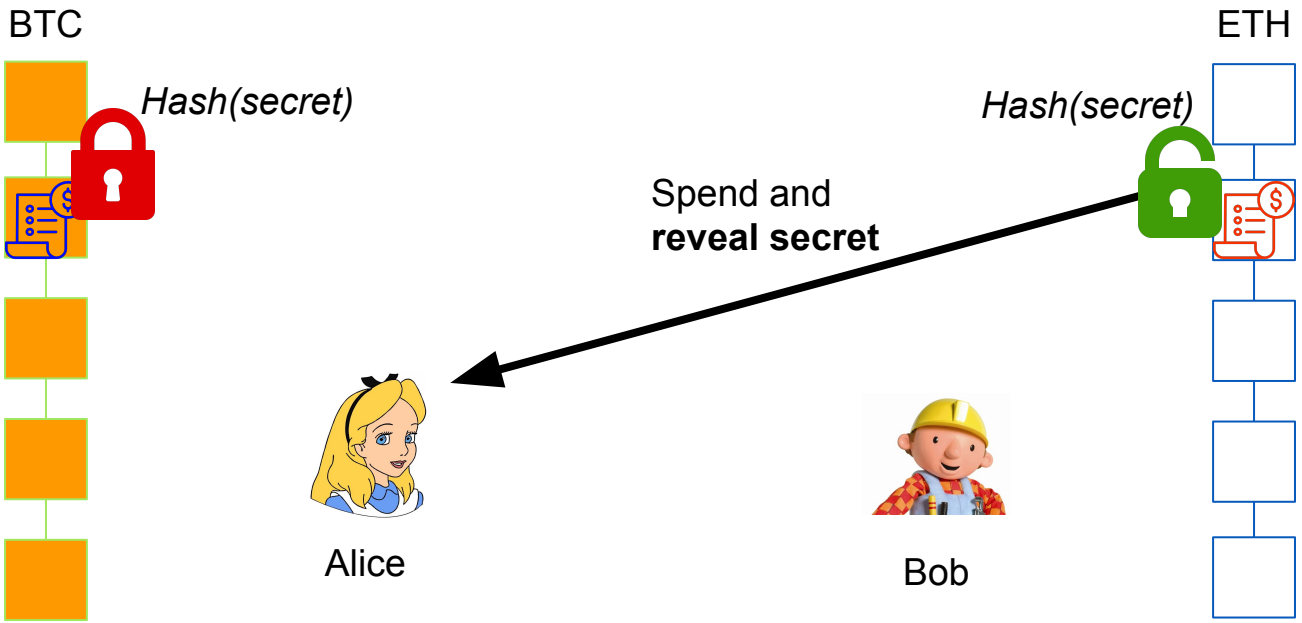
Atomic Swaps via HTLCs

Alice and Bob lock coins with the **same** lock on both chains.
HTLCs: hash lock (coins can be spent if pre-image/secret is revealed)



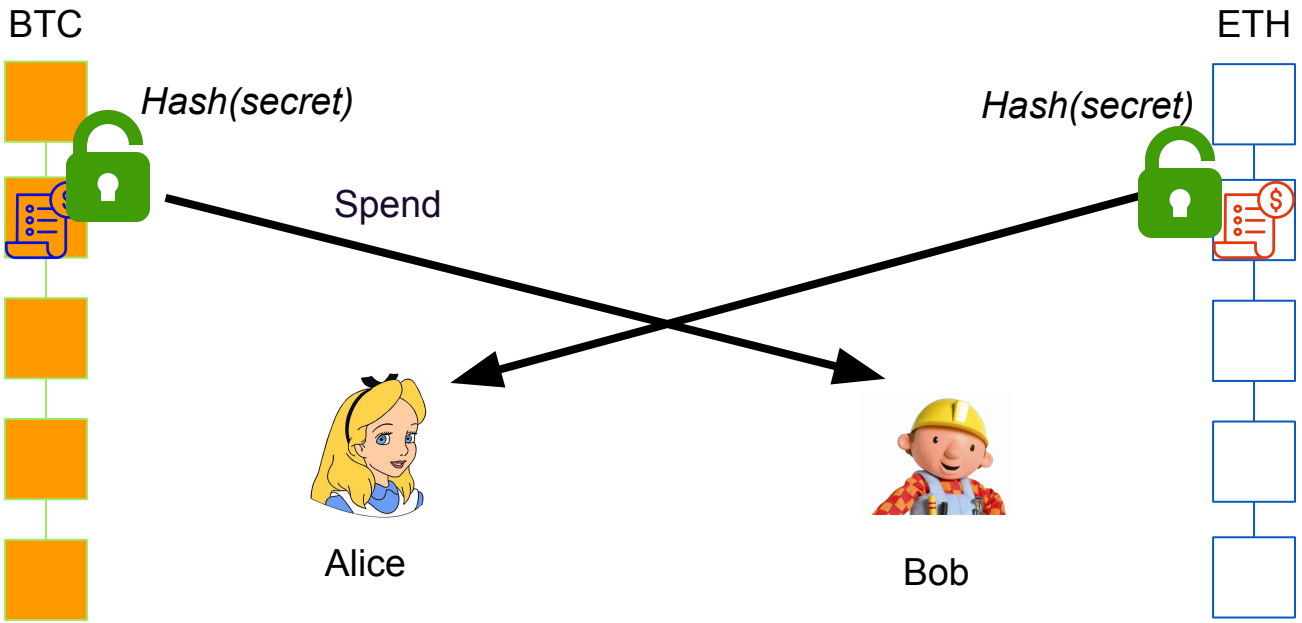
Atomic Swaps via HTLCs

If Alice spends Bob's coins, Bob can spend Alice's coins.



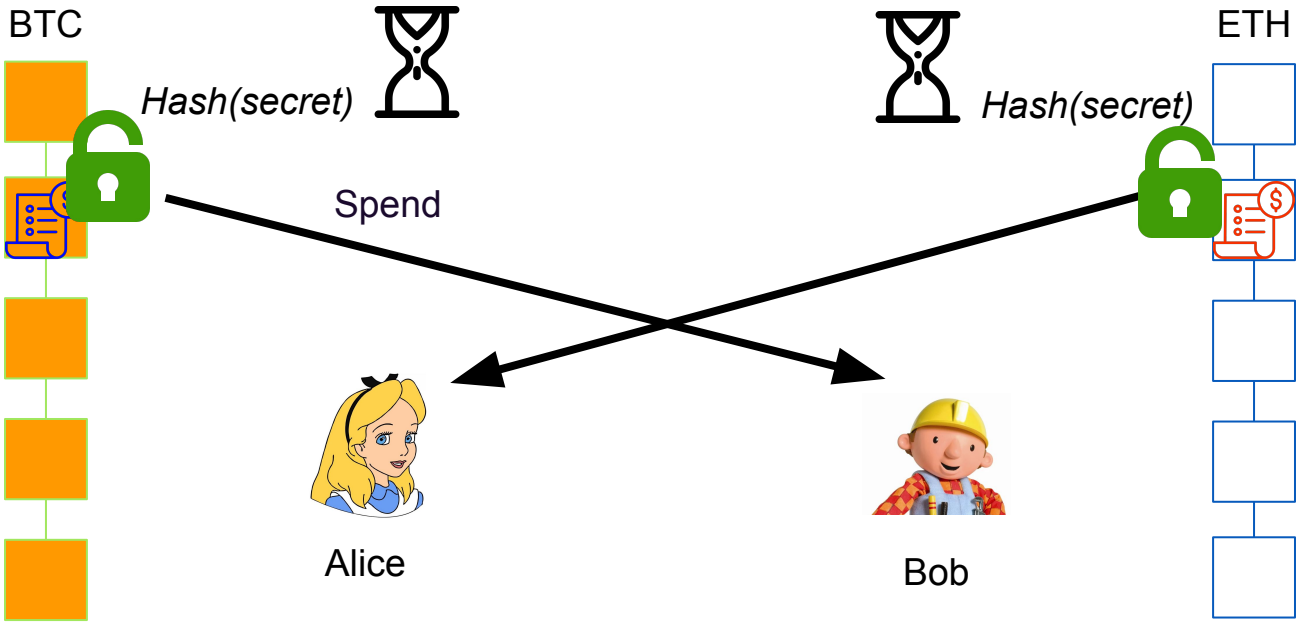
Atomic Swaps via HTLCs

If Alice spends Bob's coins, Bob can spend Alice's coins.



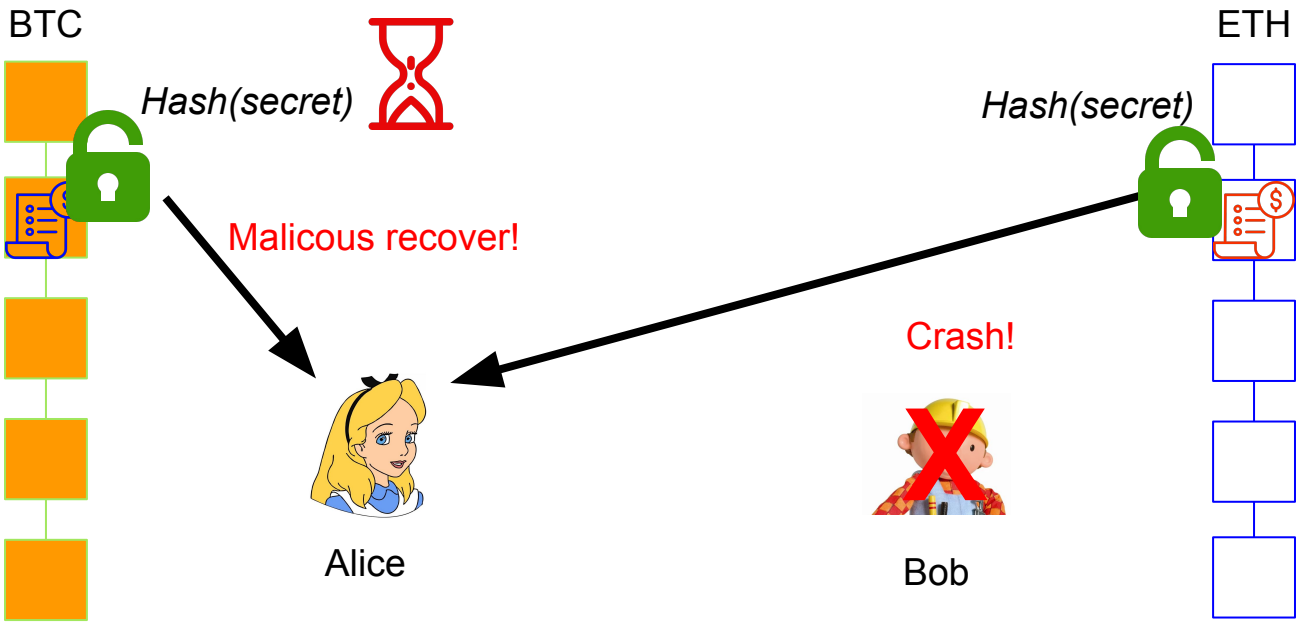
Atomic Swaps via HTLCs

Timelocks used to prevent indefinite lockup of funds
Alice and Bob can restore coins if nothing happens



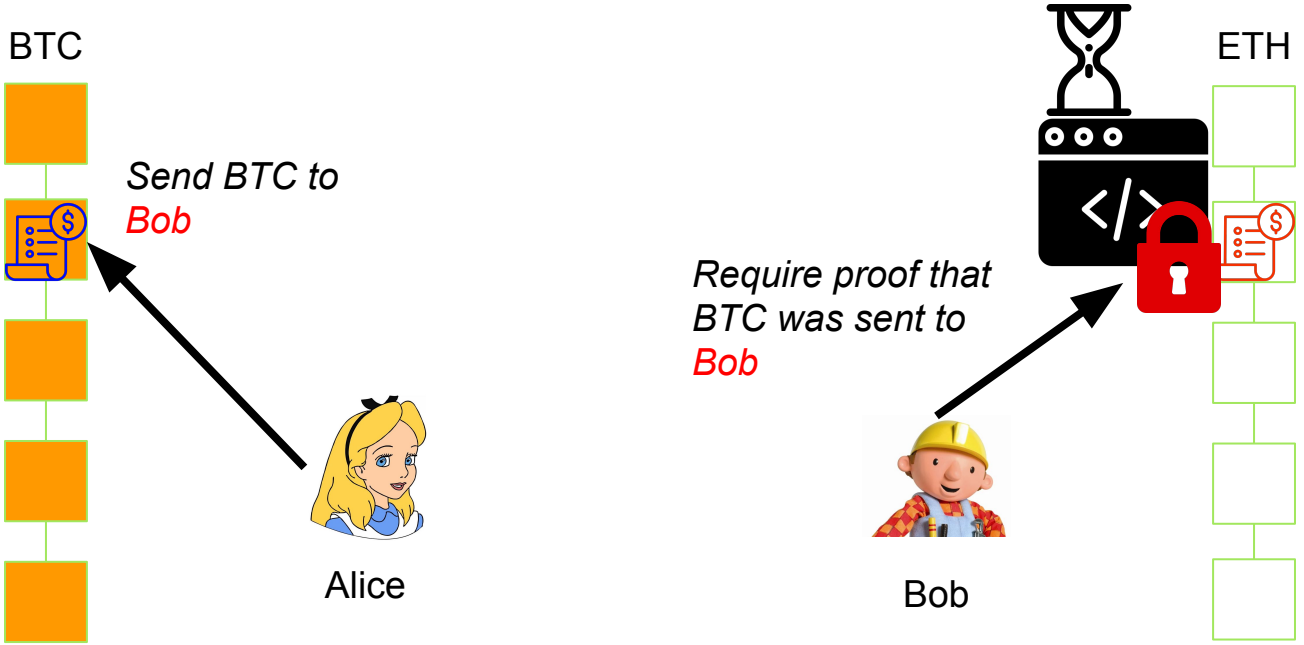
Atomic Swaps via HTLCs

Problem: Alice spends and reveals …. but Bob crashes.
Alice can maliciously recover her coins, “stealing” from Bob



Atomic Swaps via SPV Proofs

Bob locks ETH in smart contract.
Unlock condition: someone sends him BTC on Bitcoin.
Alice sends BTC to Bob.



Atomic Swaps via SPV Proofs

Alice proves to contract that she sent BTC to Bob
Contract releases ETH to Alice.
Bob **can be offline** the entire time. Alice bears risk!