ZBTC: Zero-trust wrapped Bitcoin on Ethereum

Liam Zebedee | https://liamz.co.
April 28, 2019.

Topics: ethereum, bitcoin, blockchain interop, stateless SPV, collateralised debt positions, uniswap

Abstract. Ethereum is on track to become the global trust layer for information. Bitcoin, as the forerunner, keeps its hold as a reserve currency. Bitcoin is available on Ethereum as an ERC20, ‘wrapped’ by a trusted set of actors. We propose to implement a trustless version of Bitcoin, based off the same incentives in collateralized debt positions (CDP’s). By using a chain oracle, that can be trustlessly operated by anyone, we can prove cross-chain collateral (Bitcoin) has been deposited into a CDP. Then we require an equivalent of Ether to be collateral, such that the actors can be incentivised to process ‘redeeming Bitcoin’ from WBTC, as well as insuring against race attacks between the two chains. In doing so, we can build the first real trustless wrapped Bitcoin on Ethereum, and pave the way for future innovations of cross-chain assets.

Overview

The Ethereum blockchain is set to become the infrastructure layer for global trust. Societe Generale issues $112M bond, EY has public domain-ed all its work on deploying private contracts on Ethereum, and the best scaling research (sharding, ZKP's) is being done on Ethereum.

Tokenisation and #DeFi is most innovative on Ethereum, where Maker's DAI, AZTEC, 0x, Uniswap, Augur and many other new experiments are deployed.

Bitcoin is set to become the reserve currency. Due to its first-mover advantage, it is already well-ingrained as a new asset class (hedge funds, trading on Bitmex etc.) and payment type (for restaurants and remittance alike).

On Ethereum, Bitcoin has already been deployed as WBTC (wrapped BTC), however it relies on trusted parties to function. The XCLAIM model proposes a trustless bridging of Bitcoin using cryptoeconomic design, wherein actors called vaults are responsible for proactively proving adherence to the protocol, lest they be slashed and their collateral lost. It supports Bitcoin's being bridged into an ERC20, wherein they can be traded, and **non-interactively** redeemed into a Bitcoin address from an Ethereum transaction.

We believe XCLAIM is a workable model, however it is shallow in two details - the chain oracle and the price oracle. The **chain oracle** (or 'relay') is necessary to prove the transactions on the Bitcoin blockchain. This can be feasibly implemented as a smart contract, due to the low complexity of verifying Nakamoto consensus. But no clear answer has been made on how this oracle can be sustainably funded. Likewise, how can we securely implement a **price oracle**? The risk being, if there is a price arbitrage between the staked collateral and the value of the backing bitcoin, the game equilibria falls apart. A malicious trusted oracle, could mint a large number of wrapped Bitcoin, and destabilise the value of the bridged token.

We propose realising this vision of trustless wrapped Bitcoin, ZBTC[1], on the basis of the new cryptoeconomic primitives of 1) collateralised debt positions and 2) stateless SPV proofs. Reframing XCLAIM's components as a CDP, we can naturally see how a 'stability fee' can be appropriated as sustainably funding the Bitcoin chain oracle. Likewise, we propose Uniswap as a natural complement to trusted price feeds as in MakerDAO.

[1] Throughout this document, we refer to this model as ZBTC, to differentiate from WBTC.

Cross-chain Collateralised Debt Position

A simpler understanding of XCLAIM is through the lens of Maker's CDP primitive.

A CDP is a smart contract, which mints debt based on collateral. In MakerDAO, the debt is DAI, the collateral is Ethereum. The primary clause of a CDP is the liquidation mechanics, which ensures that the debt is collateralised to a **ratio**.

XCLAIM is rather similar:

- The debt is the bridged token, ZBTC
- The collateral is Ether and Bitcoin
- The minimum collateralisation is 150%
- The liquidation conditions:
    - If undercollateralised (same as DAI)
    - *If unbacked by equal Bitcoin*
    - *If the vault does not correctly redeem ZBTC for Bitcoin*

Collateral on another chain

The second liquidation condition is an important difference. Unlike DAI, which can be generated up to the minimum collateralisation ratio, ZBTC can only be generated up to the amount of Bitcoin collateral that was deposited. Otherwise we are issuing what resembles a derivative. A synthetic asset based on Bitcoin's price is different to a debt token which the owner can redeem for the underlying Bitcoin. For one, the synthetic asset would need to mirror the monetary policy of Bitcoin (21M, reward halving) to retain the same scarcity. Secondly, without redeemability, the token has a different utility - consider a scenario where the Lightning network is cheaper in one country than the Ethereum network is.

How do we determine the Bitcoin deposits? Using proofs from a chain oracle (described in the next section) we can determine accounts and balances from transactions. Due to the parallelism of cross-chain interactions, we cannot implement the logic above in the generate/burn debt methods of the CDP. *e.g.* the Bitcoin could be moved before the contract is aware of the discrepancy.

So, using a chain oracle, XCLAIM can be conceptualised as a CDP that is collateralised by assets from two chains.

Redeeming ZBTC for BTC

The ZBTC needs to be redeemable for actual BTC. This falls outside the analogy of a CDP primitive, and into the domain of cryptoeconomic incentivisation. We found it more useful to

frame the system as a CDP for starters, and no doubt this can be reformulated into a blockchain design pattern. For now, we'll describe it as is.

XCLAIM refers to the actors that maintain these 'CCDP's as vaults. To incentivise the vault to transfer the Bitcoin when it has been redeemed, the vault must prove adherence to this protocol:

1. ZBTC owner calls `CCDP.redeem(amount, address)`. Their ZBTC is escrowed in the CCDP.
2. The vault must now transfer `amount` to the Bitcoin `address` specified
3. `CCDP.completeRedeem(address, amount)` can then be called, where the vault proves the transaction. The escrowed ZBTC is then burnt and the fee is paid.
4. If the vault does not act correctly within N blocks, they can be liquidated.

Hence the third liquidation condition.

Users must also supply some fee to the CDP owner to cover the cost of transacting on Bitcoin's network. The costs of a basic value transaction are pretty different between the two networks:

- **Bitcoin** - $0.88 USD (2019-05-03)
- **Ethereum** - $0.028 USD (2019-05-03)

Fortunately, only the creation of a CDP and the redeem require transacting on Bitcoin's network. A simple approach is letting CCDP's set the `btcTxFee` they charge, allow any ZBTC holder to call redeem on a CCDP, and use the free market to incentivise the lowest fees.

There are a couple other avenues for exploration:

- Schelling oracles / prediction markets of Bitcoin fees
- Calculating the median of transaction fees, from the chain oracle
    - what if the chain oracle is lagging behind?
- Incorporating the fee as a "stability fee" that can be governed
    - doesn't scale with number of redeem's requested

Given that we already need to incorporate the Bitcoin node software for being a vault, it is simpler and likely fairer to just get `feeTxConfirmTarget` from an API, and update the CDP accordingly.

Risk governance in Maker vs. ZBTC.

Since we are assuming two chains, there is inherently more risks (twice as many vectors for hacking, chain-scale forks etc.).

In MakerDAO, there is considerable risk mitigation already underway in the governance design of the system. It would be foolish to assume we could reimplement all of these controls, with the same precision of understanding and nuance. We could 'piggyback' on DAI and use their token for collateral, removing the risk of Ethereum's volatility.

Principally, the purpose of MKR is to govern the system which facilitates DAI. Similarly, we must account for governance of the system which facilitates bridging Bitcoin's value:

- **Chain forks** - which coin will be used?
- **Bitcoin transaction fees** - how can we ensure that the cryptoeconomic incentives are parity on both chains?
- **Changes in protocol formats for BTC headers** - how can we ensure the continued correct operation of the chain oracle?

For the purposes of a proof-of-concept, these issues of governance are out-of-scope. They are more suitable to be discussed with a team with the risk expertise of MakerDAO.

Incentivising the Chain Oracle

The Bitcoin chain oracle thus is core to the functionality of this system. XCLAIM proposes btcrelay as a foundation, which is quite brittle code in Serpent, but implements much of the logic around longest-chain rules etc. However the unresolved issue remains, from the stagnation of Btcrelay, of how can we incentivise keeping the chain oracle, up-to-date? To quote:

> "the hurdle is when BTC Relay is generating "revenue", then relayers will want a piece for it and will continue relaying, but to get "revenue" BTC Relay needs to be current with the Bitcoin blockchain and needs the relayers first"
> https://github.com/ethereum/btcrelay/issues/50#issuecomment-406979858

Who uses the chain oracle?

- **Vaults**: when they establish a CCDP, prove redeem's are processed
- **Keepers**: when they liquidate contracts

What is the cost of verifying Nakamoto consensus on Ethereum? (see block structure)

- Per block:
  - Verifying the POW - 1 SHA-256
  - Data of block headers - 80 bytes
- Per day:
  - Block every 10 minutes
  - 60/10*24 = 144 blocks
  - 144*80 = 11,520 bytes / day
- Estimation from btcrelay (which includes storing all past headers):
  - 160-200K GAS (ref) = $0.08USD

- = $11.52/day
- = $4,204.80/year

The cost per year is comparatively small, any medium-sized crypto business could finance this for a year's worth of operations. Compared with a diversity of actors submitting updates to the oracle, it is vastly simpler for a single actor to both update and receive the fees. Otherwise there are situations where the lag in unprocessed POW headers incurs disproportionate cost for some actors who update.

Unfortunately, btcrelay has shown this model of cost and incentive breaks down in practice. While the cost is relatively low ($11/day) if relayers are constantly available, there are no incentives for availability. There was a proposed fee market solution, but even this relies on the potential of future consumers of the btcrelay API to pay, rather than claiming a reward.

We believe a newer innovation, stateless SPV's, solves the incentive incompatibility. Stateless SPV approximates the finality of a transaction based on its **accumulated difficulty**. Instead of verifying and storing all block headers to date, we compute the cumulative difficulty of a set of headers. As long as the oldest header includes the transaction, and each following header builds upon the last, we can approximate the economic cost to making a fraudulent transaction. The current Bitcoin difficulty, multiplied out by six blocks, can be an approximate cost of Bitcoin's transaction finality. As there is not much material out yet (it was introduced in March 2019), pseudocode below is included to explain better:

```
def work(header):
        """Returns CPU work expended in a block header"""
        assert hash(header) >= header.difficulty
        return header.difficulty

def cumulative_work(header):
        if header.prev_block:
                return work(header) + work(header.prev_block)

        return work(header)

def longest_chain(head1, head2):
        """Determines which head refers to the longest chain of CPU work"""
        if cumulative_work(head1) > cumulative_work(head2):
                return head1
        else:
                return head2
```

This approach is rather bespoke - due to the simplicity of Bitcoin's consensus algorithm, and the network effect of its hashpower, stateless SPV is matched to this use case. If the hashpower were lower, it would be vastly more vulnerable to attacks.

To incorporate stateless SPV into a chain oracle, we would have to account for a couple of changes:

1) **Long-range attacks**. A determined attacker can build a fraudulent chain of minimum length (6 blocks), and use this proof to mint fake ZBTC. How can we protect against this? We can enforce that every proof needs to link to a previous checkpoint block. And the checkpoint block is then set to the newest finalised block (the 6th block).

   The chief benefit of stateless SPV, is that it exploits the storage-memory tradeoff inherent in Ethereum smart contract cost. Memory is an order of magnitude cheaper than storage, so verifying a chain of headers is much cheaper than storing a new one. Storing one slot in Ethereum is 20k GAS, which is sufficiently negligible to disincentivise nodes.

2) **Difficulty retargeting.** Bitcoin difficulty is retargeted every 2016 blocks, such that the block time is on average 10 minutes. We can use the difficulty target of the checkpoint block introduced above, as our threshold.

Other models using proof-of-stake sidechains, ZKSNARKS, Truebit verification games, NiPoPoW, FlyClient could be explored, but this model is evidently simple and pragmatic for today.

Processing proofs

We need to verify proofs but only if the transaction has been finalised. 6 is the recommended number of confirmations to wait for, 4 blocks is the longest chain re-org to date in Bitcoin's history. So in our `chainOracle.verifyTx` it would be possible to verify against only the 4th most recent block.

Building a Bitcoin price oracle

The Bitcoin price oracle is essential to the operating of the CCDP. We propose a combination of MakerDAO-style price feeds for Bitcoin, and an additional signal from a Uniswap exchange. Since Bitcoin can become liquid for Ether, it is possible that the liquidity of an automated market maker could accurately match the price, while remaining resistant to attacks.

Using Uniswap as a decentralized price oracle

Uniswap is an automated market maker, which prices using a constant product formula.

We can set the initial price of WBTC by offering up an amount of ETH. This is the original BTC into the system. Thereafter, arbitrage opportunities between ETH:BTC and exchange's prices will arise, trades will occur, and update the price.

Using a medianizer for the price feed is essential. An actor could buy up lots of ETH or BTC, skewing the pairing. Then, in the same stroke, use this new price to liquidate many CDP's.

User Experience

Keepers
Keepers are profit-motivated actors in the system, who perform the following:

- Update the chain oracle
- Liquidate risky CDP's
- Liquidate unbacked CDP's

Minting ZBTC
Like DAI, once ZBTC is minted, can be sold/transferred/exchanged, without interaction with the Bitcoin chain oracle.

Transaction cost
Stateless SPV proofs are an order of magnitude cheaper than btcrelay proofs. As mentioned before, Bitcoin proofs would only need to be submitted by the keepers in any case.

Proving a transaction in the Merkle tree is O(log(N)), and given 359,000 Bitcoin transactions occurred on the 28-04-2019, that's roughly 18 nodes worth of data at minimum. This is sizable but not terrifically expensive.

Potential Impact

While WBTC is out there, it's 24h volume is small (16k EUR) compared to that of the Bitcoin network. Without the necessity of trust, we envision WBTC being a fully-fledged competitor to 'native' Bitcoin.

We also would highlight the potentiality of wrapping other cryptocurrencies, such as Grin. Recently the first atomic swap was performed between Grin<->Ether. Although balances are hidden, ZKP's could potentially form the basis for a bridge.

Building our trustless wrapped Bitcoin would enable further exploration into bridging other assets to Ethereum. XCLAIM's model of proactive security (validity, rather than fraud proofs) is an unexplored foundation for cross-chain interop.

Project Plan

Components

- Smart contracts
    - Chain oracle
    - CCDP
    - WBTC token
- Chain oracle keeper
    - Bitcoin light client, Ethereum light client
- Bitcoin keeper
    - Bitcoin light client, Ethereum light client
- Dapp
    - Send BTC to this address
    - Sign message, send to CCDP
    - Perform transfers where necessary

Smart contracts

- CCDP
    - Create
    - Deposit
    - Issue
    - Redeem
    - Liquidate
- Uniswap exchange
- Chain oracle - https://github.com/crossclaim/btcrelay-sol and stateless SPV's

Keeper node software

- Bitcoin light client (bitcoind thin mode)
- Ethereum light client (Parity)
- Bitcoin integration
    - Monitor Bitcoin addresses for transactions
    - Generate Bitcoin accounts and proofs of their ownership
    - Build and send Bitcoin transactions
    - Track the longest chain, send POW updates to chain oracle
- Ethereum
    - Track the chain oracle for incorrect proofs
    - Track and liquidate risky CDP's

Dapp

The dapp is the most unspecified, since no equivalent to Metamask exists for Bitcoin, the UX needs to be fleshed out.

1. Generate a Bitcoin address B, and commit to the CDP to sending the coins there for locking
2. Send Bitcoins to B
3. Await the chain oracle to acknowledge this block (6 confirms)
4. Now call the contract, creating the CDP with this proof-of-deposit and attaching ETH collateral to the appropriate amount

Timeline

We already have experience in delivering this type of software architecture.

In the past three months between January and April, we have developed a suite of software for bridging events between all public Ethereum networks (our monorepo).

We implemented Merkle proofs, developed relayer software, ran test blockchains in the background. We also developed a cross-chain wallet dapp.

Liam was also present at EthCC in March, where he got to get acquainted with one of the academics behind XCLAIM.

Based on this experience, we estimate this project would require 2-3 months of work **implemented from scratch**. However, there are portions we are able to reuse/copy, such as the chain oracle, Maker CDP logic and frontend code, and our existing relayer setup.

Pseudocode

```
- we lock one BTC on bitcoin chain
- on ethereum, we submit the chain relay proof (statically from the
last block)
- we mint the first wrapped BTC (custom function) with a trusted
price level
   verifies x btc has been locked
   escrows our ether deposit
   mints x ybtc
   creates the uniswap exchange
```

```
   puts in this liquidity
- now whenever someone wants to buy btc, they can do it through
uniswap
- the usual mint looks like:
   verify x locked btc
   btc_price = exchange.get_btc_price()
   collateralisation_ratio = 150
   eth_collateral = msg.value
   btc_value_eth = locked_btc / btc_price
   require(
       btc_value_eth / eth_collateral <= collateralisation_ratio,
       "UNDER_COLLATERALISED"
   )
   send mint(x)

   cdp = {
       btc_value=blah,
       eth=blah
   }
- deposit
   cdp.eth += value
- withdraw
   cdp.eth -= value
- burn
   wbtc -= amt
- redeem
   cdp.burn(wbtc)
   escrowed_collateral = collateral_for_wbtc
   cdp.collateral -= escrowed_collateral

   cdp[x].lockUpForClaim(amt)
       transfers.append(
           amt,
           escrowed_collateral,
           to_addr,
           blocktime
       )
- redeem_success
   require(
       chainoracle.prove(transfer)
   )
   cdp.collateral += escrowed_collateral
- redeem_fail
```

```
    transfer = transfers[x]
    if transfer.passed:
        return false
    if (oracle.blocknumber - transfer.blocknumber) >
MAX_TIME_PROCESS_CLAIM
        transfer.escrowed_collateral
- liquidate
    if prove(btc_value) < cdp.btc_value:
    if btc_value_eth / eth_collateral <= collateralisation_ratio:
        btc_price = exchange.get_btc_price()
        eth_to_sell = collateral / liq_penalty / stability_fee
        sell eth for btc
            buyback btc and burn
```

Lightning payment channels

A lightning payment channel could be used for faster deposits.

The lightning network is a L2 scaling solution for Bitcoin transactions. It consists of a spanning graph network of bidirectional payment channels, which are constructed in a way that payments can be routed across the network without on-chain tx's for updating state.

The transactions are built using the constructs of a HLTC (Hash-Locked Time Contract). Using the HLTC, commitments and sequence numbers (Revocable Sequence Maturity Contract (RSMC)), they implement a basic state machine for transacting.

What happens if a Lightning channel defaults on a tx?
We can create Lightning transactions that are invalid on mainnet (dust https://bitcoin.stackexchange.com/questions/46730/can-you-send-amounts-that-would-typically-be-considered-dust-through-the-light?rq=1)

Channel lifetime:

- open channel
- fund channel
- Update state (ie. transact)
- close channel

How do we pay between two parties that don’t have a channel? We construct a route. Routes are very simple - they are multi-party Lightning invoices.

The basic example - 3 hop payment.

Alice, Bob, Terry

Alice wants to pay Terry but has no channel with him.
Alice creates these invoices:

- Bob -> Terry (1 LBTC)
- Alice -> Bob (1 LBTC)

Alice signs the payment to Bob, which requires Bob’s signature to claim for paying Terry. The payment is thus routed with each transacting party’s signature. Each node updates their individual channel state with the new balances between the parties.