Swingby protocol: a cross-chain decentralized P2P lending protocol for issuance of a BTC stablecoin

Swingby Protocol
https://swingby.network/

14 Dec. 2018
WORKING DRAFT

Table of Contents

1. Introduction

1.1 The cryptocurrency market

As of December 1st, 2018, the market capitalization of the virtual currency as a whole is about 135 billion dollars and the trading volume of 24 hours is about 16 billion dollars. Among them, Bitcoin, which is the most traded, has market capitalization of approximately 73 billion dollars and trading volume of approximately 5.8 billion dollars, accounting for over 54% of the market based on market capitalization. Ethereum has a market capitalization of 12 billion dollars and a transaction volume of about 2 billion dollars, having a gap of six times the capitalization of Ethereum compared to Bitcoin. One of the important indicators for deciding the value of a blockchain is the concept of liquidity. Currencies traded more often are considered more valuable because they are more in demand.

1.2 Why we need to use Bitcoin on the Ethereum Ecosystem

In recent years with an increase in applications that utilize public blockchains and *smart contract* platforms, the Ethereum network has seen an increased focus on solutions to scalability problems. However, even if we solve the scalability problem, it is separate from the fundamental liquidity of Ethereum itself. Even with compelling decentralized applications (Dapps) and exchanges (DEX), these applications need liquidity to work best. That is why we want to focus on improving the liquidity of Ethereum.

For this we looked to Bitcoin. Different from Ethereum, a world computer for decentralized applications, Bitcoin's value is a digital asset representing value itself. Bitcoin has an overwhelming number of holders, total asset value and above all liquidity.

If a token with the same value as Bitcoin could exist on the Ethereum network, a Bitcoin holder could use Dapps, DEXes and other services without changing their main store of value to the Ethereum native currency, but utilising a token that is in practise not different from regular Bitcoin.

As a result, by realizing a "Bitcoin Token" on the Ethereum network, the following advantages are created:

- Bring over Bitcoin's liquidity to other blockchains, starting from Ethereum
- Holders of Bitcoin will be able to use Ethereum's advanced features and services in less steps
- Bitcoin's scalability is improved as well by offloading some Bitcoin transactions

We can also see a wide variety of use cases for a bitcoin token on Ethereum including but not limited to:

- **Dex:** DEX provides trading of BTC token on the Ethereum network (e.g. 0x protocol). This leads to liquidity improvement at the exchange.
- **Derivatives:** Financial derivatives protocols (e.g. dYdX protocol) with short sell and options functions can use BTC token.
- **Lending:** BTC token are the subject of lending transactions on decentralized lending protocols (e.g. Dharma).
- **Payment:** Customers could buy items and/or services by cryptocurrencies just like BTC at ecommerce sites. Ethereum wallets could accept these tokens through Dapps.

In this paper, we propose a method to create an ERC-20 token representing BTC which shall have its value pegged with Bitcoin in a decentralized manner using a decentralized lending model.

2. Technical background and related work

First we will discuss related work that has already been done which is somewhat related to our goal of creating a bitcoin token.

In the past, various approaches have been proposed to realize 2-way peg between Bitcoin and other chains. However, there are many problems with all 2WP approaches, and currently there is no one running a completely decentralized model.

Below we will go over the main approaches to realising a 2-way peg. The examples range from methods to peg multiple and different blockchains, to relay methods and finally a collateral backed stablecoin method.

Later in chapter 3 we will explain the technical model for the Swingby protocol.

A. Trusted custodian

Many blockchains, including Bitcoin, support the required features to be able to create a multi-signature wallet ("multisig wallet"). Bridging a token is possible by using a *multisig* wallet controlled by a custodian or federation on two blockchains.

There is a Kyber Network project called WBTC[1] that uses this technique, and we are trying to realize Bitcoin's ERC-20 token on Ethereum.

However with a "trusted custodian" model, the end users need to trust the custodian or federation.

B. Drivechain

The project called Rootstock (RSK)[2] wanted to provide the Bitcoin blockchain with additional *smart contract* functionality and therefore created a *sidechain*[3] to the Bitcoin blockchain with their own implementation of *smart contracts*. In order for this to work they require a 2-way peg method between the *sidechain* and the Bitcoin blockchain, which they proposed with a concept called Drivechain.

Drivechain[4] is a method that uses *merge mining* for Bitcoin and relies on SPV certification. With this method it is possible to prove the movement of tokens between two blockchains in a very efficient way.
However,

- In order to realize *merge mining*, the mining process of *sidechain* is required to have the same security equivalent to the main chain
- There is a need to trust the *merkle root* included in a *sidechain*'s block

- If both chains are branched on one side, it is extremely difficult to maintain consistency between both chains
- In order to realize Drivechain, a Bitcoin *soft fork* is required in the future

C. Cosmos Zone and Peggy

Cosmos[5] is able to bridge tokens of several blockchains with 'Tendermint', the breakthrough consensus engine. However, Cosmos can prove the transaction only if the state of each blockchain is already "finalized".

The Peggy[6], implemented by the Cosmos team as the evolution of the Cosmos Zone concept, provides a pseudo *finality* on the Ethereum network. Peggy runs on the EVM, so it can unfortunately not not provide *finality* on networks not compatible with EVM.

D. Polkadot and Parachains

Polkadot[7] also supports the functionality of bridging tokens, but in order to connect multiple blockchains, it has properties similar to Cosmos in order to construct a *parachain*. Whereas Cosmos tries to implement currency remittance, Polkadot focuses on parallelizing EVM processing. As with Cosmos, there is a problem that *finality* can not be secured in networks that are not compatible with EVM.

E. BTC Relay and Relay Network, Dogethereum

The BTC Relay[8] model uses SPV proofs to verify transactions from the Bitcoin network directly on the EVM.

The Relay Network is an implementation of BTC Relay that aims to minimize the processing costs as much as possible by offloading as much as possible off-chain. However, because of this the *relayer* will need to trust the merkle root that is provided, which means that you must maintain consensus among the nodes.

Dogethereum[9] realizes a 2-way peg which generates an ERC-20 token for Dogecoin on the Ethereum network. Dogethereum's 2-way peg is aiming for a decentralized storage solution for dogecoin. However, currently the Dogecoin is stored in a *multisig* wallet and thus full decentralization is not yet achieved.

F. Collateral backed stablecoin (DAI)

DAI[10] utilises a method where the value of the currency to be pegged is collateralized by another token as a collateralized bond. When the value of the token that collaterizes the currency falls or rises, there will be instability leading to volatility. Therefore this is often referred to as a soft peg (not a perfect peg).

Furthermore it is difficult to implement a fully decentralized oracle and currently there is no complete decentralized system in place. However, in scenarios of reduced volatility between the currency and collateral, it has proven to be a useful technique for stabilizing the value of the token that you want to peg.

3. Overview of the Swingby protocol

3.1 Motivation

The goal of the Swingby protocol is to **create a usable ERC-20 token that represents BTC on the Ethereum network**. It is a protocol focusing on Bitcoin and Ethereum rather than being a generic pegging method. In this paper we will refer to the ERC-20 token as Bitcoin Token or BTCT.

The most important feature of the Swingby protocol is **to decentralize the custodian**. In the Swingby protocol the custodian refers to the actor who backs minted BTCT with collateral and is the custodian of this collateral.

In the following sections we will lay out the technical model of Swingby and what its approach is to decentralize the custodian.

3.2 Overview

Swingby is protocol that issues Bitcoin tokens on Ethereum using decentralized custodians. Swingby utilises a cross-chain decentralized lending mechanism and creates a stablecoin to represent Bitcoin.

The decentralized lending mechanism keeps custody over bitcoin and lends it as ERC-20 tokens. This results in a soft peg between the Bitcoin and Ethereum blockchains, and **makes sure that the custodian is decentralized**.

3.3 Lending mechanism (minting BTCT)

The Swingby protocol is a decentralised mechanism where users can either earn profit by lending BTC, or can come to mint BTCT. Tokens are only minted if they are fully collateralised by locking BTC and also providing ETH as collateral.

There are two main actors when minting new BTCT with the Swingby protocol: *a lender* (locking BTC) and *a borrower* (locking ETH). The Swingby protocol keeps track of the total collateral and provides the ability to trade one BTCT for one BTC at any point in time.

In the following diagram the process to mint BTCT is shown.

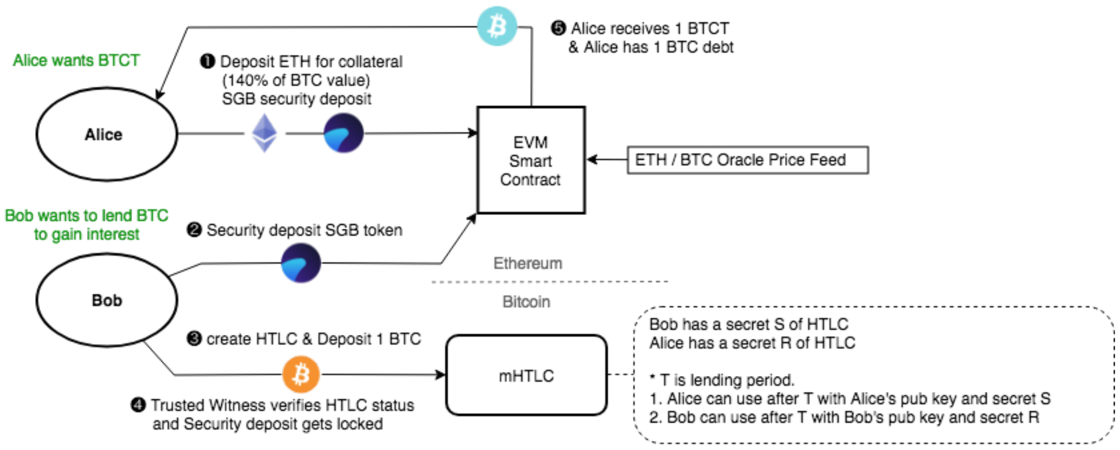
Alice wants BTCT
Alice
❶ Deposit ETH for collateral (140% of BTC value) SGB security deposit
❺ Alice receives 1 BTCT & Alice has 1 BTC debt
EVM Smart Contract
ETH / BTC Oracle Price Feed
Bob wants to lend BTC to gain interest
Bob
❷ Security deposit SGB token
Ethereum
Bitcoin
❸ create HTLC & Deposit 1 BTC
mHTLC
❹ Trusted Witness verifies HTLC status and Security deposit gets locked
Bob has a secret S of HTLC
Alice has a secret R of HTLC
* T is lending period.
1. Alice can use after T with Alice's pub key and secret S
2. Bob can use after T with Bob's pub key and secret R

Fig. 1. Diagram of lending mechanism (minting BTC)

We will continue to explain the related actors and reasoning behind the steps in this process.

Actors

Actors visible on diagram

- *The borrower* (Alice): aims to create BTCT
- *The lender* (Bob): aims to earn profit from the protocol by lending BTC to the protocol
- *EVM smart contract*: integral part of the Swingby protocol on the Ethereum network
- *mHTLC*: a multi secret hashed timelock contract which is an integral part of the Swingby protocol on the Bitcoin network
- *Trusted Oracle*: an oracle for a price feed for the value of ETH/BTC
- *Trusted Witness*: verifies the mHTLC transaction which is submitted
- *SGB token*: the Swingby protocol governance token

Steps

In the diagram above we see a trustless escrow between Alice and Bob. The steps taken are as followed:

1. Alice orders 1 BTCT and locks ETH collateral and a security deposit of SGB tokens.
    a. Alice generate Keypair $<Pub_{alice}, Priv_{alice}>$
    b. Alice generate random secret and Hash $< sR, rHash>$
    c. Alice publish Pub_{alice} to Bob.
    d. Alice creates a deposit transaction as collateral to order 1 BTCT

2. Bob locks a security deposit of SGB tokens.
3. Bob creates an mHTLC and locks 1 BTC, he submits this to the EVM *smart contract*
    a. Bob creates random secret and Hash $<sS, sHash>$
    b. Bob creates an mHTLC for Escrow BTC using $< rHash, sHash, Pub_{alice} >$
    c. Bob's BTC is locked in the mHTLC by Alice's secret $<sR>$
    d. Bob submits the transaction hash of the mHTLC to the Swingby contract
4. A trusted witness verifies the submitted mHTLC.
5. Alice receives 1 BTCT.
    a. Alice's creates a transaction to mint the BTCT which will be assigned to her.

Bob's locked BTC is passed to Alice in the form of "debt". This means that Alice has a debt of 1 BTC which is represented as 1 BTCT. Since this debt is also collateralized by a different currency (ETH), Alice has effectively created a collateralized debt position (CDP); a concept also used in the stablecoin DAI by MakerDAO. One big difference is that with the Swingby protocol it's possible for Alice to clear her position and receive real BTC by burning the BTCT. This is explained in the settlement mechanism chapter below.

The lending mechanism is used by Swingby is to be able to continuously lock BTC. Because of its cross chain nature this is built both on the Ethereum and Bitcoin blockchains. It represents an agreement between two parties and enforces this without requiring trust.

The collateral and security deposits guarantee normal operation of the mechanism. The roles of each currency are shown below.

- **ETH**: used as a collateral to preserve the value of generated BTCT
- **SGB**: the Swingby protocol governance token used as security deposit for Alice and Bob

3.4 Settlement mechanism (burning BTCT)

We have shown with the Swingby lending mechanism how debt can be created to be able to mint BTCT. It is also possible to settle this debt at a later point in time and burn the BTCT in the process. This is called the settlement mechanism.

With the Swingby protocol when BTCT is burned, the borrower (Alice) will be able to receive actual BTC instead of getting her ETH collateral back. At this time, the current value of BTC in ETH is paid to the lender (Bob) in ETH, along with interest. Whatever is left is returned to Alice.

In the following diagram the process to burn BTCT is shown.

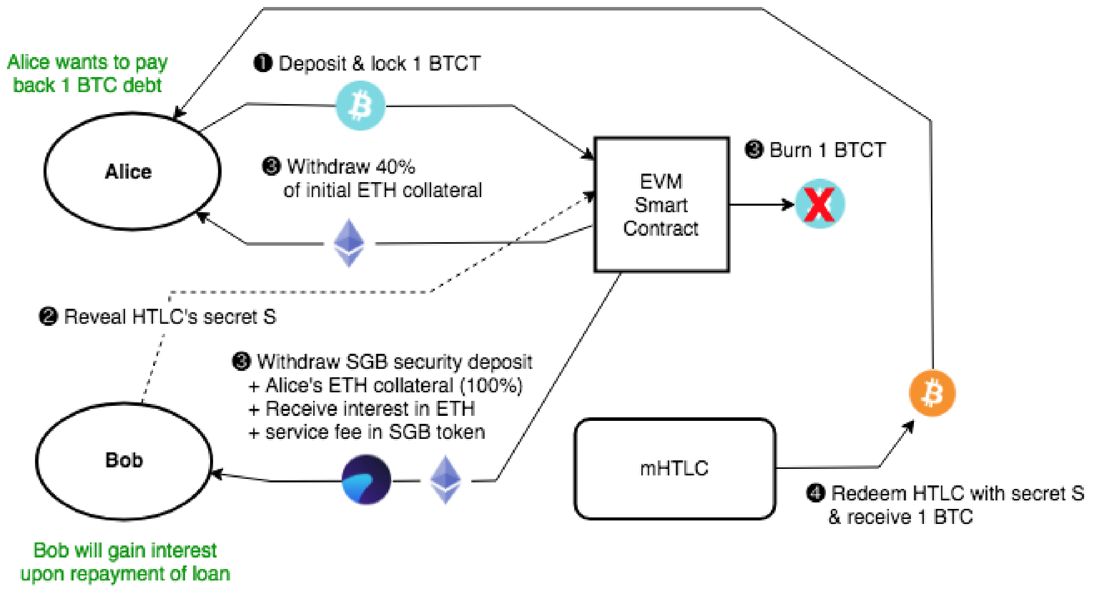
Alice wants to pay back 1 BTC debt
❶ Deposit & lock 1 BTCT
Alice
❸ Withdraw 40% of initial ETH collateral
EVM Smart Contract
❸ Burn 1 BTCT
❷ Reveal HTLC's secret S
❸ Withdraw SGB security deposit + Alice's ETH collateral (100%) + Receive interest in ETH + service fee in SGB token
Bob
mHTLC
❹ Redeem HTLC with secret S & receive 1 BTC
Bob will gain interest upon repayment of loan

Fig. 2. Diagram of settlement mechanism (burning BTCT)

Actors

Actors visible on diagram

- *The borrower* (Alice): aims to settle her debt
- *The lender* (Bob): aims to earn profit from the protocol when Alice settles her debt
- Other actors similar to Fig. 1

Steps

1. Alice sends a burn request to the contract. Hereby her 1 BTCT will be locked as a deposit.
2. Bob confirms Alice's request and makes his secret of the mHTLC public.
3. The current value of BTC in ETH is paid to Bob, along with interest and a service fee in SGB.
4. Alice is able to use the BTC locked inside the mHTLC with Bob's secret.

In the above step 3 it is required to have *finality* for the Bitcoin and Ethereum blockchains. The necessity of *finality* will continue to be discussed at "H. Blockchain finality consideration" below.

4. Swingby protocol features

We have given an overview of how the Swingby protocol works. Now we will go deeper into each feature and actor and its role in the protocol.

4.1 mHTLC (Multi-secret hashed timelock contract)

The Swingby protocol uses a *multi-secret hashed timelock contract* (hereinafter “mHTLC”). This is a variation on a traditional hashed timelock contract (HTLC)[11] often used for *atomic swaps* between two blockchains.

The mHTLC is required to be able to lock BTC as an escrow in a trustless matter. Because an mHTLC works as a trustless escrow, the Swingby protocol has less security and legal risks compared to trusted custodians (of course, depending on the implementation of the trusted custodian). In a trusted custodian model, as the total volume handled rises, so will its cost of credit, and therefore can also lead to scalability issues.

The Swingby protocol can have many lenders, creating a situation where the collateral for all minted BTCT is held in custody in a decentralized way.

Timelock

Unlike an *atomic swap*, the Swingby protocol locks one currency and issues it as a token instead of swapping two currencies on the spot. In the settlement phase, it will burn the token and complete the swap. Therefore, an mHTLC requires to use a timelock.

When the mHTLC is created a "due date for repayment" must be specified. This repayment date is determined under agreement of both parties.

Redeeming BTC

mHTLC uses a redeem script that makes use of the secrets of both parties. Therefore it’s possible for the locked BTC to be redeemed when the secret of the lender is made public and without requiring a signed transaction. Since the mHTLC is the escrow mechanism on the Bitcoin blockchain, using a secret instead of the lender’s private key makes it possible to verify this on the Ethereum side. Also, since the lender can just submit this to the Swingby *smart contract* on

Ethereum it requires no action on the Bitcoin blockchain. This means the lender can also work with a cold wallet to improve the security.

When a borrower settles their debt, the BTCT can easily be swapped for real BTC. This is the part that is similar to an atomic swap and is possible because of the mHTLC.

Script

Below are the OPcodes used for Swingby's mHTLC:

Multi-secret HTLC redeem script

```
OP_IF
   <nLockTime>
   OP_CHECKLOCKTIMEVERIFY
   OP_DROP
   OP_SHA256
   <secretSHash>
   OP_EQUALVERIFY
   OP_DUP
   OP_HASH160
   <receiverPubkeyHash>
OP_ELSE
   OP_SHA256
   secretRHash
   OP_EQUALVERIFY
   OP_DUP
   OP_HASH160
   <senderPubkeyHash>
OP_ENDIF
OP_EQUALVERIFY
OP_CHECKSIG
```

4.2 Trusted witness (mHTLC verification)

After a lender has generated an mHTLC it needs to be submitted it to the Swingby *smart contract* on the Ethereum network. The lender does this by submitting the transaction hash of the mHTLC. However, since the Ethereum *smart contract* can not directly verify the existence and correctness of this transaction hash on the Bitcoin blockchain, this is done by a *trusted witness*.

The role of *trusted witness* is to verify the amount of BTC locked inside the mHTLC and notify the Swingby *smart contract* of its existence and correctness.

A *trusted witness* is elected through voting with SGB tokens. A *trusted witness*' credit will depend on the community. A *trusted witness* can also be replaced through a vote of non-confidence.

4.3 Trusted oracle (acquisition of price rates)

While BTCT is circulating, the value of the collateral in ETH, deposited to the *smart contract*, must at all times be greater than the value of the BTCT issued. To make sure of this, the total value of the ETH collateral in comparison to BTC must always be monitored.

The role of submitting the exchange rate data to the Swingby *smart contract* is called a *trusted oracle*. The ways of collecting data are:

1. *Centralized exchanges*: get the BTC/ETH exchange rate from centralized exchanges
2. *Decentralized exchanges*: get the BTC/ETH exchange rate from decentralized exchanges that support atomic swaps

An average of all the rates of several sources with the above methods is calculated and submitted to Swingby. Then the protocol can calculate the required ETH collateral for lending contracts.

Since the data from the *trusted oracle* is not completely *trustless*, SGB tokens will be utilised to vote on the parties that can become a *trusted oracle*.

4.5 Swingby token (SGB)

The SGB token is the Swingby protocol governance and utility token. It has several uses defined as follows:

1. **service fee** of the borrower to the lender (fixed percentage)
2. **security deposit** of the borrower and lender (fixed percentage)
   To be able to slash in case a party is dishonest.
3. **project governance** of the *trusted oracle* and *trusted witness*
   It is used for the decision making process. A community is necessary to decentralize the roles of *trusted oracles* and *trusted witnesses*. These actors are elected by the community's token vote.
4. **emergency SGB issuance** as countermeasure against insufficient collateral
   It is a defense mechanism in case there is a sudden big change in the exchange rate of ETH and BTC. More info at 'F. Emergency SGB faucet' down below.

4.6 Forced liquidation & Keepers

BTCT is always secured by both BTC and ETH. The collateral rate of ETH is set at 150% or more of the value of BTC. In order to withstand changes in the exchange rate, at least 135% ETH collateral is required. This is called the **liquidation limit**.

Due to volatility of the exchange rate of BTC/ETH, when the collateral rate of ETH approaches 135%, there is danger that the ETH collateral that is circulating will not be enough. This makes it necessary for the borrowers to repay the BTCT as soon as possible and clearing their debt by settling their lending contracts.

Keepers

A keeper's role is to monitor the network and check whether there is sufficient collateral for each issued BTCT. If the collateral rate drops below the *liquidation limit* due to changes in the exchange rate, it will create a situation where there is insufficient collateral for the issued BTCT. To prevent this situation and maintain the integrity of the protocol, all lending contracts that are under-collateralized need to be settled by *keepers*. This event is called **forced liquidation**.

In the event of *forced liquidation*, in order to hasten the repayment of the debt for each lending contract that has issued BTCT, a *keeper* will make settlement the lending contract instead of the borrower. This is done by burning the BTCT instead of the borrower. The keeper is however not able to receive BTC for the BTCT that is burned, because he doesn’t know the borrower’s secret. Therefore the *keeper* is repaid in ETH for the BTCT that he burns. He will also receive an extra fee from the contract for his services.

A keeper's strategy is to buy BTCT from the market and keep it at hand to burn at any time.

Forced Liquidation event overview

Following is a step by step overview of the forced liquidation event:

(percentages may be changed in the future as the protocol is being developed)

1. A borrower makes a lending contract and mints 1 BTCT with a collateral rate of 150% ETH
2. Because of a change in the exchange rate of BTC/ETH, the ETH collateral rate drops to 133%
3. **A *keeper* invokes a *forced liquidation***
4. Instead of the borrower, the *keeper* settles the debt by burning BTCT
5. For burning the BTCT, the *keeper* receives the ETH collateral (100% out of 133%)

6. A *liquidation penalty fee* is subtracted from the remaining ETH collateral (13% out of 33% remaining)
    a. The *keeper* receives 5% out of 13% as an incentive
    b. The remaining 8% will be remitted to a shared reserve of the Swingby protocol
7. Interest is paid to the lender (eg. 3% of the remaining 20%)
8. The rest is returned to the borrower (remaining 17% in this example)
   However, in order for the lender to be able to retrieve his locked BTC, he will need the mHTLC secret of the borrower. Therefore, the borrower can only withdraw the remaining ETH after submitting his secret (sR) of the mHTLC.

Because of *forced liquidation* events, the BTCT supply and demand volume will stabilize and affect unwanted BTCT price fluctuations.

The concept of a *keeper* is based on MakerDAO's single collateral DAI design. We believe that
assuming enough liquidity, with a *keeper* model even in the situation of a rapid decline in the collateral value, it's possible to steadily execute *forced liquidations*.

Lender's interest protection

As you can see in the event overview above, a lender gains interest as usual during a *forced liquidation*, but this is interest paid in ETH, which is undesirable for situations where ETH is dropping in value. Therefore, with the Swingby protocol a lender can acquire additional interest during *forced liquidation* events.

Additional interest for lenders is paid from the Swingby reserve. The Swingby protocol will pay out additional interest depending on the stake of a lender's SGB token. This SGB stake is based on the lender's security deposit of SGB tokens when generating a Swingby lending contract. In addition, the lender can add a SGB tokens to this security deposit at any time.

With this measure, generation of BTCT will raise the demand for SGB because as long as BTCT is minted SGB will continue to be locked inside Swingby *smart contracts* as security deposit. Therefore, the liquidity of SGB is expected to improve with the demand of BTCT. By continuing to stake SGB tokens as security deposit, lender can continue to earn rewards and contribute to the system.

4.7 Emergency SGB issuance

When there is a situation of a rapid change in the exchange rate between BTC/ETH (such as a “black swan event”), there is a possibility that the collateral rate of ETH will fall below 100%. In such an event even if *forced liquidation* is triggered, this might not happen fast enough to stabilize the system. In an emergency time like this, it is necessary to urgently procure additional ETH collateral.

In this event the Swingby protocol mints new SGB tokens and adopts a method to procure additional ETH to use as collateral by auctioning off these newly minted SGB tokens. This is done with a dutch auction. Because of the nature of this emergency countermeasure, it can only be triggered through a token voting.

4.8 BTCT re-minting incentive

When minting BTCT through Swingby’s lending contract it will have a settlement date. If a borrower wants to continue holding BTCT he can mint new BTCT on the settlement date. When minting new BTCT the old position is settled once so newly minted BTCT will have a lower risk of being *force liquidated.*

However, every time a lending contract is settled and a new one opened, the borrower will need to pay an interest fee to the lender. In order to avoid this the borrower can become the lender as well and provide both the BTC and ETH collateral himself. In this case there is effectively no more interest to be paid each time a lending contract is settled and reopened, because it is from and to the same person. Also with each extended timelock when a new lending contract is reopened the *force liquidation* risk is reduced.

4.9 Blockchain finality consideration

Blockchain *finality* is one of the most important themes in blockchain. Because a blockchain is an asynchronous system, in order to connect two have a cross-chain protocol over two blockchains that have different *finality* requirements, it will be necessary to carefully think about any differences between these conditions.

Cosmos for example uses a model called CasperFFG[12] to provide a sort of pseudo *finality* to individual blockchains connected through a peg-zone. However, securing *finality* in the peg-zone can not be done with the consensus algorithm alone. There is always the possibility of a *hard fork* occuring in either of the connected blockchains, and since the *finality* provided by this algorithm is a pseudo *finality*, it is difficult to safely use and/or reorganise peg-token that already exist inside the peg-zone at the time of hard fork.

In the case of the Swingby protocol architecture, the required *finality* of each transaction is based purely on the opinion of the involved parties (the borrower and lender) and is completed based on an agreement between them. Even if a problem arises due to a difference in *finality* (because of eg. a hardfork), *finality* is judged in agreement by both parties to be obtained at the time of creating a lending contract. This makes it possible for the Swingby protocol to temporarily and safely settle all existing lending contracts.

5. Conclusion

In this paper we have looked at several methods of pegging two blockchains. We have further proposed the Swingby protocol which creates an ERC-20 token representing Bitcoin on the Ethereum network and is soft-pegged to the value of BTC. For purposes of explaining the model this token was called BTCT or Bitcoin token.

Swingby's basic design philosophy is to make use of a cross-chain decentralized P2P lending mechanism to continually lock BTC in escrow between two parties on a trustless base. This BTC is then used on the Ethereum network as collateral for BTCT.

Our main focus was to do this **while keeping the custodian decentralized** as opposed to other solutions. To strengthen this goal, the position of a lender, and thus effectively custodian, can be fulfilled by anyone who wants to participate to the Swingby protocol. It is a role that is incentivised and thus creates a situation with optimal custodian decentralization.

6. References

[1] Wrapped Tokens - A multi-institutional framework for tokenizing any asset
https://www.wbtc.network/assets/wrapped-tokens-whitepaper.pdf

[2] RootStock - Rootstock White Paper
https://docs.rsk.co/RSK_White_Paper-Overview.pdf

[3] Sidechain - Enabling Blockchain Innovations with Pegged Sidechains
https://blockstream.com/sidechains.pdf

[4] Drivechain - The Simple Two Way Peg
http://www.truthcoin.info/blog/drivechain/

[5] Cosmos - A Network of Distributed Ledgers
https://cosmos.network/cosmos-whitepaper.pdf

[6] Peggy - Peggy is a peg zone implementation
https://github.com/cosmos/peggy

[7] Polkadot: Vision for a Heterogeneous Multi-Chain Framework
https://polkadot.network/PolkaDotPaper.pdf

[8] BTC Relay - Ethereum contract for Bitcoin SPV
https://github.com/ethereum/btcrelay

[9] Dogethereum-contracts [DogeRelay]
https://github.com/dogethereum/dogethereum-contracts

[10] DAI - Stablecoin System
https://makerdao.com/whitepaper/DaiDec17WP.pdf

[11] Hashed Timelock Contract (HTLC) in Lightning-network-paper
https://lightning.network/lightning-network-paper.pdf

[12] Casper the Friendly Finality Gadget
https://arxiv.org/pdf/1710.09437.pdf