Atomic Bonded Cross-chain Debt

Mojtaba Tefagh
mtefagh@sharif.edu

Fatemeh Bagheri*
fateme.bagheri95@student.sharif.edu

Amirhossein Khajehpour*
amirhosseinkh@ce.sharif.edu

Melika Abdi*
melika.abdi@ee.sharif.edu

October 2020

Abstract

Inspired by the recent boom in *decentralized finance* (DeFi) and the unprecedented success of flash loan projects in this ecosystem, we introduce a decentralized debt derivative named *atomic bonded crosschain debt* (ABCD) to bridge the gap between the growth of lending protocols on Ethereum and other UTXO-based blockchains specifically Bitcoin. We think of ABCD as the alphabet of interoperability for DeFi and as a credit infrastructure which unlike the current protocols is not limited by requiring either smart contracts, over-collateralization, or instantaneous payback.

1 Introduction

Many financial instruments have been established and implemented in traditional fiat-based markets; among them: options, futures, loans, bonds, derivatives, etc. In the past decade, the concept of cryptocurrency has opened a new gate toward the next generation of economy and finance. This field is still open to new ideas and introduces lots of implementation challenges for DeFi.

By the invention of Ethereum smart contracts, so many decentralized financial applications were built which have resulted in the rapid growth of the ether market capital in general, and the total value locked in liquidity and lending protocols specifically [1–5]. This demonstrates the urgent need for the blockchain counterpart of well-known financial instruments, especially loans, options, and *decentralized exchanges* (DEX), and as a result, DeFi primitives are being demanded these days more than ever before [6–9].

In the present study, we tackle this challenge and design primitives for decentralized futures market applications which in addition to Ethereum blockchain, work on first-generation blockchains like Bitcoin which do not support high-level

*These authors contributed equally to this work.

Turing-complete scripting languages. Our bond issuance system and the corresponding procedures only require *hash time locked contracts* (HTLC) as their building block and they do not rely on any oracle or third party interference. To the best of our knowledge ABCD is the first protocol in DeFi which offers an atomic unsecured cross-chain bond service.

As the pioneer in decentralization, the pseudonymous Satoshi Nakamoto has devised a new path towards the peer-to-peer payment systems which are counted as a disruptive innovation today [10]. Ethereum as the next generation of decentralized computing services enables writing smart contracts on an electronic ledger [6]. Later on, by the advent of ever-increasing blockchains, one may need to exchange assets across different networks. Through utilizing atomic swaps, two parties on different blockchains make an atomic contract which transfers asset between them [11]. Up until now, several previous works have extended the usage of atomic swaps in different ways. Herlihy designed a model for analyzing atomic cross-chain swaps and suggested a protocol that not only removes incentives for any set of parties to deviate from the protocol, but also guarantees that no conforming party ends with the underwater outcome and showed that HTLCs are enough to achieved this [12]. Zamyatin et al. presented XCLAIM which is a swap frame work based on the atomic swaps that is faster and considerably cheaper than normal atomic swaps [13]. The idea of atomic cross-chain transactions in Ethereum sidechains was developed in [14]. The conflict caused by the concurrent execution of smart contracts was addressed to make an all-or-nothing atomic cross-chain commitment protocol in [15]. Furthermore, Runchao et al. put a step forth by analyzing the fairness of atomic swaps and showed that the basic atomic swap is considerably more unfair compared to its equivalent contracts in the traditional market. Besides, they proposed two enhanced atomic swap protocols and justified their fairness [16]. Liu proposed an atomic swaption component which works only using low-level scripting tools [17]. Additionally, by utilizing his swaption component, offering fully decentralized futures contracts is no longer impossible [17]. Zie et al. extended the atomic cross-chain swap contracts to a new method that does not need HTLCs and everything is managed by different party's signatures [18].

The rest of this paper is organized as follows. First of all, in section 2 after defining the required terminology and presenting the other preliminaries of our work, we introduce the first model of atomic bonded debt and discuss about the crucial requirements of an atomic bond service. Later in section 3, we redesign our model to build the first practical *atomic bonded cross-chain debt* (ABCD) primitive, and finally by adding additional features to it, we improve its stability across different market behaviours.

2 General Overview of Atomic Bonded Debt

The idea behind ABCD is inspired by flash swaps. In a flash swap, the loan is not paid to the borrower unless she pays it back immediately at the same block as borrowed [8]. In ABCD this is extended to several blocks i.e. the issuer can repurchase the bond in more than one block but it contains a secret and the bondholder's signature is required everywhere the capital is being utilized.

Unsecured bonds or debentures are not backed by some type of collateral. Since this bond is unsecured, the issuer does not have to deposit any margin, unlike the approach used in [17]. Assume Alice is the issuer and Bob is the purchaser of the bond. She needs to take the capital, exploit it in other contracts, and then repurchase the bond from Bob with some interest. She also needs Bob to deposit shortly after their contract has been started.

In this section, we are going to introduce the main challenges for having an atomic bond service. To do so, we designed a general overview of an atomic bond service shown in Fig. 1. In this , for each transaction, signatures, output amounts, and locktimes are specified. Transactions' border colors show the party who broadcasts the transaction. For now, we assume that all amounts are of the same coin.

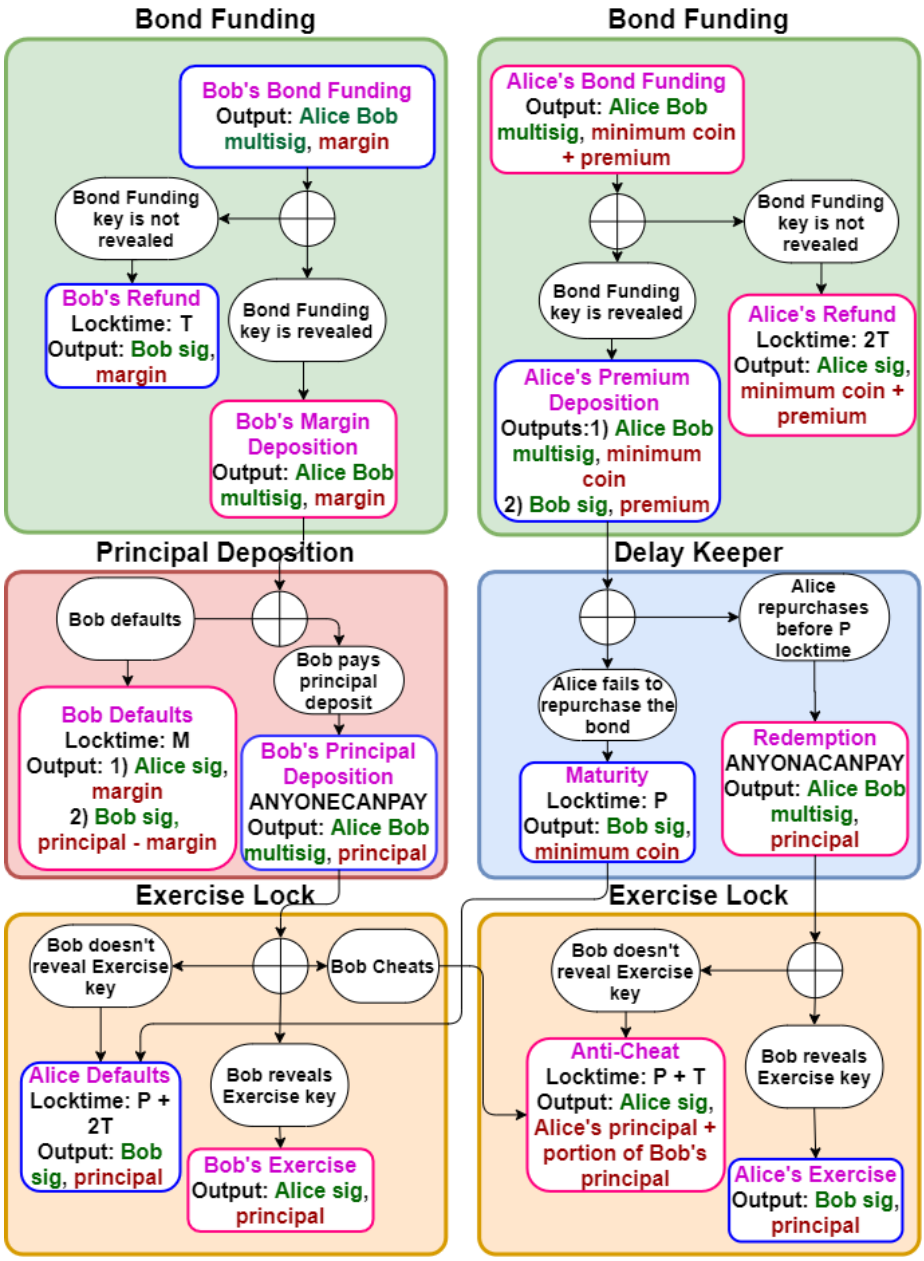

Figure 1: The general overview of an atomic bond service. Each transaction is shown as a rectangle. On each transaction, signatures, output amount, and locktimes are specified. All outputs are in the same coin. Pink-bordered transactions are broadcast by Alice and blue-bordered ones by Bob. For locktimes, Unix timestamp is used. Upper transactions are broadcast earlier than the lower ones. If there is a line between two transactions, then the source transaction is considered to be an input of the destination transaction.

Here, we employ HTLCs to make decisions. If the holder of a secret reveals it before the corresponding timelock deadline, the swap takes place. Otherwise, if she lets the locktime expire, the swap is reverted and all the locked values are given back to their initial owners. The secrets used in this model are the *bond funding* key that the issuer uses to sell and the *exercise* key that the bondholder uses to exercise the bond. In this model, we use the Unix timestamp as the locktime parameter. We represent the minimum time needed for a mined transaction to be confirmed as T. In Bitcoin, it is the time needed to have six subsequent blocks mined which is approximately one hour.

Next, we explain in detail the process of exercising an atomic bond protocol which is basically made up of transactions shown in Fig. 1. First of all, all the transactions are signed and exchanged between the two parties except the bond funding transactions. In this phase, both parties make sure that there is no way the other party cheats on them, since either it is technically impossible or they get punished in case of cheating. This approach is similar to the procedure used by Poon et al. in the lightening payment channels [19]. After that, by broadcasting each of the transactions in the proper time, the process goes on. The procedure is divided into four different stages. Depending on the application the bond is being used for, other stages can be appended to the procedure and transactions might need to be signed by more parties. However, here we explain the basic structure of the bond itself:

- **Bond Funding**: The funding transaction for Alice consists of
  - a premium,
  - and a very tiny amount for further usage (the minimum acceptable amount of output to be mined by the network miners, for example 546 Satoshis in Bitcoin network at the time of writing).

  For Bob, the bond funding transaction only contains the margin. Alice has a relatively small amount of time to reveal the bond funding key to sell the bond. If she issues the bond, the premium goes to Bob and his margin goes to the Bob's margin deposition transaction.

- **Principal Deposition**: After selling the bond, each party has to deposit their principal within a specified time interval: Bob M locktime and Alice P ($P > M$). The *Bob's principal deposition* and the *redemption* transactions have sighash type of anyone-can-pay[1] since nobody knows all of their inputs in the first place. Bob can act either ways of:
  - If Bob defaults, then Alice takes his margin by broadcasting the *Bob defaults* transaction. Additionally, she will not fulfill the redemption transaction. Therefore, Bob can broadcast the *maturity* transaction, taking the minimum amount of coins which is too small to consider.
  - If Bob deposits the principal, the bond goes to the *delay keeper* stage.

[1]The op-code ANYONECANPAY

The *premium deposition* transaction and the minimum amount of coins are needed so that using this transaction we can force a deadline on Alice depositing her payback by utilizing the locktime on the maturity transaction. Also, in the case that Alice repurchases the bond, Bob can not claim that she did not broadcast the redemption transaction.

- **Delay Keeper**: At this stage, Alice has time before a P locktime expires ($P > M$) to fulfill her redemption transaction. If she does not deposit, Bob will broadcast the maturity transaction that prevents Alice from fulfilling her redemption transaction.

- **Exercise Lock**: The bond enters this stage when Bob deposits his principal. There are three possible scenarios in here:

  - In the last stage, Bob deposited his principal but Alice did not and the maturity transaction is broadcast. Now Bob does not reveal exercise key, and using the *Alice defaults* transaction he takes his principal back.
  - Bob has deposited his principal and Alice has fulfilled her redemption transaction but Bob avoids revealing the exercise key. Subsequently, Alice can broadcast the *anti-cheat* transaction which sends her principal and an amount of punishment from Bob's principal to her.
  - Both have deposited their principals. Bob reveals exercise key and they go to the next stage if there is any[2] and if not, each takes their coins and the procedure ends.

  Note that if Bob delays in broadcasting the maturity transaction, Alice may broadcast the redemption and anti-cheat transactions at the very last minute and cheat on Bob.

The previously presented overview of the atomic bond is well analyzed and seems to be practical. However, there are some problems with its implementation not yet addressed. In every blockchain, signing and spending transactions have a different set of rules, e.g. in Bitcoin the child transaction has to sign the hash of its parent transaction besides the redeem script. The Bitcoin network considers the inputs of a transaction when calculating its hash. To be able to create a transaction that spends a transaction's output, we need to calculate the parent transaction's hash correctly. Therefore, all of a transaction's inputs have to be determined, before creating its output spender. Hence, the transactions in the exercise lock sections are impossible to be signed at the beginning of the protocol. In the subsequent pages, we will prepare our primitives to overcome this issue.

[2]In real-world applications there are usually more stages, since the bond is being used along with other contracts, otherwise it is useless giving ACoins and getting the same ACoins. An example of ABCD implementation along with atomic swap is provided in [20].

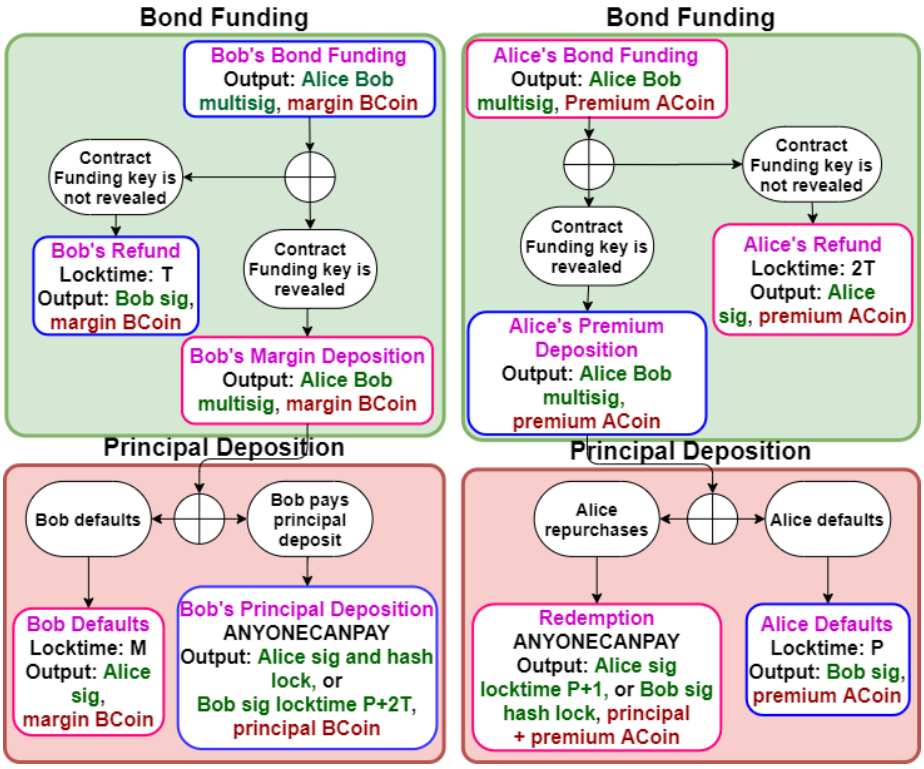

Figure 2: The ABCD component. On each transaction, signatures, output amount and locktimes are specified. All outputs are in the same coin. Pink-bordered transactions are broadcast by Alice and blue-bordered ones by Bob. For locktimes, Unix timestamp is used. Upper transactions are broadcast earlier than the lower ones. If there is a line between two transactions, then the source transaction is considered to be an input of the destination transaction.

3 Atomic Bonded Cross-chain Debt

To make the ABCD primitive, we have to consider two problems. First, the Alice defaults and anti-cheat transactions have inputs from both sides. Second, the redemption and Bob's principal deposition transactions should not have any outputs since their inputs are dynamically determined through the protocol and after creation when we did not have the hashes. To solve either problem, we are going to remove the exercise lock stage from the procedure. The main goal of this stage was to inhibit Bob's cheating. To remove this stage, we need to eliminate the cheating incentive by a different manner. The functionality of the delay keeper stage also needs to be noticed. We have to design the protocol in such a way that when Alice fulfills her redemption, Bob has the minimum needed time to reveal the exercise key.

The first ABCD component is demonstrated in Fig. 2. To present this type of ABCD we use Unix time as the locktime parameter. However, the block height can also be used. The procedure is discussed below:

- **Bond Funding**: Alice's funding includes premium and Bob's includes margin. The number T is the minimum time needed for a transaction to be confirmed. Here, the premium will not be directly sent to Bob after revealing the bond funding key and will be locked in the Alice's premium deposition transaction.

- **Principal Deposition**: After issuing the bond, similar to the general model, each party has to deposit their principal within a specified time interval: Bob M locktime and Alice P ($P > M$). Both Bob's principal deposition and Alice's redemption transactions have sighash type of anyone-can-pay since only on of their inputs is determined at the time of creating the transaction. Bob can behave in one of the two following ways:

  - He defaults, then Alice takes his margin by broadcasting the Bob defaults transaction.
  - He deposits the principal, and waits for action of Alice.

  After Bob's principal deposition, there are two possible scenarios based on decision of Alice:

  - Alice succeeds to repurchase. Bob reveals the exercise key. Finally, Alice can take her bond and the premium will be sent to Bob.
  - Alice does not broadcast the redemption transaction. Therefore, Bob avoids exposing the exercise key and his principal is sent back to himself. To achieve this, Bob gives Alice P locktime to convince him to reveal the exercise key and if this deadline is passed, he will broadcast the Alice defaults transaction which gives him the premium as well.
  - Alice fulfills her redemption transaction but Bob does not reveal the exercise key. In this case, Alice can take back her principal and premium.

  Note that Alice has only P locktime to fulfill her redemption transaction, and to get the redemption's output, she has to wait until $P + T$ locktime. Thus, Bob has the minimum required time to reveal the exercise key and receive his payback. Also, Alice can not deposit the redemption transaction at the very last moments and spend the output of redemption. Hence, the output script of the redemption transaction achieves the purpose of the delay keeper stage in the previous section.

So far, we have made the first ABCD primitive. However, one last important issue is remaining. In this form of ABCD, Bob's only inhibitor from cheating is the amount of premium. Since the cryptocurrencies market faces significant

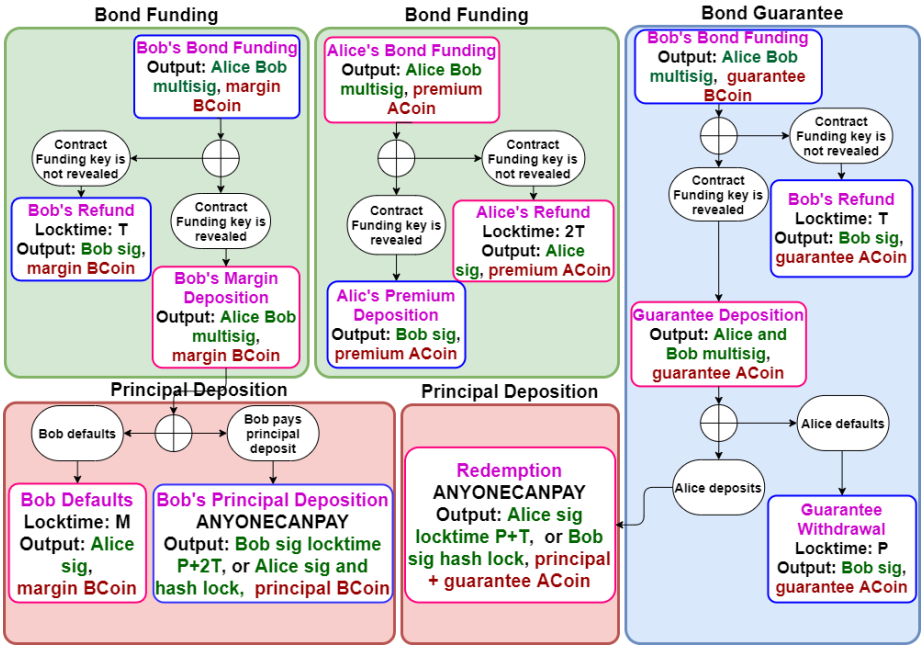

Figure 3: The ABCD across different chains. On each transaction, signatures, output amount and locktimes are specified. Bob's depositions are in BCoin and Alice's in ACoin. Pink-bordered transactions are broadcast by Alice and blue-bordered ones by Bob. For locktimes, Unix timestamp is used. Upper transactions are broadcast earlier than the lower ones. If there is a line between two transaction, then the source transactions is considered to be an input of the destination transaction. To observe an implementation of this structure where the bond is going to be used in an atomic swap, see [20].

fluctuations periodically over time, the value of the BCoin principal may rise higher than the payback plus premium value in ACoin. This rise incentivizes Bob to ignore the premium and get back his principal. To overcome this issue, before exploiting any bond contract, Bob, who is considered to be an exchange or somebody who has a reasonable amount of assets in different blockchains, can deposit some assets as a bond guarantee in any desired chain. Then, we can adjust Bob's guarantee based on the current fluctuation ratio of the market to reduce the probability of Bob's undesired decision. This modification can be seen in Fig.3. In this figure, we use Unix timestamp for locktimes and consider the maximum time among all of the involved blockchains since the number of blocks needed for confirmation is different in different blockchains. We can also use the block height.

The difference between the newly designed ABCD component and the previous ABCD primitive is the addition of the guarantee withdrawal transaction and its related funding transactios. Whether Alice defaults or the bond is successfully repurchased, Bob has to broadcast the *guarantee withdrawal* transaction. Additionally, in the case of not revealing the bond funding key, Bob can take his guarantee back. Also, the premium will be sent directly to Bob in all possible scenarios at the beginning of the protocol. Other parts are the same in both procedures.

4 Conclusion

In this paper, we first introduced the needed requirements of an atomic bond service using the general overview of ABCD. Afterwards, we derived ABCD to achieve the goal of providing an interoperable cross-chain bond. Finally, by extending its design, we empowered the ABCD primitive to resist the market fluctuations. All of the different scenarios of taking part in an ABCD protocol is tested on the Bitcoin testnet. Implementation of ABCD and also a pointer to transactions spent, are available in [20]. Collectively, we have employed the well-known atomic cross-chain swaps for building ABCD as a primitive for uncollateralized DeFi. Potential use cases include but are not limited to exploiting arbitrage opportunities between swaptions without owning any capital or any other similar use case of flash loans and flash swaps with two main improvements:

- Despite the similarities, instead of being a "flash" loan which must get repaid within a block, ABCD can span an arbitrarily long period for the issuer to trade or invest with the capital before the bond reaches maturity. The significance of this feature unfolds by noting that this is not possible even in conventional financial systems to have an unsecured debt without a credit system. More precisely, this is only possible due to the full transparency and traceability of cryptocurrencies.

- Our proposed bond primitive does not require a Turing-complete programming language. The Bitcoin scripting language is sufficient to implement our method, which only relies on HTLC. While most DeFi protocols rely heavily on smart contract custody or third parties that make them susceptible to security issues, ABCD can be flexibly used on the wide variety of HTLC-compatible blockchains and, in particular, supports Bitcoin and its lightning network natively.

5 Future Work

As mentioned earlier, ABCD can be used along with other primitives in order to form more complex contracts. Depending on the application-specific domain, the current structure of ABCD might be either sufficient or not. In a future

work, we aim to customize this structure to be fully compatible for integration in more complex systems.

On the other hand, there exist many variations of HTLC which in turn imply different categories of atomic swaps with different properties. The modular structure of our design enables similar variations on ABCD with the associated properties which can be explored in a future work. In particular, a privacy-preserving scriptless version of ABCD using adaptor signatures may be possible under the framework of Schnorr signatures [21].

References

[1] Harry Thornburg. Aave protocol whitepaper v1, January 2020.

[2] Maker Team. The dai stablecoin system, 2017.

[3] Robert Leshner Geoffrey Hayes. The money market protocol, 2019.

[4] WBTC. Wrapped tokens a multi-institutional framework for tokenizing any asset, 2019.

[5] WBTC. dydx: A standard for decentralized margin trading and derivatives, 2017.

[6] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 3(37), 2014.

[7] Uniswap. Automated liquidity protocol., 2018.

[8] Dan Robinson Hayden Adams, Noah Zinsmeister. Uniswap v2 core, March 2020.

[9] Tito Titov Krasimir Raykov. Jelly swap ecosystem, 2019.

[10] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, Manubot, 2019.

[11] Tier Nolan. Alt chains and atomic transfers. bitcointalk., May 2013.

[12] Maurice Herlihy. Atomic cross-chain swaps. In *Proceedings of the 2018 ACM symposium on principles of distributed computing*, pages 245–254, 2018.

[13] A. Zamyatin, D. Harz, J. Lind, P. Panayiotou, A. Gervais, and W. Knottenbelt. Xclaim: Trustless, interoperable, cryptocurrency-backed assets. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 193–210, 2019.

[14] Peter Robinson, David Hyland-Wood, Roberto Saltini, Sandra Johnson, and John Brainard. Atomic crosschain transactions for ethereum private sidechains. *arXiv preprint arXiv:1904.12079*, 2019.

[15] Victor Zakhary, Divyakant Agrawal, and Amr El Abbadi. Atomic commitment across blockchains. *arXiv preprint arXiv:1905.02847*, 2019.

[16] Runchao Han, Haoyu Lin, and Jiangshan Yu. On the optionality and fairness of atomic swaps. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, AFT '19, page 62–75, New York, NY, USA, 2019. Association for Computing Machinery.

[17] James A Liu. Atomic swaptions: cryptocurrency derivatives. *arXiv preprint arXiv:1807.08644*, 2018.

[18] Jean-Yves Zie, Jean-Christophe Deneuville, Jérémy Briffaut, and Benjamin Nguyen. Extending atomic cross-chain swaps. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 219–229. Springer, 2019.

[19] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016.

[20] ABCD. Reference implementation for abcd, 2020.

[21] Apoorvaa Deshpande and Maurice Herlihy. Privacy-preserving cross-chain atomic swaps. In Matthew Bernhard, Andrea Bracciali, L. Jean Camp, Shin'ichiro Matsuo, Alana Maurushat, Peter B. Rønne, and Massimiliano Sala, editors, *Financial Cryptography and Data Security*, pages 540–549, Cham, 2020. Springer International Publishing.