Capital-free Futures Arbitrage

Mojtaba Tefagh[a], Fatemeh Bagheri[b,*], Amirhossein Khajehpour[b,*], Melika Abdi[c,*]

[a] *Department of Mathematical Sciences, Sharif University of Technology*
[b] *Department of Computer Engineering, Sharif University of Technology*
[c] *Department of Electrical Engineering, Sharif University of Technology*

Abstract

Cryptocurrency futures contracts that can be traded and settled by cash are exposed to the centralized exchange single point of failure. On the other hand, derivatives and synthetic tokens under the custody of smart contracts have suffered extensively from security issues and economic vulnerabilities resulting from oracle price manipulations. The middleman, being a centralized exchange or a decentralized application, can be hacked or exploited as long as the user does not own the private keys to the underlying assets of the futures contract.

To the best of our knowledge, the only non-custodial solution proposed at the time of writing is atomic swaptions and their extensions. However, their efficiency is limited by the fact that the specified amount to be delivered by either party should be locked from the negotiation until the expiry date. We introduce a framework to overcome this limitation without compromising decentralization to enable capital-free atomic swaptions and prove several interesting properties of these financial instruments.

Keywords: decentralized finance, DeFi, HTLC, atomic swap, atomic swaption
2010 MSC: 93A14

*These authors contributed equally to this work
Email addresses: mtefagh@sharif.edu (Mojtaba Tefagh), fateme.bagheri95@student.sharif.edu (Fatemeh Bagheri), amirhosseinkh@ce.sharif.edu (Amirhossein Khajehpour), melika.abdi@ee.sharif.edu (Melika Abdi)

1. Introduction

Time flew since Nakamato in the Bitcoin genesis block [1] mentioned the New York Times article about chancellor bailout [2]. Again, in the third Bitcoin halving, the history has been repeated. The block number 630000 contains the message "NYTimes 09/Apr/2020 With $2.3T Injection, Fed's Plan Far Exceeds 2008 Rescue". To crypto lovers it means the enunciation of decentralization. The path towards self-sovereignty, independence, and removing the governance authorities from critical tasks. Through out these years, decentralized finance (DeFi) attracts attention which unfortunately is not a practical alternative for traditional finance yet. This work, tries to fill this gap and bring the DeFi closer to the market demands.

Hashed time-lock contracts (HTLC) are the chief core of many new developments in the DeFi world. They proved their potential even before DeFi where Poon et. al introduced lightning network [3].

Swaption is first introduced by Liu as the component made of two HTLCs to trade option in a peer-to-peer setting [4]. Later on, Tefagh et al. utilized swaptions to make the first non-custodial capital-free bond under first generation blockchains such as Bitcoin [5]. In this sequel, we further analyse swaptions and futures markets in detail, and make several new useful components using HTLCs and the ideas behind swaptions architecture and put a step forth by theoretically proving their safety in the last section.

Until now, there have been so many trials to re-implement the features of fiat money market with blockchain backbones. Invention of Ethereum, proved the possibility of writing smart contracts [6]. Afterward, thousands of successful Dapp projects were implemented on top of Ethereum aiming the same goals as ours, including MakerDAO, a governance token and executive voting based loan system[7], Compound, a collateralized asset-lending platform [8], dYdX, a trading platform for crypto assets [9], Aave, a depositing and borrowing money market protocol [10] and Ethereum wrapped tokens such as wrapped bitcoin [11], to name a few.

Furthermore, several different blockchains were implemented, such as Cardano, the first peer-reviewed proof of stake blockchain [12], Cosmos, a network of independent parallel blockchains [13], Monero, a private and untraceable currency [14], Zcash, a strongly private currency with low fees [15], Ripple, a peer-to-peer platform for transferring money [16], Stellar, whose consensus protocol depends on a set of selected honest nodes rather than all the nodes [17]. This growing diversity aroused the need for interoperatable protocols. The idea of atomic swap using HTLCs as the first proposed protocol is first suggested by Nolan in bitcointalk [18]. Later, Herily formalized the atomic cross-chain swaps [19]. There are analyzes on these kind of swaps like Xu et al. analyzed the success rate of HTLC-based cross-chain atomic swaps using a game-theoretic approach [20]. Zie et al. extended the approach in a way that it only needs multi-signature support, so it can be implemented in blockchains lacking HTLCs. However, it needs smart contracts on the other chain as previous approaches did [21].

Many works have been trying to either formally prove the security of atomic swaps, or to find an attack to compromise the safety of its participating parties. For example, Meyden used a multi-agent setting to assess functionality of atomic swap [22]. Then, Hirai analysed the atomic swap protocol as an asynchronous communication setting using the Kripke modal logic [23]. Han et al. also tried to provide fairness for atomic swaps, however their proposal needed smart contract [24]. Tsabary et al. devised a new attack based on miners incentive [25]. In an HTLC, one party can incentivise miners to mine the later transaction by suggesting larger fees. They also designed the MAD-HTLC to remove this vulnerability from atomic swap contracts.

Afterward, many high-level protocols implemented atomic swaps. Komodo community has implemented its own unique variation of atomic swaps, named AtomicDEX [26]. Some other swap protocols are implemented like e.g. UniSwap [27], BurgerSwap [28], SushiSwap [29] and KyberSwap [30] which swap ERC20 tokens. Jellyswap, besides offering swaps on different chains, provides liquidity for its protocol by designing Butler [31]. This way, every market maker gets profit with no restriction. However, these food-themed swaps are mostly based

on *automated market maker* (AMM) and yield farming not on atomic swaps. In AMM, liquidity providers get shares in fees or amount of governance tokens in return of injecting money to liquidity pools. These pools generate a marketplace where users can lend, borrow, or exchange tokens. yield farmers make money by trying to choose the best pools to invest in. Atomex [32] and liquality [33] also provide atomic swap service but support a rather limited number of coins. Black et al. proposed a new loan platform on Ethereum which accepts collateral in Bitcoin network [34]. Duggirala et al. designed an atomic swap using zero knowledge prove, increasing the privacy and security of participating parties in contract [35]. These newly devised protocols emphasize on the ever increasing demand for cross-chain protocols.

The significance of this work can be categorized as follows:

1. First of all, in section 2 we formalized the notion of atomic swaptions component and made a general form of them called *meta swaption* in which no smart contracts are needed and limited scrips like bitcoin script are sufficient. Then, using this meta form, we implemented several instances of swaptions depending of need, including late deposition and margin-free, despite the belief that there can not exist any margin-free swaption due to the limitation of HTLCs.

2. Employing the implemented swaptions, in section 3 we designed arbitrage opportunities on swaption market. This was not possible without our new techniques *option delegation* and *overlapping*. Overlapping helps reduce the cost of arbitrages and option delegation solves the multi-leader problem first mentioned in [19] and is independent of smart contracts.

3. Finally, in section 4 we introduce a network of swaptions and arbitrages that are in interconnection with each other. This network is called a tangled money market. Then, two theorems are proved to describe the operation of them and two protocols to solve the problem in either settings. The first theorem explains how several swaptions can be executed in concurrency and if a party participating in such a network follows our protocol,

his safety is guaranteed. The second theorem, gives a general statement in all types of possible tangled money markets with any topology while proving the safety is provided for complying parties to our protocol.

2. Atomic Swaption Revisited

To the best of our knowledge, Liu proposed the only implementation of atomic swaption which does not require either blockchain to support smart contracts [4]. Afterward, Tefagh et al. designed atomic bonded cross-chain debt (ABCD) as the first practical cross-chain bond platform in the form of atomic swaptions [5]. In this paper, we have designed a more general model of atomic swaption named *meta swaption* which is abstractly shown in Fig 1.

In what follows, we are going to briefly explain every module of the meta swaption, and through the rest of this paper, we will use this general form as the building block for making some new application-specific swaptions.

Similar to atomic swap and atomic swaption, HTLCs are used so that a party can make decisions by revealing a secret before a locktime or letting the locktime expire. In meta swaption also there are four types of such secrets:

- contract funding key
- leader key
- master key
- delegation key

Note that in HTLC contracts, generaly the locktime of the holder of the secret has to be greater than the other party.

In the Fig 1, a meta swaption is divided into five different parts, each representing a particular stage in the swaption process.

Note that in every module introduced in this paper, all transactions in the execution process are exchanged and signed by their corresponding parties before anything goes on chain. During this not-yet-confirmed transactions sharing, all parties are assured that no one can steal their money.

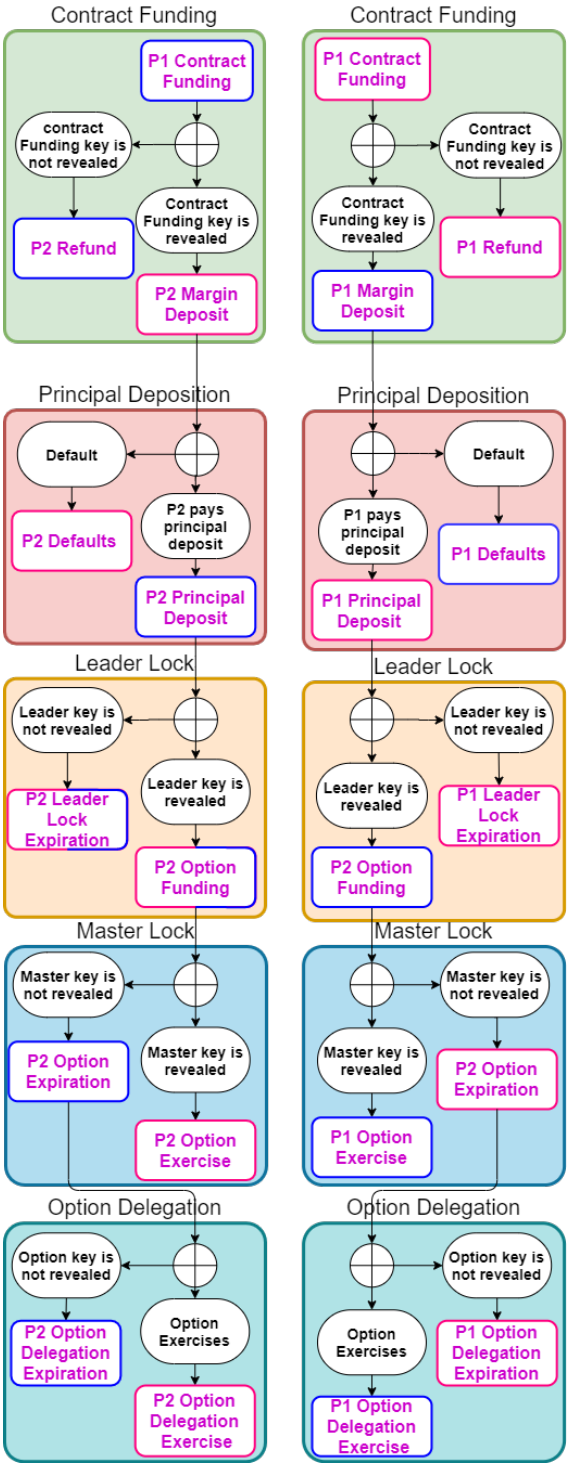


Figure 1: General overview of the meta swaption

- **Contract funding**: The funding for the swaption buyer (swaption owner) consists of premium and for the seller only margin. Depending on application, the owner may include margin in her funding. The owner has a relatively small amount of time to reveal contract funding key to buy the option. If she buys the option, premium goes to the seller and margins go to the principal deposition contracts.

- **Principal deposition**: After buying the swpation, each party has to deposit his principal within a specified time interval. If both parties cooperate, the next stage begins. The principal deposition transactions might have sighash type of any-one-can-pay[1] since nobody knows all of its inputs in the first place. In this case, since we do not know all the inputs of this transaction at the time of signing, we would not have the correct transaction ID [36]. Therefore, we can not create the transactions that get their inputs form this transaction until we discover what all the inputs of the this transaction are going to be. These inputs are only known at the time of broadcasting, not at the time of creating the transactions. So, at the time of creating the transactions in the next stage, the ID of the principal deposition transaction is unknown.

- **Leader lock**: Depending on the application, the swaption owner may decide to use a portion of her option duration as the allowed period of her late principal deposition in order to make money for her principal. In this stage, the party who deposits his principal earlier, leader key holder, locks his principal. After assurance of the owner's principal deposition, he is expected to reveal leader key determining if the swaption goes to the next stage or not. As mentioned in the last stage, the ID of the input of the transactions of this stage is not known at the time of creating. Thus, we have to use the sighash type of no-input[2][37] that allows us to create

[1] ANYONECANPAY
[2] NOINPUT

a transaction where none of its inputs are known. So, by this sighash we can create the transactions in the leader lock stage without any inputs. Later, at the time of broadcast, since the inputs of the principal deposition transaction are known and consequently we have its ID, we can use this transaction as an input for the transactions of the leader lock stage and broadcast them.

- **Master lock**: In this stage, the owner (master key holder) chooses weather she exercises the option or not using master key. By exercising the option, principals are exchanged and the swaption ends. By not revealing master key, letting the locktime expire, next stage begins. Like the last stage, since the inputs of the parent transaction of the transactions of this stage are not clear, we have to use the no-input sighash.

- **Option delegation**: This stage might be utilized in certain use-cases such as futures arbitrage discussed later. For fixing the problem of cyclic locktimes in master lock stage, we develop a method to delegate the option from owner to another desired party. The newly promoted party can later decide the execution of the swaption by the delegation key. In this stage, like last two stages we have to use the no-input sighash.

When using the no-input sighash, the party signs the transaction without specifying its inputs, so a malicious party has the ability to give any UTXO that belongs to the public key of this party. Hence, each party has to create a separate public key for each transaction that she makes. Using the stages described above, we can generate new instances of meta swaption targeting swaptions or bonds. Based on different goals, we can choose which of these stages appear in our instance. In the following subsections, we describe different swaptions devised for a variety of use-cases:

- Early deposition swaption

- Late deposition swaption

- Margin-free limited swaption

Notice that in Fig 2, Fig 3, and Fig 4 the locktimes P,T,M,E, and T' are calculated with respect to a common origin of time. In other words, the current time of the swaption establishment has to be added to all the locktimes shown in these stages, because they are not relative but absolute times.

2.1. Early Deposition Swaption

This type of swaption is the conventional form that is first introduced in [4]. Alice wants to exchange her ACoins with Bob's BCoins. We rebuild this type with our meta swaption extended form in the way depicted in Fig 2. We begin analysing each stage of this type by explaining every possible scenarios as follows:

- **Contract funding**: Alice and Bob broadcast their funding transactions. Alice's includes margin and premium and Bob's includes margin worth equal to Alice's margin.
- **Principal deposition**: In this stage, Bob is waiting for Alice to deposit her principal. There are two possible scenarios:
  - Alice does not deposit her principal. In this case, Bob also defaults and their margins are exchanged.
  - Alice deposits her principal but Bob does not. In this case, Alice is in master lock stage. So, she reveals master and takes Bob's margin besides her own principal.

  But if Bob fails to deposit his principal,
- **Master lock**: If both parties go to this stage, Alice can then use her option as mentioned earlier.

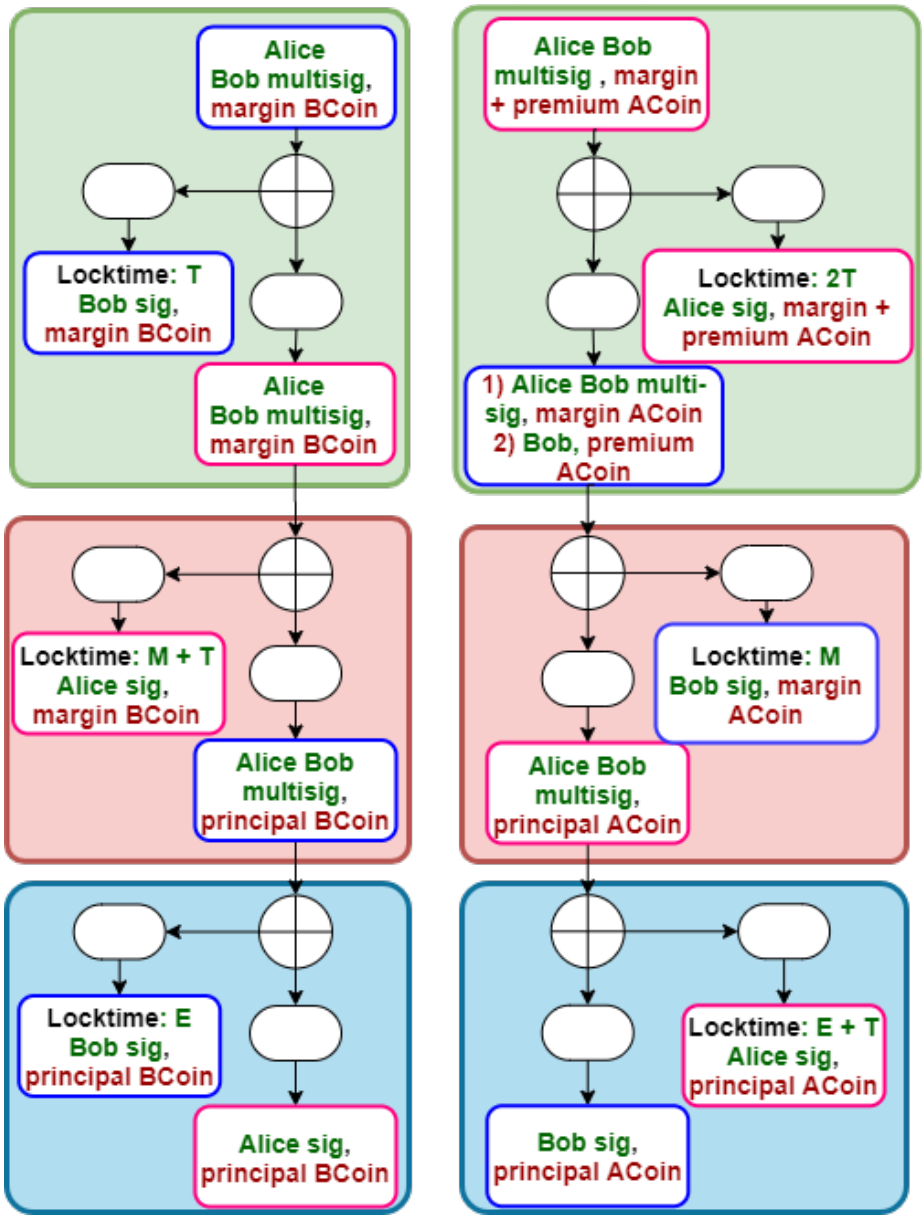


Figure 2: Early deposition swaption where the buyer (Alice) deposits her principal earlier than the seller (Bob). The pink-bordered transactions are broadcasted by Alice and the blue-bordered ones by Bob.

2.2. Late Deposition Swaption

In this type of swaption the swaption owner, Alice, needs the seller, Bob, to deposit his principal before her, so that she can settle other deals on other contracts before depositing her principal in this swaption. Alice can exploit this type of swaption in the situations where she does not own enough budget for her principal and she is going to make the required amount of capital using what she has takes from Bob. Since Bob has to wait a longer time for Alice to deposit, Alice has to pay more amount of premium compared to the early deposition swaption. One example use-case can be the futures arbitrages which will be discussed later. The overview of this swaption is shown on Fig. 3.

- **Contract funding:** The two parties broadcast their funding transactions. Alice includes margin in her funding and an amount of guarantee is added to Bob's funding besides his margin. We can use this guarantee amount to punish Bob in case of cheating. If he behaves normally, the guarantee will return back to him. The amount of guarantee is negotiable depending on the use-case.

- **Principal deposition:** When this stage begins Alice is waiting for Bob to deposit his principal. If Bob defaults, then Alice has two options to punish him:

  - She defaults, loses her margin and takes Bob's guarantee and margin.
  - She deposits her principal and enters the next stage.

- **Leader lock**: There are four possible scenarios:

  - In the last stage, Bob deposited his principal but Alice did not. Now Bob does not reveal leader key, so that Alice's margin is exchanged with Bob's margin.
  - Both have deposited their principals and Bob does not reveal leader key. Broadcasting the Alice's leader lock transactions, Alice gets Bob's guarantee.

- In the last stage, Alice deposited her principal but Bob did not (Second way to punish Bob). Now Alice broadcasts the Alice's leader lock expiration transaction and takes all of her principal back in addition to Bob's margin and guarantee that she has previously taken in the last stage.
- Both have deposited their principals. Bob reveals leader key and take back his guarantee. Afterward, they go to the next stage.

- **Master lock**: It is the time for Alice to use her option. She either exercises and principals are exchanged or reveals nothing and each party gets his principal back.

2.3. Margin-Free Limited Swaption

Until now, it was believed that Alice's margin deposit is necessary due to the limitations of HTLCs [4]. In this work, we propose a novel approach which allows Alice to participate in a swaption without depositing any margin. In the beginning of contract, Bob deposits an amount of BCoin as guarantee which is sent to Alice directly by revealing contract funding key. Alice also adds an amount of ACoin equal to Bob's guarantee to her premium, though none of them will be directly sent to Bob. Later, the premium and guarantee go to Bob in all possible situations except where Bob refuses to reveal the leader key when Alice does not default, then he will be punished by not getting back his guarantee. The amount of guarantee can vary depending on the Alice's need. In next section, we will explain the limitation imposed on the amount of guarantee in details. The execution procedure of the margin-free swaption is as follows:

- **Contract funding**: Bob pays his margin plus an amount of guarantee which prevents him from cheating in later stages. Alice also pays extra premium which in the case of Bob's honest behaviour pays back the guarantee to Bob. Guarantee in the Bob's section is directly sent to Alice after revealing the contract funding key.

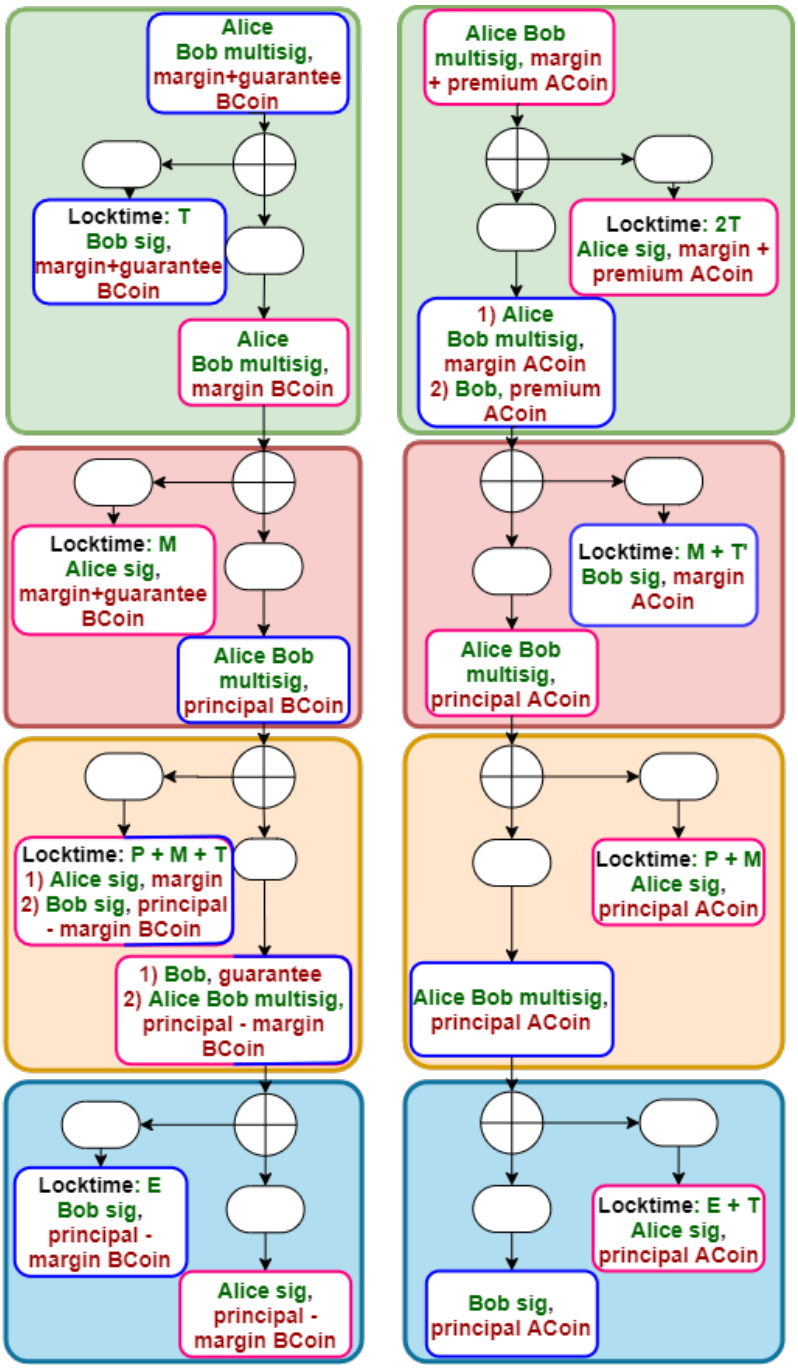


Figure 3: Late deposition swaption where the seller (Bob) deposits his principal earlier than the buyer (Alice). Pink-bordered transactions are broadcast by Alice and blue-bordered ones by Bob.

- **Principal deposition**: This stage is the same as previous principal deposition stages for Bob. He has M locktime to deposit his principal. If he does not, Alice takes his margin. In Alice's section, either she defaults and gives Bob the premium or she deposits her principal and goes to the next stage waiting for Bob to reveal the leader key.

- **Leader key**: If Bob has not deposited his principal until M locktime, Alice will also avoid depositing her principal and gives Bob the premium and ACoin guarantee while getting his margin from him. If both parties have deposited their principals when this stage begins, Bob's decision whether to reveal the leader key or not, determines the future of the swaption. If he refuses to reveal, he takes his own money back and Alice takes her own money including premium back. In this case, Bob loses his guarantee as a punishment. Otherwise, if he reveals the leader key, they both go to the next stage waiting for Alice to exercise her option. Additionally, by revealing the leader key, Bob finishes his task and it is the time to send back his guarantee. Hence at the end of this stage, Bob's guarantee will be paid back to himself.

- **Option funding**: This stage is similar to the last versions of swpation.

Locktimes in all of these swaptions have to stick to some general rules:

- T is the minimum amount of time or number of blocks needed for a mined transaction to be confirmed. In bitcoin it is 6 blocks which is approximately achieved in 1 hour.

- M is the time for Bob to deposit principal. Hence, can be equal or greater than T.

- T' can be relatively large, since it gives Alice the time she needs to deposit her principal.

- P is the locktime that Bob has to reveal the leader key. This has to be larger than T' so that Alice has to deposit before Bob's revealing time

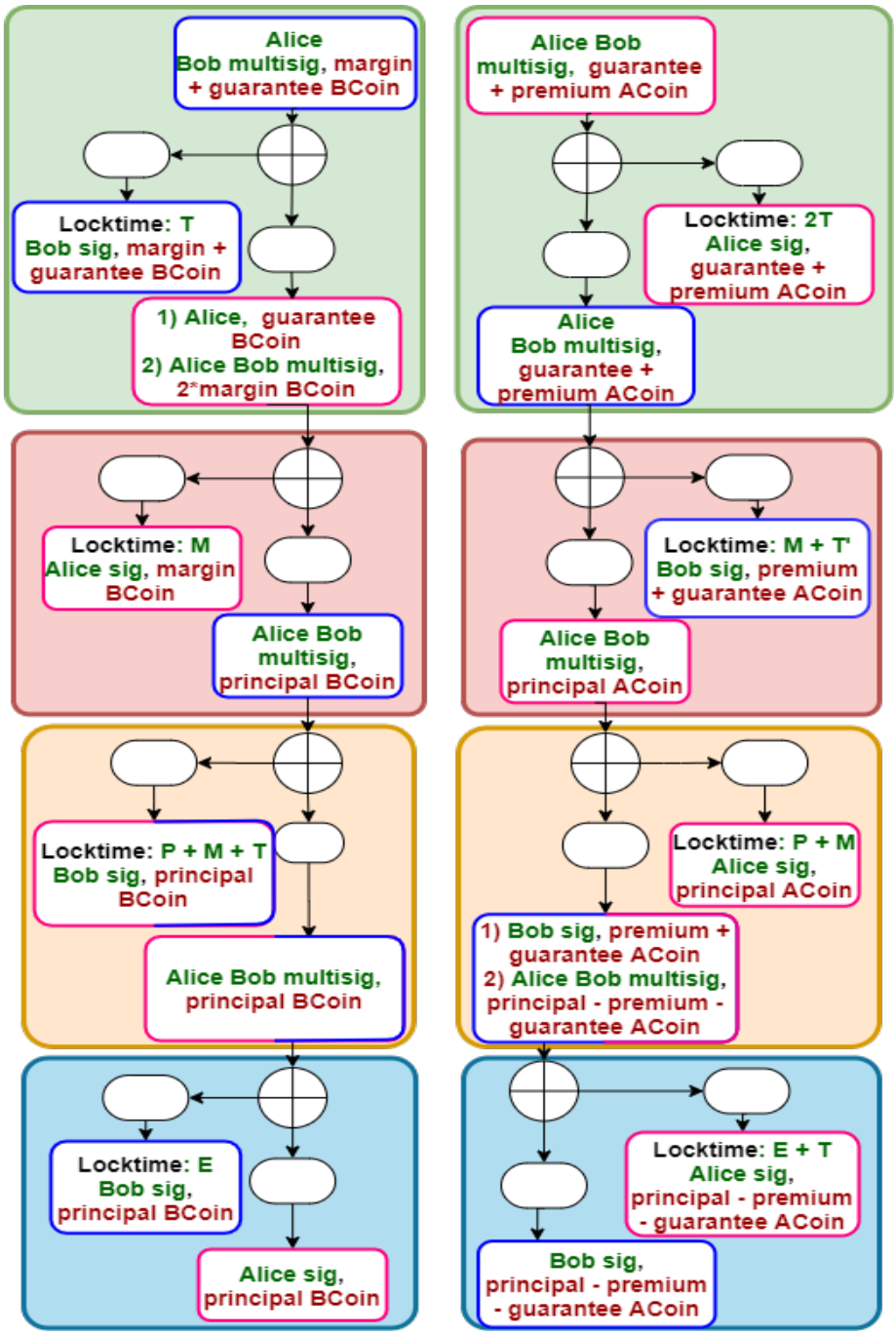


Figure 4: Margin-free limited swpation where the buyer (Alice) is not supposed to deposit any margin. Pink-bordered transactions are broadcasted by Alice and blue-bordered ones by Bob.

elapses.

- E is the time for Alice's option exercise which can be relatively large, since it is an option.

Note that we use T to make two locktimes differ in a reasonable amount of time to prevent cheating. However, this is the minimum difference needed. Hence, everywhere T is used, we can replace it with different larger values. We use T for simplicity though.

3. Futures Arbitrage

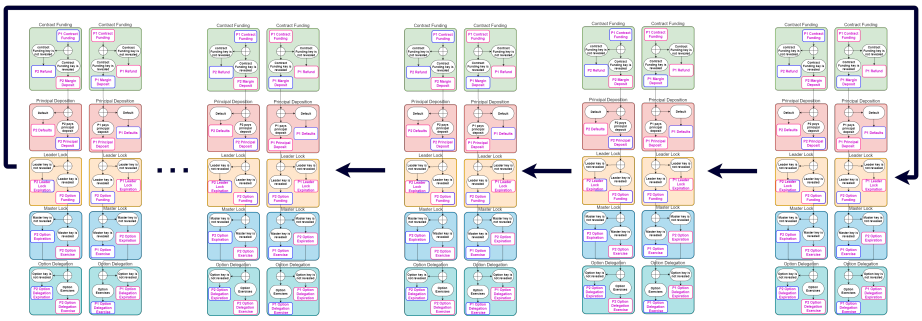


Figure 5: General form of the looped Arbitrage, composed of several instances of meta swaption.

In this section we are going to elaborate on the term of arbitrage by redefining it in swaption market in the following way: An arbitrage in the swaption market is an opportunity when a party, named exploiter, can exercise a chain of swaptions simultaneously, beginning from the exploiter's asset and also ending in the same asset, and profits by gaining more assets than the start point. Fig 5 shows the general form of an arbitrage made up of several instances of meta swaption forming a loop.

We introduce four different types of arbitrages that are possible by adoption of our meta swaption as their building blocks:

- **Early deposition** arbitrage
- **Late deposition** arbitrage

- **Margin-free** arbitrage
- **Bonded** arbitrage

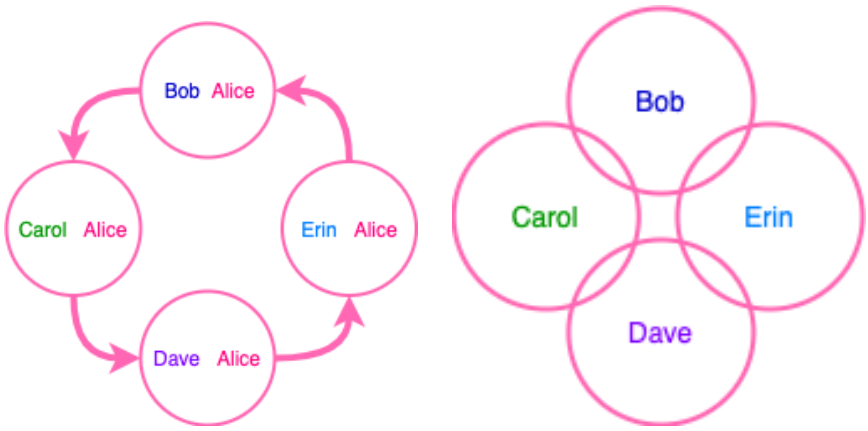
For further analysis, we give an example of arbitrage use cases. Alice, the exploiter whose initial asset is ACoin, is going to form an arbitrage making a chain of swaptions between Bob, Carol, Dave and Erin that take place concurrently. In section 4 we will explain the procedure of signing contracts for each party in details but first we discuss about the general overview of our execution processes. For convenient reason, we index each swaption by order, i.e. indexes of swaptions between Alice and Bob, and Alice and Carol, and Alice and Dave and Alice and Erin are respectively one to four.

Using the normal form of arbitrage shown in Fig. 6a, the exploiter has to pay margin in every single swaption besides generating lots of transactions and thus, transaction fees. To solve this and to exercise the swaptions concurrently, we use the overlapping technique which changes the structure of our arbitrage model to Fig. 6b.

The overlapping technique is the action of extending the output's required signatures of all transactions to other parties in order to make additional guarantee.

3.1. Early Deposition Arbitrage

Consider the following variation of our main example. Alice already has an amount of ACoin equal to the needed principal. First of all, she starts a swaption with Bob. After Bob deposits his principal, Alice can use his BCoins on her later swaption contracts. Then she trades these locked BCoins with Carol's CCoins, and these CCoins with Dave's DCoins and finally trades these DCoins with Erin's ACoins. According to the definition of arbitrage, now she has more ACoins than she primarily had. Since Alice primarily has enough ACoins, she only uses early deposition swpations. She technically can use a late deposition swaption for the first swaption, but since in that case she has to pay larger premium, it is not rational. She also uses the swaption overlapping to



(a) Arbitrage architecture using a loop of swaptions

(b) Arbitrage architecture using overlapping technique

Figure 6: Arbitrage with and without overlapping technique

transfer her assets between different swaption contracts. In Fig 7 this technique is shown. All parties share all transaction before any thing gets broadcast. After that, the procedure goes on as follows:

- **Contract funding**: Alice starts her contract with other parties simultaneously, by sharing the funding transaction with them. Then she decides whether to reveal contract funding key or not. If she does, all the swaptions go to the next stage. The locktimes are set descending from swaption number one to four, so that Alice can not cheat. In fact, Alice has to reveal contract funding before T, the minimum locktime among all swaptions, since otherwise at least one of them has expired and the entire arbitrage is not plausible any more. Note that if any of the funding stages fail, the entire arbitrage fails.

- **Principal deposition**: In this stage, every one has to deposit their principal. All swaptions are early deposition swaptions, so all the locktimes are ascending from swaption number one to four. Possible scenarios:

  – One of the parties defaults. In this case, all parties after the defaulter

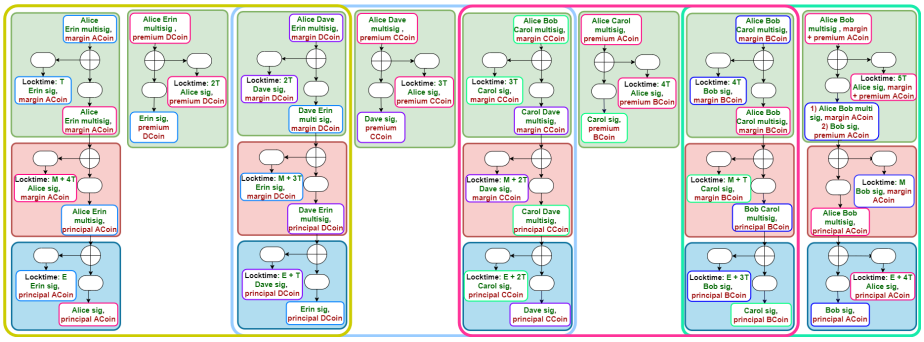


Figure 7: Example of an arbitrage, where Alice owns all ACoins primarily needed. On each transaction, signatures, output amount, and locktimes are specified. Pink-bordered transactions are broadcast by Alice, blue-bordered ones by Bob, green-bordered by Carol, purple-bordered by Dave and cyan-bordered by Erin. If there is a line between two transactions, then the source transaction is considered to be an input of the destination transaction.

also default. Hence, the first defaulter looses his margin and all the other parties transfer their margin with the next party in the swaption with the next index and the last party transfers her margin to Alice.

- All parties deposit their principal and go to the next stage.

- **Option funding**: It is time for Alice to use her option. She reveals master key for Erin, then each party reveals it for the previous one in order. Therefore, the locktimes are descending in all swaptions from swaption number one to four.

3.2. Late Deposition Arbitrage

Now, consider the case that Alice does not own enough ACoin to start such a procedure. She has to pay ACoin to Bob using a portion of ACoins she gets from Erin. This would only be possible if she had the ability to spend Bob's BCoins and do all the tradings with it and finally deposit the ACoins taken form Erin to the first swaption. To do so, she can use the late deposition swaption as the first swaption. We call it the late deposition arbitrage. Fig. 8 depicts

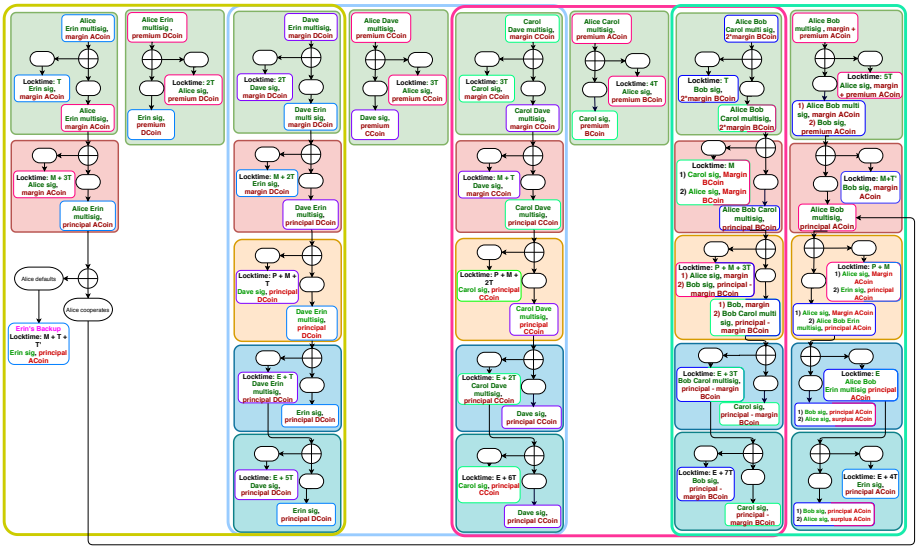


Figure 8: The schematic for late deposition arbitrage. The transactions with two border colors, can be broadcasted with both corresponding parties.

such an arbitrage. There are four swaptions, and Alice pays only premium in all of them. She also has to deposit margin in the first swaption. Alice's margin is only there to prevent her from cheating due to the cyclic locktimes on the funding stages (in the next section we will briefly explain this problem and our proposed solution). Therefore, there are five margins and four swaptions. The first swaption to start is the same as the last swaption to finish and it is actually between Erin and Bob (swaption number one). The five phases of late deposition arbitrage are:

- **Contract Funding**: Similar to the early deposition arbitrage, every one has exchanged funding and refund transactions. Three types of these stages:
  - Alice's first funding includes margin + premium.
  - Other fundings of Alice only include premium.
  - Bob's funding includes margin and an amount of guarantee. Here the amount of guarantee is equal to the margin, so Bob's funding is

worth twice other parties margins.

  - Other parties fundings include only margin.

Alice reveals the contract funding key by broadcasting Erin's margin deposit transaction. Then, Erin broadcasts Dave's margin deposit transaction and everyone broadcasts the margin deposit transaction of their previous party. No one has the incentive not to broadcast this transaction since if she does not, she is the only one who goes through a loss. When all the margin deposit transactions are broadcast, the arbitrage proceeds to the next stage.

- **Principal deposition**: All swaptions except for the first one are early deposition, so the locktimes of principal deposition on these swaptions are ascending from swaption number two to four. On the other hand, the first swaption is late deposition, so Alice's locktime has to be greater than Bob's. Alice's principal is in fact a portion of the principal that Erin deposits. Similar to the early deposition arbitrage, if any one defaults in this stage, and does not deposit his principal, the faulty party loses his margin (except for when it is Alice that exchanges her margin with Bob's) and other parties margins are exchanged with their previous party. As mentioned earlier, the first swaption is in fact between Erin and Bob, but Alice also puts margin to this swaption. The principal deposition of Erin is an any-one-can-pay transaction. The amount of $M + T'$ needs to be greater than $M + 3T$, hence if Erin defaults, Alice can default too. If Erin does not default and broadcasts her principal deposition transaction, Alice takes its output as an input to the principal deposition of the first swaption. We have given Erin a chance that if Alice does not do so, she can take her principal back and of course since Alice has defaulted, she exchanges her margin with Bob's. If everyone deposits their principal, we go to the next stage.

- **Leader lock**: Alice is waiting for Bob to reveal the leader key by broad-

casting the Erin's option funding transaction on first swaption. If Bob does so, Erin broadcasts Dave's option funding transaction and every other party broadcasts the previous party's option funding transaction respectively. Locktime for Alice is the shortest and increased from swaption number four to two and for Bob is the longest. Bob's option funding transaction can be broadcasted by both Carol and Bob. Otherwise, other parties can collude and cheat on Bob using the following attack: Bob reveals the leader key and others do not broadcast the Bob's option funding and take Bob's margin. Hence, when Bob can broadcast his own option funding, there would not be any problem of this kind.

Thus, if Bob reveals the key, every other party goes to the next stage. If Bob does not reveal, he loses his margin. If Bob reveals the leader key after $P + M$, Alice will take back her margin and immediately reveal the master key which results in taking Bob's money in both swaptions, thus discouraging Bob from delaying in exposure.

- **Option funding**: This is the time for Alice to use her option. She can reveal the master key by broadcasting the Erin's option exercise. She takes her surplus and Bob gets his ACoins. She has to reveal it within E locktime. Since in all swpations the buyer party, i.e. right-hand side party, is in charge of revealing the master key, the locktimes have to be descending. This means the locktime of the Erin's option expiration transaction has to be both longer and shorter than the Bob's option expiration. This is obviously impossible. Hence, we set the locktimes so that Carol's locktime is longer than Alice's, which makes it possible for Carol and Alice to cheat on Bob, but we solve this problem using the delegation stage. Now, if Alice exercises her option until E locktime, then every party can take their new coins. If Alice does not exercise until E, her option is delegated to Bob.

- **Option delegation**: If Alice has not exercised until E, Bob broadcasts the Erin's option expiration and Bob's option expiration transactions and

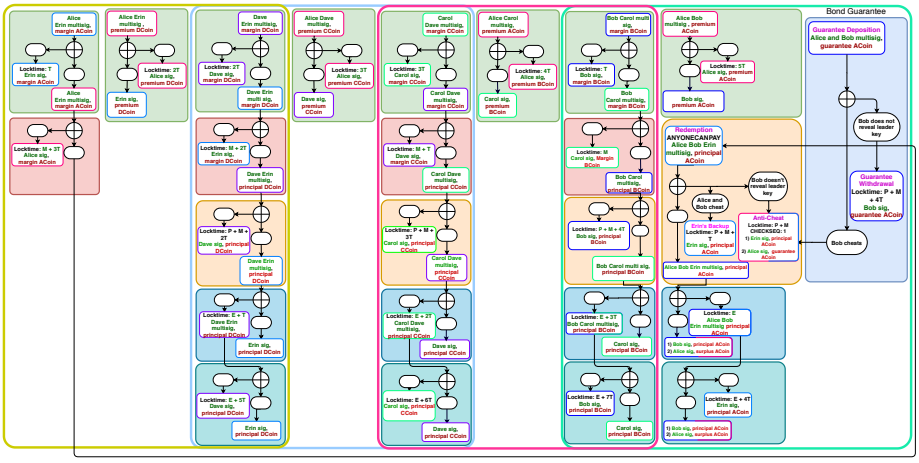


Figure 9: The bonded arbitrage using ABCD.

all other parties do the same (since it is their only chance to get their coins). The arbitrage is now in the delegation stage. Bob has the option and everyone is waiting for him. If he decides to exercise, everyone gets their new coins, otherwise everyone gets their former coins. The locktimes are descending for all swaptions except for the first one which is ascending (like last stages), since now Bob is the key holder of this stage and he is in charge of revealing the delegation key.

3.3. Margin-Free Arbitrage

Next, we are going to study the case in which Alice does not have enough ACoin for margin deposition in her swaption with Bob but still she can afford the minimal amount of ACoin as guarantee besides all the premiums. Using the margin-free swaption mentioned in previous section, an arbitrage can be built and exploited satisfying our need. Before all, we have to analyse the amount of guarantee in a margin-free swaption. If this type of swaption is used lonely without further contracts, the minimum valuable amount is enough for the amount of guarantee (even if it is less than premium). However, to use it alongside other contracts, e.g. as the first swaption in an arbitrage setting, Alice needs to set the guarantee value of Bob more than the summation of all

Stage	LT in column 1	LT in column $2i$	column $2i+1$
Contract Funding	$(N+1)T$	$(N-i+1)T$	$(N-i+1)T$
Principal Deposition	$M+T'$	$M+(N-i)T$	-
Leader Lock	$P+M$	$P+M+iT$	-
Option Contract	E	$E+(N-i)T$	-
Option Delegation	$E+NT$	$E+(2N-i)T$	-

Table 1: Locktimes for general case in an arbitrage. Each column in the table shows one column in the arbitrage.

the premiums which she wants to spend in her later contracts. Otherwise, Bob is incentivized to pretend himself as the other parties in later contracts, e.g. in our example Carol or Dave or Erin, and cheats on Alice by stealing Alice's premiums in the later swaptions by not exposing the leader key and loosing his guarantee. Note that if Bob acts honestly, the guarantees in both sides will be exchanged. In fact, assume the amount of margin is worth m times more than the amount of premium. The margin-free swaption can be used, if the number of parallel swaptions, is less than m. Otherwise, it is better to use a late deposition swaption instead. We do not include the figure for margin-free arbitrage, however, it is simply derivable from previous arguments.

3.4. Bonded Arbitrage

For final case, we suppose that Alice even does not have enough money for guarantee needs in a margin-free swaption. Alice still can make an arbitrage using the ABCD component devised in previous works, as her first swaption [5]. In this way, she does not need more money than the premiums for her arbitrage. We call this type of arbitrage, the bonded arbitrage, since the arbitrage exploiter bonds her initial asset in the first place. The bonded arbitrage is shown in Fig. 9.

Now, consider the general case for the arbitrage when there are N swaptions. Table 1 shows locktimes for each stage in a general arbitrage. Each column in the table shows one column in the arbitrage. Moreover, for simplicity we put locktimes so that different stages do not have overlap in time. In some cases

overlapping stages can cause trouble. Two examples are:

- M has to be larger than $(N+1)T$. Otherwise, Alice and Carol can broadcast the Bob defaults transaction and then broadcast the Alice's refund and take Bob's margins.

- In the third stage, $P + (N-1)T$ has to be greater than T'. Otherwise, Alice can wait untill $P + M + (N-1)T$ not depositing principal, then Bob has to broadcast the Bob's leader key expiration and then Alice will deposit and then broadcast the Alice's leader key expiration. So, she takes Bob's margin without loosing anything but premium.

Note that Alice in fact has E time to use her option. Then Bob takes the option, so we can say that Alice does not have any option but it is not true for two reasons: 1) Arbitrages are always zero-risk. 2) If she deposits her principal it means that all the parties have cooperated (if one does not, she does not deposit principal and does not lose anything, just the margins are exchanged which is inevitable). When all parties have cooperated and arbitrage is zero-risk every thing is fine so why would Alice not exercise?

Note that in such cases Alice is always going to use her option, so Alice is not in fact delegating her option to Bob, but by having the delegation stage, she only assures Bob that he would not be cheated.

4. Tangled Money Market

So far, we have studied an arbitrage with one exploiter and all other parties only as sellers. Now, imagine each of the sellers can be an exploiter in a second arbitrage using the principal from the first arbitrage. We call such intertwined arbitrages as tangled money market whose example is shown in Fig. 10. Furthermore, these tangled swaptions do not have to be in the form of arbitrage. In other words, a series of swaptions entangled together is counted as a tangled money market. In this section, we prove two theorems for tangles. And by designing a protocol we provide a constructive proof for our final theorem.

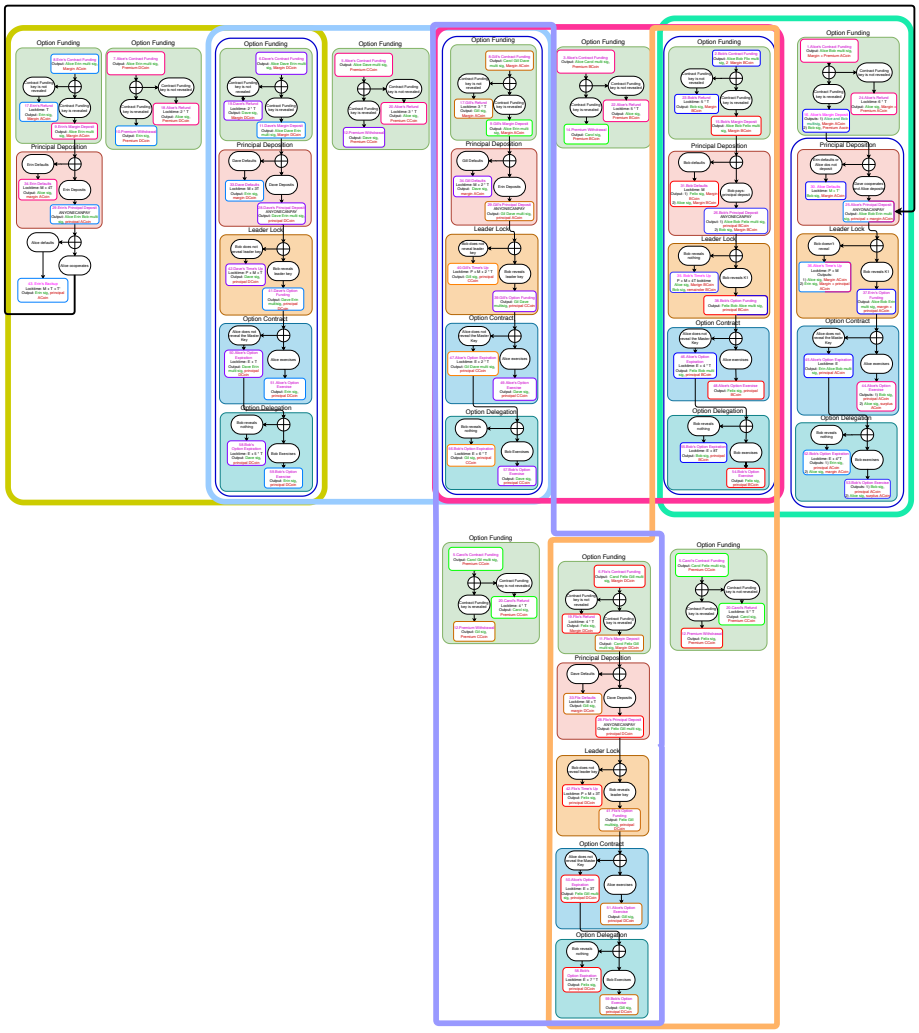


Figure 10: a tangle of two arbitrage loops; one for Alice and one for Carol

Note each transaction may have multiple inputs and outputs in real word. We only consider those transaction involved in a tangle have single input and output. We define $P = \{p_1, , p_2, .., p_n\}$ as the set of distinct parties which are participating in the tangle even through different accounts.

First of all, to analyse a tangle we introduce two graph models:

1. Swaptions graph: $\mathcal{G}_s = (V, E)$ where V is the finite set of vertices each representing a swaption, and E the finite set of edges. Each edge is an ordered pair of vertices. In a set of overlapping swaptions, each swaption is shared between three parties p_1, p_2, p_3, where p_1 is the buyer of the swaption. In order to build the $\mathcal{G}_s$, each of these swaptions is shown as two separate swaptions; one between p_1 and p_2, and the other one between p_1 and p_3. This is another interpretation of the overlapping process. Where in real world, p_1 is an intermediary who mediates between p_2 and p_3, in this interpretation, she actually takes p_2's principal to deposit in the swaption between herself and p_3. For each vertex $v \in V$ there are two functions $\beta : V \to P$ and $\sigma : V \to P$ which indicate the swaption buyer and seller respectively. Each edge represents transfer of principal between two swaptions. An example of this graph is shown in Fig. 11.

2. Principal flow graph: This graph represents the flow of principal and leader key. $\mathcal{G}_p = (V_p, E_p)$ where V_p is the finite set of vertices and E_p the finite set of edges. For each swaption in $\mathcal{G}_s$ there are two vertices in V_p corresponding to it. Each of them represents a party in the swaption. The function $\ell(v)$ where $\ell(v) : V_p \to P$ returns the party related to vertex v. Each $e \in E_p$ is an ordered pair of vertices $(h(e), \tau(e))$ that $h(e)$ and $\tau(e) \in V_p$. For the above example there would be four vertices with labels p_1, p_2, p_1 and p_3. Clearly, there is a correspondence between each vertex of $\mathcal{G}_s$ to a pair of vertices in $\mathcal{G}_p$. We define the function $\rho : E_p \to \{i, x, t\}$, such that the following properties hold:

    - For each $e \in E_p$ that $\rho(e) = i$, e connects two vertices of one swaption and represents the deposition order inside a swaption i.e. $h(e)$ has to

deposit earlier than $\tau(e)$ by the locktime rules. Note that for each $v \in V_p$, there is a unique $e \in E_p$ such that v is either $h(e)$ or $\tau(e)$, since each vertex is participating in a single swaption. We define function $\mathcal{O} : V_p \to V_p$ where for a vertex $v, \mathcal{O}(v)$ is the other vertex connected to the unique edge $\tilde{e}$ with $\rho(\tilde{e}) = i$ connected to v.

- For each $e \in E_p$ that $\rho(e) = x$, e represents the flow of principal between swaptions i.e. the principal moves from $h(e)$ to $\tau(e)$. We also define successor and precursor of a vertex as the functions $s : V_p \to V_p \cup \emptyset$ and $p : V_p \to V_p \cup \emptyset$ such that for each $v \in V_p$ if there is $e \in E_p$ that $\rho(e) = x$ and $h(e) = \mathcal{O}(v)$ then $s(v) = \tau(e)$ and if there is no such e then $s(v) = \emptyset$ and for each $v, u \in V_p$ if $s(v)$ is u than $p(u)$ is v and if there is no $v \in V_p$ such that $s(v)$ is u, then $p(u)$ is $\emptyset$. Functions s and p are well-defined since there can not be any two edges v and u with $\rho(v) = \rho(u) = x$ such that $h(u) = h(v)$ or $\tau(u) = \tau(v)$. As an intuition, given our previous example, the successor of p_1's vertex corresponding to the swaption with p_2 is his vertex corresponding to the swaption with p_3, and the precurser of p_1's vertex corresponding to his swaption with p_3 is his vertex correspondingo the swaption with p_2.

  The $\mathcal{G}_p$ graph of previous example is shown in Fig. 12. It is clear that for each $v \in V_p$, $\ell(s(v)) = \ell(v)$, since otherwise the party $\ell(v)$ transfers his asset to party $\ell(s(v))$ without getting anything back, which is not rational. With the same argument, $\ell(p(v))$ is equal to $\ell(v)$. We call this, the rationality property of $\mathcal{G}_p$.

- Consider all $v \in V_p$ that there is no edge $e \in E_p$ that $\rho(e) = x$ and $\tau(e) = v$. Then, for each $i \in \mathbb{N}$ that $\mathcal{O}(s^i(v)) \neq \emptyset$, there is an edge e' that $\rho(e') = t$ and $\tau(e') = v$ and $h(e') = \mathcal{O}(s^i(v))$. This type of edge, will later be used to impose a deposition order between two vertices in two adjacent swaptions in order to guarantee the safety of parties who inject principals to the system rather than transform it

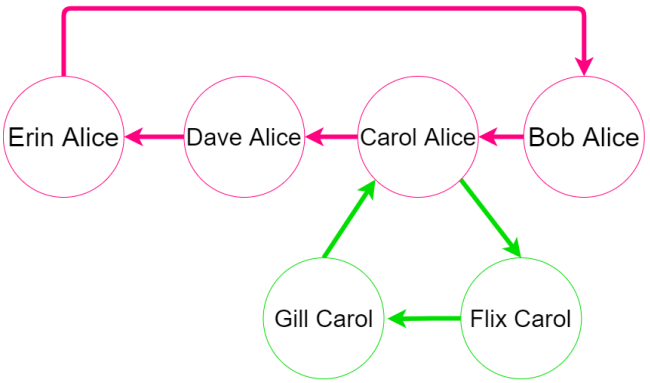


Figure 11: Example for swaptions graph

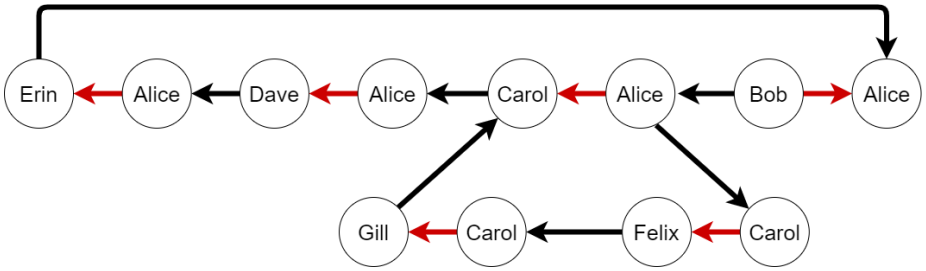


Figure 12: Example for $\mathcal{G}_p$ graph

from another swaption.

It is worth mentioning that in $\mathcal{G}_p$ there is no cycle c that for all of its edges e $\rho(e) = x$, since otherwise the principal transferred through c would not have any sources.

Definition 1. *Concurrent Tangle*

A tangle T is concurrent if the swaptions graph representing T is a strongly-connected directed graph.

Definition 2. *Association Set*

In the $\mathcal{G}_s$ graph, a vertex v is in the association set of party p_i if $\beta(v) = p_i$ or $\sigma(v) = p_i$.

For later usage, we need to find the p_{master} for a tangle. Given a swaptions graph $\mathcal{G}_s = (V, E)$ of a tangle T, we design a graph $\mathcal{G}_r = (V_r, E_r)$ such that $V_r = \mathcal{P}$ and for every $v \in V$ there is an edge $\hat{e} \in E_r$ for which $h(\hat{e}) = \beta(v)$ and $\tau(\hat{e}) = \sigma(v)$. Each vertex of $\mathcal{G}_r$ from which all the vertices are reachable [38] is a candidate whose label is eligible to be the p_{master} of T.

For every graph $\mathcal{G}_1 = (V_1, E_1)$, according to the definition of *feedback edge set* previously stated in [39], given L the feedback edge set of $\mathcal{G}_1$ that $L \subseteq E_1$, the edge-induced subgraph $\mathcal{G}_2 = (V_2, E_2)$ of $\mathcal{G}_1$ that $V_2 = V_1$ and $E_2 = E_1 - L$ is acyclic.

Definition 3. *Protected*

Assume subgraph $\mathcal{G}_t = (V', E')$ of $\mathcal{G}_p$ where $V' = V_p$ and $E' = \{e \in E_p | \rho(e) \neq t\}$. A tangle T is protected if

1. *For all the source*[3] *vertices v in $\mathcal{G}_t$, $\ell(v)$ has to be identical and we assume $\ell(v) = p_{leader}$.*

2. *$\mathcal{G}_p$ has a feedback edge set $L = \{e_0, e_1, ..e_l\}$ that for all $e_i \in L$, $\ell(\tau(e_i)) = p_{leader}$ and $\rho(e_i) \in \{i, t\}$.*

Definition 4. *Cooperable*

A concurrent tangle T is cooperable if it is protected and there is at least one party eligible to be its p_{master}.

Now, we provide an example for a cooperable tangle. Fig. 14 shows the swaptions graph, and Fig. 13 represents the principal flow graph of this example. All the source vertices in the principal flow graph have the label B, thus the party B is p_{leader}. The principal flow graph has a feedback edge set with all the edges pointing to B's vertices. Additionally, the Fig. 15 shows the $\mathcal{G}_r$ in which there is a path from vertex A to every other vertex. Therefore, the party A is the p_{master} of this concurrent tangle.

[3] A vertex v in the graph such that there is no edge e that $\tau(e) = v$

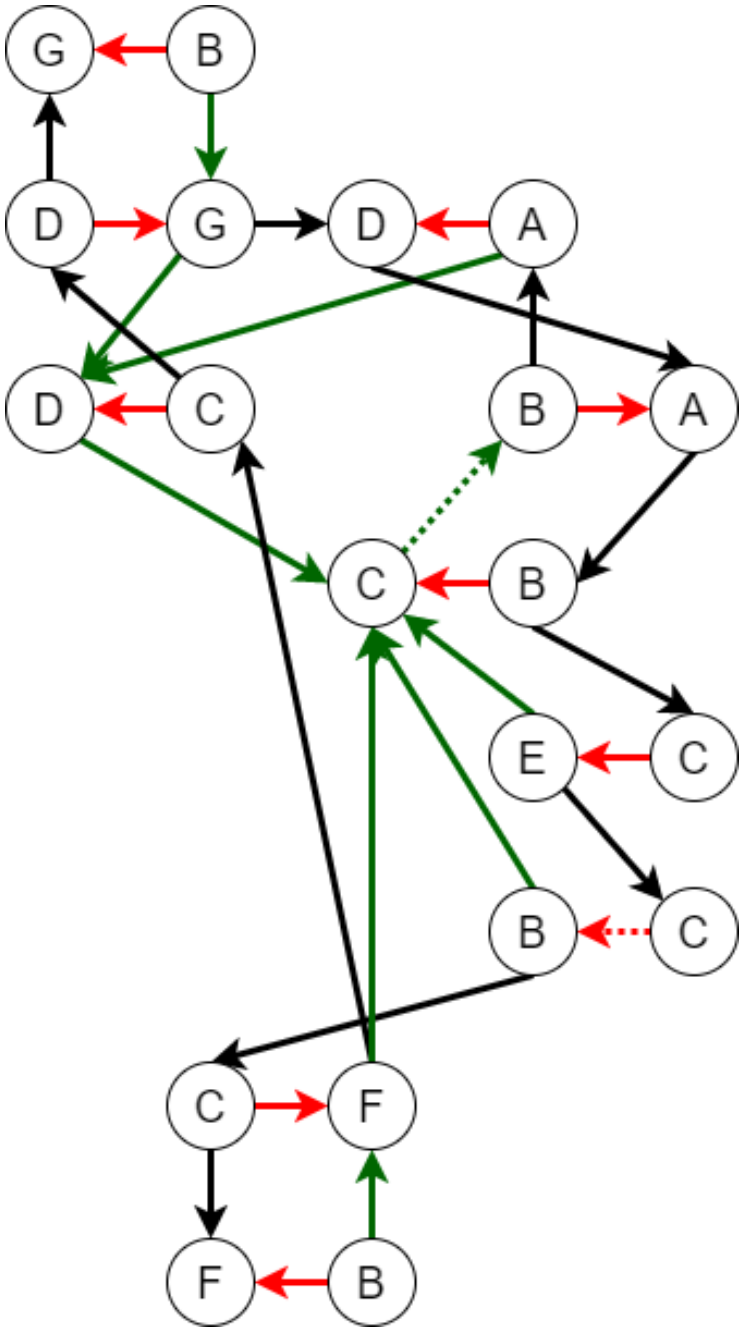


Figure 13: Principal flow graph of a cooperable tangle. Green, red and black edges represent edges whose ρ function is equal to t, x and i respectively. Dotted edges are the feedback edge set of the graph.

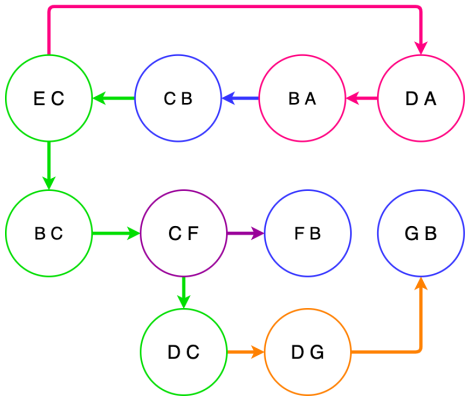


Figure 14: Swaptions graph of a cooperable tangle. A node with the colors pink, blue, green, purple, and orange represents a swaption whose buyer is A, B, C, F, and G, respectively. The color of each edge represents the party who is transmitting the principal.

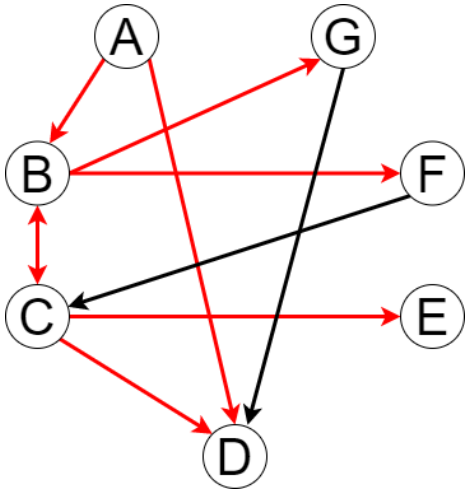


Figure 15: The $\mathcal{G}_r$ graph for our example to find the party p_{master}. Red edges show that the parties B, C, D, E, F, and G are reachable from party A.

Before delving into the first theorem, we need to define the required security property for executing such a tangle. To do so, we define multiple classes for the future of every distinct parties in the tangle. We assume that for a fix desired party, every other party can act cooperatively for cheating on him.

To be consistent with prior terminology first introduced by Herlihy in his paper "Atomic Cross-Chain Swaps", for a party p_i we define these states [19]:

- Nodeal: There is no swaption in p_i's *associated set*, that has been executed. Hence, he keeps his prior assets without getting any new assets.
- Deal: The swaptions in p_i's association set are all properly executed and the amount of principals inside them are finally swapped.
- Freeride: The party p_i does not pay anything through the entire tangle while getting some asset as premium or principal at the end of execution phase.
- Underwater: In at least one swaption, he does not gain any new principal, while he loses his principal.

Now we can properly give a definition of the needed security property for our tangle: A protocol P for a tangle T is secure if after execution of P on T, none of the parties who have adhered to P enter the underwater state. Next, it is time to state the first theorem:

Theorem 1. *For every cooperable tangle T, there is a secure protocol P.*

Lemma 1. *In the edge-induced subgraph $\mathcal{G}_k = (V_k, E_k)$ of $\mathcal{G}_p = (V_p, E_p)$ of the concurrent protected tangle T, that $V_k = V_p$ and $E_k = E_p - \{e \in E_p \mid \tau(e) = p_{leader}, \rho(e) \neq x\}$, for each sink[4] vertex s, $\ell(\mathcal{O}(s)) = p_{leader}$.*

Proof. We prove the above lemma by contradiction. Assume there is a sink vertex $s \in V_k$ that $\ell(\mathcal{O}(s)) \neq p_{leader}$.

[4] A vertex v in the graph such that there is no edge e that $h(e) = v$

Firstly, for the vertex $\mathcal{O}(s)$ there must exist an edge $e \in E_k$ such that $\rho(e) = x$ and $h(e) = \mathcal{O}(s)$, since otherwise the vertex in $\mathcal{G}_s$ corresponding to s would have no output edge and thus the $\mathcal{G}_s$ would not be strongly connected.

Secondly, there must exist an edge $e' \in E_p$ that $\tau(e') = \mathcal{O}(s)$ and $\rho(e') = x$, since otherwise the vertex $\mathcal{O}(s)$ would be one of the source vertices of $\mathcal{G}_t$ graph defined in the definition 3. Then, according to definition 3, $\ell(\mathcal{O}(s)) = p_{leader}$ which contradicts the assumption.

Thirdly, in $\mathcal{G}_k$, there must exist an edge $\tilde{e} \in E_k$ which $\rho(\tilde{e}) = i$ and $\tau(\tilde{e}) = s$. Since if $s = h(\tilde{e})$, this would contradict the assumption of s being a sink. On the other hand, as mentioned before, in $\mathcal{G}_p$ there is a unique $\tilde{e}$ which $\rho(\tilde{e}) = i$ connected to each vertex. So if there is no $\tilde{e} \in E_k$ that $\rho(\tilde{e}) = i$, this unique edge would be a member of $E_p - E_k$ and would have been removed in $\mathcal{G}_k$. According to the definition of $\mathcal{G}_k$, this means that $\ell(\mathcal{O}(s)) = p_{leader}$ which contradicts the assumption.

Finally, we know that $p(\mathcal{O}(s)) = \mathcal{O}(h(e'))$. Now, in $\}_k$, if there is $i > 1$ that $p^i(\mathcal{O}(s)) = \emptyset$, then $p^{i-1}(\mathcal{O}(s)$ has no input edge w that $\rho(w) = x$. Moreover, according to rationality property of $\mathcal{G}_p$, the $\ell(p^{i-1}(\mathcal{O}(s))) = \ell(\mathcal{O}(s)) \neq p_{leader}$. Therefore, according to the definition of $\mathcal{G}_p$ there must exist an edge $\hat{e} \in E_k$ that $\rho(\hat{e}) = t$ and $\tau(\hat{e}) = p^{i-1}(\mathcal{O}(s))$ and $h(\hat{e}) = s$. Thus, the vertex s is not a sink vertex for $\}_k$ which contradicts the assumption. If there is no such i, since the number of vertices is finite, there must exist $j < k \in \mathbb{N}$ that $p^k(\mathcal{O}(s)) = p^j(\mathcal{O}(s))$. Hence $p^{k-j}(\mathcal{O}(s)) = \mathcal{O}(s)$. Thus, $\mathcal{O}(s) = s^{k-j}(\mathcal{O}(s))$ which contradicts with s being a sink, because $s(\mathcal{O}(s)) \neq \emptyset$. □

To prove theorem 1, we try to design the protocol $\mathcal{P}$ that satisfies the requirements of the theorem. We call this protocol the *Synchronous protocol*. Since, the model does not allow us to use extra locks and stages in a swaption, the protocol has to be designed somehow that in all the swaptions the same contract funding, leader and master keys are used. Thus, the only party who actually has the option is p_{master}, and the only party who knows the leader key is p_{leader}. Other parties can have their own arbitrages but they do not have any option. Hence,

p_{master} is the one who should pay all the premiums. Furthermore, consider the case that any other arbitrage exploiter has to pay the premiums in his own loop. In this case, since p_{master} is holding the contract funding key, if any other party p_i has to pay premium, p_{master} can appear as $\mathcal{O}(p_i)$ and take his premium by early revealing contract funding key and then quitting the rest of the process. For these two reasons, p_{master} has to pay premium for all the swaptions in the tangle. So, if there are other arbitrage exploiters, we design our protocol as follows:

Each exploiter pays premiums in his own arbitrage. He takes a premium from p_{master} which is worth more than sum of all the premiums he has to pay. This way, p_{master} is in fact the only premium payer of the tangle. In fact, these exploiters act as the intermediary sellers that help the p_{master} to complete her contracts in exchange for the extra premium they get from her. Therefore, if the structure of the tangle is exposed, the p_{master} can reconstruct the tangle by making all the contracts and avoid paying extra premium. Hence, the tangle structure has to be anonymized.

The procedure of synchronous protocol is divided into separate phases which are discussed below.

1. **Contract initiation**: In this phase, transactions of contracts are written, signed and shared. Note that in this phase the contract funding transactions are not going to be signed. In every swaption, there are five locktimes for each party. To set these lockstimes, parties follow some rules. Algorithms 1. First of all, let's say a series of locktimes is in ascending or descending consistence with a *directed acyclic graph* (DAG), if it is following the topological order of the graph in ascending or descending direction.

    (a) Contract funding: Given $\mathcal{G}_p = (V_p, E_p)$ construct the graph $\mathcal{G}_c = (V_c, E_c)$ that $V_c = V_p$ and each edge $e \in E_p$ that $\rho(e) = t$ is removed and every edge $e \in E_p$ that $\rho(e) = i$ is redrew from buyer to seller of the swaption. Consider the subgraph of $\mathcal{G}_c$ in which all edges e that

$\rho(e) = i$ are removed. The parties who have to deposit margin are sources of this graph. The locktimes of contract funding transactions are consistent with $\mathcal{G}_c$ in the descending order. Since $\mathcal{G}_c$ may have some cycles, we have to add some modifications to this graph before imposing value of locktimes. These modifications will result in adding extra margins in the tangle. In what follows, we provide a solution to achieve the minimum number of margin depositions. Assume that $\hat{L} = \{\hat{e}_0, \hat{e}_1, .., \hat{e}_l\}$ is a minimum feedback edge set of $\mathcal{G}_c$. For every $\hat{e_i} \in \hat{L}$ if $\rho(\hat{e}_i)$ is i, replace $\hat{e}_i$ with $\tilde{e}_i$ that $\rho(\tilde{e}_i) = x$ and $\tau(\tilde{e}) = h(\hat{e}_i)$. There must exist such a $\tilde{e}$, since otherwise $\hat{e}_i$ could not be in any cycle which contradicts with $\hat{e}_i \in \hat{L}$. Then, $\hat{L}$ is still a minimum feedback edge set of $\mathcal{G}_c$ with all $\tilde{e} \in E_c$, $\rho(\tilde{e}) = x$. The locktimes can be set in descending consistence with the subgraph of $\mathcal{G}_c$ with edges in $\hat{L}$ removed. In addition, for each edge $\tilde{e} \in \hat{L}$ the vertex $h(\tilde{e})$ can not use the margin of $\tau(\tilde{e})$ and has to deposit an additional margin or use a marginless instance.

To achieve this minimum number, a reliable oracle is needed to inform each party about necessity of her margin. This oracle can either be a reliable third party or a peer to peer algorithm to find a feedback edge set that does not leak any additional information about the tangle structure.

(b) Principal deposition: Given T is protected, we can assume the set $L = \{e_0, e_1, .., e_l\}$ a feedback edge set of $\mathcal{G}_p$ with minimum length that for each $e_i \in L$, $\ell(\tau(e_i)) = p_{leader}$ and $\rho(e_i) \in \{i, t\}$. Set $\mathcal{G}_f$ a subgraph of $\mathcal{G}_p$ induced by removing edges in L is a DAG. The principal deposition locktime of each swaption is set consistent with $\mathcal{G}_f$ as ascending, if an oracle is existent. Otherwise, the locktimes are set in ascending consistence with the edge-induced subgraph of $\mathcal{G}_p$ in which all edges e that $\ell(\tau(e)) = p_{leader}$ and $\rho(e) \in \{i, t\}$ are removed. Some of the swaptions remain inconsistent anyway. Later,

these are proved not to cause any problem. However, the number of inconsistent swaptions is minimal if there is an oracle.

(c) Leader key: The leader key locktime of each vertex is determined exactly the same as the order of principal deposition locktimes except that here we use the descending order.

(d) Option funding: As mentioned earlier, in $\mathcal{G}_p$ there is no cycle c that for all of its edges e $\rho(e) = x$. Since the edges e that $\rho(e)$ is x are the same in $\mathcal{G}_c$ and $\mathcal{G}_p$, in each cycle in $\mathcal{G}_c$ there is at least one edge e that $\rho(e) = i$. Therefore, $\mathcal{G}_c$ has at least one feedback edge set that for each of its members e, $\rho(e) = i$. Take one of these feedback edge sets with minimum size as L_m. Now, consider the subgraph of $\mathcal{G}_c$ in which edges in L_m are removed. In case of having an oracle, the locktimes of option funding transactions are set in descending consistence with this subgraph. Then, define $\mathcal{D} \subseteq \mathcal{P}$ that for each $p_i \in \mathcal{D}$, there is an edge e that $e \in L_m$ and $p_i = \ell(\tau(e))$. If there is no such an oracle, the locktimes are set in descending consistence with $\mathcal{G}_c$. Since $\mathcal{G}_c$ might not be a DAG, some swaptions of some parties might remain inconsistent. Then, $\mathcal{D} \subseteq \mathcal{P}$ is constituted including these parties. We finally set the players on $\mathcal{D}$ the delegators of the tangle. Using the oracle, we can minimize the number of delegators through the tangle.

(e) Option delegation: The order of locktimes on this stage is the same as option funding stage. In this stage, if one of the delegators reveals his key, the contract will be exercised, and otherwise, if they let the time expire, everything will be reverted. In algorithm 1 assume that the delegation locks are primarily shared between all parties.

Now, remember we said actual swaptions use the overlapping technique. In the case of overlapping, the same rules are applied to determine locktimes, but with slight modifications. For each of the stages, for all $e \in E_p$ that $\rho(e)$ is x, the vertices $h(e)$ and $\tau(e)$ form one overlapping swaption. The

simple modification would be: if the locktimes are descending, all of these parties would set the maximum locktime and if ascending the minimum. After writing contracts properly, p_{master} reveals the contract funding key. Algorithm 2.

Algorithm 1 phase 1

1: $\mathcal{G}_s = (V, E)$ ▷ $\mathcal{G}_s$ is the swaptions graph of the tangle
2: $InitState[|V_g|][2]$ ▷ Init with *null*
3: $SwaptionStage[|V_g|][2]$ ▷ Init with *null*
4: **function** SETSTATE(swaption s, party p, stage g)
5: $InitState[s][\beta(s) == p] = g$ ▷ p creates transactions on her side of s up to the g stage.
6: **end function**
7: **function** GETSTATE(swaption s, party p)
8: **return** $InitState[s][\beta(s) == p]$
9: **end function**
10: **function** GETSTATE$\mathcal{O}$(swaption s, party p)
11: **return** $InitState[s][\sigma(s) == p]$
12: **end function**
13: **function** WAITUNTILL(condition c)
14: **while** c **do**
15: wait
16: **end while**
17: **end function**
18: **function** ECHO(stage s)
19: waitUntill(there is no swaption s in the association set of p that getState$\mathcal{O}$(s, p) is s)
20: **for each** swaption s in the association set of p **do** setState(s, p, s)
21: **end function**
22:
23: **if** p is p_{master} **then**
24: **for each** swaption s that $\beta(s)$ is p**do** setState(s, p, *principal*)
25: waitUntill(there is swaption s that $\beta(s)$ is p and getState$\mathcal{O}$(s,p) is *null*)
26: ECHO(*leader*)
27: **for each** swaption s in the association set of p **do** setState(s, p, *option*)

28: echo(*delegation*)

29: **else if** p is p_{leader} **then**

echo(*principal*)

30: **for each** swaption s in the association set of p, **do** setState(s, p, *leader*)

31: echo(*option*)

32: echo(*delegation*)

33: **else**

34: echo(*principal*)

35: echo(*leader*)

36: echo(*option*)

37: echo(*delegation*)

38: **end if**

Algorithm 2 phase 2

function SETSTAGE(swaption s, party p, transaction t)

$SwaptionStage[s][\beta(s) == p] = t$

p broadcasts her transaction t in swaption s

end function

function GETSTAGE(swaption s, party p)

return $SwaptionStage[s][\beta(s) == p]$

end function

function GETSTAGE$\mathcal{O}$(swaption s, party p)

return $SwaptionStage[s][\sigma(s) == p]$

end function

function WAITFORFUNDING(party p)

$premiums\_amount = 0$

$needed\_premiums = 0$

for each swaption s in the associated set of p that $\beta(s)$ is p **do**

$needed\_premiums$ += The amount of premium in swaption s

end for

while $premiums\_amount$ is less than $funding$ **do**

$premiums\_amount = 0$

for each swaption s in the associated set of p and $\beta(s)$ is p and getStage$\mathcal{O}$(s, p) is $funding$

$premium\_amount$ += The amount of premium in swaption s

end while

end function

function REVEAL(party p, key k) ▷ party p reveals key k by broadcasting its corresponding transaction

for each swaption s in the associated set of p **do**

if k is contract funding **then**

return setStage(s, p, $margin$)

else if k is leader **then**

return setStage(s, p, $option$)

else if k is master **then**

end if

end for

end function

if p is p_{master} **then**

for each swaption s that $\beta(s)$ is p, setStage$(s, p, funding)$ ▷ including the high-value premium for each exploiter $\sigma(s)$

for each swaption s in the associated set of p **do**

waitUntill(getStage$\mathcal{O}$(s, p) is not $funding$)

if getStage(s, p) is not $funding$, **then** setStage$(s, p, funding)$

end for

reveal(p, contract funding)

else

waitForFunding(p)

for each swaption s that $\beta(s)$ is p **do** setStage$(s, p, funding)$

for each swaption s that $\sigma(s)$ is p **do**

waitUntill($getStage\mathcal{O}$(s, p) is not $funding$)

setStage$(s, p, funding)$

end for

waitUntill(the contract funding key is not revealed)

reveal(p, contract funding)

end if

Algorithm 3 phase 3

$\mathcal{G}_t$=(V_t, E_t) ▷ $\mathcal{G}_t$=(V_t, E_t) is a subgraph of $\mathcal{G}_p$ where $V_t = V_g$ and $E_t = \{e \in E_g | \rho(e) \neq t\}$

function SWAPTIONTOVERTEX(swaption s, party p)

set v_1, v_2 the vertices of s in the $\mathcal{G}_p$.

if $\ell(v_1)$ is p **return** v_1 **else if** $\ell(v_2)$ is p **return** v_2

end function

function VERTEXTOSWAPTION(vertex v)

return the swaption s in which v is located in the $\mathcal{G}_p$.

end function

function WAITFORPS(vertex v, edge class $[]c$)

for each $d \in E$ that $\rho(d) \in c$ and $\tau(d)$ is v **while** getStage(vertexToSwaption($h(d)$), $\ell(\mathcal{O}(v))$) is not *principal* wait

end function

if p is p_{leader} **then**

for each swaption s that swaptionToVertex(s, p) is a source of $\mathcal{G}_t$ **do**

setStage($s, p, principal$)

end for

for each swaption w where s is not source of $\mathcal{G}_t$ and w is in the association set of p **do**

v = swaptionToVertex(w, p)

waitForPS($v, [x]$)

setStage($w, p, principal$)

end for

for each swaption s in the association set of p, **while** getStage$\mathcal{O}$(s, p) is not *principal* **do** wait

reveal(p, leader)

else

for each swaption s in the association set of p **do**

v = swaptionToVertex(w, p)

if p is p_{master} **then**
waitForPS($v, [x]$)
else
waitForPS($v, [i, x, t]$)
end if
setStage($w, p, principal$)
end for
while the leader key is not revealed **do**
wait
end while
reveal(p, leader)
end if

Algorithm 4 phase 4

if p is p_{master} **then**
reveal(p, master)
else
while the master key is not revealed **do**
wait
end while
reveal(p, master)
end if

2. **Principal deposition**: Parties deposit their principals according to the order of locktimes set in the contract initiation. Then, when principals are deposited in all vertices $v \in V_g$ that $\mathcal{O}(v) = p_{leader}$, p_{leader} reveals leader key. By earlier revealing the key, p_{leader} is the only vulnerable party to enter the underwater state. Algorithm 3.

3. **Option exercise**: Now, it is time for p_{master} to use her option for exer-

cising the entire tangle. Algorithm 4.

4. **Option delegation**: If in previous section, p_{master} avoids to reveal the master key, all of the delegators are incentivised to reveal their key which results in executing the tangle to prevent her from cheating in the last section.

After all, we need to prove that executing above protocol on T keeps all the involving parties in a secure state. In each swaption, no party can cheat on the other, because of the rules discussed in previous sections. On the other hand, in the tangle setting the new contingency that has to be addressed is the suspicion that a party secretly knows some key that he is not supposed to know.

To analyse the parties security we categorize them into two subsets $\Psi, \bar{\Psi} \subseteq \mathcal{P}$ that $\bar{\Psi} = \mathcal{P} - \Psi$. For each $p_i \in \Psi$ there is a vertex v in V_p that $\ell(v) = p_i$ and there is no edge e that $\rho(e)$ is x and $\tau(e)$ is v.

As it was mentioned before, in a concurrent tangle, the person who has paid all the premiums is p_{master}. All the other parties in the tangle, if the process passes the contract initiation phase, have gained an amount of premium from p_{master}. The parties in $\bar{\Psi}$, will not be in the underwater state in any possible situation, since in all of their association set swaptions, their principal is loaned i.e. after passing the contract initiation phase, they enter the freeride state, because they get an amount of premium while paying nothing, and if the process does not pass this phase, their state will be kept in nodeal. For parties in Ψ, we analyse each stage of the execution as follows:

1. **Contract funding**: By following the order of $\mathcal{G}_c$, it is guaranteed that in each swaption, the locktime of revleaing contract funding key for swaption buyer is more than of swaption seller that prevents swaption buyer from cheating on the other party.

2. **Principal deposition**: By the definition of $\mathcal{G}_p$, for each vertex v that has no input edge e that $\rho(e)$ is x, there are some input edges $\tilde{e}$ that $\rho(\tilde{e})$ is t. In fact, $h(\tilde{e})$'s principal is going to be swapped with v's principal.

According to the protocol, $h(t)$ has to deposit before v. So, player $\ell(v)$ on vertex v is guaranteed not to be in the underwater state.

On the other hand, for v that $\ell(v) = p_{leader}$, since the protocol disregarded v's input edges e that $\rho(e) = t$, the principal in $h(e)$ may never be deposited while the principal in v is already deposited. In this case, p_{leader} can simply avoid exposing the leader key.

3. **Leader key**: According to the lemma 1, for all the sink vertices s in $\mathcal{G}_k$ graph, $\ell(\mathcal{O}(s))$ is p_{leader}. Since $\mathcal{G}_s$ is the subgraph that was used to set locktimes of principal deposition, for all the sink nodes $\tilde{s}$ in $\mathcal{G}_s$, $\ell(\mathcal{O}(\tilde{s}))$ is also p_{leader}. In the protocol, p_{leader} reveals leader key only when principals are deposited in all vertices v that $\mathcal{O}(v) = p_{leader}$. Thus, when leader key gets revealed, all the principals have been already deposited. Hence, all swaptions are in the same stage.

4. **Option exercise**: In this phase also the locktimes are set such that in each swaption, the locktime of swaption buyer is more than the swaption seller except for the vertices v that $\ell(v)$ is in $\mathcal{D}$. Thus, if the master key is revealed, the parties p that $p \in \mathcal{P} - \mathcal{D}$, will enter the deal state.

5. **Option delegation**: For parties $p \in \mathcal{D}$, since there is a vertex v that $\ell(v) = p$ and v's locktime is larger than $\mathcal{O}(v)$ and $\ell(\mathcal{O}(v))$ is the party who is supposed to reveal the master key, $\mathcal{O}(v)$ can let his own locktime expire and then exercise v's option contract. In this case, p will exercise his own option in the delegation stage, taking $\mathcal{O}(v)$'s principal. Hence, p will not be in underwater state. For each party $p_i \in \Psi$ and $p_j \in \bar{\Psi}$ the state of p_i and p_j will be held on deal and freeride respectively.

So far, we designed a protocol to satisfy the requirements needed in theorem 1. Now, we need to extend our protocol to fit the case where the tangle is not necessarily concurrent, e.g. see Fig. 16. So, we define the next theorem.

Theorem 2. *Given a tangle T and all of its strongly connected components in $\mathcal{G}_s$ graph $\{c_1, c_2, ..., c_n\}$, every c_i is cooperable tangle, if and only if there is a*

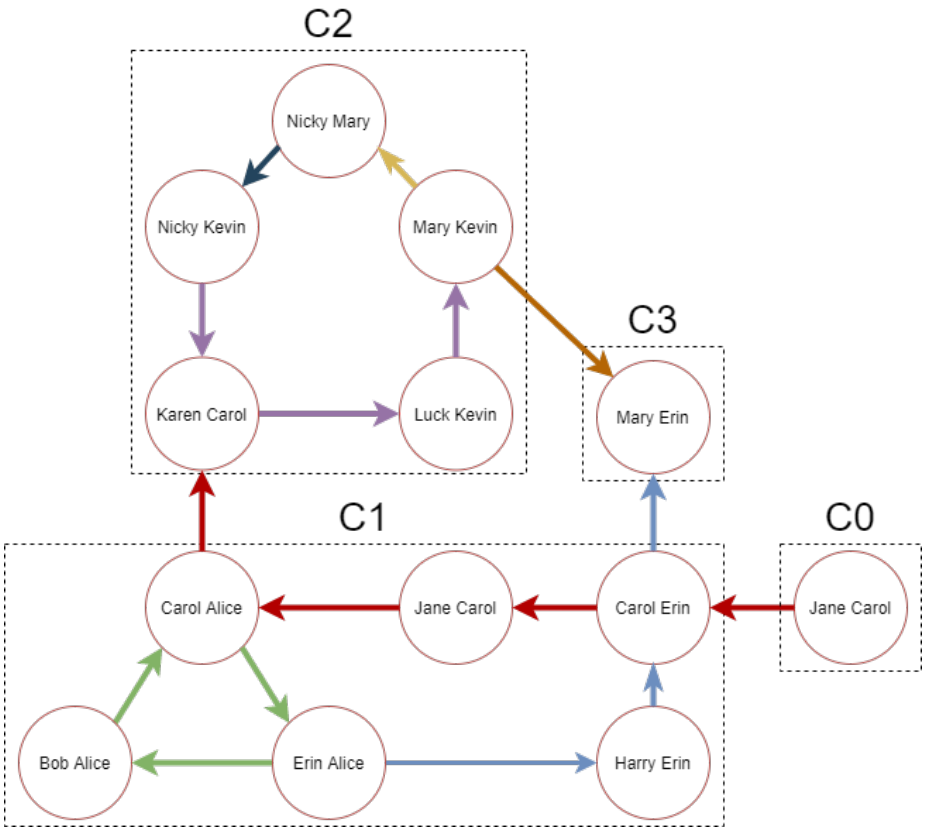


Figure 16: a tangle of four strongly connected components

protocol P that if gets executed on T then the procedure is secure for every party in T.

First, we prove that if every c_i is concurrent, then there is a secure protocol. Since there is no cycle between c_i's, there is a topological order between them. Consider the following algorithm.

Algorithm 5 Asynchronous Protocol

Let $\{c'_0, c'_1, .., c'_n\}$ is the topological order of $\mathcal{G}_s$ components.

for i **from** 0 **to** n **do**

    Execute the synchronous protocol on c'_i

end for

Since each c'_i is an cooperable tangle T, executing the synchronous protocol on each c'_i is secure due to the result of theorem 1. Hence, the entire execution process is secure. To prove the second direction of the theorem, we start by proving the following lemma:

Lemma 2. *If there is a secure protocol for tangle T, then for each key, there is exactly one party who can own it.*

We prove the lemma by contradiction. Assume in a concurrent tangle T there are two parties p_i and p_j who are the key holder of the same lock. Then we prove that there is at least one party who may enter into the underwater state. Given $\mathcal{G}_s = (V_g, E_g)$ assume $V_i \subset V$ and $V_j \subset V$ that for each $v \in V_i$, p_i is the key holder and for each $v \in V_j$, p_j is the key holder of the same stage. $\mathcal{G}_s$ is a strongly connected graph and for each $e \in E$ the same principal is overlapped between vertices $h(e)$ and $\tau(e)$, so the same lock is allowable on principals on both sides of e, hence $V_i \cap V_j$ can not be $\emptyset$. Therefore, there is a swaption $s \in V_i \cap V_j$ that principals in $\beta(s)$ side and $\sigma(s)$ side are locked by different keys. Hence, if the key of one of these principals is revealed, its depositor will go underwater.

Now assume that there is a c_i that is not cooperable tangle which means at least one of the following statements is true.

1. c_i is not protected. Now, assume subgraph $\mathcal{G}_t = (V_t, E_t)$ of $\mathcal{G}_p$ for c_i where $V_t = V$ and $E_t = \{e \in E | \rho(e) \neq t\}$. Then either $\mathcal{G}_t$ sources belong to different parties or the $\mathcal{G}_p$ has not any feedback edge set that satisfies the requirements of definition 3. For the first case, there are at least two leader key holder parties in the graph. By the result of the lemma 2 there might be parties who go underwater. In the second case, for each cycle $\mathcal{C}$ in $\mathcal{G}_t$ either for all edges $e \in \mathcal{C}$, $\rho(e)$ is x, thus the money transferring in $\mathcal{C}$ has to be the source of itself which is not acceptable, or for all edges $e \in \mathcal{C}$ that $\rho(e) \neq x$, $\tau(e)$ is not p_{leader}. If there is a protocol which executes c_i, to bypass this cycle, it must deviate from the principal deposition order imposed by edge $e \in \mathcal{C}$ that $\rho(e) \in \{i, t\}$. Hence the principal of the vertex

$\tau(e)$ is forced to be deposited before the principal deposition of $h(e)$ while $\ell(\tau(e))$ is not p_{leader}. Then the party $\ell(h(e))$ can rob $\ell(\tau(e))$'s principal by cooperating with the graph key holders by early exposing all the keys.

2. There is no party eligible to be the p_{master} of c_i. Then, if there is any protocol executing on T, then at least two locks are used for option funding stage. Hence, there is at least two key holders for the master key lock. Thus, there might be a party who goes underwater.

5. Conclusion

We proposed components that allow using HTLCs in order to have late-deposition and margin-free swaptions. The later one was claimed to be impossible in [4]. Then we showed that these components in concurrency with other components can build futures arbitrages. Building futures arbitrages and money markets became possible using techniques like overlapping and option delegation introduced in this paper. Finally, for a larger money market, comprised of many swaptions in relation with each other, we proved two theorems that describe the general rules of these money markets. We also proposed protocols without which money markets can not operate. The multi-leader problem in [19] is also solved by the option delegation technique.

6. Declaration of Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, Tech. rep., Manubot (2019).

[2] Chancellor Alistair Darling on brink of second bailout for banks, https://www.thetimes.co.uk/article/chancellor-alistair-darling-on-brink-of-second-bailout-for-banks-n9l382mn62h, accessed: 2020.09.14.

[3] J. Poon, T. Dryja, The bitcoin lightning network: Scalable off-chain instant payments (2016).

[4] J. A. Liu, Atomic swaptions: cryptocurrency derivatives, arXiv preprint arXiv:1807.08644 (2018).

[5] M. Tefagh, F. Bagheri, A. Khajehpour, M. Abdi, Atomic bonded cross-chain debt, in: 2020 the 3rd International Conference on Blockchain Technology and Applications, ICBTA 2020, Association for Computing Machinery, New York, NY, USA, 2020, p. 50–54. doi:10.1145/3446983.3446987. URL https://doi.org/10.1145/3446983.3446987

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

[7] A better money, https://makerdao.com, accessed: 2020.08.22.

[8] Compound is an algorithmic, autonomous interest rate protocol built for developers, to unlock a universe of open financial applications., https://compound.finance, accessed: 2020.08.22.

[9] The most powerful open trading platform for crypto assets, https://dydx.exchange, accessed: 2020.08.22.

[10] The money market protocol, https://aave.com, accessed: 2020.08.22.

[11] Wrapped bitcoin delivers the power of bitcoin with the flexibility of an ERC20 token, https://wbtc.network, accessed: 2020.08.22.

[12] A. Kiayias, A. Russell, B. David, R. Oliynykov, Ouroboros: A provably secure proof-of-stake blockchain protocol, in: Annual International Cryptology Conference, Springer, 2017, pp. 357–388.

[13] J. Kwon, E. Buchman, A network of distributed ledgers, Cosmos, dated (2018) 1–41.

[14] N. Van Saberhagen, Cryptonote v 2.0 (2013).

[15] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, M. Virza, Zerocash: Decentralized anonymous payments from bitcoin, in: 2014 IEEE Symposium on Security and Privacy, IEEE, 2014, pp. 459–474.

[16] D. Schwartz, N. Youngs, A. Britto, et al., The ripple protocol consensus algorithm, Ripple Labs Inc White Paper 5 (8) (2014).

[17] D. Mazieres, The stellar consensus protocol: A federated model for internet-level consensus, Stellar Development Foundation 32 (2015).

[18] T. Nolan, Alt chains and atomic transfers. bitcointalk. (May 2013). URL `https://bitcointalk.org/index.php?topic=193281.0`

[19] M. Herlihy, Atomic cross-chain swaps, in: Proceedings of the 2018 ACM symposium on principles of distributed computing, 2018, pp. 245–254.

[20] J. Xu, D. Ackerer, A. Dubovitskaya, A game-theoretic analysis of cross-chain atomic swaps with HTLCs, arXiv preprint arXiv:2011.11325 (2020).

[21] J.-Y. Zie, J.-C. Deneuville, J. Briffaut, B. Nguyen, Extending atomic cross-chain swaps, in: C. Pérez-Solà, G. Navarro-Arribas, A. Biryukov, J. Garcia-Alfaro (Eds.), Data Privacy Management, Cryptocurrencies and Blockchain Technology, Springer International Publishing, Cham, 2019, pp. 219–229.

[22] R. van der Meyden, On the specification and verification of atomic swap smart contracts, in: 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), IEEE, 2019, pp. 176–179.

[23] Y. Hirai, Blockchains as kripke models: An analysis of atomic cross-chain swap, in: T. Margaria, B. Steffen (Eds.), Leveraging Applications of Formal Methods, Verification and Validation. Industrial Practice, Springer International Publishing, Cham, 2018, pp. 389–404.

[24] R. Han, H. Lin, J. Yu, On the optionality and fairness of atomic swaps, in: Proceedings of the 1st ACM Conference on Advances in Financial Technologies, 2019, pp. 62–75.

[25] I. Tsabary, M. Yechieli, I. Eyal, MAD-HTLC: Because HTLC is crazy-cheap to attack, ArXiv abs/2006.12031 (2020).

[26] AtomicDEX and atomic swaps, https://developers.komodoplatform.com/basic-docs/start-here/core-technology-discussions/atomicdex.html#atomic-swaps, accessed: 2020.11.27.

[27] H. Adams, N. Zinsmeister, D. Robinson, Uniswap v2 core, URl: https://uniswap. org/whitepaper. pdf (2020).

[28] https://github.com/burgerswap-org/burgerswap-core, accessed: 2020.09.14.

[29] Masterchef is ready, https://sushisvapclassic.org, accessed: 2020.09.14.

[30] Fast and simple way to exchange ethereum tokens (May 2019).
URL https://www.kyberswap.com

[31] Market making for jellyswap protocol via butler, https://jellyswap.gitbook.io/liquidity/, accessed: 2020.09.14.

[32] Multicurrency hd wallet with built-in hybrid atomic swap exchange, https://atomex.me, accessed: 2020.09.14.

[33] Cross-chain crypto trading, https://liquality.io, accessed: 2020.09.14.

[34] M. Black, T. Liu, T. Cai, Atomic loans: Cryptocurrency debt instruments, arXiv preprint arXiv:1901.05117 (2019).

[35] D. Chandra, M. Eric, ZK-Swaps: Fast, liquid, private atomic swaps, Portal, 2019.
URL https://books.google.it/books?id=0si4DwAAQBAJ&pg=PT7&lpg=PT7&dq=zk-swap&source=bl&ots=v9MholZ8EH&sig=ACfU3U0TdEied2xxEkE--GSjl1x5beMARg&hl=en&sa=X&ved=2ahUKEwiNwrmmtrzsAhVPOBoKHS3gCNQQ6AEwA3oECAoQAg#v=onepage&q=zk-swap&f=false

[36] Transaction signature verification for version 0 witness program, https://github.com/bitcoin/bips/blob/master/bip-0143.mediawiki.

[37] SIGHASH-NOINPUT, https://github.com/bitcoin/bips/blob/master/bip-0118.mediawiki.

[38] J. Bang-Jensen, G. Z. Gutin, Digraphs: theory, algorithms and applications, Springer Science & Business Media, 2008.

[39] J. A. Bondy, U. S. Murty, Graph theory (2000).