On the optionality and fairness of Atomic Swaps

Runchao Han
Monash University and CSIRO-Data61
runchao.han@monash.edu

Haoyu Lin
chris.haoyul@gmail.com

Jiangshan Yu
Monash University
jiangshan.yu@monash.edu

ABSTRACT

Atomic Swap enables two parties to atomically exchange their own cryptocurrencies without trusted third parties. In this paper, we investigate the (un)fairness of the Atomic Swap protocol. First, we model the Atomic Swap as the American Call Option, and prove an Atomic Swap is equivalent to an American Call Option without the premium, thus is unfair to the swap participant. Second, we quantify the unfairness of Atomic Swap and compare it with that of conventional financial assets (stocks and fiat currencies). The quantification results show that the Atomic Swaps are much more unfair on cryptocurrencies than on stocks and fiat currencies in the same setting. Third, we use the conventional Cox-Ross-Rubinstein option pricing model in Finance to estimate the premium, and show that the estimated premium for cryptocurrencies is $2\% \sim 3\%$ of the asset value, while the premium for stocks and fiat currencies is approximately 0.3%. Last, we propose two fair Atomic Swap protocols, one is for currency exchange and the other is for American Call Options. Our protocols are based on the original Atomic Swap protocol, but implement the premium mechanism. Blockchains supporting smart contracts such as Ethereum support our protocols directly. Blockchains only supporting scripts such as Bitcoin can support our protocols by adding a simple opcode. In addition, we give the reference implementation of our protocols in Solidity, and give detailed instructions on implementing our protocols with Bitcoin script.

KEYWORDS

Blockchain; Atomic Swap; Cross-Chain Transactions; American Call Option

1 INTRODUCTION

The Atomic Swap protocols allow two parties to exchange their assets "atomically" without trusted third parties. "Atomic" means the swap either succeeds or fails for both parties at any given time. In blockchains, Atomic Swap is usually implemented by using Hashed Time-locked Contracts (HTLCs) [1]. The HTLC is a type of transaction that, the payee should provide the preimage of a hash value before a specified deadline, otherwise the payment fails - the money goes back to the payer and the payee will not get any money.

Atomic Swap has already been widely adopted in the cryptocurrency industry. In particular, it realised the Decentralised Exchanges (DEXes), and boosted the Decentralised Finance (De-Fi). In a DEX, traders publish and participate in deals, and the dealing process is regulated by a smart contract embedding the Atomic Swap protocol. Different from centralised exchanges, DEXes are non-custodial - traders does not need to deposit their money in DEXes. The non-custody property avoids DEXes from money thefts, which commonly happened on centralised exchanges. To date, the DEX market volume has reached approximately 50,000 ETH [2]. More specifically, there are more than 250 DEXes [3], more than 30 DEX protocols [4], and more than 4,000 active traders in all DEXes [2].

However, being atomic does not indicate the Atomic Swap is fair. In an Atomic Swap, the swap initiator can decide whether to proceed or abort the swap, and the default maximum time for him to decide is 24 hours [5]. This enables the the swap initiator to speculate without any penalty. More specifically, the swap initiator can keep waiting before the timelock expires. If the price of the swap participant's asset rises, the swap initiator will proceed the swap so that he will profit. If the price of the swap participant's asset drops, the swap initiator can abort the swap, so that he won't lose money.

A person entitled ZmnSCPxj initiated a discussion at the Lightning-dev mailing list [6] that, this problem is equivalent to the Optionality in Finance, which has already been studied for decades [7]. In Finance, an investment is said to have Optionality if 1) settling this investment happens in the future rather than instantly; 2) settling this investment is optional rather than mandatory. For an investment with Optionality, the option itself has value besides the underlying asset, which is called the *premium*. The option buyer should pay for the premium besides the underlying asset, even if he aborts the contract. In this way, he can no longer speculate without penalties.

In Atomic Swap, the swap initiator has the Optionality, as he can choose whether to proceed or abort the swap. Unfortunately, the swap initiator is not required to pay for the premium - the Atomic Swap does not take the Optionality into account. Furthermore, Atomic Swap should not have Optionality. Atomic Swap is designed for currency exchange, and the currency exchange has no Optionality. Instead, once both parties agree on a currency exchange, it should be settled without any chance to regret.

In this paper, we investigate the unfairness of Atomic Swap. We start from describing the Atomic Swap and the American Call Option in Finance, then we show how an Atomic Swap is equivalent to a premium-free American Call Option. After that, we then evaluate how unfair the Atomic Swap is from two different perspectives: quantifying the unfairness and estimating the premium. Furthermore, we propose an improvement on the Atomic Swap, which implements the premium mechanism, to make it fair. Our improvement supports blockchains with smart contracts (e.g. Ethereum) directly, and can support blockchains with scripts only (e.g. Bitcoin) by adding a single opcode. We also implement our protocol in Solidity (a smart contract programming language for Ethereum), and give detailed instructions on implementing our protocols on Bitcoin.

1.1 Our contributions

Our contributions are as follows:

We show that the Atomic Swap is equivalent to the premium-free American Call Option. We describe the Atomic Swap and the American Call Option, then we point out that an Atomic Swap is equivalent to a premium-free American Call Option, which is a type of Options

(in Finance). More specifically: the initiator and the participant in an Atomic Swap are the option buyer and the option seller in an American Call Option, respectively; the initiator asset and the participant asset in an Atomic Swap are the used currency and the underlying asset in an American Call Option, respectively; the participant asset's timelock in an Atomic Swap is the strike time in an American Call Option; the current price of the participant asset in an Atomic Swap is the strike price in an American Call Option; redeeming cryptocurrencies in an Atomic Swap is equivalent to exercising the contract in an American Call Option.

We show that the Atomic Swap is unfair to the participant. We show that the Atomic Swap - represented as the premium-free American Call Option in Finance - is unfair to the participant, especially in the highly volatile cryptocurrency market. In practice, the initiator can decide whether to proceed the swap while investigating the cryptocurrency market. However, proceeding or aborting the swap does not require the initiator to pay for the premium. This leads to the scenario that, if the participant's asset price rises before the strike time, he will proceed the swap to profit, otherwise he will abort the swap to avoid losing money. In this way, the swap initiator can speculate without any risk in Atomic Swaps.

We quantify the unfairness of Atomic Swap, and compare it with that of conventional financial assets. We quantify how unfair the Atomic Swap with mainstream cryptocurrency pairs is, and compare this unfairness with those of conventional financial assets (stocks and fiat currencies). We first classify the unfairness to two parts, namely the profit when the price rises and the mitigated loss when the price drops, then quantify them based on historical exchange rate volatility. Our results show that, in the default timelock setting, the profit and the mitigated loss of our selected cryptocurrency pairs are approximately 1%, while for stocks and fiat currencies the values are approximately 0.3% and 0.15%, respectively.

We use the Cox-Ross-Rubinstein option pricing model to estimate the premium. We use the Cox-Ross-Rubinstein option pricing model to estimate how much the premium should be for Atomic Swaps. In Finance, the Cox-Ross-Rubinstein model [8] is the conventional option pricing model for American-style options. Our results show that, in the default timelock setting, the premium should be approximately 2% for Atomic Swaps with cryptocurrency pairs, while the premium is approximately 0.3% for American Call Options with stocks and fiat currencies. Also, the premium values rise for all assets with the strike time increasing, then start to converge when the strike time reaches 300 days.

We propose an improvement on the Atomic Swap to make it fair. With the observation that the unfairness is from the premium, we propose an improvement on the Atomic Swap, which implements the premium mechanism, to make it fair. It supports both the currency exchange-style Atomic Swap and the American Call Option-style Atomic Swap. In the currency exchange-style Atomic Swap, the premium will go back to the swap initiator if the swap is successful. In the American Call Option-style Atomic Swap, the premium will definitely go to the swap participant if the participant participates in the swap.

We describe how to implement our protocol on existing blockchains. We give instructions to implement our protocols on existing blockchains, including blockchains supporting smart contracts and blockchains supporting scripts only. For blockchains supporting smart contracts (e.g. Ethereum), our protocol can be directly implemented. For blockchains supporting scripts only (e.g. Bitcoin), our protocol can be implemented by adding one more opcode. We call the opcode "OP_LOOKUP_OUTPUT", which looks up the owner of a specific UTXO output. We give the reference implementation in Solidity as an example of smart contracts. We also give that in Bitcoin script (which assumes "OP_LOOKUP_OUTPUT" exists) as an example of scripts.

1.2 Paper structure

The paper is structured as follows. Section 2 describes the background of Atomic Swap and options in Finance. Section 3 describes the Atomic Swap and the American Call Option, and shows how the Atomic Swap protocol is equivalent to the premium-free American Call Option. Section 4 evaluates the Atomic Swap unfairness by analysing the volatility and pricing the premium of mainstream cryptocurrency pairs. Section 5 describes our proposed fair Atomic Swap protocols. Section 6 describes how to implement our proposed protocols on existing blockchains. Section 7 discusses security issues of Atomic Swaps, other countermeasures for solving the Atomic Swap unfairness, and limitations of our protocols. Section 9 concludes our paper and outlines the future work.

2 BACKGROUND

In this section, we explain basic concepts of the Atomic Swap and the Option (in Finance).

2.1 Atomic Swap

An Atomic Swap [5] is that two parties exchange their assets "atomically". "Atomic" means the swap is indivisible: it either succeeds or fails for both parties.

In Blockchain, the Hashed Time-locked Contract (HTLC) [1] enables the Atomic Swap without trusted third parties. HTLC was originally introduced to secure routing across multiple payment channels [9]. In a HTLC-style transaction, the payee can redeem the payment prior to a deadline only by providing the preimage of a specific hash value, otherwise the payment will expire and the money will go back to the payer. This is achieved by the hashlock - to lock the payment by a hash value, and the timelock - to give the deadline of redeeming. The timelock avoids locking money in a payment forever when the payee cannot provide the preimage.

2.2 Option in Finance

In Finance, an option is a contract which gives the option buyer the right to buy or sell an asset, at a specified price prior to or on a specified date [7]. Here the option buyer can choose whether to fulfill the contract. The specified price is called the *strike price*; the specified date is called the *strike time*; the party proposing the option is called the *option seller*; the other party choosing to fulfill or abort the contract is called the *option buyer*; the asset is called the *underlying asset*; and fulfilling the contract is called *exercising*.

The option has two types: the American-style Option and the European-style Option. They differ from the *strike time*: The European-style Option buyer can only exercise the contract on the strike time, and the American-style Option buyer can exercise the contract no later than the strike time.

Who holds the option is irrelevant with who is buying the underlying asset. More specifically, the option buyer is who can decide to exercise or abort the contract. Whether the option buyer is buying or selling the underlying asset depends on the option contract. In Finance, if the option buyer is the party buying the underlying asset, this option is a "Call Option", otherwise this option is a "Put Option" [10].

Besides the underlying asset, the option contract itself is considered to have value. The value of the contract is called the *premium*. The option buyer should pay for the *premium* to the option seller once both parties agree on the option contract.

The *premium* is priced prior to the contract agreement. As the *premium* is the only variable within the option contract, pricing the *premium* is also known as the *option pricing* problem. *Option Pricing* is rather a complex task, and is still a hot research topic in Finance and Applied Mathematics.

The Black-Scholes (BS) Model is the first widely used model for option pricing [11]. It can estimate the value of European-style Options using the historical price of the underlying asset. The Cox-Ross-Rubinstein (CRR) model [8], also known as the Binomial Option Pricing model, extends the BS model for pricing American-style Options.

3 ATOMIC SWAP AND AMERICAN CALL OPTION

In this section, we describe the Atomic Swap protocol (the original version on Bitcointalk [5]) and the American Call Option, then point out that an Atomic Swap is equivalent to an American Call Option without the premium.

3.1 Atomic Swap

3.1.1 Security assumptions. First, we assume blockchains involved in the Atomic Swap are secure, and execute all transactions correctly. The Atomic Swap is based on blockchains. If the blockchains are insecure, the Atomic Swap will also be insecure.

Second, we assume the HTLC mechanism in blockchains is reliable. More specifically, 1) blockchains produce new blocks with stable speeds; 2) the hash algorithms used by HTLCs are secure; 3) blockchains execute HTLCs correctly.

Third, the time for confirming a transaction is negligible compared to timelocks in HTLCs. In practice, the swap initiator's timelock is 48 hours and the swap participant's timelock is 24 hours by default [5], while confirming a transaction is less than 1 hour for most blockchains.

3.1.2 Process. Assuming the swap initiator Alice hopes to get x_2 $Coin_2$ from the swap participant Bob in exchange of x_1 $Coin_1$. $Coin_1$ is the cryptocurrency on the blockchain BC_1, and $Coin_2$ is the cryptocurrency on the blockchain BC_2. We denote the Atomic Swap as

$$\mathcal{AS} = (x_1, Coin_1, x_2, Coin_2)$$

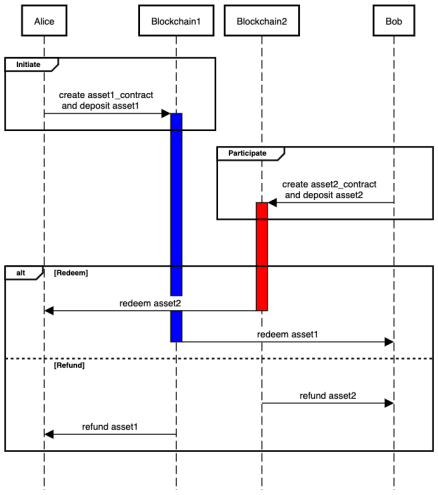

Figure 1: Sequence diagram of Atomic Swap.

Let Alice be the holder of the address $\beta_{A,1}$ on BC_1 and the address $\beta_{A,2}$ on BC_2. Let Bob be the holder of the address $\beta_{B,1}$ on BC_1 and the address $\beta_{B,2}$ on BC_2. $\beta_{A,1}$ holds $Coin_1$ with the amount no smaller than x_1, and $\beta_{B,2}$ holds $Coin_2$ with the amount no smaller than x_2.

Figure 1 shows the process of $\mathcal{AS}$. In detail, $\mathcal{AS}$ consists of four stages: **Initiate**, **Participate**, **Redeem**, and **Refund**.

Initiate. Alice initiates $\mathcal{AS}$ at this stage. First, Alice picks a random secret s only known to herself, and computes the hash $h = \mathcal{H}(s)$ of s, where $\mathcal{H}$ is a secure hash function. Then, Alice creates an HTLC script C_1 that "Alice pays x_1 $Coin_1$ from $\beta_{A,1}$ to $\beta_{B,1}$ if Bob can provide s which makes $\mathcal{H}(s) = h$ before or on a timelock δ_1 (which is a timestamp). After δ_1, Alice can refund the money - get x_1 $Coin_1$ back." After creating C_1, Alice publishes C_1 as a transaction $tx_{C,1}$ on BC_1. Note that h is published when publishing $tx_{C,1}$. Besides C_1, Alice also creates a refund script $\mathcal{R}_1$ that "Alice pays x_1 $Coin_1$ from $\beta_{A,1}$ to her another address." This is to ensure x_1 $Coin_1$ can no longer be redeemed by others. Alice can publish $\mathcal{R}_1$ only after δ_1. If Bob does not redeem x_1 $Coin_1$ and δ_1 expires, Alice can refund x_1 $Coin_1$ by publishing $\mathcal{R}_1$ as a transaction $tx_{\mathcal{R},1}$ on BC_1.

Participate. Bob participates in $\mathcal{AS}$ after **Initiate**. With the published h in $tx_{C,1}$, Bob creates another HTLC script C_2 that "Bob pays x_2 $Coin_2$ from $\beta_{B,2}$ to $\beta_{A,2}$ if Alice can provide s before or on a timelock δ_2 (which is a timestamp). After the time of δ_2, Bob can refund the money - get x_2 $Coin_2$ back." Here δ_2 should expire before δ_1. After creating C_2, Bob publishes C_2 as a transaction $tx_{C,2}$ on BC_2. Note that Alice knows s so she can redeem x_2 $Coin_2$ in $tx_{C,2}$ anytime before δ_2, but Bob cannot redeem x_1 $Coin_1$ in $tx_{C,1}$ because he does not know s. Besides C_2, Bob also creates a refund script $\mathcal{R}_2$ that "Bob pays x_2 $Coin_2$ from $\beta_{B,2}$ to his another address." This is to ensure x_2 $Coin_2$ can no longer be redeemed by Alice. Bob can do this only after δ_2. If Alice does not redeem x_2 $Coin_2$ before δ_2 expires, Bob can refund x_2 $Coin_2$ by publishing $\mathcal{R}_2$ as a transaction $tx_{\mathcal{R},2}$ on BC_2.

***Redeem** or **Refund**.* At this stage, Alice can choose either to redeem x_2 $Coin_2$ or refund x_1 $Coin_1$. Note that both **Redeem** and **Refund** are atomic: if Alice chooses to redeem x_2 $Coin_2$, Bob can

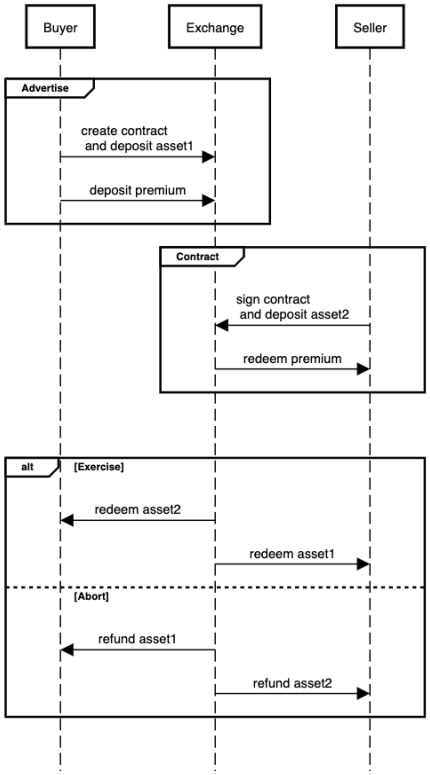

Figure 2: Sequence diagram of the American Call Option.

also redeem x_1 $Coin_1$; if Alice chooses to refund x_1 $Coin_1$, Bob can also refund x_2 $Coin_2$.

Redeem. Alice redeems x_2 $Coin_2$ by publishing s, then Bob can also redeem x_1 $Coin_1$ with the published s. First, Alice provides s to $tx_{C,2}$ in order to redeem x_2 $Coin_2$ in $tx_{C,2}$. As a result, Alice redeems x_2 $Coin_2$, but exposes s to Bob. After that, Bob provides s to $tx_{C,1}$ in order to redeem x_1 $Coin_1$ in $tx_{C,1}$. In this way, Alice and Bob successfully exchanges x_1 $Coin_1$ and x_2 $Coin_2$.

Refund. If Alice does not redeem x_2 $Coin_2$ after δ_2 expires, Bob can refund his x_2 $Coin_2$ by publishing $tx_{\mathcal{R},2}$. As a result, Alice cannot redeem x_2 $Coin_2$, and will not publish s. After δ_1, Alice can also refund her x_1 $Coin_1$ by publishing $tx_{\mathcal{R},1}$.

Atomicity analysis. We can see that $\mathcal{AS}$ either succeeds or fails for both Alice and Bob. In detail,

- If Alice misbehaves when triggering **Initiate**, Bob will lose nothing as he hasn't deposited x_2 $Coin_2$ yet.
- If Bob misbehaves when triggering **Participate**, Alice can choose to abort $\mathcal{AS}$ by triggering **Refund**.
- Alice can only choose to redeem x_2 $Coin_2$ by triggering **Redeem** or wait δ_2 to expire. Once Alice triggers **Redeem**, Bob can also trigger **Redeem**. Once δ_2 expires, Bob can trigger **Refund** to get his x_2 $Coin_2$ back.

However, one may take both x_1 $Coin_1$ and x_2 $Coin_2$ if the other does not trigger **Redeem** or **Refund** on time. For example, if Bob does not trigger **Redeem** after Alice triggers **Redeem** and δ_1 expires, Alice can also refund x_1 $Coin_1$ by triggering **Refund**. It is Bob to blame in this case, because he should have had enough time - at least $\delta_2 - \delta_1$ (48 - 24 = 24 hours by default) - to redeem x_1 $Coin_1$. Another example is that Alice broadcasts $tx_{C,1}$ after δ_2, but Bob has already triggered **Refund**. Therefore, Bob can also redeem x_1 $Coin_1$ with s in $tx_{C,1}$ before δ_1. Similarly, it is Alice to blame in this case, because she should have had enough time - before δ_2 (24 hours by default) - to trigger **Redeem**.

3.2 American Call Option

The American Call Option is a contract that "one can buy an amount of an asset with an agreed price prior to or on an agreed time in the future". Here, the agreed price is usually called the *strike price*; and the contract settlement is called *exercising*; the one who proposes the contract and buys the asset is called the *option buyer*; the one who sells the asset is called the *option seller*.

As mentioned in Section 2.2, the option contract itself has value, called the *premium*. In an American Call Option, the option buyer should pay for the premium when the contract is agreed by both parties, and should pay for the asset when the contract is exercised.

We denote an American Call Option contract Π as

$$\Pi = (\pi_1, \pi_2, K, A, T, C)$$

where the option buyer Alice with π_1 hopes to buy π_2 from the option seller Bob; π_1 and π_2 are Alice's currency and Bob's asset, respectively; K is the strike price with the unit π_2/π_1 - the price of π_2 measured in π_1; A is the amount of the asset π_2 that Bob wants to sell; T is the agreed strike time; C is the *premium* with the unit π_1.

The process of an American Call Option is as follows:

(1) **Advertise**: Alice creates and advertises an American Call Option contract $\Pi = (\pi_1, \pi_2, K, A, T, C)$.
(2) **Contract**: If Bob thinks Π is profitable and Alice does not abort Π, Bob will participate in Π. When Bob participates, Alice should pay C to Bob first. Note that Alice does not pay for A π_2 at this stage. Also note that Bob cannot abort Π after participating in Π.
(3) **Exercise** or **Abort**: Alice exercises Π - pays AK π_1 to Bob - no later than T, and Bob gives A π_2 to Alice. If Alice does not exercise Π no later than T, Π will abort - Alice gets π_1 back and Bob gets π_2 back. In other words, both of them get their underlying asset back, but Alice loses the premium C to Bob when **Contract**.

3.3 An Atomic Swap is a premium-free American Call Option

We show that an Atomic Swap is equivalent to a premium-free American Call Option. More specifically, $\mathcal{AS} = (x_1, Coin_1, x_2, Coin_2)$ is equivalent to the American Call Option contract

$$\Pi = (Coin_1, Coin_2, \frac{x_2}{x_1}, x_2, \delta_2, 0)$$

where: **Advertise** in the American Call Option is equivalent to **Initiate** in the Atomic Swap; **Contract** in the American Call Option is equivalent to **Participate** in the Atomic Swap; **Exercise** in the American Call Option is equivalent to **Redeem** in the Atomic Swap; **Abort** in the American Call Option is equivalent to **Refund** in the Atomic Swap.

In the American Call Option context, the option buyer Alice wants to buy x_2 $Coin_2$ from the option seller Bob by using x_1 $Coin_1$. $Coin_1$ is the currency Alice uses, $Coin_2$ is the asset Bob has. This is equivalent to that Alice with π_1 wants to buy π_2 from Bob. δ_2 is the timelock of the contract transaction on BC_2, which is equivalent to the strike time T in Π. In $\mathcal{AS}$ Bob can refund his asset back after δ_2 to abort $\mathcal{AS}$, while Π will be automatically aborted after the strike time T. Establishing $\mathcal{AS}$ does not require Alice to pay anything other than x_1 $Coin_1$ to Bob, which is equivalent to Π with $C = 0$.

Note that both the Atomic Swap and the American Call Option are "speculative": both the cryptocurrency exchange rates in Atomic Swaps and asset prices in the American Call Options are fluctuating overtime. Therefore, the "premium-free" property enables Alice to speculate without any risk: if Bob's asset price rises right before the strike time, she will proceed the swap to profit, otherwise she will abort the swap to avoid the loss. Therefore, without the premium, Alice is risk-free towards the market.

4 UNFAIRNESS OF ATOMIC SWAPS

In this section, we evaluate the unfairness of the Atomic Swap based on our observation in Section 3. In particular, our evaluations are from two perspectives: quantifying the unfairness and estimating the unpaid premium. Quantifying the unfairness is based on analysing the historical exchange rate volatility. Estimating the unpaid premium is based on the Cox-Ross-Rubinstein model - the conventional option pricing model for pricing American-style options in Finance. Furthermore, we also evaluate conventional financial assets - the stocks and the currency exchanges - and compare their results with cryptocurrencies.

4.1 Experimental setting

We collected relevant data of mainstream cryptocurrencies for one year, starting from May 3th, 2018 to May 3th, 2019. In particular, the cryptocurrency exchange rate data was retrieved from from CoinGecko [1]; the stock index data was retrieved from Yahoo Finance [2]; the currency exchange rate data was retrieved from Investing.com [3].

4.2 Quantifying the unfairness

Assume that Alice initiates the swap by triggering **Initiate**$(\cdot)$ at the time t, then by default $\delta_1 = t + 48$(hours). We also assume that Bob participates in the swap by triggering **Participate**$(\cdot)$, then by default $\delta_2 = t + 24$(hours).

In this way, Alice can decide whether to proceed the swap within $\delta_1 - \delta_2 = 24$(hours). When Bob's asset price rises, Alice profits directly. When Bob's asset price drops, Alice can abort the swap to avoid losing money. Based on this observation, we classify Alice's advantages to two parts, namely the profit when Bob's asset price rises and the mitigated loss when Bob's asset price drops.

We then test the unfairness by using a single Atomic Swap with the value of x USD, then we show the degree of unfairness in dollars based on the historical data. For each day, Alice may either profit α percent of x directly (when Bob's asset price rises), or mitigate β percent of x by aborting the swap (when Bob's asset price drops) on average. Assume the possibility for Bob's asset price to rise is P_α, and to drop is P_β. Then, the expected profit rate is $E_\alpha = \alpha P_\alpha$, and the expected mitigated risk rate is $E_\beta = \beta P_\beta$. Therefore, the expected unfairness is that Alice profits $E_\alpha x$ and mitigates the risk of losing $E_\beta x$. Also, as E_α and E_β are equally calculated, adding up them together $(E_\alpha + E_\beta)$ can derive the total unfairness.

Experimental methodology. In our scenario, quantifying the unfairness is to calculate E_α and E_β, so we calculate E_α and E_β for each selected cryptocurrency pair. Furthermore, we also quantify the unfairness of stock indices and fiat currencies in the same setting, in order to make comparisons. We use S&P500 and Dow Jones Index (DJI) as examples of stock indices, and USD-EUR and USD-GBP as examples of fiat currencies.

Results and analysis. Figure 3 shows the calculated E_α, E_β, the maximum daily rises max_α and the maximum daily drops max_β for 8 mainstream cryptocurrency pairs, stock indices (S&P500 and DJI) and fiat currency exchange rates (USD-EUR and USD-GBP). For each plot, points in the red Profit Area indicate that Alice profits directly at those days, and points in the green Risk Area indicate that Alice can abort the swap to mitigate the risk at those days.

We observe that for all chosen cryptocurrency pairs, max_α and max_β are considerably big - ranging from 8% to 25%. Meanwhile, max_α and max_β of stock indices are much smaller than all cryptocurrency pairs, and max_α and max_β of fiat currencies are even smaller than stock indices. This indicates that in the setting of an 24-hour Atomic Swap, the Atomic Swap with cryptocurrencies is much more unfair than with stocks, and the Atomic Swap with stocks are more unfair than with fiat currencies.

Figure 4 visualises E_α and E_β of all evaluated items in Figure 3. In particular, we classify scatters to 4 groups based on their E_α and E_β: The first group $(0 < E_\alpha < 0.005 \wedge 0 < E_\beta < 0.005)$ consists of all stock indices (S&P500 and DJI) and all fiat currency pairs (USD-GBP and USD-EUR); the second group $(0.005 < E_\alpha < 0.015 \wedge 0.005 < E_\beta < 0.015)$ consists of most cryptocurrency pairs; the third group $(0.010 < E_\alpha < 0.015 \wedge E_\beta > 0.015)$ only contains one cryptocurrency pair BTC-BNB; the fourth group $(E_\alpha > 0.015 \wedge 0.010 < E_\beta < 0.015)$ only contains the last cryptocurrency pair BTC-BCH. Moreover, we draw a line $E_\beta = E_\alpha$ to separate two areas: $E_\beta > E_\alpha$ and $E_\beta < E_\alpha$.

Obviously, the Atomic Swap with first-group items is fairer than with second-group items, and the Atomic Swap with second-group items is fair than with third-group and fourth-group items. More specifically, we can get the following results. First, the Atomic Swap with cryptocurrency pairs is more unfair than with stocks and fiat currency pairs. This result is consistent with results in Figure 3. Second, E_β and $E_\beta - E_\alpha$ of BTC-BNB are bigger than of others. This means the exchange rate of BTC-BNB, and drops generally over the last year. Meanwhile, E_α and $E_\alpha - E_\beta$ of BTC-BCH are bigger than of others. This means the exchange rate of BTC-BCH, and rises generally over the last year. Both observations indicate that BTC-BNB and BTC-BCH are highly volatile, so the Atomic Swap with those assets is more unfair than with other assets. Third, all dots are close to the line $E_\beta = E_\alpha$, while the dots of stock indices and fiat currency pairs almost lay on this line. A dot lying on $E_\beta = E_\alpha$ means the exchange rate rises and drops at the same level. Therefore,

[1] https://www.coingecko.com. Data was fetched at May 4th, 2019.
[2] https://finance.yahoo.com. Data was fetched at May 4th, 2019.
[3] https://www.investing.com. Data was fetched at May 4th, 2019.

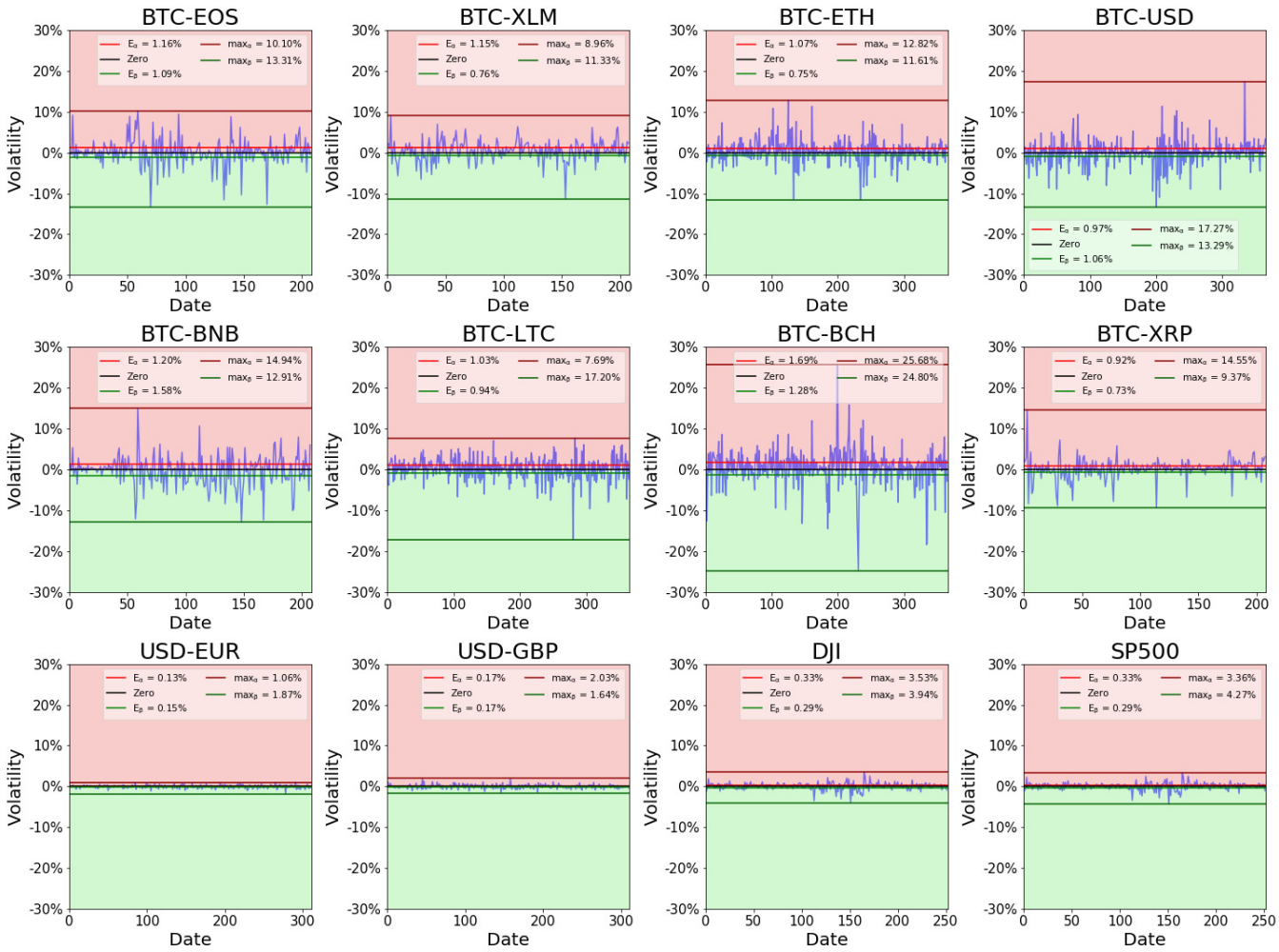

Figure 3: The daily percentage changes for all selected cryptocurrency pairs, stock indices and fiat currency pairs over one year (from 03/05/2018 to 03/05/2019). For each figure, E_α, E_β, max_α and max_β are the expected profit rate, the expected mitigated risk rate, the maximum daily profit and the maximum daily mitigated risk, respectively. The red area is the Profit Area where Alice profit from the rising asset price, and the green area is the Risk Area where Alice mitigates the loss from the dropping asset price.

although more volatile than stocks and fiat currencies, exchange rates of cryptocurrency pairs rise and drop at the same level.

4.3 Estimating the premium

The unfairness of Atomic Swap comes from the fact that Alice can abort the contract without punishment. In Finance, the premium mechanism guarantees the good behaviours. As the Atomic Swap is equivalent to the premium-free American Call Option, the Cox-Ross-Rubinstein (CRR) Model [8] can be used for estimating the premium of Atomic Swaps.

Therefore, we can evaluate the unfairness of Atomic Swap by estimating the premium for American Call Options with cryptocurrencies.

As the premium is the only variable in an option contract, estimating the premium is also called the "Option Pricing" problem. In Finance, the Black-Scholes (BS) Model [11] is utilised to price the European Call Options, while the CRR model is utilised to price the American Call Options.

Therefore, in order to evaluate the unfairness of Atomic Swap, we use the CRR Model to estimate how much the premium should be in Atomic Swaps.

4.3.1 The Cox-Ross-Rubinstein Model Explained. The Cox-Ross-Rubinstein (CRR) Model [8] - a.k.a. the Binomial Options Pricing Model (BOPM) - is a numerical method for pricing American-style Options. Intuitively, the CRR model enumerates all possible asset prices of the asset in the near future based on the price volatility, then reverse-engineers the premium based on the enumerated asset

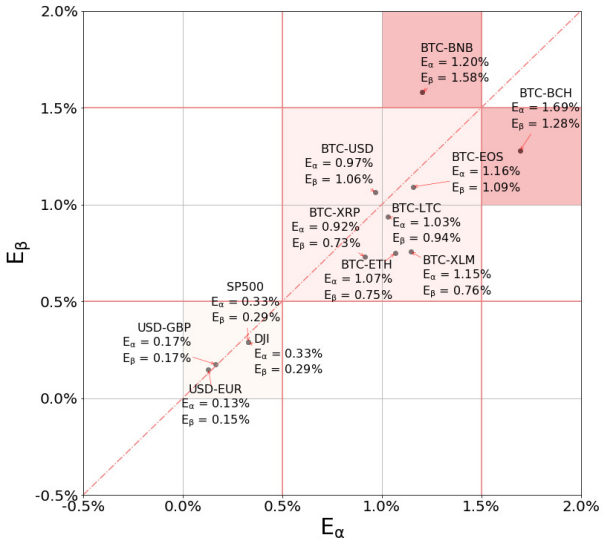

Figure 4: Visualising the expected profit rate E_α and the expected mitigated risk rate E_β for each cryptocurrency pair, stock index and fiat currency pair.

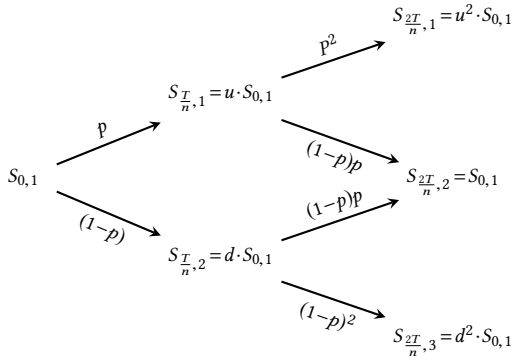

Figure 5: The binomial price tree $\mathcal{T}$.

prices. More specifically, using the CRR model to price the American Call Option $\Pi = (\pi_1, \pi_2, K, A, T, C)$ follows the steps below:

(1) Creating the binomial price tree
(2) Calculating the premiums for leaf nodes
(3) Iteratively reconstructing the premiums for non-leaf nodes

Creating the binomial price tree. The binomial price tree $\mathcal{T}$ of the height n (as shown in Fig. 5) represents the possible future prices within the time period T discretely. n can be picked arbitrarily: with larger n, the result will be more accurate, but the computing overhead will be heavier. Each node $\mathcal{T}_{t,i}$ is attached with an asset price $S_{t,i}$ and a premium price $C_{t,i}$, where $t \in \{0, \frac{T}{n}, \frac{2T}{n}, ..., T\}$ is the point of time, and i is the id of this node at its level. The CRR model assumes that the asset price will either move up or down by a specific factor per step in $\mathcal{T}$. The move-up factor is u, and the move-down factor is d. For example, given the initial asset price $S_{0,1}$, the asset price after one move-up $S_{\frac{T}{n},1}$ is $u \cdot S_{0,1}$, and the asset price after one move-down $S_{\frac{T}{n},2}$ is $d \cdot S_{0,1}$.

Table 1: Summary of symbols in the Cox-Ross-Rubinstein Model.

Variable	Description	Comment
u, d	The rising and dropping rates for prices in the binomial tree $\mathcal{T}$	$u \cdot d = 1$
σ_d, σ_a	The daily and annualised percentage change rates of asset prices	
T	The strike time (measured in years)	
n	The depth of $\mathcal{T}$	we pick $n = 36$
Δt	The time period between two adjacent nodes on $\mathcal{T}$ (measured in years)	$\Delta_t = \frac{T}{n}$
$S_{t,i}$	The asset price of the i-th node on the $\frac{t}{\Delta t}$-th level of $\mathcal{T}$	
$C_{t,i}$	The premium of the i-th node on the $\frac{t}{\Delta t}$-th level of $\mathcal{T}$	
p, q	The probabilities that the asset price rises and drops	

u and d are calculated using the annualised volatility σ_a of the underlying asset price. In the CRR model, the move-up and move-down are symmetric - $u \cdot d = 1$, and the rate of move-up and move-down is positive correlated with σ_a:

$$u = e^{\sigma_a \sqrt{\frac{T}{n}}} \tag{1}$$

$$d = e^{-\sigma_a \sqrt{\frac{T}{n}}} = \frac{1}{u} \tag{2}$$

Here, T is measured in years, and σ_a is defined as the standard deviation of the annual price changes in percentage. σ_a can be computed from the standard deviation σ_d of daily price changes in percentage as below:

$$\sigma_a = \sigma_d \sqrt{d} \tag{3}$$

$$\sigma_d = \sqrt{\frac{\sum_{i=1}^{d} (S'_i - \bar{S'})^2}{d-1}} \tag{4}$$

where d is the number of trading days within a year. For cryptocurrencies, d equals to the number of a days within a year. Note that S'_i is the percentage change of the price on day i, rather than the price itself. $\bar{S'}$ is the average value of all S'_is within the d days.

Each asset price $S_{t,i}$ can be calculated directly by $S_{t,i} = S_{0,1} \cdot u^{N_u - N_d}$, where $S_{0,1}$ is the initial asset price, and N_u, N_d are the times of move-ups and move-downs, respectively.

Calculating the premiums for leaf nodes. In the first step, only the asset prices are determined rather than the premiums. This step further determines the premiums for leaf nodes. For each leaf node $\mathcal{T}_{n,i}$, the premium is $C_{n,i} = max[(S_{n,i} - K), 0]$.

Iteratively reconstructing the premiums for earlier nodes. We back-propagate the premiums for leaf nodes to earlier premiums. Each

earlier premium is calculated from premiums of the later two nodes weighted by their possibilities. The move-up and move-down possibility are p and q where $p+q=1$, and the risk-free rate is $r=q$. More specifically, each earlier premium $C_{t-\Delta t,i}$ is calculated from later premiums as:

$$C_{t-\Delta t,i} = e^{-r\Delta t}(pC_{t,i} + qC_{t,i+1}) \tag{5}$$

where $\Delta t = \frac{T}{n}$, and p,q,r are computed as

$$p = \frac{e^{(r-q)\Delta t} - d}{u-d} \tag{6}$$

$$q = 1-p \tag{7}$$

$$r = q \tag{8}$$

such that the premium distribution simulates the geometric Brownian motion [12] with parameters r and σ.

In this way, the earliest premium $C_{0,1}$, which is our targeted estimated premium C - can be calculated by iteratively back-propagating the later premiums.

Table 1 summarises all symbols used in the Cox-Ross-Rubinstein model.

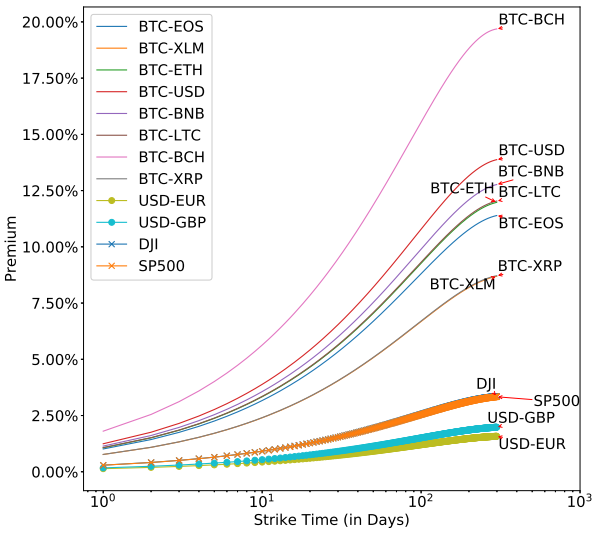

Figure 6: Estimated premium with different strike times for each cryptocurrency pair, stock index and fiat currency pair. Lines with the marker "x" are for stock indices; lines with the marker "o" are for fiat currencies; and lines without marker are for cryptocurrency pairs.

4.3.2 Experiments. We use the same data as Section 4.2, and choose $n = 36$ for the CRR model. We estimate the premium for the same assets (8 cryptocurrency pairs + 2 stock indices + 2 fiat currency pairs in Section 4.2) with the strike time T ranging from 1 to 300. Figure 6 shows our pricing results.

First, we observe that the premium of cryptocurrency pairs is much more expensive than of stocks, and the premium of stocks is more expensive than of fiat currencies at any given time. Recall the evaluated unfairness in Section 4.2, its results are consistent with the premium pricing results: the more volatile the market is, the more unfair the Atomic Swap will be, and the higher the premium should be. Second, with the default strike time $T = 1$ of the Atomic Swap, the premium for cryptocurrency pairs vary from approximately 1% to 2.3% of the underlying asset value, but the values for stocks and fiat currency pairs are approximately 0.3%. Third, for all evaluated items, the premium values rise monotonically with T increasing. This is because the longer expiration time lets Alice to have more control on the option - he has more time to predict the price and decide to exercise or abort the option.

5 FAIR ATOMIC SWAPS

In this section, we propose an improvement on the original Atomic Swap to make it fair. It implements the premium mechanism, and supports both the currency exchange-style Atomic Swap and the American Call Option-style Atomic Swap.

5.1 Design

5.1.1 Difference between Currency Exchange and Options. We first summarise the design objectives for Atomic Swap.

To our knowledge, the Atomic Swap protocol is originally designed for the fair exchange between different cryptocurrencies. However, according to our analysis, the protocol is unfair due to the Optionality and the free premium. Also, for (crypto)currency exchange, the protocol should have no Optionality. The currency exchange and the American Call Option differ in Finance: the currency exchange is a type of Spots [13], while the American Call Option is a type of Options. The Spot Contract and the Option Contract aim at different application scenarios: the Spot Contract aims at exchanging the ownership of assets, while the Option Contract aims at providing Alice an "option" to trade. More specifically, Spots and Options differ in the following aspects:

- The Spot Contract is exercised immediately, while the Option Contract is exercised on or prior to a specified date in the future.
- The Spot Contract cannot be aborted once signed by both parties, while in the Option Contract Alice can abort the contract with the loss of the premium.
- The Spot Contract itself has no value, while the Option Contract itself has value - the premium.

5.1.2 Premium for Currency Exchange and American Call Options. According to Section 5.1.1, the currency exchange-style Atomic Swaps and the American Call Option-style Atomic Swaps differ in design objectives.

Atomic Swaps for Currency Exchange. For the currency exchange, both parties are not permitted to abort the contract once signed. However, in Atomic Swaps, Alice can abort the swap by not releasing the random secret. Therefore, the protocol should discourage Alice to abort the swap. To achieve this, we can use the premium mechanism

as the collateral: Alice should deposit the premium besides her asset when **Initiate**. The premium should follow that: **Alice pays the premium to Bob if Bob refunds his asset after his timelock but before Alice's timelock. If Alice's timelock expires, Alice can refund her premium back.**

Atomic Swaps for American Call Options. For the American Call Options, Alice should pay for the premium besides the underlying asset, regardless of whether the swap is successful or not. In reality, the option sellers are trustworthy - the option sellers never abort the contract. However, in Atomic Swaps, Bob can abort the contracts like Alice. To keep the Atomic Swap consistent with the American Call Options, the premium should follow that: **Alice pays the premium to Bob if 1) Alice redeems Bob's asset before Bob's timelock, or 2) Bob refunds his asset after Bob's timelock but before Alice's timelock. If Alice's timelock expires, Alice can refund her premium back.**

5.2 Our protocol

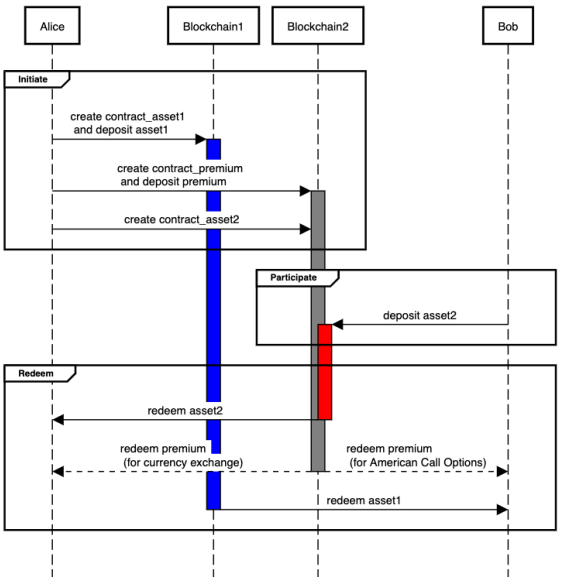

Figure 7: Sequence diagram of our Atomic Swap. For currency exchange-style Atomic Swaps, the premium will go back to Alice if the swap is successful (the left dotted line). For American Call Option-style Atomic Swaps, the premium will go to Bob if the swap is successful (the right dotted line).

We propose an improvement on the Atomic Swap, which implements the premium mechanism, to make it fair. It can fulfill design objectives of both the currency exchange and the American Call Option. Figure 7) shows the process of our Atomic Swap.

We denote our Atomic Swap protocol $\mathcal{AS}'$ as

$$\mathcal{AS}' = (x_1, Coin_1, x_2, Coin_2, pr)$$

where pr is the amount of the premium measured in $Coin_2$. In our protocol, besides x_1 $Coin_1$, Alice should also lock pr $Coin_2$ on BC_2, which will be described later.

Similar to the original Atomic Swap $\mathcal{AS}$, our protocol consists of four stages: **Initiate**, **Participate**, **Redeem** and **Refund**.

Initiate. Different from $\mathcal{AS}$, Alice also creates Bob's contract script C_2 and its associated transaction $tx_{C,2}$ when **Initiate** in $\mathcal{AS}'$.

C_1 and $tx_{C,1}$ is the same as in $\mathcal{AS}$, while C_2 and $tx_{C,2}$ are more sophisticated. C_2 contains two coherent sub-contracts C_2^{asset} and C_2^{pr}.

In C_2, C_2^{asset} is the contract for the asset x_2 $Coin_2$, which is the same as in $\mathcal{AS}$. C_2^{pr} is the contract for the premium pr, which implements the premium mechanism in the Atomic Swap. It supports both the currency exchange-style Atomic Swap and the American Call Option-style Atomic Swap. In more detail, the rules of C_2^{pr} are shown below:

- C_2^{pr} **for currency exchange** Alice pays pr to Bob with the condition: Bob refunds x_2 $Coin_2$ after δ_2 and before δ_1. If δ_1 expires, Alice can refund pr back.
- C_2^{pr} **for American Call Options** Alice pays pr to Bob with one of the two conditions: 1) Alice redeems x_2 $Coin_2$ before δ_2. 2) Bob refunds x_2 $Coin_2$ after δ_2 but before $Delta_1$ (note that $\delta_2 < \delta_1$). If δ_1 expires, Alice can refund pr back.

Alice published $tx_{C,1}$ on BC_1 and $tx_{C,2}$ on BC_2. Note that Alice only triggers C_1 and C_2^{pr} to execute at this stage. Bob will deposit x_2 $Coin_2$ trigger C_2^{asset} to execute when **Participate**.

Participate. Bob decides whether to participate in $\mathcal{AS}'$ by auditing $tx_{C,1}$ and $tx_{C,2}$. If Bob thinks contracts are fair, he will participate in $\mathcal{AS}'$, otherwise Bob will not participate and look for more profitable contracts from others. To participate in $\mathcal{AS}'$, Bob deposits x_2 $Coin_2$ in C_2^{asset}, and triggers C_2^{asset} to execute.

Redeem. Alice redeeming x_2 $Coin_2$ and Bob redeeming x_1 $Coin_1$ are the same in $\mathcal{AS}'$ and $\mathcal{AS}$. But in addition, riles in C_2^{pr} will work once triggering **Redeem** for $\mathcal{AS}'$.

Refund. Refunding x_1 $Coin_1$ for Alice and x_2 $Coin_2$ for Bob are the same as in $\mathcal{AS}$. But in addition, rules in C_2^{pr} will work once triggering **Refund** for $\mathcal{AS}'$.

6 IMPLEMENTATION

In this section, we describe how to implement our proposed protocol in Section 5 on different blockchains, including blockchains only supporting scripts (such as Bitcoin) and blockchains supporting smart contracts (such as Ethereum). In particular, we describe our design rationale, and provide reference implementations in Bitcoin scripts and Solidity smart contracts.

6.1 Requirements

To implement our protocol, the blockchain should support 1) stateful transactions, 2) the timelock and 3) the hashlock.

Stateful transactions. Transactions should be stateful: executing a transaction can depend on prior transactions. In our protocol, whether pr goes to Alice or Bob depends on the status of $Coin_2$. Therefore, the transaction of pr relies on the status of $Coin_2$ payment transaction.

Hashlock. The transactions should support the hashlock: a payment is proceeded only when the payee provides the preimage of a hash. In our protocol, exchanging $Coin_1$ and $Coin_2$ atomically is based on the hashlock - Alice redeems $Coin_2$ first by releasing the preimage, then Bob can redeem $Coin_1$ by using the released preimage.

Timelock. The transactions should support the timelock: a payment will expire after a specified time if the payee cannot redeem the payment. In our protocol, the transactions of $Coin_1$, $Coin_2$ and *pr* are all timelocked, in order to avoid locking money in transactions forever.

6.2 Smart contracts

Smart contracts support all aforementioned functionalities, so can easily implement our protocol. We use Solidity - one of programming languages for Ethereum smart contracts [14] - as an example. Our implementations are based on the original Atomic Swap Solidity implementation [15], but extend the premium mechanism. Extending the premium mechanism includes:

(1) The enumeration *PremiumState* for maintaining the premium payment state
(2) The modifiers *isPremiumRedeemable()* and *isPremiumRefundable()* for checking whether the premium can be redeemed or refunded
(3) The methods *redeemPremium()* and *refundPremium()* for redeeming and refunding the premium

```
enum AssetState { Empty, Filled, Redeemed, Refunded }
enum PremiumState { Empty, Filled, Redeemed, Refunded }
```

Listing 1: Maintaining the state of the asset and the premium.

The premium payment state PremiumState. In the original smart contract, an enumeration *State* maintains the asset state: **empty** means the asset has not been deposited; **filled** means the asset has been deposited; **redeemed** means the asset has been redeemed; **refunded** means the asset has been refunded.

In our contract, we decouple *State* to the asset state *AssetState* and the premium state *PremiumState*. Both *AssetState* and *PremiumState* are the same as the original *State*. The code is shown in Listing 1. *Empty* means Alice has not triggered **Initiate**, and has not deposited the premium yet. *Filled* means Alice has deposited the premium, indicating that Alice has triggered **Initiate**, but neither Alice nor Bob refunds or redeems the premium. *Redeemded* and *refunded* means Bob redeems the premium and Alice refunds the premium, respectively.

isPremiumRedeemable() and isPremiumRefundable(). Checking whether the premium is redeemable or refundable is the most critical part of our protocol. Because the premium payment relies on the $Coin_2$ payment, checking the premium refundability and redeemability involves checking the $Coin_2$ status - *AssetState* in our implementation.

isPremiumRedeemable() and *isPremiumRefundable()* for the currency exchange-style Atomic Swap are shown in Figure 2, and for the American Call Option-style Atomic Swap are shown in Figure 3.

```
// Premium is refundable when
// 1. Alice initiates but Bob does not participate
//    after premium's timelock expires
// 2. asset2 is redeemed by Alice
modifier isPremiumRefundable(bytes32 secretHash) {
    // the premium should be deposited
    require(swaps[secretHash].premiumState == PremiumState.Filled);
    // Alice invokes this method to refund the premium
    require(swaps[secretHash].initiator == msg.sender);
    // the contract should be on the blockchain2
    require(swaps[secretHash].kind == Kind.Participant);
    // if the asset2 timelock is still valid
    if block.timestamp <= swaps[secretHash].assetRefundTimestamp {
        // the asset2 should be redeemded by Alice
        require(swaps[secretHash].assetState == AssetState.Redeemed);
    } else { // if the asset2 timelock is expired
        // the asset2 should not be refunded
        require(swaps[secretHash].assetState != AssetState.Refunded);
        // the premium timelock should be expired
        require(block.timestamp > swaps[secretHash].premiumRefundTimestamp);
    }
    _;
}
// Premium is redeemable for Bob when asset2 is refunded
// which means Alice holds the secret maliciously
modifier isPremiumRedeemable(bytes32 secretHash) {
    // the premium should be deposited
    require(swaps[secretHash].premiumState == PremiumState.Filled);
    // Bob invokes this method to redeem the premium
    require(swaps[secretHash].participant == msg.sender);
    // the contract should be on the blockchain2
    require(swaps[secretHash].kind == Kind.Participant);
    // the asset2 should be refunded
    // this also indicates the asset2 timelock is expired
    require(swaps[secretHash].assetState == AssetState.Refunded);
    // the premium timelock should not be expired
    require(block.timestamp <= swaps[secretHash].premiumRefundTimestamp);
    _;
}
```

Listing 2: The condition to redeem and refund the premium for currency-exchange-style Atomic Swaps.

```
// Premium is refundable for Alice only when Alice initiates
// but Bob does not participate after premium's timelock expires
modifier isPremiumRefundable(bytes32 secretHash) {
    // the premium should be deposited
    require(swaps[secretHash].premiumState == PremiumState.Filled);
    // Alice invokes this method to refund the premium
    require(swaps[secretHash].initiator == msg.sender);
    // the contract should be on the blockchain2
    require(swaps[secretHash].kind == Kind.Participant);
    // premium timelock should be expired
    require(block.timestamp > swaps[secretHash].premiumRefundTimestamp);
    // asset2 should be empty
    // which means Bob does not participate
    require(swaps[secretHash].assetState == AssetState.Empty);
}
// Premium is redeemable for Bob when asset2 is redeemed or refunded
// which means Bob participates
modifier isPremiumRedeemable(bytes32 secretHash) {
    // the premium should be deposited
    require(swaps[secretHash].premiumState == PremiumState.Filled);
    // Bob invokes this method to redeem the premium
    require(swaps[secretHash].participant == msg.sender);
    // the contract should be on the blockchain2
    require(swaps[secretHash].kind == Kind.Participant);
    // the asset2 should be refunded or redeemed
    require(swaps[secretHash].assetState == AssetState.Refunded || swaps[
        secretHash].assetState == AssetState.Redeemed);
    // the premium timelock should not be expired
    require(block.timestamp <= swaps[secretHash].premiumRefundTimestamp);
    _;
}
```

Listing 3: The condition to redeem and refund the premium for American Call Option-style Atomic Swaps.

The currency exchange-style Atomic Swap and the American Call Option-style Atomic Swap differ when $AssetState = Redeemed$: in the currency exchange-style Atomic Swap the premium belongs to Alice while in the American Call Option-style Atomic Swap the premium belongs to Bob.

redeemPremium() and refundPremium(). *redeemPremium()* and *refundPremium()* are similar to *redeemAsset()* and *refundAsset()*, and their executions are secured by *isPremiumRedeemable()* and *isPremiumRefundable()*. The code is shown in Listing 4.

```
function redeemPremium(bytes32 secretHash)
    public
    isPremiumRedeemable(secretHash)
{
    // transfer the premium to Bob
    swaps[secretHash].participant.transfer(swaps[secretHash].premiumValue);
    // update the premium state to redeemded
    swaps[secretHash].premiumState = PremiumState.Redeemed;
    // notify the function invoker
    emit PremiumRedeemed(
        block.timestamp,
        swaps[secretHash].secretHash,
        msg.sender,
        swaps[secretHash].premiumValue
    );
}
function refundPremium(bytes32 secretHash)
    public
    isPremiumRefundable(secretHash)
{
    // transfer the premium to Alice
    swaps[secretHash].initiator.transfer(swaps[secretHash].premiumValue);
    // update the premium state to refunded
    swaps[secretHash].premiumState = PremiumState.Refunded;
    // notify the function invoker
    emit PremiumRefunded(
        block.timestamp,
        swaps[secretHash].secretHash,
        msg.sender,
        swaps[secretHash].premiumValue
    );
}
```

Listing 4: The functions for redeeming and refunding the premium.

6.3 Bitcoin script

Unfortunately, Bitcoin cannot support our protocol directly, because Bitcoin does not support the stateful transaction functionalities. First, the Bitcoin script is designed to be stateless [16]. Second, there is no such things like the Ethereum's "world state" [14] in Bitcoin: the only state in Bitcoin is the Unspent Transaction Outputs (UTXOs) [17].

New Opcode OP_LOOKUP_OUTPUT. In order to make Bitcoin script support our protocol, we use an opcode called OP_LOOKUP_OUTPUT. OP_LOOKUP_OUTPUT was proposed, but has not implemented in Bitcoin yet [18]. It takes the id of an output, and produces the address of the output's owner. With OP_LOOKUP_OUTPUT, the Bitcoin script can decide whether Alice or Bob should take the premium by "<asset2_output> OP_LOOKUP_OUTPUT <Alice_pubkeyhash> OP_EQUALVERIFY".

Implementing OP_LOOKUP_OUTPUT is easy in Bitcoin - it only queries the ownership of an output from the indexed blockchain database. This neither introduces computation overhead, nor breaks the "stateless" design of the Bitcoin script.

Decoupling the contract creation and the contract invocation. For smart contracts, the contract is created and invoked in separate transactions: creating the contract is by publishing a transaction which creates the smart contract, and invoking the contract is by publishing a transaction which invokes a method in the smart contract. However, Bitcoin has no smart contracts, and the "contract" is created and invoked in a single transaction. In this way, the timelock starts right after the contract creation rather than the contract invocation. This is problematic: the premium contract should not be triggered until Bob participates in the swap.

Thanks to the multi-signature transaction functionality in Bitcoin, Alice and Bob can first create the contract off-chain, then invoke the contract on-chain.

Multi-signature transactions refer to transactions signed by multiple accounts [16]. A M-of-N ($M \leq N$) multi-signature transaction means the transaction requires M out of N accounts to sign it. If less than M accounts sign the transaction, the transaction cannot be verified as valid by the blockchain. In Bitcoin, constructing a multi-signature transaction requires accounts to create a multi-signature address first [16].

With multi-signature transactions, we can decouple the contract creation and invocation as follows: first, Alice and Bob create a 2-of-2 multi-signature address; second, Alice and Bob mutually construct and sign a transaction which includes the premium payment and the $Coin_2$ payment; finally, they publish the transaction in the name of the 2-2 multi-signature address.

Note that constructing and signing the transaction is done off-chain: first, Bob creates the $Coin_2$ transaction and sends it to Alice; second, Alice creates the premium transaction which uses OP_LOOKUP_OUTPUT to check the ownership of $Coin_2$ transaction outputs; third, Alice merges the $Coin_2$ transaction and the premium transaction to a single transaction, signs the transaction, and sends it to Bob; finally, Bob signs the transaction and sends it to Alice. At this stage, both Alice and Bob have obtained the mutually signed transaction, which consists of both the premium transaction and the $Coin_2$ transaction.

The premium transaction. Listing 5 and Listing 6 show the premium transaction in the Bitcoin script , for both the currency-style and the American Call Option-style Atomic Swaps, respectively.

```
ScriptSig:
    Redeem: <Bob_sig> <Bob_pubkey> 1
    Refund: <Alice_sig> <Alice_pubkey> 0
ScriptPubKey:
    OP_IF // Normal redeem path
        // the owner of <asset2_output> should be Alice
        // which means Alice has redeemed asset2
        <asset2_output> OP_LOOKUP_OUTPUT <Alice_pubkeyhash> OP_EQUALVERIFY
        OP_DUP OP_HASH160 <Bob_pubkeyhash>
    OP_ELSE // Refund path
        // the premium timelock should be expired
        <locktime> OP_CHECKLOCKTIMEVERIFY OP_DROP
        OP_DUP OP_HASH160 <Alice pubkey hash>
    OP_ENDIF
    OP_EQUALVERIFY
    OP_CHECKSIG
```

Listing 5: The currency exchange-style Atomic Swap contract in Bitcoin script.

```
ScriptSig:
    Redeem: <Bob_sig> <Bob_pubkey> 1
    Refund: <Alice_sig> <Alice_pubkey> 0
ScriptPubKey:
    OP_IF // Normal redeem path
        // the owner of the asset2 should not be the contract
        // it should be either (redeemde by) Alice or (refunded by) Bob
        // which means Alice has redeemed asset2
        <asset2_output> OP_LOOKUP_OUTPUT <Alice_pubkeyhash> OP_NUMEQUAL
        <asset2_output> OP_LOOKUP_OUTPUT <Bob_pubkeyhash> OP_NUMEQUAL
        OP_ADD 1 OP_NUMEQUALVERIFY
        OP_DUP OP_HASH160 <Bob_pubkeyhash>
    OP_ELSE // Refund path
        // the premium timelock should be expired
        <locktime> OP_CHECKLOCKTIMEVERIFY OP_DROP
        OP_DUP OP_HASH160 <Alice pubkey hash>
    OP_ENDIF
    OP_EQUALVERIFY
    OP_CHECKSIG
```

Listing 6: The American Call Option-style Atomic Swap contract in Bitcoin script.

7 DISCUSSION

7.1 Security of Atomic Swap

Although already widely adopted, Atomic Swap has security issues.

First, the security of Atomic Swaps relies on the security of blockchains: if the blockchains involved in the swaps are insecure, the Atomic Swaps will also be insecure.

Second, the Atomic Swap contracts are written in high-level languages, so the compiled contracts can be insecure if the contract compilers are flawed.

Third, the timelock is unreliable in the cross-chain scenario. Similar to other distributed systems [19], different blockchains are unsynchronised on the time. Blockchains timestamp events by either two approaches: using the block height or using the UNIX timestamp. The block height can serialise events on a blockchain by time, but cannot serialise events outside the blockchain. In addition, the new block generation is a random process, so the block height cannot indicate the precise time in reality. Using the UNIX timestamp doesn't work, either. This is because the consensus participants are responsible for timestamping events, but the consensus participants can be unreliable: they may use the wrong time, either on purpose or by accident.

7.2 Other countermeasures

Besides our proposal, there are some other countermeasures to address the Atomic Swap unfairness. Unfortunately, to our knowledge, all of them either have security flaws or significantly reduce the usability of Atomic Swaps.

The first countermeasure is to make Atomic Swap costly by charging setting up HTLCs, or increasing the transaction fee of HTLCs. However, these two solutions do not only significantly reduce the usability of Atomic Swaps, but also affect HTLCs not aiming at setting up Atomic Swaps.

The second solution is to use shorter timelock for Atomic Swaps. Unfortunately, short timelocks may cause unexpected consequences. Confirming transactions for setting up Atomic Swaps takes time, and the time required is highly unpredictable. With short timelocks, the transactions for setting up Atomic Swaps may be confirmed after the expiration of timelocks.

The third solution is to use a trusted third party (TTP) to implement the premium mechanism. When Alice initiates an Atomic Swap, the TTP forces Alice to deposit the premium. Although this TTP does not require Alice and Bob to escrow their assets, the TTP should be trustworthy and can be a single point of failure.

7.3 Limitations of our protocols

Still, our solutions are not perfect. The initiators of Atomic Swaps need to hold some participant's asset to initiate an Atomic Swap, for either collateralising successful swaps or paying for the option itself. Unfortunately, the initiators do not always have participant's asset: they may just hope to get some participant's asset with only his asset. Before doing an Atomic Swap, the initiator should get some participant's asset by arbitrary means. For example, he can buy some participant's asset from cryptocurrency exchanges, or initiate a smaller Atomic Swap with shorter timelocks and no premium. Also, in an exchange, the option seller can change the premium at any given time until the option buyer signs the option agreement, while in our protocols the option seller cannot update the premium after publishing the option contract. In this way, the original Atomic Swap protocol and our protocols are less efficient than centralised solutions in terms of the market liquidity.

8 RELATED WORK

The Atomic Swap protocol was first proposed on the BitcoinTalk forum informally in 2013 [5]. Herlihy et al. first formalised the Atomic Swap protocol [20]. Meyden et al. first formally analysed the Atomic Swap smart contracts [21]. Several Atomic Swap variants were proposed for sidechains [22] and solving conflicts from concurrent operations [23].

Our paper is the first attempt to model and quantify the unfairness of Atomic Swap. The optionality of Atomic Swaps was first identified by an anonymous person entitled ZmnSCPxj in the Lightning-dev mail list in 2018 [6]. BitMEX Research [24] and Dan Robinson [25] further claimed that the optionality cannot be eliminated in HTLC-based Atomic Swaps. However, they do not quantify the unfairness from such optionality.

Also, our paper proposes the first fair Atomic Swap protocol. Eizinger et al. first tried to address the optionality problem by implementing the premium mechanism in Atomic Swap [26]. However, their protocol is flawed: If Bob keeps not participating in the swap, he will get the premium. Liu used the Atomic Swap to construct the option [27], but paying for the premium requires an extra blockchain besides the two blockchains, and they do not justify its fairness. IDEX [28] escrows the premium on an Ethereum smart contract for Atomic Swaps. However, this scheme can only support ERC20 tokens. Furthermore, IDEX fully controls the smart contract, so makes no difference with centralised exchanges except for the audibility. Interledger [29] proposed an Atomic Swap protocol based on payment channels. In Interledger, Alice (holding coin A) creates a payment channel with Bob (holding coin B) on the blockchain of coin A, and Bob creates a payment channel with Alice on the blockchain of coin B. After that, Alice gradually pays coin A to Bob, while Bob gradually pays coin B to Alice. After both payments are finished, they settle both payment channels. However, this scheme suffers from time-consuming interactive operations and poor efficiency.

9 CONCLUSION

In this paper, we investigate the unfairness of Atomic Swap. We show that an Atomic Swap is equivalent to a premium-free American Call Option, and Atomic Swap is unfair to the participant.

We then evaluate the unfairness of Atomic Swap protocol, and compare the unfairness between mainstream cryptocurrencies and conventional financial assets. Our evaluation consists of quantifying the unfairness and estimating the unpaid premium. The evaluation results show that the Atomic Swap with cryptocurrencies is much more unfair than with stocks and fiat currencies in the same setting, because the cryptocurrency market is highly volatile.

Furthermore, we propose an improvement on Atomic Swap, which implements the premium mechanism, to make it fair. It supports both the currency exchange-style Atomic Swap, and the American Call Option-style Atomic Swap. We implement our protocol in Solidity as an example of blockchains with smart contracts such as Ethereum. We also give instructions on implementing our protocol using Bitcoin script, which requires adding a single opcode to the Bitcoin script.

REFERENCES

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

[2] 2019. DEXWatch | DEX explorer. (2019). https://dex.watch/.

[3] 2019. distribuyed/index: A comprehensive list of decentralized exchanges (DEX) of cryptocurrencies, tokens, derivatives and futures, and their protocols. (2019). https://github.com/distribuyed/index.

[4] 2019. evbots/dex-protocols: A list of protocols for decentralized exchange. (2019). https://github.com/evbots/dex-protocols.

[5] Tier Nolan. 2013. Alt chains and atomic transfers. bitcointalk.org. (2013). https://bitcointalk.org/index.php?topic=193281.0.

[6] ZmnSCPxj. 2018. An Argument For Single-Asset Lightning Network. Lightning-dev. (2018). https://lists.linuxfoundation.org/pipermail/lightning-dev/2018-December/001752.html.

[7] Desmond J Higham. 2004. *An introduction to financial option valuation: mathematics, stochastics and computation*. Volume 13. Cambridge University Press.

[8] John C Cox, Stephen A Ross, and Mark Rubinstein. 1979. Option pricing: A simplified approach. *Journal of financial Economics*, 7, 3, 229–263.

[9] Almkglor and Harding. 2018. Payment channels - Bitcoin Wiki. (2018). https://en.bitcoin.it/wiki/Payment_channels.

[10] 2004. *A history of the global stock market: from ancient rome to silicon valley*. University of Chicago press, 20.

[11] Fischer Black and Myron Scholes. 1973. The pricing of options and corporate liabilities. *Journal of political economy*, 81, 3, 637–654.

[12] Ioannis Karatzas and Steven E Shreve. 1998. Brownian motion. In *Brownian Motion and Stochastic Calculus*. Springer, 47–127.

[13] John Hull. 1991. *Introduction to futures and options markets*. prentice Hall Englewood Cliffs, NJ.

[14] Gavin Wood et al. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151, 1–32.

[15] 2019. AltCoinExchange/ethatomicswap: Ethereum atomic swap. (2019). https://github.com/AltCoinExchange/ethatomicswap.

[16] Krzysztof Okupski. 2014. Bitcoin developer reference. *Eindhoven*.

[17] Satoshi Nakamoto et al. 2008. Bitcoin: a peer-to-peer electronic cash system.

[18] Martin Brown. 2015. Bitcoin script for a competitive crowdfunding-like contract. (2015). https://bitcoin.stackexchange.com/questions/36229/bitcoin-script-for-a-competitive-crowdfunding-like-contract.

[19] George F Coulouris, Jean Dollimore, and Tim Kindberg. 2012. *Distributed systems: concepts and design*. (fifth edition). pearson education, 2.

[20] Maurice Herlihy. 2018. Atomic cross-chain swaps. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*. ACM, 245–254.

[21] Ron van der Meyden. 2018. On the specification and verification of atomic swap smart contracts. *arXiv preprint arXiv:1811.06099*.

[22] Peter Robinson, David Hyland-Wood, Roberto Saltini, Sandra Johnson, and John Brainard. 2019. Atomic Crosschain Transactions for Ethereum Private Sidechains. *arXiv preprint arXiv:1904.12079*.

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

[24] BitMEX Research. 2019. Atomic Swaps and Distributed Exchanges: The Inadvertent Call Option. (2019). https://blog.bitmex.com/atomic-swaps-and-distributed-exchanges-the-inadvertent-call-option/.

[25] Dan Robinson. 2019. HTLCs Considered Harmful. (2019). https://cyber.stanford.edu/sites/g/files/sbiybj9936/f/htlcs_considered_harmful.pdf.

[26] Thomas Eizinger, Lloyd Fournier, and Phillip Hoenisch. 2018. The state of atomic swaps. diyhpl.us. (2018). http://diyhpl.us/wiki/transcripts/scalingbitcoin/tokyo-2018/atomic-swaps/.

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

[28] 2019. Whitepaper - Idex. (2019). https://idex.market/whitepaper.

[29] Stefan Thomas and Evan Schwartz. 2015. A protocol for interledger payments. *URL https://interledger. org/interledger. pdf*.