Cost Fairness for Blockchain-Based Two-Party Exchange Protocols

Matthias Lohr*, matthiaslohr@uni-koblenz.de
Benjamin Schlosser†, benjamin.schlosser@tu-darmstadt.de
Jan Jürjens*‡, juerjens@uni-koblenz.de
Steffen Staab§¶, steffen.staab@ipvs.uni-stuttgart.de
* CCRDMT, University of Koblenz-Landau, Koblenz, Germany
† Chair of Applied Cryptography, Technical University of Darmstadt, Darmstadt, Germany
‡ Fraunhofer ISST, Dortmund, Germany
§ Institute for Parallel and Distributed Systems (IPVS), University of Stuttgart, Germany
¶ University of Southampton, Southampton, United Kingdom

***Abstract*—Blockchains can guarantee fairness during the exchange of digital goods such that in a two-party exchange no one is defrauded by a malicious opponent. While several notions of fairness have been discussed in the literature, they all ignore that damage cannot only be incurred by the malicious failure of the exchange, but also by an unfair allocation of transaction costs. To address this issue we: 1. define the novel concept of cost fairness, which 2. builds on the notion of maximum cost matrices that formalize transaction costs in different combinations of benevolent and malicious behavior. 3. We show how limited notions of cost fairness can be achieved by modifying an existing exchange protocol or using a container protocol. In particular, we also provide 4. a tool that let us predict the maximum cost matrix for a specific protocol execution and, thus, gives trade exchange parties the possibility to weigh not only the value of transaction of exchanged goods but also the associated transaction costs.**

***Index Terms*—blockchain, fairness, cost fairness, fair exchange protocols**

I. Introduction

When talking about the term *fairness* in electronic commerce [3], it usually refers to the property of a protocol that no party can take advantage over the other party by behaving maliciously. In the context of data exchange, fairness deals with the whereabouts of the data to be transferred before and after the exchange and the funds paid for it. It was shown, e. g., by Pagnia and Gärtner, that a trusted third party is required in order to achieve fairness [16], [19]. In real-world applications, trusted third parties can be public institutions, notaries, company consortiums, etc. With the emergence of blockchains, several approaches have been presented how such a system can be used to improve on existing protocols used in distributed systems [11], [13], [4], [7]. In particular, blockchains can be used to play the role of a distributed trusted third party in case of two or more parties involved in the protocol do not trust each other [8], [9], [10], [17].

Using the services of a trusted third party incurs costs regardless of whether a centralized trusted third party is employed or whether a blockchain plays this role. Classic trusted third parties are usually paid with a fixed, e.g., monthly, rate, or per transaction. In both cases, it is known before the exchange which participant has to bear which cost. In the Ethereum blockchain, the transaction fees to be paid stem from the operations carried out by the smart contract that implements the trusted third party functionalities [18]. The fees have to be paid by the person invoking the smart contract and may vary depending on various system parameters as well as the smart contract implementation. In particular, the costs depend on both the invocation of smart contract methods and the size of transferred method parameters.

If an exchange protocol is realized via blockchains, the smart contract of the protocol can offer multiple methods, which have to be called in a defined order by the respective seller or buyer to conduct a fair exchange. The execution cost of these methods can differ depending on parameters. Different sizes and complexities among the smart contract methods can result in transaction fees to be paid by the seller that are different than the fees to be paid by the buyer.

Since the concept of fairness as defined by Asokan [3] does not cover the distribution of the cost to be paid, situations may arise which contradict to the intuitive understanding of fairness, as it might be possible to create situations in which an honest participant has to bear significantly more cost than a cheating adversary, although formally fulfilling the characteristic of fairness.

Until now, formal specifications of fairness in digital exchange protocols may enforce that either the two goods are exchanged or no exchange takes place at all, but they ignore who bears incurred transaction costs of a failed exchange. The assignment of transaction costs is already a non-negligible matter when the exchange succeeds. When one of the two parties causes the exchange to fail and the other party has to bear significant transaction costs, the overall functioning of the exchange platform is at stake as parties may be deterred from using such a platform in the first place. Therefore, our main contribution is the introduction of *cost fairness* as an additional criterion that an exchange protocol on a blockchain should fulfill.

In order to enable the examination and evaluation of fair protocols with regard to cost fairness, we discuss and answer

978-0-7381-0495-9/20/$31.00 ©2020 IEEE
DOI 10.1109/Blockchain50366.2020.00062

the following research questions:

RQ1 How can cost fairness be defined?

RQ2 How can exchange protocols be optimized regarding execution cost?

RQ3 How can exchange protocols be optimized regarding cost imbalances between participating parties?

To support the definition of cost fairness, we introduce the concept of a *maximum cost matrix*, which informs about the cost that can be imposed on the two trading parties in different configurations of cooperative or non-cooperative behaviour. Furthermore, as our second contribution, we present *BDTsim*, a framework for simulating the behavior of the parties of two-party exchange protocols in order to determine the maximal costs under different combinations of cooperative and non-cooperative behavior of the two parties, which provides input for a plausible and enforceable allocation of transaction costs on the two parties.

In Section II we survey related work describing properties of trading protocols and various notions of fairness found in the literature. In Section III we present our definition of cost fairness and Section IV presents how cost fairness can be determined. We discuss general aspects of cost fairness regarding blockchain-based fair exchange protocols using examples from existing exchange protocols in Section IV-C - IV-E. We conclude our work in Section V and provide our ideas for future work.

II. Related Work on Exchange Protocols

A. Fairness of Exchange Protocols

Informally, a protocol is said to be fair if no party can take advantage over an honestly behaving counterpart. According to Asokan [3], the formal definition of *strong fairness* means that at the point of protocol termination either the exchange proceeded successfully, i. e., both parties received what they aimed for, or the exchange failed, i. e., no party received what it wanted.

A relaxation of this notion is the flavor of *weak fairness* [3]. According to this property, it is not required that the exchange either completely succeeded or completely failed at the time of protocol termination. Instead, it is only guaranteed that if strong fairness is not given, an honest party can prove to a third party that the other party received the expected item while itself did not. Note, it is not required that the third party can resolve the dispute in the sense that the exchange successfully happened at the end. Instead, it just needs to be able to verify that the exchange was unfair.

B. FairSwap

The FairSwap protocol by Dziembowski et al. [9] provides *strong fairness* for both parties. It requires a trusted third party in case of a dispute which is realized by a smart contract. The implementation provided by the authors works over the Ethereum blockchain.

The protocol is applicable for any exchange of data which can be represented as a binary string against money (e. g., online purchases of movies or music files). An important requirement on the data is that it can be identified by its hash value. In the given example, the Merkle hash of the file is used as an identifier. This identifier is known to both parties at the start of the protocol.

An exchange that uses the FairSwap protocol is started by initializing a smart contract with an offer. The offer contains the identifier of the digital good, the demanded price, and some metadata about the digital good that is to be exchanged, e.g., the hash of the encrypted data which is transferred to the buyer at the same time as the contract gets initialized. After receiving the encrypted good and seeing the initialized smart contract, the buyer can decide to accept the offer by depositing the required amount of money in the smart contract. Afterward, the seller reveals the encryption key by sending it to the smart contract. Upon receiving the key via the smart contract, the buyer decrypts the data received at the beginning and compares it with the expected good. If he obtains the correct data, he finalizes the exchange by sending a final message to the smart contract. This triggers the payout for the seller and terminates the protocol. Otherwise, if the obtained data is not equal to the file corresponding to the identifier agreed-upon and stored in the smart contract, the buyer uses the information obtained by the encrypted data to create a public certificate of the other party's malicious behavior. In FairSwap, this certificate is called *proof of misbehavior*. Using this proof provided by the buyer, the smart contract is able to detect the seller's behavior and will refund the money deposited by the buyer when accepting the transfer. After execution, the protocol terminates resulting in a failed exchange. It is important that the buyer is only able to create a valid proof of misbehavior if the seller did not send the correct data. Therefore, the buyer is not able to falsely request a refund. This property is called *defamation free*.

It is easy to see that the execution of FairSwap requires both parties to pay transaction fees when interacting with the smart contract. Moreover, even if the buyer behaves honestly, he needs to increase his financial expenses when sending the proof of misbehavior to the judge smart contract. This situation can be considered unfair since the honest party is punished.

C. Financial Fairness

Financial fairness guarantees that an adversary is monetarily penalized, if he aborts upon receiving the output of the exchange before the honest party gets to know the output. Financial fairness is often provided by blockchain-based protocols with an inherent cryptocurrency (e. g., Bitcoin, Ethereum) [5], [12], [2], [1], [14], [15]. This notion does not ensure that the exchange either succeeds or fails as in strong fairness and neither guarantees that the honest party has evidence about the status of the transfer. Financial fairness targets to compensate the honest party in the case that the honest party reveals or releases its goods towards the cheating party, but the cheating party then, after learning about or receiving the goods, leaves the protocol without giving money or the goods to be exchanged in return. In this case, a previously deposited amount of money (also called security deposit or penalty) is

paid as compensation for the missing goods or money as well as the cost the honest party has to pay for.

However, this notion does not cover protocol abortions prior to the release or revelation of the goods to be exchanged. Since cost may also be incurred for, e.g., initialization, which must also be paid if one of the parties leaves the protocol, it is desirable for the honest party to be secured, e. g., financially, to prevent cost without corresponding compensation. Since this relates to the entire execution of the protocol and not just to a certain phase as in financial fairness, we need a concept that includes the cost of the entire protocol execution. In this work, we define our notion of *cost fairness* which covers the costs of the complete execution of the protocol and not, as in financial fairness, limited to a single phase.

It is necessary to mention that a protocol can also be designed to achieve none of the aforementioned notions. In this case, the protocol is said to guarantee *no fairness* at all. It might seem not intuitive to aim for a protocol that achieves no fairness, but such protocol might be cheaper with regard to transaction costs. Depending on the protocol, achieving stronger notions of fairness requires additional mechanisms or interaction rounds, which might be expensive. Moreover, depending on the value of the exchanged items, a party might be willing to take some additional risk in order to use a cheaper protocol.

III. Cost Fairness

In this section, we define the concept of cost fairness. We relate cost fairness to blockchain-based exchange protocols in Section IV.

When conducting a two-party exchange which requires a trusted third party, the following costs may be encountered by seller and/or buyer:

- Payment: the items/money used to pay for a traded good.
- Fees: the money used to pay for trusted third party operations.
- Fines: cost a buyer or seller must bear bear in case of protocol violation. The protocol defines the recipient of the fines, usually the trading partner as compensation or even to a third party (e. g., an infrastructure operator, or a charitable organization).

In the remainder of our work the term *transaction costs* refers to the sum of costs that do not include payment, i. e., fees plus fines spent minus fines received.

A. Modeling Cost Fairness as Non-cooperative Game

Game theory models behavior by two (or more) parties that individually pursue strategies that maximize their payoffs. In our setting, the available strategies are to complete the exchange of goods by conforming to the exchange protocol or not.

For each valid combination of strategies, we model the costs for the two involved parties. The core idea of our proposal is that we use a blockchain as a trusted third party that enforces payoffs and limits costs in such a way that parties are motivated to conform to the fair and cost-fair exchange protocol.

When talking about two-party protocols, we have four possible combinations of strategies on how the parties S and B can behave:

- S and B both follow the protocol
- S follows the protocol while B does not
- S does not follow the protocol while B does
- neither S nor B follow the protocol

For the remainder of the paper, we call S and B to be *honest* if they follow the protocol and *malicious* if they do not.

Depending on the protocol, the seller and buyer can act maliciously in different variants (e. g., in FairSwap, the seller by sending wrong data or the wrong key). Since different variants can result in different costs, and we are interested in an upper bound costs estimation of exchange protocols for each of these cases, we consider the maximum costs for each party for each of the combinations of strategies listed above. By summarizing the costs of all operations during a protocol execution and computing the maximum costs for all possible sequences of operations, we can calculate the costs for seller and buyer for the cases that both parties are honest (S_{hh} and B_{hh}), honest seller with malicious buyer (S_{hm} and B_{hm}), malicious seller with honest buyer (S_{mh} and B_{mh}), and both parties malicious (S_{mm} and B_{mm}). We write down these results of upper bound costs in a *maximum cost matrix* (see Table I).

Maximum Cost Matrix	B honest	B malicious
S honest	(S_{hh}, B_{hh})	(S_{hm}, B_{hm})
S malicious	(S_{mh}, B_{mh})	(S_{mm}, B_{mm})

TABLE I: Maximum cost matrix providing information about upper bound transaction costs for seller S and buyer B.

A maximum cost matrix can also be used to determine whether the risk in terms of financial loss in case of cheating is distributed evenly between both parties, i. e., $S_{hm} \approx B_{mh}$.

We provide an example of how a maximum cost matrix can be calculated for a given blockchain-based protocol in Section IV-C.

B. Full Cost Fairness

Definition 1. Given a maximum cost matrix M with entries S_{hh}, B_{hh}, S_{hm}, B_{hm}, S_{mh}, B_{mh}, S_{mm} and B_{mm}, an exchange protocol P achieves *full cost fairness*, if all of the following criteria are met:

CF1 In case of both parties are honest, the maximum costs for protocol execution (S_{hh}, B_{hh}) have to be known to the parties before the actual protocol starts.

CF2 In case of one party stays honest while the other party acts maliciously, the honest party must get compensated for any transaction costs it has incurred, i. e., $S_{hm} = 0 \wedge B_{mh} = 0$.

Note that property CF1 allows both parties to decide if they are willing to accept the required costs. Moreover, property CF2 does not upper bound the costs for malicious behavior

since a malicious party can deviate arbitrarily from the protocol specification.

The requirements CF1 and CF2 result in the maximum cost matrix to be achieved by an exchange protocol as depicted in Table II in order to call a protocol to be full cost fair. As long as all values from the maximum cost matrix of a given exchange protocol are lower or equal compared to the values in the maximum cost matrix required to achieve for full cost fairness, a protocol achieves full cost fairness.

Full Cost Fairness	B honest	B malicious
S honest	(S_{hh}, B_{hh})	$(0, \infty)$
S malicious	$(\infty, 0)$	(∞, ∞)

TABLE II: Maximum cost matrix to be achieved for full cost fairness.

C. Partial Cost Fairness

Depending on the environment, it might be non-trivial to achieve full cost fairness. In the literature, deposits are foreseen for the case that the buyer (seller) abandons the protocol. The deposited funds can be paid out to the other party as a compensation.[1] One might consider to require deposits before the execution of a transaction. If the depositing itself, however, also incurs transaction costs, it implies a recursive requirement to put forward deposits for protecting upcoming operations. It follows that if the first operation of the protocol is charged with fees, it is not possible to secure the first transaction using a deposit for compensation in case of one party abandoning the protocol after the first operation. As full cost fairness seems impossible to be achieved in the aforementioned scenario, we have developed a weaker notion of cost fairness:

Definition 2. Given a maximum cost matrix M with entries S_{hh}, B_{hh}, S_{hm}, B_{hm}, S_{mh}, B_{mh}, S_{mm} and B_{mm}, an exchange protocol P is *partial cost fairness* for the S (B), if all of the following criteria are met:

PCF1 In case of both parties are honest, the maximum costs for protocol execution (S_{hh}, B_{hh}) have to be known to the parties before the actual protocol starts.

PCF2 In case of S (B) is honest while B (S) is malicious, S (B) should get compensated for any transaction cost incurred ($S_{hm} = 0 \ \vee \ B_{mh} = 0$).

PCF1 and PCF2 result in a maximum cost matrix as depicted in Table III.

Partial Cost Fairness	B honest	B malicious
S honest	(S_{hh}, B_{hh})	(S_{hm}, ∞)
S malicious	(∞, B_{mh})	(∞, ∞)

TABLE III: Maximum cost matrix to be achieved for partial cost fairness with the condition $S_{hm} = 0 \ \vee \ B_{mh} = 0$.

[1]In the literature, depositing funds for this purpose is also referred to as security deposit or penalty.

D. Container Protocol for (Re-)Distribution of Transaction Cost

Using a container protocol as described below, the seller and buyer can agree on different cost distributions than given by the protocol itself. The container protocol consists of a deposit phase, where one or both parties deposit funds into the container protocol's smart contract. Then, the contained protocol is executed. In the end, the container protocol will do a payout as agreed at the beginning of the protocol.

1) Uninformed Container Protocol: When the contained protocol is used without modifications, the container protocol cannot distinguish between the strategies used by the seller or buyer. Therefore, there cannot be a strategy-dependent payoff in the payout phase of the container protocol. Therefore, the seller and buyer can only agree on one amount Δ, which has to be deposited at the beginning of the protocol and will be paid off at the end (see Table IV). The uninformed container protocol can be used to achieve partial cost fairness.

Uninf. C. P.	B honest	B malicious
S honest	$(S_{hh} + \Delta, B_{hh} - \Delta)$	$(S_{hm} + \Delta, B_{hm} - \Delta)$
S malicious	$(S_{mh} + \Delta, B_{mh} - \Delta)$	$(S_{mm} + \Delta, B_{mm} - \Delta)$

TABLE IV: Maximum cost matrix for the uninformed container protocol.

2) Informed Container Protocol: With minimum protocol modifications, it is possible to inform the container protocol about the strategies used by the seller or buyer as determined by the smart contract in order to ensure e. g. strong fairness. Using this information, the container protocol can do a strategy-dependent payout at the end of the protocol. Therefore, it is possible that the seller and buyer agree on different amounts Δ_{xy} which will be paid off depending on the protocol outcome (see Table V). Note: as long as the deposit phase of the container protocol is charged with fees, it is still not possible to use the container protocol to achieve *full cost fairness*.

Inf. C. P.	B honest	B malicious
S honest	$(S_{hh} + \Delta_{hh}, B_{hh} - \Delta_{hh})$	$(S_{hm} + \Delta_{hm}, B_{hm} - \Delta_{hm})$
S malicious	$(S_{mh} + \Delta_{mh}, B_{mh} - \Delta_{mh})$	$(S_{mm} + \Delta_{mm}, B_{mm} - \Delta_{mm})$

TABLE V: Maximum cost matrix for the informed container protocol.

IV. Cost Fairness in Blockchain-based Exchange Protocols

We will relate our definitions of cost fairness to blockchain-based exchange protocols. To this end, we first elaborate on the costs incurred by blockchain-based protocols and present a way to estimate these costs by our simulation tool BDTsim. Next, we analyze the FairSwap protocol in regard to cost fairness and discuss methods to influence transaction costs.

A. Calculation of Protocol Execution Cost

When all costs are well defined, we can calculate the costs for an exchange protocol analytically as well as in an evaluative manner. Since we focus our work on blockchain-based exchange protocols, we need a cost definition for blockchain interactions, which is provided in [18, p. 25] for the Ethereum blockchain. We assume that the costs for additional infrastructure required for using an Ethereum based fair exchange protocol (e. g., internet connection, computational resources) are negligibly low compared to the costs arising from Ethereum transactions in terms of transaction fees. Therefore, we only consider the Ethereum transaction fees in the following.

Note that the Ethereum blockchain has a special characteristic regarding the costs to be paid not only for the number of bytes sent to the smart contract, but also the value of the bytes. Ethereum charges more transaction fees for bytes with values unequal to zero than for bytes equal zero. Since blockchains make extensive use of cryptographic hashes whose values are deemed to be unpredictable regarding the input parameters, even a slight change of a transaction (e. g., different execution time) will result in a different hash. These differences may alter the number of byte-wise zeros and may therefore require different transaction fees. This can even happen for multiple protocol executions with an identical set of parameters. Nevertheless, these differences are negligibly small and do not tackle the concept of cost fairness in general. However, when applying the definitions of cost fairness to an Ethereum-based protocol, the limits presented by the maximum cost matrix should be compared with a certain tolerance.

B. BDTsim: Framework for Blockchain-based Data Trading Simulations

Since the costs in the form of transaction fees may depend on a large number of parameters (see Section I), an analytical analysis of protocol execution costs might be cumbersome to conduct. For this reason, we developed BDTsim[2], a simulation tool for Ethereum-based exchange protocols. We have chosen Ethereum as (first) platform to be supported, since Ethereum is the most known blockchain platform with support for smart contracts. This also allows us to run BDTsim on derivates of the Ethereum blockchain like Quorum[3]. BDTsim is written in Python and available as open-source software. In BDTsim, protocol developers and users can model the behavior of seller and buyer (i. e., by specifying the variants of operations conducted by the parties, including operations not allowed by the protocol to simulate a malicious party) and run a simulation using the original, unmodified Ethereum smart contract presented along with the concept of the protocol. As a result, BDTsim returns the transaction fees to be paid and possible incoming fund transfers in all honest/malicious cases as listed in Section III-A. These values can be used to create a maximum cost matrix for the simulated protocol (for the set of used protocol parameters during protocol execution). Furthermore, BDTsim has the ability to visualize possible protocol paths and costs per-operation to help identifying expensive parts of a protocol. This way, BDTsim can be used during protocol development to show and reduce the cost to be paid.

As an example of a protocol which can be used for conducting an exchange, we briefly describe a rudimentary exchange protocol that achieves no fairness for the buyer. In the *Simple Payment Protocol* the two parties agree off-chain about the digital good and the amount of money to be exchanged. After the agreement, the buyer transfers the demanded payment to the seller. Upon receiving the money, the seller can send the digital good to the buyer which completes the exchange protocol. The advantages of this protocol are its simplicity and the low cost. Since the protocol only contains a single on-chain transaction which, additionally, is only a cheap payment transaction, the overall transaction costs are low. However, this protocol does not provide any fairness guarantee for the buyer. After sending the money to the seller, the seller can freely decide whether or not to send the agreed-upon digital good. Moreover, there is no way for the buyer to show the dishonesty of the seller and fairness cannot be recovered. Additionally, although the transaction cost are rather low, the complete financial expenses are paid by the buyer. This holds even if the buyer is honest but the seller behaves maliciously.

BDTsim currently contains simulation support for the SimplePayment protocol (used for demonstration and testing issues) as well as FairSwap (more protocols will be added in the future, support is planned for the protocol of Delgado-Segura [8], OptiSwap [10] and SmartJudge [17]). In Section IV-C we describe a simulation of FairSwap using BDTsim and present the results. BDTsim is currently limited to smart contracts running on the Ethereum blockchain and is only able to monitor interactions, money transfers, and resulting costs of on-chain transactions. As long as it is within these limits and the number of possible sequences of operations allowed by the protocol is within reasonable limits, a protocol can be simulated by BDTsim.

For implementing support for a new protocol, the protocol author has to model the behavior of seller and buyer (all distinguishable variants) as code, which interacts with the environment (blockchain) provided by BDTsim. Off-chain interactions, such as direct data transfer, does not need to be implemented, since BDTsim only monitors blockchain interactions. However, since later steps in the protocol might require to access results from previous off-chain interactions between seller and buyer, sometimes off-chain interactions have to be implemented partially (e. g., in FairSwap the buyer has to use data encrypted by the seller in its last step of the protocol, so it is required to implement the encryption).

Listing 1 shows the Python source code modeling the behavior (honest and malicious) for seller and buyer for the SimplePayment protocol: For the modeling, we focus on

[2]Blockchain Data Trading Simulator
Sources: https://gitlab.com/MatthiasLohr/bdtsim/
Documentation: https://matthiaslohr.gitlab.io/bdtsim/

[3]Permissioned variant of the Ethereum blockchain – https://consensys.net/quorum/

rational and distinguishable behavior. E.g., the buyer could send no money or an amount smaller than the price if acting maliciously. However, since both would result in the same situation that the goods are not (fully) paid and a rational malicious buyer would try to optimize his transaction costs, he would prefer the cheaper way of not sending any funds.

BDTsim internally monitors method calls to `protocol_path.decide(...)`, builds up a tree of all possible protocol paths and runs a simulation for each possible path. Correct functionality of BDTsim is ensured by unit tests.

C. Cost Fairness Evaluation of FairSwap

FairSwap [9] allows to perform an exchange of digital good against money over a blockchain like Ethereum. The authors provide a full security proof showing that their protocol is fair according to the strong fairness definition. We evaluate which definitions of cost fairness FairSwap fulfills.

We model all distinguishable variants of seller and buyer behavior in BDTsim. For the simulation, we use the Ethereum smart contract initially published by the authors of FairSwap [4], except for small bug fixes and fixes for compatibility issues incurred by updates of the Solidity language specification. The models as well as the smart contract used for simulation are contained in the BDTsim project.

FairSwap supports two parameters for protocol execution. The first one states the number of chunks into which the complete data is split (in the practical implementation limited to values of the power of 2) and is denoted by n. The second parameter denotes the size of a single chunk in bytes (in the implementation limited to a multiple of 32).

FairSwap ($n = 2, s = 32$)	B honest	B malicious
S honest	(1,335,228, 51,363)	(1,263,039, 0)
S malicious	(1,320,350, 57,522)	(1,320,350, 57,522)

(a) FairSwap simulation with $n = 2, s = 32$

FairSwap ($n = 128, s = 32$)	B honest	B malicious
S honest	(1,336,840, 51,308)	(1,264,695, 0)
S malicious	(1,321,984, 65,355)	(1,321,984, 65,355)

(b) FairSwap simulation with $n = 128, s = 32$

FairSwap ($n = 128, s = 256$)	B honest	B malicious
S honest	(1,337,862, 51,330)	(1,265,739, 0)
S malicious	(1,322,994, 91,039)	(1,322,994, 91,039)

(c) FairSwap simulation with $n = 128, s = 256$

TABLE VI: Maximum cost matrix of the FairSwap [9] protocol. Costs are given in Gas as used as operation cost unit in Ethereum [18, p. 4].

As we can see when comparing Table VI to the requirements for full cost fairness or partial cost fairness, FairSwap does not achieve either, although it meets the criteria of strong fairness (see the proof in [9]). This shows that when using FairSwap in practice, both parties can use the protocol to mislead the other party to pay transaction fees without any compensation and thus cause financial damage. Conducting a more detailed analysis, we can see that a major part of the total costs is caused by smart contract deployment ($1,263,039$ gas when $n = 2, s = 32$, see Table VIa). When the buyer and the seller agree on a trade using FairSwap (the initial agreement is done off-chain) and the seller initializes the protocol by deploying the smart contract, the buyer can leave the protocol without any costs while the seller has to pay for a significant amount of gas. In reverse, if the buyer is honest but the seller acts maliciously, the buyer has to pay more gas compared to the situation where the seller is honest to reclaim the money deposited for paying for the data to be traded. The difference gets even higher with an increasing number of chunks or data size (see Tables VIb and VIc).

[4]https://github.com/lEthDev/FairSwap

D. Cost Fairness in Public and Private Blockchains

As already discussed in Section III-B, it is not possible to achieve full cost fairness as long as the first protocol operation is charged with a fee. Since public blockchains like Bitcoin and Ethereum charge for any transaction, it is not possible to create a protocol that realizes full cost fairness by relying solely on on-chain transactions. For now, we also do not know about any mechanism using off-chain transactions for achieving full cost fairness for a protocol running on a public blockchain.

Public blockchains use transaction fees as an incentive for people to participate in extending the blockchain, the situation is different for permissioned blockchains. Permissioned blockchains are usually operated in a specific context, e. g., by a company or a consortium of companies. There are multiple possible reasons for operating a permissioned blockchain instead of using a public instance, e. g., privacy, well-known identities of participants (they have to authenticate in order to access the permissioned instance), performance and cost reduction. Since the operator can decide about the conditions for accessing and interacting with the blockchain, he can also decide about the transaction fees to be paid. Hyperledger Fabric [6], a framework for realizing customized private blockchains does not even have a concept for digital currency at all, which implies that no transaction fees can be charged. However, transaction fees can be realized when creating a smart contract ("chaincode") for a Hyperledger Fabric network.

When working in a blockchain environment where no transaction fees are charged, full cost fairness regarding transaction fees is provided for any protocol. However, there might be other costs (e. g., monthly fees for participating in the permissioned blockchain) which should be considered when discussing cost fairness.

E. Minimizing Cost towards Cost Fairness

As mentioned in Sections III-C and IV-D, it is not possible to achieve full cost fairness when only considering on-chain transactions charged with transaction fees. However, for practical application, there is a general interest in reducing costs. For blockchain-based protocols, this implicates the reduction of transaction fees.

Listing 1: Python source code for modelling the behavior of seller and buyer for the SimplePayment (payment after goods release, direct payment without smart contract) protocol for simulation in BDTsim.

```
class SimplePayment(Protocol):
  def execute(self, protocol_path: ProtocolPath, environment: Environment,
              data_provider: DataProvider, seller: Account, buyer: Account, price: int):
    release_goods = protocol_path.decide(seller, 'release goods?', ['yes', 'no'])
    if release_goods == 'yes':  # honest seller
      # At this position, off-chain operations for releasing the goods could be implemented.
      # Since BDTsim does not monitor off-chain operations, implementation is not required by BDTsim.
      pay = protocol_path.decide(buyer, 'pay?', ['yes', 'no'])
      if pay == 'yes':  # honest buyer
        environment.send_direct_transaction(buyer, seller, price)  # payment
      else:  # malicious buyer
        pass  # buyer received the goods but leaves protocol without payment
    else:  # malicious seller
      pass  # seller leaves the protocol unexpectedly, no reaction from buyer
```

As we mentioned in III-C and IV-D, the obstacle for not achieving full cost fairness are the costs to be paid for the first transaction in the protocol. Partial cost fairness, which we can achieve using a container protocol (see Section III-D), implies an imbalance between costs for seller and buyer (see Sections III-C and III-A). In order to reduce the imbalance of payments for a protocol that already achieves partial cost fairness, the first transaction has to be as cheap as possible. Simultaneously, to have the initializing party deposited funds in order to provide partial cost fairness, the first transaction must come along with a funds transfer, as long as no off-chain concept is used for depositing the money. In Ethereum, a simple money transfer to an actual account (not a smart contract) requires exactly 21,000 gas. Transferring money to a smart contract requires at least 21,000 gas, increasing dependently on the number of operations the smart contract method called for the transfer. Therefore, to reduce the imbalance as much as possible, the smart contract method used for initializing the protocol and receiving the money deposit from the initializing party should be as small as possible.

V. Conclusion

This work focuses on the cost in terms of transaction fees which must be paid for executing two-party exchange protocols. For a comparable notation of the costs, we introduced the concept of a *maximum cost matrix*. For cost calculation, we presented BDTsim, our simulation framework for blockchain-based data trading. From the concept of a maximum cost matrix, we derived the definition of *cost fairness*, answering RQ1. We used BDTsim for simulation of the FairSwap protocol, showing that, even if it reaches formal fairness, cost fairness can not be achieved. We presented a general container protocol which can be used to reduce imbalances between the costs of participating parties (answering RQ3) and presented some ideas, how a blockchain-based exchange protocol can be optimized in order to generally reduce protocol execution cost, tackling RQ2.

We figured out smart contract deployment to be a major cost driver. Depending on the protocol, it might be possible to create reusable smart contracts for the protocol, which may initially be more expensive to deploy due to increased complexity. However, subsequent protocol executions do not require to deploy the smart contract again, therefore, the average costs for deploying the smart contract distributed among all protocol executions decrease with an increasing number of protocol executions. In future work, we want to conduct a broader evaluation of existing exchange protocols (e. g., using [10], [8] [17], etc.) for identifying the major cost drivers such as smart contract deployment. Using the results of the evaluation, we aim to reduce the overall cost of exchange protocols as well as the imbalance between the cost to be paid by participating parties.

Furthermore, we want to research if our definition of full cost fairness can be achieved in public blockchains applying approaches known from blockchain-based state channels.

Acknowledgments

This work was partly funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – SFB 1119 – 236615297 (Project S7), by the iBlockchain project (grant nr. 16KIS0902) funded by the German Federal Ministry of Education and Research (BMBF), and by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE, by the Horizon2020 projects AMable, Qu4lity, TRUSTS, and DataPorts, and the BMWi project "Industrie 4.0 Recht-Testbed".

References

[1] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. Fair two-party computations via the bitcoin deposits. *IACR Cryptol. ePrint Arch.*, 2013:837, 2013.

[2] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. Secure multiparty computations on bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP*, pages 443–458, 2014.

[3] N. Asokan. *Fairness in electronic commerce*. PhD thesis, IBM, 1998.

[4] Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. Bitcoin as a transaction ledger: A composable treatment. In *Advances in Cryptology - CRYPTO 2017, Proceedings, Part I*, pages 324–356, 2017.

[5] Iddo Bentov and Ranjit Kumaresan. How to use bitcoin to design fair protocols. In *Proceedings of the 2018 Advances in Cryptology - CRYPTO*, pages 421–439, 2014.

[6] Christian Cachin. Architecture of the hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, volume 310, 2016.

[7] Matteo Campanelli, Rosario Gennaro, Steven Goldfeder, and Luca Nizzardo. Zero-knowledge contingent payments revisited: Attacks and payments for services. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS*, pages 229–243, 2017.

[8] Sergi Delgado-Segura, Cristina Pérez-Solà, Guillermo Navarro-Arribas, and Jordi Herrera-Joancomartí. A fair protocol for data trading based on bitcoin transactions. *Future Gener. Comput. Syst.*, 107:832–840, 2020.

[9] Stefan Dziembowski, Lisa Eckey, and Sebastian Faust. Fairswap: How to fairly exchange digital goods. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS*, pages 967–984, 2018.

[10] Lisa Eckey, Sebastian Faust, and Benjamin Schlosser. Optiswap: Fast optimistic fair exchange. *IACR Cryptol. ePrint Arch.*, 2019:1330, 2019.

[11] Aggelos Kiayias, Hong-Sheng Zhou, and Vassilis Zikas. Fair and robust multi-party computation using a global transaction ledger. In *Proceedings of Advances in Cryptology - EUROCRYPT 2016*, pages 705–734, 2016.

[12] Ranjit Kumaresan and Iddo Bentov. How to use bitcoin to incentivize correct computations. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 30–41, 2014.

[13] Ranjit Kumaresan and Iddo Bentov. Amortizing secure computation with penalties. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 418–429, 2016.

[14] Alptekin Küpçü and Anna Lysyanskaya. Usable optimistic fair exchange. In *Topics in Cryptology - CT-RSA 2010, The Cryptographers' Track at the RSA Conference 2010. Proceedings*, pages 252–267, 2010.

[15] Andrew Y. Lindell. Legally-enforceable fairness in secure two-party computation. In *Topics in Cryptology - CT-RSA 2008, The Cryptographers' Track at the RSA Conference 2008. Proceedings*, pages 121–137, 2008.

[16] Henning Pagnia and Felix C Gärtner. On the impossibility of fair exchange without a trusted third party. Technical report, Technical Report TUD-BS-1999-02, Darmstadt University of Technology, Darmstadt, Germany, 1999.

[17] Eric Wagner, Achim Völker, Frederik Fuhrmann, Roman Matzutt, and Klaus Wehrle. Dispute resolution for smart contract-based two-party protocols. In *IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2019*, pages 422–430, 2019.

[18] Gavin Wood et al. Ethereum: A Secure Decentralised Generalised Transaction Ledger. *Ethereum Project Yellow Paper*, 2014.

[19] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, 1986.