IEEE ICBC'24 1570978742

Fast, Favorable, and Fair Blockchain-based Exchange of Digital Goods using State Channels

N.N.

***Abstract*—When exchanging data with an untrusted counterpart, there is a risk that the counterpart will not behave honestly. Fair exchange protocols provide fairness guarantees to involved parties, e.g., by employing blockchains as trusted third parties. However, blockchain transaction fees and block creation times render such protocols expensive and slow. Furthermore, grieving attacks impose the risk of significant unilateral costs. To improve on all three, we propose a state channel-based fair exchange protocol with a mechanism to prevent grieving attacks. Our protocol lowers the cost of repeating exchanges and increases performance while preserving security guarantees of state-of-the-art fair exchange protocols. Based on a prototypical implementation using the Ethereum blockchain and the Perun state channel framework, we evaluate our protocol about cost and performance showing significant improvements in comparison to the state-of-the-art.**

***Index Terms*—Fair Exchange, Grieving Attack, Blockchain, State Channel, Smart Contract**

I. Introduction

In electronic commerce, *fair exchange protocols* [1], [2] can reduce the risk of being cheated when selling goods to an unknown party, e.g., when selling data over the Internet. Fair exchange protocols require a trusted third party, which moderates the exchange and enforces fairness [3], [4]. Typical examples of such a trusted third party are central institutions, such as notaries. Recent research has shown the possibility of implementing a decentralized trusted third party in the form of a smart contract, eliminating the risk of being cheated by a compromised centralized trusted third party.

State-of-the-art blockchain-based fair exchange protocols [5]–[9] usually only consider the goods to be exchanged for the fairness assessment, but ignore the costs due to the trusted third party, e.g., blockchain transaction fees. Interacting with a blockchain smart contract can raise non-negligible costs, as shown by Lohr et al. [10]. Furthermore, current blockchain-based fair exchange protocols are open to *grieving attacks* [7], [11], which can cause high costs for the faithful party, with the exchange formally still being considered fair.

For practical use in an industrial context, security, cost, and performance are the most significant aspects of an application. Thus, to use a blockchain-based fair exchange protocol in an industrial context, the cost must be reasonable, and, therefore must be included in the consideration. At the same time, it must be ensured that the application provides reasonable performance, e.g., low-latency and high-frequency data exchanges. In addition, risks that lead to unreasonably high costs, such as a successful grieving attack, must be mitigated.

Since most blockchain-based fair exchange protocols have never progressed beyond prototype status, we are not aware of any blockchain-based fair exchange protocol meeting the requirements above. In principle, cost of smart contract execution can be reduced while simultaneously increasing performance of a smart contract using so-called *state channels* [12]–[14], but the possible extent of the improvement is unclear. Furthermore, using state channels alone does not affect the vulnerability to grieving attacks. To fill this gap, we introduce FairSCE, a state channel-based fair exchange protocol. To this end, we pose and answer the following research questions:

RQ1 How can grieving attacks against a blockchain-based fair exchange protocol be prevented?

RQ2 Which cost reduction can be achieved using state channels for a blockchain-based fair exchange protocol?

RQ3 Which performance improvement can be achieved using state channels for a fair exchange protocol?

For answering the research questions, we provide the following contributions:

1) *FairSCE*, a blockchain-based two-party fair exchange protocol, based on FairSwap [6], using state channels and providing a proof-of-concept implementation [15] for the Ethereum blockchain [16] using the Perun [13].
2) *Extra Deposit*, a wrapper contract allowing to preventing grieving attacks for two-party contracts.
3) A comparative evaluation regarding cost and execution time comparing the proposed FairSCE protocol to the original FairSwap [6] protocol and of the Extra Deposit.

In Section II, we present the foundations for our work. In Section III, we present our approach of FairSCE, a two-party fair exchange protocol for digital goods with grieving attack protection and discuss its security in Section IV. We evaluate our approach and compare it to FairSwap in Section V. Finally, we conclude in Section VII.

II. Background

A. Blockchain Smart Contracts

A *blockchain* [17] is a distributed append-only data structure in which state changes are published by appending new *blocks*, which are mutually validated by other blockchain participants against a set of rules. Only if the majority of blockchain participants agrees, a new block is accepted. Due to the append-only characteristics, it is always possible to consecutively track all state changes and validate them.

The rule set for valid state changes can either be part of the blockchain specification (e.g., metadata constraints) or user-defined in the form of a *smart contract*, a program stored on the blockchain. This way, complex rules for valid blocks

can be defined. For example, the Ethereum blockchain [16] provides the *Ethereum Virtual Machine (EVM)*, a Turing-complete environment to execute smart contracts. Whenever a smart contract is executed, all execution parameters are stored on the blockchain. Thus, state changes in the smart contract remain traceable and can be verified by anyone.

Traditionally, a blockchain is related to a *cryptocurrency* such as *Ether* on the Ethereum blockchain that can be used for payments by users or smart contracts. Furthermore, the currency is used to charge fees for blockchain usage (e.g., sending transactions). Fees are measured in *Gas* and calculated on the base of the complexity of the transaction to be executed.

B. FairSwap

For several two-party fair exchange protocols, proof-of-concept smart contracts (FairSwap [6], OptiSwap [7], SmartJudge [8], FastSwap [9], etc.) were published, in which the blockchain acts as a trusted third party. To increase the applicability of our work, we chose FairSwap as reference protocol since, to the best of our knowledge, it is the first blockchain-based protocol providing the guarantee for a fair exchange, and many other works use or are based on FairSwap.

The FairSwap [6] protocol defines a sequence of states to ensure a fair exchange good, e.g., a file, to a buyer [6]. As shown in Fig. 1, a seller sends an encrypted file to a buyer and registers the hashes of the plain file, the encrypted file, and the key needed for decryption in a smart contract (*Initialized*). After validating the registered data, the buyer sends the payment (in the form of cryptographic funds) to the smart contract (*Accepted*), causing the seller to release the decryption key to the smart contract (*Key Revealed*). If the seller sends wrong information (wrong file or key), using the information registered in the smart contract, the buyer can send a *complaint* to the smart contract to reclaim the payment. If the buyer falsely tries to accuse the seller, the same information can be used to detect and discard an invalid complaint. The seller can claim the payment after a timeout, in which no valid complaint has been sent.

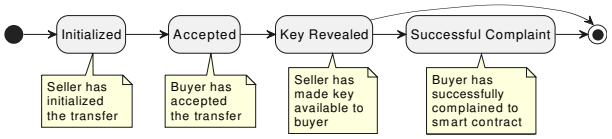

Fig. 1. FairSwap protocol [6] state chart.

Dziembowski et al. have proven the security and fairness (following the fairness definition of Asokan [1]) of FairSwap. However, in the fairness assessment, they only considered the file to be exchanged and the payment paid in return. The fairness assessment lacks consideration of protocol execution cost, which arises for both seller and buyer. A FairSwap protocol execution requires at least four blockchain interactions, whereby the initialization by the seller is the most expensive one with a blockchain transaction fee of approximately $1{,}500{,}000$ Gas (approx. 140 USD as of Dec. 1st, 2023).

C. State Channels

Instead of sending each transaction to a blockchain, parties interacting with each other (e.g., using a fair exchange protocol) can collect transactions off-chain in a so-called *state channel* and only send one aggregated transaction to the blockchain. Thereby, the state channel preserves the security guarantees of a blockchain. This way, waiting times for the blockchain and transaction fees can be avoided. As long as all parties behave faithfully, only the blockchain transactions required to register the aggregated transaction must be waited for and paid. In case of a dispute, the state channel executes the smart contract on-chain to resolve the dispute.

State channels can execute arbitrarily complex smart contracts, usually providing cost savings and increased performance compared to on-chain execution [12]. Referring to Bitcoin and Ethereum as *layer-one* blockchains, state channels can be seen as *layer-two* blockchains [14]. Several state channel concepts have been proposed, such as Perun [13], the Connext Network [18] and the statechannels.org initiative [19].

We focus on *Perun*, as it is the most mature framework and the only one provided with formal proof of its correctness and security, having a ready-to-use smart contract implementation [13]. It consists of two main smart contracts. First, the *AssetHolder* smart contract provides the functionality to open a Perun state channel, deposit funds (*initialization*), and distribute the deposited funds according to the final state (*settlement*). Second, the *Adjudicator* smart contract, which is called by the AssetHolder smart contract for retrieving the final fund distribution, provides functionality to register the final state and a *dispute* mechanism. The Adjudicator decides disputes based on an application-specific smart contract called *app* that provides a set of accepted states and state changes. A Perun state channel passes through the following four phases:

1) Open: When creating a new state channel, all participants agree on the initial state, comprising the initial balances of funds, by mutually signing this state (off-chain, using direct communication, e.g., via the Internet). Afterwards, on-chain transactions are conducted to register the state channel by sending these initial funds to the AssetHolder smart contract.

2) Transact: During the transact phase, app state changes are communicated off-chain between the state channel participants. This is realized by consecutively exchanging signed off-chain messages containing the state of the app and the changed amount of funds as payload.

3) Finalize: At any point in time, channel participants can request to close the state channel (e.g., to initiate a payoff) and to determine the final state of the app. In case of mutual consensus, all participants sign the final app state off-chain and send it to the Adjudicator smart contract. Otherwise, the final state is determined by executing the app on-chain. Each participant registers its final app state with the Adjudicator smart contract. This contract verifies the registered states by calling the app smart contract and bindingly decides on the final state.

4) Settle: After determining the final app state, participants can request a payoff of the funds previously deposited according to the fund's distribution defined by the final app state.

D. Grieving Attacks

A *grieving attack* is particularly relevant in the context of fair exchange protocols, such as FairSwap [6]. In a grieving attack, the attacking party performs actions to the disadvantage of another party (e.g., causing cost for the attacked party) without suffering disadvantages (e.g., own cost) themselves [7], [11]. For example, in FairSwap, a grieving attack can be conducted by a buyer requesting a fair exchange from the seller using a direct communication channel with no or negligible cost for the buyer. According to the FairSwap protocol, the seller initializes the fair exchange by deploying an instance of the FairSwap smart contract to a blockchain with non-negligible costs. If the buyer leaves the exchange with no further interaction, the seller has to bear the cost alone while the buyer has no cost.

III. Fair Exchange of Digital Goods using Blockchain State Channels

To the best of our knowledge, existing fair exchange protocols usually do not consider production environments with simultaneous and fast-repeated execution but reside on a conceptual basis, in some cases, along with formal proof of their security guarantees regarding fairness. This leads to a general lack of evaluation and experience concerning the cost and performance of blockchain-based fair exchange protocols under practical application conditions. Furthermore, grieving attacks are still an open threat.

To this end, we present FairSCE, our production-ready blockchain-based two-party fair exchange protocol with grieving attack protection, which aims for low cost and high performance. Our protocol is based on FairSwap [6], which simultaneously serves as a reference for cost and performance comparison with FairSCE, and uses the Perun state channel framework [13]. The implementation, the code written for evaluation, and all evaluation data, are provided at [15]. FairSCE consists of two smart contracts: the *Extra Deposit* contract for grieving attack prevention and the app smart contract for repeated fair exchanges in a state channel.

A. Extra Deposit Smart Contract

In FairSwap, a grieving attack can be conducted by the buyer by requesting the seller to deploy the smart contract, which causes significant cost. If the buyer deposited a security deposit prior to his fair exchange request, this deposit could be used to compensate the seller. However, since in typical blockchains such as the Ethereum blockchain [16], creating a security deposit also raises costs, it would enable the other party to carry out a grieving attack by first demanding a security deposit and then leaving the protocol itself.

To prevent grieving attacks for blockchain-based interactions that use a deposit mechanism, we introduce the concept of a reusable *Extra Deposit* contract that wraps around the smart contract to be protected against a grieving attack. Instead of individual subsequent deposits of the parties, the Extra Deposit smart contract offers the possibility to provide all

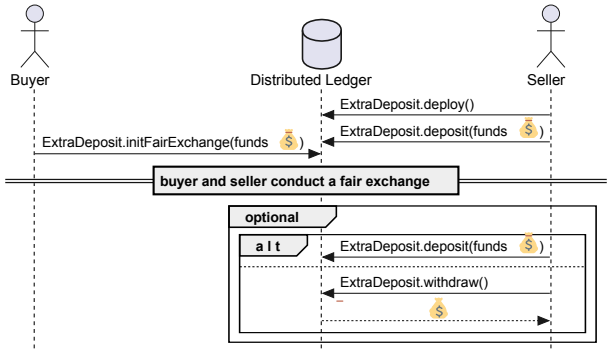

Fig. 2. Extra Deposit smart contract wrapping a fair exchange protocol.

deposits to the wrapped smart contact in the same transaction. As shown in Fig. 2, one party with the intention to perform many exchanges with different parties sets up the Extra Deposit that is not bound to a specific execution of the wrapped smart contract. Other parties intending to start an execution of the wrapped smart contract can request the funds from the Extra Deposit to be transferred to the wrapped smart contract by providing their deposit with their request. This way, the expenses of an attacked party will be refunded in case the other party executes a grieving attack. The Extra Deposit smart contract only needs to be deployed a single time. Therefore, for repeated execution, the deployment cost of Extra Deposit can be neglected. For further exchanges, the seller might need to add additional, as for each fair exchange initialization funds are transferred from the Extra Deposit to the wrapped fair exchange smart contract. In the case of no buyer requests a fair exchange, the seller has the possibility to eventually withdraw his deposits.

B. Reusable Fair Exchange Smart Contract

As the purpose of a Perun state channel is to determine the final funds distribution according to a state channel app execution, we have to identify the computations made to find the final fund's distribution. These computations are reflected by the state transitions of FairSwap in which the final fund's distribution changes. For this, we first analyze the transitions of the original FairSwap smart contract (see Fig. 1) as a starting point for our app definition. Subsequently, we introduce the app contract and its embedding in Perun.

a) Fair Swap Transitions relevant for Perun: To design the state channel app smart contract, we need to identify the transitions of FairSwap that change the fund's distribution. Other transitions, such as initialization and settlement are covered by the Perun smart contracts themselves. Two transitions change the fund's distribution in FairSwap:

Accepted $\rightarrow$ *Key Revealed.* In FairSwap, the buyer sends funds for the payment to the FairSwap smart contract as part of the acceptance of the exchange parameters. After revealing the key, the seller (in case of no further action from the buyer) is now able to claim more funds than before, this represents a change in the fund's distribution.

Key Revealed → Successful Complaint. When the buyer can prove that the seller sent the wrong file key (called *complaint* [6]) the eligibility for the seller to request the payment for that exchange is revoked. Since a successful complaint reduces the amount of funds the seller is eligible to claim, this also represents a change in the fund's distribution.

In FairSwap, the funds distribution does not change in the transition *Initialized→Accepted*, since the buyer is still in control of the funds he sent and can request a refund until the seller reveals the key. Hence, for the adaptation of FairSwap to the app smart contract, it is possible to merge the states *Initialized* and *Accepted*. Since the funds distribution in case of a possible dispute is still identical after merging (no exchange, since neither the buyer can decrypt the file nor the seller gets access to the payment), we retain a fair exchange.

b) Construction of a Perun State Channel App: Based on the fund distribution changing transitions, the states *Accepted*, *Key Revealed*, and *Successful Complaint* are required for our Perun app smart contract. Further, since a Perun state channel can run indefinitely, if all participants mutually agree on and sign app state updates, an unlimited number of exchanges can be conducted by allowing a transition from *Key Revealed* back to *Accepted*. As soon as there is a disagreement, the state channel will go to the *Finalize* phase and the app smart contract will be executed on-chain. Bringing together the Perun state channel phases with app states, this results in the complete state chart in Fig. 3.

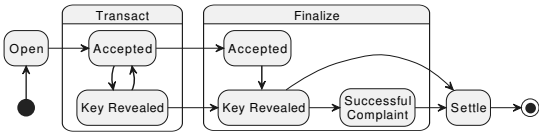

Fig. 3. Full state chart of FairSCE with app states in context of Perun phases.

IV. Protocol Security

A. FairSCE Protocol Security

Considering FairSwap as a baseline, there is one central security guarantee that fair exchange protocols must fulfill: An honest party will not experience any disadvantages in case of interacting with a cheating party. For exchanges using the plain FairSwap protocol, Dziembowski et al. have shown that fairness is guaranteed when at least one party follows the protocol [6]. We must consider two differences between the original FairSwap protocol and FairSCE. First, we must show that changing the execution context from direct interaction with a blockchain to a state channel has no negative security impact, and second, our adaptions for executing FairSwap in a state channel do not threaten the protocol security.

1) Smart Contract Execution Context: We extended the smart contract execution context from a purely on-chain execution to an off-chain execution in a state channel. Since Perun provides the same security guarantees for apps [12] as a blockchain provides for on-chain smart contracts [16], [17], extending the execution context does not make a difference for the security consideration.

2) App States: FairSwap and FairSCE differ in their states and transitions. Following the security discussion of FairSwap, we must show equivalence for three security criteria: *Termination*, *Sender Fairness*, and *Receiver Fairness* [6].

a) Termination: Since FairSwap has a limited set of states without loops and allows to switch to a subsequent state at the latest after a timeout, it is guaranteed to terminate. While our approach contains one loop, any party can leave the loop between the *Accepted* and *Key Revealed* state in the *Transact* phase at any time by registering a dispute, which will progress the state machine to the *Finalize* phase. In the *Finalize* phase, the FairSCE app has the same states and transitions as in FairSwap except for the *Initialization* and no loops exist. Both FairSwap and FairSCE allow to switch to a subsequent state at the latest after a timeout has elapsed. Also, equally to FairSwap, as soon as being in the *Finalize* phase, all state transitions are executed on-chain.

b) Sender Fairness: In FairSwap, sender fairness means that the receiver cannot decrypt the exchanged file unless the honest sender is guaranteed to be paid. The decryption key is only exchanged after the payment has been locked by sending it to the Perun smart contract. Like FairSwap, the seller can claim the funds when two conditions are met: First, the key must be available to the buyer in a publicly verifiably manner, which is done when the *Key Revealed* state of the *Finalize* phase is reached. Second, a timeout must be passed in which the buyer did not send a complaint, which he can only do successfully if the key or the transferred data is incorrect.

c) Receiver Fairness: Like FairSwap, the buyer must lock the funds for payments before he receives the file and can verify its correctness. To this end, a mechanism is required to withdraw the funds in case of a failed file transfer (no file or wrong file or wrong key), which is realized in FairSwap by *proof of misconduct* [6]. Since we adopted this mechanism without modification to our approach (as part of the on-chain *Finalize* phase), the same guarantees apply.

In summary, neither the changed execution context nor our adaptation of the smart contract to an efficient execution in a state channel has a negative impact on the security of FairSCE. Possible attacks such as majority attacks [20] or collusion attacks [21] are either general attack on blockchains or are handled by the leveraged state channel approach [12], [13]. When using FairSCE, users are provided with the same security guarantees as when using FairSwap.

B. Grieving Attack Prevention

When using the Extra Deposit smart contract, the following cases may occur, all of which exclude a grieving attack against both the seller and the buyer. For all cases listed below, we assume the seller already deployed the Extra Deposit smart contract, waiting for a buyer to request a fair exchange.

a) No buyer: When a seller sets up the Extra Deposit smart contract, including a first security deposit but no buyer uses it, the seller can withdraw the funds deposited at any time, but initialization, deposit, and withdrawal are charged. Despite there are no compensating revenues, e.g., successful

fair exchanges, the decision to set up and deposit funds is alone in preparation for possible future transactions, not necessarily at the request of an individual buyer. Therefore, this is not a grieving attack but a general investment (comparable to the deployment cost of a web shop). Even if the Extra Deposit setup is conducted as a reply to a request of a single grieving buyer, the Extra Deposit can still be used for further buyers later in time. Even if buyers repeatedly request a fair exchange without participating, the seller has no additional cost for subsequent requests, as the Extra Deposit smart contract is already deployed and provided with a deposit.

b) Buyer initializes the fair exchange, then behaves unfaithfully: After a seller prepares the Extra Deposit smart contract and a buyer initializes a fair exchange, the Extra Deposit contract forces the buyer to pay a predefined deposit during the initialization. If the buyer unfaithfully leaves the fair exchange protocol, its deposit can be used to compensate the seller. Therefore, the buyer cannot conduct a grieving attack if the seller follows the wrapped smart contract, e.g., FairSCE.

c) Buyer initializes the fair exchange, then the seller behaves unfaithfully: Since the Extra Deposit smart contract provides a pre-defined amount of funds from the deposit of the seller to the wrapped smart contract, e.g., for the deposit of FairSCE, together with the deposit of the buyer, the buyer can be compensated if the seller starts to behave unfaithfully, e.g., leaving the protocol unfaithfully or by providing a file different from the one expected by the buyer. Since the buyer can investigate the Extra Deposit upfront, the seller is not able to conduct a grieving attack.

V. Evaluation

In our evaluation, we investigate the cost incurred due to the *Extra Deposit* and its practical feasibility (RQ 1) and examine the difference between FairSwap and FairSCE in terms of cost, as asked for in RQ 2, and performance, as asked for in RQ 3.

A. Evaluation Setup

1) Cost Evaluation (RQ 1 & RQ 2): In FairSwap and FairSCE, each interaction with the smart contract of the fair exchange protocol incurs transaction costs when executed on the underlying blockchain. To show the practical feasibility of Extra Deposit, we must compare its additional cost with the cost arising due to grieving attacks. To investigate the cost incurred by the two fair exchange protocols and of Extra Deposit, we invoke their smart contract methods (for FairSCE including Extra Deposit) in sequences reflecting different behaviors of seller and buyer, while measuring the cost of each interaction with the smart contract.

The behavior of the parties in a fair exchange can be distinguished into those parties that follow the protocol (referred to as *faithful* behavior) and those that deviate from the actions specified by the fair exchange protocol (referred to as *unfaithful* behavior). Since faithful behavior is assumed to be the most common case, the cost of joint faithful execution is important to know for repeated use in a production environment.

Whenever the protocol requires an action from a party, it may unfaithfully perform an action different from the expected one or do nothing, the latter being considered *unfaithfully leaving*. However, due to validations performed by the smart contract, an unfaithful party cannot perform arbitrary actions. For example, in FairSwap, being in the *Accepted* state, the only possible action for the seller to perform is to reveal the key he committed to by adding the cryptographic hash value of the key. Any other action will be rejected by the smart contract.

We assume both seller and buyer to act rationally, and therefore, apart from possibly trying to cheat the other party (which could raise their benefit in case the other party does not act rationally), they never behave in a way that is inevitably to their disadvantage. Also, we do not consider cases in which both parties simultaneously do not follow the protocol since this cannot be considered protocol execution. For this reason, there is only a limited number of unfaithful behaviors to consider besides unfaithful leaving, namely those that the smart contract cannot directly validate and reject:

Buyer leaves unfaithfully. After the exchange has been initialized (FairSwap smart contract deployment or Perun state channel opening), the buyer leaves without further action. If the initialization is charged with non-negligible cost (such as the smart contract deployment done by the seller), this represents a grieving attack.

Seller leaves unfaithfully. The seller leaves at an arbitrary point in time after he has confirmed to provide the data requested by the buyer and the buyer provided the payment (to the FairSwap smart contract or as state channel transaction). Even if the seller must bear costs, this can be considered rational behavior if the disadvantages caused to the buyer are higher than the cost for the seller. For example, doing so, the seller can temporarily cause the buyer to block funds that the buyer cannot use for another exchange at the same time.

Seller provides wrong/modified file. The seller encrypts a file different from the file requested by the buyer and subsequently sends it to the buyer. After the seller reveals the key, the buyer can decrypt and discover the mismatch and conduct a complaint (which is wrapped in a dispute when a Perun state channel is used). After the buyer has submitted a valid proof of misbehavior of the buyer to the fair exchange smart contract, he can request a refund of the payment.

As already mentioned in the FairSwap [6] paper, the cost of conducting a complaint depends on the size of the input to the complaint method invoked to submit the complaint to the smart contract. Therefore, we executed all simulations with different file sizes in powers of two, starting with 1 KiB.

2) Performance Evaluation (RQ 3): To answer RQ 3, we aim to determine the performance in terms of protocol executions per time unit of FairSwap and FairSCE. For this purpose, we measured the performance of FairSwap and FairSCE conducting recurring file exchanges.

Since we want to determine the maximum performance and an unfaithful party can slow down protocol execution (e.g., by idling for some time before reacting to transactions of the other party) or just stop its participation in the protocol, we

only consider the case where both parties cooperatively follow the exchange protocol faithfully. For each file exchange, we measure the time used for initialization, encryption of the file to be exchanged, file transfer, decryption by the buyer, and validation if the decrypted file equals the expected file.

FairSwap has two ways to complete a fair exchange successfully: After the key has been revealed, the buyer can optionally confirm the correctness of the key to the smart contract, which then releases the payment to the seller. Without confirmation, the seller can always claim the funds after a timeout passes. Usually, a rational buyer would not send the confirmation to save transaction fees but for maximizing performance this is necessary. Therefore, we let the buyer send the confirmation to prevent the sender from waiting for the timeout.

For FairSCE, we assume both parties to be interested in recurring file exchanges using the same state channel. To this end, we include a single initial contract deployment on the blockchain in our evaluation. At the end of each fair exchange, we assume the seller to settle the state channel to be able to claim the funds since he has the highest incentive to do so (since the buyer already got the files). However, which party conducts the settlement does not influence the performance.

B. Implementation

For the evaluation, we used the existing prototypical FairSwap implementation and implemented FairSCE with Extra Deposit. Both are implemented using *Solidity* [22].

FairSwap offers the buyer three different types of complaints (*root*, *node*, and *leaf*) that slightly differ implementation-wise in their cryptographic proof, therefore being subject to different cost and performance. We refer to Dziembowski et al. [6] for technical details. As our implementation of FairSCE is based on FairSwap, these aspects equally apply.

To simulate the different behaviors of seller and buyer and to monitor their interactions with the blockchain, we implemented *BFEBench* [15], a fair exchange benchmark tool. BFEBench allows to define different behaviors for seller and buyer to be spawned as individual processes, which can directly communicate with each other and access a blockchain instance that is shared between seller and buyer.

For the evaluation we used Ganache [23], an Ethereum-compatible blockchain simulation software, that uses the same internal pricing model as Ethereum. To shorten the runtime of the cost evaluation, we configured the Ganache blockchain to create a new block every second, which does not influence the cost compared to a real-world protocol execution. For the performance evaluation, we configured the Ganache blockchain to create a block every 15 seconds, corresponding to the usual interval between two blocks of the Ethereum blockchain. Thus, the rate of blockchain smart contract interactions of our evaluation (one interaction per block) corresponds to the best-case rate on the Ethereum blockchain.

For each experiment, two single-threaded processes representing seller and buyer were spawned, both connecting to the same Ganache instance that was not used by other processes in parallel. Consequently, our results need to be interpreted as best-case and might be worse when using the protocols with a shared blockchain. The experiments were run on a server with two Intel(R) Xeon(R) E5-2690 CPUs (8 cores/16 threads each) with 2.90 GHz and 128 GB of RAM.

TABLE I
COST OF FILE-SIZE-INDEPENDENT OPERATIONS AND INTERACTIONS WITH THE FAIRSWAP AND FAIRSCE SMART CONTRACT MEASURED IN *Gas*.

		Seller		Buyer	
		Cost	σ	Cost	σ
FairSwap	deploy	1,495,092	698	-	-
	accept	-	-	36,080	14
	revealKey	58,984	18	-	-
	refund	29,372	3,568	32,728	23
FairSCE	ExtraDeposit.deploy	493,342	3	-	-
	ExtraDeposit.deposit	21,055	0	-	-
	ExtraDeposit.initStateChannel	-	-	94,153	4
	AssetHolder.deposit	-	-	46,308	4
	Adjudicator.register (Dispute)	116,242	21	115,659	492
	Adjudicator.progress (Dispute)	106,973	26	107,292	27
	Adjudicator.conclude (Dispute)	132,475	2,017	121,479	11,967
	Adjudicator.concludeFinal (Consent)	163,636	9,325	-	-
	AssetHolder.withdraw	44,727	12	44,726	12
	ExtraDeposit.withdraw	30,844	6	-	-

C. Evaluation Results

In this section, we discuss the results of our experiments.

1) Cost Evaluation: We simulated the different seller and buyer behaviors for FairSwap and FairSCE with Extra Deposit to determine the respective cost of each interaction with the underlying blockchain. Table I shows all detailed measurements on the granularity of individual smart contract methods.

a) Cost for Initializing a Fair Exchange: The static overhead for deploying FairSwap or FairSCE has a significant impact on the cost for exchanges, which we discuss in what follows. To be able to offer a fair exchange using FairSCE, Perun (5,844,671 Gas) and the FairSCE app smart contract (3,147,654 Gas) needs to be deployed once. These contracts can be reused by any parties for an arbitrary number of FairSCE state channels. To initialize a state channel for a FairSCE fair exchange, the buyer has to pay 46,308 Gas. This state channel can be reused for an arbitrary number of fair exchanges. In FairSwap, initialization cost (1,495,092 Gas) has to be paid per exchange by the seller. Therefore, even if the seller would have to deploy Perun and FairSCE smart contracts, initialization of FairSCE would be cheaper for the seller starting from the 7^{th} exchange. Since the seller typically passes on his costs to the buyer anyway, using FairSCE is beneficial for the buyer as well due to lower total cost.

b) Cost for Faithful Fair Exchanges: In addition to the initialization cost, both protocols incur costs for successfully completing fair exchanges. In FairSwap, these costs are 88,356 Gas for the seller and 36,080 Gas for the buyer per fair exchange. In FairSCE, these costs arises per state channel, which can be used for any number of fair exchanges, and are 208,363 Gas for the seller and 46,308 Gas for the buyer. Therefore, as shown in Fig. 4, even if only one fair exchange is carried out per state channel, the total cost for the seller using FairSCE

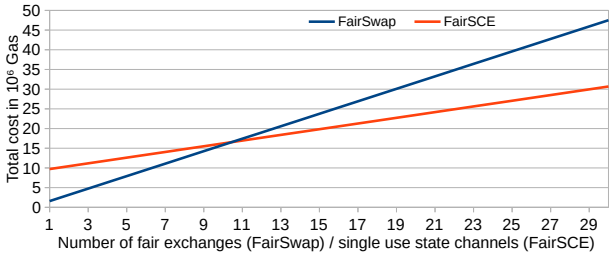

Fig. 4. Total cost for fair exchanges in FairSwap and FairSCE including contract deployment in relation to the number of conducted exchanges.

(including smart contract deployment) is cheaper compared to FairSwap from the eleventh faithful exchange onwards.

c) Cost for Conducting a Complaint: In the case the seller behaves unfaithfully by providing the buyer with the wrong file or key, costs arise for conducting a complaint. Table II shows the cost for the buyer to conduct a complaint in case of a cheating seller. The cost of a complaint depends on its type, for technical details we refer to Dziembowski et al. [6], and the size of the file received. Fig. 5 visualizes these costs. It can be seen that the cost depends logarithmically on the size of the file received from the seller.

d) Cost for Grieving Attack Prevention: Finally, we discuss the cost for using the grieving attack prevention and how easily an attacker could cause higher cost if it is not used. The costs for grieving attack prevention consist of setting up the Extra Deposit smart contract (493,342 Gas) and conducting an initial deposit (21,055 Gas). The cost for initializing a state channel for FairSCE increases by 47,845 Gas. Ongoing cost per exchange can arise for depositing additional funds in the Extra Deposit (21,055 Gas) or withdrawing deposited funds (30,844 Gas). The costs for the actual exchange do not change. Assuming the worst case of only one customer, grieving attack prevention creates additional cost of 562,242 Gas a single time. For the grieving attack prevention to be useful in practice, an attacker must be able infer higher cost. Without Extra Deposit, the FairSCE initialization could be done by any party (46,308 Gas) which are the cost at stake per exchange in case of grieving. Summarized, if an adversarial, e.g., buyer can spawn identities at negligible cost, as it is, e.g., with the Ethereum blockchain, the financial damage to the seller exceeds the Extra Deposit cost already after the 12th successful attack.

To summarize the performance evaluation, Table III aggregates the individual costs of the parties into a comparative overview of FairSwap and FairSCE (for simplicity limited to a file size of 1 MiB). In the faithful case, the costs are more balanced between seller and buyer in FairSCE compared to FairSwap. To answer RQ 2, especially for faithful exchanges, our evaluation shows a reduction of total cost as well as individual cost for the seller because FairSCE has reusable smart contracts and does not incur costs for each exchange.

In case of a complaint, the buyer has higher cost in FairSCE, which could be seen as a significant disadvantage at first sight. However, when Extra Deposit is used, the cost information collected allows us to calculate the minimum deposits of each

TABLE II
FILE-SIZE-DEPENDENT COST OF COMPLAINTS MEASURED IN *Gas*.

		root		node		leaf	
		Cost	σ	Cost	σ	Cost	σ
FairSwap	1KiB	45,424	10	51,058	22	52,752	17
	2KiB	46,301	10	52,684	21	54,373	20
	4KiB	47,153	13	54,432	20	56,080	18
	8KiB	47,985	12	56,108	24	57,753	19
	6KiB	48,860	12	57,789	21	59,481	21
	32KiB	49,778	13	59,431	22	61,127	21
	64KiB	50,586	16	61,191	20	62,835	21
	128KiB	51,438	16	62,878	24	64,508	21
	256KiB	52,253	14	64,521	25	66,255	23
	512KiB	53,169	18	66,230	27	67,945	26
	1MiB	54,044	18	67,889	25	69,642	22
FairSCE	1 KiB	82,245	17	94,822	22	99,234	24
	2 KiB	83,662	18	97,625	24	102,027	21
	4 KiB	85,070	17	100,404	23	104,832	22
	8 KiB	86,486	17	103,220	23	107,624	24
	16 KiB	87,867	20	105,984	26	110,391	25
	32 KiB	89,298	22	108,776	32	113,211	28
	64 KiB	90,681	20	111,579	25	115,978	25
	128 KiB	92,098	19	114,387	29	118,799	29
	256 KiB	93,515	20	117,181	33	121,589	29
	512 KiB	94,917	23	119,970	28	124,374	29
	1 MiB	96,304	20	122,768	25	127,195	30

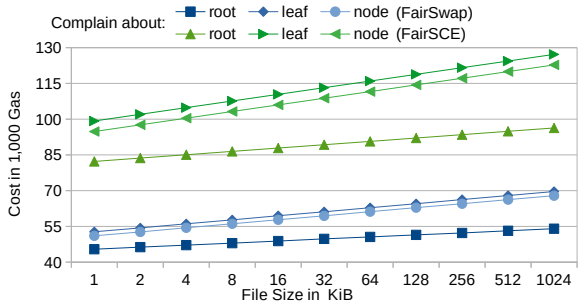

Fig. 5. Logarithmic growth of the cost a buyer must pay for conducting a complaint in FairSwap and FairSCE depending on the size of the file received.

party needed to be able to compensate the faithful if the other party behaves unfaithfully. For initializing FairSCE with Extra Deposit, the amount of the initial deposit should be as high as the fees at stake for a buyer when initializing a fair exchange in case the seller leaves unfaithfully, resulting in a total of 376,017 Gas including the dispute. For complaints, since the file size is the only non-static factor influencing cost, in particular the insight that the cost for a complaint grow logarithmically with the file size, allows for reasonable estimations that are sufficient even for unexpectedly large files. For example, we estimate the cost for leaf complaint for a file of 1TiB to cost approx. 181,000 Gas. This results in a total of 557,017 Gas to be able to compensate for a complaint of a 1 TiB file. Since only one of the two cases can occur, the seller must provide the Extra Deposit smart contract with sufficient funds to compensate for 557,017 Gas.

2) Performance Evaluation: For the performance evaluation, we measured the time needed for multiple recurring file exchanges using FairSwap and FairSCE. Table IV shows the average time consumed per exchange out of 100 consecutive fair exchanges for both protocols, using different file sizes.

The execution time of FairSwap is independent of the file

TABLE III
MAXIMUM COSTS OF FAIRSWAP AND FAIRSCE WITH EXTRA DEPOSIT FOR A SINGLE FAIR EXCHANGE.

	File Size	FairSwap cost		FairSCE cost	
		Seller	Buyer	Seller	Buyer
faithful exchange	all	1,583,448	36,080	722,760	94,153
unfaithful seller	1 MiB	-	105,722	-	503,212
unfaithful buyer	1 MiB	1,495,092	-	807,841	-

TABLE IV
NUMBER OF EXCHANGES PER MINUTE FOR FAIRSWAP AND FAIRSCE.

	FairSwap		FairSCE		
	Avg. Time	Perf.	Avg. Time	Perf.	Speedup
1 KiB	1m 0s	1.0/min	1s	50.2/min	50.2
2 KiB	1m 0s	1.0/min	1s	49.9/min	50.0
4 KiB	1m 0s	1.0/min	1s	50.3/min	50.3
8 KiB	1m 0s	1.0/min	1s	50.2/min	50.2
16 KiB	1m 0s	1.0/min	1s	50.1/min	50.2
32 KiB	1m 0s	1.0/min	1s	44.7/min	44.7
64 KiB	1m 0s	1.0/min	2s	36.5/min	36.5
128 KiB	1m 0s	1.0/min	2s	26.7/min	26.7
256 KiB	1m 0s	1.0/min	4s	16.7/min	16.7
512 KiB	1m 0s	1.0/min	6s	9.8/min	9.8
1 MiB	1m 0s	1.0/min	12s	5.1/min	5.1
2 MiB	1m 15s	0.8/min	24s	2.5/min	3.1
4 MiB	1m 33s	0.6/min	55s	1.1/min	1.7
8 MiB	2m 49s	0.4/min	2m 18s	0.4/min	1.2
16 MiB	9m 44s	0.1/min	9m 04s	0.1/min	1.1

size for files between 1 KiB and 1 MiB and always one minute for each exchange. The reason behind this is that the computational work of FairSwap (computing and validating cryptographic hash values, encrypting and decrypting the file to be exchanged) is executed faster than it takes for Ethereum to create a new block. After the computational work, a party must wait for the next block, which takes up to 15 seconds. Since FairSwap requires four interactions, each exchange it takes at least one minute. Starting from 2 MiB, the computational work exceeds the block creation time of 15 seconds and, therefore, results in increased durations.

For FairSCE, the block creation time is not a limiting factor, as exchanges are conducted using the state channel. Once the state channel is created, only available resources (e.g., CPU power or Internet connection) limit the performance.

To answer RQ 3, concerning the extent to which we can reduce the time required for a fair exchange, this extent depends on the size of the files to be exchanged. For small files with a size below 32 KiB, we can reduce the required time by around 98% by avoiding waiting for block creations.

3) Threats to Validity: In general, the experiment is reproducible except for a slight variation in cost due to technical reasons that can be neglected. This slight variation is caused by a different price charged by the Ethereum blockchain for bytes equaling zero compared to bytes not equaling zero. The cost calculations of Ganache could differ from those of Ethereum. Since Ganache is widely used in the blockchain community to develop, test, and debug Ethereum smart contracts, realistic results can be assumed. Running all three processes (seller, buyer, and Ganache) on the same host machine neglects the influence of a remote connection. Since Internet connections usually have low latencies compared to latencies due to waiting for blocks, this can be neglected. The assumption that transactions sent to the blockchain will always be accepted in the next created block might not hold. On the Ethereum blockchain, a transaction can have to wait several blocks to be included. This can result in poorer performance or in higher costs for FairSwap but does not affect faithful exchanges using FairSCE. Finally, due to the choice of FairSwap and Perun as a reference, the observed outcomes might not apply to other state channels or fair exchange protocols, or other blockchains.

VI. RELATED WORK

Beside our work, multiple other approaches base on or aim at improving FairSwap [6]. E.g., FastSwap [9] and OptiSwap [7] introduce interactive challenge-response procedures for disputes to reduce overhead for honest parties in the optimistic case. SmartJudge [8] reduces blockchain transaction fees per protocol execution by adding a less costly *mediator* contract to an existing *verifier* contract (e.g., FairSwap). The verifier contract is assumed to be more expensive to use as the mediator contract and will only be involved in case of a dispute. Each of these examples requires repeated on-chain interaction per protocol iteration. Therefore, performance and cost depend on the underlying layer-one blockchain.

Besides Layer-Two approaches [14] such as Perun [13], so-called *commit chains* can be used [14], [24], [25], which is a small and low-cost blockchain that used an established blockchain as trust anchor and can be used to reduce the number of interactions with the established blockchain.

Instead of blockchains with non-negligible fees, other ledger concepts can be used. E.g., the *EOSIO blockchain* [26] eliminates transaction fees, instead, smart contract developers must buy capacity (called *RAM*) for their smart contracts to be executed. Alternatively, in *IOTA* [27], instead of paying transaction fees, each party intending to send a transaction to the IOTA network must do a small Proof-of-Work computation. Similar to the funds needed to pay for blockchain transaction fees, CPU time is also a limited resource and cannot be scaled up without a significant investment.

VII. CONCLUSION AND OUTLOOK

In this work, we have presented FairSCE, a blockchain-based two-party fair exchange protocol using state channels for lowering the transaction cost on blockchains and increasing performance. Furthermore, we introduced Extra Deposit as a mechanism for grieving attack prevention (RQ 1). We demonstrated the feasibility of FairSCE in our evaluation concerning the cost and performance inferred by executing FairSwap and FairSCE (RQ 2 and 3). We show significantly lower transaction costs for the faithful case considering data of different sizes and an increase in performance, especially for repeated exchanges small files. In unfaithful cases, FairSCE allows to refund the cheated party.

REFERENCES

[1] N. Asokan, "Fairness in electronic commerce," Ph.D. dissertation, University of Waterloo, 1998.

[2] H. Pagnia, H. Vogt, and F. C. Gärtner, "Fair Exchange," *The Computer Journal*, vol. 46, no. 1, pp. 55–75, Jan. 2003.

[3] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *Journal of the ACM*, vol. 32, no. 2, pp. 374–382, Apr. 1985.

[4] H. Pagnia, "On the Impossibility of Fair Exchange without a Trusted Third Party," Darmstadt University of Technology, Department of Computer Science, Tech. Rep. TUD-BS-1999-02, 1999.

[5] S. Delgado-Segura, C. Pérez-Solà, G. Navarro-Arribas, and J. Herrera-Joancomartí, "A fair protocol for data trading based on Bitcoin transactions," *Future Generation Computer Systems*, vol. 107, pp. 832–840, Jun. 2020.

[6] S. Dziembowski, L. Eckey, and S. Faust, "FairSwap: How To Fairly Exchange Digital Goods," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. Toronto Canada: ACM, Oct. 2018, pp. 967–984.

[7] L. Eckey, S. Faust, and B. Schlosser, "OptiSwap: Fast Optimistic Fair Exchange," in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*. Taipei Taiwan: ACM, Oct. 2020, pp. 543–557.

[8] E. Wagner, A. Völker, F. Fuhrmann, R. Matzutt, and K. Wehrle, "Dispute Resolution for Smart Contract-based Two-Party Protocols," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, May 2019, pp. 422–430.

[9] M. Hall-Andersen, "FastSwap: Concretely Efficient Contingent Payments for Complex Predicates," https://eprint.iacr.org/2019/1296, 2019.

[10] M. Lohr, K. Skiba, M. Konersmann, J. Jürjens, and S. Staab, "Formalizing Cost Fairness for Two-Party Exchange Protocols using Game Theory and Applications to Blockchain," in *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2022.

[11] S. Eskandari, S. Moosavi, and J. Clark, "SoK: Transparent Dishonesty: Front-Running Attacks on Blockchain," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, A. Bracciali, J. Clark, F. Pintore, P. B. Rønne, and M. Sala, Eds. Cham: Springer International Publishing, 2020, pp. 170–189.

[12] S. Dziembowski, S. Faust, and K. Hostáková, "General State Channel Networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. Toronto Canada: ACM, Oct. 2018, pp. 949–966.

[13] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski, "Perun: Virtual Payment Hubs over Cryptocurrencies," in *2019 IEEE Symposium on Security and Privacy (SP)*. San Francisco, CA, USA: IEEE, May 2019, pp. 106–123.

[14] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, "SoK: Layer-Two Blockchain Protocols," in *Financial Cryptography and Data Security*, J. Bonneau and N. Heninger, Eds. Cham: Springer International Publishing, 2020, vol. 12059, pp. 201–226.

[15] Anonymized for double-blind reviews.

[16] G. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," https://ethereum.github.io/yellowpaper/paper.pdf, Jun. 2022, accessed on 2023-02-06.

[17] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," Tech. Rep., 2008.

[18] "Connext Network," https://connext.network, accessed on 2023-11-27.

[19] "Nitro State Channels Network," https://statechannels.org/, accessed on 2023-11-27.

[20] S. Dey, "Securing Majority-Attack in Blockchain Using Machine Learning and Algorithmic Game Theory: A Proof of Work," in *2018 10th Computer Science and Electronic Engineering (CEEC)*, Sep. 2018, pp. 7–10.

[21] S. Thakur and J. G. Breslin, "Collusion Attack from Hubs in The Blockchain Offline Channel Network," in *Mathematical Research for Blockchain Economy*, P. Pardalos, I. Kotsireas, Y. Guo, and W. Knottenbelt, Eds. Cham: Springer International Publishing, 2020, pp. 31–44.

[22] "Solidity Programming Language," https://soliditylang.org/, accessed on 2023-11-27.

[23] "Truffle Suite One Click Blockchain," https://trufflesuite.com/ganache/, accessed on 2023-11-27.

[24] R. Khalil and A. Gervais, "Revive: Rebalancing Off-Blockchain Payment Networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. Dallas Texas USA: ACM, Oct. 2017, pp. 439–453.

[25] R. Khalil, A. Zamyatin, G. Felley, P. Moreno-Sanchez, and A. Gervais, "Commit-Chains: Secure, Scalable Off-Chain Payments," https://eprint.iacr.org/2018/642, 2018.

[26] "EOSIO Blockchain," https://eos.io/, accessed on 2023-11-27.

[27] S. Popov, "The Tangle," https://bit.ly/42os8QA, Oct. 2017, accessed on 2023-06-08.