SmartZKCP: Towards Practical Data Exchange Marketplace Against Active Attacks

Xuanming Liu[a,b,*], Jiawen Zhang[a,b,*], Yinghao Wang[a,b], Xinpeng Yang[a] and Xiaohu Yang[a,b,**]

[a] *College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, Zhejiang, China*
[b] *The State Key Laboratory of Blockchain and Data Security, Hangzhou 310027, Zhejiang, China*

ARTICLE INFO

Keywords:
Blockchain
Fair exchange
Data marketplace
Zero knowledge

ABSTRACT

The trading of data is becoming increasingly important as it holds substantial value. A blockchain-based data marketplace can provide a secure and transparent platform for data exchange. To facilitate this, developing a fair data exchange protocol for digital goods has garnered considerable attention in recent decades. The Zero Knowledge Contingent Payment (ZKCP) protocol enables trustless fair exchanges with the aid of blockchain and zero-knowledge proofs. However, applying this protocol in a practical data marketplace is not trivial.

In this paper, several potential attacks are identified when applying the ZKCP protocol in a practical public data marketplace. To address these issues, we propose SmartZKCP, an enhanced solution that offers improved security measures and increased performance. The protocol is formalized to ensure fairness and secure against potential attacks. Moreover, SmartZKCP offers efficiency optimizations and minimized communication costs. Evaluation results show that SmartZKCP is both practical and efficient, making it applicable in a data exchange marketplace.

1. Introduction

The rapid increase in big data has led to the swift development of data exchange marketplaces. Research on these marketplaces has been extensive. A significant challenge within the data marketplace is ensuring the fairness of data exchanges. Consider a scenario where Alice has a Sudoku puzzle solution s to sell, and Bob wishes to purchase s with a payment v. The primary concern is how to ensure that in an atomic exchange, Alice receives the payment from Bob, and Bob receives the correct good s from Alice. It has been established that strong fairness is unattainable without the assistance of a trusted third party (TTP) [18] until the emergence of blockchain technology.

The Zero Knowledge Contingent Payment (ZKCP) protocol, proposed by Gregory Maxwell [15], is a blockchain-based data exchange solution that ensures fairness in transactions. In ZKCP, Alice first encrypts the digital good s as ciphertext z and sends it to Bob along with the hash of the encryption key $h_k := \mathsf{SHA256}(k)$. Bob then establishes a script called Hashed Time Lock Contract (HTLC) on the blockchain, locking the payment v and the hash h_k into the script. The script is designed to release the payment to Alice only if she reveals the key k such that $h_k = \mathsf{SHA256}(k)$ within a specified time frame. Upon revealing k, Alice can claim the payment v from the script, and Bob can use the key to decrypt z and obtain the good s. Here the blockchain and the script serve as somewhat a decentralized TTP, ensuring the fairness and atomicity of the exchange. Furthermore, Alice must convince Bob, using an expensive zero-knowledge (ZK) proof [19], that z indeed contains the desired digital good s and that the preimage of h_k is the correct encryption key, without revealing any information about the good s. For a detailed illustration of the protocol, refer to Figure 1.

It appears feasible to implement the ZKCP protocol between the buyer and seller in a peer-to-peer data exchange. However, several possible challenges emerge when applying the protocol in a practical public data marketplace. In this work, we identify three potential risks associated with ZKCP in such environments:

- **Risk of DoS attack.** A malicious buyer might repeatedly request goods from the seller and then cancel the transaction after the seller has expended significant effort in producing the ZK proof. This behavior can deplete the seller's computational resources.
- **Risk of data breach.** In ZKCP, once the key k is revealed, it becomes completely public. Any eavesdropper monitoring the communication channel between the buyer and the seller could obtain the ciphertext and decrypt it with the revealed key to access the confidential data.
- **Risk of privately verified proof.** In ZKCP, the proof is verified privately by the buyer, which poses risks in a data marketplace where reputation is crucial. A malicious buyer may falsely claim that the seller provided incorrect goods or proofs, potentially damaging the seller's reputation.

Looking ahead, this work illustrated that these risks could be exploited by malicious adversaries to disrupt the data marketplace, adversely affecting the interests of honest participants. Another potential optimizable point in data marketplaces is that multiple buyers may be interested in the same digital good simultaneously. In ZKCP, this requires the seller to generate separate proofs for different buyers, even though the item is identical, which is inefficient. Furthermore, the vanilla ZKCP uses $\mathsf{SHA256}$ for hashing the

[*] These authors contributed equally to this work.
[**] The corresponding author.

✉ hinsliu@zju.edu.cn (X. Liu); yangxh@zju.edu.cn (X. Yang)
ORCID(s): 0009-0005-9115-4316 (X. Liu)

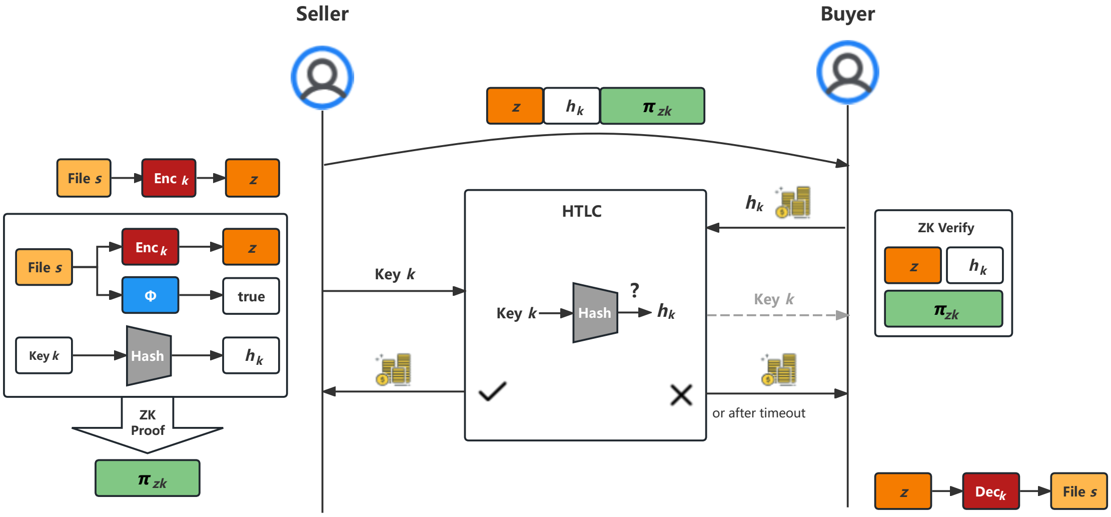


Figure 1: An illustration of the ZKCP protocol. Here HTLC stands for Hashed Time Lock Contract, the functionality of which is described in Fig. 3.

encryption key and employs a stream cipher derived from SHA256 for symmetric encryption, making the generation of ZK proofs costly.

In this work, we care about both security and efficiency concerns. For security, we analyze practical attacks on the vanilla ZKCP protocol and propose modifications to rectify these flaws. For efficiency, we enhance the protocol by employing more efficient cryptographic primitives. Putting these together, we introduce SmartZKCP, a novel fair data exchange protocol that is particularly suited for blockchain-based public data marketplaces. Our proposal extends the data exchange to accommodate multi-buyer scenarios and provides a comprehensive security analysis. We evaluate the performance of SmartZKCP in a "Pay to Sudoku solution" application, where it achieves an 11-65× improvement in seller efficiency compared to the vanilla ZKCP. The performance is also comparable to a seminal optimization of ZKCP [14]. Additionally, our proposal significantly reduces the verification overhead and communication costs relative to [14]. The protocol is efficient, practical, and can be implemented in any existing public blockchain.

We summarize our contributions as follows:

- We identify three potential attacks on the classical ZKCP protocol in a practical data marketplace, provide an analysis of these attacks, and address them with novel methods.
- Putting the methods together, we introduce SmartZKCP, a secure and efficient fair data exchange protocol that addresses the challenges identified. SmartZKCP is particularly well-suited for use in blockchain-based public data marketplaces.
- We implement and evaluate the performance of our proposal. All the implementations are publicly available. Our evaluation demonstrates that the protocol matches the efficiency of the state-of-the-art optimizations of ZKCP and is especially appropriate for data exchanges facilitated by blockchain and smart contracts.

The rest of the paper is organized as follows. Section 2 provides the necessary background and preliminaries. Section 3 delivers an overview of the general structure and our security model. Section 4 details the potential attacks on ZKCP and provides fixes for these flaws. Section 5 introduces SmartZKCP and discusses several optimizations. Section 6 describes the implementation of SmartZKCP and evaluates its performance. The final section concludes the paper.

1.1. Related Works

It is well recognized that without additional assumptions, such as a Trusted Third Party (TTP), fair exchange is unachievable [18]. However, the advent of blockchain technology and smart contracts has made fair exchange feasible. Blockchain-based fair exchanges can be classified into two types: optimistic solutions and zero-knowledge proof based solutions. Optimistic solutions, such as FairSwap [8, 1] and OptiSwap [9], assume that most participants are honest and misbehavior is infrequent. These protocols offer lightweight fair exchange mechanisms and allow participants to submit a “Proof of Misbehavior” to the arbiter in case of a dispute. However, these methods may lead to delays in resolving disputes if misbehavior increases, which can be unacceptable in fast-paced exchanges. Moreover, recording these proofs on the blockchain incurs significant storage and gas costs, presenting a substantial burden and expense. Clearly, these methods offer weaker fairness and higher risk compared to zero-knowledge proof based solutions.

Zero-knowledge proof based solutions provide a higher level of security and fairness. The pioneering solution, ZKCP, was proposed by Maxwell [15] and employs the Bitcoin script along with a Pinocchio [19] zero-knowledge proof system to enable fair exchanges. Campanelli et al. [4] revisited this proposal and identified some potential attacks on ZKCP. Further security analyses on ZKCP were conducted in [10, 17], primarily focusing on the zero-knowledge proof system it employs. The attacks identified in this work have not yet been studied. A recent optimization of ZKCP, known as ZKCPlus [14], enhances both the protocol and the underlying zero-knowledge proof system. However, the potential attacks identified in this work are also applicable to ZKCPlus. Other works, such as those extending the application of ZKCP [24], can be viewed as complementary to our proposal.

There are other proposals [23, 5] focusing on fair data exchange in blockchain-based marketplaces. However, these proposals either require stronger assumptions or depend on the support of trusted hardware, which is not necessary in this work.

2. Preliminaries

2.1. Notation

Let λ be the security parameter. Let $\mathsf{negl}(\cdot)$ denote a negligible function. Let $\mathbb{G}$ be a cyclic group of prime order, and a public generator of the group is defined as g. Let $\mathbb{F}$ be a finite field of prime order. "PPT" stands for probabilistic polynomial time.

2.2. Blockchain and Smart Contract

A blockchain [16] is a distributed protocol wherein a group of nodes collaboratively maintains a ledger, comprising an ordered sequence of *blocks*. Each user possesses an *address*, a public key and a private key. Users can initiate *transactions* by signing them with their private key. Transactions are grouped into blocks, which are then appended to the ledger.

Beyond achieving consensus, many blockchains such as [22] facilitate expressive, user-defined programs known as *smart contracts*. These are stateful programs whose state is preserved on the blockchain. In essence, the state of a smart contract can be conceptualized as a key-value store. Users initiate transactions to interact with a smart contract, thereby potentially altering its state. Platforms like [22] typically impose a fee (commonly referred to as gas) for each computation and storage step performed. The costs are notably high. for example, storing 1 KB of data on [22] incurs a cost of approximately 0.008 ETH, equivalent to around $25 at the time of this writing. Thus, minimizing on-chain computation and storage costs remains a paramount objective.

2.3. Succinct Non-interactive Arguments of Knowledge

In the original ZKCP protocol, the seller utilizes Zero-Knowledge Succinct Non-interactive Arguments of Knowledge (zk-SNARKs) [12] to assure the buyer that the good in question meets a specified property without disclosing any details about the item itself. We provide a formal definition of zk-SNARKs as follows:

Definition 1 (zk-SNARKs). Let $\mathcal{R}$ be an efficiently decidable binary relation, which defines the NP language $\mathcal{L} := \{x|\exists w : (x, w) \in \mathcal{R}\}$. Here x is called the (public) statement and w is called the (private) witness. A zk-SNARK consists of a triple of algorithms $\Pi_{\mathsf{ZK}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Veri})$, where:

- $\mathsf{Setup}(1^\lambda, \mathcal{R})$. It is the common reference string (CRS) generation algorithm that takes input as the security parameter λ and the relation $\mathcal{R}$. It outputs the CRS crs.
- $\mathsf{Prove}(\mathsf{crs}, x, w)$. It is the proof generation algorithm that takes input as crs, the public input x, and the witness w. It generates a proof π.
- $\mathsf{Veri}(\mathsf{crs}, x, \pi)$. It is the verification algorithm that takes input as crs, the public input x, and the proof π. It outputs a bit $b \in 0/1$ to indicate whether the proof is accepted or not.

We say the triple of algorithms $\Pi_{\mathsf{ZK}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Veri})$ is a zk-SNARK if it satisfies the following properties:

- **Perfect completeness**. For every crs output by Setup, a valid statement-witness pair $(x, w) \in \mathcal{R}$, the following relation holds:

$$\Pr\left[\begin{matrix}\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{R}), \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)\end{matrix} : \mathsf{Veri}(\mathsf{crs}, x, \pi) = 1\right] = 1$$

- **Computational knowledge soundness**. For any PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}$ such that the following probability is negligible:

$$\Pr\left[\begin{matrix}\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{R}), \\ ((x, \pi); w) \leftarrow \mathcal{E}^{\mathcal{A}}(\mathsf{crs}, \mathcal{R})\end{matrix} : \begin{matrix}(x, w) \notin \mathcal{R}, \\ \mathsf{Veri}(\mathsf{crs}, x, \pi) \quad = 1\end{matrix}\right]$$

- **Computational zero-knowledge**. For all PPT adversary $\mathcal{A}$, there exists a PPT simulator Sim such that:

$$\Pr\left[\begin{matrix}\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{R}), \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)\end{matrix} : \mathcal{A}(\mathsf{crs}, \mathcal{R}, x, \pi) = 1\right]$$
$$\approx \Pr\left[\begin{matrix}\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{R}), \\ \pi \leftarrow \mathsf{Sim}(\mathsf{crs}, x)\end{matrix} : \mathcal{A}(\mathsf{crs}, \mathcal{R}, x, \pi) = 1\right]$$

  Here $\approx$ is used to denote the two distributions are computationally indistinguishable.

- **Succinctness**. We say it is *succinct* if the proof size $|\pi|$ is of $\mathsf{poly}(\lambda, |x|, \log |w|)$.

In practice, the relation $\mathcal{R}$ is represented in a circuit C that takes as input x and w, and outputs 1 if and only if $(x, w) \in \mathcal{R}$.

2.4. Commitment Scheme

A commitment scheme enables a party to commit to a value and subsequently reveal it. This scheme must exhibit both binding and hiding properties. The binding property ensures that once a commitment is made, it cannot be altered to reveal different values. The hiding property guarantees that the commitment does not disclose any information about the committed value. In this work, we utilize two algorithms of a commitment scheme:

- $\mathsf{Setup}(1^\lambda)$. It is the public parameter generation algorithm that takes input as the security parameter λ, and outputs the public parameter pp.
- $\mathsf{Commit}(\mathsf{pp}, m, r)$. It is the commitment generation algorithm that takes input as the public parameter pp, the message m, and the random coin r. It generates a commitment com_m. When the randomness r is out of the scope, it may be omitted.

3. Protocol Overview

We target on designing SmartZKCP, a blockchain-based fair data exchange protocol, to facilitate a practical data marketplace. We capture three primary entities in the protocol.

- **Sellers $\mathcal{S}$**: A seller, as a data owner, wishes to sell data to other buyers via the marketplace. We assume that each seller possesses a digital good $s \in \{0,1\}^\lambda$ that satisfies a specified property $\phi(s) = 1$, where $\phi : \{0,1\}^\lambda \rightarrow \{0,1\}$ is a public predicate function used to evaluate this property. For instance, if s represents a Sudoku solution, then ϕ is a Sudoku checking algorithm and $\phi(s) = 1$ indicates that s is a valid solution to the designated Sudoku puzzle.
- **Buyers $\mathcal{B}$**: A buyer is keen on purchasing goods from sellers and is prepared to pay a specified amount v in exchange for a valid good. Furthermore, we assume that in a public marketplace, there may be multiple buyers interested in acquiring the same good.
- **The smart contract $\mathcal{J}$**: In the blockchain-based data marketplace, a smart contract is utilized to facilitate the exchange of goods between sellers and buyers. We assume that the smart contract is tamper-resistant and accessible to all marketplace participants.

Threat model. In this study, we address a threat model wherein a malicious adversary may attempt to disrupt the data exchange process. We first identify two primary adversaries in the data marketplace:

- **Seller.** A malicious seller aims to secure payment from the buyer without delivering the correct good.
- **Buyer.** A malicious buyer aims to extract information about the good from the seller without paying the agreed-upon amount.

Beyond these primary adversaries, we also consider additional potential active threats in a practical data marketplace:

- **Eavesdropper.** Operating within a public blockchain-based marketplace does not always ensure secure communication between the seller and the buyer. Consequently, an eavesdropper might intercept their communication, aiming to acquire information about the good without making a payment.
- **DoS attack.** A malicious third party could disrupt the data exchange process by launching a Denial of Service (DoS) attack. For example, the adversary might request goods from the seller and then cancel the transaction after the seller has provided much effort, thus depleting the seller's computational resources. Given that the ZKCP protocol requires the seller to expend considerable effort to compute a proof validating the good, this attack can be particularly damaging.
- **Reputation attack.** Data marketplaces operate as reputation systems, where buyers are more inclined to engage with sellers who have a good reputation. A malicious third party could damage an honest seller's reputation by falsely claiming that the seller provided incorrect goods or proofs. This type of attack poses a significant threat to the seller's interests.

Protocol goals. To counteract the aforementioned adversaries and establish a practical data marketplace, we aim to design SmartZKCP, a fair data exchange protocol that fulfills the following security properties:

- **Fairness.** The protocol guarantees a fair data exchange between the seller and the buyer. For buyer fairness, if the account balance of a malicious seller $\mathcal{S}^*$ increases, then the buyer acquires knowledge of the good s, such that $\phi(s) = 1$. Regarding seller fairness, if the balance of $\mathcal{S}$ does not increase, then any malicious buyer $\mathcal{B}^*$ learns nothing about s.
- **Practicality.** We define practicality as the protocol's capability to defend against the outlined adversaries in a real-world data marketplace: (i) A malicious eavesdropper cannot gain any knowledge about the good s from the communication between the seller and the buyer. (ii) The protocol resists DoS attacks, ensuring that the seller's computational resources are not depleted by onerous computation tasks. (iii) The protocol facilitates public verification of the exchange, thus safeguarding the seller's reputation from potential reputation attacks.
- **Efficiency.** The protocol's computational and communication costs should be optimally minimized and efficient. Both the seller and the buyer should be able to complete the exchange process within a reasonable time frame and at a reasonable onchain cost.

3.1. An Overview of Workings

In this section, we provide an overview of the fair exchange protocol and discuss the workings of SmartZKCP.

In a data exchange, the process begins with the seller $\mathcal{S}$ and the buyer $\mathcal{B}$ negotiating to agree on a predicate function ϕ, which the good s must satisfy. After reaching an agreement, $\mathcal{S}$ exchanges the good s with $\mathcal{B}$ in return for a payment v. ZKCP [3] offers a solution to the "TTP dilemma" [18] through the use of blockchain technology. Informally speaking, within the protocol, the seller $\mathcal{S}$ initially proves to the buyer $\mathcal{B}$ that a specific ciphertext z indeed represents the encryption of a good s that satisfies

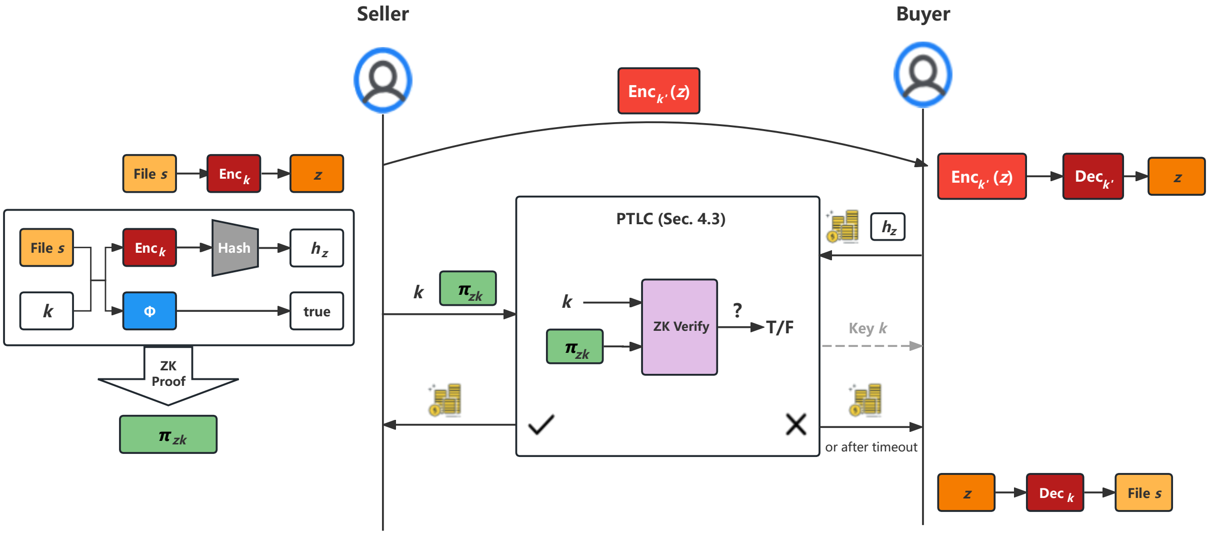


Figure 2: An overview of SmartZKCP.

the conditions specified by the predicate ϕ, through a zk-SNARK proof. Subsequently, the data exchange is reduced to an atomic swap involving the decryption key k and the payment v. This transaction can be efficiently managed using a Hash Time Locked Contract (HTLC) [2] (Fig. 3).

Although the protocol achieves the predefined fairness property (discussed in detail in Section 4.1), it possesses several limitations. In this work, we identify three primary challenges when applying ZKCP in a practical data exchange marketplace: (i) The protocol lacks privacy protection against eavesdroppers; (ii) It is susceptible to DoS attacks; and (iii) It does not facilitate public verification of the exchange, which is vital for reputation management in a public marketplace. Our contribution lies in analyzing these challenges and giving solutions to these challenges. Detailed discussions are provided in Section 4.

With the resolution of the aforementioned threats, we introduce SmartZKCP, a novel secure and efficient fair data exchange protocol. As illustrated in Figure 2, SmartZKCP adheres to the foundational principles of ZKCP, thereby preserving the essential fairness property. Crucially, SmartZKCP enhances protection against the practical threats identified earlier, making it well-suited for real-world data marketplaces. Additionally, it can be easily extended to a multi-buyer setting, enabling the seller to distribute the same good to multiple buyers with minimal additional burden. A detailed construction of SmartZKCP is provided in Section 5.

4. Practical Attacks on ZKCP

In this section, we provide an introduction to the original ZKCP protocol and identify potential vulnerabilities. We then discuss strategies to address these issues, which are further elaborated in the design of SmartZKCP.

4.1. Review ZKCP

We formalize the ZKCP protocol as following:

1. **Delivery.** The seller $\mathcal{S}$ encrypts the good s using a key k to generate a ciphertext $z \leftarrow \mathsf{Enc}_k(s)$, where Enc represents a symmetric encryption function. $\mathcal{S}$ also commits to the encryption key k by setting $h_k \leftarrow \mathsf{Com.Commit}(k)$. $\mathcal{S}$ sends the ciphertext z, the commitment h_k to the buyer $\mathcal{B}$.

2. **Validation.** The seller $\mathcal{S}$ uses a zk-SNARK to produce a proof π demonstrating the knowledge of s and k about the following relation:

$$\begin{aligned}\{(x, w) | x = (z, h_k), w = (s, k),& \quad (1)\\ \phi(s) = 1 \wedge z = \mathsf{Enc}_k(s) \wedge \mathsf{Com.Commit}(k) = h_k\}&\end{aligned}$$

   $\mathcal{S}$ also sends the proof π to $\mathcal{B}$.

3. **Lock.** Upon receipt from $\mathcal{S}$, $\mathcal{B}$ first verifies the proof π with (z, h_k). If the proof is valid, $\mathcal{B}$ sends (Lock, v, h_k) to the functionality $\mathcal{F}_{\mathsf{HTLC}}$, as outlined in Fig. 3, to locks the payment v.

4. **Reveal.** To receive the payment, $\mathcal{S}$ sends (Reveal, k) to $\mathcal{F}_{\mathsf{HTLC}}$ to disclose the encryption key k. If k matches the committed message, the payment v is released to $\mathcal{S}$. If $\mathcal{S}$ does not reveal k within the predefined time τ, $\mathcal{B}$ can reclaim the payment v.

The protocol was first implemented in Bitcoin [15], leveraging HTLC to implement the functionality $\mathcal{F}_{\mathsf{HTLC}}$ via a Bitcoin script. The commitment scheme Com is realized using the $\mathsf{SHA256}$ hash function. Importantly, this protocol ensures the aforementioned fairness: Intuitively, for the buyer fairness, after the seller reveals the encryption key k and obtains the payment, the buyer can decrypt the ciphertext z to retrieve the legitimate good s. For the seller fairness, the buyer cannot learn any information about the good s without first making the payment v. However, we will show that the protocol is vulnerable to several practical threats in a real-world data marketplace.

The functionality $\mathcal{F}_{\mathsf{HTLC}}$

The functionality is parameterized with a commitment scheme Com and a time duration τ.

- Initialize the buyer $\mathcal{B}$'s account balance $v_{\mathcal{B}}$ and the seller $\mathcal{S}$'s account balance $v_{\mathcal{S}}$.
- Upon receiving (Lock, v, h_k) from $\mathcal{B}$:
  - Assert $v_{\mathcal{B}} \geq v$;
  - Store h_k and the current timestamp t_0;
- Upon receiving (Reveal, k) from $\mathcal{S}$:
  - Get current timestamp t_1 and assert $t_1 \leq t_0 + \tau$;
  - Assert $\mathsf{Com.Commit}(k) = h_k$;
  - Set $v_{\mathcal{S}} := v_{\mathcal{S}} + v$;
  - Set $v_{\mathcal{B}} := v_{\mathcal{B}} - v$;
  - Send k to $\mathcal{B}$;

Figure 3: The functionality of HTLC $\mathcal{F}_{\mathsf{HTLC}}$.

4.2. Attack ZKCP

Recall our threat model where an eavesdropper can observe all messages communicated between the buyer and the seller, and a malicious third party may attempt to disrupt the data marketplace by launching DoS and reputation attacks for its own interests. In this section, we identify three attacks that are not only possible in a real-world data marketplace but also detrimental to the practicality of the data exchange protocol.

Reputation attack. In the protocol, the proof π is sent to the buyer $\mathcal{B}$ who verifies it independently. $\mathcal{B}$ can then decide whether to proceed with the exchange based on the validity of the proof. In a reputation-aware data marketplace, $\mathcal{B}$ also has the option to report a seller providing a false proof or an invalid good, which could negatively impact the seller's reputation. However, a potential issue arises if an adversarial buyer $\mathcal{B}'$ consistently claims that the provided proofs are invalid, even when they are correct, with the intention of damaging the seller's reputation. Since the proof is only verified by $\mathcal{B}'$, the seller $\mathcal{S}$ must undertake additional efforts to affirm the correctness of the proof to other potential buyers.

DoS attack. To sell a digital good, the seller $\mathcal{S}$ always needs to initially generate a zk-SNARK proof to demonstrate the validity of the good s and prove the knowledge of a unique key k, which is a computationally intensive task. In contrast, the buyer $\mathcal{B}$ does not expend much effort to verify the proof. This process is vulnerable to exploitation by an adversary: an attacker could pose as different buyers and repeatedly request proof of the good from the seller and cancel the transaction after receiving the proof, significantly draining the seller's computational resources. This type of attack, known as a DoS attack, is common in real-world scenarios and can be particularly effective in a data marketplace, especially when the seller is a resource-constrained device.

Eavesdropper's attack. In the ZKCP protocol, the encrypted good z is initially delivered to the buyer $\mathcal{B}$ through a private channel, and $\mathcal{B}$ can decrypt the ciphertext using the revealed key k after making the payment. However, since the key k is publicly revealed on a blockchain, any adversary can also obtain this key. Thus, if an eavesdropper controls the communication channel between $\mathcal{B}$ and $\mathcal{S}$, it can acquire the good s without making any payment through the following steps: The eavesdropper first silently intercepts the ciphertext z, waits for $\mathcal{S}$ to reveal the key k, and then uses it to decrypt the ciphertext z and immediately obtain the good. A similar attack is also noticed by [21].

The first and second attacks are particularly prevalent. Regarding the third attack, it is important to note that in real-world data marketplaces, peer-to-peer communication is not always secure. Consequently, these attacks are tangible and significantly impact the practicality of the data exchange protocol. It is noticed that these attacks are also applicable to the optimized version of ZKCP, such as [14].

4.3. Fix ZKCP

We now discuss how to address the aforementioned flaws. For clarity, we will outline strategies to counter each attack in this section and defer the detailed construction of SmartZKCP to the next section.

To mitigate the reputation attack, we note that verifying the proof π through a smart contract, rather than by the buyer $\mathcal{B}$ alone, can be an effective strategy. This method ensures public verification of the proof, preventing an adversary from falsely denying an accepted proof and thus protecting the seller's reputation. However, it introduces a new challenge: verifying the proof related to Relation 1 onchain requires the smart contract to read the entire ciphertext z. Given that read operations are costly in terms of gas fees, this approach could incur significant expenses and potentially render the protocol impractical due to the high gas costs.

To minimize the gas cost, we suggest that the seller $\mathcal{S}$ generate a proof for a new relation (instead of Relation 1):

$$\{(x, w) | x = (h_z, k), w = s, \\ \phi(s) = 1 \wedge \mathsf{Com.Commit}(\mathsf{Enc}_k(s)) = h_z\} \tag{2}$$

The distinction from Relation 1 lies in the fact that the seller commits to the ciphertext $z = \mathsf{Enc}_k(s)$ rather than the encryption key k. To verify a proof related to this relation, the only information to publish to the smart contract is the commitment h_z of the ciphertext z. This commitment can be instantiated as a hash, which is significantly smaller than the ciphertext itself, thereby making onchain verification feasible. Subsequently, the fair exchange is simplified to an atomic swap that involves exchanging the decryption key k, and the valid proof π for the payment v. To enable this atomic swap, we introduce a novel primitive termed Proof Time Lock Contract (PTLC) for the first time in the literature, which is adapted from HTLC. The functionality of the PTLC is illustrated in Fig. 4.

The functionality $\mathcal{F}_{\mathsf{PTLC}}$

The functionality is parameterized with a zk-SNARK scheme Π_{ZK} (along with the CRS crs) and a time duration τ.

- Initialize the buyer $\mathcal{B}$'s account balance $v_{\mathcal{B}}$ and the seller $\mathcal{S}$'s account balance $v_{\mathcal{S}}$.
- Upon receiving (Lock, v, h_z) from $\mathcal{B}$:
  - Assert $v_{\mathcal{B}} \geq v$;
  - Store h_z and the current timestamp t_0;
- Upon receiving $(\mathsf{Reveal}, k, \pi)$ from $\mathcal{S}$:
  - Get current timestamp t_1 and assert $t_1 \leq t_0 + \tau$;
  - Assert $\Pi_{\mathsf{ZK}}.\mathsf{Veri}(\mathsf{crs}, (h_z, k), \pi) = 1$;
  - Set $v_{\mathcal{S}} := v_{\mathcal{S}} + v$;
  - Set $v_{\mathcal{B}} := v_{\mathcal{B}} - v$;
  - Send k to $\mathcal{B}$;

Figure 4: The functionality of PTLC $\mathcal{F}_{\mathsf{PTLC}}$.

The following modifications are made to the original ZKCP protocol (Sec. 4.1): We swap the order of Step 2 and Step 3, such that the buyer locks the payment before the seller generates the proof. During the Reveal phase (Step 4), $\mathcal{S}$ sends $(\mathsf{Reveal}, k, \pi)$ to the PTLC, which verifies the proof π, instead of directly to $\mathcal{B}$, and then disburses the payment to $\mathcal{S}$.

We also notice that swapping the order of Step 2 and Step 3 can effectively mitigate the DoS attack. Recall that the DoS attack originates from the requirement for the seller $\mathcal{S}$ to generate a computationally intensive zk-SNARK proof at the beginning of each exchange, while the buyer $\mathcal{B}$ incurs minimal costs until the payment is locked. By reversing the order of Step 2 and Step 3, $\mathcal{B}$ must lock the payment before $\mathcal{S}$ generates the proof. This modification necessitates $\mathcal{B}$ to invest a significant amount of money to initiate a DoS attack, thereby deterring the adversary. Furthermore, in our modified protocol, once the $\mathcal{B}$ has locked the payment, an honest $\mathcal{S}$ can generate the proof to redeem the payment, thereby ensuring that the computational resources of the seller are not wasted.

Finally, to counteract the eavesdropper's attack, we propose to adopt a double encryption [13]. More specifically, in the Delivery step, the seller $\mathcal{S}$ computes and delivers the ciphertext $c \leftarrow \mathsf{Enc}_{k'}(\mathsf{Enc}_k(s))$, where the good s is double encrypted. In this scheme, k serves its original purpose, and k' is a new key, negotiated by $\mathcal{S}$ and $\mathcal{B}$ through a classical STS key agreement protocol [6]. Intuitively, the second layer of encryption ensures that the eavesdropper cannot decrypt the ciphertext without knowing k'. Given that the STS key agreement protocol is secure against eavesdroppers, this strategy effectively mitigates the eavesdropper's attack. In the following section, we will present a detailed construction of SmartZKCP, which includes a deferred description of the key agreement protocol as well. Jumping ahead, we find that this double encryption strategy does not significantly increase the complexity of the transaction or the computational overhead of the protocol. This is because: (i) the STS key agreement protocol is lightweight, and (ii) the relation that the ZK proof aims to prove remains unchanged, and the cost of proof generation stays the same as before.

5. Construction of SmartZKCP

We now summarize our corrections and give the construction of SmartZKCP, a novel blockchain-based fair data exchange protocol suitable for practical data marketplaces. In this protocol, the commitment scheme Com is implemented using a hashing function H, and the functionality $\mathcal{F}_{\mathsf{PTLC}}$ is realized through a smart contract $\mathcal{J}$. A detailed description of the protocol is provided in Figure 5. In the protocol, the parties first execute an STS key agreement protocol to generate a shared key k', which is used in the double encryption of the good s. It is important to note that in this phase, both the seller and buyer require a public key pair for digital signatures. Given that the protocol operates on a public blockchain, users can directly utilize the key pairs associated with their blockchain accounts. The core idea of the remainder of the protocol is to place the validation on the blockchain, where the proof is verified by the smart contract. The security analysis of the protocol is deferred to Sec. 5.3.

5.1. Instantiation and Optimization

In the actual implementation, we employ the $\mathsf{Groth16}$ zk-SNARK [12] to instantiate Π_{ZK}, owing to its concise proof size and efficient verification process. Additionally, the elliptic curve used in $\mathsf{Groth16}$ is natively supported by blockchains such as Ethereum [22]. However, we point out that Π_{ZK} can also be instantiated with other proof systems with succinct proof size and efficient verification algorithm.

Cryptographic primitives. The cryptographic primitives used in SmartZKCP, i.e., the encryption scheme Enc and hashing schemes H, are crucial for the efficiency of the protocol. This is because traditional cryptographic primitives may require extensive bitwise computation, making them unsuitable for circuit representation and computationally unfriendly for zk-SNARKs. For example, traditional hashing functions, such as $\mathsf{SHA256}$, are "unfriendly" for circuit representation and a $\mathsf{SHA256}$ hashing brings thousands of gates to prove, which makes the proof generation burdensome. This is exactly the problem resulting in long proof generation times in the vanilla ZKCP protocol [20].

To instantiate the encryption and hashing schemes, an important criterion is the efficiency of the cryptographic primitives in the context of circuit representation. In this

The Protocol of SmartZKCP

Let $\mathbb{F}$ be a finite field and $\mathbb{G}$ be a cyclic group with a public generator g. Let H be a hashing function, Enc be a symmetric encryption scheme, Π_{ZK} be a zk-SNARK and $\mathcal{J}$ be a smart contract that realizes $\mathcal{F}_{\mathsf{PTLC}}$. Before the protocol, the parities run Π_{ZK}.Setup to generate the CRS crs according to Relation 2. The fair data exchange between $\mathcal{S}$ and $\mathcal{B}$, consists of the following steps:

1. **Key Generation.** $\mathcal{B}$ and $\mathcal{S}$ run an STS key agreement protocol to generate a shared key k'.
   - $\mathcal{S}$ randomly samples $a \xleftarrow{\$} \mathbb{F}$ and computes g^a to send to $\mathcal{B}$.
   - $\mathcal{B}$ randomly samples $b \xleftarrow{\$} \mathbb{F}$ and computes $k' = g^{ab}$. $\mathcal{B}$ responses with g^b and a token $\mathsf{Enc}_{k'}(\mathsf{S}_{\mathcal{B}}(g_a, g_b))$ where $\mathsf{S}_{\mathcal{B}}(g_a, g_b)$ is the signature of g_a, g_b signed by $\mathcal{B}$'s private key.
   - $\mathcal{S}$ computes $k' = g^{ab}$, decrypts the token and verifies $\mathcal{B}$'s signature. $\mathcal{S}$ responses with a token $\mathsf{Enc}_{k'}(\mathsf{S}_{\mathcal{S}}(g_a, g_b))$ where $\mathsf{S}_{\mathcal{S}}(g_a, g_b)$ is the signature of g_a, g_b signed by $\mathcal{S}$'s private key.
   - $\mathcal{B}$ similarly verifies $\mathcal{S}$'s signature.
2. **Delivery.**
   - $\mathcal{S}$ randomly samples a key $k \xleftarrow{\$} \mathbb{F}$. $\mathcal{S}$ encrypts the good s with the key k and obtains the ciphertext $z \leftarrow \mathsf{Enc}_k(s)$. $\mathcal{S}$ commits to z as $h_z \leftarrow \mathsf{H}(z)$.
   - $\mathcal{S}$ double encrypts the ciphertext z with the key k' and obtains the ciphertext $c \leftarrow \mathsf{Enc}_{k'}(z)$.
   - $\mathcal{S}$ sends c, h_z to $\mathcal{B}$.
3. **Lock.**
   - $\mathcal{B}$ uses the key k' to decrypt the ciphertext c and obtains $z := \mathsf{Enc}_k(s)$. $\mathcal{B}$ computes $\mathsf{H}(z)$ and checks if it matches h_z.
   - If the check passes, $\mathcal{B}$ sends (Lock, v, h_z) to $\mathcal{J}$ to lock the payment v.
4. **Validation & Reveal.**
   - $\mathcal{S}$ checks the consistency between the hash published by $\mathcal{B}$ and h_z.
   - $\mathcal{S}$ runs $\Pi_{\mathsf{ZK}}.\mathsf{Prove}(\mathsf{crs}, (h_z, k), s)$ to generate a proof π.
   - $\mathcal{S}$ sends $(\mathsf{Reveal}, k, \pi)$ to $\mathcal{J}$. If the proof is valid, the payment v is released to $\mathcal{S}$. If $\mathcal{S}$ does not reveal k, π within the predefined time τ, $\mathcal{B}$ can reclaim the payment v.

Figure 5: The SmartZKCP protocol.

work, we instantiate the hashing function with POSEIDON [11], which is both efficient and optimized for zk-SNARKs. To maintain the hiding property of the commitment scheme, randomness is added to the input of the hashing function as well. For the symmetric encryption scheme, we use Ciminion [7], a lightweight symmetric encryption scheme that minimizes the number of multiplication gates required in a circuit. By combining these two schemes, we achieve an efficient instantiation of the encryption and hashing functions in our protocol, significantly improving overall prover efficiency.

5.2. Dealing with Multiple Buyers

In this section, we explore extending SmartZKCP to a "multiple buyer" scenario. In real-world data marketplaces, it is common for multiple buyers to be interested in the same item. A naive approach would involve the seller running the SmartZKCP protocol individually with each buyer. However, this method is inefficient and time-consuming, as it requires the seller to generate a separate proof for each buyer, despite the item being identical.

An important observation is that the Relation 2 to be proved consists of two parts: the predicate function $\phi(s) = 1$ for the good s, and the commitment for the ciphertext z. Typically, proving the predicate function $\phi(s) = 1$ consumes the most time. Thus, our proposal is to generate a single proof for the predicate function $\phi(s) = 1$ and use it for convincing all buyers. In each individual transaction with a buyer, the seller would only need to prove the correctness of the commitment and encryption, which involves the knowledge of different keys. These two phases are linked by the commitment h_s of the good s.

Specifically, in transactions involving multiple buyers, the seller $\mathcal{S}$ first generates a proof π_ϕ for the following relation:

$$\begin{aligned}\{(x, w) | x = h_s, w = s,& \\ \phi(s) = 1 \wedge \mathsf{H}(s) = h_s\}&\end{aligned}$$

where h_s represents the commitment (e.g., a hashing digest) of the good s. $\mathcal{S}$ can publish this proof to a specified smart contract for validation. Thereafter, in each individual transaction with a buyer $\mathcal{B}_i$, $\mathcal{S}$ generates a proof π_i for the following relation:

$$\{(x, w) | x = (h_{z_i}, h_s, k_i), w = s, \\ \mathsf{H}(\mathsf{Enc}_{k_i}(s)) = h_{z_i} \wedge \mathsf{H}(s) = h_s\}$$

where z_i, h_{z_i} represents the ciphertext (encrypted by a key k_i) and corresponding commitment for the buyer $\mathcal{B}_i$. With these two proofs linked by the commitment h_s, a buyer can be convinced of the good's validity. In this approach, the $\mathcal{S}$ only needs to expend effort to generate a proof for the predicate function once, which significantly reduces the computational overhead for the seller in a multi-buyer scenario.

5.3. Security Analysis

In this section, we give a security analysis of SmartZKCP against various adversaries in a practical data marketplace. We will consider potential threats defined in Section 3.

5.3.1. *Malicious Seller and Buyer*

We first formalize the definition of *fairness* through the following definition:

Definition 2 (Fairness). A data exchange protocol is fair if it satisfies:

- Buyer fairness. For any malicious seller $\mathcal{S}^*$, if its account balance increases with non-negligible possibility, then the honest buyer $\mathcal{B}$ learns s such that $\phi(s) = 1$ with possibility 1.
- Seller fairness. For any malicious buyer $\mathcal{B}^*$, if the account balance of the honest seller $\mathcal{S}$ does not increase, then $\mathcal{B}^*$ learns no information about s.

The above definition has captured potential attacks from malicious sellers and buyers. We now prove that the protocol depicted in Fig. 5 satisfies the fairness property.

Theorem 1. *If the underlying zk-SNARK Π_{ZK} satisfies perfect completeness, computational knowledge soundness and computational zero-knowledge, and the commitment scheme* Com *is hiding and biding, the SmartZKCP protocol satisfies the fairness property.*

For buyer fairness, if a malicious seller $\mathcal{S}^*$ increases its account balance, then it must be the case that in Step 4 the contract $\mathcal{J}$ receives (k, π) such that

$$\Pi_{\mathsf{ZK}}.\mathsf{Veri}(\mathsf{crs}, (h_z, k), \pi) = 1$$

Due to the computational knowledge soundness of Π_{ZK}, there exists an extractor $\mathcal{E}$ that can output s' such that

$$\phi(s') = 1 \wedge \mathsf{H}(\mathsf{Enc}_k(s')) = h_z$$

except for negligible probability. If the buyer $\mathcal{B}$ does not learn s', then it must be the case $\mathcal{S}^*$ finds some (s, k) such that s is different from s' but $\mathsf{H}(\mathsf{Enc}_k(s)) = h_z$. This breaks the binding property of H or the security of Enc.

For seller fairness, if an honest seller $\mathcal{S}$ does not increase its account balance, then it must be the case that a malicious buyer $\mathcal{B}^*$ aborts after the delivery in Step 2. Due to the zero-knowledge property of Π_{ZK}, $\mathcal{B}^*$ learns nothing about s. Moreover, $\mathcal{B}^*$ cannot learn anything from the ciphertext z due to the hiding property of H and the security of Enc.

5.3.2. *Practical Attacks*

We now give intuition for the security of our protocol against other practical attacks mentioned in Section 4.2.

Reputation attack. A reputation attack is no longer feasible. Since in SmartZKCP, the verification of ZK proofs is conducted inside the smart contract, the contract will automatically release the payment to the seller with probability 1 if the proof is valid. Therefore, a malicious buyer cannot tarnish the reputation of an honest seller anymore.

Eavesdropper. The eavesdropper attack is also mitigated in SmartZKCP. The STS key agreement protocol (Step 1 in Figure 5) ensures that the eavesdropper cannot learn the shared key k' between the seller and the buyer. In SmartZKCP, an eavesdropper can only observe the double encrypted ciphertext c, rather than z. Without k', the eavesdropper can decrypt c to learn s with negligible probability due to the semantic security of Enc.

DoS attack. Finally, the DoS attack is mitigated in our protocol. The adversary hopes to exhaust the seller's computational resources by requesting proofs and then aborting the process after the seller's effort. However, in the current protocol, the burdensome proof generation occurs in Step 4, after the buyer locks the payment. Consequently, the buyer must lock the payment as a prerequisite to requesting the proof. Moreover, once the payment is locked, the buyer cannot abort the process. An honest seller always provides a valid proof and claims the payment with probability 1, thereby compensating for the effort involved in generating the proof.

6. Evaluation

We implement our protocol and evaluate it in a specified data exchange scenario, and our codebase is publicly available[1]. The circuits are implemented using the Circom library[2], and a proof is generated by a Groth16 zk-SNARK. To instantiate the circuit, we employ Ciminion symmetric encryption and the POSEIDON hashing function, as detailed in Section 5.1. We implement and deploy the smart contract on a private Ethereum blockchain. All experiments are conducted on an Ubuntu 20.04 system equipped with a 2.50 GHz AMD EPYC 7K83 64-Core Processor and 16 GB of RAM.

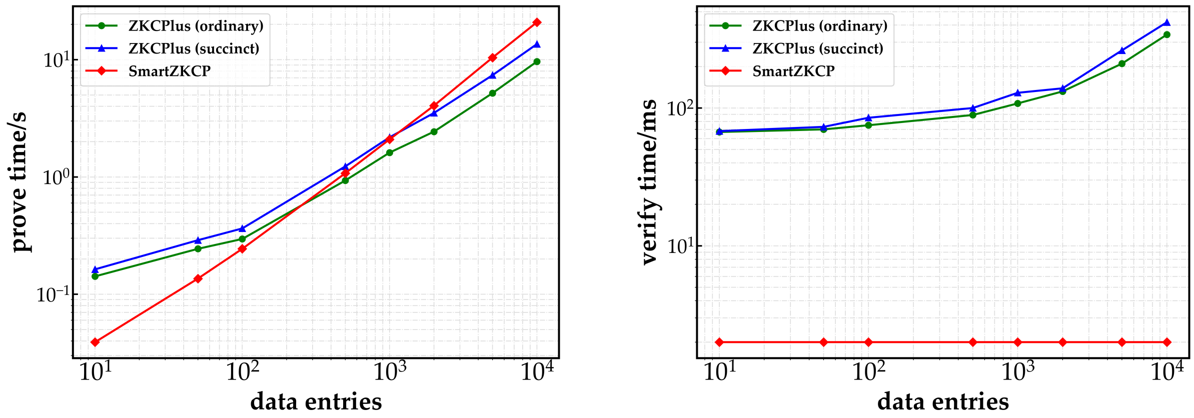
In the benchmark, we mainly focus on the efficiency of generating a zk-SNARK proof, as this is the most burdensome task in the protocol. The evaluations primarily focus on the following research questions: Is our protocol efficient for practical fair data exchange? To answer this question,

[1] https://github.com/Blockchain-Research-Center/SmartZKCP.git

[2] https://github.com/iden3/circom

Table 1
Comparison with ZKCP in a scenario "Pay to 16×16 Sudoku solution". The superscript s denotes a single-thread benchmark, and m denotes a multi-thread benchmark.

Protocol	Constraints	CRS size (MB)	Setup Time (s)	Proving Time(s)	Proof Size (KB)	Verify Time (ms)
ZKCP	236522	69.75	39.18	15	0.29	27
SmartZKCP^s	46112	22	2.87	1.37	0.28	2
SmartZKCP^m	46112	22	0.7	0.48	0.28	2



(a) The seller's proving cost.

(b) The verification cost.

Figure 6: Comparison between SmartZKCP and ZKCPlus, varying the scale of Sudoku puzzle.

we conduct two groups of experiments: (i) Benchmark the proof generation against the vanilla ZKCP protocol [20], and (ii) Benchmark the whole protocol against ZKCPlus [14], which represents a state-of-the-art optimization of ZKCP. We select a "Pay to Sudoku" scenario where the good s to be exchanged is "a solution to a Sudoku puzzle" and ϕ is the corresponding Sudoku verification algorithm.

6.1. Comparison with ZKCP

The initial comparison involves the proof generation between SmartZKCP and the vanilla ZKCP protocol in the context of "Pay to 16×16 Sudoku". We analyze the overhead, including the setup time, the proving time, the proof size, and the verify time. For SmartZKCP, we benchmark in both single-thread and multi-thread modes. The results are displayed in Table 1.

Compared to ZKCP, our proposal demonstrates significant improvements in proving time, proof size, and verification time. This is due to (i) SmartZKCP's use of the more efficient Groth16 zk-SNARK, compared to ZKCP's Pinocchio zk-SNARK [19], and (ii) the adoption of more efficient symmetric encryption and hashing functions in SmartZKCP. In single-thread mode, SmartZKCP is 11-65× more efficient than ZKCP during the proving phase, underscoring the efficiency and practicality of SmartZKCP.

6.2. Comparison with ZKCPlus

ZKCPlus [14] represents a state-of-the-art optimization of ZKCP, enhancing both the protocol's procedures and its underlying proof system. It offers two versions: ordinary and succinct. The ordinary version facilitates faster proof generation but results in larger proof sizes, while the succinct version yields smaller proofs albeit requiring more time for proof generation. In the second experiment, we compare the costs of the entire protocol from [14] and our proposal across various scales of Sudoku puzzles, defined as data entries ranging from 10 to 10,000. All the tests are conducted in a single-thread setting. Our primary focus is on the seller's proving overhead, the verification overhead (in [14], verification is conducted by the buyer, akin to the vanilla ZKCP), and the total communication cost.

Computation overhead. Figure 6 provides a comparison between ZKCPlus and our proposal. The seller's proving cost of our proposal is comparable to that of ZKCPlus. Regarding verification costs, the figure shows our protocol maintains a stable rate at approximately 2ms, whereas ZKCPlus experiences a rapid increase as the data volume grows. This disparity originates from the different zk-SNARKs utilized by the two protocols, making the verification in SmartZKCP feasible in a smart contract.

Communication overhead. We also compare the total communication cost of the two protocols, primarily consisting of the proof π and the ciphertext. We present the comparison between the two protocols involving a 36×36 Sudoku puzzle in Table 2 and analyze the percentage breakdown in Figure 7. The total communication cost for our proposal is only 32.82 KB, with the proof size accounting for only 0.8%,

Table 2
Comparison with ZKCPlus in a scenario "Pay to 36 × 36 Sudoku solution". Here the **size** column represents the size of each item, and the **takes** column indicates the percentage that each item contributes.

	ZKCPlus(ordinary)		ZKCPlus(succinct)		SmartZKCP	
	size (KB)	**takes**	**size (KB)**	**takes**	**size (KB)**	**takes**
Proof	170.66	82%	31.9	46%	0.28	0.8%
Ciphertext of data	33.34	16%	33.34	48%	32	97.5%
Others	4.12	4%	4.12	6%	0.54	1.7%
Total	208.12		69.36		32.82	

while ZKCPlus (Succinct) incurs a cost of 69 KB, with the proof size constituting 46%. Notably, the proof size for ZKCPlus increases dramatically, even surpassing the size of the ciphertext, rendering it impractical for large-scale data exchanges. The small communication cost of our proposal makes it more suitable for deployment in a blockchain-based data marketplace.

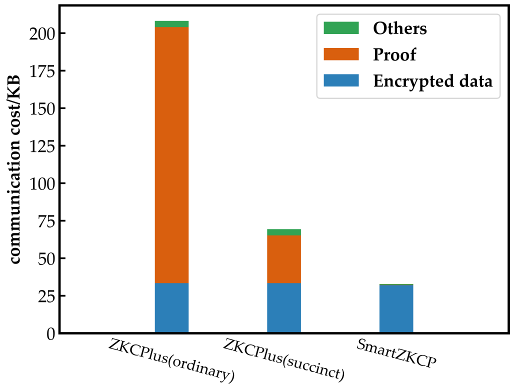


Figure 7: The percentage breakdown of the communication cost in a scenario "Pay to 36 × 36 Sudoku solution".

7. Conclusion

In this work, we identify three practical attacks on the classic ZKCP protocol and propose SmartZKCP, a secure and efficient fair data exchange protocol that addresses these challenges. Our proposal is particularly well-suited for use in a blockchain-based public data marketplace. Additionally, we extend the data exchange to accommodate a multiple-buyer setting and provide a security analysis of our proposal. Our evaluations demonstrate that SmartZKCP is both efficient and practical, making it suitable for implementation in any existing public blockchain.

Acknowledgement

This work was supported by the "Pioneer" and "Leading Goose" R&D Program of Zhejiang (No. 2022C01126), and the National Key Research and Development Program of China (No. 2023YFB2704000).

References

[1] Avizheh, S., Haffey, P., Safavi-Naini, R., 2022. Privacy-preserving FairSwap: Fairness and privacy interplay. Proceedings on Privacy Enhancing Technologies 2022, 417–439. doi:10.2478/popets-2022-0021.

[2] Bitcoin Wiki, 2024a. Hash time locked contracts. URL: https://en.bitcoin.it/wiki/Hash_Time_Locked_Contracts.

[3] Bitcoin Wiki, 2024b. Zero knowledge contingent payment. URL: https://en.bitcoin.it/wiki/Zero_Knowledge_Contingent_Payment.

[4] Campanelli, M., Gennaro, R., Goldfeder, S., Nizzardo, L., 2017. Zero-knowledge contingent payments revisited: Attacks and payments for services, in: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (Eds.), ACM CCS 2017: 24th Conference on Computer and Communications Security, ACM Press. pp. 229–243. doi:10.1145/3133956.3134060.

[5] Dai, W., Dai, C., Choo, K.K.R., Cui, C., Zou, D., Jin, H., 2019. Sdte: A secure blockchain-based data trading ecosystem. IEEE Transactions on Information Forensics and Security 15, 725–737.

[6] Diffie, W., 1988. The first ten years of public-key cryptography. Proceedings of the IEEE 76, 560–577. doi:10.1109/5.4442.

[7] Dobraunig, C., Grassi, L., Guinet, A., Kuijsters, D., 2021. Ciminion: Symmetric encryption based on Toffoli-gates over large finite fields, in: Canteaut, A., Standaert, F.X. (Eds.), Advances in Cryptology – EUROCRYPT 2021, Part II, Springer, Heidelberg. pp. 3–34. doi:10.1007/978-3-030-77886-6_1.

[8] Dziembowski, S., Eckey, L., Faust, S., 2018. FairSwap: How to fairly exchange digital goods, in: Lie, D., Mannan, M., Backes, M., Wang, X. (Eds.), ACM CCS 2018: 25th Conference on Computer and Communications Security, ACM Press. pp. 967–984. doi:10.1145/3243734.3243857.

[9] Eckey, L., Faust, S., Schlosser, B., 2020. OptiSwap: Fast optimistic fair exchange, in: Sun, H.M., Shieh, S.P., Gu, G., Ateniese, G. (Eds.), ASIACCS 20: 15th ACM Symposium on Information, Computer and Communications Security, ACM Press. pp. 543–557. doi:10.1145/3320269.3384749.

[10] Fuchsbauer, G., 2019. WI is not enough: Zero-knowledge contingent (service) payments revisited, in: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (Eds.), ACM CCS 2019: 26th Conference on Computer and Communications Security, ACM Press. pp. 49–62. doi:10.1145/3319535.3354234.

[11] Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., Schofnegger, M., 2021. Poseidon: A new hash function for zero-knowledge proof systems, in: Bailey, M., Greenstadt, R. (Eds.), USENIX Security 2021: 30th USENIX Security Symposium, USENIX Association. pp. 519–535.

[12] Groth, J., 2016. On the size of pairing-based non-interactive arguments, in: Fischlin, M., Coron, J.S. (Eds.), Advances in Cryptology – EUROCRYPT 2016, Part II, Springer, Heidelberg. pp. 305–326. doi:10.1007/978-3-662-49896-5_11.

[13] Hoang, V.T., Tessaro, S., 2017. The multi-user security of double encryption, in: Coron, J.S., Nielsen, J.B. (Eds.), Advances in Cryptology – EUROCRYPT 2017, Part II, Springer, Heidelberg. pp. 381–411. doi:10.1007/978-3-319-56614-6_13.

[14] Li, Y., Ye, C., Hu, Y., Morpheus, I., Guo, Y., Zhang, C., Zhang, Y., Sun, Z., Lu, Y., Wang, H., 2021. ZKCPlus: Optimized fair-exchange protocol supporting practical and flexible data exchange, in: Vigna, G., Shi, E. (Eds.), ACM CCS 2021: 28th Conference on Computer and Communications Security, ACM Press. pp. 3002–3021. doi:10.1145/3460120.3484558.

[15] Maxwell, G., 2016. The first successful zero-knowledge contingent payment. URL: https://bitcoincore.org/en/2016/02/26/zero-knowledge-contingent-payments-announcement/.

[16] Nakamoto, S., 2008. Bitcoin: A peer-to-peer electronic cash system. URL: https://bitcoin.org/bitcoin.pdf. accessed: 2015-07-01.

[17] Nguyen, K., Ambrona, M., Abe, M., 2020. WI is almost enough: Contingent payment all over again, in: Ligatti, J., Ou, X., Katz, J., Vigna, G. (Eds.), ACM CCS 2020: 27th Conference on Computer and Communications Security, ACM Press. pp. 641–656. doi:10.1145/3372297.3417888.

[18] Pagnia, H., Gärtner, F.C., et al., 1999. On the impossibility of fair exchange without a trusted third party. Technical Report. Citeseer.

[19] Parno, B., Howell, J., Gentry, C., Raykova, M., 2013. Pinocchio: Nearly practical verifiable computation, in: 2013 IEEE Symposium on Security and Privacy, IEEE Computer Society Press. pp. 238–252. doi:10.1109/SP.2013.47.

[20] Sean Bowe, 2016. Implementation of pay-to-sudoku. URL: https://github.com/zcash-hackworks/pay-to-sudoku.

[21] Song, R., Gao, S., Song, Y., Xiao, B., 2022. Zkdet: A traceable and privacy-preserving data exchange scheme based on non-fungible token and zero-knowledge, in: Proceedings - 2022 IEEE 42nd International Conference on Distributed Computing Systems, ICDCS 2022, Institute of Electrical and Electronics Engineers Inc.. pp. 224–234. doi:10.1109/ICDCS54860.2022.00030. funding Information: ACKNOWLEDGEMENT This work was partially supported by the HK RGC GRF PolyU No. 15216220 and 15217321. Publisher Copyright: © 2022 IEEE.; 42nd IEEE International Conference on Distributed Computing Systems, ICDCS 2022 ; Conference date: 10-07-2022 Through 13-07-2022.

[22] Wood, G., 2017. Ethereum: A secure decentralised generalised transaction ledger eip-150 revision (759dccd - 2017-08-07). URL: https://ethereum.github.io/yellowpaper/paper.pdf. accessed: 2018-01-03.

[23] Zheng, X., 2020. Data trading with differential privacy in data market, in: Proceedings of 2020 6th International Conference on Computing and Data Engineering, pp. 112–115.

[24] Zhou, Z., Cao, X., Liu, J., Zhang, B., Ren, K., 2021. Zero knowledge contingent payments for trained neural networks, in: Bertino, E., Shulman, H., Waidner, M. (Eds.), ESORICS 2021: 26th European Symposium on Research in Computer Security, Part II, Springer, Heidelberg. pp. 628–648. doi:10.1007/978-3-030-88428-4_31.