The Data Exchange Protocol over Multi-chain Blockchain Using Zero-Knowledge Proof

AoXuan Li[1(✉)], Gabriele D'Angelo[2], and Su-Kit Tang[1]

[1] Faculty of Applied Sciences, Macao Polytechnic University, Macao SAR, China
{aoxuan.li,sktang}@mpu.edu.mo

[2] Department of Computer Science and Engineering, University of Bologna, Bologna, Italy
g.dangelo@unibo.it

Abstract. The implementation of blockchain technology is becoming popular among cyber-physical systems. However, the current solutions suffer from scalability and privacy issues. In this position paper, we leverage zero-knowledge proof and multichain technology to propose an efficient system for data transferring across different components. Each component may maintain a private chain storing its data, and the system acts as a relayer between different chains, in which multiple private chains are efficient for appending new data. Only encrypted data is transferred from a source chain to a destination chain. The relayer handles data transferring in two phases: send and receive, and the relayer keeps a Merkle tree of all sent data. In fact, it only transfers the data if the receiver can submit a valid zero-knowledge proof that proves the ownership of the data. The zero-knowledge proof discloses nothing but the statement is true; therefore it protects anonymity for the data owners. This system is secure and satisfies relevant properties such as ledger indistinguishability, transaction non-malleability, and matchability.

Keywords: Zero-knowledge proof · Multichain · Blockchain · Cyber-physical system

1 Introduction

A cyber-physical system combines physical and computational components, and each part is tightly connected [12]. Blockchain is a distributed ledger that may hold any data, which is decentralized and transparent. All records on the blockchain are immutable, and no single party may manipulate the data. Blockchain is widely employed in cyber-physical systems [25], e.g., healthcare

This work is supported in part by the research grant (No.: RP/ESCA-04/2020) offered by Macao Polytechnic University.

Z. Xiao et al. (Eds.): ICECI 2022, LNICST 478, pp. 18–29, 2023.
https://doi.org/10.1007/978-3-031-28990-3_2

[8,10,17], IoT [14,16,22], and smart grid [23]. However, those solutions may suffer from scalability issues on the large scale of data, or they cannot guarantee privacy and anonymity for sensitive data.

Popular blockchain projects, e.g., Bitcoin [20] and Ethereum [29], are limited in throughput due to the computational cost for generating and appending new blocks to the blockchain, in fact, the nodes have to verify all previous blocks are valid. As the blockchain size grows, the cost for adding a new block becomes too expensive. Also, all data on the blockchain is transparent, and all parties may access any data on-chain. Moreover, for sensitive data, e.g., health data and financial data, the blockchain is unable to provide the needed privacy and anonymity.

A multichain approach allows each party to keep a private blockchain, and each blockchain may exchange data via a cross-chain transactions. In a cross-chain or single-chain transaction, the source blockchain network will transfer data to the destination blockchain network. Since each private blockchain is small compared with a single chain containing all data, the multichain approach could accelerate block generations. However, in such a protocol, the sender and receiver addresses are openly available on-chain, and one may track the transaction graph. Many research works show that this setting cannot provide privacy [7,24]. To address this issue, we employ a similar solution as in Zcash [2,15], that is a fork of Bitcoin [20]: the transaction is encrypted with the public key of the receiver, and then this receiver needs to prove the knowledge of the private key. If a sender *Alice* on blockchain A transfers some data to a receiver *Bob* on blockchain B, we want both *Alice* and *Bob* to be anonymous and no other party may know any information about the data.

For example, suppose blockchain A is a private chain of a hospital, and blockchain B belongs to a research institute. *Alice* is a patient in the hospital, and *Bob* is a researcher at the institute. *Bob* wants to retrieve someone's medical records for academic purposes, and *Alice* wants to contribute her data. After the transfer of *Alice*'s data from blockchain A to blockchain B, no one can figure out what the data contains and who the original owner of the data is. To address the first requirement, *Alice* may encrypt her data with *Bob*'s public key. The second requirement commonly refers to transaction unlinkability [4]. When *Alice* sends the data to *Bob*, *Bob* is the new owner of the data. To prove his ownership, *Bob* has to prove that he can correctly decrypt the message. A simple solution is to disclose *Bob*'s private key, and then one may verify the private key by decrypting data sent by *Alice*; however, this solution will also disclose *Alice*'s identity and the data content. To address the issue, we leveraged zero-knowledge proof and Merkle tree to verify the ownership of data. We explained zero-knowledge proof and the Merkle tree in detail in Sects. 2.1 and 3.5.

All data sent from chain A to chain B groups a Merkle tree, and the data sent from *Alice* to *Bob* is one of the leaves. To prove *Bob*'s ownership, he proves the following statements using zero-knowledge proof:

1. he knows a path from the data to the Merkle tree root, and
2. he knows the correct private key.

A zero-knowledge proof reveals nothing but the statements are correct. Since records on a blockchain are public to all nodes, if *Bob* writes the received data to chain B without modification, a malicious user who has access to both chain A and chain B may discover *Alice*'s identity by matching data content. For example, the malicious user may lookup chain A for a sending request where the data content is the same as a record on chain B. Therefore, *Bob* encrypts the data content with a different public key and writes the new ciphertext down on B. The encryption algorithm can guarantee that a malicious user cannot figure out two ciphertexts containing the same plain text if it has been encrypted with different public keys.

We can generalize the two-chain solution to the multichain system. As we mentioned before, a cyber-physical system may combine many components, and each component manages a private chain. To reduce communication complexity across the system, we introduced a middleware of multiple chains, called a relayer, to handle data exchange. Our proposed protocol is a general-purpose protocol so that it can fit in any multichain system supporting smart contracts. For example, financial institutes may employ our protocol to exchange data without a trusted third party. In this case, the institutes increase the data liquidity while protecting user privacy. The protocol also allows different sensors in an IoT system to transfer data efficiently and securely.

Our Contribution. We proposed a data exchange protocol over multi-chain blockchain using zero-knowledge proof. In the protocol, each component maintains a private blockchain, and different components can exchange data through cross-chain transactions. We also designed a relayer to carry data from a source chain to a destination chain, which keeps a Merkle tree of all sent data. This relayer is also responsible for verifying zero-knowledge proofs. This system provides both scalability and privacy for data transfer between different components in a cyber-physical system. It is noteworthy that the sender and the receiver may be on the same chain, i.e., the user sends a single-chain transaction, and there is a single chain that is both the origin and the destination of the transaction. Alternatively, the user can send a cross-chain transaction, and the source chain and the destination chain are on different chains. We illustrate the process in Fig. 1.

2 Background and Related Work

In this section, we describe the background technologies and methodologies that are necessary for understanding this paper.

2.1 Merkle Tree

Merkle tree [19], also called hash tree, is a tree in which every non-leaf node is labeled as the hash value of its child nodes' labels. The top of the tree is the root hash, and each leaf node has an authentication path to the root. It allows efficient verification of membership in a large data set. Figure 2 is an example of a Merkle tree, which contains a root hash rt and labels l_1, l_2, l_3, l_4. Verifying the membership of label l_1 only needs H_2 and H_{34}.

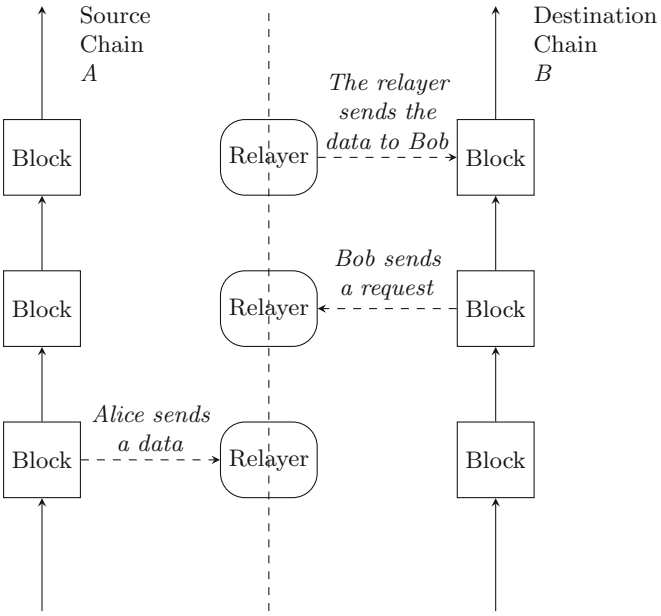

Fig. 1. Protocol Overview.

2.2 Blockchain

A blockchain is a distributed ledger among many network nodes, and records encode as ordered transactions. The transactions compose blocks, and each block links to the previous block with a hash value. The links of hash values make records on-chain immutable. If a malicious user wants to manipulate an intermediate block, the user must modify all the following blocks. Since the blockchain architecture is distributed and decentralized, usually modification is impossible. Bitcoin [20] is the first implementation of a blockchain, which supports asset transactions. Ethereum [29] introduced *smart contracts* into the blockchain, which enabled decentralized applications.

All network nodes have to store the same copy of a blockchain. *Consensus algorithms* allow nodes to agree on the status of the ledger. When a new block joins the chain, the majority of parties need to agree on the validity of the block and the containing transactions. There are different consensus algorithms, and the most popular algorithms are proof-based algorithms and vote-based algorithms [21]. In a proof-based algorithm, the first party who solves a hard puzzle has the right to append a block. In a vote-based algorithm, a block joins the chain if enough parties append the same block. More advanced structures are under investigation now [18,26].

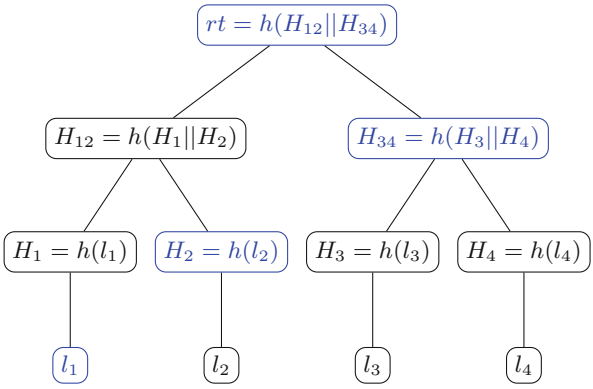

Fig. 2. Merkle Tree.

2.3 Zero-Knowledge Proof

Zero-knowledge proof was introduced by Goldwasser, Micali, and Rackoff [9], which allows a prover to convince the verifier that a statement is true without revealing anything but the truth of the statement. Blum, Feldman and Micali [5] extended the protocol to a non-interactive protocol. Zero-knowledge proof protocols are theoretical advances until recent implementation. The zk-SNARK algorithm [3,13] is the most popular instance of the protocol.

Zk-SNARK stands for Zero-Knowledge Succinct Non-Interactive Argument of Knowledge, in which the proof size is succinct and independent from the complexity of the statement. Zk-SNARK generates proofs in three phases:

1. The setup phase outputs public parameters and the SNARK for a language L in NP.
2. Prover generates a proof π with the instance x and the witness w.
3. Verifier can check the validity of π and x.

2.4 Related Works

Zero-knowledge proof (ZKP) is widely adopted among privacy-preserving blockchains. Zerocash [2] is one of the first projects using zero-knowledge proof on private transactions. However, it only supports intra-chain transactions, and the transaction type is limited to direct payment. Zexe [6] extends the functionality of zerocash and adds supports for conditional smart contracts. It requires users to compute the smart contract on their plain inputs off-line, and then the user generates a ZKP proving the correctness of the off-line computation. Zkay [28] further integrates ZKP with the blockchain by proposing a language for writing private smart contacts, and ZeeStar [27] is a succeeding work of Zkay which introduced a language based on homomorphic encryption. However, those works are very computationally intensive [1], and they are not optimized for data exchange protocols.

3 Preliminaries

In this section, we describe how to construct the protocol with zk-SNARK and other cryptography building blocks.

3.1 Cryptographic Building Blocks

We introduce the formal notation of the cryptography building blocks we use. λ denotes the security parameter. This part is similar to [2] section 4.1.

Collision-Resistant Hashing. We use a collision-resistant hash function $CRH : \{0,1\}^* \rightarrow \{0,1\}^{O(\lambda)}$.

Pseudorandom Functions. We use a pseudorandom function family $\mathbb{PRF} = \{PRF_x : \{0,1\}^* \leftarrow \{0,1\}^{O(\lambda)}\}_x$. We then instance three pseudorandom random functions from the same $PRF_x s \xleftarrow{\$} \mathbb{PRF}$ and add different prefix to the input. Namely,

- $PRF_x^{addr}(z) := PRF_x(00||z)$,
- $PRF_x^{sn}(z :) = PRF_x(01||z)$, and
- $PRF_x^{pk}(z) := PRF_x(10||z)$.

Moreover, we require PRF^{sn} to be collision resistant, i.e. one cannot find $(x, z) \neq (x', z')$ s.t. $PRF_x^{sn}(z) = PRF_{x'}^{sn}(z')$.

Statistically-Hiding Commitments. We use a computationally binding and statistically hiding commitment scheme $COMM$. Namely, $\{COMM_x : \{0,1\}^* \rightarrow \{0,1\}^{O(\lambda)}\}_x$ where x denotes the trapdoor parameter.

One-Time Strongly-Unforgeable Digital Signatures. We use a digital signature scheme $Sig = (G_{sig}, K_{sig}, S_{sig}, V_{sig})$.

- $G_{sig}(1^\lambda) \rightarrow pp_{sig}$. Given a security parameters λ, G_{sig} samples public parameters pp_{sig} for the signature scheme.
- $K_{sig}(pp_{sig}) \rightarrow (pk_{sig}, sk_{sig})$. Given public parameters pp_{sig}, K_{sig} samples a public key and a secret key for a single user.
- $S_{sig}(sk_{sig}, m) \rightarrow \sigma$. Given a secret key sk_{sig} and a message m, S_{sig} signs m to obtain a signature σ.
- $V_{sig}(pk_{sig}, m, \sigma) \rightarrow b$. Given a public key pk_{sig}, message m, and the signature σ, V_{sig} outputs $b = 1$ if validated or otherwise $b = 0$.

We require Sig to be one-time strong unforgeable against chosen-message attacks (SUF-1CMA security).

Key-Private Public-Key Encryption. We use a public-key encryption scheme $PKC = (G_{pkc}, K_{pkc}, PKC.Enc, PKC.Dec)$.

- $G_{pkc}(1^\lambda) \rightarrow pp_{pkc}$. Given a security parameter λ, G_{pkc} samples public parameters pp_{pkc} for the encryption scheme.

- $K_{pkc}(pp_{pkc}) \rightarrow (pk_{pkc}, sk_{pkc})$. Given public parameters pp_{pkc}, K_{pkc} samples a public key and a secret key for a single user.
- $PKC.Enc_{pk_{pkc}}(m) \rightarrow Ct$. Given a public key pk_{pkc} and a message m, $PKC.Enc$ encrypts m to obtain a cipher text Ct.
- $PKC.Dec_{sk_{pkc}}(Ct) \rightarrow m$. Given a secret key sk_{pkc} and a cipher text Ct, $PKC.Dec$ decrypts Ct to obtain the plain message m (or $\perp$ if decryption fails).

The encryption scheme PKC is secure against chosen-ciphertext attack and provides both ciphertext indistinguishability IND-CCA and key indistinguishability IK-CCA.

3.2 Concrete Design

In this section, we describe how we instantiate each building block in Table 1.

Table 1. Concrete Design.

CRH PRF $COMM$	Poseidon [11]
Sig	Elliptic Curve Digital Signature Algorithm
PKC	Elliptic-Curve Integrated Encryption Scheme

3.3 Transactions

We introduce two types of transactions.

- Send transactions. A send transaction tx_{send} is a tuple $(cm, con, *)$, where cm is the data commitment, con is the data's encrypted content, and $*$ are other information, e.g., randomness. The transaction tx_{send} records that a user transfer data with commitment cm and encrypted content con.
- Receive transactions. A receive transaction $tx_{receive}$ is a tuple $(rt, sn, cm^{new}, con^{new}, \pi_{\mathsf{RECEIVE}}, *)$, where rt, sn is the Merkle root and the serial number for the data on the source chain, cm^{new} is commitment of data on the destination chain, con^{new} is new encrypted content, and $*$ denotes other information. The transaction $tx_{receive}$ records that a user receives some data and writes it down on the destination chain.

3.4 Data

A data is an object d, which contains commitment, encrypted content, serial number, and address.

- commitment, denoted as cm: a string that appears on the ledger once d is sent.
- encrypted content, denoted as con: the encrypted content of d.
- serial number, denoted as sn: a unique string associated with d.
- address, denoted as $addr_{pk}$: an address public key, representing who owns d.

3.5 zk-SNARKs for Receiving Data

We use zk-SNARK to prove a NP statement $RECEIVE$. For the definition of zk-SNARK, we refer to [3] for a detailed explanation. We first provide an informal definition of zk-SNARKs. Given a field $\mathbb{F}$, a **zk-SNARK** for $\mathbb{F}-$arithmetic circuit satisfiability is a triple of polynomial-time algorithm $(\mathsf{KeyGen}, \mathsf{Prove}, \mathsf{Verify})$:

- $\mathsf{KeyGen}(1^\lambda, C) \rightarrow (pk, vk)$. On input: a security parameter λ and an $\mathbb{F}-$arithmetic circuit C, the *key generator* KeyGen probabilistically samples a proving key pk and a verification key vk.
- $\mathsf{Prove}(pk, x, a) \rightarrow \pi$. On input a proving key pk and any $(x, a) \in R_C$, the *prover* **Prove** outputs a non-interactive proof π for the statement $x \in L_C$.
- $\mathsf{Verify}(vk, x, \pi) \rightarrow b$. On input: a verification key vk, an input x, and a proof π, the *verifier* **Verify** outputs $b = 1$ if he/she is convinced that $x \in L_C$.

We recall the corresponding receive transaction $tx_{receive} = (rt,\ sn,\ cm^{new},\ con^{new},\ \pi_{\mathsf{RECEIVE}},\ *)$. To receive a data d, a user u should show that

1. u owns d,
2. commitment of d appears on the ledger,
3. sn is the calculated correctly as the serial number of d,
4. content are matched,

which is formalized as a statement $RECEIVE$ and proved with zk-SNARK. We then define the statement as follows.

- Instances is $x := (rt, sn, h, cm^{new}, h_{sig})$, which specifies a set rt, sn, h for old data, where rt is the root for a CRH-based Merkle tree, sn is the serial number, and h is the signature. It also specifies the public value v^{pub}, commitment of new data cm^{new}, and fields h_{sig} used for non-malleability.
- Witnesses are of the form $a := (path, c, addr_{sk}, d^{new})$ where

$$\begin{aligned} d &= (addr_{pk}, con, \rho, r, s, cm) \\ addr_{pk} &= (a_{pk}, pk_{pkc}) \\ d^{new} &= (addr_{pk}^{new}, v^{new}, \rho^{new}, r^{new}, s^{new}, cm^{new}) \\ addr_{pk}^{new} &= (a_{pk}^{new}, pk_{pkc}^{new}) \end{aligned}$$

  Thus, the witness a specifies an authenticated path from root rt to the data's commitment, the entirety information of the data d, the address secret key.

Given a $RECEIVE$ instance x, a witness a is valid for x if:

1. The data's commitment cm appears on the ledger, i.e., $path$ is a valid authentication path for leaf cm in a CRH-based Merkle tree with root rt.
2. The address secret key a_{sk} matches the address public key, i.e., $a_{pk} = PRF_{a_{sk}}^{addr}(0)$.
3. The nullifier key nk matches the address secret key, i.e., $nk = PRF_{a_{sk}}^{addr}(1)$.

4. The serial number sn is computed correctly, i.e., $sn = PRF^{sn}_{nk}(\rho)$.
5. The data d is well formatted, i.e., $cm = COMM_s(COMM_r(a_{pk}||\rho)||con)$.
6. The address secret key a_{sk} ties to h_{sig} to h, i.e., $h = PRF^{pk}_{a_{sk}}(h_{sig})$.
7. New data d^{new} are well formatted, i.e.,
   $cm = COMM_{s^{new}}(COMM_{r^{new}}(a^{new}_{pk}||\rho^{new})||con^{new})$.
8. Content are matched, i.e.,
   $con^{new} = PKC.Enc_{pk^{new}_{pkc}}(PKC.Dec_{sk_{pkc}}(con))$.

3.6 Security

Security of the protocol is characterized by three properties, which we call ledger *indistinguishability*, *transaction non-malleability*, and *matchability*.

Definition 1. *A protocol is secure if it satisfies ledger indistinguishability, transaction non-malleability, and balance.*

We provide the informal definition below.

Ledger Indistinguishability. This property captures the requirement that the ledger reveals no new information to the adversary beyond the publicly-revealed information (e.g. plain text address, coin's public value).

Transaction Non-malleability. This property means no bounded adversary may modify the data stored in a valid receive transaction.

Matchability. This property requires no bounded adversary could receive data other than he/she received from the send transaction.

4 Protocol Overview

Suppose a user *Alice*, denoted as u_A, on Block A want to send data with encrypted content *con* to *Bob*, denoted as u_B, on Block B. Let $PRF^{addr}_x(\cdot)$, $PRF^{sn}_x(\cdot)$ and $PRF^{pk}_x(\cdot)$ denote three pseudorandom functions for a seed x. Each user u_i generates an address key pair $(addr_{pk,i}, addr_{sk,i})$, where $addr_{pk,i} = (a_{pk,i}, pk_{pkc,i})$ and $addr_{sk,i} = (a_{sk,i}, sk_{pkc,i})$, and a nullifier key nk. $a_{pk,i}$ is generated as $PRF^{addr}_{a_{sk}}(0)$. nk is generated as $PRF^{addr}_{a_{sk}}(1)$.$(pk_{pkc,i}, sk_{pkc,i})$ are key-private encryption scheme. Here, we outline the protocol in three steps:

(1) u_A generates randomness r, s, and ρ, where ρ is the data's serial number randomness. Let $COMM$ denote a commit scheme and $PKC.Enc$ denote a public-key encryption scheme. Let $addr_{pk} := (a_{pk}, pk_{enc})$ be u_B's address pair. u_A encrypts data content with pk_{enc} and generates *con* at first. u_A commits the serial number in two steps:
   1. $k = COMM_r(a_{pk,1}||\rho)$
   2. $cm := COMM_s(con||k)$

   Then, u_A computes the ciphertext $Ct = PKC.Enc_{pk_{pkc}}(con, \rho, r, s)$. The tuple (con, k, s, cm, Ct) is the new transaction tx_{send}. The ledger will keep a CRH(collision-resistant hash)-based Merkle tree $CMList$ of all committed serial numbers (cm). If cm is already in the ledger, the transaction will be rejected. Logically, the data u_A sends to u_B is defined as $d := (addr_{pk}, v, \rho, r, s, cm)$.

(2) u_B can scan over the public ledger and find the transaction tx_{send}. The user then decrypts Ct and gets (con, ρ, r, s).
(3) When u_B wants to receive the data (or more than one received coins), u_B will generate a data d^{new} with newly encrypted content con^{new} and a *zk-SNARK* proof π_{RECEIVE} over the following statements:
For old data d, given the Merkle root rt, serial number sn, u_B knows d and address secret key $a_{sk,1}$ s.t.
– d is well-formatted.
– The address secret key matches the public key, i.e., $a_{pk} = PRF^{addr}_{a_{sk}}(0)$.
– The nullifier key matches the address secret key, i.e., $nk = PRF^{addr}_{a_{sk}}(1)$.
– The serial number is computed correctly, i.e., $sn = PRF^{sn}_{nk}(\rho)$.
– The data commitment cm appears as a leaf of Merkle-tree with root rt.
– New data d^{new} is well formatted.
– $con^{new} = PKC.Enc_{pk^{new}_{pkc}}(PKC.Dec_{sk_{pkc}}(con))$.

The receive transaction $tx_{receive} := (rt, sn, cm^{new}, con^{new}\pi_{\mathsf{RECEIVE}}, *)$ is appended to the ledger, where rt, sn are the Merkle root and the serial number for the old data. The relayer will verify the proof and check if all sn do not appear on the ledger. It will write the new data d^{new} on blockchain B if validated. Furthermore, we employ a message authentication code (MAC) scheme to prevent malleability attacks. A MAC is a code that authenticates a message's source and its integration. When receiving a data, the user samples a key pair (pk_{sig}, sk_{sig}) and use sk_{sig} sign every value associated with the $tx_{receive}$ transaction. The user also computes $h_{sig} := CRH(pk_{sig})$ and $h := PRF^{pk}_{a_{sk}}(h_{sig})$, which acts like a MAC to sign the secret address key. The user then modifies the statement to prove that h is computed correctly. The signature σ along with pk_{sig} are included in the $tx_{receive}$ transaction.

5 Conclusion

In this position paper, we proposed a data exchange protocol over multi-chain blockchain using zero-knowledge proof. The protocol leverages advanced cryptography algorithms and blockchain technologies to provide an efficient and private data transferring algorithm. This protocol preserves the anonymity of the original owner, and the system is secure against malicious users, and provides indistinguishability, transaction non-malleability, and matchability. In future work, we will implement the protocol on the Ethereum Test Network and Hyperledger Fabric and evaluate the performance of the proposed architecture. Moreover, we will simulate the protocol's security against various attack scenarios.

References

1. Almashaqbeh, G., Solomon, R.: SoK: privacy-preserving computing in the blockchain era. In: 2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P), pp. 124–139. IEEE (2022)

2. Ben Sasson, E., et al.: Zerocash: decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy, pp. 459–474 (2014). https://doi.org/10.1109/SP.2014.36
3. Bitansky, N., Chiesa, A., Ishai, Y., Paneth, O., Ostrovsky, R.: Succinct non-interactive arguments via linear interactive proofs. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 315–333. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_18
4. Bleumer, G.: Unlinkability. In: van Tilborg, H.C.A., Jajodia, S. (eds.) Encyclopedia of Cryptography and Security, p. 1350. Springer, Boston (2011). https://doi.org/10.1007/978-1-4419-5906-5_236
5. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, STOC 1988, pp. 103–112. Association for Computing Machinery, New York (1988). https://doi.org/10.1145/62212.62222
6. Bowe, S., Chiesa, A., Green, M., Miers, I., Mishra, P., Wu, H.: Zexe: enabling decentralized private computation. In: 2020 IEEE Symposium on Security and Privacy (SP), pp. 947–964. IEEE (2020)
7. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy (SP), pp. 315–334 (2018). https://doi.org/10.1109/SP.2018.00020
8. Ekblaw, A., Azaria, A., Halamka, J.D., Lippman, A.: A case study for blockchain in healthcare: "MedRec" prototype for electronic health records and medical research data. In: Proceedings of IEEE Open & Big Data Conference, vol. 13, p. 13 (2016)
9. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing, STOC 1985, pp. 291–304. Association for Computing Machinery, New York (1985). https://doi.org/10.1145/22145.22178
10. Gordon, W.J., Catalini, C.: Blockchain technology for healthcare: facilitating the transition to patient-driven interoperability. Comput. Struct. Biotechnol. J. **16**, 224–230 (2018)
11. Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., Schofnegger, M.: Poseidon: a new hash function for zero-knowledge proof systems. In: 30th USENIX Security Symposium (USENIX Security 2021), pp. 519–535. USENIX Association (2021). https://www.usenix.org/conference/usenixsecurity21/presentation/grassi
12. Griffor, E., Greer, C., Wollman, D., Burns, M.: Framework for cyber-physical systems: volume 1, overview (2017). https://doi.org/10.6028/NIST.SP.1500-201
13. Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_19
14. He, Q., Xu, Y., Liu, Z., He, J., Sun, Y., Zhang, R.: A privacy-preserving internet of things device management scheme based on blockchain. Int. J. Distrib. Sens. Netw. **14**(11), 1550147718808750 (2018)
15. Hopwood, D., Bowe, S., Hornby, T., Wilcox, N.: Zcash protocol specification. GitHub: San Francisco, CA, USA, p. 1 (2016)
16. Huh, S., Cho, S., Kim, S.: Managing IoT devices using blockchain platform. In: 2017 19th International Conference on Advanced Communication Technology (ICACT), pp. 464–467. IEEE (2017)
17. Jiang, S., Cao, J., Wu, H., Yang, Y., Ma, M., He, J.: Blochie: a blockchain-based platform for healthcare information exchange. In: 2018 IEEE International Conference on Smart Computing (SmartComp), pp. 49–56. IEEE (2018)

18. Li, A., Serena, L., Zichichi, M., D'Angelo, G., Tang, S.K., Ferretti, S.: Modelling of the internet computer protocol architecture. In: 4th International Congress on Blockchain and Applications, BLOCKCHAIN 2022 (2022, to appear)
19. Merkle, R.C.: A digital signature based on a conventional encryption function. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 369–378. Springer, Heidelberg (1988). https://doi.org/10.1007/3-540-48184-2_32
20. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Decentralized Business Review, p. 21260 (2008)
21. Nguyen, G.T., Kim, K.: A survey about consensus algorithms used in blockchain. J. Inf. Process. Syst. **14**(1), 101–128 (2018)
22. Ouaddah, A., Elkalam, A.A., Ouahman, A.A.: Towards a novel privacy-preserving access control model based on blockchain technology in IoT. In: Rocha, Á., Serrhini, M., Felgueiras, C. (eds.) Europe and MENA Cooperation Advances in Information and Communication Technologies. AISC, vol. 520, pp. 523–533. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-46568-5_53
23. Pieroni, A., Scarpato, N., Di Nunzio, L., Fallucchi, F., Raso, M.: Smarter city: smart energy grid based on blockchain technology. Int. J. Adv. Sci. Eng. Inf. Technol. **8**(1), 298–306 (2018)
24. Poelstra, A., Back, A., Friedenbach, M., Maxwell, G., Wuille, P.: Confidential assets. In: Zohar, A., et al. (eds.) FC 2018. LNCS, vol. 10958, pp. 43–63. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-662-58820-8_4
25. Rathore, H., Mohamed, A., Guizani, M.: A survey of blockchain enabled cyber-physical systems. Sensors **20**(1), 282 (2020)
26. Serena, L., Li, A., Zichichi, M., D'Angelo, G., Ferretti, S., Tang, S.K.: Simulation of the internet computer protocol: the next generation multi-blockchain architecture. In: 2022 IEEE/ACM 26th International Symposium on Distributed Simulation and Real Time Applications (DS-RT), pp. 119–126. IEEE (2022)
27. Steffen, S., Bichsel, B., Baumgartner, R., Vechev, M.: Zeestar: private smart contracts by homomorphic encryption and zero-knowledge proofs. In: 2022 IEEE Symposium on Security and Privacy (SP), pp. 179–197 (2022). https://doi.org/10.1109/SP46214.2022.9833732
28. Steffen, S., Bichsel, B., Gersbach, M., Melchior, N., Tsankov, P., Vechev, M.: zkay: specifying and enforcing data privacy in smart contracts. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 1759–1776 (2019)
29. Wood, G., et al.: Ethereum: a secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper **151**(2014), 1–32 (2014)