International Journal of Social Sciences and Management Review

Volume: 05, Issue: 03 " May - June 2022"

ISSN 2582-0176

ZERO KNOWLEDGE PROOF FOR HOMOMORPHICALLY ENCRYPTED TRANSACTIONS IN 5IRE BLOCKCHAIN

VILMA MATTILA, PRATEEK DWIVEDI, PRATIK GAURI & DHANRAJ DADHICH
5ire (Sustainable Distributed Computing)
Unit Number 101, IFZA Dubai - Building A2, Dubai Silicon Oasis,
United Arab Emirates

https://doi.org/10.37602/IJSSMR.2022.5320

ABSTRACT

Since public blockchains are permissionless, it is subject to passive adversarial attack. In 5irechain we have addressed the security problem related to this passive adversarial activity by applying 5ireHE, a homomorphic encryption technique that encrypts the transactional details using the receiver's public key. Since the transaction is encrypted by the receiver's public key, it is harder for other validators to validate the transaction in 5ire. In this paper, we introduce ZKP for validating the transaction in a sense that validator can check if the sender's previous balance and the remaining balance are in harmony with the amount of the transaction despite the difference in public keys that are used for the encryption of transaction and the encryption of account balance.

1.0 INTRODUCTION

In [8] homomorphic encryption technique, called 5ireHE, is proposed for the 5ire blockchain to encrypt transaction data. If Transaction data is encrypted by the receiver's public key, then it is difficult to prove to the external versifiers about the authenticity and validity of the transaction. We are the first to propose zero-knowledge proof (ZKP) in blockchain to prove transactional validity to the external validators. Unlike a normal scenario, where there is one prover, who needs to prove the secret to the external world without revealing the secret, here there are two provers - the sender and the receiver. Let the balance in the sender's account before the transaction is p and the amount of the transaction is t. What is to be proved here is that $t < p$, i.e., one should not be able to pay more than what she has in her account. Since in 5irechain, the amount of transaction t is encrypted with the receiver's public key and the balance in the sender's account is encrypted using the sender's public key, the underlying need for ZKP is different from the standard ZKP. In this heterogeneous environment, both parties are expected to perform some extra computation on some extra data. Though there are chances of some negligible leakage through these auxiliary data, however, we note that in 5irechain, since validators are trusted and protected by a sustainable scoring mechanisms, there will never be a case when a validator may attempt to act as an active adversary. This is because the advantage of learning through this negligible leakage is less than the award provided to honest validators through the sustainable scoring system in the 5irechain.

2.0 RELATED WORK

Zero-knowledge proof in permissionless blockchain along with encrypted trans-actional details has a compelling advantage. With the growing popularity of blockchain, several research works have been conducted to explore privacy in the context of blockchain [1–7, 9]. Rather than customized solutions for specific use cases, we here focus on the privacy of transaction data from passive adversaries. Since a blockchain-based IoT network is public, so transactional details and encrypted keys are exposed to everybody in that network. Thus, anybody in the network can infer critical information of users from this public infrastructure. In [5] authors discussed the privacy issues caused due to the integration of blockchain in IoT applications. The work in [7] evaluates blockchain's roles in strengthening cybersecurity and protecting privacy. Owing to the majority of data being stored in the cloud, the authors also provided a comparison as to how blockchain performs vis-vis the cloud in various aspects of security and privacy. In [3] authors pro-posed a blockchain-based model for data storage and addressed the problem of data synchronization. To improve the performance of users' workstations, they designed the DEPLEST method to be embedded in the front of the existing database to capture sensitive data. They also implemented a stochastic homomorphic elliptic curve cryptography (SHECC) encryption model to improve data security and efficiency. In [6], a private smart contract called HAWK was proposed which is based on ZKPs. Hawk assumed a semi-trusted manager, who is trusted with pro- testing the privacy of the users' inputs but not for correctness of computation. In [4] Ekiden was proposed which relies on trusted hardware rather than a trusted manager. Following this, research has been conducted to avoid trusted parties or hardware. In [2] Zether was proposed which targeted smart contract privacy for Ethereum. Its reliance on additively homomorphic encryption restricts its functionality to private currency transfer and a limited class of private smart contracts. The authors noted that though Zether upholds anonymity, this feature cannot be implemented on Ethereum as the cost will exceed the gas limit per block. Zkay [9] proposed a compiler for private smart contracts. Zkay follows the pure ZKP approach by overloading end users and depends on off-chain coordina- tion to handle multi-user inputs. Zexe in [1] enhanced privacy further by also preserving function privacy. Following the pure ZKP approach, Zexe operates in the UTXO-based model which attempted to limit the supported functionality to extending Zerocash scripts used to spend currency.

3.0 PROBLEM DEFINITION

In 5ire chain, the issue of scalability is addressed by maintaining parallel chains. To uphold this, 5ire chain allows multiple transaction pools, one for each parallel chain. Whenever the number of transactions in a transaction pool surpasses a threshold, that particular transaction pool is divided into two different pools. For this purpose, we make use of hash functions. A transaction goes into one of the pools depending upon the hash value of the public key of the transaction-sender.

Roughly speaking, if there are n transaction pools, each pool is dynamically assigned a number which is a bit-string of size at most logn. Thus from the pool of information, the sender's identity can be derived publicly, however, if the amount of the transaction appears in plain text, then that may raise the privacy concern for both the sender and receiver. Encrypting the amount will come at the cost of losing the usability of the data in the blockchain. In [8] This problem is solved by introducing 5ireHE which encrypts the transaction data using homomorphic encryption. Since transaction data is encrypted, there should be a mechanism for validators to check the validity of the transaction.

4.0 OVERVIEW OF THE SCHEME

Here we restate the system overview, the threat model, and the homomorphic encryption process from [8] for smooth reading so as to facilitate readers under-standing how ZKP is plugged at the top of this structure.

4.1 System Overview

5ire introduces the nested blockchain where nodes can create concurrent blocks and maintain smaller chains, which can later be merged into 5irechain. Thus, instead of having a linear blockchain 5irechain can be view upon as to be a tree-structured blockchain followed by a 5ire block, which will nest the smaller chains together. In order to process the transactions at scale, it is crucial for nodes from autonomous groups to be able to concurrently process transactions and form their own chains. Nested-Chains in 5irechain addresses this issue of scalability by maintaining the parallel chains. These chains are created on a need basis depending on the load on the network. However, once a chain is cre- ated it will continue adding blocks until the chain is joined with the 5irechain using a 5ire block. Figure 1 shows the structure of the nested-chain. The nested chains not only support the scalability in the blockchain, but it also enables us to support the creation of parallel chains without adding new nodes (Assemblers, Attesters/voters, ESG Experts). This essentially means that nodes will be run- ning multiple parallel chains on a single node, but as a separate process. 5ire will use the scheduling algorithms to make sure the maximum utilization of the nodes. Therefore, unlike the conventional blockchains where nodes will sit idle and wait for their turn to create the block, the nodes in the 5ire ecosystem may get turns to create blocks into another parallel chain in the nested-chain. The nodes in all the parallel chains will be selected in a similar way i.e. based on their weights (Reliability Score, Stake, ESG score, and Randomized voting).

4.2 Threat Model

Suppose Alice is sending money to Bob. From the transaction block information, conventionally which is kept in the form of plain text, it is possible for a passive adversary to learn this transaction. It can be further observed that such an the adversary can extract a fair amount of wallet information from Alice by tracking all the inflow and outflow of money corresponding to Alice’s account by noting all transactions involving Alice. Similarly, Bob and others' wallet information can be extracted.

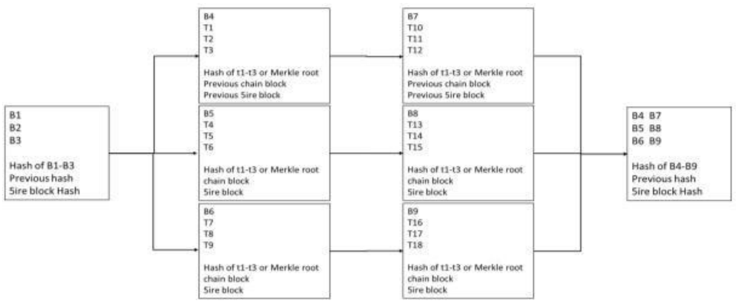

Fig. 1: 5ire Nested Chain

4.3 Overview of HE architecture in 5ire blockchain

Figure 2 represents the HE architecture in 5ire blockchain. Let Alice wants to pay Bob and she creates a transaction block for this in a transaction pool which is determined by the hash value of Alice's transaction public key. Unlike traditional block creation, here Alice uses Bob's public key for Homomorphic Encryption to encrypt the amount while creating the block. This block is then broadcasted to all members in the 5ire blockchain corresponding to that transaction pool for validation. Once it is verified, according to 5ire architecture, it is added to 5ire block and the money moves to Bob's wallet in an encrypted form. Bob can vary the amount of decryption and can perform the aggregate operation of the all credited amount directly on the encrypted wallet information.

4.4 The Scheme

4.5 Cryptographic tools

Pseudo-Random Prime number Generator (PRNG). PRNG takes as input security parameter 1γ and outputs γ bit long prime numbers. Since this is a probabilistic algorithm, we denote it as p ←−$ PRNG (1γ).

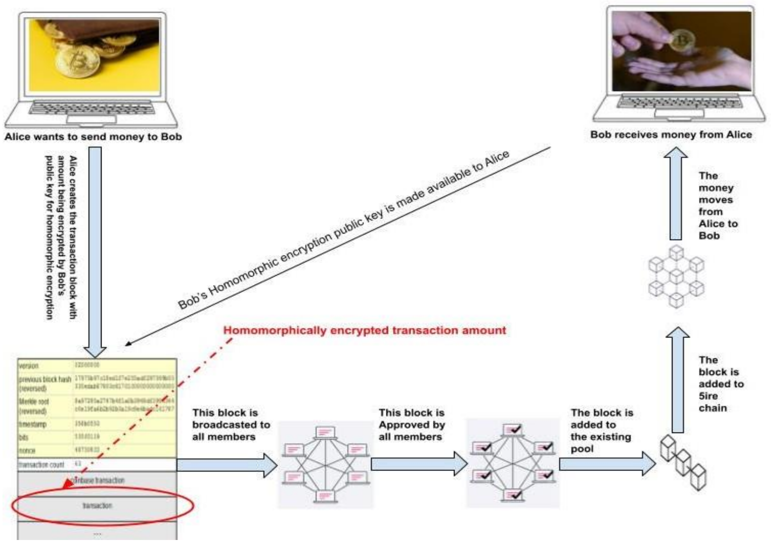
Alice wants to send money to Bob
Alice creates the transaction block with amount being encrypted by Bob's public key for homomorphic encryption
Bob's Homomorphic encryption public key is made available to Alice
Homomorphically encrypted transaction amount
This block is broadcasted to all members
This block is Approved by all members
The block is added to the existing pool
The block is added to 5ire chain
The money moves from Alice to Bob
Bob receives money from Alice
version
previous block hash (reversed)
Merkle root (reversed)
timestamp
bits
nonce
transaction count
coinbase transaction
transaction

Fig. 2: Secure Transaction data Computing in 5ire nested chain architecture using Homomorphic Encryption

Paillier encryption. This public key cryptosystem has three algorithms, namely KeyGen, Encryption and Decryption.

- KeyGen(γ) : Choose two γ-bit prime numbers p and q randomly and inde- pendently of each other such that gcd(pq, (p - 1)(q - 1)) = 1. This prop- erty is assured if both primes are of equal length. Compute n = pq and λ = lcm (p - 1, q - 1). Select random integer g where g € $_{n*}$$^{2.}$ Ensure n divides the order of g by checking the existence of the following modular multiplicative inverse: $\mu = (L\,(g^{\lambda} \bmod n^{2}))^{-1} \bmod n$, where function L is defined as $L(x) = x-1$. Finally set pk = (n, g) and sk = (λ, μ)

- Encryption(m, pk) : Let m be a message to be encrypted where $0 < m < n$. Select random $0 < r < n$. Compute ciphertext as: c = gm · rn mod n2.
- Decryption(m, pk) : Let c be the ciphertext to decrypt, where $c \in Z^{*}_{n}{}^{2}$. Compute the plaintext message as: $m = L\,(c^{\lambda} \bmod n^{2}) \cdot \mu \bmod n$.

5ire Block Structure. A standard 5ire block is composed of a header and a body, where a header contains the hash of previous block, a timestamp, Nonce and the Merkle root. The Merkle root is the root hash of a Merkle tree which is stored in the block body.

We denote a 5ire block as = , , where is parsed as = h, t, no, m , where h is previous hash, t is the timestamp, no is the nounce and m is the markle

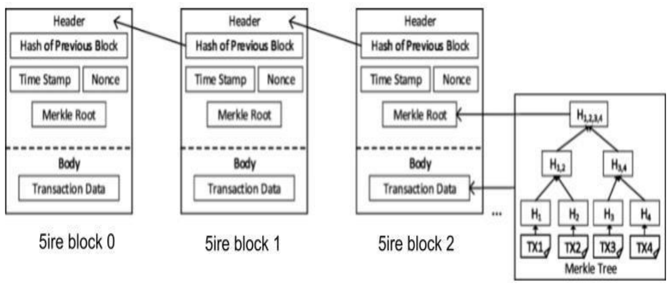

Fig. 3: 5ire Block Structure

root. The body of the block is parsed as = sid, rid, a, where sid is the sender's ID, rid is the receiver's ID and a is the amount being sent.

5.0 THE ZERO KNOWLEDGE PROOF IN 5IRECHAIN

Let, in a 5ire transaction Alice is the sender and Bob is the receiver. Also let Alice has the current balance p and is sending the amount t. Let, after transaction the remaining balance in Alice's account is r.

Validators want to check that $t = p - r$ and $r > 0$

Let the public keys for transaction of Alice and Bob are (g1, n1) and (g2, n2) respectively. Alice computes T , P and R corresponding to t p and r. The com- putation for P and R takes Bob's public key which is available. Computations are done as follows:

$$T = g_1^{t} \cdot k_a^{n_1} \bmod n_1^{2}. \quad (1)$$

$$P = g_2^{p} \cdot k_b^{n_2} \bmod n_2^{2}. \quad (2)$$

$$R = g_2^{r} \cdot k_c^{n_2} \bmod n_2^{2}. \quad (3)$$

Now, upon receipt of T , P and R validator first computes

$$F = \frac{P}{R} = g_2^{p-r} \cdot k^{n_2} \bmod n_2^2,$$

Where k = kb/kc

When k = ka, the validator can verify as follows:

Validator will then ask Bob to provide Q1 = $g^t \bmod n^2$

and the validator will ask the sender to provide Q2 = $g^{p-r} \bmod n^2$

Finally, validator will check if

$$Q_2^{n_1} . T^{n_2} \bmod n_2^2 == Q_1^{n_2} . F^{n_1} \bmod n_1^2.$$

In the following theorem, we study the correctness of the ZKP as described above. Theorem 1. For a 5ire transaction, of p, r and t are the previous and remaining balance of the sender and if t is the amount of the transaction, then the following hold

$$Q_2^{n_1} . T^{n_2} \ mod\ n_1^2 n_2^2 = Q_1^{n_2} . F^{n_1} \ mod\ n_1^2 n_2^2 \ if\ k = k_a$$

Proof. Let the randomness chosen for transaction encryption is ka and also let the randomness kc for encrypting the remaining balance is chosen by setting kc = kb . Since ka is uniformly distributed over Zp, so is k_c. Therefore, k = k_a $\frac{k_b}{k_c}$ = Now,

$$\begin{aligned} Q_2^{n_1} . T^{n_2} \bmod n_1^2 n_2^2 &= g_2^{p-r^{n_1}} \cdot g_1^{t^{n_2}} \cdot k_a^{n_1 . n_2} \bmod n_1^2 n_2^2 \\ &= g_1^{n_2 . t} \cdot g_2^{n_1 . t} \cdot k_a^{n_1 n_2} \bmod n_1^2 n_2^2 \\ &= g_1^{n_2 . t} \cdot g_2^{n_1 . t} \cdot k^{n_1 . n_2} \bmod n_1^2 n_2^2 \\ &= Q_1^{n_2} . F^{n_1} \bmod n_1^2 n_2^2 \end{aligned}$$

5.1 5ireHE protocol with ZKP

Here we present the 5ireHE scheme from [8] along with the ZKP for the 5ireHE. We term this as 5ireHE ZKP. The 5ireHE presented in [8] is restated here for smooth reading.

5ireHE is a tuple of PPT algorithms which is presented as

5ireHE ZKP = (KeyGen, Encrypt, Decrypt, Encrypt5ireWallet, Decrypt5ireWallet, 5ireZKP). In the following figures we present these algorithms.

Algorithm 1 KeyGen()

```
Data: Security parameter $\gamma$
Result: Homomorphic Encryption keypair $(pk, sk) p$
$\xleftarrow{\$} PRNG(1^{\gamma})$;
$q \xleftarrow{\$} PRNG(1^{\gamma})$;
Compute $n = pq$; $\lambda = lcm(p-1, q-1)$;
$g \xleftarrow{\$} Z^{*}_{n^2}$ Ensure $n|o(g)$ by checking the existence of the following modular multiplica-
tive inverse: $\mu = (L(g^{\lambda} \bmod n^2))^{-1} \bmod n$, where function $L$ is defined as $L(x) = \frac{x-1}{n}$.
set $pk = (n, g)$ and $sk = (\lambda, \mu)$
```

Algorithm 2 Encrypt()

```
Data: plaintext $m$, HM public key $pk = (n, g)$
Result: ciphertext $c$
$r \xleftarrow{\$} Z_n \setminus \{0\}$;
Compute $c = g^m \cdot r^n \bmod n^2$;
```

Algorithm 3 Decrypt()

```
Data: ciphertext $c$, HM secret key $sk = (\lambda, \mu)$
Result: plaintext $m$
Compute $m = L(c^{\lambda} \bmod n^2) \cdot \mu \bmod n$
```

Algorithm 4 Encrypt5ireWallet ()

```
Data: receiver's publickey $pk_{rid} = (n, g)$, the 5ire transaction block BC
Result: secure block $\overline{BC}$
parse the block BC = ⟨H, B⟩
parse the body B = $\langle sid, rid, a \rangle$;
compute $c$ = Encrypt($a$, $pk_{rid}$);
Set the timestamp $t$ to the current time; Update
the markle tree root $m$;
Set $no = \perp$ /* this is set by miners */
Set H = $\langle h, t, no, m \rangle$;
Update $B$ as $B = \langle sid, rid, c \rangle$
set $\overline{BC} = \langle H, B \rangle$;
return BC;
```

Algorithm 5 Decrypt5ireWallet()

```
Data: receiver's secretkey $sk_{rid} = (\mu, \lambda)$, the encryptrd 5ire transaction block $\overline{BC}$
Result: the amount $a$ in plaintext
parse the block BC = ⟨H, B⟩
parse the body B = $\langle sid, rid, c \rangle$;
compute $a$ = Decrypt($c$, $sk_{rid}$);
return $a$;
```

```
Algorithm 6 5ireZKP ()
Data: Sender's public key $pk_{sid} = (n_1, g_1)$ and receiver's publickey pkrid = $(n_2, g_2)$
      and oracle access to sender (.) and receiver (.)
Result: $b$
Compute $T, P, R$ using sender (.) functionality access as follows
      $T = g_1^t \cdot k_a^{n1} \bmod n_1^2;$
      $P = g_2^p \cdot k_b^{n2} \bmod n_2^2;$
      $R = g_2^r \cdot k_c^{n2} \bmod n_2^2;$
Compute using receiver(.), $Q_1 = g_1^t \bmod n_1^2;$
Compute using sender(.), $Q_2 = g_2 \bmod n_2^2;$
Finally compute $b = (Q_2^{n1} . T^{n2} \bmod n_1^2 n_2^2 == Q_1^{n2} . P^{n1} \bmod n_1^2 n_2^2) ? 1 : 0;$
return $b$;
```

6.0 CONCLUSION

In the public blockchain, the integrity of data is maintained due to the security constructs of blockchain. However this does not guarantee the safety against leakage of transaction data due to passive adversarial presence. To address this, in [8] 5ireHE encryption technique was proposed to encrypt transaction data. Though this solves the security issue, however, to prove the validity of transac- tion to an external validator remains an open problem to be addressed. In this paper we proposed a zero knowledge proof based method to prove the validity of transaction to an external validator. We coined the term 5ireZKP for this. The 5ireHE of [8] along with 5ireZKP is termed as 5ireHE ZKP.

REFERENCES

Sean Bowe, Alessandro Chiesa, Matthew Green, Ian Miers, Pratyush Mishra, and Howard Wu. Zexe: Enabling decentralized private computation. In 2020 IEEE Symposium on Security and Privacy (SP), pages 947–964. IEEE, 2020.

Benedikt Bu¨nz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: To- wards privacy in a smart contract world. In International Conference on Financial Cryptography and Data Security, pages 423–443. Springer, 2020.

Yun Chen, Hui Xie, Kun Lv, Shengjun Wei, and Changzhen Hu. Deplest: A blockchain-based privacy-preserving distributed database toward user behaviors in social networks. Information Sciences, 501:100–117, 2019.

Raymond Cheng, Fan Zhang, Jernej Kos, Warren He, Nicholas Hynes, Noah Johnson, Ari Juels, Andrew Miller, and Dawn Song. Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts. In 2019 IEEE European Symposium on Security and Privacy (EuroS&P), pages 185–200. IEEE, 2019.

Muneeb Ul Hassan, Mubashir Husain Rehmani, and Jinjun Chen. Privacy preserva- tion in blockchain based iot systems: Integration issues, prospects, challenges, and future research directions. Future Generation Computer Systems, 97:512–529, 2019.

Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papaman- thou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In 2016 IEEE symposium on security and privacy (SP), pages 839–858. IEEE, 2016.

Nir Kshetri. Blockchain's roles in strengthening cybersecurity and protecting pri- vacy. Telecommunications policy, 41(10):1027–1038, 2017.

Vilma Mattila, Prateek Dwivedi, Pratik Gauri, and MD Ahbab. Homomorphic encryption in 5ire blockchain. International Journal of Social Sciences and Man- agement Review, 05, 2022.

Samuel Steffen, Benjamin Bichsel, Mario Gersbach, Noa Melchior, Petar Tsankov, and Martin Vechev. zkay: Specifying and enforcing data privacy in smart contracts. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communi- cations Security, pages 1759–1776, 2019.