International Journal of Social Sciences and Management Review

Volume: 05, Issue: 02 " March - April 2022"

ISSN 2582-0176

HOMOMORPHIC ENCRYPTION IN 5IRE BLOCKCHAIN

VILMA MATTILA, PRATEEK DWIVEDI, PRATIK GAURI & MD AHBAB

5ire (Sustainable Distributed Computing) 160 City Rd, London,
United Kingdom, Corresponding Author: ahbab@5ire.org

https://doi.org/10.37602/IJSSMR.2022.5219

ABSTRACT

With the advent of blockchain technology, decentralized system is gaining huge popularity as it provides a new solution for data storage and sharing as well as keeping privacy in place. Blockchains are of two categories; public and private. Public blockchains are permissionless, mostly used for exchanging and mining cryptocurrency, where anyone can join and thus exposed to the risk of a privacy breach. If the content is the transaction information, one might opt for not sharing these data in the public domain. One solution could be to encrypt the information, but that comes at the cost of losing the usability of the data. This paper investigates the security problem related to this passive adversarial activity and proposes a new technology that leverages the best parallel chain architecture of 5ire blockchain and homomorphic encryption (HE) so as to retain the advantages of a public blockchain without compromising the privacy of transaction information. We have coined the term 5ireHE for this encryption architecture. We achieve the security protection and integrity check of wallet data by enforcing the 5ireHE which is efficient.

1.0 INTRODUCTION

With the advent of Blockchain technology, a decentralized system is a reality that provides a new solution for data storage and sharing while keeping privacy in place. Publicly available data can be secured by encrypting, however, this re- dux the usability of the data in a sense that unless decrypted, no operation can be performed on the encrypted data. Towards this, the idea of homomorphic encryption has been conceptualized where certain operations are possible on the encrypted data directly without having to decrypt it. When combined with blockchain, these features of homomorphic encryption could lead to a new platform of agile and yet highly resilient secure computing strategies. Although the idea of homomorphic encryption was introduced in 1978, however, the first fully homomorphic encryption scheme, in short, FHE, was becoming a reality following the phenomenal work done by Gentry et al [5]. The goal of homomorphic encryption is to allow an infinite number of additions or multiplications of encrypted data with the target of having the ciphertext that would be produced, had the same operations been performed on the corresponding plaintexts. The problem is that designing such an encryption algorithm is really hard. As a result, there are a few different "types" of homomorphic encryption that describe how close a particular algorithm is to that final goal. Despite being an ideal solution for solving a variety of major business challenges, there is no commercial use of FHE so far. One major problem with FHE is that it is not efficient. Depending on the degree of freedom in terms of achieving full functionality over the ciphertext domain, there are three categories of homomorphic encryption; fully homomorphic encryption, somewhat homomorphic encryption, and partially homomorphic encryption.

From the pool information in 5irechain, the sender's identity can be derived publicly which is acceptable, and also this is the basic assumption of 5irechain. However, we note that if the amount of the transaction appears in plain text, then that may raise privacy concerns for both the sender and receiver. Encrypt- in the amount will come at the cost of losing the usability of the data in the blockchain. 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 adversary can extract a fair amount of wallet information of Alice by tracking all the inflow and outflow of money corresponding to Alice's ac- count by noting all transactions involving Alice. Similarly, Bob and others' wallet information can be extracted.

5ireHE. Figure 3 represents the HE architecture in the 5ire blockchain. We coined the term 5ireHE for this. 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 the 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 operations of all credited amounts directly on the encrypted wallet information.

This paper proposes a new technology that leverages the best parallel chain archetecture of 5ire blockchain and homomorphic encryption (HE). We achieve the security protection and integrity check of wallet data by enforcing HE which is efficient. On the one hand, HE is an efficient solution for privacy-preserving wallet computation but on the other hand, HE is susceptible to IND-CC2 attack. We introduce a second layer of authentication so as to provide safeguards against such attacks. On the one hand, homomorphic encryption is an efficient solution for privacy-preserving wallet computation but on the other hand, it is susceptible to IND-CC2 attack. It is to be noted that the IND-CC2 attack assumes access to the decryption oracle. However, we observe that since the 5ireHE architecture enforces each individual to generate the 5ireHE key pair and that the transaction to be encrypted by the public key of the receiver for which the corresponding private key is to be kept solely by the receiver, IND-CPA security of transaction will suffice to assert the transaction security in 5ireHE. We introduce a second layer of authentication so as to provide safeguards against such attacks.

2.0 RELATED WORK

With the growing popularity of blockchain, several research works have been conducted to explore privacy in the context of blockchain [1–4, 6–8, 10]. 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 [6] authors discussed the privacy

Volume: 05, Issue: 02 " March - April 2022"

ISSN 2582-0176

issues caused due to the integration of blockchain in IoT applications. The work in [8] evaluates blockchain's roles in strengthening- ing 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 per- forms vis-vis the cloud in various aspects of security and privacy. In [3] authors proposed 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 homo- morphic elliptic curve cryptography (SHECC) encryption model to improve data security and efficiency. In [7], a private smart contract called HAWK was proposed which is based on ZKPs. Hawk assumed a semi-trusted manager, who is trusted with protecting the privacy of the users' inputs but not for the 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. Okay [10] proposed a compiler for private smart contracts. Okay follows the pure ZKP approach by overloading end users and depends on off-chain coordination to handle multi-user inputs. Sexe 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 the 5ire chain, the issue of scalability is addressed by maintaining parallel chains. To uphold this, the 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.

4.0 OVERVIEW OF THE SCHEME

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

Volume: 05, Issue: 02 " March - April 2022"

ISSN 2582-0176

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 created 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 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 running- in 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 adversary can extract a fair amount of wallet information of 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.

4.3 Privacy Protection within 5ireChain Homomorphic Encryption

The blockchain is composed of multiple blocks and each node in the 5ireChain network stores the same blockchain. The emanate block includes the blockhead and the block body.

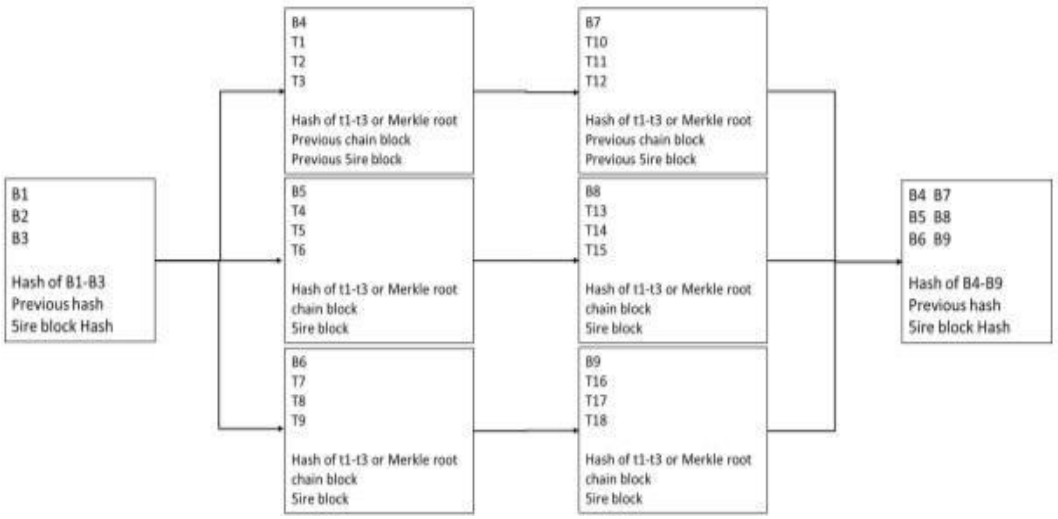


Fig. 1: 5ire Nested Chain

Where the Block Body contains information stored as meta-data, and the Block Head holds the input metadata information like version, timestamp, characteristic value, difficulty value, etc. The next block of the emanate block is known as the parent block, and the proceeding

Volume: 05, Issue: 02 " March - April 2022"

ISSN 2582-0176

next block also includes the blockhead and the block body, so each block is in the blockchain filled with a similar set of information. The block data stored in the block is associated with the block data stored in the parent block, which ensures the security of the input information in the block. The steps of this algorithm are as follows.

Step 1: Key Generation: Initialize random primes p and q and meet the condition of: gcd(PQ, (p − 1)(q − 1)) = 1

Module calculation n = pq, λ = lcm(p−1, q−1), where lcm is to seek the least common multiple of p −1 and q −1. Select the random number g ∈ Z∗n2 and meet μ = L g mod n mod n the greatest common divisor of L g mod n and n Zn∗ 2 represents the set of integers coprime to n2 Zn2. The public key for encryption is (n, g) and the private key is (λ, μ). Select integer in encryption and decryption process, r(r ∈ Zn2), and the plaintext is m(m ∈ Zn) and m < n.

Step 2: (encryption→− Enc(m, pk)): Let m be the information to be encrypted and m ∈ Zn. Compute the ciphertext: c from m mod n. The encryption process is c = E(m) = gm · rn mod n2.

where c is the ciphertext data corresponding to the plaintext m and c Z∗n2. Mark the encryption algorithm as c=E(m, r). It may be noted that for the same ciphertext m, the value of r, which is randomly selected in the encryption process may be different and so is the corresponding ciphertext data c. This property ensures the semantic security of ciphertext data.

Step 3: (proxy re-encryption): Compute the public key and private key (Rsk, Rpk).

The re-encryption ciphertext is generated by the RSA algorithm, and the public key Rpk is sent to the server.

Step 4: (decryption→− Dec(c, sk)): Here the ciphertext c ∈ Z∗n2 for which following is computed: m = Dec(c, sk) = L(cλ mod n2) · μ mod n

4.4 Overview of HE architecture in 5ire blockchain

Figure 2 represents the HE architecture in the 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 the 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.

International Journal of Social Sciences and Management Review

Volume: 05, Issue: 02 " March - April 2022"

ISSN 2582-0176

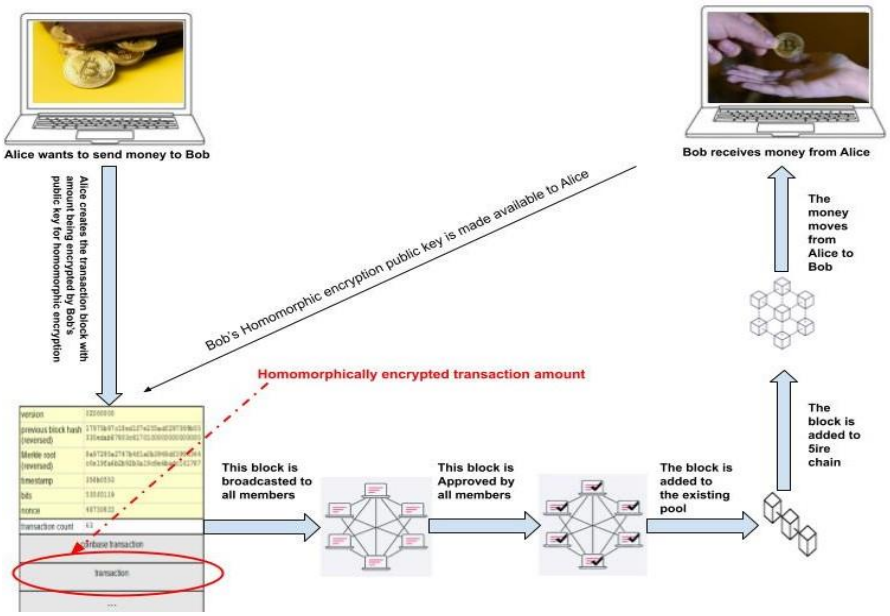


Fig. 2: Secure Transaction data computing in 5ire nested chain architecture using Homomorphic Encryption

5.0 THE SCHEME

5.1 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).

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 independently 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: μ = (L(gλ mod n2))−1mod 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 ∈ Z∗n2. Compute the plaintext message as: m = L(cλ mod n2) · μ mod n.

5ire Block Structure. A standard 5ire block is composed of a header and a body, where a header contains the hash of the 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.

Volume: 05, Issue: 02 " March - April 2022"

ISSN 2582-0176

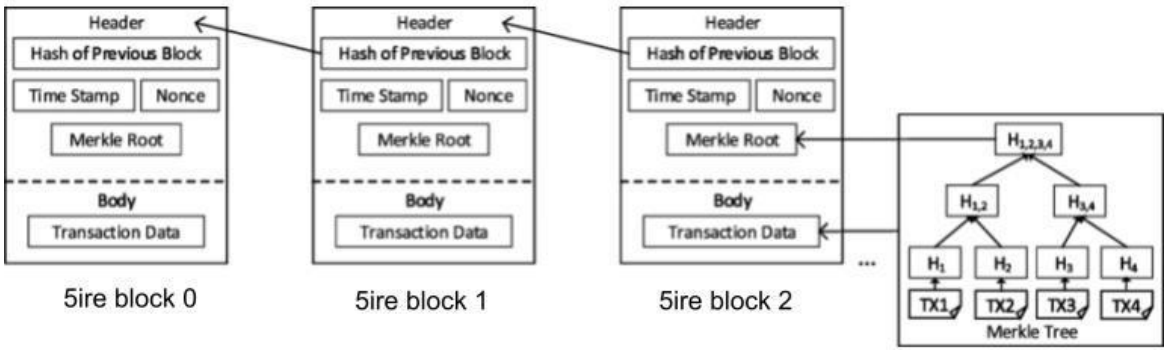


Fig. 3: 5ire Block Structure

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

5.2 5ireHE protocol

The HM5ire scheme is applied on top of a public smart contract-enabled nested 5irechain. It can be viewed as the mechanism needed to support privacy-preserving transactions from passive adversaries.

Here we present the 5ireHE scheme which focuses on the new modules needed to support privacy-preserving transactions where the receiver can directly operate on the encrypted wallet encryption.

5ireHE is a tuple of PPT algorithms which is presented as 5ireHE = (KeyGen, Encrypt, Decrypt, Encrypt5ireWallet, Decrypt5ireWallet). In the following figures, we present these algorithms.

Algorithm 1 KeyGen ()
Data: Security parameter γ
Result: Homomorphic Encryption
keypair (pk, sk) $p \xleftarrow{\$} PRNG\ (1^{\gamma})$;
$q \xleftarrow{\$} PRNG\ (1^{\gamma})$;
Compute $n = pq$; $\lambda = LCM(p - 1, q - 1)$;
$g \xleftarrow{\$} Z^*$; Ensure $n|o(g)$ 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 $L(x) = \frac{x-1}{n}$
set $pk = (n, g)$ and $sk = (\lambda, \mu)$

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

Volume: 05, Issue: 02 " March - April 2022"

ISSN 2582-0176

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 BC
parse the block BC = ⟨H, B⟩ parse the body B = ⟨*sid*, *rid*, *a*⟩; 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 = ⟨*h*, *t*, *no*, *m*⟩; Update B as B = ⟨*sid*, *rid*, *c*⟩ set BC = ⟨*H*, *B*⟩; return BC;

Algorithm 5 Decrypt5ireWallet ()

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

5.3 Security of the 5ireHE protocol

A passive adversary in 5ireHE is one who can passively see all the 5ire transactions and takes no actions in beyond recording the transactions and attempting to learn about the plaintext of documents. We introduce the notion of Transaction Indistinguishability to model the possible attacks and prove that 5ireHE is secure against such attacks.

Transaction Indistinguishability. Transaction indistinguishability ensures that the ledger produced by the 5ireHE protocol does not reveal additional information about transaction data beyond what can be inferred from what is publicly revealed. We define a Transaction-Indistinguishability-Game to capture this.

Volume: 05, Issue: 02 " March - April 2022"

ISSN 2582-0176

Definition 1. (Transaction Indistinguishability). An encrypted transaction in the blockchain satisfy transaction indistinguishability, or TI in short, if for all PPT adversaries A, AdvT I (γ) < negl(γ). We call such protocol TI-secure.

Exp^{TI} (γ) A
$(pk = (g, n), sk = (\mu, \lambda) \leftarrow KeyGen(1^{\gamma})$ $Encrypt(pk, \cdot)$ $(tx_0, tx_1, st) \leftarrow A_0$ $\$$ $d \leftarrow \{0, 1\}$ $c \leftarrow Encrypt(pk, tx_d)$ $Encrypt(pk, \cdot)$ $d' \leftarrow A_1 \quad (c, st)$ Return $(d = d')$
/* tx_0 and tx_1 are distinct with different amount. */   /* A_0, and A_1 cannot make a $Encrypt(pk, \cdot)$ query on an amountthat belongs to either tx_0 or tx_1, but not both. */

Before going into the proof, we introduce Decisional Composite Residuosity assumption here.

$DCR_B(\gamma)$
$g, q, n \leftarrow Setup(1^{\lambda})$, where $n = pq$ and $gcd(n, \phi(n)) = 1$
$(r_0, r_1) \leftarrow B_0(n)$, where $r_0 = r^n \bmod n^2$ for some $r \xleftarrow{\$} Z^*_{n^2}$ and $r_1 = r$
$b \xleftarrow{\$} \{0, 1\}$
$b' \leftarrow B_1(r_b, n)$
Return $(b = b')$

Definition 2 (Decisional Composite Residuosity Assumption (DCS)).

[9] We say DCR is hard if for all PPT adversary B = (B0, B1),

AdvDCS(γ) ≤ negl(γ).

Theorem 1. 5ireHE protocol is TI-secure.

Proof. Let there exists a PPT adversary A = (A0, A1) such that AdvT I (γ) >

negl(γ). For such A, we show that a PPT adversary C

simulate the adversary C as follows :

Volume: 05, Issue: 02 " March - April 2022"

ISSN 2582-0176

1. C runs KeyGen(1γ) and sets pk = (g, n), sk = (μ, λ).

A can solve DCS. We

2. C runs A Encrypt (pk,·) to obtain tx0 and tx1 ans state st.

3. C runs B0(n) to obtain c0 = rn mod n2, and c1 ←−

Z∗n2

4. C selects b ←−$ {0, 1} and d ←−

{0, 1}

5. C sets c = gtxd · rb mod n2

6. C runs and gets d′ ←− AEncrypt(pk,·)(c, st)

7. if d = d′, C outputs 1 else it outputs 0.

It can be noted that when b = 0, A1 receives a proper 5ireHE cipher

c = gtxd .r0 mod n2 = gtxd .rn mod n2.

But when b = 1, from A1's point of view, it is purely random over Z∗n2 as in this case as

r1 = r ←−$

Z∗n2 and c = gtxd .r1 mod n2 = gtxd .r mod n2

Thus ADVC(γ) = 1 AdvDCS(γ) ≥ AdvT I (γ) > negl(γ).

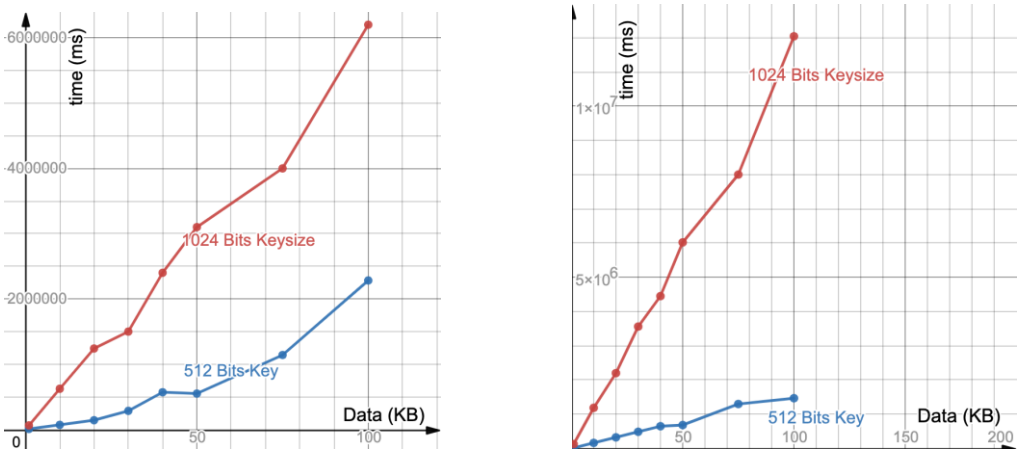
2 Clearly AdvDC ⊔⊓

5.4 Performance

In this section, we provide the performance results of our proposed scheme 5ireHE. We build a 5ireHE using c++ on the Linux platform. The implementation is done on Asus A Series Core i3 laptop ((4 GB/ 1 TB HDD) 90NB0652-M32310 XX2064D). We use C++ Library 'libpaillier' for the cryptographic primitives.

In Figure 4 we represent the transaction encryption and decryption timing using 5ireHE. Along X-axis we plot transaction data presented in KB and along Y-axis we plot the timing for the operation in milliseconds. In 4 (a) The performances of encryption in 5ireHE are compared for two modes of implementations corresponding to two key sizes 512 bit and 1024 bit. For 1KB size of the transaction, 5ireHE encryption takes 8070 milliseconds with a 512-bit key and 63062 milliseconds with a 1024-bit key. In Figure 4 (b) shows the decryption performance for two different key sizes using 5ireHE decryption functionality. For 1KB size

of the transaction, 5ireHE decryption takes 17000 milliseconds with 512-bit key and 126209 milliseconds with a 1024-bit key.



(a) 5ireHE Encryption timing **(b) 5ireHE Decryption timing**

Fig. 4: 5ireHE Encryption and Decryption performance graph

The timing for computation is very less compared to the encryption and decryption timing. With 512-bit key, the timing for multiplying 1 KB ciphertexts, is 71 milliseconds and that with 1024-bit key is 180 milliseconds. For the scalar multiplications of 1KB transaction cipher with a constant, 12, is 254 milliseconds and 645 milliseconds with 512-bit key and 1024-bit key respectively.

6.0 PRIVACY PROTECTION WITHIN 5IRECHAIN

The blockchain is composed of multiple blocks and each node in the 5ireChain network stores the same blockchain. The emanate block includes the block head and the block body. Where the Block Body contains information stored as meta- data, and the Block Head holds the input metadata information like: version, timestamp, characteristic value, and difficulty value, etc. The next block of the emanate block is known as the parent block, and proceeding next block also includes the block head and the block body, so each block is in the blockchain filled with a similar set of information. The block data stored in the block is associated with the block data stored in the parent block, which ensures the security of the input information in the block. The steps of this algorithm are as follows.

Step 1: Key Generation: Initialize random primes p and q and meet the condition of: gcd (pq, (p-1)(q-1)) = 1

Module calculation n=pq. =1cm(p1,q1), where 1cm is to seek the least com- mon multiple of p1 and q1 Select the random number g(g Z*n2) and meet = Lg mod n2-1 mod n The greatest common divisor of L(g mod n2) and n. Zn2 represents the set of integers coprime to n2 in Zn2. The encrypted public key of function Lx = x 1/n is n, g, and the private key is (,). Select integer in encryption and decryption process, r(r Zn22), and the plaintext is r(r Zn) and m ¡ n.

Volume: 05, Issue: 02 " March - April 2022"

ISSN 2582-0176

Step 2: (encryptionEnc (m, pk): Let m be the information to be encrypted and m Zn. Compute the ciphertext: c = m mod n. The encryption process is c=E(m) = gm .rn mod n2

Where c is the ciphertext data corresponding to the plaintext m and c Zn2*. Mark the encryption algorithm as c=E(m, r). It can be known that for the same ciphertext m, the value of r, which is randomly selected in the encryption process may be different and so is the corresponding ciphertext data after being encrypted so as to ensure the semantic security of ciphertext data.

Step 3: (proxy re-encryption): Compute the public key and private key (Rsk, Rpk).

The re-encryption ciphertext is generated by the RSA algorithm, and the public key Rpk is sent to the server.

Step 4: (decryption Dec(c, sk)): Ciphertext c Zn m=D(c)=Lc mod n2* (mod n)

After receiving E(di)(i p), decrypt it to get (di)(i p), sign it, get and send

Signq(di)(i p) to ENq. Step 5: (after ENq receives (di)(i p), divide the degree of dispersion into two kinds). Make the set of users corresponding to the kind with more elements as Q and the set of users corresponding to the degree of dispersion as G. As malicious users take up a small proportion, Q mainly includes normal users with the target value increasing after the completion of tasks, while that of G decreases after tasks end. Therefore, it has introduced two parameters and to control the increase and decrease after the target value is updated, and the target value changes according to the following equation: rinew= ri . (1-v)ri . (1-ri) . if i Gif Q where and are both positive and ¡ 1.

7.0 CONCLUSION

In the public blockchain, the intigrity 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 advesarial presence. Encrypting the data using traditional encryption system comes at the cost of loosing the usability of the data in a sence that unless decrypted the owner can not use the data for future computation. Given the huge size of data, it is not always feasible. In this paper we combine the best of homomorphic encryption and blockchain and propose 5ireHE protocol to safeguard the transaction data in 5irechain, which is a public blockchain. This results can be generalized for any public blockchain.

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: Towards privacy in a smart contract world. In International Conference on Financial Cryptography and Data Security, pages 423–443. Springer, 2020.

Volume: 05, Issue: 02 " March - April 2022"

ISSN 2582-0176

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.

Craig Gentry. A fully homomorphic encryption scheme. Stanford University, 2009.

Muneeb Ul Hassan, Mubashir Husain Rehmani, and Jinjun Chen. Privacy preservation 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.

