Tree-ORAP: A Tree-Based Oblivious Random-Access Protocol for Privacy-Protected Blockchain

Youshui Lu, *Member, IEEE*, Bowen Cai, Xiaojun Tang, *Member, IEEE*, Lei Liu, *Member, IEEE*, Jun Du, *Senior Member, IEEE*, Shui Yu, *Fellow, IEEE*, Mohammed Atiquzzaman, and Schahram Dustdar, *Fellow, IEEE*

***Abstract*—Since the introduction of Bitcoin in 2008, blockchain technology has found widespread applications across various domains. While blockchain offers convenience and immense research value, it also raises privacy and security concerns among users and society at large. Notably, numerous studies have demonstrated the vulnerability of blockchain anonymity. Existing solutions based on bloom filters and SGX(Software Guard Extensions) may safeguard users' access patterns but remain susceptible to novel attacks, including protocol-level and side-channel attacks. To address these issues, we propose a Tree-based Oblivious Random Access Protocol (Tree-ORAP) that not only provides access pattern protection in privacy-preserving blockchain systems but also preserves the original blockchain performance. Furthermore, we design a Tree-ORAP State Version Controller to manage state synchronization across nodes in a multi-client blockchain network. We also analyze the system's security and implement a Tree-ORAP prototype, conducting a series of experiments to demonstrate its efficiency and technical feasibility. In summary, our protocol offers enhanced protection for blockchain systems against a wider range of attacks compared to previous methods, all while maintaining superior security performance and equal or better efficiency.**

***Index Terms*—Blockchain, privacy-preserving, ORAM, access pattern protection.**

Manuscript received 2 June 2023; revised 23 October 2023; accepted 24 December 2023. Date of publication 27 December 2023; date of current version 12 June 2024. This work was supported in part by the National Key R&D Program of China under Grant 2022YFF0903402, in part by China Postdoctoral Science Foundation under Grant 2022M712535, and in part by Natural Science Foundation Research Program of Shaanxi Province under Grant 2023-JC-QN-0749. *(Youshui Lu and Bowen Cai contributed equally to this work.) (Corresponding authors: Xiaojun Tang; Lei Liu.)*

Youshui Lu and Xiaojun Tang are with the School of the Instrument Science and Technology, and the School of Electrical Engineering, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: yolu6176@uni.sydney.edu.au; xiaojun_tang@xjtu.edu.cn).

Bowen Cai is with the School of Computer Science and Engineering, University of Minnesota, Twin City, MN 55455 USA (e-mail: cai00254@umn.edu).

Lei Liu is with the Xidian Guangzhou Institute of Technology, Guangzhou 510555, China (e-mail: tianjiaoliulei@163.com).

Jun Du is with the Department of Electronic Engineering, Tsinghua University, Beijing 100084, China (e-mail: jundu@tsinghua.edu.cn).

Shui Yu is with the School of Software, University of Technology Sydney, Sydney, NSW 2007, Australia (e-mail: shui.yu@uts.edu.au).

Mohammed Atiquzzaman is with the School of Computer Science, University of Oklahoma, Norman, OK 73019 USA (e-mail: atiq@ou.edu).

Schahram Dustdar is with the Distributed Systems Group, TU Wien, 1040 Vienna, Austria (e-mail: dustdar@dsg.tuwien.ac.at).

Digital Object Identifier 10.1109/TSC.2023.3347589

I. Introduction

BLOCKCHAIN technology has experienced a significant surge in adoption across various industries since the publication of the Bitcoin paper by Satoshi Nakamoto in 2008 [1]. Unlike traditional centralized systems, blockchains are decentralized, transparent, and immutable, which means that the information stored on a blockchain is publicly accessible and verifiable. However, the public nature of blockchains can compromise the anonymity of participants, as research [2], [3], [4], [5], [6], [7] has shown that the information on public blockchains such as Bitcoin can be de-anonymized by establishing associations among transactions and addresses. Moreover, in line with the General Data Protection Regulation (GDPR) [8], many blockchain-enabled systems in various fields such as finance, industry [9], and healthcare [10] require sensitive information to be kept anonymous and confidential on the blockchain. In the IoT domain, which has widely implemented blockchain systems, the confidentiality and security of product data, client data, and administrator data are paramount. Any data breaches can lead to significant damage for companies. For instance, in the Internet of Vehicles, as described in Lu et al. [11], attackers can steal gas emission data, causing significant financial losses for automobile manufacturers. Therefore, there is a growing interest from both academia and industry in building reliable privacy-protected blockchain systems to address these challenges.

In the past decade, various privacy-protected blockchains have been launched, such as Zerocash [12] and Monero [13]. However, as research has progressed, these privacy-protected blockchains still face new and sophisticated methods of privacy analysis, such as protocol-level attacks [14], [15], [16] and side-channel attacks [17].

To address the security of privacy-protected blockchain models, Gervais et al. [18] proposed a bloom-filter based method to provide privacy-preserving query services. While this method can prevent address leakage with high probability, it fails to protect access patterns. Data holders (full nodes) can still deduce clients' addresses by monitoring transactions and access patterns. Matitic et al. [19] and Niu et al. [20] have leveraged SGX to hide access patterns from full nodes. However, their approaches have two fundamental issues. First, it is not realistic to have every node equipped with SGX. Second, using SGX only targets

the linkage between IP addresses and query tokens, but not the connection between transactions and user addresses. Therefore, we propose achieving unlinkability between user information while hiding access patterns in the context of privacy-protected blockchains.

Our proposed model aims to prevent malicious full nodes from learning the links between published/queried information and actual user identities. To achieve this goal, we draw inspiration from the Path-ORAM model [21] and propose a high-level idea to design an oblivious random access protocol between the client machine (querier) and the server (full node where encrypted data is stored).

The ORAM model has found extensive implementation in cloud-based storage systems. In contrast to blockchain systems, it lacks a fundamental real-time responsiveness requirement and is typically regarded as a single-client service, both of which present considerable challenges for its integration into blockchain platforms. Consequently, we can affirm that incorporating Path-ORAM into existing privacy-protected blockchain systems is confronted by four fundamental challenges:

- The high complexity of the algorithm required for access pattern protection can significantly impact system performance, even leading to a slow system.
- Most existing blockchain systems use their own data structures, such as the state tree in the Ethereum-based ecosystem and Unspent Transaction Output (UTXO) in bitcoin-based and some consortium blockchains [22]. These structures are designed specifically for predefined functions such as search or validation, making it difficult to implement other structures.
- ORAM-based methods are typically designed for a fixed memory size, whereas the number of blocks in traditional blockchain settings increases with the amount of data generated.
- ORAM is a single-client based protocol, while blockchain networks are composed of multiple-client nodes and servers, making it challenging to implement ORAM's rationale in a multi-client blockchain network.

To address these challenges, we propose a tree-based oblivious random-access protocol (Tree-ORAP) specifically designed for privacy-protected blockchain systems. Tree-ORAP introduces a novel blockchain data structure (ORAP-Tree) that enables the reorganization of physical addresses without sacrificing the intrinsic property of the Merkle tree. Moreover, we optimize and reduce the computation overhead to an acceptable range to guarantee the blockchain's performance.

Furthermore, ORAP-Tree is compatible with ever-growing data blocks due to its novel tree structure, which synchronizes newly generated child nodes within the blockchain network. We also present a position map version controller to synchronize the version of the position map of each machine in the network.

Finally, we implemented a prototype of the Tree-ORAP blockchain system and evaluated its performance through a series of experiments. Our results show that Tree-ORAP meets the performance requirements of existing blockchain systems, despite the relatively long time required for these systems to generate new blocks. For instance, Ethereum takes 180 seconds to generate one block, while Zerocash takes 75 seconds, and the time delay for transaction communication can reach up to 240 milliseconds. By comparison, Tree-ORAP only incurs a maximum communication delay of 200 milliseconds. In terms of memory usage, Tree-ORAP is well-matched to the storage capacity of full-node machines and does not sacrifice machine performance. When the number of blocks reaches 13,804,589, which is the exact block number of Ethereum before December 15th, 2021, the storage usage of Tree-ORAP is only 30 GB. In contrast, the total storage usage of Ethereum on a full node machine is at least 2 terabytes, and it can reach up to 20 terabytes under the archive model. In summary, our results demonstrate that the time cost and memory usage of Tree-ORAP are affordable even for large data sizes, and the system's performance is technically feasible.

To address privacy concerns in blockchain systems, we propose a tree-based oblivious random access protocol, Tree-ORAP, specifically designed to preserve user access patterns and protect user privacy from various attacks. Our contributions in this paper are summarized as follows:

- We introduce Tree-ORAP, which can protect access pattern leakage, mitigate external attacks, and prevent graph analysis in privacy-protected blockchain settings, while other methods are only feasible to certain types of exploitation.
- We design a novel data structure named ORAP-Tree, which achieves secure access pattern protection without sacrificing the intrinsic properties of the Merkle tree. This integration of ORAM features into the blockchain data structure leads to a better security performance framework. This should be the first oblivious access protocol applied to privacy-protected blockchain from our knowledge.
- We construct a position map version controller to efficiently synchronize the position map with each node machine in a multi-client blockchain setting, ensuring seamless communication and coordination.
- We implement a prototype of Tree-ORAP and conduct a series of experiments to evaluate its performance. Our results demonstrate that Tree-ORAP's memory usage and time cost are technically feasible for deployment in a real production environment.

In summary, our proposed Tree-ORAP protocol and ORAP-Tree data structure make significant contributions to the privacy protection of blockchain systems and provide a practical solution for ensuring secure and private transactions.

The rest of this article is organized as follows. In Section II, we provide an overview of related work. In Section III, we introduce the necessary theoretical background of blockchain and the preliminary concepts relevant to our work. In Section IV, we define the problem and outline the design goals of our model. In Section V, we present the system architecture, data structure, and protocol details. Specifically, we introduce the novel data structure ORAP-Tree and the position map version controller, which are critical components of the proposed Tree-ORAP protocol. In Section VI, we analyze the security performance of Tree-ORAP in detail, with a focus on how it protects against various attacks. In Section VII, we present the results of our experiments and evaluate the performance of

Tree-ORAP. Finally, in Section VIII, we conclude the article and discuss the implications of our work for the future of blockchain technology and privacy protection.

II. Related Work

As blockchain vulnerabilities continue to increase, privacy concerns have become a pressing issue for both industry and academia [18], [27], [28], [29]. Research in this field can be broadly classified into two categories. The first category focuses on designing new protocols to address the privacy issues of current blockchain systems. This includes protocols that are direct extensions of the Bitcoin network [30], as well as those that use cryptographic primitives such as ring signatures for Monero [13] and zero-knowledge proofs for Zcash [31]. Researches like Bulletproofs [32] reduce transaction size and improve privacy in Bitcoin, and MimbleWimble [33], a privacy-oriented blockchain with a focus on scalability and privacy. Other examples are like Fabric-privatechain [34] and Quorum [35]. The second category involves examining new attack methods or privacy and security vulnerabilities in current blockchain systems [27], [36], [37]. For instance, Tramer et al. [17] proposed a timing side-channel and traffic analysis attack on light nodes in Zcash and Monero. Other types of side-channel attacks on blockchain inclues power consumption analysis [38]. Biryukov et al. [15] analyzed privacy issues for Zcash and proposed an active attack to reproduce users' transactions. Other researches have focused on improving privacy in dApps, such as privacy-preserving smart contracts [34] and privacy-enhanced decentralized exchanges (DEX) [39].

Several previous works have also aimed to protect light client privacy. Gervais et al. [18] employed Bloom filters [40] to enable clients to mask target addresses with probabilistic requests for the total transaction subset. Osuntokun et al. [23] proposed modifications that move the bloom-filter application from full nodes to clients to improve the false positive rate. Rahman et al. [24] proposed a privacy-preserving light client architecture, where the client only requests information for relevant transactions, rather than downloading the entire blockchain, also used Bloom filter. Matetic et al. proposed employing SGX [19], [20] on full nodes to serve privacy-preserving requests from light clients, and Niu et al. [20] extended this research on the efficiency of this framework. Moreover, Liang et al. [25] proposed a homomorphic encryption method to enable light clients to perform searches on the blockchain without revealing their search queries. The proposed system protects the privacy of user queries while allowing the client to retrieve relevant information from the blockchain. Kokoris et al. [26] proposed a novel method to protect the privacy of light clients by using a secure computation framework. The proposed method uses a secret sharing technique to prevent information leakage and enables clients to verify transactions without downloading the entire blockchain.

The previous works mentioned above have made significant contributions to protecting the privacy of light clients in blockchain systems. However, these methods still suffer from several weaknesses. For bloom-filter-based methods, user behavior significantly affects privacy. If a user does not request filter headers from distinct full nodes, the target address can be easily deduced. For SGX-based methods, they protect access patterns, but they are susceptible to side-channel attacks and protocol-level attacks. The multi-party computation and homomorphic encryption-based methods are computationally expensive and may require significant resources to implement, which may limit their practicality, we give a comparison in Table I. In contrast, Our proposed protocol addresses these vulnerabilities by providing privacy protection, particularly in preventing access pattern leakage, and defending against novel attacks such as side-channel attacks, graph analysis, and protocol-level attacks. Furthermore, our experimental results indicate that the overhead of the system meets the technical requirements of the blockchain system, making our proposed protocol feasible and practical.

III. Background and Preliminaries

In this section, we will first discuss the current status of blockchain technology applications in the IoT domain. We will then introduce several important concepts related to our work, including privacy-protected blockchain and Oblivious RAM (ORAM).

A. Background of IoT Blockchain

The integration of blockchain technology into the IoT domain has enabled the development of new features and solutions for various scenarios. For instance, Kang et al. [41] and Javaid et al. [42] propose IoT data management systems based on blockchain to authenticate user identity and improve security performance. In the domain of Internet of Vehicles, blockchain has been applied to enable secure data sharing. For example, Shi et al. [43] and Chen et al. [44] propose blockchain frameworks for effective sharing of vehicle data and introduce negotiation mechanisms for data sourcing. Additionally, blockchain has also shown great potential in motivating the development of security, decentralized solutions, and peer-to-peer structures in other IoT domains, such as industrial IoT networks [9] and medical IoT [10].

However, few works have addressed the privacy requirements within IoT blockchain networks. To ensure that clients' personal information is properly guarded in production environments, a naive blockchain system is not enough. Our protocol is based on a privacy-preserving blockchain network that provides a more confidential communication solution, thereby efficiently improving the security of IoT blockchain networks.

B. Privacy-Protected Blockchain

Privacy-protected blockchains, such as anonymous digital currencies and privacy confidential consortium blockchains, employ techniques such as zero-knowledge proofs and parallel hidden ledgers to conceal transactions and user addresses from the public. Zerocash and Monero are typical examples of privacy-protected blockchains, but due to their feasibility and popularity, we will only focus on Zerocash in this section.

Zerocash was first proposed by Miers et al. [31] and is an extension of Bitcoin that provides strong anonymity to users.

TABLE I
COMPARISON OF RELATIVE WORKS

Relative Works	Methods Type	Goals	Limitations
Gervais *et al.* [18], Osuntokun *et al.* [23]	Bloom-filter-based	Probabilistic request	User behavior sensitive
Rahman *et al.* [24], Matetic *et al.* [19], [20]	SGX-based	Hard-ware protection	Side-channel/protocol-level attacks exploitable
Niu *et al.* [20], Liang *et al.* [25]	Multi-party computation	Secret sharing technique	Computationally expensive
Kokoris *et al.* [26]	Homomorphic encryption	Encrypted request	Computationally expensive

Ben-Sasson et al. [12] proposed improvements to Zerocash from a decentralization perspective. Zerocash does not use digital signatures to validate coins, nor does it require a central party to prevent double-spending. Instead, it employs zero-knowledge proofs to verify whether coins belong to a public list of valid coins, effectively hiding public information such as identities and transaction details on the list. In this way, zero-knowledge proofs enable anonymity, and we will provide some details on the technique and its anonymity achievements in the following sections.

1) Zero-Knowledge Proof: Zero-knowledge proof is the underlying technology of Zerocash, providing users with strong anonymity. To better understand its functioning, we provide a simple example in this section. An individual, Alice, can participate in the Zerocash system in two ways - minting and redeeming coins. To mint a coin, Alice generates a serial number S and commits to it with a secure digital commitment scheme. She then encapsulates it into the coin C, which can only be opened with a random number r known only to her. She publishes the coin to the network, along with \$1 physical currency, which all users accept only if it is consistent with the sum of currency. To redeem the coin C, Alice pulls the set of valid commitments $(C_1, \ldots, C_N)$ from the network, published by other users. She provides a non-interactive zero-knowledge proof π, which states: (1) she knows $C \in (C_1, \ldots, C_N)$, and (2) she knows the random value r to open C. Alice then publishes a "spend" transaction with (S, π) for other users to examine the proof π and check for double spending through S. If all conditions are satisfied, the transaction is validated, otherwise, it is rejected.

2) Anonymity Achievement: The protocol described above achieves several important goals. First, Alice does not need to expose the source of her assets or her address. This is because the blockchain can easily prevent double spending with the use of zero-knowledge proofs without tracking the transaction source. In order to track her transaction, other users would need to know either r or which C Alice provided proof for, but neither of these are revealed by the proof. As a result, Zerocash provides an anonymous system with encrypted transactions and addresses based on Bitcoin.

C. ORAM Model

The Oblivious RAM (ORAM) model, initially proposed by Goldreich and Ostrovsky [45], is designed to safeguard information from adversaries who can observe the memory access pattern. With the development of cloud storage and computation, recent research has focused on making ORAM schemes more practical and reliable. Two notable schemes are Path-ORAM [21] and PRO-ORAM [46].

Path-ORAM is an efficient ORAM protocol that protects the access pattern of clients from external servers. It consumes less bandwidth and has low latency compared to other methods. The scheme is easy to understand: the server organizes the data as a tree with N buckets, where each bucket can contain Z blocks. The client maintains a *position map*, which maps the block identifier to a node in the server's tree. It also defines a variable *path*, which represents the path from a single leaf node to the root. When the client retrieves data, it queries all the nodes on the path where the data is stored. If there is new data to be added, the client re-encrypts and reorganizes all the data along the path and writes it back to the server. The *position map* is updated during this process, and the server waits for the next query or write.

Path-ORAM is highly practical and has a small client storage requirement. For blocks of size $B = \Omega(\log^2 N)$ bits, its bandwidth overhead is $O(\log N)$, and it consumes only $O(\log N)w(1)$ of client-side storage. Here, $w(1)$ denotes a constant value for any function $h(n)$. For such sizes, Path-ORAM performs better than other schemes in terms of asymptotic complexity. Additionally, it has excellent security and is proven to fail with mostly negligible probability.

IV. PROBLEM DEFINITION

In this section, we present the threat model, security definitions and the design goals of the proposed protocol.

A. Threat Model

In this section, we define the threat model for the typical communication process between client nodes and full nodes in a blockchain network. The communication process involves two types of parties: the client node, also known as the light node, which has limited storage such as personal computers or mobile phones, and the full node, which is the server that stores the entire data of the blockchain network.

In this model, we consider two types of nodes:

1) Honest-but-Curious Full Node: The full nodes can observe all the queries and transactions that are encrypted. The full nodes can provide the zero-knowledge proof for transactions but cannot decrypt them. However, the server can analyze

clients' access pattern and storage content to examine the linkage between the transactions and the addresses.

2) Independent Client Node: Client nodes are independent and anonymous to other nodes in the blockchain network. They share their account information only when they publish transactions and send them to the full node. Clients want to protect their privacy and avoid leaking any private information to the full node.

To provide a more formal expression of the threat model, we can define the following entities and their relationships:

Let $\mathcal{N}$ be the set of all nodes in the blockchain network, and let $\mathcal{N}_l \subseteq \mathcal{N}$ be the set of all light nodes, also known as client nodes. Let $\mathcal{N}_f \subseteq \mathcal{N}$ be the set of all full nodes, also known as server nodes.

For any client node $n_l \in \mathcal{N}_l$, there exists a corresponding account A_l that is associated with n_l. The account A_l has a public key PK_l and a private key SK_l, which are used to sign transactions and provide authentication.

For any full node $n_f \in \mathcal{N}_f$, there exists a corresponding database DB_f that stores the entire blockchain data. The database DB_f can observe all the queries and transactions that are encrypted. The database can provide zero-knowledge proofs for transactions but cannot decrypt them.

In this model, we assume that the full nodes are honest-but-curious, meaning they will not tamper with the data but may attempt to gain additional information about the clients by analyzing their access patterns and storage content. The concrete model for basic access pattern inference can be described as follows:

Definition 1 (Access Pattern Inference): The request sequence, denoted as $\vec{y}$, is initiated by n_l. The responsive storage address for input $\vec{y}$ is denoted as $O(\vec{y})$, situated in DB_f. The objective of the attacker is to ascertain a specific responsive address

$$o' = O(\vec{y'}).$$

This allows attacker n_f to deduce that the private data o' is associated with input $\vec{y'}$.

Subsequently, we elaborate on this definition to provide the security framework outlined in Section IV-B.

Our goal is to design a communication protocol that protects the privacy of the client nodes and their transactions while preventing the full nodes from learning any additional information beyond what is necessary to verify the integrity of the transactions. This requires a robust defense against access security, external security, and unlinkability attacks, as well as ensuring the confidentiality and authenticity of the data being transmitted between the nodes.

B. Access Pattern Security

We use the definition of access pattern security from Stefanov et al. [47] to evaluate the security of the access protocol.

Definition 2 (Request sequence definition):

Let

$$\vec{y} := ((\mathsf{op}_M, x_M, \mathsf{data}_M), \ldots, (\mathsf{op}_1, x_1, \mathsf{data}_1)),$$

denote a sequence of data request with length M, where op_i denotes READ from location x_i or WRITE data_i to x_i, and data_i denotes the data to be written.

Definition 3 (Secure access definition): Let variable $A(\vec{y})$ denote the request sequence generated by protocol to the server given the sequence of data $\vec{y}$. The construction is secure if it meets tow property:

1) for any two data request sequences $\vec{y_1}$, $\vec{y_2}$ of the same length, their access $A(\vec{y_1})$, $A(\vec{y_2})$ are computationally indistinguishable by anyone but the client,
2) the output of construction is consistent with input $\vec{y}$ with probability $\geq 1 - negl(|\vec{y}|)$.

In simpler terms, a secure access construction ensures that the server does not know: (1) which data is being accessed; (2) whether the same data is being accessed; (3) whether the data is being read or written; (4) the access pattern (e.g., sequential, random); and (5) the time of the last access.

C. Design Requirements

The objective of this paper is to propose a protocol that comprehensively protects the privacy of blockchain users. Specifically, our model should satisfy the following requirements:

- *Access security:* The server providing the full node service cannot analyze the access pattern to establish a linkage between transactions and addresses corresponding to *Definition 3*. This means that data access patterns from two sequences of query or publish operations cannot be distinguished.
- *External security:* We consider an attacker who can passively monitor the traffic between the victim and server or participate in the P2P network. We require that the attacker cannot infer the linkage between any transactions and addresses of users using external attack methods such as side-channel attacks.
- *Unlinkability:* The attacker cannot infer users' private information by analyzing on-chain information using methods such as graph analysis or other protocol-level attacks.

Our protocol should satisfy these requirements to ensure that users' private information remains secure while using the blockchain network.

V. System Design

In this section, we introduce the routing to outline the main part of our article, namely the framework of the *Tree-ORAP*. The primary objective of the Tree-ORAP is to safeguard the access pattern within the blockchain network, effectively preventing any unauthorized surveillance by full nodes on the memory addresses accessed by clients.

A. Overview

As mentioned earlier, multiple accesses from a single client can provide the full node with necessary information to associate each client with memory addresses, which could lead to severe privacy breaches. Our system can be run on a privacy-protected

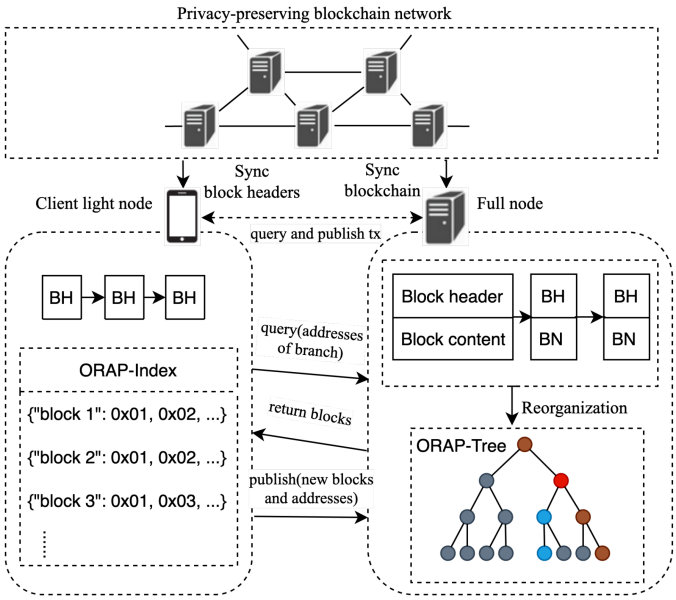

Fig. 1. Framework of Tree-ORAP.

blockchain network such as Zcash or Monero, which implement zero-knowledge proof schemes to encrypt transaction amount and addresses on the blockchain. In other situations, an encrypted consortium blockchain can provide a relatively safe transaction environment, making it compatible with the goals we have proposed in Section IV.

The general architecture of our system is inspired by oblivious RAM, which preserves the original input and output of an algorithm but makes it independent of the distribution of memory access patterns. We have developed the oblivious random access protocol and related data structure to adapt to the blockchain network environment, referring to general ORAM algorithms and PathORAM trees. All the information on the chain is reorganized into the form of a tree, which aligns with the original state of the Merkle tree on the chain. When the full nodes receive requests from clients, they return the full path where the required data is located on the tree. Clients can then send new transactions to the full node, and update their position map and stash stored locally to synchronize with the full node. The full nodes verify the newly received transactions with zero-knowledge proofs and package them into the next block. The newly published transactions will be reorganized into the ORAP-Tree we mentioned above to be ready for the next request.

B. Architecture

The architecture of Tree-ORAP is depicted in Fig. 1. The system comprises two main entities: the client light node and the full node.

- *Full Node:* Each full node periodically synchronizes the entire blockchain data with the network, and it is responsible for generating transactions, publishing blocks, and responding to requests. Unlike ordinary full nodes, the full node in our proposed system maintains a novel tree data structure called ORAP-Tree, which is used to reorganize the original blockchain data. We will discuss ORAP-Tree in detail in Section V-D.
- *Light Node (Client):* The client node only stores the block header. Although it cannot publish blocks on its own, it can validate transactions, query the blockchain, and send transactions to the full node. Additionally, the light node in our proposed system has an extra functionality, which is to maintain an ORAP-Index that indexes the block identifiers to the addresses of branch on the tree, providing the access pattern protection property. We will discuss the algorithm in Section V-D.
- *Blockchain Network:* The blockchain network is a peer-to-peer network that uses standard protocols to communicate. The full node plays the primary role in receiving and distributing information.

C. Events and Workflow

The Tree-ORAP protocol serves as a communication protocol between client nodes and full nodes. Both client nodes and full nodes need to maintain compatible data structures, namely ORAP-Tree and ORAP-Index. The general workflow of the protocol can be summarized as follows:

- *Initialization:* The client nodes and full nodes initialize their data structures, including ORAP-Tree and ORAP-Index, before the protocol can be used.
- *Query:* The client node requests transactions from the full node by specifying a set of addresses (i.e., the branch of the node in the tree that we will discuss in detail in Section V). The full node returns all data on the satisfied addresses.
- *Publish Transactions:* When a client node publishes new transactions, it submits the new transactions with the entire branch to the full node. Upon validation, the transaction is packaged into the next block and published.
- *Validation & Update:* The full node updates ORAP-Tree, synchronizes the changes of the ORAP-Index with the client node, and waits for the next request.

D. Tree-ORAP Protocol

In this section, we provide a formal introduction to our proposed protocol, notations can be found in Table II. In the blockchain network, the light node stores only a small amount of data in the local stash, while the full node stores the entire blockchain data. The data structure used in our protocol is similar to the ordinary Merkle tree, but it possesses the capability to exchange the position of its nodes, enabling the protocol to conceal the access pattern from the full node.

1) Full Node Data Storage: The data on the full node is stored as a tree structure called ORAP-Tree. While it doesn't necessarily have to be a binary tree, we chose to use a binary tree in our protocol for the sake of simplicity.

ORAP-Tree: The full node uses a binary tree data structure, called ORAP-Tree, to store all the data on the blockchain. The height of the tree is denoted by L, and the number of leaves is 2^L. The relationship between the height and the number of nodes is given by $L = \lceil \log_2(N) \rceil$, where N is the total number of data blocks in the blockchain. The levels of the tree are labeled from

TABLE II
NOTATIONS

Symbol	Definition
N	Total number of data blocks stored on chain
L	Height of binary tree
B	Block size(in bits)
S	Client's local stash
index	Index of blocks to the memory address on full node
$\mathcal{P}$	Branch of ORAP-Tree, from root to leaf node
$p := \mathcal{P}(x)$	Branch from leaf node to the root of tree where the node x is located
$x :=$ index[a]	Block a is associated with leaf node x in tree, i.e., block a resides somewhere along branch$[x]$
$p :=$ branch[a]	Branch from leaf node to the root of tree where the block a is located, here is the mapping from node to path distinguishing from $\mathcal{P}(\cdot)$

0 to L, with level 0 representing the root and level L representing the leaves. The binary tree can be easily represented as an array stored on a disk. To each data block, we add an address pointer that corresponds to the sequence of the array, based on the original data. As the blockchain data grows, the ORAP-Tree can be extended by simply adding new child nodes to the array.

Branch of the Tree: A branch in the ORAP-Tree refers to a path from the root to a leaf node in the binary tree data structure. Each leaf node in the tree is assigned a unique index $x \in \{0, 1, \ldots, 2^L - 1\}$, where L is the height of the tree. We denote the branch corresponding to node x as $\mathcal{P}(x)$. Since each leaf node corresponds to a block on the blockchain, the branches in the ORAP-Tree provide a mapping between block identifiers and their corresponding locations in the tree.

2) Clients' Local Storage: For the client node, we have developed the ORAP-Index, which serves as an index for the ORAP-Tree on the full node and allows the client to locate block addresses.

ORAP-Index: The ORAP-Index maps each block identifier to a corresponding leaf node in the ORAP-Tree, denoted as $x :=$ index[a]. This mapping enables the client to identify the path in the tree where the block is located. The ORAP-Index is stored locally by the client and is updated whenever the blocks are rearranged on the full node.

Stash: Additionally, the client node maintains a local stash, denoted as S, which stores a small number of blocks that have been recently accessed. When the client accesses data, it first searches the stash before accessing the full node.

3) Storage Initialization: Algorithm 1 shows the initialization process for the client node. The inputs include the number of blocks, the depth of the tree, and all block identifiers. Using uniform random distribution, the location of each block is allocated, which generates the index and the mapping from block identifiers to branches in ORAP-Tree. It's important to note the difference between branch and index variables. index maps a block identifier to a single node's location in ORAP-Tree, while branch maps a block identifier to a series of nodes that form a single branch in ORAP-Tree. Initially, the client node does not store any block data, only the ORAP-Index that includes index and branch.

Algorithm 1: Client Initialization.

Input: N, L, blockId
Output: index, branch
Initialization:
NodeIds $= \{0 \ldots 2^L - 1\}$, fullNodeIds $= \emptyset$
LOOP Process 1:
for $i = 0$ to N **do**
  tmp $\leftarrow$ UniformRandom($\{0 \ldots 2^L - 1\} -$ fullNodeIds)
  index[i] $\leftarrow$ tmp
  fullNodeIds $\leftarrow$ fullNodeIds $\cup$ tmp
end for
LOOP Process 2:
for $i = 0$ to N **do**
  branch[i] $\leftarrow \mathcal{P}$(index[i])
end for

Algorithm 2: Server Initialization.

Input: N, index, Blocks
LOOP Process:
for $i = 0$ to N **do**
  Address[index[i]] $\leftarrow$ Blocks[i]
end for

Algorithm 3: ORAP Access.

Input: L, index, branch, block_id, block_new
Output: index, branch
Initialization:
positions $=$ branch[block_id], block_content $= []$, randomId $=$ UniformRandom$\{0 \ldots 2^L - 1\}$
QUERY Process:
block_content $\leftarrow$ READ(positions)
WRITE Process:
for i $= 0$ to L **do**
if block_content[i] $==$ None or i $==$ randomId **then**
  block_content[i] $\leftarrow$ block_new
    positions[i] $\leftarrow$ block_new.id
    index[block_new.id] $\leftarrow$ positions[i]
    branch[block_new.id] $\leftarrow$ positions
  end if
  WRITE(block_content, positions)
end for

Algorithm 2 demonstrates the initialization process for the full node. The inputs include the number of blocks, the block index, and the blocks that are already stored on the full node. The initialization process is straightforward. The block contents are allocated to the addresses on the full node's disk, according to the index sent by the client node. During the initialization phase, the full node reorganizes all the block contents. This operation only occurs once in the node's lifetime.

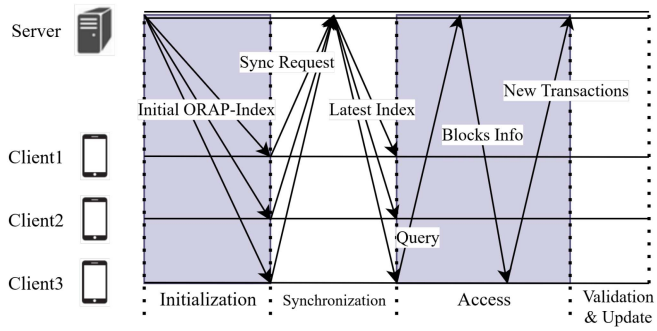

Fig. 2. Communication process of Tree-ORAP protocol.

4) Query and Publish: Querying and publishing transactions are two essential operations in a blockchain system. In our proposed system, we introduce the Access protocol to implement these operations. Specifically, to query a block, a client can call the Access((block_id, None)) function, while to publish a new block, the client can call the Access(block_id, block_new) function. The detailed Access protocol is presented in Algorithm 3, which can be summarized as follows:

- *Query Process (lines 1–2):* The client first reads the path positions of the block block_id from the ORAP-Index.
- *Update ORAP-Index (lines 4–8):* Before writing the new block data to the server, the client updates its ORAP-Index with the address of the new block.
- *Write Process (line 9):* After sending the requested data back to the client, the full node server updates the content of ORAP-Tree by writing the new block on the address specified by the client.

By using the Access protocol, our system can securely query and publish blocks while preserving the access pattern privacy of clients.

E. Position Map Version Controller

As previously mentioned, our proposed protocol is based on the classical ORAM algorithm to provide privacy-preserving access to the honest-but-curious data repository. However, traditional ORAM is only compatible with a single user, so a novel data structure is required for a multi-client blockchain system. In our construction, we use the ORAP-Index, including branch and index, to correspond the block and the real location or branch on the server. These structures are stored at the client-side to reduce communication overheads. However, in a multi-client setting, the server must ensure the consistency of its data state across all clients to prevent users from accessing incorrect data with an outdated ORAP-Index.

To address this issue, we introduce the Position Map Version Controller as shown in Fig. 2 to implement the following modifications:

- The server processes the initialization of ORAP-Index (index and branch).
- Before executing ORAP-Access, the client node pulls the latest version of ORAP-Index.
- After executing ORAP-Access, the client node pushes the latest ORAP-Index version to the server. The server only validates the update if the modifications of the index are consistent with the memory address.

The synchronization process works as a version controller, and communication only occurs when there is an update of content. Therefore, the communication overhead is relatively small, and the size of the ORAP-Index is also very small. Lastly, the procedure of the Tree-ORAP can be summarized as follows:

- *Initialization of the server and the client:* The data structure of the Tree-ORAP is built based on the original data stored on the server, and the ORAP-Index will synchronize with the client nodes.
- Before the client sends the access request to the server, it pulls the latest ORAP-Index and finds the node location of the requested block.
- *Formal access:* The client sends the formal access request to the server, querying the nodes, writing back the new block, and updating the local ORAP-Index.
- After the access process, the server validates the transactions and updates the ORAP-Index.

VI. Security Analysis

In this section, we perform a comprehensive security analysis of the Tree-ORAP, based on the three design requirements outlined in Section IV.

A. Access Secure

For the internal security, or the access pattern security, whenever the client queries from or writes data on the server, the data is encrypted, which means the server can not decrypt without the private keys. At the same time, the protocol obscures the address of the target data, and mixes the address of the new data into a certain branch. Therefore, it is difficult for an adversary to infer the access pattern of the querying clients.

To prove this, let $\vec{y}$ and $A(\vec{y})$ denote the request sequence and generated sequence of size M, respectively. As mentioned in *Definitions 2* and *3*, by definition of Tree-ORAP, the generated request sequence $A(\vec{y})$ is

$$\vec{p} = (\mathsf{branch}[\mathsf{a}_M], \ldots, \mathsf{branch}[\mathsf{a}_1]), \tag{1}$$

where $\mathsf{branch}_i[\mathsf{a}_i] (1 \leq i \leq M)$ is the ith request generated by branch

$$\mathsf{branch}[\mathsf{a}_i] = (x_0, \ldots, x_k, \ldots, x_L), \tag{2}$$

where L is the depth of the tree, x_k is the data in the tree associated with a_i, and x_L equals to $\mathsf{index}[a_i]$.

Each block identifier is mapped to a branch on the tree. The branches are encrypted with randomized encryption and it is computationally indistinguishable from randomized bit strings.

According to Algorithm 3, we know that once $\mathsf{branch}[\mathsf{a}_i]$ is exposed to the server, the block is remapped to another random node, therefore, $\mathsf{branch}[\mathsf{a}_i]$ is independent from $\mathsf{branch}[\mathsf{a}_j]$ when $j \leq i$ and $\mathsf{a}_j = \mathsf{a}_i$. In addition, when $j \leq i$ and $\mathsf{a}_j \neq \mathsf{a}_i$, since the mapping of different block identifier is not linked to the other's in our model, $\mathsf{branch}[\mathsf{a}_i]$ and $\mathsf{branch}[\mathsf{a}_j]$ are independent from

each other. We have demonstrated $\mathsf{branch}[\mathsf{a}_i]$ and $\mathsf{branch}[\mathsf{a}_j]$ are independent from each other for $j \leq i$, then we could apply Bayes rule

$$\mathbb{P}(\vec{p}) = \prod_{j=1}^{M} \mathbb{P}(\mathsf{branch}[\mathsf{a}_i]) = \left(\frac{1}{2^L}\right)^M. \tag{3}$$

It proves that $A(\vec{y})$ is computationally indistinguishable from a random sequence of bit strings.

B. External Secure

The Tree-ORAP provides defense against external eavesdropping and inference attacks. However, attackers can exploit timing side-channels or leaked communication patterns or content to try and identify the P2P node used by the secret payee of transactions, potentially linking the user's account to its identity. In this conservative scenario, the attacker can capture the entire data package during each communication cycle. Although the attacker cannot derive the correct query sequence, they can still acquire all the transmitted data, which can be represented as a set S

$$S = \{\{x_0, \ldots, x_{L_M}\} \cup \cdots \cup \{x_0, \ldots, x_{L_1}\}\}. \tag{4}$$

For simplicity, we denote the set of blocks x as

$$s(\mathsf{a}_i) := \{x_0, \ldots, x_{L_i}\}. \tag{5}$$

However, to establish the linkage between transaction and the monitored user, the attacker must identify the exact data required by the user for each query. The probability of successfully identifying x_i (the information required by the client) given a set of blocks $s(\mathsf{a}_i)$ is denoted as $\mathbb{P}(x_i \mid s(\mathsf{a}i))$, and can be calculated as follows:

$$\mathbb{P}(S) = \prod_{j=1}^{M} \mathbb{P}(x_i \mid s(\mathsf{a}_i)) = \left(\frac{1}{L}\right)^M. \tag{6}$$

This means that the probability of identifying the information required by the client is extremely low.

To prevent timing side-channel and inference attacks, the time cost for responding to queries or transactions is kept stable, and the generation process of zero-knowledge proofs is guarded by the oblivious access algorithm. Therefore, it is impossible for adversaries to exploit timing side-channels, communication patterns, or proof generation. In summary, our proposed model is inherently robust against the aforementioned attacks.

C. Unlinkability

We can define unlinkability using the symbols from the threat model as follows:

Let $\mathcal{N}$ be the set of all nodes in the blockchain network, and let $\mathcal{N}_l \subseteq \mathcal{N}$ be the set of all client nodes. Let $\mathcal{A}$ be the set of all user account information, including their public keys and transaction history.

The goal of unlinkability is to ensure that for any two nodes $n_1, n_2 \in \mathcal{N}_l$ and any transaction set $T_1 \subseteq \mathcal{T}$ and $T_2 \subseteq \mathcal{T}$ belonging to n_1 and n_2, respectively, it is difficult or impossible to link T_1 and T_2 to the same user account in $\mathcal{A}$.

TABLE III
SECURITY PROPERTY COMPARISON BETWEEN OUR MODEL AND EXISTING MODELS

Models	Access secure	External secure	Unlinkability
Empirical analysis protection [14]–[16]	×	✓	×
BITE [19], SGX method [20]	✓	×	×
Bloom-filter [18], [48]	×	✓	✓
Tree-ORAP	✓	✓	✓

Formally, we can define unlinkability as follows:

Definition 4 (Unlinkability): Let $f : \mathcal{N}_l \to \mathcal{A}$ be a function that maps each client node to a corresponding user account in $\mathcal{A}$. Unlinkability holds if and only if:$\forall n_1, n_2 \in \mathcal{N}_l$ and $\forall T_1 \subseteq \mathcal{T}, \forall T_2 \subseteq \mathcal{T}$ belonging to n_1 and n_2, respectively, it is computationally difficult to find a function

$$g : T_1 \cup T_2 \to \mathcal{A}, \tag{7}$$

that maps each transaction in $T_1 \cup T_2$ to a corresponding user account in $\mathcal{A}$ such that

$$g(T_1) = f(n_1) \wedge g(T_2) = f(n_2). \tag{8}$$

To ensure unlinkability, it is important that adversaries are unable to link transactions to the information on-chain. Attack methods that break unlinkability include transaction graph analysis and protocol-level attacks. In privacy-protected blockchain settings, the transaction content is only available to the participants of the transaction. With Tree-ORAP, transactions are encrypted, allowing users to hide their real identity using anonymous IP and dummy addresses. As adversaries are unable to forge cryptographic primitives, it is difficult for them to learn the encrypted information and link transactions to real identities.

In comparison to other models, Table III shows that Tree-ORAP possesses all security properties. Empirical analysis methods only focus on specific types of privacy-protected blockchains and do not address access pattern inference or graph analysis attacks. SGX-based models, such as those proposed by Matetic et al. [19] and Niu et al. [20], aim to protect privacy during communication but do not address external attacks [17] or transaction sourcing, which are vital vulnerabilities. Bloom-filter-based methods, such as those proposed by Kanemura et al. [48] and Gervais et al. [18], are relatively practical, but still lack protection against access pattern attacks.

VII. PERFORMANCE EVALUATION

In this section, we present a series of experiments to evaluate the performance of the Tree-ORAP prototype in a blockchain network.

Our construction comprises two major parties: the client node group and the full node group. The full node server is set up with an Intel(R) Core(TM) i7-10700 CPU @ 2.90 GHz and 16 GB RAM, and we use Infura and levelDB for the production network simulation. The client machine is equipped with an Intel(R) Core(TM) i7-4770HQ CPU @ 2.20 GHz and 16 GB RAM.

To evaluate the performance of the prototype system, we implement a simplified version of Tree-ORAP in a private Ethereum network. We have established our blockchain system's deployment using the Zebra codebase and APIs as the foundation. Building upon this foundation, we have meticulously re-engineered the Path-ORAM Protocol originally designed in ObliVM, tailoring it to seamlessly integrate with blockchain settlements. To achieve this, we leveraged the power of the libbitcoin library, enabling smooth interaction with essential blockchain data structures. Since the system integrates the Tree-ORAP protocol into the blockchain system, the performance of Tree-ORAP can represent the overall performance by adding the appropriate scalars. For example, to evaluate memory consumption, we test the memory used by Tree-ORAP and then add the general memory consumption of a specific full node process to obtain the overall performance result. In the experiment, we focus on the performance of the protocol part rather than the blockchain network, as the performance metrics for the latter can be obtained from official documents or the blockchain community.

Furthermore, the rationale behind selecting memory consumption and time delay as primary metrics lies in their direct impact on the perceived performance by both end users and developers. End users on client nodes are primarily concerned with the duration required to retrieve essential data from the server or a full node. From the perspective of full nodes or servers, the key inquiry pertains to whether their computing resources can accommodate the necessary computations. This consideration hinges solely on CPU capacity and memory utilization, with CPU capacity being reflected in time delay. Consequently, we conducted a comprehensive assessment of memory utilization. The omission of bandwidth overhead testing stems from the fact that all experiments were conducted within typical wide area network (WAN) configurations. Consequently, we posit that the time delay metric suffices to elucidate the genuine daily usage scenario.

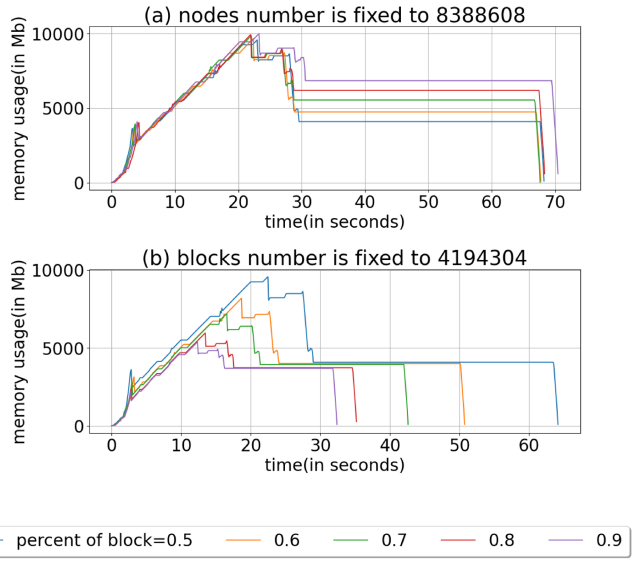

Fig. 3. Memory usage of once query and transaction publishing.

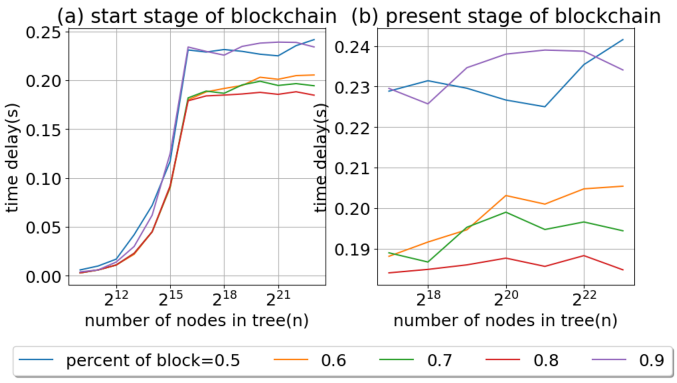

Fig. 4. Time consumption of once query and transaction publishing.

A. Memory Usage

Here we present the results of our experiments on the memory usage of the Tree-ORAP prototype in a private Infura Ethereum network. Since memory usage is a significant performance indicator for any system, we use the ratio of the number of blocks to the number of tree nodes as a measure of the memory overhead.

We begin by comparing the memory consumption over time with fixed numbers of blocks and nodes, respectively. Fig. 3(a) shows that the memory usage reaches a peak of approximately 10 GB as the access operations progress, after which it decreases to different levels depending on the block-to-node ratio. When the number of nodes is fixed, a lower block-to-node ratio results in lower memory usage, as seen in the decreasing trend in Fig. 3(a). Conversely, when the number of blocks is fixed, a lower block-to-node ratio leads to higher memory usage due to the need for more dummy nodes to maintain the required security, as shown in Fig. 3(b).

It is important to find a balance between memory overhead and security performance. Our experiments show that the memory overhead of Tree-ORAP is practical and can be implemented in the general configuration of a full node machine. However, as the security performance improves, the memory usage also increases. Therefore, the trade-off between these two factors should be carefully considered when deploying the system.

B. Time Cost

Then we focus on the time cost analysis. As shown in Fig. 4, the time cost increases with the number of blocks. Fig. 4(a) shows the initial stage after the blockchain network is deployed, where the number of blocks starts from 1024. The time cost grows exponentially over the exponential axis at this stage but remains under 200 ms until 2^{16}. However, to prevent the time delay from growing without limit, we fix the depth of the ORAP-Tree L to 17 when the number of blocks exceeds 2^{16}, and distribute the blocks into different trees. Certainly, despite the ongoing increase in block numbers, the ORAP-Tree consistently maintains a depth of 17. Our modification entails assigning an index exclusively to the roots of the trees. This strategic decision reflects a balance between fortifying security and optimizing efficiency within the context of our experiment. This rationale explains the notable inflection point observed at 2^{16} nodes. In

the second stage, shown in Fig. 4(b), the growth of time cost decelerates. Until 2^{21}, which is close to the actual quantity of blocks on Zcash or Monero networks, the time cost reaches its maximum at about 240 ms. Additionally, it indicates that the lower proportion of blocks in the tree, the higher the time cost required. This is because a lower proportion requires more nodes to maintain.

Compared to the time cost of block confirmation on Ethereum (about 180 s), Zcash (75 s), and Monero (120 s), and the node latency of Ethereum (between 1-300 ms), the time cost of Tree-ORAP to collect transactions into blocks is at the same order of magnitude. Therefore, it can be considered practical in real-life settings.

VIII. Conclusion

Recent research has highlighted the significant threat to blockchain privacy. Existing privacy-preserving systems lack safe communication protocols capable of defending against various types of attacks. To address this issue, we propose Tree-ORAP, a pluggable communication protocol for privacy-protected blockchains. By leveraging tree structures and oblivious access patterns, our protocol provides robust defense against access security, external security, and unlinkability attacks.

The framework of Tree-ORAP is described, along with the interaction between clients and full nodes. Our protocol has undergone a detailed security performance analysis, which demonstrates its ability to defend against eavesdropping and access pattern inference attacks. Additionally, it is internally robust to transaction source analysis.

Although there is room for improvement in areas such as memory usage of data structures and efficiency in reorganizing data, Tree-ORAP has the potential to significantly contribute to users' anonymity and blockchain's credit. Our experimental results indicate that the overhead of the system meets the technical requirements of the blockchain system, making our proposed protocol both feasible and practical.

In summary, the proposed Tree-ORAP protocol offers a promising solution to the blockchain privacy problem. By providing a secure communication protocol that protects against various attacks, it will contribute to enhancing the anonymity of users and improving the overall security of blockchain systems. With further refinement, Tree-ORAP has the potential to become a critical component in privacy-preserving blockchains.

References

[1] Bitcoin: A peer-to-peer electronic cash system, 2008. [Online]. Available: https://bitcoin.org/en/bitcoin-paper

[2] T. Sander and A. Ta-Shma, "Auditable, anonymous electronic cash," *Lecture Notes Comput. Sci. (including subseries Lecture Notes Artif. Intell. Lecture Notes Bioinf.)*, vol. 1666, pp. 555–572, 1999.

[3] J. Camenisch, S. Hohenberger, and A. Lysyanskaya, "Compact e-cash," *Lecture Notes Comput. Sci.*, vol. 3494, pp. 302–321, 2005.

[4] Y. Lu, Y. Qi, S. Qi, Y. Li, H. Song, and Y. Liu, "Say no to price discrimination: Decentralized and automated incentives for price auditing in ride-hailing services," *IEEE Trans. Mobile Comput.*, vol. 21, no. 2, pp. 663–680, Feb. 2022.

[5] Y. Lu et al., "Accelerating at the edge: A storage-elastic blockchain for latency-sensitive vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 11862–11876, Aug. 2022.

[6] Y. Lu et al., "Safety warning! Decentralised and automated incentives for disqualified drivers auditing in ride-hailing services," *IEEE Trans. Mobile Comput.*, vol. 22, no. 3, pp. 1748–1762, Mar. 2023.

[7] Y. Lu et al., "Secure deduplication-based storage systems with resistance to side-channel attacks via fog computing," *IEEE Sensors J.*, vol. 22, no. 18, pp. 17529–17541, Sep. 2022.

[8] G. D. P. Regulation, "General data protection regulation (GDPR)," *Intersoft Consulting*, vol. 24, no. 1, 2018.

[9] K. Gai, Y. Wu, L. Zhu, M. Qiu, and M. Shen, "Privacy-preserving energy trading using consortium blockchain in smart grid," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3548–3558, Jun. 2019.

[10] M. Du, Q. Chen, J. Chen, and X. Ma, "An optimized consortium blockchain for medical information sharing," *IEEE Trans. Eng. Manag.*, vol. 68, no. 6, pp. 1677–1689, Dec. 2021.

[11] Y. Lu et al., "STRICTs: A blockchain-enabled smart emission cap restrictive and carbon permit trading system," *Appl. Energy*, vol. 313, 2022, Art. no. 118787. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261922002355

[12] E. Ben-Sasson et al., "Zerocash: Decentralized anonymous payments from bitcoin," in *Proc. IEEE Symp. Secur. Privacy*, 2014, pp. 459–474.

[13] S. Noether, A. Mackenzie, and T. M. Research Lab, "Ring confidential transactions," *Ledger*, vol. 1, pp. 1–18, 2016.

[14] M. Möser et al., "An empirical analysis of traceability in the monero blockchain," *Proc. Privacy Enhancing Technol.*, vol. 2018, no. 3, pp. 143–163, 2018.

[15] A. Biryukov, D. Feher, and G. Vitto, "Privacy aspects and subliminal channels in ZCaSH," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2019, pp. 1813–1829.

[16] G. Kappos, H. Yousaf, M. Maller, and S. Meiklejohn, "An empirical analysis of anonymity in Zcash," in *Proc. 27th USENIX Secur. Symp.*, 2018, pp. 463–477.

[17] F. Tramèr, D. Boneh, and K. G. Paterson, "Remote side-channel attacks on anonymous transactions," in *Proc. 29th USENIX Secur. Symp.*, 2020, pp. 2739–2756.

[18] A. Gervais, G. O. Karame, D. Gruber, and S. Capkun, "On the privacy provisions of bloom filters in lightweight bitcoin clients," in *Proc. 30th Annu. Comput. Secur. Appl. Conf.*, 2014, pp. 326–335.

[19] S. Matetic, K. Wüst, M. Schneider, K. Kostiainen, G. Karame, and S. Capkun, "BITE: Bitcoin lightweight client privacy using trusted execution," in *Proc. 28th USENIX Secur. Symp.*, 2019, pp. 783–800.

[20] Y. Niu, C. Zhang, L. Wei, Y. Xie, X. Zhang, and Y. Fang, "An efficient query scheme for privacy-preserving lightweight bitcoin client with intel SGX," in *Proc. IEEE Glob. Commun. Conf.*, 2019, pp. 6–11.

[21] E. Stefanov et al., "Path ORAM: An extremely simple oblivious RAM protocol," *J. ACM*, vol. 65, no. 4, pp. 1–25, 2018.

[22] R. Konrad and S. Pinto, "Bitcoin utxo lifespan prediction," CS229. stanford. edu, 2015.

[23] bips/bip-0157.mediawiki at master bitcoin/bips, 2009. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0157.mediawiki

[24] M. A. Rahman et al., "Blockchain-based mobile edge computing framework for secure therapy applications," *IEEE Access*, vol. 6, pp. 72469–72478, 2018.

[25] W. Liang, D. Zhang, X. Lei, M. Tang, K.-C. Li, and A. Y. Zomaya, "Circuit copyright blockchain: Blockchain-based homomorphic encryption for IP circuit protection," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 3, pp. 1410–1420, Third Quarter 2021.

[26] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via sharding," 2018. [Online]. Available: https://eprint.iacr.org/undefined/undefined

[27] S. Meiklejohn et al., "A fistful of bitcoins: Characterizing payments among men with no names," in *Proc. Conf. Internet Meas. Conf.*, 2013, pp. 127–140.

[28] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Proc. IEEE Symp. Secur. Privacy*, 2016, pp. 839–858.

[29] S. Meiklejohn and R. Mercer, "Möbius: Trustless tumbling for transaction privacy," *Proc. Privacy Enhancing Technol.*, vol. 2018, pp. 881–881, Apr. 2018.

[30] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "CoinShuffle: Practical decentralized coin mixing for bitcoin," *Lecture Notes Comput. Sci. (including subseries Lecture Notes Artif. Intell. Lecture Notes Bioinf.)*, vol. 8713, pp. 345–364, 2014.

[31] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from bitcoin," in *Proc. IEEE Symp. Secur. Privacy*, 2013, pp. 397–411.
[32] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *Proc. IEEE Symp. Secur. Privacy*, 2018, pp. 315–334.
[33] G. Yu, "Mimblewimble non-interactive transaction scheme," 2022. [Online]. Available: https://eprint.iacr.org/undefined/undefined
[34] M. Brandenburger, C. Cachin, R. Kapitza, and A. Sorniotti, "Blockchain and trusted computing: Problems, pitfalls, and a solution for hyperledger fabric," May 2018, *arXiv:1805.08541*. [Online]. Available: http://arxiv.org/abs/1805.08541
[35] A. Baliga, I. Subhod, P. Kamat, and S. Chatterjee, "Performance evaluation of the quorum blockchain platform," Jul. 2018, *arXiv:1809.03421*. [Online]. Available: http://arxiv.org/abs/1809.03421
[36] D. Ron and A. Shamir, "Quantitative analysis of the full bitcoin transaction graph," *Lecture Notes Comput. Sci. (including subseries Lecture Notes Artif. Intell. Lecture Notes Bioinf.)*, vol. 7859, pp. 6–24, 2013.
[37] G. Bissias, A. P. Ozisik, B. N. Levine, and M. Liberatore, "Sybil-resistant mixing for bitcoin," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2014, pp. 149–158.
[38] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Vienna Austria, 2016, pp. 17–30. [Online]. Available: https://dl.acm.org/doi/10.1145/2976749.2978389
[39] J. Xu, K. Paruch, S. Cousaert, and Y. Feng, "SoK: Decentralized exchanges (DEX) with automated market maker (AMM) protocols," *ACM Comput. Surv.*, vol. 55, no. 11, pp. 238:1–238:50, Feb. 2023. [Online]. Available: https://doi.org/10.1145/3570639
[40] J. K. Mullin, "A second look at bloom filters," *Commun. ACM*, vol. 26, no. 8, pp. 570–571, 1983.
[41] J. Kang et al., "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4660–4670, Jun. 2019.
[42] U. Javaid, M. N. Aman, and B. Sikdar, "DrivMan: Driving trust management and data sharing in VANETs with blockchain and smart contracts," in *Proc. IEEE Veh. Technol. Conf.*, 2019, pp. 1–5.
[43] K. Shi, L. Zhu, C. Zhang, L. Xu, and F. Gao, "Blockchain-based multimedia sharing in vehicular social networks with privacy protection," *Multimedia Tools Appl.*, vol. 79, pp. 8085–8105, Mar. 2020. [Online]. Available: https://link.springer.com/article/10.1007/s11042--019-08284-8
[44] Y. Chen, X. Hao, W. Ren, and Y. Ren, "Traceable and authenticated key negotiations via blockchain for vehicular communications," *Mobile Inf. Syst.*, vol. 2019, pp. 1–10, 2019.
[45] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious RAMs," *J. ACM*, vol. 43, no. 3, pp. 431–473, 1996.
[46] S. Tople, Y. Jia, and P. Saxena, "PRO-ORAM: Practical read-only oblivious RAM," in *Proc. 22nd Int. Symp. Res. Attacks Intrusions Defenses*, 2019, pp. 197–211.
[47] E. Stefanov, E. Shi, and D. X. Song, "Towards practical oblivious RAM," in *Proc. 19th Annu. Netw. Distrib. Syst. Secur. Symp.*, San Diego, California, USA, 2012. [Online]. Available: https://www.ndss-symposium.org/ndss2012/towards-practical-oblivious-ram
[48] K. Kanemura, K. Toyoda, and T. Ohtsuki, "Design of privacy-preserving mobile bitcoin client based on -deniability enabled bloom filter," in *Proc. IEEE Int. Symp. Pers. Indoor Mobile Radio Commun.*, 2018, pp. 1–6.

Youshui Lu (Member, IEEE) received the BS degree from The Australian National University, Australia, in 2013, the MS degree from The University of Sydney, Australia, in 2015, and the PhD degree from the School of Computer Science and Technology, Xi'an Jiaotong University, China, in 2021. He is currently an assistant professor with the School of Electrical Engineering and School of Instrument Science and Technology, Xi'an Jiaotong University. His research interests include blockchain technology, distributed systems, Internet of Things, data security and smart grid.

Bowen Cai received the BS and MS degrees from The Xi'an Jiaotong University, China, in 2020 and 2023, respectively. He is currently working toward the PhD degree with the School of Computer Science and Engineering, University of Minnesota, Twin City, Minnesota. His research interests include blockchain technology, machine learning, code analysis and system security.

Xiaojun Tang (Member, IEEE) received the BS and MS degrees in control theory and control engineering from the Xi'an University of Technology, Xi'an, in 1998 and 2001 respectively, and the PhD degree in instrument science & technology from Xi'an Jiaotong University, Xi'an, in 2004. From 2007 to 2008, he was a postdoctoral fellow with the University of New Orleans. Since 2014, he has been professor with the Department of Measurement & Control, School of Electrical Engineering and School of Instrument Science and Technology, Xi'an Jiaotong University. His research interests include smart sensor and instrumentation, condition monitoring technology for power equipment and smart control.

Lei Liu (Member, IEEE) received the BEng degree in communication engineering from Zhengzhou University, Zhengzhou, China, in 2010, and the MSc and PhD degrees in communication engineering from Xidian University, Xi'an, China, in 2013 and 2019, respectively. From 2013 to 2015, he worked with Technology Company. From 2018 to 2019, he was supported by China Scholarship Council to be a Visiting PhD Student with the University of Oslo, Oslo, Norway. He is currently a lecturer with the Department of Electrical Engineering and Computer Science, Xidian University. His research interests include vehicular ad hoc networks, intelligent transportation, mobile-edge computing, and Internet of Things.

Jun Du (Senior Member, IEEE) received the B.S. degree in information and communication engineering from the Beijing Institute of Technology, Beijing, China, in 2009, and the M.S. and Ph.D. degrees in information and communication engineering from Tsinghua University, Beijing, in 2014 and 2018, respectively. From October 2016 to September 2017, she was a Sponsored Researcher, and she visited Imperial College London. She is currently an Assistant Professor with the Department of Electrical Engineering, Tsinghua University. Her research interests include communications, networking, resource allocation and system security problems of heterogeneous networks, and space-based information networks. She was a recipient of the Best Student Paper Award from IEEE GlobalSIP in 2015, the Best Paper Award from IEEE ICC 2019, and the Best Paper Award from IWCMC in 2020.

Shui Yu (Fellow, IEEE) received the PhD degree from Deakin University, Australia, in 2004. He is currently a professor with the School of Computer Science, University of Technology Sydney, Australia. His current H-index is 67. He has published five monographs, edited two books, and more than 500 technical papers at different venues, such as the *IEEE Transactions on Dependable and Secure Computing*, the *IEEE Transactions on Parallel and Distributed Systems*, the *IEEE Transactions on Computers*, the *IEEE Transactions on Information Forensics and Security*, the *IEEE Transactions on Mobile Computing*, the *IEEE Transactions on Knowledge and Data Engineering*, the *IEEE Transactions on Emerging Topics in Computing*, the *IEEE/ACM Transactions on Networking*, and INFOCOM. He has been promoting the research field of networking for Big Data since 2013, and his research outputs have been widely adopted by industrial systems, such as Amazon cloud security. His research interests include cybersecurity, network science, Big Data, and mathematical modeling. He is an elected member of Board of Governors of the IEEE VTS and ComSoc. He is a member of ACM and AAAS. He is also serving the editorial boards for the *IEEE Communications Surveys and Tutorials* (area editor) and the *IEEE Internet of Things Journal* (editor). He is also a distinguished visitor of the IEEE Computer Society. He served as a distinguished lecturer for the IEEE Communications Society from 2018 to 2021.

Mohammed Atiquzzaman received the MS and PhD degrees in electrical engineering and electronics from the University of Manchester, Manchester, U.K., in 1984 and 1987, respectively. He currently holds the Edith Kinney Gaylord Presidential Professorship with the School of Computer Science, The University of Oklahoma, Norman, Oklahoma. He has more than 450 refereed technical publications. He received the NASA Group Achievement Award, the IEEE Satellite and Space Communications Technical Recognition Award, the IEEE Distinguished Technical Achievement Award, and the IEEE Distinguished Service Award. He is the editor-in-chief of the *Journal of Networks and Computer Applications*, the founding editor-in-chief of *Vehicular Communications*, and the former co-editor-in-chief of *Computer Communication*, and has served/serving on the editorial boards of many highly ranked journals, such as *IEEE Transactions on Mobile Computing* and *IEEE Journal on Selected Areas in Communications*.

Schahram Dustdar (Fellow, IEEE) received the PhD degree in business informatics from the University of Linz, Linz, Austria, in 1992. He is currently a full professor of computer science (informatics) with a focus on internet technologies heading the Distributed Systems Group, TU Wien, Wein, Austria. He has been the chairman of the Informatics Section of the Academia Europaea, since December 2016. He has been a member of the IEEE Conference Activities Committee (CAC), since 2016, the Section Committee of Informatics of the Academia Europaea, since 2015, and the Academia Europaea: The Academy of Europe, Informatics Section, since 2013. He was a recipient of the ACM Distinguished Scientist Award, in 2009 and the IBM Faculty Award, in 2012. He is an associate editor for the *IEEE Transactions on Services Computing*, *ACM Transactions on the Web*, and *ACM Transactions on Internet Technology*. He is on the editorial board of IEEE.