ИТМО

НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
сентябрь–октябрь 2024 Том 24 № 5 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
September–October 2024 Vol. 24 No 5 http://ntv.ifmo.ru/en/
ISSN 2226-1494 (print) ISSN 2500-0373 (online)

НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

doi: 10.17586/2226-1494-2024-24-5-797-805

Enhancing attribute-based access control with Ethereum and ZK-SNARK technologies

Maher Maalla[1], Sergey V. Bezzateev[2]✉

[1,2] ITMO University, Saint Petersburg, 197101, Russian Federation
[2] Saint Petersburg State University of Aerospace Instrumentation, Saint Petersburg, 190000, Russian Federation

[1] maher.malla7@gmail.com, https://orcid.org/0000-0002-4806-8608
[2] bsv@guap.ru✉, https://orcid.org/0000-0002-0924-6221

Abstract
Attribute Based Access Control (ABAC) is one the most efficient, scalable, and well used access control. It's based on attributes not on users, but even when the users want to get access to some resource, they must submit their attributes for the verification process which may reveal the privacy of the users. Many research papers suggest blockchain-based ABAC which provides an immutable and transparent access control system. However, the privacy of the system may be compromised depending on the nature of the attributes. A Zero-Knowledge Proof, Ethereum-Based Access Control (ZK-ABAC) is proposed in this paper to simplify the management of access to the devices/objects and provide an efficient and immutable platform that keeps track of all actions and access management and preserve the privacy of the attributes. Our ZK-ABAC model utilizes smart contracts to facilitate access control management, Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (ZK-SNARK) protocol to add privacy to attributes, InterPlanetary File System (IPFS) network to provide distributed storage system, and Chainlink to manage communications and data between on/off-chain systems. Comprehensive experiments and tests were conducted to evaluate the performance of our model, including the implementation of ZK-SNARK on the Ethereum blockchain. The results demonstrated the scalability challenges in the setup and proving phases, as well as the efficiency gains in the verification phase, particularly when scaled to higher numbers of users. These findings underscore the practical viability of our ZK-ABAC model for secure and privacy-preserving access control in decentralized environments.

Keywords
ABAC, Ethereum, ZK-SNARK, zero-knowledge proofs, privacy, blockchain

For citation: Maalla M., Bezzateev S.V. Enhancing attribute-based access control with Ethereum and ZK-SNARK technologies. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2024, vol. 24, no. 5, pp. 797–805. doi: 10.17586/2226-1494-2024-24-5-797-805

УДК 004.056.55

Усовершенствование контроля доступа на основе атрибутов с помощью технологий Ethereum и ZK-SNARK

Махер Маалла[1], Сергей Валентинович Беззатеев[2]✉

[1,2] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация
[2] Санкт-Петербургский государственный университет аэрокосмического приборостроения, Санкт-Петербург, 190000, Российская Федерация

[1] maher.malla7@gmail.com, https://orcid.org/0000-0002-4806-8608
[2] bsv@guap.ru✉, https://orcid.org/0000-0002-0924-6221

Аннотация
Система разграничения доступа на основе атрибутов (Attribute-Based Access Control, ABAC) — одна из наиболее эффективных и широко используемых систем контроля доступа, обеспечивающая масштабируемость. Когда пользователи хотят получить доступ к информационному ресурсу, им необходимо предоставить свои атрибуты для процесса верификации, что может, в свою очередь, раскрыть их личные данные. Во многих

исследовательских работах предлагается контроль доступа, основанный на атрибутах, использующий блокчейн, что обеспечивает защищенную от искажений (Zero-Knowledge, ZK) и прозрачную систему контроля доступа. Однако конфиденциальность системы может быть нарушена в зависимости от характера атрибутов. В работе предлагается использовать систему контроля доступа на основе Ethereum и доказательств без разглашения (ZK-ABAC). Система упрощает управление доступом к устройствам/объектам, обеспечивает эффективную и защищенную от искажений платформу, контролирующую все действия и управление доступом. При этом сохраняется конфиденциальность атрибутов. Предлагаемая модель контроля доступа на основе технологии ZK-ABAC использует смарт-контракты для управления доступом. Протокол ZK-SNARK обеспечивает конфиденциальность атрибутов пользователей. Система InterPlanetary File System применяется для создания распределенной системы хранения данных, а Chainlink для управления связью и данными между внутренними/внешними блокчейн-системами. Для оценки работоспособности предложенной модели проведены эксперименты и тесты, включая использование ZK-SNARK с блокчейн-технологией Ethereum. Результаты экспериментов продемонстрировали проблемы масштабируемости на этапах настройки и проверки, а также повышение эффективности на этапе верификации при масштабировании для большего числа пользователей. Полученные результаты подтвердили практическую эффективность предложенной модели ZK-ABAC для безопасного управления доступом с сохранением конфиденциальности в децентрализованных средах.

Ключевые слова
контроль доступа, основанный на атрибутах (ABAC), ZK-SNARK, доказательство без разглашения, конфиденциальность, блокчейн

Ссылка для цитирования: Маалла М., Беззатеев С.В. Усовершенствование контроля доступа на основе атрибутов с помощью технологий Ethereum и ZK-SNARK // Научно-технический вестник информационных технологий, механики и оптики. 2024. Т. 24, № 5. С. 797–805 (на англ. яз.). doi: 10.17586/2226-1494-2024-24-5-797-805

Introduction

The importance of user privacy has garnered increasing attention, particularly in the era of social media and the alleged privacy violations by large technology companies seeking market dominance. Incorporating privacy features into access control systems has become a critical requirement, especially when dealing with highly sensitive information, financial records, and public blockchain platforms where data is immutable and visible to all participants. Furthermore, the integration of Attribute-Based Access Control (ABAC) with blockchain technology, which leverages the immutability characteristic of blockchains, necessitates a method to preserve user privacy when submitting attributes to smart contracts for access authorization. Zero-Knowledge Proofs (ZKPs) emerge as a solution to this challenge, providing a mechanism to verify user attributes for ABAC without disclosing those attributes on the blockchain, thereby maintaining privacy.

ZKPs have rapidly evolved from theoretical concepts to practical tools, revolutionizing several aspects of digital systems [1–3]. In the realm of data management, ZKPs have become instrumental in ensuring storage integrity. They allow verification of data authenticity and completeness without exposing sensitive information, a crucial capability for industries handling confidential records. In the financial sector, ZKPs are transforming digital asset transfers [4]. By enabling users to prove ownership and conduct transactions without revealing personal details or transaction amounts, ZKPs strike a balance between privacy and transparency in blockchain-based finance. Moreover, ZKPs are addressing one of the most pressing challenges in blockchain technology: scalability [5, 6]. By allowing complex computations to be performed off-chain and efficiently verified on-chain, ZKPs significantly reduce the computational load on blockchain networks. This breakthrough paves the way for increased transaction throughput without compromising security or decentralization. As ZKP technology continues to advance, these applications in storage integrity [7], private digital asset transfers, and blockchain scalability are expected to drive significant innovations in secure and efficient digital systems.

The integration of ZKPs with blockchain technology has been the focus of numerous research initiatives. Among these, several systems have particular relevance to our work. One notable example is a study that explores the application of blockchain and Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (ZK-SNARK) in adding privacy to a healthcare system for Internet of Things (IoT) [8]. This innovative approach leverages the inherent anonymity of blockchain to protect user privacy, while employing ZK-SNARK-based authentication mechanisms to prevent unauthorized access to sensitive medical data. The study [9] introduces a blockchain-based Access Control (AC) system for IoT that uses Zero-Knowledge Rollups (ZK-rollups) to address issues of low transaction speed and high latency in high-traffic environments. By batching AC authorization requests into a single zero-knowledge proof, the system enhances trustworthiness and efficiency. Experiments show that the system significantly reduces authorization time, particularly in high-traffic scenarios, while also preventing malicious behaviors.

The issue of this system alone is where should we store the data related to the access management, security policies, and user management, this issue is resolved by using a decentralized InterPlanetary File System (IPFS) which is designed for distributed peer-to-peer sharing which solve the centralization problem. IPFS offers a decentralized way of storing and sharing data, enhancing efficiency and speed by retrieving files from the nearest node. It resists censorship, reduces redundancy, and provides a more robust, version-controlled system for a persistent and resilient internet. This technology is particularly beneficial for decentralized applications, content distribution, and digital archiving [10]. We can communicate with this IPFS network by using Chainlink which connects existing systems to any public or private blockchain and enables secure cross-chain communication, in this way we shift storing data on-chain to off-chain IPFS in secure way using Chainlink and provide better performance on Ethereum network [11].

Proposed Model

The proposed model for a Zero-Knowledge Proof, Ethereum-Based Attribute-Based Access Control (ZK-ABAC) system, integrated with the Ethereum blockchain IPFS through Chainlink, presents a novel approach that provides efficient, transparent, and decentralized access control while preserving the privacy of user attributes. This model leverages traditional ABAC to ensure scalable access control, with the entire management and mechanism process executed by the Ethereum network via smart contracts, thereby imparting immutability to the system. All requisite data for the model is stored on the decentralized IPFS, and the entirety of communication between on-chain and off-chain components is governed by smart contracts.

Smart contract is the core element in the model since it controls the whole process, verifies the attributes provided by the users, grants/denies the access, and manages access policies and all other data related to IoT devices on IPFS [12]. Instead of attributes we use ZK-SNARK to verify the attributes without revealing them.

One of the innovative aspects of this model is the integration with Chainlink, a decentralized oracle network. This integration allows the smart contracts to interact with off-chain data sources and services securely and reliably. Specifically, attributes and device information, which are crucial for the ABAC system, are stored on IPFS servers. Chainlink oracles provide a bridge between these off-chain data stored on IPFS and the on-chain smart contracts on Ethereum. This ensures that the AC system can access up-to-date and accurate information about users and devices, which is essential for making correct access control decisions [13, 14].

Many research study different types of blockchain integrating with different types of access control [15–18]. However, no research addresses the privacy of users whose identities could potentially be revealed through the attributes (e.g., roles, positions, locations) used to gain access to resources. Our main contribution is applying ZKP to this system so we can assure privacy. The main focus is pointed at structuring the system and making ZKP integration its main core.

ZK-SNARK is a complex cryptographic construct. To understand how they could be used in conjunction with an ABAC system, it's important to delve into some technical aspects and equations that underpin ZK-SNARK. ZK-SNARK is built on a foundation of polynomial equations that are essential for their operation. These equations transform a computation, such as checking attributes for ABAC, into a set of equations within a finite field. A prevalent method for representing these polynomial equations is through Quadratic Arithmetic Programs (QAPs). QAPs play a crucial role in verifying the accurate execution of the ABAC policy check. Additionally, ZK-SNARK employs elliptic curve pairings which are instrumental for efficient proof generation and verification. These pairings link the polynomial equations to cryptographic components, enhancing ZK-SNARK functionality.

There are three major steps for our proposed model.

Step 1. System Setup. In which we configure the system and initializing it to integrate the required technologies used in this mode which includes:

- **Defining ABAC Policies:** is the core of the access control mechanism, by defining the attributes that the ABAC model will process and deal with, also defining the policies that will manage the access control procedure. Those policies and object data is stored on IPFS;
- **Ethereum Smart Contracts:** is responsible for communicating with the users and the required objects, handling the ABAC checks, and managing data on IPFS servers. These contracts contain the logic to verify access based on user attributes;
- **ZK-SNARK Setup:** is about performing the trusted setup for ZK-SNARK to generate public parameters (proving and verification keys) and developing a ZK-SNARK circuits that can take user attributes and generate a proof without revealing the attributes themselves.

User Attribute Tokenization

Step 2. This step is about representing the attributes by tokens to be processed when the access is triggered and initializing the ZK-SNARK proofs which includes:

- **Issue Attribute Tokens:** Users receive tokens representing their attributes. These tokens are stored on Ethereum and can be verified by the smart contract;
- **ZK-SNARK Proof Generation:** Users generate ZK-SNARK proof that they possess tokens with the required attributes. This proof asserts the presence of attributes without revealing what they are.

Step 3. Access Request. This step is the actual process after setting up the environment, which includes:

- **Submit Access Request:** When a user wants to access a resource, they interact with the Ethereum smart contract. They submit their ZK-SNARK proof along with the access request;
- **Smart Contract Verifies Proof:** The smart contract uses the ZK-SNARK verification key to verify the proof. If the proof is valid, it confirms that the user has the necessary attributes;
- **Grant or Deny Access:** Based on the result of the ZK-SNARK proof verification, the smart contract grants or denies access to the resource.

This model is represented in Fig. 1.

Integrating technologies like ZK-SNARK, IPFS, and Chainlink with ABAC on Ethereum provides a distributed, efficient, and privacy-preserving access control mechanism.

The Proposed Model Architecture

We will dive into details about configuring this model and integrating all the mentioned technologies together to draw the full image of the proposed model.

Define attributes

The first step to implement our model is to define the attributes and policies according to ZK-SNARK proofs.

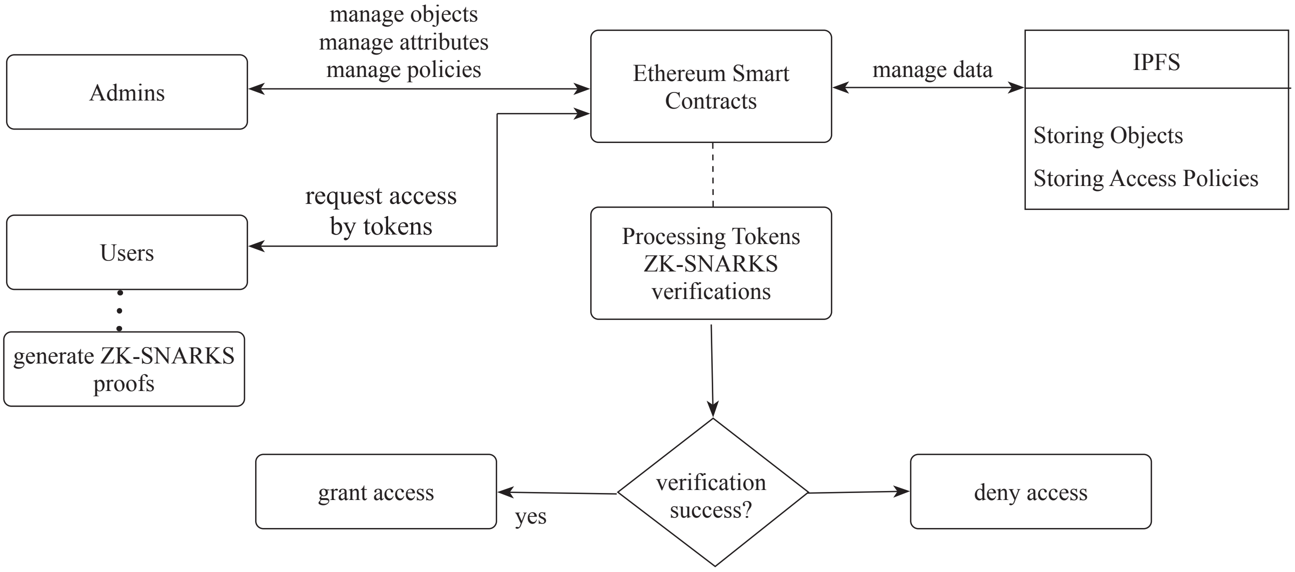

Fig. 1. System flow

Those policies should be presented as polynomial equations to be used in the ZK-SNARK setup phase.

Many studies discussed how to define attributes in a generic way that generalizes the nature of attributes that can apply and serve wide range of different systems [13, 19]. Attributes can be grouped, symbolized as $A \in \{S, O, P, E\}$, where A = {name: value}:

- S signifies the subject attribute, which can be presented as an ID, profession… etc.
- O pertains to the object attribute, which can be IP address, category, device…etc.
- P relates to the permission attribute, like read, write, delete, or executing.
- E is the environment attribute, like time, date, physical location… etc.

Represent Policies as Polynomial Equations

Having the attributes presented as set of variables: $A \in \{S, O, P, E\}$. Each variable can take multiple values based on the specific attribute it represents. For example, S_i for different roles, O_i for different objects, etc. Taking into consideration that the policy to access a certain object should be presented by one (or more) value for every attribute, the user request should be built using at least one value for each attribute type. Therefore, missing one attribute should make the access process invalid. Moreover, the policy can contain several values for the same attribute type. We should aim for a balance where the polynomials are complex enough to ensure security but not so complex that they become inefficient to compute. Therefore, we consider the access policy to be presented by set of quadratic polynomials, one polynomial for each attribute, and these can be combined to form the overall policy representation.

ZK-SNARK often uses QAPs to turn the check of validity of the access into a set of quadratic equations. QAPs can efficiently represent complex computations and are suitable for a wide range of ABAC policies.

Based on the previous, we propose the polynomial equation for the access policy to be built as follows:

- Each attribute type (Subject, Object, Permission, Environment) is represented by a set of possible values;
- Let A represents an attribute type (e.g., Subject), and A_i represents each possible value of A.

We propose generic equation for each attribute type as follows:

- For an attribute A requiring to be a specific single value A_k

$$P_A = (A - A_k)^2;$$

- For an attribute A requiring multiple values $A_1, A_2, \ldots, A_n$ (AND operation)

$$P_A = \sum_{i=1}^{n} (A - A_i)^2;$$

- For an attribute A allowing any one of several values $A_1, A_2, \ldots, A_n$ (OR operation)

$$P_A = \prod_{i=1}^{n} (A - A_i).$$

Constructing a new access policy should be set of four attribute types (four quadratic equations) depending on the rule itself:

$$P_S + P_O + P_P + P_E = 0.$$

For example, let's consider a policy where Subjects S_1 or S_3 have Permissions P_1 and P_2 on Object O_1 with Environment E_7. The polynomial equation representing this access policy should be constructed using quadratic equation for each attribute type as follows:

$$(S - S_1)\,(S - S_3) + (P - P_1)^2 + (P - P_2)^2 + \\ + (O - O_1)^2 + (E - E_7)^2 = 0.$$

Depending on the required access conditions, the admin can create a specific policy using the previous logic and link it to a specific object.

ZK-SNARK Circuit Design and Setup

In the development of our ZK-SNARK system for ABAC policy verification, the circuit design is a crucial component. This circuit is meticulously engineered to assess the formulated polynomial equations that represent the ABAC policies. It operates by accepting user attributes as inputs and computing the corresponding polynomial value. The design ensures that if all specified attribute conditions within a policy are satisfied, the polynomial evaluation results in zero, signifying adherence to the policy. The system trusted setup is integral to its security architecture, involving the generation of essential cryptographic materials, specifically the proving key and the verification key, both of which are crucial to the ZK-SNARK framework.

The trusted setup procedure begins with the selection of cryptographic parameters, including appropriate elliptic curves and other foundational elements that underpin the security of the ZK-SNARK. In a secure environment, secret random values are then generated to create the proving and verification keys, ensuring true randomness in this critical step. The proving key, which is typically large, enables users to create proofs demonstrating compliance with ABAC policies without revealing their attributes. This key must be securely distributed to users. Simultaneously, the verification key generated using the same secret randomness allows the Ethereum smart contract to verify the proofs submitted by users. Unlike the proving key, the verification key is much smaller and is deployed within the Ethereum smart contract.

After generating the keys, the secret randomness used in their creation is securely and irreversibly destroyed to prevent the possibility of generating false proofs. The proving key is then distributed to users through secure channels, ensuring that every user who needs to generate proofs has access to it, while the verification key is embedded within the Ethereum smart contract responsible for verifying access requests. Secure record-keeping of the cryptographic parameters and keys (excluding the secret randomness) is maintained for system maintenance and auditing purposes. Finally, the ZK-SNARK system is fully integrated with the policy management system, allowing for proofs to be generated based on the latest policies stored on IPFS and verified through Ethereum.

To summarize, ZK-SNARK is built on three main phases: Setup, Prove, and Verify:

— The Setup phase generates a pair of keys used by the prover and verifier.

$$Setup(l^{\lambda}) \rightarrow (p_k, v_k). \tag{1}$$

— The Prove phase takes the proving key, a statement to be proved (in terms of public inputs), and private inputs (known as the witness), and produces a proof π.

$$Prove(p_k, x, w) \rightarrow \pi. \tag{2}$$

— The Verify phase uses the verification key, the public inputs, and the proof to determine whether the proof is valid, i.e., whether it correctly demonstrates the truth of the statement without the verifier needing to know the private inputs.

$$Verify(v_k, x, \pi) \rightarrow \{0, 1\}, \tag{3}$$

where l^{λ} is the security parameter (indicating the size and strength of the cryptographic setup); p_k is the Proving Key which will be used by the prover; v_k is the Verification Key which will be used by the verifier; x represents the public inputs to the statement being proved; w is the witness or private inputs known only to the prover; π is the proof that the prover constructs, demonstrating that the inputs satisfy the statement without revealing the witness.

The output is a binary value, where 1 indicates that the proof is valid and 0 indicates that it is not.

After successfully generating the required key, we can represent ZK-SNARK circuit design by the next algorithm in Algorithm 1 — ZK-SNARK Proof Generation with Detailed Implicit Policy Equation.

```
procedure GenerateAccessProof(UserAttributes, ObjectID, IPFS_Client, ProverKey, EthereumSC_Address)
    Fetch the policy for the requested object using ObjectID from IPFS.
    PolicyData ← IPFS_Client.Fetch(ObjectID)
    IsValidPolicy ← VerifyPolicyHash(PolicyData, EthereumSC_Address)
    if IsValidPolicy then
        ZK-SNARK_Proof ← ZK-SNARKProvingAlgorithm(UserAttributes, ProverKey)
        AccessRequest ← package(ZK-SNARK_Proof, ObjectID)
        AccessGranted ← EthereumSC.VerifyProof(AccessRequest, EthereumSC_Address)
        if AccessGranted then
            return Access token from Smart Contract
        else
            return Access denied
        end if
    else
        return Invalid policy data
    end if
end procedure
```

The algorithm for ZK-SNARK Proof Generation with Implicit Policy Equation is designed to enable a user to generate a zero-knowledge proof for accessing a specific object based on predefined access control policies. The algorithm operates within an environment where policies are stored on IPFS and verified through Ethereum, where:

— UserAttributes: The attributes of the user, such as their role, permissions, and other relevant details.

— ObjectID: The unique identifier of the object that the user wants to access.

— IPFS_Client: A client interface to interact with IPFS for retrieving policy data.

— ProverKey: A cryptographic key obtained from the ZK-SNARK trusted setup phase, used for proof generation.

— EthereumSC_Address: The address of the Ethereum Smart Contract used for verifying the proof.

Smart Contracts, IPFS, and Chainlink Oracle

In our paper we will focus only on the ZK-SNARK verification process for our model and not on the smart contracts or IPFS and Chainlink integrations. About Smart Contracts, there are many studies about it and it's been discussed in detail how many smart contracts needed and the structure of those contracts to facilitate the process [20, 21, 13]. Also we already conducted detailed research about that part in another article, including IPFS and Chainlink integrations [22].

The system employs a series of smart contracts to manage access control and data handling, leveraging the strengths of blockchain, IPFS, and Chainlink.

— **AccessRequestContract** handles user access requests by processing submitted attributes, retrieving relevant policies, and initiating the policy evaluation process.

— **AdminPolicyManager** allows administrators to manage policies, including adding, updating, viewing, and deleting them, ensuring that the system rules are up-to-date and properly enforced.

— **IoTDataManager** is responsible for managing the resources allowing for the registration, updating, and deletion of resources, as well as retrieving specific resource information.

— **PolicyEvaluator** plays a critical role by evaluating user attributes against the stored policies to grant or deny access based on compliance.

— **IPFSDataHandler** interfaces with the IPFS network to store and retrieve data, enhancing data availability and security through decentralized storage.

— **ChainlinkOracleAdapter** facilitates secure communication with off-chain data sources using Chainlink oracles, enabling the system to access external data reliably.

These smart contracts collectively create a robust and efficient access control system, ensuring secure, transparent, and decentralized management of resources and data. Therefore, this work is a continuation of our work regarding smart contracts and the other setups.

Experiments and results

The experiments and tests to evaluate the performance of our model have been conducted on a PC equipped with an Intel i7 processor (2.60 GHz) and 16 GB of RAM for the prototype implementation. For the development of smart contracts, the Solidity language was used. These smart contracts were created using Solidity and deployed on the Goerli testnet, which serves as a testing platform for Ethereum smart contract development.

In our experiments, we utilized ZoKrates in conjunction with the Groth16 proving scheme to explore and validate the efficacy of implementing ZK-SNARK on the Ethereum blockchain [23–25]. ZoKrates provided a streamlined development environment enabling to define and compile privacy-preserving arithmetic circuits using its specialized domain-specific language. These circuits formed the backbone of our experimental setup, allowing us to generate zero-knowledge proofs efficiently. We then leveraged Groth16, renowned for its succinct proofs and efficient verification properties, to ensure the integrity and non-disclosure of sensitive computation data within our tests. This integration was crucial for demonstrating the potential of ZK-SNARK in enhancing privacy and scalability in blockchain applications.

We tested the functions presented in formulas (1)–(3). We measured the time cost of the three functions by building different circuits based on different number of users and calculated the required time for each function to complete off-chain, the result is shown in Table.

And the chart presenting the result is shown in Fig. 2.

Analysis of Results

— As the number of users increases, the time taken for both setup and proving phases increases significantly. This indicates that the computation and resource requirements scale with the number of users, which is expected since each user would require a unique proof.

— The setup time increases more dramatically than the proving time. For instance, between 100 and 10,000 users, setup time increases by over 90 times, while

Table. ZK-SNARK cost time by Number of Users

Number of Users	Setup	Prove	Verify
100	0.285	0.190	0.194
500	1.345	0.966	0.201
1,000	2.887	1.789	0.209
5,000	9.446	4.102	0.285
10,000	26.224	11.325	0.206

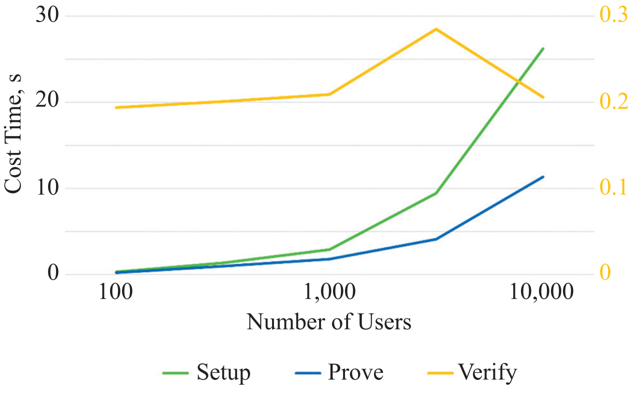

Fig. 2. Setup, Prove, and Verify cost time by number of users

proving time increases by about 60 times. This suggests that the initial parameter generation is highly resource-intensive and might become a bottleneck at scale.

- The verify phase does not scale in the same way as the setup and prove phases. The verification time remains relatively consistent and increases only marginally with the number of users. This is characteristic of ZK-SNARK, where verification is typically quick and does not significantly depend on the number of users.
- The fact that the verify phase times are small and stable is highly advantageous for scenarios where many independent verifications need to occur, such as on a blockchain network.

This experiment demonstrates the scalability challenges of ZK-SNARK in terms of setup and proving times. While verification remains fast regardless of the user count, the setup phase, in particular, may pose challenges for large-scale implementations due to its super-linear growth in time requirement. It's crucial for applications that utilize ZK-SNARK to consider these performance implications, especially for systems that require frequent setup or proving operations.

We also conducted an experiment about only the Verify process when executing it on Ethereum, we already did an experiment about smart contracts performance in general [22], and now we're testing the verification process on Goerli testnet. We adjusted the smart contract used for verification (PolicyEvaluator) to use ZoKrates for testing the time cost for the verification instead of the normal process described in [22], and the result is shown in Fig. 3.

Analysis of Results

- **ZK-SNARK Efficiency:** The orange line representing the verification time using ZK-SNARK is consistently below the blue line, which suggests that using ZK-SNARK for verification is more time-efficient than the traditional verification method.
- **Scalability:** As the number of requests increases, both methods show an increase in cost time. However, the increase in the time cost for the traditional verification method is more pronounced, which indicates that the verification with ZK-SNARK scales better with the number of requests.
- **Performance at Scale:** At lower numbers of requests (e.g., 10 to 100), the difference in verification time between the two methods is relatively small. However, as the number of requests grows (e.g., from 500 to 1000), the difference becomes much more significant. This suggests that the benefits of ZK-SNARK become more apparent as the system is scaled up.
- **Goerli Testnet Context:** It's important to note that these tests were performed on the Goerli testnet which is an Ethereum test network. Real-world conditions on the Ethereum mainnet might lead to different performance characteristics due to network congestion and gas prices, although the relative performance between the two methods might be expected to remain consistent.

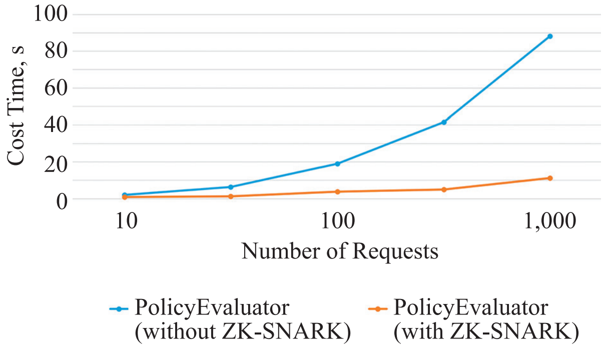

Fig. 3. Cost time for verification with ZK-SNARK and without ZK-SNARK

The use of ZK-SNARK (via ZoKrates) for the verification process in smart contracts significantly reduces the verification time, especially as the number of requests increases. This can lead to performance improvements in blockchain applications that require a large number of verifications. The lower time cost associated with ZK-SNARK verification can lead to more efficient smart contract operations, which can be particularly beneficial for applications with high throughput requirements.

All other tests related to the Smart Contracts in our work are included in the previously published paper [22].

Conclusion

This paper presents a pioneering approach to Access Control in the realm of IoT and beyond, through the integration of ABAC with blockchain technology, ZK-SNARK, IPFS, and Chainlink oracles. Our proposed ZK-ABAC system represents a significant advancement in addressing the critical challenge of preserving user privacy while maintaining a robust, immutable, and transparent access control mechanism.

Our ZK-ABAC model innovatively combines the flexibility and efficiency of ABAC with the robustness and transparency of blockchain technology. By employing Ethereum smart contracts, the system ensures a decentralized and tamper-proof record of access control policies and transactions. The utilization of ZK-SNARK is pivotal in safeguarding user privacy; it enables users to prove their attribute-based access rights without revealing the actual attributes, thus maintaining confidentiality in every interaction, and the results show that the verification process is more efficient and less time consuming than the normal process and makes it suitable for on-chain applications.

Furthermore, the integration of the IPFS network facilitates a distributed storage solution, ensuring that access control policies are not only decentralized but also resilient and scalable. This feature is particularly crucial in addressing the concerns of centralized data management and single points of failure, which are common in traditional access control systems.

In conclusion, the ZK-ABAC model stands as a testament to the potential of combining blockchain technology, zero-knowledge proofs, distributed storage, and oracle networks to revolutionize access control systems. It paves the way for future research and development in this field, setting a new standard for privacy-preserving, decentralized access control in the digital age.

References

1. Goldwasser S., Micali S., Rackoff C. The knowledge complexity of interactive proof-systems. *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, 2019, pp. 203–225. https://doi.org/10.1145/3335741.3335750
2. Chiesa A., Hu Y., Maller M., Mishra P., Vesely N., Ward N. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. *Lecture Notes in Computer Science*, 2020, vol. 12105, pp. 738–768. https://doi.org/10.1007/978-3-030-45721-1_26
3. Campanelli M., Gailly N., Gennaro R., Jovanovic P., Mihali M., Thaler J. Linear time prover snarks with constant size proofs and square root size universal setup. *Lecture Notes in Computer Science*, 2023, vol. 14168, pp. 331–351. https://doi.org/10.1007/978-3-031-44469-2_17
4. Fuchsbauer G., Orrù M., Seurin Y. Aggregate cash systems: A cryptographic investigation of mimblewimble. *Lecture Notes in Computer Science*, 2019, vol. 11476, pp. 657–689. https://doi.org/10.1007/978-3-030-17653-2_22
5. Ozdemir A., Wahby R. Scaling verifiable computation using efficient set accumulators. *Proc. of the 29th USENIX Conference Security Symposium*, 2020, pp. 2075–2092.
6. Xie T., Zhang J., Cheng Z., Zhang F., Zhang Y., Jia Y., Boneh D., Song D. zkbridge: Trustless cross-chain bridges made practical. *Proc. of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 3003–3017. https://doi.org/10.1145/3548606.3560652
7. Parno B., Howell J., Gentry C., Raykova M. Pinocchio: Nearly practical verifiable computation. *Communications of the ACM*, 2016, vol. 59, no. 2, pp. 103–112. https://doi.org/10.1145/2856449
8. Luong D.A., Park J.H. Privacy-preserving blockchain-based healthcare system for IoT devices using ZK-SNARK. *IEEE Access*, 2022, vol. 10, pp. 55739–55752. https://doi.org/10.1109/access.2022.3177211
9. Lin X., Zhang Y., Huang C., Xing B., Chen L., Hu D., Chen Y. An access control system based on blockchain with zero-knowledge rollups in high-traffic IoT environments. *Sensors*, 2023, vol. 23, no. 7, pp. 3443. https://doi.org/10.3390/s23073443
10. Norvill R., Pontiveros B.B.F., State R., Cullen A. IPFS for reduction of chain size in Ethereum. *Proc. of the IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2018, pp. 1121–1128. https://doi.org/10.1109/cybermatics_2018.2018.00204
11. Breidenbach L., Cachin C., Chan B., Coventry A., Ellis S., Juels A., Koushanfar F., Miller A., Magauran B., Moroz D., Nazarov S., Topliceanu A., Tramer F., Zhang F. *Chainlink 2.0: Next steps in the evolution of decentralized oracle networks*. Chainlink Labs, 2021, 136 p.
12. Ouaddah A. A blockchain based access control framework for the security and privacy of IoT with strong anonymity unlinkability and intractability guarantees. *Advances in Computers*, 2019, vol. 115, pp. 211–258. https://doi.org/10.1016/bs.adcom.2018.11.001
13. Figueroa S., Anorga J., Arrizabalaga S., Irigoyen I., Monterde M. An attribute-based access control using chaincode in RFID systems. *Proc. of the 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2019, pp. 1–5. https://doi.org/10.1109/ntms.2019.8763824
14. Cruz J.P., Kaji Y., Yanai N. RBAC-SC: Role-based access control using smart contract. *IEEE Access*, 2018, vol. 6, pp. 12240–12251. https://doi.org/10.1109/access.2018.2812844
15. Wang S., Zhang Y., Zhang Y. A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. *IEEE Access*, 2018, vol. 6, pp. 38437–38450. https://doi.org/10.1109/access.2018.2851611
16. Khan F., Li H., Zhang L., Shen J. An expressive hidden access policy CP-ABE. *Proc. of the IEEE Second International Conference on Data Science in Cyberspace (DSC)*, 2017, pp. 178–186. https://doi.org/10.1109/dsc.2017.29
17. Xu R., Chen Y., Blasch E., Chen G. BlendCAC: A smart contract enabled decentralized capability-based access control mechanism for the IoT. *Computers*, 2018, vol. 7, no. 3, pp. 39. https://doi.org/10.3390/computers7030039
18. Nishide T., Yoneyama K., Ohta K. Attribute-based encryption with partially hidden encryptor-specified access structures. *Lecture Notes in Computer Science*, 2008, vol. 5037, pp. 111–129. https://doi.org/10.1007/978-3-540-68914-0_7

Литература

1. Goldwasser S., Micali S., Rackoff C. The knowledge complexity of interactive proof-systems // Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali. 2019. P. 203–225. https://doi.org/10.1145/3335741.3335750
2. Chiesa A., Hu Y., Maller M., Mishra P., Vesely N., Ward N. Marlin: Preprocessing zkSNARKs with universal and updatable SRS // Lecture Notes in Computer Science. 2020. V. 12105. P. 738–768. https://doi.org/10.1007/978-3-030-45721-1_26
3. Campanelli M., Gailly N., Gennaro R., Jovanovic P., Mihali M., Thaler J. Linear time prover snarks with constant size proofs and square root size universal setup // Lecture Notes in Computer Science. 2023. V. 14168. P. 331–351. https://doi.org/10.1007/978-3-031-44469-2_17
4. Fuchsbauer G., Orrù M., Seurin Y. Aggregate cash systems: A cryptographic investigation of mimblewimble // Lecture Notes in Computer Science. 2019. V. 11476. P. 657–689. https://doi.org/10.1007/978-3-030-17653-2_22
5. Ozdemir A., Wahby R. Scaling verifiable computation using efficient set accumulators // Proc. of the 29th USENIX Conference Security Symposium. 2020. P. 2075–2092.
6. Xie T., Zhang J., Cheng Z., Zhang F., Zhang Y., Jia Y., Boneh D., Song D. zkbridge: Trustless cross-chain bridges made practical // Proc. of the 2022 ACM SIGSAC Conference on Computer and Communications Security. 2022. P. 3003–3017. https://doi.org/10.1145/3548606.3560652
7. Parno B., Howell J., Gentry C., Raykova M. Pinocchio: Nearly practical verifiable computation // Communications of the ACM. 2016. V. 59. N 2. P. 103–112. https://doi.org/10.1145/2856449
8. Luong D.A., Park J.H. Privacy-preserving blockchain-based healthcare system for IoT devices using ZK-SNARK // IEEE Access. 2022. V. 10. P. 55739–55752. https://doi.org/10.1109/access.2022.3177211
9. Lin X., Zhang Y., Huang C., Xing B., Chen L., Hu D., Chen Y. An access control system based on blockchain with zero-knowledge rollups in high-traffic IoT environments // Sensors. 2023. V. 23. N 7. P. 3443. https://doi.org/10.3390/s23073443
10. Norvill R., Pontiveros B.B.F., State R., Cullen A. IPFS for reduction of chain size in Ethereum // Proc. of the IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). 2018. P. 1121–1128. https://doi.org/10.1109/cybermatics_2018.2018.00204
11. Breidenbach L., Cachin C., Chan B., Coventry A., Ellis S., Juels A., Koushanfar F., Miller A., Magauran B., Moroz D., Nazarov S., Topliceanu A., Tramer F., Zhang F. Chainlink 2.0: Next steps in the evolution of decentralized oracle networks. Chainlink Labs, 2021. 136 p.
12. Ouaddah A. A blockchain based access control framework for the security and privacy of IoT with strong anonymity unlinkability and intractability guarantees // Advances in Computers. 2019. V. 115. P. 211–258. https://doi.org/10.1016/bs.adcom.2018.11.001
13. Figueroa S., Anorga J., Arrizabalaga S., Irigoyen I., Monterde M. An attribute-based access control using chaincode in RFID systems // Proc. of the 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS). 2019. P. 1–5. https://doi.org/10.1109/ntms.2019.8763824
14. Cruz J.P., Kaji Y., Yanai N. RBAC-SC: Role-based access control using smart contract // IEEE Access. 2018. V. 6. P. 12240–12251. https://doi.org/10.1109/access.2018.2812844
15. Wang S., Zhang Y., Zhang Y. A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems // IEEE Access. 2018. V. 6. P. 38437–38450. https://doi.org/10.1109/access.2018.2851611
16. Khan F., Li H., Zhang L., Shen J. An expressive hidden access policy CP-ABE // Proc. of the IEEE Second International Conference on Data Science in Cyberspace (DSC). 2017. P. 178–186. https://doi.org/10.1109/dsc.2017.29
17. Xu R., Chen Y., Blasch E., Chen G. BlendCAC: A smart contract enabled decentralized capability-based access control mechanism for the IoT // Computers. 2018. V. 7. N 3. P. 39. https://doi.org/10.3390/computers7030039
18. Nishide T., Yoneyama K., Ohta K. Attribute-based encryption with partially hidden encryptor-specified access structures // Lecture Notes in Computer Science. 2008. V. 5037. P. 111–129. https://doi.org/10.1007/978-3-540-68914-0_7

19. Liu H., Han D., Li D. Fabric-IoT: A blockchain-based access control system in IoT. *IEEE Access*, 2020, vol. 8, pp. 18207–18218. https://doi.org/10.1109/access.2020.2968492
20. Ding S., Cao J., Li C., Fan K., Li H. A novel attribute-based access control scheme using blockchain for IoT. *IEEE Access*, 2019, vol. 7, pp. 38431–38441. https://doi.org/10.1109/access.2019.2905846
21. Zhou Z., Huang D., Wang Z. Efficient privacy-preserving ciphertext-policy attribute based-encryption and broadcast encryption. *IEEE Transactions on Computers*, 2015, vol. 64, no. 1, pp. 126–138. https://doi.org/10.1109/tc.2013.200
22. Maalla M.A., Bezzateev S.V. An Ethereum based attribute-based access control for IoT. *Proceedings of the Instittute for Systems Analysis Russian Academy of Sciences (ISA RAS)*, 2024, vol. 74, no. 1, pp. 29–34. https://doi.org/10.14357/20790279240104
23. Eberhardt J., Tai S. ZoKrates - scalable privacy-preserving off-chain computations. *Proc. of the IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2018, pp. 1084–1091. https://doi.org/10.1109/cybermatics_2018.2018.00199
24. Baghery K., Pindado Z., Ràfols C. Simulation extractable versions of Groth's ZK-SNARK revisited. *Lecture Notes in Computer Science*, 2020, vol. 12579, pp. 453–461. https://doi.org/10.1007/978-3-030-65411-5_22
25. Baghery K., Kohlweiss M., Siim J., Volkhov M. Another look at extraction and randomization of Groth's ZK-SNARK. *Lecture Notes in Computer Science*, 2021, pp. 457–475. https://doi.org/10.1007/978-3-662-64322-8_22

19. Liu H., Han D., Li D. Fabric-IoT: A blockchain-based access control system in IoT // IEEE Access. 2020. V. 8. P. 18207–18218. https://doi.org/10.1109/access.2020.2968492
20. Ding S., Cao J., Li C., Fan K., Li H. A novel attribute-based access control scheme using blockchain for IoT // IEEE Access. 2019. V. 7. P. 38431–38441. https://doi.org/10.1109/access.2019.2905846
21. Zhou Z., Huang D., Wang Z. Efficient privacy-preserving ciphertext-policy attribute based-encryption and broadcast encryption // IEEE Transactions on Computers. 2015. V. 64. N 1. P. 126–138. https://doi.org/10.1109/tc.2013.200
22. Maalla M.A., Bezzateev S.V. An Ethereum based attribute-based access control for IoT // Труды ИСА РАН. 2024. Т. 74. № 1. С. 29–34. https://doi.org/10.14357/20790279240104
23. Eberhardt J., Tai S. ZoKrates - scalable privacy-preserving off-chain computations // Proc. of the IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). 2018. P. 1084–1091. https://doi.org/10.1109/cybermatics_2018.2018.00199
24. Baghery K., Pindado Z., Ràfols C. Simulation extractable versions of Groth's ZK-SNARK revisited // Lecture Notes in Computer Science. 2020. V. 12579. P. 453–461. https://doi.org/10.1007/978-3-030-65411-5_22
25. Baghery K., Kohlweiss M., Siim J., Volkhov M. Another look at extraction and randomization of Groth's ZK-SNARK // Lecture Notes in Computer Science. 2021. P. 457–475. https://doi.org/10.1007/978-3-662-64322-8_22

Authors

Maher Maalla — PhD Student, ITMO University, Saint Petersburg, 197101, Russian Federation, https://orcid.org/0000-0002-4806-8608, maher.malla7@gmail.com

Sergey V. Bezzateev — D.Sc., Professor, Associate Professor, ITMO University, Saint Petersburg, 197101, Russian Federation; Head of Department, Saint Petersburg State University of Aerospace Instrumentation, Saint Petersburg, 190000, Russian Federation, sc 6602425996, https://orcid.org/0000-0002-0924-6221, bsv@guap.ru

Авторы

Маалла Махер — аспирант, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, https://orcid.org/0000-0002-4806-8608, maher.malla7@gmail.com

Беззатеев Сергей Валентинович — доктор технических наук, доцент, профессор, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация; заведующий кафедрой, Санкт-Петербургский государственный университет аэрокосмического приборостроения, Санкт-Петербург, 190000, Российская Федерация, sc 6602425996, https://orcid.org/0000-0002-0924-6221, bsv@guap.ru

Received 06.05.2024
Approved after reviewing 15.08.2024
Accepted 16.09.2024

Статья поступила в редакцию 06.05.2024
Одобрена после рецензирования 15.08.2024
Принята к печати 16.09.2024