PASSP: A Private Authorization Scheme Oriented Service Providers

Weiqi Dai, Liangliang Yu, Yang Zhou, Kim-Kwang Raymond Choo, *Senior Member, IEEE*, Deqin Zou, Xia Xie, and Hai Jin, *Fellow, IEEE*

***Abstract*—In our data-centric society, major service providers have access to vast amounts of user information (e.g., user-generated content such as social media posts, and device-generated content such as geolocation data) for convenient and efficient services. There are privacy implications when users authorize share personal data managed by service providers. To make authorization private and controllable, in this paper, we propose a private authorization scheme oriented service providers. A decentralized publicly-verifiable re-encryption method based on IPFS is proposed to minimize the reliance on service providers, by shifting to a distributed storage and computation model. Besides, we propose a trustless authorization authentication method that hides the authorization relationship to protect user privacy. We also evaluate the security of our scheme, as well as its performance to demonstrate utility.**

***Index Terms*—Authorization, privacy, blockchain, IPFS, zero-knowledge proof.**

I. Introduction

THERE has been increased focus on user privacy, particularly due to the number of high profile 'scandals' alleging data abuse or leakage [1], [2], [3], [4], [5]. For example, it was alleged that a major social media platform provided personal data of its users to other organizations around the world for several years, according to the New York Times. This topic has also attracted the attention of policy-makers, such as the '*S.1667 - Social Media Privacy Protection and Consumer Rights Act of 2021*' introduced by U.S. Senator Klobuchar in May 2021. However, regulation on its own is insufficient.

Services through direct ciphertext computing (e.g., homomorphic encryption [6], secure multi-party computation [7], and trusted execution environment [8]) may not be practical due to a broad range of reasons, such as computational overheads or user inconvenience (e.g., due to the use of encryption). Even though some practical schemes based on the above technologies [9], [10], [11], [12], [13], [14] have been proposed, most of them have limitations (e.g., apply to narrow scenarios or inability to handle complex services). Therefore, it is very difficult and almost impossible for service providers to provide general services through crypto state calculations, the service provider access to plaintext for service is inevitable. In addition, there are legitimate reasons why service providers need to collect and exchange user information, such as to provide single sign-on [15] and other authentication services for a seamless user online experience. Therefore, we focus on minimizing information available to service providers and dependence on service providers, in order to mitigate any potential risks of data leakage problem due to the indiscriminate collection by service providers.

An important source of data acquisition and analysis for service providers is the use of platform data by users, such as authorization, cross-platform use, etc., thereby generating the derived value of the data. The OAuth 2.0 protocol is widely used by service providers since it allows third-party applications (e.g., data receiver) to obtain specific resources stored by the resource owner (e.g., user) at the service provider (e.g., data source) [16]. This protocol is a centralized authentication protocol that relies on the authenticity of the data source (e.g., service provider). As the only authorized party, service providers of the data source not only participate in authentication and authorization but also control the source and flow of data. However, how do we prevent dishonest or selfish service providers from sharing additional private content without the explicit approval of the user? One can easily observe that this protocol exposes so much user information to service providers that user privacy is reliant on the service providers. Hence, when designing data protection solutions, one must bear in mind that as the sole data storage, authorizer, and authenticator, service providers have an unusually large authority (a case of asymmetry or power imbalance), with implications on data flows, user profiling, and other nefarious activities. This reinforces the importance of placing the topic

Manuscript received 5 January 2024; revised 15 April 2024; accepted 13 June 2024. Date of publication 28 June 2024; date of current version 21 August 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFB1006000 and in part by the National Natural Science Foundation of China under Grant 62072202 and Grant 62362023. The work of Kim-Kwang Raymond Choo was supported only by the Cloud Technology Endowed Professorship. The associate editor coordinating the review of this article and approving it for publication was Y. Liu. *(Corresponding author: Xia Xie.)*

Weiqi Dai, Liangliang Yu, Yang Zhou, and Deqin Zou are with the National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Hubei Key Laboratory of Distributed System Security, Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: wqdai@hust.edu.cn; yull@hust.edu.cn; zhouyang_cse@hust.edu.cn; deqingzou@hust.edu.cn).

Kim-Kwang Raymond Choo is with the Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249 USA (e-mail: raymond.choo@fulbrightmail.org).

Xia Xie is with the School of Computer Science and Technology, Hainan University, Haikou 570100, China (e-mail: shelicy@hainanu.edu.cn).

Hai Jin is with the National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: hjin@hust.edu.cn).

Digital Object Identifier 10.1109/TNSM.2024.3420726

of data security during the authorization process under the microscope.

These privacy problems raised above are inherent in Web 2.0 (the Internet bound by an architecture of oligopolistic entities). Most websites and almost all applications in the Web 2.0 era rely on some form of centralized database to pass data and help enable functionality. Therefore, it is very difficult to solve these problems under the Web 2.0 framework. At this time, Web 3.0 [17], [18], [19] comes into our sight and lights up our ideas. Unfortunately, so far, Web 3.0 is only a concept without a clear definition or complete architecture, so the perception of Web 3.0 is still preliminary. The current consensus on Web 3.0 is that it should be decentralized, open, and privacy-preserving, which is a core attraction. This means that consumers are no longer limited to accessing the Internet through services provided by service providers such as Google, Amazon, Facebook, etc., but rather individuals have more authority and ability to manage their information. This poses a serious challenge to the centralization, surveillance, and exploitative advertising of Web 2.0.

We propose a new private authorization scheme oriented service providers (PASSP) to **make the authorization process private and controllable**. Here, the "private" indicates that extra user privacy (e.g., the user gives which data to which application) is not exposed to service providers during the data authorization process, thereby hiding the user's intention and demand for using data, preventing service providers from analyzing users' preferences and social relationships. The "controllable" means that the authorized content is totally controlled by the user, and the service provider does not participate in the authorization process. Therefore, the service provider cannot send additional unauthorized information mixed with some authorized data or send data with missing content. The solution is able to reflect the key spirit of Web 3.0 well, avoiding the service provider-centered Internet architecture of traditional scenarios.

A. Contributions

In this scheme, we innovatively design a decentralized publicly-verifiable proxy re-encryption method based on InterPlanetary File System (IPFS) to facilitate secure data sharing. In the traditional centralized storage scenario, service providers have absolute control over user data, and any access and use of data are monitored by service providers, which inevitably exposes data transmission paths. Therefore, in our scheme, we utilize IPFS [20], a decentralized storage model (the acquisition and transmission of data without service providers), as the data storage and sharing model to minimize the asymmetry or power imbalance, as well as the impact of an attack when the service providers are compromised.

Not only that, facing the need for disclosure and privacy protection in the data sharing process, we further design the authorization and authentication and proxy re-encryption (PRE) schemes to meet the demand. Users are free to authorize data without disclosing privacy to service providers, unlike traditional authorization and authentication schemes that rely on the services of trusted third parties. Blockchain, known for its underlying decentralization and trustless [21], [22], can provide us with an open and transparent platform to record the authorization process and support payments. We focus on zero-knowledge proof that make it impossible for service providers to infer authorization content and other information about the object from the chain. And proxy re-encryption can adequately handle "one-to-many" sharing scenarios of data. We combine IPFS and PRE to fully utilize their advantages (turning the Internet into a database) for data storage and sharing.

A summary of our contribution is as follows:

- We propose a decentralized publicly verifiable proxy re-encryption method based on IPFS. In this method, distributed IPFS nodes serve as storage and computing nodes, which not only eliminates the dependence on the semi-trusted third party in the re-encryption algorithm but also weakens the (excessive) centralized power of service providers. The computation results can also be verified publicly to ensure data integrity and correctness.
- We propose a trustless authorization authentication method that hides the authorization relationship. In order to conceal the user's identity and separate the data address from the data authorization, we adopt zero-knowledge proof to authenticate authorization relationships. Authentication can be successfully performed without leaking any additional valuable information about the authorizer.
- We propose a private authorization scheme-oriented service providers (PASSP) to solve the privacy problem under centralization and lay the foundation for Web 3.0 to realize data privatization and private authorization. During the authorization process, control of licensed content flows from service providers to users; thus, allowing users to control the authorization. In addition, data content, data flow, and user behavior can be hidden. Hence, data use is shielded from service providers' prying eyes.

Our scheme provides practical solutions for Web 3.0 scenarios that address privacy and security issues in the current centralized environment.

B. Outline

Section II introduces related background material. In Section III, we present the system and threat models. Then, we introduce our proposed PASSP scheme in Section IV, before evaluating its security in Section V. In Section VI, we simulate the performance of the PASSP prototype in the scenario of image licensing editing and evaluate its performance. In our scheme, data authorization is not bound to the authorized object (supporting prior authorization and 'one authorization, multiple uses'); thus, allowing for data authorization and authorization authentication to be performed at intervals. And beyond that, in the context of Web 3.0 we do not have to consider block confirmation time. As for on-chain verification time, as shown in Section VII, the zero-knowledge proof verification time is less than 1.5s, and the re-encryption verification time is in the order of milliseconds, which is similar to existing current third-party authorization confirmation performance. Although we use Ethereum in our experiment, our scheme can also be implemented on private

TABLE I
SUMMARY OF NOTATION

Notation	Description
λ	Security parameters for cryptographic algorithms
k	The key used for symmetric encryption
m	User data to be encrypted
pk, sk	Public-private key pairs for proxy re-encryption algorithms
rk	Re-encryption key for proxy re-encryption algorithm
$\boldsymbol{c}_a$	Ciphertext obtained by encrypting with public key pk_a
$\boldsymbol{c}'_s$	The ciphertext obtained by re-encrypting $\boldsymbol{c}_a$ using $rk_{a,s}$ can be decrypted using the private key sk_s
$\mathcal{C}$	Results of cryptographic commitment
$addr_{\text{data}}$	Address where data is stored in IPFS
$\mathcal{A}$	Adversaries that seek to undermine the security and privacy of the scheme

blockchain, consortium blockchain, or other efficient and low-cost blockchain; thus, allowing transaction fees. Section VIII introduces the related approaches in the literature, before concluding this work in the last section.

II. Background

This section describes the notation used in this paper as well as the various techniques that support the proposed scheme, i.e., proxy re-encryption and zero-knowledge proof. Notation is summarized in Table I.

A. Proxy Re-Encryption

Proxy re-encryption [23] converts ciphertext uploaded by some user, say Alice, with her own public key into another form of ciphertext. This algorithm allows another user, say Bob, to decrypt the converted ciphertext with his own private key. This eliminates the possibility of leaking any corresponding plaintext information and data decryption keys during the entire conversion process. Unlike encryption schemes, for example, attribute-based encryption (ABE), PRE requires Alice's data to be obtained through the 'request + confirmation' method. This means that Bob must apply to Alice every time Bob wants to get Alice's data, and Bob can receive and view the data only after Alice agrees.

We draw on the definitions of proxy re-encryption schemes [24], [25], [26], [27] and the need to perform public verification, in this paper, the publicly verifiable proxy re-encryption used needs to contain the following algorithms and fulfill the corresponding requirements.

- $KeyGen(\lambda) \rightarrow (sk_x, pk_x)$ Each time the security parameter λ is entered, a pair of public and private keys can be generated for encryption and decryption. If the subscripts are the same, the public and private keys are correspondences, i.e., the ciphertext encrypted by public key pk_x can be decrypted by private key sk_x, otherwise not.
- $Enc(pk_a, m) \rightarrow \boldsymbol{c}_a$ Input the public key pk_a and the plaintext m, a ciphertext can be generated, denoted as $\boldsymbol{c}_a$, which represents the ciphertext without re-encryption, and the decrypting private key is sk_a.
- $RkGen(sk_a, pk_s) \rightarrow rk_{a,s}$ Enter private key sk_a and public key pk_s to generate a re-encrypted key $rk_{a,s}$.

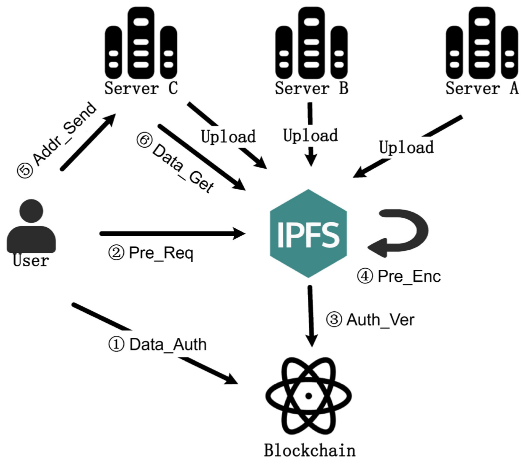


Fig. 1. The system model.

- $ReEnc(rk_{a,s}, \boldsymbol{c}_a) \rightarrow \boldsymbol{c}'_s$ Input re-encrypted key $rk_{a,s}$, ciphertext $\boldsymbol{c}_a$ encrypted by public key pk_a. Output ciphertext $\boldsymbol{c}'_s$ that can be decrypted by private key sk_s.
- $VerRe(\boldsymbol{c}'_s, pk_s) \rightarrow \{0, 1\}$ Input the re-encrypted ciphertext $\boldsymbol{c}'_s$ and public key pk_s to verify that the ciphertext has been correctly re-encrypted.
- $Dec(\boldsymbol{c}, sk) \rightarrow m$ Enter the ciphertext $\boldsymbol{c}$ or $\boldsymbol{c}'$ and the corresponding private key sk to get the plaintext m.

B. Zero-Knowledge Proof

The zero-knowledge proof [28] allows the verifier to determine whether the prover's statement is correct (e.g., knowing a key piece of information) without revealing the information itself. A zero-knowledge proof system should satisfy completeness (i.e., the prover can convince the verifier that a statement is correct), reliability (i.e., if the statement is false, the cheating prover cannot convince the verifier), and zero-knowledge (i.e., protocol interactions only reveal whether a statement is correct and do not reveal any other information).

We adopt the widely used zero-knowledge succinct non-interactive argument of knowledge (zk-SNARKs) [29] in our work.

III. System and Threat Models

A. System Model

As shown in Fig. 1, our scheme includes four roles, namely: server, user, IPFS node, and blockchain node.

- *Server:* Responsible for providing the required services to the users, in this paper, one type of service provider uses IPFS to provide data storage services to the users, and the other type of service provider needs the user's data to provide other services. Upon initialization of the system, each server will be assigned a pair of public and private keys pk_s, sk_s. For user's data $m_1, m_2, \ldots,$, the service provider encrypts them with different symmetric keys $k_1, k_2, \ldots$, and the resulting ciphertext $[m_j]_{k_j}$ then needs to be uploaded to IPFS. $[m_j]_{k_j}$ denotes the ciphertext obtained by encrypting data m_j with a secure symmetric encryption algorithm and key k_j, abbreviated as $[m]$.

At the same time, the server sends $k_1, k_2, \ldots,$ to user through a secure channel to facilitate the user's later authorization. For business reputation considerations, all service providers are honest and curious. This paper does not consider data leakage between service providers outside the system.

- *User:* The owner of the data resource, who can authorize access and use of the data to other entities. Unlike the server, to prevent the public key from being bound to the user's identity and thus revealing the user's identity, each user can use a different public and private key pair (pk_u, sk_u) for each encryption, which can be deleted as soon as it is used (with no additional key management cost). After obtaining the symmetric keys, the user encrypts the symmetric key encryption using their own public key and stores it on the IPFS. When the user wants to authorize the data, he/she sends a re-encryption request to the IPFS to convert the ciphertext of the symmetric key into a ciphertext that can be decrypted by the authorized party. In the above process, the user does not want to disclose his/her identity privacy or data content.
- *IPFS node:* IPFS node is the node in the decentralized storage network that may store part of the user data. After getting the proxy re-encryption request from a user, the node needs to verify the request and calculate the data. Clearly, only the node that provides the correct calculation result will be allowed to access income. Some IPFS nodes attempt to give incorrect computation results, break the confidentiality of data, or correlate the identity of data authorizers.
- *Blockchain:* In our system, the blockchain network can record all types of transactions, including special transactions proposed in the scheme (e.g., authorization transactions, zero-knowledge proof verification transactions, etc.). In addition to this, it also provides payment service functions. The blockchain network is considered to be secure and reliable and attacks on it are beyond the scope of this paper.

In the Web 3.0 scenario, users have more freedom to control their own data and decide which service provider to authorize in order to get the expected service. Even in a public environment, no one can be informed of who authorized the data and who was authorized, and user privacy can be fully guaranteed. We will now use the following simple example to explain the system model. Let us assume a scenario where Alice, a user, wants to use the service of Clipchamp (video editor and maker software) to edit some pictures and videos on Google Photos. Hence, Alice must authorize part of the data in Google Photos to Clipchamp. In this process, Alice does not want Google Photos (or other unrelated stakeholders) to know which part of the photos he uses and who is authorized to access the photos. By default the user's data has been uploaded by Google Photo for encrypted storage on IPFS and the user owns the address where the data and encryption key are stored. The work-flow is as follows:

- Data_Auth: Alice issues an authorization transaction TX_{auth} to authorize the relevant data $m_1, m_2, \ldots,$ (photos in our context), which is attributed to Google Photos, on the blockchain.
- Pre_Req: Alice sends a proxy re-encryption request (containing zero-knowledge proof) to the IPFS network, puts the relevant records on the chain, and pays the appropriate fees.
- Auth_Ver: After getting the proof provided by Alice, the selected idle IPFS node calls the zero-knowledge verification contract to perform authorization verification on the blockchain.
- Pre_Enc: Only if the verification passes, the node will calculate the proxy re-encryption request of Alice. After the calculation, the node needs to invoke the re-encryption verification contract to verify whether the calculation results are correct. Only if the verified results are recognized, the node will upload the corresponding results to IPFS and obtain the corresponding benefits. Otherwise, the calculation will not be accepted, and the node will fail to obtain the benefits.
- Addr_Send: After obtaining the re-encrypted key address, Alice would provide the new key address and data address to Clipchamp, the video editor.
- Data_Get: Clipchamp first retrieves the encrypted key and data from IPFS based on the address, then uses his own private key to obtain the symmetric key and then successfully decrypt to obtain data $m_1, m_2, \ldots,$ (previously belonged to Google Photos).

B. Key Objectives and Security Requirements

Taking into account the concerns about user privacy authorization mechanisms discussed in the previous section, this paper proposes the following three key objectives and security requirements:

- Secure sharing of data with fine-grained authorization. For the authorization system, it is important to ensure that the whole authorization scheme can be executed correctly and securely. That is to say, when users use the authorization system, they can be assured that authorization can be performed and the authorized party can securely get the complete data that the user wants to share. Moreover, users can authorize data at a fine-grained level to avoid unnecessary data being shared incidentally and minimize the possibility of data leakage. During the whole authorization process, the adversary cannot get any content of the authorized data.
- Authorization and re-encryption with public verifiability. Key stages of the authorization process must be publicly verifiable to ensure that the participants have faithfully executed the pre-defined scheme and that the entire authorization process is correct. The recipient of the data needs to be able to successfully receive the correct and complete data, which must be consistent with the authorization granted by the authorizing party. The authorized content is completely under the control of the authorized party and no adversary or participant can send error messages that deviate from the protocol. Correct execution of the proxy re-encryption operation is incentivized.
- Enabling privacy of authorizer's identity. In the authorization process, a large amount of information is publicly available, and it is necessary to hide the authorizer's

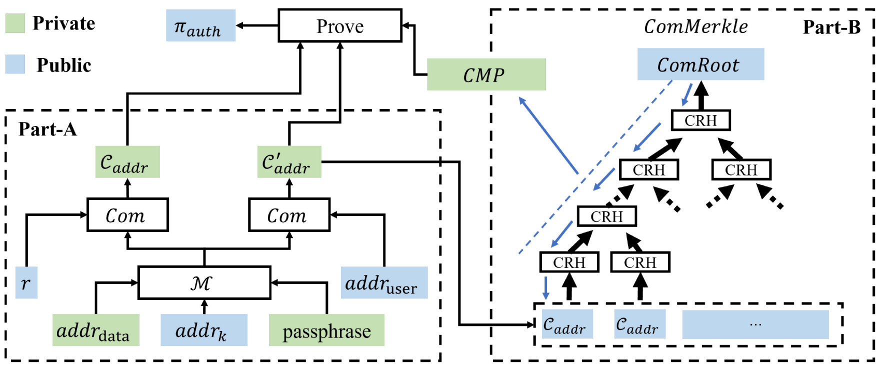


Fig. 2. The construction of authorization proof π_{auth}.

identity and separate the authorization behavior from the authorizer's identity (address) so that the adversary or participant cannot obtain the data content and data flow through analysis. In other words, the adversary or participant cannot obtain additional user privacy by monitoring the user's behavior, thus hiding the user's real intention and need to use the data, and preventing the adversary or participant from analyzing and obtaining the user's favorite preferences and social relationships.

IV. Proposed PASSP Scheme

We will now discuss the two core functions of the PASSP scheme, namely: authorization & authentication, publicly-verifiable re-encryption.

A. Authorization and Authentication

This core function is used for data authorization and post-authorization authentication; thus, ensuring data privacy throughout the process. This function also facilitates the hiding of authorization relationships, without revealing the data flows and authorized content.

Authorization: Here, We will discuss how to construct an authorization transaction, including the composition and data structure of the commitment in this transaction. In the Data_Auth phase, if Alice wants to authorize the data, she must issue an authorization transaction TX_{auth} on the blockchain, which contains the cryptographic commitment $\mathcal{C} = Com(r; \mathcal{M})$ used to hide the authorized information. Commitment schemes can issue a commitment $\mathcal{C}$ about message $\mathcal{M}$ without revealing any information about $\mathcal{M}$, thus, allowing one to prove that $\mathcal{C}$ is a commitment bound to the information $\mathcal{M}$ by opening the trapdoor r. In IPFS, one uses the encrypted hash of the content to identify the content. Users with the hash address of the data can access the data, and hence access permission of the data can only be controlled by hiding the data addresses and encrypting the data. Therefore, we use data $[m]$ address, key address (symmetric encryption key k), and authorization passphrase (pp) as hidden information to construct a commitment scheme $\mathcal{C}_{addr} = Com(r; addr_{\text{data}}, addr_k, pp)$, and use public $\mathcal{C}_{addr}$ to prove the user's authorization for the data without revealing the address of the data. At the same time, in order to facilitate the subsequent proof of the existence of the $\mathcal{C}_{addr}$ without exposing $\mathcal{C}_{addr}$, we need to maintain a global commitment tree to store the value of $\mathcal{C}_{addr}$. It can be stored in the form of a Merkle tree, recorded as *ComMerkle*, and the root of the tree is *ComRoot*. Consequently, in proving that a certain leaf node $\mathcal{C}_{addr}$ exists in *ComMerkle*, only the path from the leaf node to the root node needs to be provided. As is shown in Fig. 2 Part-B, the complexity is $\mathcal{O}(\log(n))$, where n is the number of leaf node $\mathcal{C}_{addr}$.

Authentication: The following part will introduce the circuit structure of zero-knowledge proof for authentication, including the $\mathcal{NP}$ statement construction, the proof generation process, and the proof verification step. When Alice wants to share (use) data after issuing an authorization transaction, she must generate zero-knowledge proof of authorization (recorded as π_{auth}) locally, which is used to prove that the data has been authorized. First of all, Alice must generate $\mathcal{C}_{addr}$ corresponding to the authorized address to prove that it knows the related data address, key address, and authorization passphrase. In the meantime, for the sake of preventing malicious users from stealing others' π_{auth} and using replay attacks to pass authentication, the proof also needs to be bound to the address used by Alice at this time. In other words, Alice needs to construct another commitment condition $\mathcal{C}'_{addr} = Com(addr_{\text{user}}; addr_{\text{data}}, addr_k, pp)$, using $addr_{\text{user}}$ as a trapdoor for this commitment condition. The whole structure is presented in Fig. 2 Part-A.

But these conditions alone are not enough. When Alice makes a proxy re-encryption request to the IPFS network and provides π_{auth} to the IPFS node, the related node will authenticate whether the proof π_{auth} is legitimate from the blockchain. In this process, the key address needs to be exposed. However, we seek to hide $\mathcal{C}_{addr}$ and $\mathcal{C}'_{addr}$ to minimize the risk of exposing the relationship with authorization. Therefore, in order to ensure the legitimacy of the commitment, the condition that $\mathcal{C}_{addr}$ must exist in the *ComMerkle* needs to be added to π_{auth}. As is shown in Fig. 2 Part-B, it means that Alice needs to generate the corresponding proof *commit merkle proof(CMP)* as the private input using $\mathcal{C}_{addr}$ and *ComRoot*, when constructing π_{auth}. All in all, the π_{auth} provided by Alice must prove the following $\mathcal{NP}$ statement $\mathcal{NP}_{auth}$:

For public inputs ($ComRoot$, $addr_k$, $addr_{\text{user}}$, r) and private inputs ($\mathcal{C}_{addr}$, $\mathcal{C}'_{addr}$, $addr_{\text{data}}$, pp, CMP), the following $\mathcal{NP}$ statement is established:

- $\mathcal{C}_{addr} = Com(r; addr_{\text{data}}, addr_k, pp)$
- $\mathcal{C}'_{addr} = Com(addr_{\text{user}}; addr_{\text{data}}, addr_k, pp)$
- The commit $\mathcal{C}_{addr}$ must be a leaf in the $ComMerkle$ which root is called $ComRoot$

Finally, the IPFS node that receives the request will use the proof π_{auth} provided by the user and some public parameters (e.g., common reference string, public inputs, etc.) to verify the authorization to the blockchain network. If the verification fails, the proxy re-encryption calculation will not be performed by the node. Only upon successful authorization verification will the corresponding node perform the re-encryption calculation. Once the re-encryption calculation has been completed, the node uploads the calculation result to IPFS and calls the re-encryption verification contract to verify the correctness of the calculation (this will be described in detail in the next section). When the verification is successful, the node will obtain the calculation revenue, and consequently, the user can obtain the correct re-encrypted data address.

B. Publicly-Verifiable Re-Encryption

We will introduce the design ideas and system flows of PRE in this part. In the beginning, the user Alice has generated a pair of public and private keys pk_a, sk_a. The symmetric key k (corresponding to the ciphertext of the message) has been encrypted using pk_a through the $Enc(pk_a, k)$ and uploaded to IPFS. When Alice wants to share data, the IPFS node can receive the re-encryption key $rk_{a,s} = RkGen(sk_a, pk_s)$ calculated by Alice during Pre_Req phase. Therefore, after authentication, the proxy re-encryption calculation of k's ciphertext $\boldsymbol{c}_a$ will be performed directly on the IPFS node. This calculation is designed to be embedded in the underlying function of IPFS. On the basis of not damaging the underlying storage network of IPFS, the *ReEnc* algorithm will be appended, and the requirement for a semi-trusted third-party proxy in the original re-encryption service is removed. So that the calculation would be performed directly on IPFS nodes, no longer dependent on the user's computing power. After the calculation is completed, the IPFS node gets a new ciphertext $\boldsymbol{c}'_s$ and needs to call the re-encryption verification contract to verify the correctness and integrity of the calculation to the blockchain network through *VerRe* algorithm. If the verification is successful, Alice will get the address of the new ciphertext $\boldsymbol{c}'_s$. So that Alice can send both data and key address to the service provider. Next, the service provider obtains ciphertexts through IPFS and decrypts them by its private key to view the corresponding authorization data through the $Dec(\boldsymbol{c}'_s, sk_s)$ algorithm. The overall proxy re-encryption process is shown in Fig. 3.

During the Pre_Enc phase, the IPFS node to do the re-encryption calculation can be selected by users themselves or randomly, because the choice of IPFS node does not have security risks for our scheme. The security of IPFS nodes is guaranteed in many aspects of our scheme. In the authorization

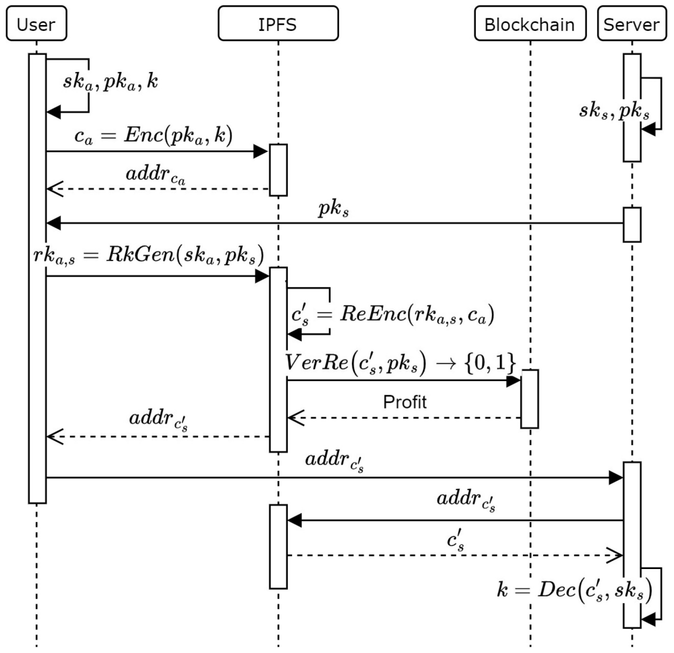


Fig. 3. Sequence diagram of proxy re-encryption process.

verification process, the authorization relationship is hidden from IPFS nodes (concealing Commit through zero-knowledge proof). In the verification process, it is requested to verify the correctness of the calculation results provided by IPFS nodes, which prevents IPFS nodes from doing evil by themselves. Besides, it makes no sense for the user who initiates the request to collaborate with the IPFS node to commit evil acts, since the data itself is the user's own. so There is no point in spending to gain his own privacy.

Next, we will introduce the concrete step realization of each algorithm. For re-encryption calculations, the IPFS node uses the re-encryption key $rk_{a,s}$, provided by Alice, to convert the original ciphertext $\boldsymbol{c}_a$ into ciphertext $\boldsymbol{c}'_s$. However, for the sake of meeting the requirements of verifiable calculation, simple ciphertext conversion can not satisfy our requirements. Therefore, we utilize the idea of signcryption [30] by embedding the signature information in the ciphertext. As shown in the Appendix Algorithm 1 is the specific implementation of *Enc*.

We utilize and modify the inspiration in the scheme [24], [25], [26], so that the unforgeability of the calculation results can be realized algorithmically. We use *ReEnc* algorithm to convert ciphertext $\boldsymbol{c}_a$ about Alice into the ciphertext $\boldsymbol{c}'_s$ about sever, $\boldsymbol{c}'_s = ReEnc(\boldsymbol{c}_a, rk_{a,s})$, which is shown in the Appendix Algorithm 2.

For converted ciphertext $\boldsymbol{c}'$, it will be verified in the verification smart contract to determine whether it is the correct ciphertext. Clearly, the correctness of the re-encryption process can be publicly verified due to the nature of the signature in $\boldsymbol{c}'$. This process is implemented in the *VerRe* algorithm, as shown in the Appendix Algorithm 3.

At last, after receiving the re-encrypted ciphertext $\boldsymbol{c}'_s$, the service provider uses its private key to decrypt it and obtains the symmetric key k'. In addition, in order to ensure the integrity and security of ciphertext during transmission, the

Dec algorithm will also re-verify $\boldsymbol{c}'_s$, which is the same as *VerRe*. Only if the verification is passed, the service provider can trust and obtain k' by decryption. The implementation of the *Dec* algorithm is presented in Appendix Algorithm 4.

V. Security Analysis

We will now analyze the security of the proposed scheme (for the security requirements described in Section III-B). Specifically, the scheme must satisfy the following properties: Fine-Grained Access Control, Authentication Non-malleability, Calculation Verifiability, and Privacy of the authorizer's identity. The security proofs are formulated as games played between adversary $\mathcal{A}$ and challenger. The definition of PASSP scheme is ζ : $(\mathsf{Data\_Auth}, \mathsf{Pre\_Req}, \mathsf{Auth\_Ver}, \mathsf{Pre\_Enc}, \mathsf{Addr\_Send}, \mathsf{Data\_Get})$.

A. Fine-Grained Access Control

First, the user's data is split according to certain rules and encrypted using a symmetric encryption algorithm. Each part of the data m_i uses a corresponding key k_i. $\{k_1, k_2, \ldots, \}$ are independent of each other. The encrypted data is stored on IPFS, and it is clear that without key *k*, any probabilistic polynomial time adversary $\mathcal{A}$ has a negligible probability of breaking any ciphertext. The key *k* is encrypted by the user using the public key and stored on IPFS as well, and due to the security of PRE, both the ciphertext after the first encryption and the ciphertext obtained after re-encryption have confidentiality. In addition, the re-encryption process is still publicly verifiable and IPFS nodes outputting wrong results will be detected. In order to achieve fine-grained sharing of data m_i, the user only needs to send a request to the IPFS to re-encrypt the key k_i corresponding to $[m_i]_{k_i}$, provided that the visitor has passed the user's authorization.

B. Authentication Non-Malleability

For authentication non-malleability, we need to ensure that no adversary $\mathcal{A}$ can revamp any information within the transaction of *Verify*. Authentication non-malleability prevents adversaries from modifying others' successful transactions or proof to pass verification.

Authentication non-malleability can be characterized by an experiment named A-NM, where $\mathcal{A}$ can interact with the PASSP scheme oracle. In A-NM experiment, $\mathcal{A}$ can construct a tx' to win the game, only if there exists $tx \in \{TX_{Verify}\}$ and satisfies the following conditions: (i) $tx' \neq tx$, (ii) tx' includes the same commitment content within tx, (iii) both tx' and *tx* are valid. In a world, we say ζ is A-NM secure, only if for every adversary $\mathcal{A}$ in poly(λ)-size and sufficiently large λ,

$$Adv_{\zeta,\mathcal{A}}^{\mathrm{A-NM}}(\lambda) \leq negl(\lambda)$$

where $Adv_{\zeta,\mathcal{A}}^{\mathrm{A-NM}}(\lambda) := Pr[\mathrm{A-NM}(\zeta, \mathcal{A}, \lambda) = 1]$ means $\mathcal{A}$'s advantage in the A-NM experiment.

As described above, we are going to analyze possible events that lead to $\mathcal{A}$ winning.

$Event_{\pi_{auth}}$: $\mathcal{A}$ wins A-NM, and there exists π'_{auth}, $Verify(\pi'_{auth}, *) = ture$. $*$ is the other public input within *Verify*.

$Event_{Verify}$: $\mathcal{A}$ wins A-NM, and there exists $\mathcal{C}_{addr} \in ComMerkle$, such that (i) $addr \neq addr'$, (ii) $\mathcal{C}_{addr}$ is the commitment of *addr*, $\mathcal{C}_{addr'}$ is the commitment of $addr'$, (iii) $\mathcal{C}_{addr} = \mathcal{C}_{addr'}$.

Therefore, $\mathcal{A}'s$ advantage in the A-NM experiment is

$$Adv_{\zeta,\mathcal{A}}^{\mathrm{A-NM}}(\lambda) = Pr[Event_{\pi_{auth}}] + Pr\left[Event_{Verify}\right]$$

So as to prove $Adv_{\zeta,\mathcal{A}}^{\mathrm{A-NM}}(\lambda)$ is negligible in λ, it is worth discussing each event is almost negligible in λ.

Bounding the probability of $Event_{\pi_{auth}}$: During $Event_{\pi_{auth}}$, $\mathcal{A}$ can construct an invalid π'_{auth}, and prove $Verify(\pi'_{auth}, *) = true$. In the process, $\mathcal{A}$ uses the trapdoor of zk-Proof to attack zero-knowledge algorithm. However, the zk-SNARKs used in the scheme have been shown to be secure and the probability of breaking the algorithm is negligible.

Bounding the probability of $Event_{Verify}$: During $Event_{Verify}$, $\mathcal{A}$ can find $\mathcal{C}_{addr} \in ComMerkle$ and use $addr'$ to construct $\mathcal{C}_{addr'}$, which equals to $\mathcal{C}_{addr}$. If $\mathcal{A}$ can succeed, this suggests that collisions for collision-resistant hashing function (CRH) are found. Due to CRH's collision resistant, the probability of $Event_{Verify}$ must be negligible in λ.

All in all, $Adv_{\zeta,\mathcal{A}}^{\mathrm{A-NM}}(\lambda)$ is almost negligible in λ and the PASSP scheme satisfies authentication non-malleability.

C. Calculation Verifiability

For calculation verifiability, we define that $\mathcal{A}$ cannot profit from the proposed scheme by providing incorrect calculation results. The calculation result provided by $\mathcal{A}$ must pass the public verification of the re-encryption verification contract.

Calculation Verifiability is characterized by an experiment C-VER. In the C-VER experiment, first, challenger initializes global parameters and sends them to $\mathcal{A}$. Then $\mathcal{A}$ initiates a series of inquiries above and simulates that he has got $\boldsymbol{c}$ and re-encrypted it. Last, $\mathcal{A}$ constructs the ciphertext $\boldsymbol{c}'$ as calculation result which is not re-encrypted through *ReEnc* algorithm. If $\boldsymbol{c}'$ can pass the *Ver* algorithm in verification contact and get return value, $\mathcal{A}$ wins the game. We define that ζ is C-VER secure, only if for every adversary $\mathcal{A}$ in poly(λ)-size and sufficiently large λ,

$$Adv_{\zeta,\mathcal{A}}^{\mathrm{C-VER}}(\lambda) \leq negl(\lambda)$$

where $Adv_{\zeta,\mathcal{A}}^{\mathrm{C-VER}}(\lambda) := Pr[\mathrm{C-VER}(\zeta, \mathcal{A}, \lambda) = 1]$ means $\mathcal{A}'s$ advantage in C-VER experiment.

Next, we are going to analyze the possibility of $\mathcal{A}$ winning the game. The *ReEnc* algorithm used in the scheme has the unforgeability of converted ciphertext which has been verified, and in the *Ver* algorithm, $\boldsymbol{c}'$ can also be verified publicly. So we can prove mathematically that $\mathcal{A}$ is scarcely possible to forge $\boldsymbol{c}'$ to pass the verification. That is to say, the probability of $\mathcal{A}$ to win the game is almost negligible in λ. Therefore, $Adv_{\zeta,\mathcal{A}}^{\mathrm{C-VER}}(\lambda)$ is almost negligible in λ and the PASSP scheme satisfies Calculation Verifiability.

D. Privacy of Authorizer's Identity

In a data authorization process, the user achieves authorization using cryptographic commitments and zero-knowledge proofs techniques. $addr_{\text{data}}$ and *CMP* are used as private inputs that are not accessible to anyone. That is, although the final authorization proof π is publicly verifiable on the blockchain, no one can associate π with $addr_{\text{data}}$. Similarly, *ComMerkle* can be publicly verified, but there is no access to the specific $\mathcal{C}_{addr}$. Due to the nature of the re-encryption algorithm, either the ciphertext $\boldsymbol{c}$ or *rk* is observed and can be considered as independent random numbers. All publicly available information cannot be associated with the user. In the case of multiple authorizations by the user, the privacy of the authorizer's identity is guaranteed, on the one hand, by the fact that the user can re-select his public key, and on the other hand, due to the random nature of the cryptographic commitments and the zero-knowledge proofs described above (even if the same authorization is executed twice, the resulting π_1 and π_2 are different).

VI. Implementation

Here, we will introduce how to implement the whole PASSP scheme, including the implementation and combination of several important models. First of all, in order to be more representative, we adopt Ethereum as our public chain choice. Besides, our implementation uses the client source code implemented in the Go language. In our experiment, the Ethereum version is geth-1.9.14 and the IPFS version is ipfs-0.8.0.

A. Support Zk-SNARKs and Re-Encryption Verification in Ethereum

In order to implement the PASSP scheme described above, we need to expand the ethereum virtual machine (EVM) to support zero-knowledge proof and re-encryption verification. Since the verification of zero-knowledge proof and re-encryption involves a lot of elliptic curves and bilinear mapping calculations, it is not satisfactory to directly use the EVM instruction set to implement zero-knowledge or re-encryption verification, which will produce a lot of consumption. Therefore, here we use pre-compiled contracts to expand EVM to support zero-knowledge and re-encryption verification. Some pre-compiled contracts have been built into Ethereum to support the normal execution of EVM, with excellent performance and transaction flexibility, so we only need to edit zero-knowledge and re-encryption verification pre-compiled contracts on the original basis.

B. Support Proxy Re-Encryption in IPFS

Considering that users may use light nodes or other nodes that can not provide adequate computing power, we put the re-encryption computing on IPFS nodes to achieve distributed proxy re-encryption. We modify the most mature and popular go-ipfs (implemented in the go language). In go-ipfs, the command *get*, which is used to obtain data in the storage network, is modified to add the PRE function. The *ipfs*, *get* command downloads the IPFS/IPNS object data of the specified hash path to the disk. In this command, we add the $-r$ option to determine whether re-encryption is required. When the user needs to perform a re-encryption calculation, he can select the re-encryption option.

C. Zk-SNARKs Implementation

For zk-SNARKs, its execution performance is severely restricted by the proof generation which depends on the implementation. The low system performance is unacceptable in our scheme. To improve the system performance as much as possible, we use Rust, an efficient and reliable language, to implement zk-SNARKs. In the process of implementing this algorithm, some of the basic functionalities need to be realized by the Bellman library [31], providing some circuit traits, primitive structures, and basic rank-1 constraint system (R1CS) module support. We need to obtain first-order constraints by analyzing $\mathcal{NP}$ statements and type these constraints using the interfaces provided in Bellman to generate quadratic arithmetic program (QAP).

D. Proxy Re-Encryption Implementation

For the sake of ensuring the algorithm's security, a proxy re-encryption algorithm based on bilinear pairing is adopted in our scheme. The computational efficiency of the bilinear curve will seriously affect the overhead of the re-encryption algorithm. Our scheme relies on the PBC Library [32], which includes various optimizations to make pairing-based cryptography very efficient. For public verification of re-encryption, we borrowed ideas from the schemes of other researchers about signcryption with proxy re-encryption. Using the idea of the signcryption scheme, we change the ciphertext into a signcryption and verify the correctness of re-encryption calculation by using the parameters in signcryption, which has been described detailedly in Section IV-B.

E. Overall System Implementation

Fig. 4 is the overall implementation framework of the system. In order to connect each module of our scheme in tandem, we use the Python library as the module controller and package the relevant Python code into an API as the access interface. Therefore, our authorization protocol can be easily applied to other projects. We utilize the Web3 interface to communicate with geth nodes, including account management, transaction publishment, and contract deployment. To have access to the IPFS network service, ipfshttpclient (an API library using Python to connect IPFS clients) can be used. In the zero-knowledge proof library, we provide the relevant library interface to acquire the R1CS instance and access the zero-knowledge public parameters to generate the proof. At the same time, interfaces of proxy re-encryption are provided to achieve related functional requirements in the PRE library. As a result, the geth node, receiving transactions sent by users and maintaining the global ledger and blockchain state, directly invokes the pre-compiled contracts for zero-knowledge and re-encryption verification. Among them, the zero-knowledge verification contract calls the compiled function in the zero-knowledge library through CGO. The re-encryption verification contract uses the function in the PRE library to verify the calculated results. For IPFS nodes, the node needs to receive the re-encryption request sent by the user and call the contract to verify the legitimacy of the authorization. Besides, the node can obtain benefits by

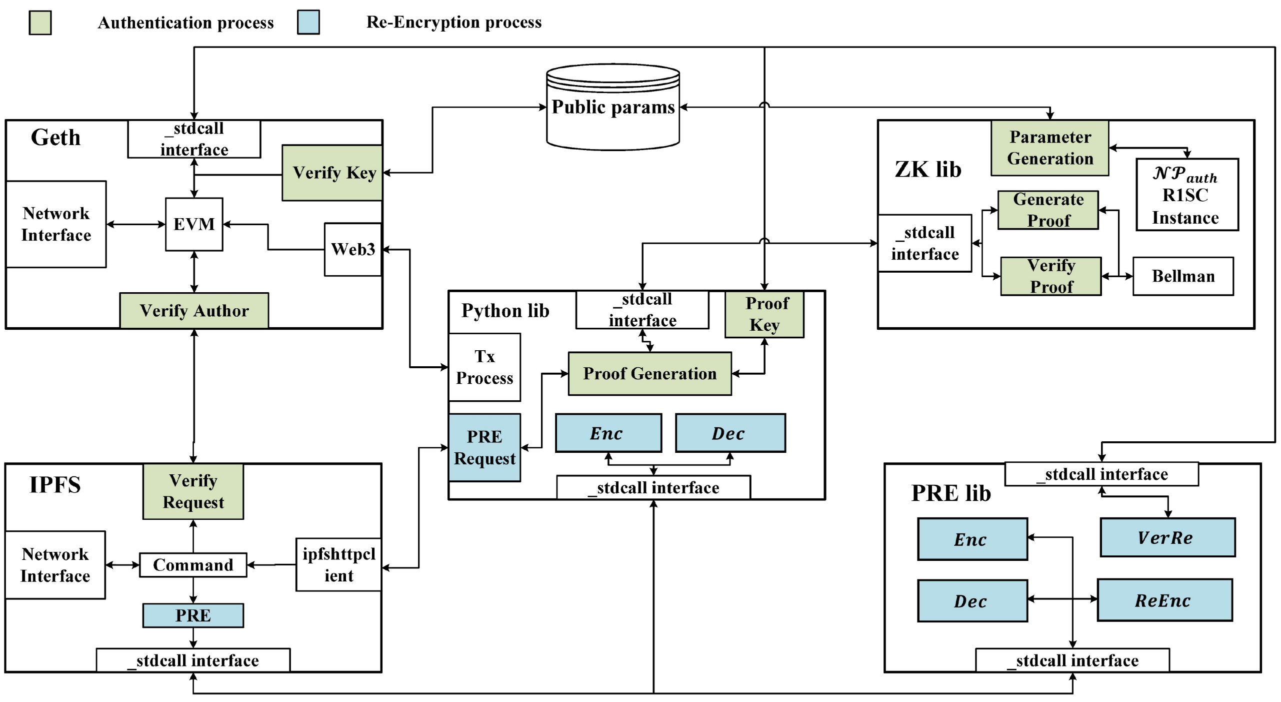


Fig. 4. The overall implementation framework of PASSP scheme.

invoking contracts to verify the correctness of the calculated results.

VII. Evaluation

According to the above description, we apply our PASSP scheme to the scenario of image licensing editing mentioned in Section III. We simulate Clipchamp and Google Photos with two simple open-source projects the photo album and the photo editor, respectively. Then, we use our scheme as the authorization protocol to realize the authorization, authentication, and sharing of images. On this basis, we deployed our prototype on the server in the lab which configuration is Intel Xeon E-2630@2.3GHz CPU, 64GB memory. Next, we conducted several experiments, each executed 500 times taking the average to measure the performance of the system, and the results are as follows.

A. Overhead of Zero-Knowledge Proof

A zero-knowledge proof algorithm can be mainly divided into three steps: `Setup`, `Prove`, and `Verify`. For the `Setup` algorithm, we focus on the size of the proof key *PK*, verification key *VK* (that is, storage cost *S*), and time cost *T*. The key we care about in the `Prove` algorithm is the memory space of the proof π_{auth} and the generation time of π_{auth}. In the `Verify` phase, we pay attention to the verification time of this algorithm. At the same time, since proof verification in our scheme needs to be realized through the contract, we also ought to consider the verification time in the contract. The experiment results are listed in Table II. We can conclude that the requirements of storage and time cost are all reasonable and acceptable. Although the verification process is implemented in the contract, which leads to an increase in time cost, the average cost of 1.25 seconds in our experiment is totally acceptable.

TABLE II
Overhead of Zero-Knowledge Proof

Setup			Prove		Verify	
S_{PK}	S_{VK}	T	S_π	T	T_{local}	$T_{contract}$
21.25M	1.32K	4.89s	289B	0.79s	0.03s	1.25s

B. Performance of Proxy Re-Encryption

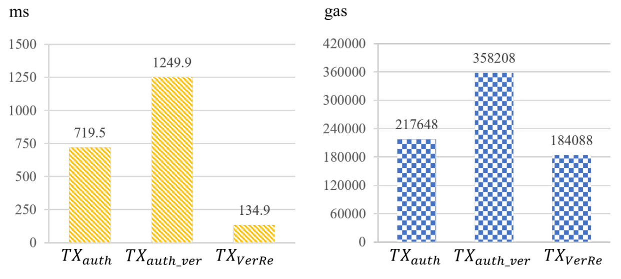
In our scheme, we do not directly perform proxy re-encryption operations on data, but perform the calculation on symmetric keys. Therefore, the performance of proxy re-encryption may be related to the length of the symmetric key. Here, we use the AES algorithm, where the key *k* size supports 128bit, 192bit, 256bit. So we have tested the re-encryption performance of these three key sizes. The units of test results are all milliseconds, as shown in Table III. Through the table, it can be found that the key size at this level is not enough to have a sufficient impact on the efficiency of the algorithm, so we don't need to care too much about the key size. On the other hand, during *ReEnc*, the calculation time on the IPFS node is significantly higher than that of the local calculation, caused by the IPFS node having to retrieve and download the data. For the *VerRe* algorithm, the contract time is not much higher than that of the local operation, indicating that our design to expand the verification function into the pre-compiled contract is successful. In short, the calculation time of each algorithm of PRE is in milliseconds, which satisfies the system requirements well.

C. Performance of System Transactions

Finally, we simulate the complete image authorization process and evaluate the performance of several core modules

TABLE III
PERFORMANCE OF PROXY RE-ENCRYPTION

Key Size	Enc	$ReEnc$		$VerRe$		Dec
		Local	IPFS	Local	Contract	
128bit	51.1	72.4	203.2	117.3	131.1	120.6
192bit	51.7	72.6	204.0	117.5	136.7	120.7
256bit	52.4	72.7	203.5	117.7	140.0	120.7



(a) Time cost (b) Gas cost

Fig. 5. Performance of System Transactions.

overhead involved. The new on-chain transactions involved in the proposed scheme mainly include three transactions, which are TX_{auth} in the Data_Auth phase, TX_{auth_ver} in the Auth_Ver phase, and TX_{VerRe} in the Pre_Enc phase. Therefore, we can test their time cost and gas cost for these three transactions, as shown in Fig. 5. In Ethereum, the cost of gas is not fixed, depending on the complexity of the transaction operation. Generally speaking, the more resources (computation, memory, etc.) are consumed, the more gas is required. For TX_{auth} transactions, we need to permanently store the commitment C_{addr} and update the Merkle tree *ComMerkle* in the related contract, which uses the keyword storage (persistent contract state variable), so it needs to consume a lot of gas. In the other two transactions, there is also a large amount of computing required to consume gas. But overall, gas consumed by these transactions is in an acceptable range. From the perspective of time cost, TX_{auth} and TX_{VerRe} are both milliseconds, and TX_{auth_ver} transaction will incur time costs due to zero-knowledge verification, but the overall time is less than 1.5s, which is completely within the acceptable range.

Besides, the monetary cost of these transactions on blockchain may be interesting because of the high price of cryptocurrency. We think the high cost can be somewhat mitigable. On the one hand, our scheme can realize "one authorization, multiple use". That is, the same data can be reused after one authorization. On the other hand, we use Ethereum in our experiment to demonstrate the broad applicability of this scheme, in fact, the security and privacy of our scheme do not depend on Ethereum. We can replace Ethereum with other efficient and low-cost blockchains. Of course, blockchain is the core and key component of Web 3.0. The high cost it brings is a problem that must be solved in the development of Web 3.0 and does not need to be discussed in our Scheme.

VIII. RELATED WORK

This section reviews a number of current privacy protection research in terms of blockchain authorization and data sharing.

A. Blockchain Authorization

To date, there are many researches on the application of blockchain authorization, taking advantage of the decentralized and immutable characteristics of blockchain for personal data management [33], [34], access control [35], [36], [37], [38], [39] and trusted authorization [40], [41], [42]. In particular, some studies place emphasis on cross-organization and cross-domain authorization [43], [44], [45]. For example, Lu et al. [43] stored users' access control list through blockchain to manage who has access to these cross-organization software resources, so as to carry out permission control and authorization. But this way only hides the link relationship between users and organizations from the outside. There is no consideration of privacy hidden for organizations, which will know what data the user has accessed. In the research of Xiao et al. [44], they proposed EMRShare, using blockchain to construct a cross-organization medical data framework to solve trust issues between patients, clinicians, insurance agents, and governments. The scheme through this way is able to guarantee the privacy, validity, and integrity of data. However, similarly, when a patient would like to access data (e.g., another hospital visit or medical reimbursement), this behavior will definitely be collected by the relevant hospital storage party, seriously exposing the user's privacy. The same in [38], the use of patients' Electronic Health Records (EHRs) is completely exposed to the service provider. As shown in Table IV, all these schemes support external transparency (hiding authorization objects from unrelated parties that are outside the system), but transparency for service/storage providers is only supported in our scheme.

B. Data Sharing

In terms of data sharing, there have been numerous studies focusing on data security and privacy leakage during the process of data sharing. Many encryption technologies have been applied in their researches, including attribute-based encryption [46], [49], proxy re-encryption [47], [50], [51], SGX [11], [48], etc. In [46], ABE technology is used to encrypt each patient's personal health record (PHR) to achieve fine-grained and extensible data access control. For [47], Su et al. based on PRE technology, regard node state as a part of the parameters of re-encryption in their scheme PRTA, which is under the control of the authorization server to ensure data security and reliability. In [48], through the hardware-based trusted computing technology SGX, the fine-grained access control of remote monitoring of the Internet of Things in scheme SRM is realized. As a result, the decrypted medical data can only be viewed and operated in a trusted environment. However, the data sharing schemes listed in Table IV generally focus only on cases where the service/storage provider does not have access to the data during data transmission (i.e., data is encrypted by the user and cannot be decrypted by the storage provider to obtain the plaintext). However, in practice, service providers generally manage and generate user-sensitive plaintext data. As a result, a dishonest or malicious service provider can easily monitor users' data usage.

TABLE IV
COMPARISON BETWEEN PASSP AND PREVIOUS STUDIES

	Data				Transparency	
	Privacy	Validity	Integrity	Decentralization Storage	External	Server
Lu et al. [43]	✓				✓	
Xiao et al. [44]	✓	✓	✓		✓	
Sharma et al. [38]	✓	✓	✓	✓	✓	
Li et al. [46]	✓	✓	✓		✓	
Su et al. [47]	✓	✓	✓		✓	
Chen et al. [48]	✓	✓			✓	
Our scheme	✓	✓	✓	✓	✓	✓

Algorithm 1 *Enc*

```
Input: $pk_a, sk_a$, message: $m$
Output: $\boldsymbol{c}_a$
1: function ENC($pk_a, sk_a, m$)
2:     $R{:}= g^r$, $T{:}= e(pk_a, pk_a)^r$
3:     $r_1{:}= H_1(R), r_2{:}= H_2(T), c_1{:}= m \oplus r_2$
4:     $h{:}= H_3(pk_a, r_1, c_1), c_2{:}= g^{sk_a \cdot h}, c_3{:}= r - sk_a \cdot h$
5:     return $\boldsymbol{c}_a{:}= (c_1, c_2, c_3)$
6: end function
```

Algorithm 2 *ReEnc*

```
Input: $\boldsymbol{c}_a{:}= (c_1, c_2, c_3), rk_{a,s}$
Output: $\boldsymbol{c}'_s = (c_1, c'_2, c'_3, c_4)$
1: function REENC($\boldsymbol{c}_a, rk$)
2:     $c'_2{:}= g^{c_3} \cdot c_2, c_4{:}= e(c'_2, rk)$
3:     $c'_3{:}= H_4(H_1(c'_2), c_2, c_1, c_4)^{c_3}$
4:     return $\boldsymbol{c}'_s{:}= (c_1, c'_2, c'_3, c_4)$
5: end function
```

Algorithm 3 *VerRe*

```
Input: $\boldsymbol{c}'_s{:}= (c_1, c'_2, c'_3, c_4), pk_a, pk_s$
Output: isValid
1: function VerRe($\boldsymbol{c}'_s, pk_a, pk_s$)
2:     $r'_1{:}= H_1(c'_2)$
3:     $h'{:}= H_3(pk_a, r'_1, c_1)$
4:     $h''{:}= H_4(r'_1, pk_s^{h'}, c_1, c'_3)$
5:     if $e(c_4, g) == e(h'', c'_2)e(h'', pk_s)^{-h'}$ then
6:         return ture
7:     else
8:         return false
9:     end if
10: end function
```

Algorithm 4 *Dec*

```
Input: $\boldsymbol{c}'{:}= (c_1, c'_2, c'_3, c_4), pk_a, sk_s$
Output: message: $m$
1: function Dec($\boldsymbol{c}'_s, pk_a, sk_s$)
2:     $r'_1{:}= H_1(c'_2)$
3:     $r'_2{:}= (c'_3)^{sk_s^{-1}}$
4:     $h'{:}= H_4(pk_a, r'_1, c_1)$
5:     $h''{:}= H_5(r'_1, pk_s^{h'}, c_1, c'_3)$
6:     if $e(c'_4, g) == e(h'', c'_2)e(h'', pk_s)^{-h'}$ then
7:         $m{:}= c_1 \oplus r'_2$
8:         return $m$
9:     else
10:        return error
11:    end if
12: end function
```

IX. CONCLUSION

As the number of data abuse scandals involving service providers increases, so does the need to strengthen the protection of user data, for example by minimizing the risk of exposure to service providers during the authorization process. Therefore, we proposed a private authorization scheme oriented service providers (PASSP) in Web 3.0 concept, which is the first private authorization scheme to minimize privacy risks associated with service providers. Specifically, the trustless authorization authentication method uses zero-knowledge proof technology as the core to hide the authorization relationship in authorization, and conducts public verification on the blockchain, so that the adversary cannot obtain any authorization object and content. The decentralized publicly-verifiable proxy re-encryption method based on IPFS proposed in our scheme guarantees the security and integrity of the data during the authorization process and also prevents adversaries from profiting from it. Besides, We evaluated both the security and the performance of the scheme to demonstrate its utility. For example, the on-chain verification time of zero-knowledge proof is generally less than 1.5s, and on the other hand, the re-encryption verification time is in milliseconds. While our scheme may not be able to prevent service providers from leaking data privately, our scheme does restrict service providers from obtaining additional information to some extent. Hence, the scheme serves as an additional measure to ensure user data privacy authorization. At the same time, our scheme provides a reliable solution for the future development of Web 3.0 to solve user data privatization and data privacy authorization.

APPENDIX
PROXY RE-ENCRYPTION ALGORITHM

See Algorithms 1, 2, 3, 4.

REFERENCES

[1] L. Mathews, “Equifax data breach impacts 143 million Americans.” 2017. Accessed: Sep. 11, 2023. [Online]. Available: https://www.forbes.com/sites/leemathews/2017/09/07/equifax-data-breach-impacts-143-million-americans

[2] S. Gressin, “The Marriott data breach.” 2018. Accessed: Dec. 20, 2023. [Online]. Available: https://consumer.ftc.gov/consumer-alerts/2018/12/marriott-data-breach

[3] K. Conger, “Uber settles data breach investigation for $148 million.” 2018. Accessed: Sep. 21, 2022. [Online]. Available: https://www.nytimes.com/2018/09/26/technology/uber-data-breach.html

[4] D. Deahl, “Verizon partner data breach exposes millions of customer records.” 2017. Accessed: Sep. 21, 2022. [Online]. Available: https://www.theverge.com/2017/7/12/15962520/verizon-nice-systems-data-breach-exposes-millions-customer-records

[5] M. Isaac and S. Frenkel, *Facebook Security Breach Exposes Accounts of 50 Million Users*, New York Times, New York, NY, USA, vol. 28, 2018, Accessed: May. 13, 2023. [Online]. Available: https://www. nytimes. com/2018/09/28/technology/facebook-hack-data-breach. html

[6] C. Gentry, “Fully homomorphic encryption using ideal lattices,” in *Proc. 41st Annu. ACM Symp. Theory Comput.*, Bethesda, MD, USA, May 2009, pp. 169–178.

[7] O. Goldreich, *The Foundations of Cryptography—Volume 2: Basic Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[8] M. Sabt, M. Achemlal, and A. Bouabdallah, “Trusted execution environment: What it is, and what it is not,” in *Proc. IEEE Trustcom/BigDataSE/ISPA*, vol. 1. Helsinki, Finland, Aug. 2015, pp. 57–64.

[9] S. Kim, K. Lee, W. Cho, Y. Nam, J. H. Cheon, and R. A. Rutenbar, “Hardware architecture of a number theoretic transform for a bootstrappable RNS-based homomorphic encryption scheme,” in *Proc. IEEE 28th Annu. Int. Symp. Field-Programmable Custom Comput. Mach.*, Fayetteville, AR, USA, May 2020, pp. 56–64.

[10] J.-E. Ekberg, K. Kostiainen, and N. Asokan, “The untapped potential of trusted execution environments on mobile devices,” *IEEE Security Privacy*, vol. 12, no. 4, pp. 29–37, Jul./Aug. 2014.

[11] G. Ayoade, V. Karande, L. Khan, and K. Hamlen, “Decentralized IoT data management using blockchain and trusted execution environment,” in *Proc. 19th IEEE Int. Conf. Inf. Reuse Integr.*, Salt Lake City, UT, USA, Jul. 2018, pp. 15–22.

[12] D. G. Nair, V. P. Binu, and G. S. Kumar, “An improved E-voting scheme using secret sharing based secure multi-party computation,” 2015, *arXiv:1502.07469*.

[13] M. Hirt and K. Sako, “Efficient receipt-free voting based on homomorphic encryption,” in *Proc. Int. Conf. Theory Appl. Crytograph. Techn.*, Bruges, Belgium, May 2000, pp. 539–556.

[14] X. Fu, H. Wang, P. Shi, X. Ma, and X. Zhang, “Teegraph: Trusted execution environment and directed acyclic graph-based consensus algorithm for IoT blockchains,” *Sci. China Inf. Sci.*, vol. 65, no. 3, pp. 1–3, 2022.

[15] J. De Clercq, “Single sign-on architectures,” in *Proc. Int. Conf. Infrastruct. Security*, Bristol, U.K., Oct. 2002, pp. 40–58.

[16] D. Hardt, “The OAuth 2.0 authorization framework,” RFC 6749, IETF, pp. 1–76, 2012.

[17] U. W. Chohan, “Web 3.0: The future architecture of the Internet?” *SSRN Electron. J.*, p. 14, Feb. 2022.

[18] G. Korpal and D. Scott, “Decentralization and Web3 technologies,” *Authore*, Preprints, 2023.

[19] Z. Liu et al., “Make Web3.0 connected,” *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 5, pp. 2965–2981, Sep./Oct. 2022.

[20] J. Benet, “IPFS—Content addressed, versioned, P2P file system,” 2014, *arXiv:1407.3561*.

[21] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, “Blockchain,” *Bus. Inf. Syst. Eng.*, vol. 59, no. 3, pp. 183–187, 2017.

[22] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, “An overview of blockchain technology: Architecture, consensus, and future trends,” in *Proc. 6th IEEE Int. Congr. Big Data*, Honolulu, HI, USA, Jun. 2017, pp. 557–564.

[23] M. Blaze, G. Bleumer, and M. Strauss, “Divertible protocols and atomic proxy cryptography,” in *Proc. Int. Conf. Theory Appl. Crytograph. Techn.*, Espoo, Finland, May 1998, pp. 127–144.

[24] S. Chandrasekar, K. Ambika, and C. P. Rangan, “Signcryption with proxy re-encryption,” IACR Cryptol. ePrint Arch., IACR, Bellevue, WA, USA, Rep. 2008/276, 2008.

[25] C. Wang and X. Cao, “An improved signcryption with proxy re-encryption and its application,” in *Proc. 7th Int. Conf. Comput. Intell. Security*, Dec. 2011, pp. 886–890.

[26] H. Li and C. Lan, “Signcryption of proxy re-encryption with publicly verification,” *J. Comput. Appl.*, vol. 33, no. 4, pp. 1055–1060, 2013.

[27] V. Kirtane and C. P. Rangan, “RSA-TBOS signcryption with proxy re-encryption,” in *Proc. 8th ACM Workshop Digit. Rights Manage.*, Alexandria, VA, USA, Oct. 2008, pp. 59–66.

[28] O. Goldreich and Y. Oren, “Definitions and properties of zero-knowledge proof systems,” *J. Cryptol.*, vol. 7, no. 1, pp. 1–32, 1994.

[29] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, “SNARKs for C: Verifying program executions succinctly and in zero knowledge,” in *Proc. 33rd Annu. Int. Cryptol. Conf.*, Santa Barbara, CA, USA, Aug. 2013, pp. 90–108.

[30] Y. Zheng, “Digital Signcryption or how to achieve cost(signature & encryption) $<<$ cost(signature) $+$ cost(encryption),” in *Proc. 17th Annu. Int. Cryptol. Conf.*, Santa Barbara, CA, USA, Aug. 1997, pp. 165–179.

[31] “Zkcrypto/Bellman.” Accessed: Feb. 24, 2023. [Online]. Available: https://github.com/zkcrypto/bellman/

[32] B. Lynn, “PBC (pairing-based cryptography) library,” Accessed: Feb. 24, 2023. [Online]. Available: https://crypto.stanford.edu/pbc/

[33] N. B. Truong, K. Sun, G. M. Lee, and Y. Guo, “GDPR-compliant personal data management: A blockchain-based solution,” *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1746–1761, 2020.

[34] N. B. Truong, K. Sun, and Y. Guo, “Blockchain-based personal data management: From fiction to solution,” in *Proc. 18th IEEE Int. Symp. Netw. Comput. Appl.*, Cambridge, MA, USA, Sep. 2019, pp. 1–8.

[35] S. Alansari, F. Paci, and V. Sassone, “A distributed access control system for cloud federations,” in *Proc. 37th IEEE Int. Conf. Distrib. Comput. Syst.*, Atlanta, GA, USA, Jun. 2017, pp. 2131–2136.

[36] C. Lin, D. He, X. Huang, K.-K. R. Choo, and A. V. Vasilakos, “BSeIn: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0,” *J. Netw. Comput. Appl.*, vol. 116, pp. 42–52, Aug. 2018.

[37] O. J. A. Pinno, A. R. A. Grégio, and L. C. E. D. Bona, “Controlchain: Blockchain as a central enabler for access control authorizations in the IoT,” in *Proc. IEEE Global Commun. Conf.*, 2017, pp. 1–6.

[38] B. Sharma, R. Halder, and J. Singh, “Blockchain-based interoperable healthcare using zero-knowledge proofs and proxy re-encryption,” in *Proc. 12th IEEE Int. Conf. Commun. Syst. Netw.*, Bengaluru, India, 2020, pp. 1–6.

[39] Z. Guan, W. Yang, L. Zhu, L. Wu, and R. Wang, “Achieving adaptively secure data access control with privacy protection for lightweight IoT devices,” *Sci. China Inf. Sci.*, vol. 64, no. 6, pp. 1–14, 2021.

[40] C. Esposito, M. Ficco, and B. B. Gupta, “Blockchain-based authentication and authorization for smart city applications,” *Inf. Process. Manag.*, vol. 58, no. 2, 2021, Art. no. 102468.

[41] H. Ma, E. X. Huang, and K.-Y. Lam, “Blockchain-based mechanism for fine-grained authorization in data crowdsourcing,” *Future Gener. Comput. Syst.*, vol. 106, pp. 121–134, May 2020.

[42] A. A. Battah, M. M. Madine, H. Alzaabi, I. Yaqoob, K. Salah, and R. Jayaraman, “Blockchain-based multi-party Authorization for accessing IPFS encrypted data,” *IEEE Access*, vol. 8, pp. 196813–196825, 2020.

[43] P. J. Lu, L. Yeh, and J. Huang, “An privacy-preserving cross-Organizational authentication/Authorization/accounting system using blockchain technology,” in *Proc. IEEE Int. Conf. Commun.*, Kansas City, MO, USA, May 2018, pp. 1–6.

[44] Z. Xiao et al., “EMRShare: A cross-organizational medical data sharing and management framework using permissioned blockchain,” in *Proc. 24th IEEE Int. Conf. Parallel Distrib. Syst.*, 2018, pp. 998–1003.

[45] G. Ali et al., “xDBAuth: Blockchain based cross domain authentication and authorization framework for Internet of Things,” *IEEE Access*, vol. 8, pp. 58800–58816, 2020.

[46] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, “Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 131–143, Jan. 2013.

[47] M. Su, B. Zhou, A. Fu, Y. Yu, and G. Zhang, “PRTA: A proxy re-encryption based trusted Authorization scheme for nodes on CloudIoT,” *Inf. Sci.*, vol. 527, pp. 533–547, Jul. 2020.

[48] Y. Chen, W. Sun, N. Zhang, Q. Zheng, W. Lou, and Y. T. Hou, “Towards efficient fine-grained access control and trustworthy data processing for remote monitoring services in IoT,” *IEEE Trans. Inf. Forensics Security*, vol. 14, pp. 1830–1842, 2018.

[49] J. Hur and D. K. Noh, “Attribute-based access control with efficient revocation in data outsourcing systems,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 7, pp. 1214–1221, Jul. 2011.

[50] G. Wei, R. Lu, and J. Shao, “EFADS: Efficient, flexible and anonymous data sharing protocol for cloud computing with proxy re-encryption,” *J. Comput. Syst. Sci.*, vol. 80, no. 8, pp. 1549–1562, 2014.

[51] W. Feng, Z. Zhang, J. Wang, and L. Han, “A novel authorization delegation scheme for multimedia social networks by using proxy re-encryption,” *Multimedia Tools Appl.*, vol. 75, pp. 13995–14014, Nov. 2016.