PeerJ Computer Science

Blockchain-enabled data governance for privacy-preserved sharing of confidential data

Jingchi Zhang[1] and Anwitaman Datta[1,2]

[1] College of Computing and Data Science, Nanyang Technological University, Singapore, Singapore
[2] De Montfort University Leicester, Leicester, United Kingdom

ABSTRACT

In traditional cloud storage systems, users benefit from the convenience of data accessibility but face significant risks related to security. Ciphertext-policy attribute-based encryption (CP-ABE) schemes are employed to achieve fine-grained access control in cloud services to ensure confidentiality while maintaining data-sharing capabilities. However, existing approaches are impaired by two critical issues: illegal authorization and privacy leakage. Despite extensive discussions in the literature on interoperability, performance, scalability, and stability, the security of ABE-based cloud storage and data-sharing systems against adversaries—particularly those involving adaptively corrupt attribute authorities gaining unauthorized access to users' data—has not been sufficiently explored. Notably, few existing works even address security in the presence of adversaries, raising concerns about the practicality of these systems in real-world scenarios where malicious behavior is a genuine threat. Another pressing issue is privacy leakage, where sensitive user information, such as medical histories in healthcare use cases, embedded within the access policies, may be exposed to all users. This problem is exacerbated in ABE schemes that integrate blockchain technology for enhanced decentralization and interoperability, as using a public ledger shared across multiple users can further compromise privacy. To address these, we propose an enhanced blockchain-based data governance system that employs blockchain technology and attribute-based encryption to prevent illegal authorization and privacy leakage. Our novel ABE encryption system supports multi-authority use cases while hiding access policy and ensuring identity privacy, which also protects data sharing against corrupt authorities. Utilizing the Advanced Encryption Standard (AES) for data encryption, our system is optimized for real-world efficiency. Notably, the encrypted data is stored in a decentralized storage system, like the InterPlanetary File System (IPFS), which does not rely on any centralized service provider and can, therefore, be leveraged to achieve resilience against single-point failures. With the integration of smart contracts and multi-authority attribute-based encryption, coupled with blockchain's inherent transparency and traceability, our system realizes a balanced solution for fine-grained access control with preserved privacy, further fortifying against credential misuse. Besides the system design, we also present security proofs to demonstrate the robustness of the proposed system.

Subjects Cryptography, Security and Privacy, Blockchain
Keywords Attribute-based encryption, Blockchain, Data sharing, Multi-authority, Privacy

Submitted 27 June 2024
Accepted 14 November 2024
Published 20 December 2024

Corresponding author
Jingchi Zhang,
jingchi001@e.ntu.edu.sg

Academic editor
Yue Zhang

Additional Information and Declarations can be found on page 43

DOI **10.7717/peerj-cs.2581**

How to cite this article Zhang J, Datta A. 2024. Blockchain-enabled data governance for privacy-preserved sharing of confidential data. *PeerJ Comput. Sci.* **10**:e2581 http://doi.org/10.7717/peerj-cs.2581

INTRODUCTION

Notwithstanding the many advantages of cloud computing, which have led to its widespread adoption and continuous growth, it also presents certain risks that prompt the exploration of alternative architectures. In particular, due to the inherent centralization of cloud services, they can become a single point of failure. This presents issues regarding service availability, censorship, and end-user privacy concerns. These challenges are further exacerbated by potential insider attacks and the service provider's own agency. For instance, Apple's decision in 2021 to roll out a Child Sexual Abuse Material detection technology by scanning images stored in its iCloud service led to numerous collateral privacy concerns and criticisms (*Mitchell, 2022*). Even though the original plan was eventually scuppered because of strong public backlash, the fundamental vulnerability of such centralized systems, which are subject to privacy violations or censorship, remains. (Portions of this text were previously published as part of a preprint (*Zhang & Datta, 2023*)).

To address some of these issues inherent in centralized cloud storage, many encryption schemes such as AES, RSA, proxy re-encryption, identity-based encryption, and attribute-based encryption (ABE) have been used to secure data confidentiality (*Sudha & Monica, 2012*; *Alowolodu et al., 2013*; *Yan, Rong & Zhao, 2009*; *Yang et al., 2020*). However, some encryption schemes may not be amenable to a wide variety of common data-sharing use cases.

Consider a complex surgical procedure that requires collaboration among specialized experts from around the world. These experts need access to the patient's electronic health records (EHRs) to plan the operation and ensure all necessary equipment and preparations are in place. Therefore, the patient (data owner, *DO*) must securely share their private EHRs with doctors, hospitals, and other relevant institutions (data users, *DU*s) involved in the surgery. The primary challenge is ensuring that the EHRs are shared with the appropriate professionals on a need-to-know basis without exposing sensitive patient information, while also supporting flexible accessibility so that new DUs can access the EHRs without significant additional effort. Relying on a single trusted entity for data sharing can introduce risks, especially if that entity is compromised, and no single doctor or hospital typically has contact with all the required specialists.

To address this, a reliable and decentralized data-sharing system is essential. This system could publicly advertise the medical requirements to attract qualified experts worldwide while authorizing specific vetted individuals and institutions to access the patient's EHRs. In this context, a blockchain-based data-sharing system is preferable (*Wang, Zhang & Zhang, 2018*; *Gao et al., 2020*; *Qin et al., 2021*). Blockchain's key features—immutability, transparency, and decentralized control—offer a secure environment for sharing sensitive medical information. By integrating advanced encryption techniques, the system ensures that only authorized parties with the correct access privileges can view and interact with the EHRs, maintaining patient privacy and safeguarding data integrity.

In a traditional public-key encryption (PKE) system like RSA or an Identity-Based Encryption (IBE) scheme, the patient must re-encrypt the EHR using the doctor or institution's public key or generate a new identity-based ciphertext. These access control

options are limited in flexibility and scalability, as the *DO* may have to take additional steps on demand to ensure that the encrypted content is accessible to a new *DU*. In contrast, Ciphertext-Policy Attribute-based Encryption (CP-ABE) schemes offer a more flexible approach. Successful decryption can be carried out only if a user's attribute set satisfies the access policy embedded in the ciphertext. Therefore, regardless of whether the number of users is predetermined, new *DU*s can access the EHRs without requiring significant additional effort from the *DO*.

However, current CP-ABE schemes may introduce privacy issues. The embedded policy in the encrypted EHR could reveal the identity of potential *DU*s, leading to the unintended disclosure of the patient's information (*Wu, Xu & Zhu, 2023*). This issue is exacerbated when such information is uploaded on-chain to leverage the benefits of trustable and immutable logging (*Wu, Xu & Zhu, 2023*), whether using a public ledger like the Ethereum network or a private ledger like Hyperledger Fabric. For instance, records indicating visits to specialized hospitals or interactions with an insurance company may be used to infer a patient's condition (*Gao et al., 2020*). Unauthorized entities, such as pharmaceutical companies that target patients with advertisements, could exploit this information. Similarly, other hospitals or insurance companies might use this data for marketing or discriminate pricing. It is thus crucial to hide the access policy to protect patient privacy.

Another practical concern is the risk of illegal authorization, where a corrupt authority might issue attribute keys to unauthorized *DU*s (*Hei et al., 2021*). Revisiting the use case above, companies interested in accessing patients' information could bribe an attribute authority to obtain decryption keys. Alternatively, a company acting as an attribute authority might introduce a 'backdoor' during the setup phase, allowing future unauthorized access. Consequently, ensuring the security of blockchain-based data-sharing systems against corrupt authorities is also essential.

Motivations

In this subsection, we elaborate on prior ABE schemes' limitations and practical challenges, highlighting the need for our work.

Decentralization

In traditional cloud services, the data owner (*DO*) typically uploads ciphertext to a cloud storage server, which means the user may take the risk of assuming that the cloud storage server will provide reliable service to ensure the availability of the uploaded data (*Wang, Zhang & Zhang, 2018*). However, this reliance inherently introduces a single point of failure. Another drawback of centralized cloud services arises when integrating CP-ABE for improved flexibility and scalability: it relies on intermediary entities like a trusted third party (TTP) and a central authority (CA) to perform operations such as attribute key distribution, policy verification, and data retrieval honestly. These operations are critical for ensuring the security and trustworthiness of data access control, but the reliance on centralized entities poses significant risks.

To better safeguard data availability and ensure the reliability and traceability of operations, it is crucial to design decentralized alternatives for trust mechanisms and

enforce traceability throughout the access control system (*Wang, Zhang & Zhang, 2018*). Consider the application scenario described above: if the potential data users (*DUs*) are not determined in advance, and the purpose of data sharing also includes attracting public attention, such as coordinating a challenging surgery or managing world-level tasks, a blockchain-based solution is inherently advantageous. Blockchain's decentralized nature allows for broader participation and enables the system to support complex, multi-faceted applications that require secure and transparent access control mechanisms.

A blockchain-based access control system with CP-ABE has the potential to address these issues effectively. By leveraging blockchain, we can eliminate the need for a centralized TTP to control the data. Each node in the network maintains a distributed ledger that tracks a growing list of transactions, which are verified and confirmed by consensus mechanisms before being recorded. The integrity of transactions can be secured by hashing, Merkle trees, time stamping, and incentive mechanisms. This hybrid approach enhances resilience against single-point failures and the misuse of credentials by ensuring that no single entity has complete control over the access control system. Additionally, the decentralized nature of blockchain inherently supports transparency and traceability, preventing any individual authority or entity from acting maliciously without detection (*Gao et al., 2020*; *Hei et al., 2021*; *Wu et al., 2019*).

Practicality

Several blockchain-based access control systems have been proposed since the emergence of public blockchain systems (*Nakamoto, 2008*) and the advent of Attributed-based Encryption (*Sahai & Waters, 2005*). Some efforts leverage the immutable public ledger to build a transaction-based access control system for secure data sharing (*Maesa, Mori & Ricci, 2017*; *Ouaddah, Abou Elkalam & Ait Ouahman, 2016*; *Wang, Zhang & Zhang, 2018*). In contrast, others leverage the self-executing smart contracts to establish a smart contract-based access control system for flexibility and traceability (*Qin et al., 2021*; *Hei et al., 2021*; *Wu, Xu & Zhu, 2023*). However, just employing blockchain technology and CP-ABE encryption for an access control system is inadequate for several practical purposes, such as cross-domain data sharing and privacy leakage.

On the one hand, information is not always shared inside a single domain or organization. For example, driver's licenses and university registration information may be managed by separate entities. If one central authority is responsible for attribute management and key distribution like the proposed system (*Kaur, Rani & Kalra, 2024*), it also has trust issues, as discussed above. Therefore, multi-authority attribute-based encryption (MA-ABE), originally proposed by *Chase (2007)*, is used to solve the access problem involving attributes belonging to various authorities. This scheme permits any number of independent authorities to distribute secret keys, which the data owner later chooses to encrypt a message. However, this MA-ABE scheme also relies on a *CA* that issues seeds to each AA, giving the *CA* the capability to decrypt any ciphertext. To eliminate the need for a 'super-power' *CA* and achieve full decentralization, *Lewko & Waters (2011)* proposed a fully secure decentralized CP-ABE solution without requiring cooperation among multiple AAs.

Another issue is privacy concerns, which encompass both policy-hiding and receiver privacy. Since in the classic CP-ABE schemes, an access structure specified in terms of user attributes is explicitly transmitted alongside ciphertext, whoever accesses the ciphertext is also aware of the corresponding access policy. Therefore, multi-authority CP-ABE schemes (*Chase, 2007*; *Lewko & Waters, 2011*; *Rouselakis & Waters, 2015*) are still unsuitable for certain use cases since access policies contain sensitive information. This calls for mechanisms to hide access policies for CP-ABE systems. Additionally, *DU* needs to provide a full set of user attributes to each authority for an attribute key, inevitably compromising the key receiver's privacy.

In pursuit of addressing these concerns, several CP-ABE schemes that feature policy-hiding have been proposed (*Cui et al., 2018*; *Zhang, Zheng & Deng, 2018*; *Yang et al., 2018*; *Gao et al., 2020*; *Michalevsky & Joye, 2018*; *Zhang et al., 2021*; *Kaur, Rani & Kalra, 2024*). Despite these efforts, they do not completely fulfill various practical requirements. Some of them (*Cui et al., 2018*; *Zhang, Zheng & Deng, 2018*; *Gao et al., 2020*; *Kaur, Rani & Kalra, 2024*) only support a single central authority. Schemes such those from *Yang et al. (2018)* and *Zhang et al. (2021)* are prone to the leakage of *DU*'s confidential attribute information during the key generation or encryption process. *Michalevsky & Joye (2018)* introduced a fully policy-hiding decentralized CP-ABE scheme, which protects attribute information attached to the access policy and even addresses the issue of receiver privacy.

Security

Beyond the basic security goals of access control systems, such as data confidentiality, ABE schemes need to address another security issue: collusion between users. Specifically, even if data users (*DU*s) collude by sharing their attribute keys, they should not be able to decrypt ciphertexts unless each of their issued attribute keys **individually** satisfies the access policy (*Bethencourt, Sahai & Waters, 2007*). Additionally, different types of ABE-based systems address varying security concerns, such as accountability, which is essential in accountable ABE (*Zhang et al., 2020b*).

In the context of multi-authority or decentralized ABE, the security goal of collusion resistance is further complicated by the possibility of keys being issued by different authorities. More importantly, the corruption of some but not all authorities should not compromise the confidentiality of the system's data (*Lewko & Waters, 2011*). However, many existing ABE-based solutions either fail to adequately address security in the presence of adversaries, as seen in the works like (*Xue et al., 2017*; *Wang, Zhang & Zhang, 2018*; *Wu, Xu & Zhu, 2023*), or define security goals for their application scenarios with noticeable omissions, as found in schemes (*Gao et al., 2020*; *Nasiraee & Ashouri-Talouki, 2020*; *Porwal & Mittal, 2020*). In general, the adversarial models in these works are often overly idealized and do not reflect real-world scenarios.

The issue of corrupted attribute authorities is a widely discussed security problem in most existing ABE-based systems that support multi-authority environments (*Yang et al., 2018*; *Qin et al., 2021*; *Zhang et al., 2021*; *Zhao et al., 2022*). However, these systems are typically proven secure only against static corruption of authorities, where enquiring about corrupt authorities are made at the beginning of the game (*Chen et al., 2023*). This implies

that the set of corrupted authorities must be fixed, and authority keys must be requested upfront, limiting the adversary's ability to adaptively change its attack strategy. Moreover, these schemes assume that all attribute authorities must join the system simultaneously, an assumption that is impractical in real-world deployments.

We further identify that several ABE schemes realized through inner-product predicate encryption (*Michalevsky & Joye, 2018*; *Tseng & Gao, 2022*) are vulnerable to **rogue-key attacks** under the fully adaptive security model, where an adversary can corrupt authorities at any point in time. In such an attack, a malicious attribute authority (*AA*) can generate and register an aggregate public key based on public information from other honest authorities. This rogue key can then be used to decrypt ciphertexts without possessing the necessary attribute keys to satisfy the access policy. Furthermore, works such as *Zhang et al. (2020a)* and *Agrawal, Goyal & Tomida (2021)*, which build upon the scheme in *Michalevsky & Joye (2018)*, may also inherit this vulnerability.

Another potential issue in the **setup** phase of the MA-ABE scheme, which has not drawn as much attention as corrupt authorities, is the reliance on a central or trusted authority for the generation of the global public key. For instance, in works such as *Hei et al. (2021)*, only the central authority is involved in this global setup process. This assumption introduces the risk of a single point of failure and overlooks the potential for adversaries to introduce a "backdoor" during setup, which could be exploited later to carry out more harmful attacks on ciphertexts. We also identify a potential leakage of sensitive information in the scheme proposed by *Michalevsky & Joye (2018)* if a trusted setup is not employed for the generation of the global public key.

As a result, the challenge of securely storing user data, enabling efficient data sharing, and managing multi-authority scenarios while concurrently maintaining a balance of decentralization, traceability, privacy, security, and efficiency constitutes a complex problem that requires innovative solutions.

Contributions

In this paper, we propose a multi-party CP-ABE-based storage outsourcing system that uses blockchain technology to address decentralization, practicality, and security problems. Our solution achieves fine-grained access control by allowing data owners to define precise access policies based on user attributes while ensuring user anonymity by concealing both access policies and user identities during data access. Additionally, it is resilient against rogue-key attacks under the fully adaptive corruption assumption introduced in the work *Datta, Komargodski & Waters (2023)*, ensuring stronger security compared to other Inner Product Predicate Encryption (IPPE)-based schemes, even in the presence of adaptively corrupted authorities.

The core contributions of this work are as follows:

- **Capability gap and vulnerability analysis of the state-of-the-art:** We examine several widely discussed ABE schemes that support multi-authority and privacy-preserving properties and select the scheme presented in *Michalevsky & Joye (2018)* as the most suitable for real-world scenarios to build our data-sharing system upon. We then closely analyze this scheme under a more realistic security model, **fully adaptive security**,

Table 1 Summary of access control system using attribute-based encryption (Part I).

Approach	Authority	Policy	Universe	Policy-hiding	Receiver-hiding	Access control	Storage
Wang, Zhang & Zhang (2018)	Single	AND	Small	No	No	Smart contract	IPFS
Cui et al. (2018)	Single	LSSS	Large	Partially	No	CSP	CSP
Zhang, Zheng & Deng (2018)	Single	LSSS	Large	Partially	No	CSP	CSP
Yang et al. (2018)	Multiple	LSSS	Small	No	Yes	CSP	CSP
Gao et al. (2020)	Single	AND	Large	Fully	No	Smart contract	CSP
Qin et al. (2021)	Multiple	LSSS	Small	No	No	CSP	CSP
Zhao et al. (2022)	Multiple	LSSS	Large	Fully	No	CSP	CSP
Tseng & Gao (2022)	Multiple	AND	Large	No	No	CSP	CSP
This work	Multiple	AND	Small	Fully	Yes	Smart contract	IPFS

and identify that it is vulnerable to a rogue-key attack, where a malicious *AA* can decrypt ciphertext without possessing the necessary attribute keys required to satisfy the policy. Furthermore, the scheme is exposed to a potential risk where an adversary might infer sensitive information from the published ciphertext due to poorly chosen public parameters. These vulnerabilities are thoroughly analyzed in "Attack" and "Vulnerability", respectively.

- **Rogue-key attack resilient protocol design:** To counteract the rogue-key attack and alleviate some potential risks, we modify the algorithms of **Setup** and **Auth Setup** in *Michalevsky & Joye (2018)* as described in Definition 5. Firstly, we introduce a multi-party protocol inspired by *Bowe, Gabizon & Green (2018)* for public key generation, which is detailed in **Trusted Setup** of "Trusted Setup". Secondly, we impose a prerequisite for each *AA* to prove the knowledge of published information during the process of **Auth Setup**. This is elaborated in "Authority Setup". We further demonstrate that our enhanced system successfully mitigates the aforementioned security concerns, as outlined in "Proof of security of our approach" and "Proof of security with our approach".
- **System architecture for blockchain integration:** In order to incorporate transparency and decentralization, we integrate blockchain technologies such as smart contracts and content addressing, alongside multi-authority attribute-based encryption. An overview of the system architecture is presented in "System Overview". This hybrid approach enhances the practicality and security of the system, which makes it resilient against single-point failures and misuse of credentials. Given that transparency and traceability are inherent attributes of blockchain, a blockchain-enabled ABE system realizes a balanced solution for data sharing while simultaneously preserving privacy.
- **Comprehensive comparison with related works:** We provide a comprehensive comparison of existing ABE-based data-sharing systems in terms of decentralization, privacy, and security, as discussed in "Related Work" and summarized in Table 1. To position our proposed work relative to existing solutions regarding efficiency, we evaluate computational complexity in "Asymptotic Comparisons" and present the experimental results comparing two closely related IPPE-based schemes with our enhanced construction, along with analysis in "Experimental Result".

Overall, we propose a secure, privacy-preserving data governance system based on blockchain technology and an improved decentralized policy-hiding CP-ABE scheme with receiver privacy. Using a combination of ABE and the Advanced Encryption Standard (AES) makes the system practical. The special ABE encryption scheme is capable of handling multi-authority use cases while protecting identity privacy and ABE's policy. The adoption of AES helps assure the confidentiality of user data, which is furthermore stored in a decentralized storage system, specifically the InterPlanetary File System (IPFS), which does not rely on a central service provider, thus avoiding a single point of failure.

Organization

The rest of the paper is structured as follows: "Related Work" contains related work that reviews traditional Attribute-based Encryption schemes and conducts an analysis of some recent solutions for access control systems with ABE technology, elaborated in Table 1. "Preliminary" summarizes the preliminaries that the techniques developed in this paper build upon. The proposed system protocol is overviewed in "System Overview" and discussed in depth in "System Design". "Security Analysis" contains systematic security analysis, while "Performance Analysis" provides a comparative study of our system against related works. Finally, our conclusions and future plans are presented in "Concluding Remarks".

RELATED WORK

ABE was first introduced by Sahai and Waters in 2005 (*Sahai & Waters, 2005*). Subsequently, numerous proposals for a single-authority ABE system (*Goyal et al., 2006*; *Bethencourt, Sahai & Waters, 2007*) have been put forth. In these systems, the data owner (*DO*) encrypts data and employs a boolean formula over a set of attributes to restrict access. If the data user (*DU*) possesses the secret keys issued by a central authority (*CA*) that satisfy the boolean formula attached to the ciphertext, *DU* can retrieve the original data. However, these single-authority ABE systems (*Sahai & Waters, 2005*; *Goyal et al., 2006*; *Bethencourt, Sahai & Waters, 2007*) encounter constraints such as performance bottlenecks and key escrow issues.

Therefore, *Zhang et al. (2014)* proposed an enhanced ABE scheme, which alleviates the performance bottleneck issue by reducing the computation cost and ciphertext length. It has been further explored in *Wang, Zhang & Zhang (2018)* to create a framework that integrates decentralized storage, smart contract, and CP-ABE techniques to achieve fine-grained access control.

Another concern with the single-authority ABE system is key escrow, where *CA* issues all the attribute secret keys, thereby gaining the ability to decrypt each ciphertext generated by data owners. To address this issue, *Chase & Chow (2009)* introduced a multi-authority attribute-based (MA-ABE) scheme without the need for CA. *Lewko & Waters (2011)* further developed this multi-authority scheme in their work allowing any authority to join or leave the system independently. Based on it, *Qin et al. (2021)* designed a blockchain-based multi-authority access control scheme to address performance and single-point failure issues.

In an effort to extend the usability of ABE schemes, Nishide et al. presented a desirable property, hidden access policy, in *Nishide, Yoneyama & Ohta (2008)*. This approach protects sensitive attribute values while leaving attribute names public, denoted as partially hiding. Since then, multiple enhanced schemes (*Lai, Deng & Li, 2011*; *Cui et al., 2018*; *Zhang, Zheng & Deng, 2018*; *Gao et al., 2020*; *Xu et al., 2023b*) have been proposed. To support a wide variety of access structures, a fully secure policy-hiding ABE was proposed in *Lai, Deng & Li (2011)*. *Gao et al. (2020)* used the optimized scheme of *Lai, Deng & Li (2011)* to build a blockchain-based access control system that achieves trustworthy access while maintaining the privacy of policy and attributes. To improve the expressiveness of the access policy, a partially hidden ABE scheme under the Linear Secret Sharing Scheme (LSSS) policy was proposed in *Cui et al. (2018)*. *Zhang, Zheng & Deng (2018)* proposed a privacy-aware access control system, denoted as PASH, which supports a large universe ABE scheme with partially hidden ABE. There are several similar approaches providing policy-hiding as well as ensuring accountability for key abuse, for example, Li's work (*Li et al., 2022*) based on large universe ABE construction (*Rouselakis & Waters, 2013*) and the scheme of *Wu et al. (2019)* based on attribute bloom filter (ABF) (*Dong, Chen & Wen, 2013*). We also note that there is a longer list of desired features, such as keyword-searchable technology in ABE schemes or decentralized settings, as explored in works like (*Wang, Zhang & Zhang, 2018*; *Xu et al., 2023b*; *Xu et al., 2023a*).

Nevertheless, most of the aforementioned schemes either neglect the attribute of policy-hiding or exist as single-authority ABE systems. This gap is addressed by multi-authority attribute-based encryption schemes with a hidden access policy (*Zhong et al., 2016*; *Belguith et al., 2018*; *Michalevsky & Joye, 2018*; *Zhao et al., 2022*). The MA-ABE scheme featuring policy-hiding was initially introduced by *Zhong et al. (2016)*, and subsequently improved by *Belguith et al. (2018)* that significantly diminishes computational cost by delegating the decryption work to a semi-trusted cloud server.

In addition to the above, there are a few other proposals *Yang et al. (2018)*, *Zhao et al. (2022)* in this area that, unfortunately, give rise to additional issues. For instance, a system developed by *Yang et al. (2018)* keeps the user's identity private from the attribute authority (*AA*) if they are not in the same domain. Yet, this approach creates a new privacy issue that users might request *AA*s within the domain to ask secret attribute keys from other *AA*s on their behalf, implying that an *AA* could potentially possess a complete set of a *DU*'s secret keys. *Zhao et al. (2022)* presented a data sharing scheme that adopts the access policy of linear secret sharing scheme (LSSS) and supports the MA-ABE scheme with policy-hiding to achieve privacy-preserving functionality. However, this system is vulnerable to user key abuse due to its dependence on a single central authority for key generation.

In 2018, Michalevsky and Joye introduced the first practical multi-authority attribute-based encryption (MA-ABE) scheme with the policy-hiding property (*Michalevsky & Joye, 2018*), realized through Inner-Product Predicate Encryption (IPPE). This scheme provides a security proof in the random oracle model, against static corruption of authorities, where the list of corrupted authorities is fixed at the beginning of the security game. The scheme also supports various access policy types, including conjunctions, disjunctions, and threshold policies. Additionally, *Michalevsky & Joye (2018)* addressed the issue of

receiver privacy through the use of vector commitment. Notably, this scheme is one of the few that achieves fully hiding CP-ABE, ensuring that no attribute information is leaked with the access policies. Fully hiding CP-ABE can only be achieved indirectly *via* IPE or through threshold policies (*Zhang et al., 2020b*). Consequently, it is reasonable to build a data-sharing system based on this scheme, with appropriate modifications, to support real-world applications such as electronic health records or financial records.

However, the scheme of *Michalevsky & Joye (2018)* has its limitations, including support for only fixed-size attributes and authorities, and the need for coordination among authorities during the setup phase. More critically, we demonstrate that the scheme is vulnerable to a rogue-key attack in the presence of adaptively corrupted authorities, where corruption queries can be made at any point in time. This notion is later formalized as fully adaptive security by the work of *Datta, Komargodski & Waters (2023)* at EuroCrypt 2023. In real-world decentralized settings, it is realistic to expect that some authorities may join the data-sharing system later, due to factors such as network delays. Thus, the assumption that all authorities join simultaneously and that the list of corrupted authorities is fixed at the beginning of the security game does not reflect the dynamic and unpredictable nature of real-world scenarios. In practice, attackers may adaptively change their strategies based on the information available at any given time. The fully adaptive security model perfectly captures this situation and should be used to evaluate the security of proposed systems, ensuring they are practical enough to support real-world use cases. Specifically, in a rogue-key attack, a compromised authority may decrypt ciphertexts even without possessing the required attribute keys.

In Table 1, we compare and position our blockchain-enabled data-sharing system with existing works *Wang, Zhang & Zhang (2018)*, *Cui et al. (2018)*, *Zhang, Zheng & Deng (2018)*, *Yang et al. (2018)*, *Gao et al. (2020)*, *Qin et al. (2021)*, *Zhao et al. (2022)*, *Tseng & Gao (2022)* that are closely related to ours with regard to flexibility, scalability, privacy, and decentralization across the following assessment criteria:

1. Attribute authority: Whether the authorities involved in CP-ABE schemes are divided into single thus central authority or multi-authority.
2. Policy: LSSS which supports AND gate, OR gate, and threshold gate *versus* only AND.
3. Attribute universe: We define the complete set of supported attributes as an attribute universe and only take into account two types of the universe: the large universe and the small universe. In large universe ABE, the attribute universe size has no effect on the size of the system's public key.
4. Privacy: There are two aspects of privacy involved in CP-ABE schemes: policy-hiding and receiver-hiding. For the policy-hiding scheme, the CP-ABE system is available in two forms: fully hidden and partially hidden. The former means that none of the attributes can be revealed from the access policies, whereas the latter refers to only hiding sensitive attribute values in the access policies. For the receiver-hiding scheme, it prevents any *AA*s from learning the full set of attributes the receiver (*i.e.,* the *DU*) possesses, hence relieving the *DU* from disclosing them while requesting attribute keys.
5. Storage: From a technical perspective, traditional cloud service provider (CSP) and decentralized storage systems such as IPFS, Storj, and Sia, are two distinct popular

solutions for data storage and sharing. CSPs may take advantage of their comprehensive control over data, but end users are exposed to the risks of a single point of failure, privacy violation, and censorship.

6. Access control: We indicate whether access control enforcement is through a smart contract and thus logically decentralized or by a cloud service provider and thus logically centralized.

From Table 1, we observe that very few schemes (*Li et al., 2022*; *Yang et al., 2018*; *Zhao et al., 2022*) achieve fine-grained access control and support multi-authority with privacy-preserved characteristics, such as policy-hiding and receiver privacy. However, they all rely on a trusted third party (TTP) or cloud service provider (CSP) to offer centralized storage and access control management and are thus susceptible to the inherent vulnerabilities of such centralized systems in terms of privacy issues. In contrast, our proposed scheme stands out by leveraging smart contracts for access control management and integrating with the IPFS network for storage to realize an architecture with completely decentralized data storage and governance.

PRELIMINARY

To initiate, we revisit certain foundational principles employed within our system. A summary of crucial notations utilized throughout the manuscript is provided in Table 2.

Bilinear mapping

Consider $\mathcal{G}$ as an algorithm that accepts a security parameter λ and constructs three multiplicative cyclic groups of prime order p: $\mathbb{G}_1 = \langle g_1 \rangle, \mathbb{G}_2 = \langle g_2 \rangle$, and $\mathbb{G}_T$. We introduce $\hat{e}$ as a bilinear map, with $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The bilinear map $\hat{e}$ has the following characteristics:

1. **Bilinearity**: for all $a, b \in \mathbb{Z}$, $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$.
2. **Non-degeneray**: $\hat{e}(g_1, g_2) \neq 1$.
3. **Computability**: for all $a, b \in \mathbb{Z}$, $\hat{e}(g_1^a, g_2^b)$ can be efficiently computed.

Auxiliary methods and definitions

We make an assumption of possessing an algorithm, denoted as COMMIT, which takes strings of arbitrary length as input and produces outputs as determined by a random oracle. While this assumption aids our security analysis, in practical implementations, we could use the BLAKE-2 hash function in place of COMMIT. For the inputs h that can not be mapped directly to integers, especially in the case of group elements, we represent them using byte-strings.

Additionally, we introduce several auxiliary methods to facilitate the verification procedure for certain special properties.

The following definitions and claims are first proposed in the work *Bowe, Gabizon & Green (2018)*.

Definition 1 Given a bilinear mapping $\hat{e} : \mathbb{G}_1 \setminus \{0\} \times \mathbb{G}_2 \setminus \{0\} \to \mathbb{G}_T$, elements $A, B \in \mathbb{G}_1 \setminus \{0\}$ and $C, D \in \mathbb{G}_2 \setminus \{0\}$. If $\hat{e}(A, D) = \hat{e}(B, C)$, we may use the term **SameRatio**$((A, B), (C, D))$ to represent this relation.

Table 2 Notation description.

Notation	Description
p	A prime number used for $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ and $\mathbb{Z}$
$\mathbb{G}_1, \mathbb{G}_2$	Two additive cyclic groups.
$\mathbb{G}_T$	A multiplicative cyclic group.
$\mathbb{Z}_p$	A set of integers with order p
λ	A security parameter for the input size
$\mathcal{U}$	A set of attribute authorities in the universe
$\mathcal{X}$	A set of attributes in the universe
$\mathcal{S}$	A set of attributes possessed by each attribute authority
$\mathcal{R}$	A set of attributes possessed by data user
n	The number of attribute authorities
l	The number of supported attributes
GID	A Data User's Global identifier
k	The parameter for the *k-lin* assumption, representing the linear independence of group elements.
PP	Public parameters for the use of Attribute-Based Encryption or Vector Commitment
α	A scalar used for generation of PP
e	A set of secret elements used for **Trusted Setup** or **Authority Setup**
h	A hash of committed elements in **Trusted Setup** of **Authority Setup**
π	A proof of knowledge for an element
$\boldsymbol{rp}_s$	A *s-pair* of the element s in group $\mathbb{G}_1$. The superscript 2 of $\boldsymbol{rp}_s^2$ represent *s-pair* elements in group $\mathbb{G}_2$
L	A list of *s-pair* consisting of all the committed group elements
(PK, SK)	A key pair which is used for ABE encryption
$\boldsymbol{X}, \boldsymbol{\tau}, \sigma$	A set of secret elements in SK
$\boldsymbol{A}, \boldsymbol{U}$	The secret exponents used in PP of ABE
K	A component of the attribute key for each individual attribute
sk	The consolidated secret key issued by an attribute authority. Given that an attribute authority can oversee multiple attributes, sk might comprise several K components
$\boldsymbol{x}$	A policy vector
$\boldsymbol{v}$	An attribute vector
$\boldsymbol{C}$	A Vector Commitment associated with a specific Data User, derived from its attribute vector and global identifier
m	A special message used in Vector Commitment
op	An opening proof to reveal the Vector Commitment
$o_i, o_{i,j}$	The elements in PP_{VC} where $i, j \in [n], i \neq j$
z	A secret exponent of group element o
$\boldsymbol{aux}$	A collection of message m
BPK	A public key registered in a blockchain
BSK	A private key registered in a blockchain

Algorithm 1 Determin if two pairs (A,B) and (C,D) have certain relationship

Require: $A,B \in \mathbb{G}_1 \backslash \{0\}$ and $C,D \in \mathbb{G}_2 \backslash \{0\}$

function SAMERATIO($(A,B),(C,D)$)
  if $\hat{e}(A,D) = \hat{e}(B,C)$ **then**
    return true
  else
    return false
  end if
end function

Definition 2 Given a bilinear mapping $\hat{e} : \mathbb{G}_1 \setminus \{0\} \times \mathbb{G}_2 \setminus \{0\} \to \mathbb{G}_T$, $s \in \mathbb{Z}_p^*$ and cyclic group of order p, an *s-pair* is a pair (A,B) such that $A,B \in \mathbb{G}_1$, or $A,B \in \mathbb{G}_2$; and $s \cdot A = B$. For such an *s-pair* (A,B) in $\mathbb{G}_1$ or $\mathbb{G}_2$, we may represent it using the notation $\boldsymbol{rp}_s$ or $\boldsymbol{rp}_s^2$ respectively.

Claim 1 ***SameRatio*** $((A,B),(C,D)) =$ *true if and only if there exists* s *such that* (A,B) *is an s-pair in* $\mathbb{G}_1$ *and* (C,D) *is an s-pair in* $\mathbb{G}_2$.

Finally, we can construct our special *s-pair* as follows.

Definition 3 Given a bilinear mapping $\hat{e} : \mathbb{G}_1 \setminus \{0\} \times \mathbb{G}_2 \setminus \{0\} \to \mathbb{G}_T$ and a matrix $s \in \mathbb{Z}_p^{l \times k}$, a special *s-pair* is a pair (A,B) such that $A,B \in \mathbb{G}_1^{l \times k}$ or $A,B \in \mathbb{G}_2^{l \times k}$; and

$$B[i,j] = A[i,j]^{s[i,j]}$$

For such a special *s-pair* (A,B) in $\mathbb{G}_1$ or $\mathbb{G}_2$, we may also denote it as $\boldsymbol{rp}_s$. Given that a vector can be considered a matrix with a single column, we can also use the notation $\boldsymbol{rp}_s$ to represent an *s-pair* when $s \in \mathbb{Z}_p^k$.

Assumptions

Given a bilinear mapping $\hat{e} : \mathbb{G}_1 \setminus \{0\} \times \mathbb{G}_2 \setminus \{0\} \to \mathbb{G}_T$ with associated generators $\{g_1, g_2, g_T\}$ and group order p, our work builds upon a variety of standard assumptions, which are detailed below.

Assumption 1 Symmetric External Diffie-Hellman (SXDH) assumption (*Ballard et al., 2005*). It is hard to distinguish $\mathcal{D}_0 = (g_1, g_2, g_1^a, g_1^b, g_1^{ab})$ from $\mathcal{D}_1 = (g_1, g_2, g_1^a, g_1^b, g_1^c)$ where $a,b,c \xleftarrow{\$} \mathbb{Z}_p$. This also holds to the tuples $\mathcal{D}_0' = (g_1, g_2, g_2^a, g_2^b, g_2^{ab})$ and $\mathcal{D}_1' = (g_1, g_2, g_2^a, g_2^b, g_2^c)$ in different group.

Assumption 2 K-Linear assumption (*Boneh, Boyen & Shacham, 2004*). It is hard to distinguish $\mathcal{D}_0 = (g_1, g_2, g_1^{a_1}, g_1^{a_2}, \ldots, g_1^{a_k}, g_1^{a_1 b_1}, g_1^{a_2 b_2}, \ldots, g_1^{a_k b_k}, g_1^{b_1 + b_2 + \cdots + b_k})$ from $\mathcal{D}_1 = (g_1, g_2, g_1^{a_1}, g_1^{a_2}, \ldots, g_1^{a_k}, g_1^{a_1 b_1}, g_1^{a_2 b_2}, \ldots, g_1^{a_k b_k}, g_1^c)$ where $a_1, \ldots, a_k, b_1, \ldots, b_k, c \xleftarrow{\$} \mathbb{Z}_p$. This also holds in the group $\mathbb{G}_2$.

The matrix $\boldsymbol{A}$ and vector $\boldsymbol{a}^\perp$ are defined as:

$$\boldsymbol{A} = \begin{bmatrix} \mathrm{diag}(a_1, a_2, \ldots, a_k) \\ \mathbf{1}^\intercal \end{bmatrix} \tag{1}$$

$$\boldsymbol{a}^{\perp} = \left(a_1^{-1}, a_2^{-1}, \ldots, a_k^{-1}, -1\right)^{\intercal} \tag{2}$$

where the a_i are sampled elements from $\mathbb{Z}_p$. These are constructed such that $\boldsymbol{A}^{\intercal}\boldsymbol{a}^{\perp} = 0$.

Assumption 3 Special k-Linear assumption (*Michalevsky & Joye, 2018*). Given a randomly generated matrix $\boldsymbol{A} \in \mathbb{Z}_p^{(k+1)\times k}$ and a vector $\boldsymbol{s} \in \mathbb{Z}_p^{(k+1)}$, the tuples $\mathcal{D}_0 = (g_1, g_2, g_1^{\boldsymbol{A}}, g_1^{\boldsymbol{As}})$ and $\mathcal{D}_1 = (g_1, g_2, g_1^{\boldsymbol{A}}, g_1^{\boldsymbol{s}})$ are computationally indistinguishable by any pol-ynomial-time $\mathcal{A}$. The structure of matrix $\boldsymbol{A}$ is described in Eq. (1) and the vector $\boldsymbol{a}$ is derived from $\boldsymbol{A}$ as detailed in Eq. (2).

Assumption 4 Square Computational Diffie-Hellman assumption (*Burmester, Desmedt & Seberry, 1998*). Given (g, g^a) for a random number a in a cyclic group $\mathbb{G}$ of order p, a *PPT* algorithm $\mathcal{A}$ outputs g^{a^2} with non-negligible probability.

Assumption 5 Knowledge of Coefficient assumption (*Bowe, Gabizon & Miers, 2017*). Given a string of arbitrary length h, and a uniformly chosen $C \in \mathbb{G}_2$ (independent of h), an efficient algorithm $\mathcal{A}$ exists that can randomly generate $B \in \mathbb{G}_1$ and $D \in \mathbb{G}_2$. Meanwhile, for the same inputs (C, h), there is an efficient deterministic algorithm $\mathcal{X}$ cable of extracting a scalar b. The probability that both are true: (1) $\mathcal{A}$ 'succeeds', meaning it satisfies the condition that SameRatio $((g_1, B), (C, D))$(2) $\mathcal{X}$ 'fails', meaning $B \neq g_1^b$is negligible.

Proof of knowledge

We adopt the well-established Schnorr identification protocol (*Schnorr, 1990*), utilizing it as our Non-interactive Zero-knowledge (NIZK) proof. Provided with an *s-pair* $rp_s = (A, B = s \cdot A)$ and a string h, we establish NIZK (Algorithm 2). This can serve as proof that the originator of the string h is aware of s in the *s-pair* rp_s.

Furthermore, we define VerifyNIZK (Algorithm 3), which checks the validity of the provided proof π.

Algorithm 2 Construct a proof of knowledge of s

Require: rp_s is an *s-pair*
Require: h is a string
1: **function** NIZK($rp_s = (A, B), h$)
2: $\alpha \xleftarrow{\$} \mathbb{Z}_p^*$
3: $R \leftarrow \alpha \cdot A$
4: $c \leftarrow$ **COMMIT**$(R||h) \in \mathbb{Z}_p^*$
5: $u \leftarrow \alpha + c \cdot s$
6: **return** $\pi = (R, u)$
7: **end function**

Algorithm 3 Verify a proof of knowledge of s

Require: rp_s is an *s-pair*
Require: h is a string
function VERIFYNIZK($rp_s = (A, B), \pi = (R, u), h$)
  $c \leftarrow$ **COMMIT**$(R||h) \in \mathbb{Z}_p^*$
  if $u \cdot A == R + c \cdot B$ **then**
    return true
  else
    return false
  end if
end function

Vector commitment

We ensure our attribute-hiding property through the utilization of a Vector Commitment scheme, as described in *Catalano & Fiore (2013)*. The summarized scheme is as follows:

Definition 4 This Vector Commitment system commits to an ordered sequence of attribute elements $\boldsymbol{v} = (v_1, v_2, \ldots, v_{l+1})$ as commitment $\boldsymbol{C}$, then opens it in a certain position of $\boldsymbol{v}$ to a corresponding attribute authority (AA), and finally proves that only authorized value existed in the previously supplied commitment $\boldsymbol{C}$. The system normally consists of four algorithms:

- **Key generation** $(1^\lambda, n) \rightarrow PP_{VC}$: This is a decentralized key generation (DKG) algorithm. It takes as input the security parameter λ and the number of attribute authorities, n, in the system, and outputs global public parameters $PP_{VC} = \{g_1, g_2, \{o_i\}, \{o_{i,j}\}\}$ where $i, j \in [n], i \neq j$. The element o_i is generated and published by AA_i. Following that, the elements $\{o_{i,j}\}$ can be issued by each AA_i based on the shared $\{o_i\}$.
- **Commitment** $(\boldsymbol{aux} = \{m_i\}_{i \in [n]}) \rightarrow \boldsymbol{C}$: This algorithm is run by a data user (DU). It takes as input the message m_i generated based on the authorized attributes from $AA_i, i \in [n]$, and outputs the commitment $\boldsymbol{C}$.
- **Open** $(m_i, i, \boldsymbol{aux}) \rightarrow op_i$: This algorithm is also run by a DU. It takes as input the auxiliary information $\boldsymbol{aux}$ and index i, and outputs the opening proof op_i.
- **Verify** $(\boldsymbol{C}, m_i, i, op_i) \rightarrow$ (1 or 0): The Verify algorithm is run by AA. It takes as inputs the commitment $\boldsymbol{C}$, message m_i, index i, and opening proof op_i, and outputs the result of the verification. It outputs 1 when it accepts the proof.

Decentralized inner-product predicate encryption

Definition 5 A multi-authority attribute-based encryption with policy-hiding scheme (*Michalevsky & Joye, 2018*) consists of a tuple of probabilistic polynomial-time (PPT) algorithms, such that:

- **Setup** $(1^\lambda) \rightarrow PP$: It takes as input the security parameter λ and then outputs the public parameters PP.
- **Authority setup** $(PP, i) \rightarrow (PK_i, SK_i)$: It takes as input public parameter PP and authority index i, and outputs a pair of authority keys (PK_i, SK_i) where $SK_i := \{\boldsymbol{X}, \boldsymbol{\tau}, \sigma\}$.

- **Key generation** $(PP, i, SK_i, \{PK\}, GID, \boldsymbol{v}) \rightarrow sk_{i,GID,\boldsymbol{v}}$: It takes as input the global public parameters PP, the authority index i, its secret key SK_i, all the public keys $\{PK_i\}_{i\in[n]}$, and DU's global identifier GID and the attribute vector $\boldsymbol{v}$, and outputs the secret keys $sk_{i,GID,\boldsymbol{v}} := \{K_j\}_{j\in\mathbb{S}_i}$.
- **Encryption** $(PP, \{PK\}, \boldsymbol{x}, F) \rightarrow CT_F$: It takes as inputs the global parameters PP, the public keys of all the authorities $\{PK_i\}_{i\in[n]}$, the ciphertext policy vector $\boldsymbol{x}$ and a file F, and outputs a ciphertext CT_F.
- **Decryption** $(\{sk_{i,GID,\boldsymbol{v}}\}_{i\in n}, CT_F) \rightarrow F$: It takes as inputs the collection of secret keys $\{sk_{i,GID,\boldsymbol{v}}\}$ from AA_i and the ciphertext CT_F, and outputs the message F if the access policy has been satisfied.

Blockchain technology

Our proposed system integrates blockchain technology with MA-ABE and vector commitment mechanisms to advance decentralization. Blockchain was initially conceptualized by *Nakamoto (2008)* in 2009. It eliminates the need for a trusted third party (TTP) to oversee data management, favoring a distributed ledger maintained by consensus nodes. This ledger keeps track of a chronologically ordered list of transactions, which are validated *via* a consensus algorithm, such as proof of work (POW), before being permanently added to the ledger. In the Bitcoin system, miners solve complex cryptographic puzzles—a process known as PoW—to add blocks to the blockchain by packaging new transactions.

Ethereum (*Wood, 2014*), an evolution from the foundational Bitcoin, introduced a new platform for decentralized application with two distinct account types: external owned accounts (EOA) for the standard transaction and contract accounts for deploying self-executing and self-verifying protocols known as smart contracts. Each EOA is associated with a 20-byte hexadecimal address derived from the user's public key (BPK), and transactions are authorized using the corresponding private key (BSK).

To address the storage scalability challenges of blockchains like Bitcoin and Ethereum (*Nakamoto, 2008*; *Wood, 2014*), the interplanetary file system (IPFS) (*Benet, 2014*) was developed. It is a peer-to-peer distributed file system offering content-addressed high-throughput storage, akin to a decentralized cloud service. When data is uploaded to IPFS, a unique hash of the file is generated, enabling users to access their data similarly to how URLs work on the traditional web.

SYSTEM OVERVIEW

System architecture

The system comprises the following logical entities:

Data owner (DO): DO is an entity (individual or organization) that owns a certain file F. For secure storage and sharing, DO encrypts F using the AES key AK and uploads the encrypted file CT_F to the IPFS network, records the returned file location loc, and embeds AK and loc into the metadata M which is subsequently encrypted using the ABE system and published CT_M in the Ethereum network.

Data user (DU): DU is a data client for DO. It asks the attribute authority AA for permission to get the necessary attribute secret keys $\{sk\}$, which are then used to decrypt the associated CT_M stored on the Ethereum network. After getting the key AK and the location loc from M, DU can download the encrypted file CT_F from the IPFS network and recover the original file F.

Attribute authority (AA): AA is an entity (individual or organization) that contributes to the generation of the public parameters of the ABE system PP_{ABE} and the vector commitment scheme PP_{VC}, publishes the public key PK for the DO to encrypt metadata M, owns a set of attributes and issues secret key sk for the owned attributes upon the request of the DU.

Trusted attributed authority (AA_{trust}): AA_{trust} is a trusted attribute authority that mainly generates a secret key sk for DU and deploys system contracts for setup and registration. AA_{trust}, unlike normal AA, owns no attributes but is in charge of a specific position in the attribute vector $\boldsymbol{\nu}$. It is important to note that, similar to our proposal, the scheme in *Michalevsky & Joye (2018)* also includes a AA_{trust}. While this introduces a central element into the system, it is necessary for certain administrative tasks and security assurances.

Service user (SU): In the system, SU is a general entity comprising DO, DU and AA.

Participant (P): P is a special entity that represents each AA during the process of **Trusted Setup**. The index i of P_i denotes the chronological order of each piece of public parameter generated and shared by AA. **Blockchain**: Each user (DO, DU, AA and AA_{trust}) possesses a pair of keys (BPK, BSK) and a corresponding wallet address $addr$ on the blockchain. Our system employs two blockchains: IPFS for data storage and Ethereum for data governance.

and contracts:

Trust setup contract (SC_{sys}): The contract SC_{sys} is deployed to the Ethereum network by the AA_{trust} and can only be invoked by an authorized AA within the time window specified. It is responsible for generating the global public parameters PP_{ABE}.

Authority setup contract (SC_{auth}): Contract SC_{auth} is deployed to the Ethereum network by the AA_{trust}. It can only be invoked by the authorized AA within the specified time window. It is used to generate the global public parameters PP_{VC} and to keep track of the valid information about AA's address $addr$, public key PK, and supported attributes $\mathcal{S}$.

User registration contract (SC_{reg}): Contract SC_{reg} is deployed to the Ethereum network by the AA_{trust} and can be invoked by all the potential DUs. To register the $addr$ in the system, DU needs to make sufficient payment to the SC_{reg} and then get back the GID which can later be used to request secret key sk from AA.

Utility contract (SC_{util}): Contract SC_{util} is deployed to the Ethereum network by the AA_{trust} and can only be invoked by other contracts deployed by AA_{trust}. It is mainly used to verify group elements published by AA.

Log contract (SC_{log}): Contract SC_{log} is deployed to the Ethereum network by the AA_{trust}. When it receives a new transaction from DO, it records the encrypted data CT_M of metadata M and triggers the event to the subscribers.

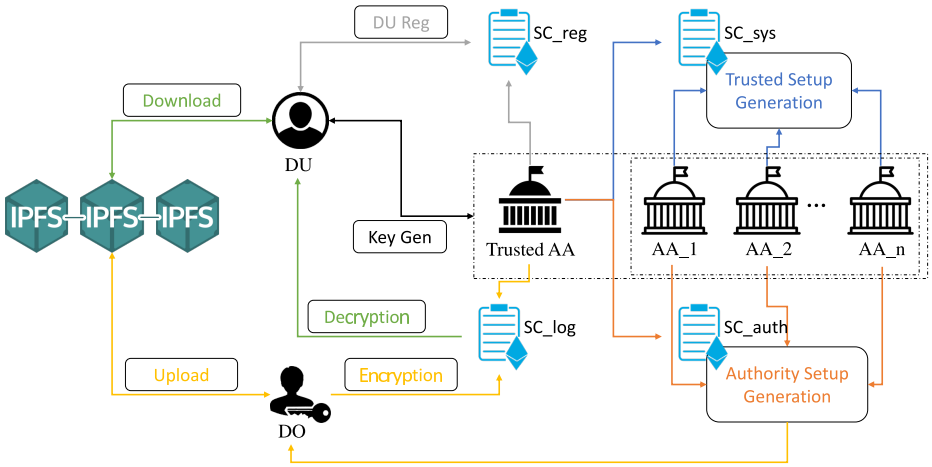

Figure 1 The system consists of six processes, each of which is represented by a different color: Blue for the process Trusted Setup, orange for the process Authority Setup, gray for the process Data User Registration, black for the process Key Generation, yellow for the process Encryption and Upload, and green for the process Download and Decryption. Single or double-arrow connectors indicate the interactions between service users and two blockchain networks, Ethereum and IPFS. Note that these four contracts deployed on Ethereum are used for data governance, while IPFS is used for data storage. For a detailed description of the system flow between smart contracts, IPFS, and various entities, please check "System Overview" for the Interaction Overview and "System Design" for the System Design.

Full-size DOI: 10.7717/peerjcs.2581/fig-1

Architecture

The system architecture is shown in Fig. 1.

Interactions overview

In this section, we describe the overview of our proposed system to show how smart contracts, IPFS, vector commitment, and MA-ABE with policy-hiding are composed together to build a secure, privacy-preserving, and blockchain-enabled data governance system. When it ought to be clear from the context, we omit most indices, like i and j of elements, and superscript in $\boldsymbol{rp}_s^2$ for readability.

1. **Trusted setup**
   1 First, a community of normal attribute authorities (AAs) with size $n-1$ and a special trusted attribute authority (AA_{trust}) must be determined. AA_{trust} selects the security parameter λ and two generators g_1, g_2 for the bilinear mapping, and defines following algorithms: COMMIT, NIZK, VERIFYNIZK and POWERMULTI.
   2 AA_{trust} deploys one system contract SC_{sys} and one utility contract SC_{util}.
   3 Each AA randomly samples a set of secret elements e: two matrixes $\boldsymbol{A} \in \mathbb{Z}_p^{(k+1)\times k}$ and $\boldsymbol{U} \in \mathbb{Z}_p^{(k+1)\times(k+1)}$, two scalars $\alpha_{\boldsymbol{A}}$ and $\alpha_{\boldsymbol{U}}$ in $\mathbb{Z}_p^*$, and two scaled matriex $\alpha_{\boldsymbol{A}} \cdot \boldsymbol{A}$ and $\alpha_{\boldsymbol{U}} \cdot \boldsymbol{U}$, and publishes a corresponding set of *s-pair* $\{\boldsymbol{rp}_{\boldsymbol{A}}, \boldsymbol{rp}_{\boldsymbol{U}}, \boldsymbol{rp}_{\alpha_{\boldsymbol{A}}}, \boldsymbol{rp}_{\alpha_{\boldsymbol{U}}}, \boldsymbol{rp}_{\alpha\cdot\boldsymbol{A}}, \boldsymbol{rp}_{\alpha\cdot\boldsymbol{U}}\}$ as defined in Definitions 2 and 3 to the contract SC_{sys}.
   4 After that, AA computes and publishes the commitments $h := COMMIT(\{h_s\}||)$ to SC_{sys}, where $h_s := COMMIT(\boldsymbol{rp}_s), s \in e$.

5 Every AA then needs to prove the knowledge of each element $s \in e$ by outputting the proofs $\{\pi_s\}$ using algorithm NIZK (Algorithm 2) as the argument of function Prove of contract SC_{sys}, which verifies them using algorithm VerifyNIZK (Algorithm 3).
6 In the **Round 1**, we define one attribute authority AA as participant P_1 who firstly publishes group elements in $\mathbb{G}_1$: $(V_1 := g_1^{\boldsymbol{A}_1}, \theta_{V_1} := g_1^{\alpha_{A_1}}, V_1' := g_1^{\alpha_{A_1} \cdot \boldsymbol{A}_1})$, based on the previously verified set of elements e.
7 Participant $P_{i=2,\ldots,n}$ computes $(V_i, \theta_{V_i}, V_i')$ based on previous $(V_{i-1}, \theta_{V_{i-1}}, V_{i-1}')$ using algorithm powerMult (Alg. 6) and publishes these as the arguments of the function Compute of contract SC_{sys} to check validity.
8 We define the last valid V received by contract SC_{sys} as one piece of the public parameter $g_1^{\boldsymbol{A}}$.
9 In the **Round 2**, the first AA, also known as participant P_1, publishes group elements in $\mathbb{G}_1$: $(W_1 := ((g_1^{\boldsymbol{A}})^{\intercal})^{\boldsymbol{U}_1}, \theta_{W_1} := g_1^{\alpha_{U_1}}, W_1\prime := ((g_1^{\boldsymbol{A}})^{\intercal})^{\alpha_{U_1}\boldsymbol{U}_1})$, also based on the previously verified set of elements e.
10 Participant P_i, where $i = 2, \ldots, n$ computes its $(W_i, \theta_{W_i}', W_i')$ based on previous $(W_{i-1}, \theta_{W_{i-1}}', W_{i-1}')$ using algorithm powerMult and publishes these as the arguments of the function Generate.
11 We also define the last valid W received by contract SC_{sys} as last piece of the public parameter $g_1^{\boldsymbol{U}^{\intercal}\boldsymbol{A}}$. Therefore, we have $PP_{ABE} := \{g_1, g_2, g_1^{\boldsymbol{A}}, g_1^{\boldsymbol{U}^{\intercal}\boldsymbol{A}}\}$.

2. **Authority setup**
1 AA_{trust} deploys contract SC_{auth} for authority setup.
2 Each AA randomly samples another set of secret element e': a matrix $\boldsymbol{X} \in \mathbb{Z}_p^{(k+1)\times(k+1)}$, a vector $\boldsymbol{\tau} \in \mathbb{Z}_p^{k+1}$, two numbers $\sigma, z \in \mathbb{Z}_p^*$, a scalar α_z and a scaled number $\alpha_z \cdot z$. Using that, AA takes $SK := \{\boldsymbol{X}, \boldsymbol{\tau}, \sigma\}$ as secret keys, and publishes a corresponding set of *s-pair* $\{\boldsymbol{rp}_{\boldsymbol{X}}, \boldsymbol{rp}_{\boldsymbol{\tau}}, \boldsymbol{rp}_{\sigma}, \boldsymbol{rp}_{z}, \boldsymbol{rp}_{\alpha_z \cdot z}\}$ to the contract SC_{auth}.
3 After that, AA computes and publishes the commitment $h' := \text{COMMIT}(\{h_s\}||)$ to SC_{auth}, where $h_s := \text{COMMIT}(\boldsymbol{rp}_s), s \in e'$.
4 Every AA then needs to prove the knowledge of elements $s \in e'$ by generating the proofs $\{\pi_s\}$ using algorithm **NIZK** as the argument of function Prove of contract SC_{auth} for validity check.
5 We define each AA with index $i \in [n-1]$ based on the receiving order of the complete set of valid $\{\pi_s\}_{s \in e'}$ and set attribute authority AA_{trust} with index n.
6 Therefore, we have the verified sets of elements $PK_i := \{g_1^{\boldsymbol{X}_i^{\intercal} \cdot \boldsymbol{A}}, \hat{e}(g_1^{\boldsymbol{\tau}_i^{\intercal}\boldsymbol{A}}, g_2), g_2^{\sigma}\}$ and $\{o_i := g_1^{z_i}, g_1^{\alpha_{z_i}}, g_1^{\alpha_{z_i} \cdot z_i}\}$ for each AA_i.
7 In the last stage, for each $i, j \in [n], j \neq i$, AA_i needs to compute a set of group elements in $\mathbb{G}_1 : (O_i := \{(o_j)^{z_i}\}, \theta_{O_i} := \{(g_1^{\alpha_{z_j}})^{\alpha_{z_i}}\}, O_i' := \{(g_1^{\alpha_{z_j} \cdot z_j})^{\alpha_{z_i} \cdot z_i}\}$. Then AA_i publishes these elements, with the number of supported attributes l_i as the argument of the function Setup.
8 The contract SC_{auth} checks the validity of these elements published by $AA_i, i \in [n]$ and then registers its address $addr_i$ with the elements (l_i, PK_i).
9 In the end, we have $PP_{VC} := \{g_1, g_2, \{o_i\}_i, \{o_{i,j}\}\}$ where $i, j \in [n], i \neq j$ for vector commitment scheme.

3. **Data user registration**
   1 AA_{trust} deploys contract SC_{reg} for service registration.
   2 Data User (DU) makes a direct registration payment to the contract SC_{reg} to get the global identifier GID which is the hash value of DU's address $addr$.
   3 Afterwards, DU can setup a secure channel with each $AA_i, i \in [n]$ that possesses the needed attributes and can verify DU's identity.
   4 AA_i verifies DU's identity and sends back the set of acknowledged attributes $\mathcal{R}_{i,GID}$ through the secure channel.
   5 Upon receiving all the $\mathcal{R}_{i,GID}$ from AA_i, DU defines a set of 'N/A' attributes $\mathcal{R}_{j,GID}$ for those $AA_j, i \neq j$ can not issue the attribute set and finally gets a complete set of attributes $\mathcal{R}_{GID}$ by combing $\mathcal{R}_{i,GID}$ and $\mathcal{R}_{j,GID}$ together.
4. **Key Gen**
   1 DU generates an attribute vector $\boldsymbol{v}_{GID}$ from set of attributes $\mathcal{R}_{GID}$, creates a vector commitment $\boldsymbol{C}$ for $\boldsymbol{v}_{GID}$ and sends it with opening proof op_i to each $AA_i, i \in [n]$ through separate secure channels.
   2 $AA_i, i \in [n]$ firstly checks the validity of its responsible part in the commitment $\boldsymbol{C}$ using op_i, then issues DU's requested attribute secret key $sk_{i,GID,\boldsymbol{C}}$, and finally sends it back to DU through the channel.
   3 Upon receiving responses from each $AA_i, i \in [n]$, DU gets a complete set of secret keys $\{sk_{i,GID,\boldsymbol{C}}\}_{i\in[n]}$.
5. **Encryption and upload**
   1 AA_{trust} deploys last system contract SC_{log} to record encrypted related information of file F
   2 Data Owner (DO) randomly samples an AES key AK, encrypts F to obtain the ciphertext CT_F, and uploads it to the IPFS network.
   3 After successfully receiving the CT_F from DO, IPFS network returns a special hash value loc as a file location on the IPFS network.
   4 Then, DO constructs a metadata $M := (K, loc)$, specifies a policy vector $\boldsymbol{x}$ based on selected attributes from each AA_i, uses published $\{PK_i\}$ to encrypt the metadata M and publishes this encrypted information CT_M to contract SC_{log}.
6. **Download and decryption**
   1 DU reads every new coming CT_M from the contract SC_{log} and checks if its owned secret keys $\{sk_{i,GID,\mathrm{C}}\}$, where $i \in [n]$, satisfies the access policy $\boldsymbol{x}$ to recover the metadata M.
   2 DU retrieves the file location loc and AES key AK from the metadata M and requests the ciphertext CT_F from the IPFS network with the file location loc.
   3 DU uses the AES key AK to recover the original file F.

SYSTEM DESIGN

In this section, we provide more details on the processes of **Trusted Setup**, **Authority Setup**, **Data User Registration**, **Key Generation**, **Encryption and Upload**, and **Download and Decryption**.

The code for the MA-ABE scheme with Non-interactive Zero-Knowledge Proof in **Auth Setup** is available on GitHub: https://github.com/Guy1m0/Attack-on-IPPE, but it does not include the code for any contracts, since the actual implementation may vary based on the version of the Solidity compiler used, which might affect the performance and gas cost of each contract.

Before the start of **Trusted Setup**, the trusted authority (AA_{trust}) deploys the utility contract SC_{util}.

Trusted Setup

The process of **Trusted Setup** consists of four stages: *Initiate*, *Commit and Reveal*, *Verify*, and *Generate*, and finally outputs the global public parameter PP_{ABE} for the ABE system.

In the initial three stages, each attribute authority (AA) sends its transactions independently to the contract SC_{sys}. In contrast, during the final *Generate* stage, each participant P (where we use the placeholder notation P in *Commit and Reveal* to represent each AA) must send transactions to SC_{sys} in a sequential manner. This sequentiality is necessary because each incoming transaction is generated based on the preceding P's transaction received by contract SC_{sys}.

Initiate

At the start, a fixed-numbered community of size n will be determined, which will include all of the normal attribute authorities AA and one special trusted authority AA_{trust}. AA_{trust} represents this community to set the global security parameter to be λ and the generators of the $\mathbb{G}_1, \mathbb{G}_2$ with prime order p to be g_1, g_2 respectively. Therefore, the bilinear map can be $\hat{e} : \mathbb{G}_1 \setminus \{0\} \times \mathbb{G}_2 \setminus \{0\} \rightarrow \mathbb{G}_T$.

AA_{trust} also deploys two distinct system contracts: contract SC_{sys} for trusted setup and contract SC_{util} for resolving the problem of allowing complex cryptographic computations to be used in the system. AA_{trust} also specifies the deadlines (ddl_1, ddl_2, ddl_3) for the **Trusted Setup** process and sets an authorized list *AAlist* to restrict SC_{sys} access. For a simple system description, we assume that each attribute authority AA submits the required transactions within the deadlines.

To realize the generation of PP_{ABE}, this process highly depends on the interaction between each attribute authority AA and contract SC_{sys}, which has five main functions, Commit, Reveal, Prove, Compute and Generate with the help from contract SC_{utils}. Generally, these functions can only be invoked by a blockchain address owned by AA, which is included in the authorized list *AAlist*, and executed before the pre-defined deadline $ddl_{1,2,3}$.

[1] The value of k in this context does not derive from the security parameter λ. Rather, it is a reference to Assumption 3.

Commit and reveal

Every AA randomly picks a set of elements e: a matrix $\boldsymbol{A} \xleftarrow{\$} \{\text{Diagonal matrices in } \mathbb{Z}_p^{k\times k}\} \cup \left(\mathbf{1}\right)$, a matrix $\boldsymbol{U} \xleftarrow{\$} \mathbb{Z}_p^{(k+1)\times(k+1)}$[1], their corresponding scalar values, $\alpha_{\boldsymbol{A}}$ and $\alpha_{\boldsymbol{U}}$, and scaled matrixes $\alpha_{\boldsymbol{A}} \cdot \boldsymbol{A}, \alpha_{\boldsymbol{U}} \cdot \boldsymbol{U}$. Then, AA has

$$e = \{\boldsymbol{A}, \boldsymbol{U}, \alpha_{\boldsymbol{A}}, \alpha_{\boldsymbol{U}}, \alpha_{\boldsymbol{A}} \cdot \boldsymbol{A}, \alpha_{\boldsymbol{U}} \cdot \boldsymbol{U}\} \tag{3}$$

and then generates a set of *s-pair*.

For such element $s \in e$, we refer to the *s-pair* in $\mathbb{G}_1$ by $\boldsymbol{rp}_s$ and in $\mathbb{G}_2$ by $\boldsymbol{rp}_s^2$ as Definition 2. These *s-pair* in both $\mathbb{G}_1$ and $\mathbb{G}_2$ are listed as follows:

- For matrix $\boldsymbol{A}$: $(\boldsymbol{rp}_{\boldsymbol{A}}, \boldsymbol{rp}_{\boldsymbol{A}}^2) = (g, g^{\boldsymbol{A}})$
- For matrix $\boldsymbol{U}$: $(\boldsymbol{rp}_{\boldsymbol{U}}, \boldsymbol{rp}_{\boldsymbol{U}}^2) = (g^{\boldsymbol{A}^\intercal}, g^{\boldsymbol{A}^\intercal \boldsymbol{U}})$
- For scalar α_A: $(\boldsymbol{rp}_{\alpha_A}, \boldsymbol{rp}_{\alpha_A}^2) = (g, g^{\alpha_A})$
- For scalar α_U: $(\boldsymbol{rp}_{\alpha_U}, \boldsymbol{rp}_{\alpha_U}^2) = (g, g^{\alpha_U})$
- For scaled matrix $\alpha_A \boldsymbol{A}$: $(\boldsymbol{rp}_{\alpha_A \boldsymbol{A}}, \boldsymbol{rp}_{\alpha_A \boldsymbol{A}}^2) = (g, g^{\alpha_A \cdot \boldsymbol{A}})$
- For scaled matrix $\alpha_U \boldsymbol{U}$: $(\boldsymbol{rp}_{\alpha_U \boldsymbol{U}}, \boldsymbol{rp}_{\alpha_U \boldsymbol{U}}^2) = (g^{\boldsymbol{A}^\intercal}, g^{\alpha_U \cdot \boldsymbol{A}^\intercal \boldsymbol{U}})$

Other than these *s-pair* listed above, *AA* also commits each element $s \in e$. For each $s \in e$:

$$h_s := \text{COMMIT}(\boldsymbol{rp}_s || \boldsymbol{rp}_s^2)$$

Subsequently, the overall commitment is:

$$h := \text{COMMIT}(h_{\boldsymbol{A}} || h_{\boldsymbol{U}} || h_{\boldsymbol{\alpha_A}} || h_{\alpha_U} || h_{\alpha_A \boldsymbol{A}} || h_{\alpha_U \boldsymbol{U}}) \tag{4}$$

After that, *AA* publishes the commitment h to the contract SC_{sys} through blockchain transaction by calling function Commit, which works similarly as a hash function.

The state variable *h_collector* of SC_{sys} will store the value h with the key as *msg.sender*, also known as *AA*'s blockchain address. Apart from *h_collector*, we define few state variables used in contract SC_{sys} as follows:

1. *h_collector* (State Variable): A mapping collection from the blockchain address belonged to one attribute authority to its commitment h
2. *unverified_elements* (State Variable): A mapping collection from the blockchain address belonged to one attribute authority to its unverified list of *s-pair* s $L = \{(\boldsymbol{rp}_s, \boldsymbol{rp}_s^2) | s \in e\}$ in both $\mathbb{G}_1$ and $\mathbb{G}_2$
3. *verified_elements* (State Variable): A mapping collection from the blockchain address belonged to one attribute authority to its verified list of *s-pair* s $L = \{(\boldsymbol{rp}_s, \boldsymbol{rp}_s^2) | s \in e\}$ in both $\mathbb{G}_1$ and $\mathbb{G}_2$

After h has been received by contract SC_{sys}, the sender needs to reveal committed element $s \in e$ by passing a list of *s-pair* in both $\mathbb{G}_1$ and $\mathbb{G}_2$

$$L_{rp} = \{(\boldsymbol{rp}_s, \boldsymbol{rp}_s^2) | s \in e\} \tag{5}$$

as argument of the function Reveal (Algorithm 4) before deadline ddl_1, which checks the existence of the h published by *msg.sender*, and verifies that indeed $h = COMMIT(\{h_s\}||)$[2] as follows:

Finally, each pair $(\boldsymbol{rp}_s, \boldsymbol{rp}_s^2), s \in e$ will be stored in another state variable *unverified_elements* with the key as *msg.sender*.

[2]For brevity, we will use this shorthand notation to represent the above concatenation where s takes on all value in the set $e = \{\boldsymbol{A}, \boldsymbol{U}, \alpha_A, \alpha_U, \alpha_A \boldsymbol{A}, \alpha_U \boldsymbol{U}\}$.

Verify

After deadline ddl_1 set by AA_{trust} in the first stage *Initiate*, the system enters into the stage *Verify*. In this stage, we need to check that each attribute authority *AA* possesses the knowledge of the exponent s used in the list of *s-pair* L.

Algorithm 4 Contract SC_{sys}: Part 1

```
1:  function REVEAL(L_rp)
2:      for all (a, b) ∈ L_rp do
3:          h ← h|| HASH(a) || HASH(b)
4:      end for
5:      if msg.sender ∉ unverified_elements then          ▷ Resubmitting check
6:          unverified_elements[msg.sender] ← L_rp
7:      end if
8:  end function
9:
10: function PROVE(L_π)
11:     L_rp ← unverified_elements[msg.sender]
12:     for i ← 0, 5 do
13:         (rp, rp2) ← L_rp[i]
14:         pi ← L_π[i]
15:         if not SAMERATIO(rp, rp2) then
16:             throw
17:         end if
18:         h_tmp ← h_collector[msg.sender]||HASH(rp_s)
19:         if not CHECKPOK(rp_s, π_s, h_tmp) then
20:             throw
21:         end if
22:     end for
23:     verified_elements[msg.sender] ← L_rp
24:     unverified_elements[msg.sender] = []
25: end function
```

Every *AA* generates the proof $\pi_s := NIZK(\boldsymbol{rp}_s, h||h_s)$ using Algorithm 2 for each $s \in e$, and broadcasts these proofs as a list

$$L_\pi = \{\pi_{\boldsymbol{A}}, \pi_U, \pi_{\alpha_A}, \pi_{\alpha_U}, \pi_{\alpha_A \cdot \boldsymbol{A}}, \pi_{\alpha_U \cdot U}\} \tag{6}$$

through a blockchain transaction to get them verified. The function PROVE (Algorithm 4) from contract SC_{sys} takes input L_π and processes this verification work.

As shown above, it firstly calls function SAMERATIO, similar to Algorithm 1, of contract SC_{util} to examine the authenticity of the published $\boldsymbol{rp}_s$ and $\boldsymbol{rp}_s^2$.

Afterwards, it computes $h_{tmp} := h||COMMIT(\boldsymbol{rp}_s)$, and takes h_{tmp} with verified $\boldsymbol{rp}_s$ and provided π_s as input to the function CHECKPOK of contract SC_{util}, which works similarly as Algorithm 3 and returns *true* if the given proof π_s is valid.

Finally, the function PROVE (Algorithm 4) will remove the list of *s-pair* L_{rp} from the state variable *unverified_elements* and store it in the state variable *verified_elements*. This indicates that the *AA* possesses knowledge of the exponents for the set of *s-pair*.

Compute and generate

In this stage, the system will generate the public parameters for the attribute-based encryption in two rounds by interacting with two functions COMPUTE and GENERATE of contract SC_{sys} (Algorithm 5). Some state variables used are defined as follows:

1. V_curr (State Variable): The most recent published value V
2. V_curr' (State Variable): The most recent published value V'
3. θ_curr (State Variable): The most recent published scalar θ
4. W_curr (State Variable): The most recent published value W
5. W_curr' (State Variable): The most recent published value W'
6. $\theta\_curr\_$ (State Variable): The most recent published scalar θ

We also use below notation POWERMULTI (A, B) for the following Algorithm 6:

Round 1: We define the first attribute authority AA as participant P_1, who broadcasts $(V_1, \theta_{V_1}, V_1')$ as argument of function COMPUTE (Algorithm 5) in contract SC_{sys}. The elements $(V_1, \theta_{V_1}, V_1')$ are constructed as follows: $V_1 := g_1^{\boldsymbol{A}_1}, \theta_{V_1} := g_1^{\alpha_{\boldsymbol{A}_1}}, V_1' := g_1^{\alpha_{\boldsymbol{A}_1} \cdot \boldsymbol{A}_1}$. And the next participant P_i, $i = 2, 3, \ldots, n$, generates corresponding elements $(V_i, \theta_{V_i}, V_i')$ using Algorithm 6, and also broadcasts them to the contract SC_{sys}:

$$V_i := \text{POWERMULTI}(V_{i-1}, \boldsymbol{A}_i)$$

$$\theta_{V_i} := (\theta_{V_{i-1}})^{\alpha_{\boldsymbol{A}_i}}$$

$$V_i' := \text{POWERMULTI}(V_{i-1}', \alpha_{\boldsymbol{A}_i} \boldsymbol{A}_i)$$

Since receiving the elements (V_1, θ_1, V_1') from first participant P_1, function COMPUTE (Algorithm 5) of SC_{sys} checks the validity of each incoming elements $(V_i, \theta_{V_i}, V_i')$ published by P_i.

In the end, we define the last valid V_i as one piece of the public parameter:

$$g_1^{\boldsymbol{A}} = \text{POWERMULTI}(\text{POWERMULTI}(\ldots(\text{POWERMULTI}(\text{POWERMULTI}(g_1^{\boldsymbol{A}_1}, \boldsymbol{A}_2), \boldsymbol{A}_3)\ldots, \boldsymbol{A}_n).$$

Round 2: In this round, we also define the first attribute authority AA as participant P_1, who broadcasts (W_1, θ_1', W_1') as argument of the function GENERATE (Algorithm 5) in contract SC_{sys}. The elements (W_1, θ_1', W_1') are constructed as follows: $W_1 := ((g_1^{\boldsymbol{A}})^{\intercal})^{\boldsymbol{U}_1}, \theta_1' := g_1^{\alpha_{\boldsymbol{U}_1}}, W_1' := ((g_1^{\boldsymbol{A}})^{\intercal})^{\alpha_{\boldsymbol{U}_1} \boldsymbol{U}_1}$. And the next participant P_i where $i = 2, 3, \ldots, n$, generates corresponding elements $(W_i, \theta_{W_i}, W_i')$ using Algorithm 6, and also broadcasts them to the contract SC_{sys}:

$$W_i := \text{POWERMULTI}(W_{i-1}, \boldsymbol{U}_i)$$

$$\theta_{W_i} := (\theta_{W_{i-1}})^{\alpha_{\boldsymbol{U},i}}$$

$$W_i' := \text{POWERMULTI}(W_{i-1}', \alpha_{\boldsymbol{U},i} \boldsymbol{U}_i)$$

Algorithm 5 Contract SC_{sys}: Part 2

function COMPUTE(V, θ, V')
  if $unverified\_elements[msg.sender] \neq []$ OR $block.timestamp < ddl_2$ OR $block.timestamp > ddl_3$ **then**
    throw
  end if
  if not V_curr **then** ▷ Initialize V, θ, V'
    $V\_curr \leftarrow V$
    $\theta\_curr \leftarrow \theta$
    $V\_curr' \leftarrow V'$
    return
  end if
  $(rp_A, rp_{\alpha_A}, rp_{\alpha_A A}) \leftarrow verified\_elements[msg.sender]$
  $acc_1 \leftarrow$ SAMERATIO($(V\_curr, V), rp_A$)
  $acc_2 \leftarrow$ SAMERATIO($(\theta\_curr, \theta), rp_{\alpha_A}$)
  $acc_3 \leftarrow$ SAMERATIO($(V\_curr', V'), rp_{A\alpha_A}$)
  if $acc_1 == acc_2 == acc_3 ==$ true **then** ▷ Check validity
    $V\_curr \leftarrow V$
    $\theta\_curr \leftarrow \theta$
    $V\_curr' \leftarrow V'$
  else
    throw
  end if
end function

function GENERATE(W, θ, W')
  if $unverified\_elements[msg.sender] \neq []$ OR $block.timestamp < ddl_2$ OR $block.timestamp > ddl_3$ **then**
    throw
  end if
  if not W_curr **then** ▷ Initialize W, θ, W'
    $W\_curr \leftarrow W$
    $\theta\_curr\_ \leftarrow \theta$
    $W\_curr' \leftarrow W'$
    return
  end if
  $(rp_U, rp_{\alpha_U}, rp_{\alpha_U U}) \leftarrow verified\_elements[msg.sender]$
  $acc_1 \leftarrow$ SAMERATIO($(W\_curr, W), rp_U$)
  $acc_2 \leftarrow$ SAMERATIO($(\theta\_curr\_, \theta), rp_{\alpha_U}$)
  $acc_3 \leftarrow$ SAMERATIO($(W\_curr', W'), rp_{\alpha_U U}$)
  if $acc_1 == acc_2 == acc_3 ==$ true **then** ▷ Check validity
    $W\_curr \leftarrow W$
    $\theta\_curr\_ \leftarrow \theta$
    $W\_curr' \leftarrow W'$
  else
    throw
  end if
end function

Algorithm 6 Computing power matrix A by matrix B

Require: group elements A and matrix s have same size $l \times k$

function POWERMULTI(A, s)
  for $i \leftarrow 1, l$ **do**
    for $j \leftarrow 1, k$ **do**
      $B[i,j] \leftarrow A[i,j]^{s[i,j]}$
    end for
  end for
  return B
end function

Invoked by these transactions, contract SC_{sys} checks the validity of each received elements $(W_i, \theta_{W_i}, W_i')$ and updates the value of W_i.

As a result of this procedure, the final piece of the public parameter is defined as the last valid W_i received:

$$g_1^{\boldsymbol{U}^\intercal \boldsymbol{A}} = W_n = \textsc{PowerMulti}(\textsc{PowerMulti}(\ldots(\\ \textsc{PowerMulti}(\textsc{PowerMulti}(((g_1^{\boldsymbol{A}})^\intercal)^{\boldsymbol{U}_1}, \boldsymbol{U}_2), \ldots, \boldsymbol{U}_n))))^\intercal$$

At the end of this stage, each Service User (SU) can easily get the global public parameters PP_{ABE} of the attribute-based encryption system based on the published values.

$$PP_{ABE} := \{g_1, g_2, g_1^{\boldsymbol{A}}, g_1^{\boldsymbol{U}^\intercal \boldsymbol{A}}\}. \tag{7}$$

Authority setup

This step consists of 3 stages: *Commit and Reveal*, *Verify*, and *Generate*, and finally outputs another global public parameter PP_{VC} and public key PK for each attribute authority AA.

Initiate

The contract SC_{auth} deployed by trusted authority AA_{trust} has four main functions, Commit, Reveal, Prove and Generate, which also interacts with utility function SC_{util}. Its accessibility is also limited by deadlines $(ddl_1\prime, ddl_2\prime, ddl_3\prime)$ and the authorized list $AAlist$ set by AA_{trust}.

Commit and reveal

Every AA firstly samples a set of elements e': a matrix $\boldsymbol{X} \xleftarrow{\$} \mathbb{Z}_p^{(k+1)\times(k+1)}$, a vector $\boldsymbol{\tau} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$, two secret elements $\sigma, z \in \mathbb{Z}_p^*$, a scalar α_z and a scaled element $\alpha_z \cdot z$.

Therefore, the AA defines a set e' for vector commitment as:

$$e' := \{z, \alpha_z, \alpha_z \cdot z\}$$

and also designates:

$$SK := \{\boldsymbol{X}, \boldsymbol{\tau}, \sigma\}$$

as the secret key. Then, AA computes a set of *s-pair* for each element s in both e' and SK. We refer to the *s-pair* in $\mathbb{G}_1$ by $\boldsymbol{rp}_s$, and the *s-pair* in $\mathbb{G}_2$ by $\boldsymbol{rp}_s^2$ as Definition 2. These *s-pair* are defined as follows:

- For matrix $\boldsymbol{X}^\intercal$: $\boldsymbol{rp}_{\boldsymbol{X}} := (g_1^{\boldsymbol{A}}, g_1^{\boldsymbol{X}^\intercal \cdot \boldsymbol{A}})$
- For vector $\boldsymbol{\tau}^\intercal$: $\boldsymbol{rp}_{\boldsymbol{\tau}} := (g_1^{\boldsymbol{A}}, g_1^{\boldsymbol{\tau}^\intercal \boldsymbol{A}})$
- For element σ : $\boldsymbol{rp}_\sigma^2 := (g_2, g_2^\sigma)$
- For element z : $(\boldsymbol{rp}_z, \boldsymbol{rp}_z^2) = (g, g^z)$
- For scalar α_z: $(\boldsymbol{rp}_{\alpha_z}, \boldsymbol{rp}_{\alpha_z}^2) = (g, g^{\alpha_z})$
- For scaled element $\alpha_z z$: $(\boldsymbol{rp}_{\alpha_z z}, \boldsymbol{rp}_{\alpha_z z}^2) = (g, g^{\alpha_z \cdot z})$

After that, AA computes h' as follows:

$$\begin{gathered} h_{\boldsymbol{X}} \coloneqq \text{COMMIT}(\boldsymbol{rp}_{\boldsymbol{X}}) \quad h_{\boldsymbol{\tau}} \coloneqq \text{COMMIT}(\boldsymbol{rp}_{\boldsymbol{\tau}}) \quad h_{\sigma} \coloneqq \text{COMMIT}(\boldsymbol{rp}_{\sigma}^2) \\ h_z \coloneqq \text{COMMIT}(\boldsymbol{rp}_z || \boldsymbol{rp}_z^2) \quad h_{\alpha_z} \coloneqq \text{COMMIT}(\boldsymbol{rp}_{\alpha_z} || \boldsymbol{rp}_{\alpha_z}^2) \quad h_{\alpha_z \cdot z} \coloneqq \text{COMMIT}(\boldsymbol{rp}_{\alpha_z \cdot z} || \boldsymbol{rp}_{\alpha_z \cdot z}^2) \\ h' \coloneqq \text{COMMIT}(h_{\boldsymbol{X}} || h_{\boldsymbol{\tau}} || h_{\sigma} || h_z || h_{\alpha_z} || h_{\alpha_z \cdot z}) \end{gathered} \tag{8}$$

and broadcasts it to the contract SC_{auth} through blockchain transaction as the argument of the function Commit, which is exactly same as it of the contract SC_{sys}. It will store the value h' into state variable *h_collector* if the transaction is valid.

After h' recorded by contract SC_{auth}, AA needs to reveal each committed element by passing two lists of *s-pair*

$$L_{sk} = \{\boldsymbol{rp}_{\boldsymbol{X}}, \boldsymbol{rp}_{\boldsymbol{\tau}}, \boldsymbol{rp}_{\sigma}^2\} \tag{9}$$

$$L_{e'} = \{(\boldsymbol{rp}_s, \boldsymbol{rp}_s^2) | s \in e'\} \tag{10}$$

as arguments of the function Reveal of the contract SC_{auth}, which computes the hash result of L_{sk} and L_{vc} using function Hash of utility contract SC_{util}, and compares the result with the value stored in state variable *h_collector*. Finally, the valid set of *s-pair* will be stored into state variables *unverified_elements* and *unverified_sk* of the contract SC_{auth} respectively.

Verify

The system enters into the stage *Verify* after ddl_1' set by trusted authority AA_{trust}.

First of all, attribute authority AA generates the proof $\pi_s \coloneqq NIZK(\boldsymbol{rp}_s, h||h_s)$ for each s in both e' and SK, and broadcast these proofs $\{\pi\}$ as a list

$$L_{\pi}' = \{\pi_z, \pi_{\alpha_z}, \pi_{\alpha_z z}, \pi_{\boldsymbol{X}}, \pi_{\boldsymbol{\tau}}, \pi_{\boldsymbol{\sigma}}\} \tag{11}$$

through transaction before deadline ddl_2'. The function Prove of contract SC_{auth} takes input L_{π}' as the argument and checks the validity of these proofs by using utility functions SameRatio and CheckPoK of contract SC_{utils}. The Reveal-Prove algorithms in the contract SC_{auth} function similarly to those in contract SC_{sys}. Both require all authorities to commit their secret keys and later prove that they possess the knowledge of these keys.

We assign each AA an index $i \in [n-1]$, based on the order in which SC_{auth} receives the complete set of valid published proofs $\{\pi\}$. The trusted authority, denoted as AA_{trust}, is assigned the index n.

At the end of this stage, we have the verified elements $PK_i = (g_1^{\boldsymbol{X}_i^{\intercal}\boldsymbol{A}}, \hat{e}(g_1, g_2)^{\boldsymbol{\tau}_i^{\intercal}\boldsymbol{A}}, g_2^{\sigma})$ and $\{o_i \coloneqq g_1^{z_i}, g_1^{\alpha_{z_i}}, g_1^{\alpha_{z_i} z_i}\}$ for each AA_i.

Generate

In the last stage *Generate*, AA_i needs to generate a set of group elements (O, θ_O, O'), selects a reasonable number of supported attributes l_i, and broadcasts them to contract SC_{auth} before the deadline ddl_3'.

The elements (O, θ_O, O') provided by AA_i are constructed as follows:

$$O_i \coloneqq o_{i,j} = \{(o_j)^{z_i}\}_{i \neq j, j \in [n]}$$

$$\theta_{O_i} := \{(g_1^{\alpha_{z_j}})^{\alpha_{z_i}}\}_{i\neq j, j\in[n]}$$

$$O_i' := \{(g_1^{\alpha_{z_j} z_j})^{\alpha_{z_i} z_i}\}_{i\neq j, j\in[n]}$$

Algorithm 7 Contract SC_{auth}

```
function GENERATE(O, θ, O', l)
    if msg.sender ∉ AAlist OR block.timestamp > ddl then        ▷ Requirement check
        throw
    end if
    rp_z, rp_{α_z}, rp_{α_z z} ← verified_elements[msg.sender]
    i ← index[msg.sender]
    for j ← 0, n − 1 do
        if i == j then
            continue
        end if
        check1 ← SAMERATIO((rp_z[1], O[j]), rp_z)
        check2 ← SAMERATIO((rp_α[1], θ[j]), rp_{α_z})
        check3 ← SAMERATIO((rp_{α_z z}[1], O'[j]), rp_{α_z z})
        if check1 == check2 == check3 == true then
            verified_O[msg.sender] ← O
        else
            throw
        end if
    end for
    attribute_size[msg.sender] ← l
end function
```

Invoked by this transaction, function GENERATE (Algorithm 7) of SC_{auth} firstly checks the validity of elements $(O_i, \theta_{O_i}, O_i')$ and the value of l_i, and then records AA_i's blockchain address *addr* with the claimed attribute size l_i. The function GENERATE and some state variables used are defined as follows:

1. *verified_O* (State Variable): A mapping collection from the blockchain address belonged to one attribute authority to its set of elements $(\{o_j^{z_i}\}_{i\neq j, i, j\in[n]}$ in the public parameter of vector commitment PP_{VC}
2. *attribute_size* (State Variable): A mapping collection from the blockchain address belonged to one attribute authority to its number of supported attribute

For example, if AA_i owns the set of attributes $\{entry, mid, senior, agent, manager\}$, the value of attribute size l_i is 5.

After the contract SC_{auth} receives all the pairs $\{o_i, o_{i,j}, l_i\}$ from $AA_i \in \{AA_1, AA_2, \ldots, AA_n\}$, every Service User (SU) can get the global parameter for the vector commitment system

$$PP_{VC} := \{g_1, g_2, \{o_i\}_{i\in[n]}, \{o_{i,j}\}_{i,j\in[n], i\neq j}\} \tag{12}$$

Table 3 A example of address-*AA*-attribute vector mapping table.

Address	$addr_1$		$addr_2$	.....	$addr_{n-1}$		$addr_n$
Attribute authority	AA_1		AA_2	.....	AA_{n-1}		AA_{trust}
Attribute representation	v_1	v_2	v_3	.....	v_{l-1}	v_l	v_{l+1}

and generate a mapping table that maps each blockchain address of AA_i to its corresponding information as shown in Table 3. We use l to represent the size of supported attributes set $\mathcal{X}$ and v to represent the owned attributes for each AA.

Data user registration

To get the global identifier *GID*, which will be used in the process of *Key Generation*, the data user (DU) needs to register the owned address $addr_{DU}$ by calling the function USER_REGISTER of the contract SC_{REG}.

DU needs to send predefined amount of *GWEI* to the contract SC_{REG} as the registration fee payment. This amount defaults to 1,000,000 *GWEI*, which is approximately equivalent to 1.63 USD as of September 2023. In return, DU receives a value *GID*, which is the hash value of DU's address $addr_{DU}$, also known as the $msg.sender$ of this transaction call.

Following that, DU establishes a secure channel or employs some off-chain methods with AA_i that have the required attributes and can verify DU's identity. We assume that DU is an agent for one insurance company, and the company itself runs the consensus node of AA_i in this system. Therefore, DU may easily get verified by showing an ID badge to the person who manages the AA_i. DU then requests that AA_i issues a set of attributes $\mathcal{R}_{i,GID}$ in regards to DU's identity. The format of a set of attributes might look like this: $(entry, N/A, N/A, agent, N/A)$ out of the full set of attributes $(entry, mid, senior, agent, manager)$.

For those $AA_j, j \neq i$ that do not contain the needed attributes or cannot verify DU's identity, DU may just set $\mathcal{R}_{j,GID}$ to be $(N/A, N/A, N/A)$ with $l_j = 3$.

Finally, DU receives $\mathcal{R}_{i,GID}$ from AA_i, sets $\mathcal{R}_{j,GID}$ for $AA_j, j \neq i$, and combines them as the set of attributes $\mathcal{R}_{GID}$ which will be used in the process of Key Generation.

Key generation

Without loss of generality, we suppose that there are a total set of attributes $\mathcal{X}$, indexed from 1 to l and a total set of attributes authorities $\mathcal{U}$ including AA_{trust}, indexed from 1 to n. Assume that the attribute authority AA_i has a subset of attributes $\mathcal{S}_i$, then we have $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$ for $i \neq j$ and $i, j \in |n|$ and $\mathcal{S}_1 \cup \mathcal{S}_2 ... \cup \mathcal{S}_n = \mathcal{X}$.

To get the secret key $sk_{i,GID,\boldsymbol{C}}$ which is comprised of multiple key parts $\{K_{j,GID,\boldsymbol{C}}\}_{j \in \mathcal{S}_i}$ from AA_i, Data User (DU) must initially generate the attribute vector $\boldsymbol{v}_{GID}$. This is based on $\mathcal{R}_{GID}$ that was acquired during the **Data User Registration** process.

Given that the DU's set of attributes $\mathcal{R}_{i,GID} \in \mathcal{X}$ and the mapping Table 3 generated from the process **Authority Setup**, the attribute vector $\boldsymbol{v}_{GID}$ is set as follows:

1. Set the first l entries such that $v_k = \begin{cases} 1 & i \in R_{GID} \\ 0 & i \notin R_{GID} \end{cases}$
2. Set the $l+1$ entry to be 1. (AA_{trust} is responsible for this entry)

Table 4 A example of m for vector commitment.

Attribute authority	AA_1			AA_2	
Attributes	**entry**	**mid**	**senior**	**agent**	**manager**
Vector element	v_1	v_2	v_3	v_4	v_5
Element value	1	0	0	1	0
nonce	r_1			r_2	
m_i	COMMIT$(v_1 \| \| v_2 \| \| v_3 \| \| r_1)$			COMMIT$(v_4 \| \| v_5 \| \| r_2)$	

Then, DU randomly chooses $r_1, r_2, /dots, r_n \xleftarrow{\$} \mathbb{Z}_p^*$ for each AA and combines the arguments r_i with the attribute vector $\boldsymbol{v}_{i,GID}$ (represented as a bit string) to produce a committed value $m_i \coloneqq \text{COMMIT}(v_{i,1} \| \| v_{i,2} \ldots v_{i,j} \| \| r_i)$, where $i \in [n], j \in [|\mathcal{S}_i|]$.

In Table 4, for instance, we have two attribute authorities AA_1 and AA_2 that possess attributes (entry, mid, senior) and (agent, manager) respectively. If there is a data user DU with a set of attributes $\mathcal{R}$ =(entry, agent), DU's attribute vector is $\boldsymbol{v} = (1,0,0,1,0)$. For AA_1 and AA_2, DU samples two random values $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p^*$ and then computes auxiliary information $\boldsymbol{aux} = (m_1, m_2)$ using COMMIT. In general, based on the public parameter for vector commitment $PP_{VC} = \{g_1, g_2, \{o_i\}_{i \in [n]}, \{o_{i,j}\}_{i,j \in [n], i \neq j}\}$, generated in **Authority Setup**, DU can calculate the commitment $\boldsymbol{C}$ on the attribute vector $\boldsymbol{v}$ from its $\mathcal{R}$:

$$\boldsymbol{C} \coloneqq o_1^{m_1} o_2^{m_2} \ldots o_n^{m_n}$$

To request the secret key part $K_{j,GID,\boldsymbol{C}}, j \in \mathcal{S}_i$, DU establishes another secure channel with AA_i and sends commitment C along with an opening op_i and nonce r_i. Such opening op_i is calculated as follows:

$$op_i = \prod_{j=1, j \neq i}^{n} o_{i,j}^{m_j} = (\prod_{j=1, j \neq i}^{n} o_j^{m_j})^{z_i}$$

Based on the DU's GID, AA_i firstly retrieves the information of the set of attributes $\mathcal{R}_{i,GID}$ which have been issued in the previous process **Data User Registration** and then verifies commitment $\boldsymbol{C}$ using the opening op_i and nonce r_i received

$$\hat{e}(\boldsymbol{C}/o_i^{m_i'}, g_2^{z_i}) \stackrel{?}{=} \hat{e}(op_i, g_2)$$

where m_i' is the value calculated by the $\mathcal{R}_{i,GID}$ issued to DU. If the above check passes, AA_i uses a pre-defined random oracle $\mathcal{H} : \mathbb{G}_2 \times \{0,1\}^\lambda \times \mathbb{Z}_p^{k+1} \to \mathbb{Z}_p^{k+1}$ to generate masking value $\boldsymbol{\mu_i} \in \mathbb{Z}_p^*$

$$\boldsymbol{\mu_i} = \sum_{j=1}^{i-1} \mathcal{H}(y_j^{\sigma_i}, GID, \boldsymbol{C}) - \sum_{j=i+1}^{n} \mathcal{H}(y_j^{\sigma_i}, GID, \boldsymbol{C})$$

and hash functions $H_1(GID, \boldsymbol{C}), \ldots, H_{k+1}(GID, \boldsymbol{C})$ to generate $g_2^{\boldsymbol{h}}$ where $\boldsymbol{h} \in \mathbb{Z}_p^{k+1}$

$$\boldsymbol{h} : H(GID, \boldsymbol{C}) = (H_1(GID, \boldsymbol{C}), \ldots, H_{k+1}(GID, \boldsymbol{C}))^\intercal$$

Finally, AA_i computes the secret keys

$$sk_{i,GID,\boldsymbol{C}} \coloneqq \{K_{j,GID,\boldsymbol{C}}\}_{j \in \mathcal{S}_i}, \tag{13}$$

which consists of key parts

$$K_{j,GID,\boldsymbol{C}} \coloneqq g_2^{\boldsymbol{\tau}_i} - v_j \boldsymbol{X}_i \boldsymbol{h} + \boldsymbol{\mu}_i \tag{14}$$

for each possessed attributes by AA_i and send $sk_{i,GID,\boldsymbol{C}}$ back to the DU through the secure channel.

To get the special secret key part $sk_{n,GID,\boldsymbol{C}}$ for the $l+1$ entry, DU also needs to communicate with trusted authority AA_{trust} and provides the commitment $\boldsymbol{C}$ with the opening op_n and nonce r_n through the secure channel. As shown in the Table 3, AA_{trust} sets the m'_n to be COMMIT$((v_{l+1}||r_n))$ where $v_{l+1}=1$ and then checks the following equation

$$\hat{e}(\boldsymbol{C}/o_n^{m'_n}, g_2^{z_n}) \stackrel{?}{=} \hat{e}(op_n, g_2)$$

If it passes, AA_{trust} computes the secret key part similar as $sk_{i,GID,\boldsymbol{C}}$

$$sk_{n,GID,\boldsymbol{C}} \coloneqq K_{l+1} = g_2^{\boldsymbol{\tau}_n} - v_n \boldsymbol{X}_n \boldsymbol{h} + \boldsymbol{\mu}_n \tag{15}$$

and sends it back to DU.

Upon receiving all the responses from each $AA_i, i \in [n]$, DU will finally gets a complete set of secret keys $\{sk_{i,GID,\boldsymbol{C}}\}_{i\in[n]}$.

Encryption and upload

Given that ABE is significantly more expensive than symmetric key encryption (*Wang et al., 2014*), the files that the data owner (DO) wants to share are not directly encrypted with ABE. Instead, hybrid encryption of ABE and AES is used for efficiency.

Firstly, DO randomly samples an AES key AK from the key space and encrypts the file F as the ciphertext CT_F.

Then, DO uploads the ciphertext CT_F to IPFS and records the file location loc returned by IPFS. The metadata message M can then be constructed as:

$$M \coloneqq (K, loc)$$

Using the policy vector $\boldsymbol{x}$ discussed above, DO samples a random vector $\boldsymbol{s} \in \mathbb{Z}_p^k$, generates a policy vector $\boldsymbol{x} \coloneqq (x_1, x_2, \ldots, x_n) \in \mathbb{Z}_p^n$ acting as the ciphertext policy, and outputs the ciphertext CT_M consisting of

$$\begin{aligned} ct_0 &= g_1^{\boldsymbol{A}\boldsymbol{s}} \\ ct_i &= g_1^{(x_i \boldsymbol{U}^\intercal + \boldsymbol{X}_i^\intercal)\boldsymbol{A}\boldsymbol{s}} \\ ct' &= M \cdot \hat{e}(g_1, g_2)^{\boldsymbol{\tau}^\intercal \boldsymbol{A}\boldsymbol{s}}, \end{aligned} \tag{16}$$

where $\boldsymbol{\tau} = \sum_{i=1}^n \boldsymbol{\tau}_i$.

Lastly, the ciphertext CT_M is sent to the contract SC_{log} with the optional keyword kw that may ease the data retrieval process. The contract SC_{log} will emit this new uploading information to the subscribers.

Download and decryption

All the encrypted metadata CT_M will be recorded sequentially. If the DU is interested in one of DO's files, DU may subscribe to the event created by the contract SC_{log} to obtain the encrypted metadata CT_M.

To decrypt the ciphertext CT_M, DU computes

$$\hat{e}(ct_0, \prod_{j=1}^{l+1} K_j) \cdot \hat{e}(\prod ct_i^{v_i}, \boldsymbol{h}) = \hat{e}(g_1, g_2)^{\boldsymbol{\tau}^\top As}$$

and tries to recover the message

$$ct'/\hat{e}(g_1, g_2)^{\boldsymbol{\tau}^\top As} = M' \tag{17}$$

If DU's attribute vector $\boldsymbol{v}$ satisfies the policy vector $\boldsymbol{x}$ selected by DO, DU can retrieve the AES key AK and file location loc from the metadata M. Finally, DU downloads the encrypted file CT_F from the IPFS based on the location loc and uses AK to decrypt CT_F to recover the original file F.

Correctness. Since $CT_M = (ct_0, \{ct_i\}_{i=1}^n, ct')$, $\{sk_{i,GID,\boldsymbol{C}} = \{K_j\}_{j \in \mathcal{R}_i}\}$ and $n = l+1$, we can compute

$$\begin{aligned}
&\hat{e}(ct_0, \prod_{j=1}^{l+1} K_j) \cdot \hat{e}(\prod_{i=1}^{n} ct_i^{v_i}, \boldsymbol{H}(GID, \boldsymbol{C})) \\
&= \hat{e}(g_1^{\boldsymbol{As}}, g_2^{\sum_{i=1}^n \boldsymbol{\tau}_i - v_i \boldsymbol{X}_i h + \mu_i}) \\
&\cdot \hat{e}(g_1^{\sum_{i=1}^n v_i (x_i \boldsymbol{U}^\top + \boldsymbol{X}_i^\top) \boldsymbol{As}}, g_2^{\boldsymbol{h}}) \\
&= \hat{e}(g_1, g_2)^{\boldsymbol{\tau}^\top \boldsymbol{As} - \sum_{i=1}^n v_i \boldsymbol{h}^\top \boldsymbol{X}_i^\top \boldsymbol{As}} \\
&\cdot \hat{e}(g_1, g_2)^{<\boldsymbol{x}, \boldsymbol{v}> \boldsymbol{h}^\top U^\top As + \sum_{i=1}^n v_i \boldsymbol{h}^\top X_i^\top As} \\
&= \hat{e}(g_1, g_2)^{\boldsymbol{\tau}^\top \boldsymbol{As}} \cdot \hat{e}(g_1, g_2)^{<\boldsymbol{x}, \boldsymbol{v}> \boldsymbol{h}^\top U^\top As}
\end{aligned} \tag{18}$$

If $<\boldsymbol{x}, \boldsymbol{v}> = 0$, we obtain $\hat{e}(g_1, g_2)^{\boldsymbol{\tau}^\top As}$ and can recover the message.

SECURITY ANALYSIS

The security of the original scheme (Definition 5), while explored within the framework by *Michalevsky & Joye (2018)* and proven to be secure against selective adversaries in random oracle model (ROM), does not adequately capture certain real-world circumstances. Recognizing this limitation, we have devised a more flexible approach to model the security challenges more accurately.

This new model, defined in Definition 6, is derived from the original scheme but adapted to bridge the gap between theoretical models and the complexities of real-world applications. In *Datta, Komargodski & Waters (2023)*, this notion has been defined as fully adaptive security, where the adversary can decide the set of authorities to corrupt at any time.

Definition 6 The scheme is secure against static corruption of authorities if the advantage in winning the following game is negligible.

1. **Setup phase:** $\mathcal{S}$ picks a random bit $b \in \{0, 1\}$ and outputs the public parameters PP.
2. **Query phase:**
    1 $\mathcal{A}$ can adaptively decide the identity of each attribute authority, determining whether it is normal or corrupted, as well as the order in which each authority's keys are generated or received by $\mathcal{S}$.

- If the authority is normal, $\mathcal{A}$ requests the keys $\{PK\}$ from $\mathcal{S}$, which are generated through the **Auth Setup** process.
- If the authority is corrupted, $\mathcal{A}$ generates the keys $\{PK, SK\}$ itself and provides $\{PK\}$ to $\mathcal{S}$.

2 **Finally**, $\mathcal{A}$ marks each corrupted authority and outputs the set of corrupt authorities A^* based on the decisions made in the previous step.

3. **Challenge phase:**
   1 $\mathcal{A}$ outputs two policies $\boldsymbol{x}_0, \boldsymbol{x}_1$ and two equal-length messages M_0, M_1.
   2 $\mathcal{S}$ outputs a challenge ciphertext $CT \xleftarrow{\$}$ **Encrypt**$(\boldsymbol{x}_b, M_b)$, where $b = 0$ or 1
4. **Query phase 2:** $\mathcal{A}$ request secret attribute keys $\{sk\}$ for vectors $\boldsymbol{v}$ from $\mathcal{S}$, which are not orthogonal to $\boldsymbol{x}_0$ or $\boldsymbol{x}_1$.
5. **Guess:** $\mathcal{A}$ outputs a guess b' and wins the game if $b' = b$.

Contrasting with *Michalevsky & Joye (2018)* in the **Query Phase**, which requires the adversary $\mathcal{A}$ fix a set of corrupt authorities and provide their public parameters PK **prior** to $\mathcal{S}$ executes **AuthSetup** for the non-corrupt authorities as follows:

Query phase:

1. $\mathcal{A}$ **first** outputs the set of corrupt authorities A^*, generates authority keys $\{PK, SK\}_i$ for every $AA_i \in A^*$, and provides $\{PK\}_i$ to $\mathcal{S}$.
2. $\mathcal{A}$ **then** requests the remaining public keys $\{PK\}_j$ generated from $\mathcal{S}$ for each non-corrupt authority $AA_j \notin A^*$.

This adaptation introduces an aspect of real-world unpredictability and tests the system's robustness on a broader array of scenarios.

Rogue-key attack analysis

First, we provide proof that the original scheme, as proposed by *Michalevsky & Joye (2018)* and outlined in Definition 5, is vulnerable to a rogue-key attack in the game (Definition 6). Following this, we discuss how our blockchain-based data governance mechanism effectively mitigates this vulnerability.

Attack

We demonstrate that a corrupt authority can learn the key parts K_{l+1} corresponding to $v_{l+1} \in \boldsymbol{v}$ issued by trusted authority AA_{trust} and thus decrypt the ciphertext in Michalevsky and Joys's scheme (*Michalevsky & Joye, 2018*), without having the required secret key sk satisfying the attached access policy. This is a typical attack, called a rogue-key attack, in which the adversary uses a public key, a function of honest users' keys (*Ristenpart & Yilek, 2007*).

Proof First, an adversarial authority AA_{ad} makes one of the corrupted authorities hold until all the other attribute authorities $AA_i, i \neq ad, i \in n$ publish their public keys, in accordance with the game (Definition 6).

$$PK_i = (g_1^{\boldsymbol{X}_i^\top \boldsymbol{A}}, \hat{e}(g_1, g_2)^{\boldsymbol{\tau}_i^\top \boldsymbol{A}}, y := g_2^{\sigma})$$

and then calculates the public key as follows:

$$g_1^{\boldsymbol{X}_{ad}^{\top}\boldsymbol{A}} := -\sum_{j=0, j\neq ad}^{n} g_1^{\boldsymbol{X}_j^{\top}\boldsymbol{A}}$$

$$\hat{e}(g_1, g_2)^{\boldsymbol{\tau}_{ad}^{\top}\boldsymbol{A}}, y_{ad} := g_2^{\sigma_i}$$

After receiving the challenge ciphtertext $CT := \{ct_0, ct_i, ct'\}$ from $\mathcal{S}$ in the **Challenge Phase**, an adversarial data user DU_{ad} creates an attribute vector $\boldsymbol{\nu}' = (0, 0, \ldots, 0, 1)$ and requrests key parts K_i from all the attribute authorities AA_i. As in the encryption phase, data owner DO will collect all the public keys $PK_i, i \in [n+1]$ from each AA_i and outputs the cipher text CT consisting of

$$ct_0 = g_1^{\boldsymbol{As}}$$

$$ct_i = g_1^{(x_i\boldsymbol{U}^{\top} + \boldsymbol{X}_i^{\top})\boldsymbol{As}}$$

$$ct' = m \cdot \hat{e}(g_1, g_2)^{\boldsymbol{\tau}^{\top}\boldsymbol{As}},$$

where $\boldsymbol{\tau} = \sum_{i=1}^{n} \boldsymbol{\tau}_i$. Third, given the ciphertest $(ct_0, \{ct_i\}, ct')$ and received secret keys $\{K_i\}$, adversary DU_{ad} decrypts it as following:

$$\begin{aligned}
&\hat{e}\ \ (ct_0, \prod_{j=1}^{l+1} K_i) \cdot \hat{e}(\prod_{i=1}^{n} ct_i^{\nu_i}, \boldsymbol{H}(GID, \boldsymbol{C})) \\
&\quad = \hat{e}(g_1^{\boldsymbol{As}}, g_2^{\sum_{i=1}^{n} \boldsymbol{\tau}_i - \nu_i \boldsymbol{X}_i \boldsymbol{h} + \boldsymbol{\mu}_i}) \\
&\quad \cdot \hat{e}(g_1^{\sum_{i=1}^{n} \nu_i (x_i \boldsymbol{U}^{\top} + \boldsymbol{X}_i^{\top})\boldsymbol{As}}, g_2^{\boldsymbol{h}}) \\
&\quad = \hat{e}(g_1, g_2)^{\boldsymbol{\tau}^{\top}\boldsymbol{As} - \sum_{i=1}^{n} \nu_i \boldsymbol{h}^{\top} \boldsymbol{X}_i^{\top} \boldsymbol{As}} \\
&\cdot \hat{e}(g_1, g_2)^{<\boldsymbol{x}, \boldsymbol{\nu}> \boldsymbol{h}^{\top} \boldsymbol{U}^{\top} \boldsymbol{As} + \sum_{i=1}^{n} \nu_i \boldsymbol{h}^{\top} \boldsymbol{X}_i^{\top} \boldsymbol{As}} \\
&= \hat{e}(g_1, g_2)^{\boldsymbol{\tau}^{\top}\boldsymbol{As}} \cdot \hat{e}(g_1, g_2)^{<\boldsymbol{x}, \boldsymbol{\nu}> \boldsymbol{h}^{\top} \boldsymbol{U}^{\top} \boldsymbol{As}}
\end{aligned}$$

Since DU_{ad}'s attribute vector is $\boldsymbol{\nu}' = (0, 0, \ldots, 0, 1)$, we have

$$\hat{e}(g_1, g_2)^{\boldsymbol{\tau}^{\top}\boldsymbol{As}} \cdot \hat{e}(g_1, g_2^{\boldsymbol{h}})^{x_{l+1}\boldsymbol{U}^{\top}\boldsymbol{As}} \tag{19}$$

In order to decrypt the ciphertext, we need to cancel out the last element in Eq. (19). Therefore, adversary $\mathcal{A}$ could use the published $\{ct_i\}$ to calculate attacking component ω.

$$\omega := \prod g_1^{(x_i\boldsymbol{U}^{\top} + \boldsymbol{X}_i^{\top})\boldsymbol{As}}$$
$$= g_1^{\sum x_i \boldsymbol{U}^{\top}\boldsymbol{As}} \cdot g_1^{\sum (\boldsymbol{X}_i^{\top}\boldsymbol{A})\boldsymbol{s}}.$$

Based on $g_1^{\boldsymbol{X}_{ad}^{\top}\boldsymbol{A}} := -\sum_{j=0, j\neq ad}^{n} g_1^{\boldsymbol{X}_j^{\top}\boldsymbol{A}}$, we obtain $\omega = g_1^{\sum_{i=1}^{n} x_i \boldsymbol{X}^{\top}\boldsymbol{As}}$ and can further cancel the last component in Eq. (19) as $x_{l+1} = -\sum_{i=1}^{n} x_i$. In the end, the adversary recovers the message by computing

$$ct' / \hat{e}(g_1, g_2)^{\boldsymbol{\tau}^{\top}\boldsymbol{As}} = m.$$

□

Based on the provided chosen message (M_0, M_1) in the **Challenge Phase**, $\mathcal{A}$ can easily output a correct guess b' that $b' = b$ with high probability, which violates the game defined by Definition 6.

Proof of security of our approach

In the absence of a certificate authority (CA), one way to mitigate this requires a Non-interactive Zero-knowledge (NIZK) proof in a decentralized manner. Since every *AA* needs to generate a commitment of secrets used for public key *PK* and then creates its proof of knowledge using NIZK (Algorithm 2), which is built upon Schnorr Identification Protocol (*Schnorr, 1990*).

Proof Each attribute authority (*AA*) generate commitments to their secrets $\{\boldsymbol{X}, \boldsymbol{\tau}, \sigma\}$ and creates a NIZK of knowledge π, as described in **Authority Setup**. We thus just need to prove the NIZK satisfies the Knowledge of Coefficient assumption (Assumption 5) of the work *Bowe, Gabizon & Miers (2017)*. Suppose an adversary $\mathcal{A}$ successfully crafts a proof $\pi_{ad,s} = (R_{ad,s}, u_{ad,s})$ for a secret $s = \sum_{i=1, i\neq ad}^{n-1} \boldsymbol{X}_i^{\intercal} \boldsymbol{A}$ for a secret s without knowing s, which implies that given *s-pair* $\boldsymbol{rp}_s = (A, B)$ and string h, the probability of finding a valid u such that $u \cdot A = R + c \cdot B$ would be **non-negligible**. Consiering another *s-pair* $\boldsymbol{rp}_\sigma^2 = \{g_2, g_2^\sigma\}$, $\mathcal{A}$ computes $R' \leftarrow g_1^{\alpha'}$ and $c' \leftarrow \textbf{Hash}(R' || h)$ as in Definition 2 based on the randomly chosen value α'. Assigning $r \coloneqq g_2$ and $y \coloneqq (g_2^\sigma)^{c'}$ in the **SameRatio**$((g_1, x), (r, y))$, which implies that $x = g_1^{\sigma \cdot c'}$, if $\mathcal{A}$ can find u' such that $u' = \alpha' + c' \cdot s$, this would satisfy the **SameRatio** $((g_1, x), (r, y))$ condition without knowing $\sigma \cdot c'$, violating the Assumption 5. Thus, $\mathcal{A}$ cannot win the game, and our scheme is secure against rogue-key attacks. □

Inferring the secret vector in ciphertext

With its further study of the Decentralized Policy-hiding ABE scheme (*Michalevsky & Joye, 2018*), we also identified a potential risk associated with generating public parameters during **Setup**. The details of the risk and proof of security are described in the following sections.

Vulnerability

As defined in Definition 5, a trusted third-party (TTP) or an attribute authority (*AA*) needs to pick a set of random numbers $a_1, a_2, \ldots, a_k \xleftarrow{\$} \mathbb{Z}_p^*$, and then generate a random matrix $\boldsymbol{A} = \begin{pmatrix} a_1 & 0 & & 0 \\ 0 & a_2 & & 0 \\ \vdots & & \ddots & \\ 0 & 0 & \cdots & a_k \\ 1 & 1 & \cdots & 1 \end{pmatrix} \in \mathbb{Z}_p^{(k+1)\times k}$ with $\boldsymbol{a}^\perp = \begin{pmatrix} a_1^{-1} \\ a_2^{-1} \\ \vdots \\ a_k^{-1} \\ -1 \end{pmatrix} \in \mathbb{Z}_p^{(k+1)}$, which $\boldsymbol{A}^{\intercal} \boldsymbol{a}^\perp = 0$ during **Setup**.

Proof If the above generation is conducted by a PPT adversary $\mathcal{A}$, or if the sensitive information $\boldsymbol{a}^\perp$ is exposed, $\mathcal{A}$ can infer the value of g_1^s from any published ciphertext CT. As we know, to generate a ciphertext CT, data owner (*DO*) firstly samples a random vector $\boldsymbol{s} = (s_1, s_2, \ldots, s_k) \in \mathbb{Z}_p^k$, then creates a policy vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_{n-1}, x_n) \in \mathbb{Z}_p^n$ acting as the ciphertext policy, and finally outputs the ciphertext CT consisting of

$$ct_0 = g_1^{\boldsymbol{As}}$$

$$ct_i = g_1^{(x_i \boldsymbol{U}^{\intercal} + \boldsymbol{X}_i^{\intercal})\boldsymbol{As}}$$

$$ct' = M \cdot \hat{e}(g_1, g_2)^{\boldsymbol{\tau}^{\intercal} \boldsymbol{As}}$$

where $\boldsymbol{\tau} = \sum_{i=1}^{n} \boldsymbol{\tau}_i$ and $\hat{e}(g_1, g_2)^{\boldsymbol{\tau}_i^{\intercal}}$ collected from published public keys PK_i.Since

$$\boldsymbol{As} = \begin{pmatrix} a_1 s_1 \\ a_2 s_2 \\ \vdots \\ a_k s_k \\ s_1 + s_2 + \cdots + s_k \end{pmatrix}$$

$\mathcal{A}$ can use the result $ct_0 = g_1^{\boldsymbol{As}}$ and known $\boldsymbol{a}$ to calculate:

$(g_1^{\boldsymbol{As}})^{\boldsymbol{a}} = g_1^{\boldsymbol{Asa}}$

For the exponent $\boldsymbol{Asa}$, we have

$$\begin{aligned} \boldsymbol{Asa} &= \begin{pmatrix} a_1 s_1 \\ a_2 s_2 \\ \vdots \\ a_k s_k \\ s_1 + s_2 + \cdots + s_k \end{pmatrix} \cdot (a_1^{-1}, a_2^{-1}, \ldots, a_k^{-1}, -1) \\ &= \begin{pmatrix} s_1 & a_1 a_2^{-1} s_1 & \ldots & a_1 a_k^{-1} s_1 & -a_1 s_1 \\ a_1^{-1} a_2 s_2 & s_2 & \ldots & a_2 a_k^{-1} s_2 & -a_2 s_2 \\ \vdots & \ddots & & & \\ a_1^{-1} a_k s_k & a_2^{-1} a_k s_k & \ldots & s_k & -a_k s_k \\ a_1^{-1} \sum s & a_2^{-1} \sum s & \cdots & a_k^{-1} \sum s & -\sum s \end{pmatrix} \end{aligned}$$

The ith elements from exponents $\boldsymbol{As}$ and $\boldsymbol{a}$ partially cancel each other out, and the remaining element s_i is the targeting exponent. Therefore, adversary $\mathcal{A}$ might extrapolate g_1^{s} from the published ciphertext $ct_0 = g_1^{\boldsymbol{Asa}}$ with the knowledge of $\boldsymbol{a}^{\perp}$. □

Proof of security with our approach

One potential mitigation of the inferring attack is to generate a composite public parameter PP by multiple AA, such that neither of those individual AAs knows the composite

$$g_1^{\boldsymbol{A}} = \textsc{powerMulti}(\textsc{powerMulti}(\ldots(\\ \textsc{powerMulti}(\textsc{powerMulti}(g_1^{\boldsymbol{A}_1}, \boldsymbol{A}_2), \boldsymbol{A}_3) \ldots, \boldsymbol{A}_n),$$

and no participant learns the secrets of others unless one colludes with every other participant under Assumption 5.

Another potential security vulnerability arises from inconsistencies in the published sets $(V_i, \theta_{V_i}, V_i')$ or $(W_i, \theta_{W_i}, W_i')$ that, *i.e.,* an adversary AA with index i can strategically choose different $(\boldsymbol{A}_i, \boldsymbol{A}_i')$ values to compute

$V_i \coloneqq \textsc{powerMulti}(V_{i-1}, \boldsymbol{A}_i')$

$V_i' \coloneqq \textsc{powerMulti}(V_{i-1}', \alpha_{\boldsymbol{A}_i} \boldsymbol{A}_i),$

results in the final composite becoming invalid for further operations. However, given that each AA has committed and disclosed a set of group elements

$$e = \{\boldsymbol{A}, \boldsymbol{U}, \alpha_{\boldsymbol{A}}, \alpha_{\boldsymbol{U}}, \alpha_{\boldsymbol{A}} \cdot \boldsymbol{A}, \alpha_{\boldsymbol{U}} \cdot \boldsymbol{U}\}$$

in both $\mathbb{G}_1$ and $\mathbb{G}_2$, as described in "Commit and Reveal", it becomes possible to ascertain that each AA's $(V_i, \theta_{V_i}, V_i')$ is a proper multiple of previous published components:

$$\hat{e}(V_i, g_2) = \hat{e}(V_{i-1}, g_2^{\boldsymbol{A}_i})$$

$$\hat{e}(\theta_{V_i}, g_2) = \hat{e}(\theta_{V_{i-1}}, g_2^{\alpha_i})$$

$$\hat{e}(V_i', g_2) = \hat{e}(V_{i-1}', g_2^{\alpha_{\boldsymbol{A}_i}\boldsymbol{A}_i})$$

Even if all the other participants have collaborated, this is still a robust and sufficient setup scheme, even if only one participant is honest and does not reveal the secrets. Hence, the greater the number of unrelated participants in a trusted setup, the less likely the possibility of invalid and unusable public parameter PP_{ABE} (*Wilcox, 2016*).

Receiver privacy

According to the Definition 5, the receiver must provide its attribute vector $\boldsymbol{v}$ to each attribute authority AA from which a key is requested. In consequence, AA learns not only if the user possesses the attribute that AA owns but also all of the user's other attributes. This appears to violate the privacy of the user in a decentralized setting.

Therefore, Michalevsky and Joye propose an enhancement that provides additional privacy protection for the attribute vector $\boldsymbol{v}$ in their work *Michalevsky & Joye (2018)*, which uses it to hide the set of receiver attributes $\boldsymbol{v}$ from authorities. This technique is called *position-binding* Vector Commitments, introduced by *Catalano & Fiore (2013)*. Our access control system is based on the work *Catalano & Fiore (2013)*, but it has been slightly modified to work with asymmetric pairings $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

Since hiding is not a critical property and can be easily achieved in the realization of vector commitments (*Catalano & Fiore, 2013*), only the proof of position binding is provided below:

Proof Suppose an efficient adversary $\mathcal{A}$ can produce two valid openings op to different messages (m, m') at the same position i. In that case, we can build an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break the Square Computational Diffie-Hellman assumption. For a sequence of messages $m_1, m_2, \ldots, m_n$ and public parameter $pp = (g_1, g_2, \{o_i\}_{i\in[n]}, \{o_{i,j}\}_{i,j\in[n],i\neq j})$, the vector commitment is $\boldsymbol{C} = o_1^{m_1} o_2^{m_2} \cdot o_n^{m_n}$ and the opening for position i is $op_i = (\prod_{j=1,j\neq i}^{n} o_j^{m_j})^{z_i}$. The efficient algorithm $\mathcal{B}$ takes as input a tuple (g_1, g_1^r, g_2^r) and aims to compute $g_1^{r^2}$ to break Assumption 4. First, $\mathcal{B}$ selects a random position $i \xleftarrow{\$} [n]$ on which adversary $\mathcal{A}$ will break the position binding. Next, $\mathcal{B}$ chooses $z_j \xleftarrow{\$} \mathbb{Z}_p^*$ where $\forall j \in [n], j \neq i$ and then computes:

$$\forall j \in [n] \setminus i : o_j = g_1^{z_j}, o_{i,j} = (g_1^r)^{z_j}, o_i = g_1^r$$

$\forall k, j \in [n] \setminus i, k \neq j : o_{k,j} = g_1^{z_k z_j}$

Second, $\mathcal{B}$ sets $pp = (g_1, g_2, \{o_i\}_{i\in[n]}, \{o_{i,j}\}_{i,j\in[n],i\neq j})$ and runs $\mathcal{A}(pp)$, that outputs a tuple $(\boldsymbol{C}, m, m', j, op_j, op_j')$ such that $m \neq m'$ and both op_j and op_j' are valid.Finally, $\mathcal{B}$ computes

$g_1^{r^2} = (op_i'/op_i)^{(m_i - m_i')^{-1}}$

Since openings verify the two messages (m_i, m_i') correctly at position i, then it holds:

$\hat{e}(\boldsymbol{C}, g_2^r) = \hat{e}(o_i^{m_i}, g_2^r)\hat{e}(op_i, g_2) = \hat{e}(o_i^{m_i'}, g_2^r)\hat{e}(op_i', g_2)$

which means that

$\hat{e}(o_i, g_2^r)^{m_i - m_i'} = \hat{e}(op_i'/op_i, g_2)$

Since $o_i = g_1^r, op_i'/op_i = (g_1^{r^2})^{m_i - m_i'}$, we have:

$\hat{e}(o_i, g_2^r)^{m_i - m_i'} = \hat{e}(g_1, g_2)^{r^2(m_i - m_i')}$

$\hat{e}(op_i'/op_i, g_2) = \hat{e}((g_1^{r^2})^{m_i - m_i'}, g_2) = \hat{e}(g_1, g_2)^{r^2(m_i - m_i')}$,

which justifies the correctness of $\mathcal{B}$'s output. Therefore, if the Square Computational Diffie-Hellan assumption holds, the scheme satisfies the position-binding property. □

Other security requirements

Policy-hiding

Policy-hiding means that the ciphertext policy is hidden from inspection. In our approach, we achieve a weaker concept known as weakly attribute-hiding, which ensures that the policy remains unknown to all parties except those who can decrypt the ciphertext. Our access control system is constructed on top of the decentralized inner-product predicate encryption scheme in *Michalevsky & Joye (2018)*. It provides detailed proof that, in the existence of corrupted authorities, the advantage of a PPT adversary $\mathcal{A}$ in winning a sequence of games is negligible in the security parameter λ.

Trustability

Most existing solutions require an intermediary entity to ensure reliable and secure data management, resulting in expensive costs to prevent a single point of failure and privacy leakage. To overcome these obstacles, our approach employs the characteristics of blockchain distribution, decentralization, transparency, and immutability. By publishing the encrypted metadata, which consists of the AES key AK and file location loc, to the blockchain, we can maintain the integrity of access control management without requiring any intermediaries.

Traceability

Our system can track and validate access control data on the blockchain. Any activities, including setup, registration, key generation, encryption, and data uploading, are recorded as immutable transactions.

Table 5 Summary of system using attribute-based encryption.

Approach	Group	Security	Auth setup	Encryption	Decryption
Wang, Zhang & Zhang (2018)	Prime	S-Rom	1 pair + 2 exp	3 exp	2 pair
Cui et al. (2018)	Prime	S-STM	$(16I+3)$ exp + $(12I+9)$ pm	$(8I+2)$exp	$(6I+1)$pair + Iexp
Zhang, Zheng & Deng (2018)	Composite	F-STM	$5I$ exp + $6I$ pm	$(6I+4)$exp*	$(I+2)$pair + $2I$exp*
Yang et al. (2018)	Composite	F-ROM	1 pair + 2 exp 2 pm	$(7I+1)$exp	$2I$pair + $4I$exp
Gao et al. (2020)	Composite	F-STM	1 pair + 2 exp + 2 pm	$(l+2)$exp	$(l+1)$pair
Qin et al. (2021)	Composite	F-Rom	1 pair + $(2n+2)$exp + $2n$ pm	$(5I+1)$ exp+pair	$2I$pair+ Iexp
Zhao et al. (2022)	Prime	S-STM	2 exp	$5I$pair + 4exp + 2pm	$2I$pair + $3I$exp
Tseng & Gao (2022)	Prime	S-ROM	1 pair + $(l+2)$ exp	$(l+2)$exp + $(ln+2n+l+1)$pm	2 pair + $(l-1)$ exp + $l(n+1)$ pm
This work	Prime	F-ROM**	2pair + 23exp + 16 pm	$(2n+1)$exp	2 pair + $2n$ exp

Notes.

*Several works did not present the complexity information. Therefore, we have either extrapolated the complexity on our own or referenced results presented in the survey (*Zhang et al., 2020b*).

**This is not entirely accurate. We primarily address the rogue-key attack under the fully adaptive security framework. Thus, the actual security of our scheme is slightly stronger than selective security but does not fully achieve fully adaptive security.

PERFORMANCE ANALYSIS

In this section, we present a comparative analysis of our proposed Blockchain-enabled data governance system against existing related works *Wang, Zhang & Zhang (2018)*, *Qin et al. (2021)*, *Nishide, Yoneyama & Ohta (2008)*, *Gao et al. (2020)*, *Cui et al. (2018)*, *Zhang, Zheng & Deng (2018)*, *Yang et al. (2018)*, *Zhao et al. (2022)*, *Tseng & Gao (2022)* in terms of performance and security. In Table 5, we position these aforementioned works that are closely related to our work, providing comprehensive comparisons in the bilinear mapping group, security model, and time complexity of **Auth Setup**, **Encryption**, and **Decryption**.

It is important to note that real execution times, a critical aspect of performance evaluation, are often inconsistently reported in the literature. Among the works previously discussed, only (*Gao et al., 2020*; *Cui et al., 2018*; *Zhang, Zheng & Deng, 2018*; *Zhao et al., 2022*; *Tseng & Gao, 2022*) provide real execution times, but these are based on diverse platforms. Furthermore, the lack of a shared code base for these works precludes the replication of experiments to verify and compare execution times in a uniform environment.

As a result, we have selected a few works (*Michalevsky & Joye, 2018*; *Tseng & Gao, 2022*) for implementation. These works were chosen because both are IPPE-based schemes, and Tseng and Gao claimed that their work outperforms MJ's scheme, upon which our construction is built, in the four algorithms: **Setup**, **Auth Setup**, **Encryption**, and **Decryption**.

It is important to note that we did not include the trusted setup in the comparison for the following reasons:

1. A malicious central authority is not the main concern of this work.
2. Traffic delays, such as transactions not being mined, are quite common in blockchain networks. Therefore, the complexity cost of the **trusted setup** may not accurately represent real-time execution. In contrast, proof generation and verification in **Auth**

Setup for the attribute secret keys can be done offline, which is not impacted by network delays.

Asymptotic comparisons

The notations used are: *exp* for exponentiation operation, *pair* for bilinear pair operation, *pm* for point multiplication operation, and I for access policy complexity for LSSS.

1. Group: There are two types of groups used in CP-ABE: prime-order and composite-order groups. It is noted that the design of prime-order CP-ABE is more efficient than composite-order CP-ABE but is more challenging to achieve full security.
2. Security model: Standard model (STM) and random oracle model (ROM) are two typical types of security models considered in the CP-ABE scheme. And adversaries are categorized into selective adversaries and adaptive adversaries. If a CP-ABE scheme is secure against adaptive adversaries under the standard model, we denote this scheme as F-STM achieving full security. Likewise, S-ROM represents a CP-ABE scheme robust against selective adversaries under the random oracle model, and F-ROM if it is secure against adaptive adversaries under the random oracle model.
3. Computation cost: This assessment considers both the encryption and decryption costs in terms of their complexity, measured in terms of certain standard (cryptographic) operations. The following notations are used:
   - *exp*: exponentiation operation
   - *pair*: bilinear pair operation
   - *pm*: point multiplication operation
   - I: the access policy complexity for LSSS
   - l: the size of attributes
   - n: the size of attribute authorities

In terms of computational costs, bilinear pairing (pair) is the most expensive operation compared to exponentiation (exp) and point multiplication (pm), and the LSSS is a special matrix whose rows are labeled by attributes such that the cost of I might be similar as n or l. Note that, each attribute authority is responsible for at least 1 attribute, which means that $n \leq l$. Consequently, our scheme is superior to most of these works with respect to encryption and decryption cost, as shown in Table 5, but it is constrained in terms of the conjunction access policy.

Experimental result

In this section, we present the experimental results of two IPPE-based schemes (*Michalevsky & Joye, 2018*; *Tseng & Gao, 2022*) and our enhanced construction, implemented in Python. We analyze the execution time in terms of all five algorithms defined for IPPE (Definition 5). The experiments were conducted on a platform featuring an i5-13600KF 20-Core processor with 128GB of RAM, running Ubuntu version 22.04.1 LTS. These implementations were developed using Python and the Charm-Crypto@0.50 cryptography library (*Akinyele et al., 2013*), utilizing the 'MNT-159' elliptic curve for asymmetric bilinear mapping and the 'SS512' elliptic curve for symmetric bilinear mapping. According to *Cui et al. (2018)*, both curves provide an 80-bit security level. We set $k = 2$ as a standard value for the security parameter, which is widely adopted in similar works.

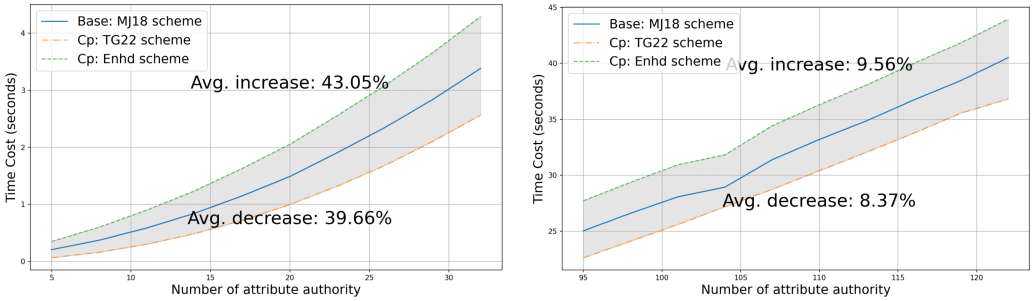

Figure 2 **Comparision of total cost of schemes with varied numbers of attribute authorities.**
Full-size DOI: 10.7717/peerjcs.2581/fig-2

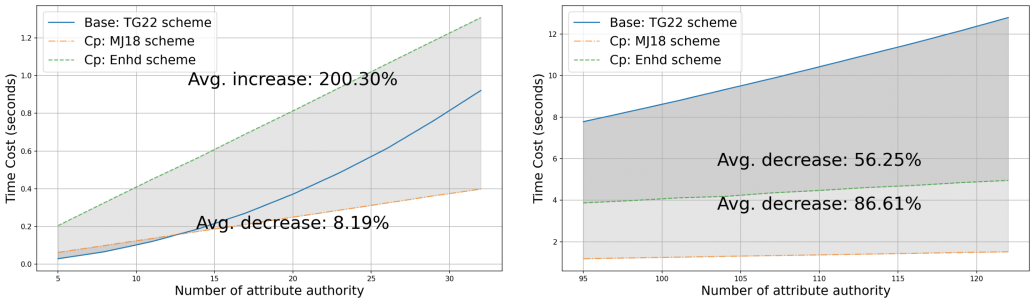

Figure 3 **Comparison of auth setup with varied numbers of attribute authorities.**
Full-size DOI: 10.7717/peerjcs.2581/fig-3

In our performance evaluation, we use the abbreviations MJ18, TG22, and Enhd to represent the three schemes from *Michalevsky & Joye (2018)*, *Tseng & Gao (2022)*, and our proposed enhanced IPPE-based scheme, respectively, as depicted in the following figures.

Using Michalevsky and Joye's work (*Michalevsky & Joye, 2018*) as the baseline (represented by the blue line in Fig. 2), we observe that our enhanced construction requires significantly more computation time, as indicated by the green dashed line. In contrast, *Tseng & Gao (2022)*, depicted by the orange dash-dotted line, outperforms the baseline in efficiency. Meanwhile, we note that as the number of attribute authorities increases, the performance differences—both in terms of average increase and decrease—shrink from around 40%, as seen in Fig. 2A to 10%, as shown in Fig. 2B. We note that these relative computational cost overheads align with our expectations qualitatively, specifically, the enhanced security achieved with our approach incurs more computation.

To further investigate the factors contributing to these performance differences quantitatively, we analyzed the time cost of each algorithm, focusing initially on the **Auth Setup** phase in Fig. 3.

Instead of using the work of *Michalevsky & Joye (2018)* as the baseline for total cost comparison, we have selected *Tseng & Gao (2022)* as the baseline for evaluating the performance of the **Auth Setup** algorithm. Surprisingly, although the scheme of *Tseng & Gao (2022)* scheme demonstrated efficiency in previous comparisons, its performance in the **Auth Setup** phase (represented by the blue line in Fig. 3) is not superior to *Michalevsky*

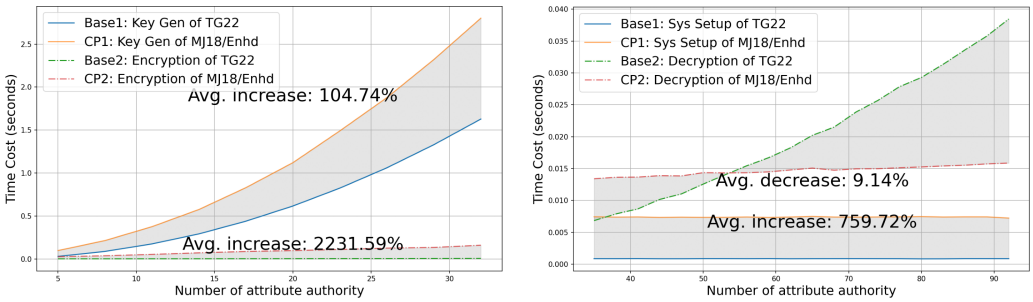

Figure 4 **Comparison of other algorithms with varied numbers of attribute authorities.**
Full-size DOI: 10.7717/peerjcs.2581/fig-4

& Joye (2018) (indicated by the orange dash-dotted line) even when the number of attribute authorities is small, as shown in Fig. 3A. Additionally, the performance gap between our proposed scheme (depicted by the green dashed line) and the scheme of *Tseng & Gao (2022)* narrows as the number of attribute authorities increases. When n exceeds 95, our enhanced scheme, which incorporates NIZK proofs to defend against rogue-key attacks, outperforms the scheme of *Tseng & Gao (2022)*, as illustrated in Fig. 3B.

Since our enhanced scheme only modifies the **Auth Setup**, the performance results of the other algorithms—**Sys Setup**, **Key Gen**, **Encryption**, and **Decryption**—are identical to those in *Michalevsky & Joye (2018)*. The comparison among the three schemes is summarized in Fig. 4.

As shown in Figs. 4A and 4B, the **Encryption** and **Decryption** algorithms, all schemes (with the scheme of *Tseng & Gao (2022)* represented by the green dash-dotted line, *Michalevsky & Joye (2018)* and our enhanced scheme both by the red dash-dotted line) exhibit linear growth as the number of attribute authorities increases. Specifically, the **Decryption** algorithm in the work of *Tseng & Gao (2022)* increases at a faster rate than our enhanced scheme, surpassing our scheme's time cost when the number of attribute authorities n reaches 53. For the **Key Gen** algorithms, all three schemes(with TG's scheme represented by the blue line, MJ's work, and ours both by the orange line) exhibit exponential growth; however, the performance gap widens as the number of authorities increases. In contrast, the **Sys Setup** algorithm remains stable and is unaffected by the number of attribute authorities, maintaining consistent performance even as n increases from 35 to 95.

In summary, although our proposed scheme with Non-Interactive Zero-Knowledge (NIZK) proofs to defend against rogue-key attacks performs better in the **Auth Setup** and **Decryption** algorithms compared to the scheme introduced in *Tseng & Gao (2022)*, the poor performance of the inherited **Sys Setup**, **Encryption**, and especially **Key Generation** algorithms from *Michalevsky & Joye (2018)* causes our scheme to require additional time in the total cost, as shown in Fig. 2. However, as illustrated in Figs. 3B and 4B, the gap between our scheme and the work of *Tseng & Gao (2022)* narrows as the number of attribute authorities n increases, and eventually our scheme outperforms theirs. This indicates that the relative cost of the additional security measures in our scheme diminishes

with a larger number of attribute authorities, suggesting that our scheme is suitable for scenarios involving a large number of attribute authorities, while, when the number of authorities is small, our approach is also practical in terms of the absolute cost even though it is relatively costlier than the other schemes.

CONCLUDING REMARKS

Securing cloud data access and protecting identity privacy are legitimate concerns for many use cases, which this work addresses with a blockchain-based data governance system that is secure and privacy-preserving. A combination of attribute-based encryption (ABE) and the Advanced Encryption Standard (AES) makes the system efficient and responsive to real-world conditions. Our data-sharing system can handle multi-authority scenarios while protecting identity privacy and hiding ABE's policy against rogue-key attacks under **fully adaptive corruption**.

However, because our system is built on top of *Michalevsky & Joye (2018)* and *Bowe, Gabizon & Green (2018)*, it has inevitably inherited a few drawbacks: First, it only supports fixed-size attributes and authorities, which means that any changes to these may necessitate requesting a new system setup or a new key for DU from all authorities. Second, because each AA's public key must be shared with others for the computation of masking terms μ_i, the system requires coordination among authorities during the setup phase. Third, the implementation of a trusted setup to protect against rogue-key attacks adds complexity to the setup process, making it less suitable for scenarios where authorities frequently join or leave. Addressing these limitations is part of our planned future work.

Our proposed system addresses the key challenges of decentralization, practicality, and security. Specifically, it provides resilience against rogue-key attacks, considers dynamic setups through fully adaptive security, and mitigates the potential risk of information leakage caused by inference attacks. Considering the immediacy of the more fundamental security needs, the current work does not explore further common real-world desirable functionalities, such as interoperability, enabling data-sharing across different cloud environments, scalability, particularly in handling large-scale data and high-concurrency access, and other desired features like trustworthy and privacy-preserving keyword search schemes in encrypted decentralized storage. Furthermore, we aim to extend our research to address specific security compliance requirements in cloud environments, such as data governance, data sovereignty, and cross-regional data transfer. Addressing these aspects would significantly enhance the applicability of our system in real-world scenarios.

Thus, our future work will focus on addressing these shortcomings while further exploring ABE-based data-sharing systems in terms of decentralization, flexibility, interoperability, scalability, and security.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

The authors received no funding for this work.

Competing Interests

Anwitaman Datta is an Academic Editor for PeerJ Computer Science.

Author Contributions

- Jingchi Zhang conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the article, and approved the final draft.
- Anwitaman Datta conceived and designed the experiments, authored or reviewed drafts of the article, and approved the final draft.

Data Availability

The following information was supplied regarding data availability:

The code is available at Zenodo: Zhang, J. (2024). Performance comparison in three IPPE-based schemes. Zenodo. https://doi.org/10.5281/zenodo.13950139.

REFERENCES

Agrawal S, Goyal R, Tomida J. 2021. Multi-party functional encryption. In: *Theory of cryptography: 19th international conference, TCC 2021, Raleigh, NC, USA, November 8–11, 2021, Proceedings, Part II*. Cham: Springer, 224–255 DOI 10.1007/978-3-030-90453-1_8.

Akinyele JA, Garman C, Miers I, Pagano MW, Rushanan M, Green M, Rubin AD. 2013. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering* **3**:111–128 DOI 10.1007/s13389-013-0057-3.

Alowolodu O, Alese B, Adetunmbi A, Adewale O, Ogundele O. 2013. Elliptic curve cryptography for securing cloud computing applications. *International Journal of Computer Applications* **66(23)**:10–17 DOI 10.5120/11254-5818.

Ballard L, Green M, De Medeiros B, Monrose F. 2005. Correlation-resistant storage via keyword-searchable encryption. *Available at https://eprint.iacr.org/2005/417*.

Belguith S, Kaaniche N, Laurent M, Jemai A, Attia R. 2018. Phoabe: securely outsourcing multi-authority attribute based encryption with policy hidden for cloud assisted iot. *Computer Networks* **133**:141–156 DOI 10.1016/j.comnet.2018.01.036.

Benet J. 2014. Ipfs-content addressed, versioned, p2p file system. ArXiv arXiv:1407.3561.

Bethencourt J, Sahai A, Waters B. 2007. Ciphertext-policy attribute-based encryption. In: *2007 IEEE symposium on security and privacy (SP'07)*. Piscataway: IEEE, 321–334.

Boneh D, Boyen X, Shacham H. 2004. Short group signatures. In: *Annual international cryptology conference*. Cham: Springer, 41–55.

Bowe S, Gabizon A, Green MD. 2018. A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK. In: *International conference on financial cryptography and data security*. Cham: Springer, 64–77.

Bowe S, Gabizon A, Miers I. 2017. Scalable multi-party computation for zk-SNARK parameters in the random beacon model. *Available at https://eprint.iacr.org/2017/1050*.

Burmester M, Desmedt Y, Seberry J. 1998. Equitable key escrow with limited time span (or, how to enforce time expiration cryptographically) extended abstract. In: *International conference on the theory and application of cryptology and information security*. Cham: Springer, 380–391.

Catalano D, Fiore D. 2013. Vector commitments and their applications. In: *International workshop on public key cryptography*. Cham: Springer, 55–72.

Chase M. 2007. Multi-authority attribute based encryption. In: *Theory of cryptography conference*. Cham: Springer, 515–534.

Chase M, Chow SS. 2009. Improving privacy and security in multi-authority attribute-based encryption. In: *Proceedings of the 16th ACM conference on Computer and communications security*. New York: ACM, 121–130.

Chen J, Chu Q, Gao Y, Ning J, Wang L. 2023. Improved fully adaptive decentralized MA-ABE for NC1 from MDDH. In: *International conference on the theory and application of cryptology and information security*. Cham: Springer, 3–32.

Cui H, Deng RH, Lai J, Yi X, Nepal S. 2018. An efficient and expressive ciphertext-policy attribute-based encryption scheme with partially hidden access structures, revisited. *Computer Networks* **133**:157–165 DOI 10.1016/j.comnet.2018.01.034.

Datta P, Komargodski I, Waters B. 2023. Fully adaptive decentralized multi-authority ABE. In: *Annual international conference on the theory and applications of cryptographic techniques*. Cham: Springer, 447–478.

Dong C, Chen L, Wen Z. 2013. When private set intersection meets big data: an efficient and scalable protocol. In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. New York: ACM, 789–800.

Gao S, Piao G, Zhu J, Ma X, Ma J. 2020. Trustaccess: a trustworthy secure ciphertext-policy and attribute hiding access control scheme based on blockchain. *IEEE Transactions on Vehicular Technology* **69(6)**:5784–5798 DOI 10.1109/TVT.2020.2967099.

Goyal V, Pandey O, Sahai A, Waters B. 2006. Attribute-based encryption for fine-grained access control of encrypted data. In: *Proceedings of the 13th ACM conference on Computer and communications security*. New York: ACM, 89–98.

Hei Y, Liu J, Feng H, Li D, Liu Y, Wu Q. 2021. Making MA-ABE fully accountable: a blockchain-based approach for secure digital right management. *Computer Networks* **191**:108029 DOI 10.1016/j.comnet.2021.108029.

Kaur J, Rani R, Kalra N. 2024. Attribute-based access control scheme for secure storage and sharing of EHRs using blockchain and IPFS. *Cluster Computing* **27(1)**:1047–1061 DOI 10.1007/s10586-023-04038-2.

Lai J, Deng RH, Li Y. 2011. Fully secure cipertext-policy hiding CP-ABE. In: *Information security practice and experience: 7th international conference, ISPEC 2011, Guangzhou, China, May 30–June 1, 2011. Proceedings 7*. Cham: Springer, 24–39.

Lewko A, Waters B. 2011. Decentralizing attribute-based encryption. In: *Annual international conference on the theory and applications of cryptographic techniques*. Cham: Springer, 568–588.

Li J, Zhang Y, Ning J, Huang X, Poh GS, Wang D. 2022. Attribute based encryption with privacy protection and accountability for CloudIoT. *IEEE Transactions on Cloud Computing* **10(2)**:762–773 DOI 10.1109/TCC.2020.2975184.

Maesa DDF, Mori P, Ricci L. 2017. Blockchain based access control. In: *IFIP international conference on distributed applications and interoperable systems*. Cham: Springer, 206–220.

Michalevsky Y, Joye M. 2018. Decentralized policy-hiding ABE with receiver privacy. In: *European symposium on research in computer security*. Cham: Springer, 548–567.

Mitchell C. 2022. Activists respond to Apple choosing encryption over invasive image scanning plans. The Verge. *Available at https://www.theverge.com/2022/12/9/23500838/apple-csam-plans-dropped-eff-ncmec-cdt-reactions*.

Nakamoto S. 2008. Bitcoin: a peer-to-peer electronic cash system. *Available at https://bitcoin.org/bitcoin.pdf* .

Nasiraee H, Ashouri-Talouki M. 2020. Anonymous decentralized attribute-based access control for cloud-assisted IoT. *Future Generation Computer Systems* **110**:45–56 DOI 10.1016/j.future.2020.04.011.

Nishide T, Yoneyama K, Ohta K. 2008. Attribute-based encryption with partially hidden encryptor-specified access structures. In: *International conference on applied cryptography and network security*. Cham: Springer, 111–129.

Ouaddah A, Abou Elkalam A, Ait Ouahman A. 2016. FairAccess: a new Blockchain-based access control framework for the Internet of Things. *Security and communication networks* **9(18)**:5943–5964 DOI 10.1002/sec.1748.

Porwal S, Mittal S. 2020. A privacy preserving and efficient multi authority–CP-ABE scheme for secure cloud communication. *Journal of Cyber Security and Mobility* **9(4)**:601–626 DOI 10.13052/jcsm2245-1439.945.

Qin X, Huang Y, Yang Z, Li X. 2021. A blockchain-based access control scheme with multiple attribute authorities for secure cloud data sharing. *Journal of Systems Architecture* **112**:101854 DOI 10.1016/j.sysarc.2020.101854.

Ristenpart T, Yilek S. 2007. The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks. In: *Annual international conference on the theory and applications of cryptographic techniques*. Cham: Springer, 228–245.

Rouselakis Y, Waters B. 2013. Practical constructions and new proof methods for large universe attribute-based encryption. In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. New York: ACM, 463–474.

Rouselakis Y, Waters B. 2015. Efficient statically-secure large-universe multi-authority attribute-based encryption. In: *International conference on financial cryptography and data security*. Cham: Springer, 315–332.

Sahai A, Waters B. 2005. Fuzzy identity-based encryption. In: *Annual international conference on the theory and applications of cryptographic techniques*. Cham: Springer, 457–473.

Schnorr C-P. 1990. Efficient identification and signatures for smart cards. In: *Advances in cryptology—CRYPTO'89 proceedings*. Cham: Springer, 239–252.

Sudha M, Monica M. 2012. Enhanced security framework to ensure data security in cloud computing using cryptography. *Advances in Computer Science and its Applications* **1(1)**:32–37.

Tseng Y-F, Gao S-J. 2022. Decentralized inner-product encryption with constant-size ciphertext. *Applied Sciences* **12(2)**:636 DOI 10.3390/app12020636.

Wang S, Zhang Y, Zhang Y. 2018. A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. *IEEE Access* **6**:38437–38450 DOI 10.1109/ACCESS.2018.2851611.

Wang X, Zhang J, Schooler EM, Ion M. 2014. Performance evaluation of attribute-based encryption: Toward data privacy in the IoT. In: *2014 IEEE international conference on communications (ICC)*. Piscataway: IEEE, 725–730.

Wilcox Z. 2016. The design of the ceremony. Electric Coin Company (blog post). *Available at https://electriccoin.co/blog/the-design-of-the-ceremony/*.

Wood G. 2014. Ethereum: a secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper* **151(2014)**:1–32.

Wu A, Zhang Y, Zheng X, Guo R, Zhao Q, Zheng D. 2019. Efficient and privacy-preserving traceable attribute-based encryption in blockchain. *Annals of Telecommunications* **74**:401–411 DOI 10.1007/s12243-018-00699-y.

Wu N, Xu L, Zhu L. 2023. A blockchain based access control scheme with hidden policy and attribute. *Future Generation Computer Systems* **141**:186–196 DOI 10.1016/j.future.2022.11.006.

Xu D, Peng C, Wang W, Dev K, Khowaja SA, Tian Y. 2023a. Multikeyword-ranked search scheme supporting extreme environments for internet of vehicles. *IEEE Internet of Things Journal* **11(3)**:3868–3880 DOI 10.1109/JIOT.2023.3275386.

Xu D, Peng C, Wang W, Liu H, Shaikh SA, Tian Y. 2023b. Privacy-preserving dynamic multi-keyword ranked search scheme in multi-user settings. *IEEE Transactions on Consumer Electronics* **69(4)**:890–901 DOI 10.1109/TCE.2023.3269045.

Xue K, Xue Y, Hong J, Li W, Yue H, Wei DS, Hong P. 2017. RAAC: robust and auditable access control with multiple attribute authorities for public cloud storage. *IEEE Transactions on Information Forensics and Security* **12(4)**:953–967 DOI 10.1109/TIFS.2016.2647222.

Yang Y, Chen X, Chen H, Du X. 2018. Improving privacy and security in decentralizing multi-authority attribute-based encryption in cloud computing. *IEEE Access* **6**:18009–18021 DOI 10.1109/ACCESS.2018.2820182.

Yan L, Rong C, Zhao G. 2009. Strengthen cloud computing security with federal identity management using hierarchical identity-based cryptography. In: *IEEE international conference on cloud computing*. Piscataway: IEEE, 167–177.

Yang C, Tan L, Shi N, Xu B, Cao Y, Yu K. 2020. AuthPrivacyChain: a blockchain-based access control framework with privacy protection in cloud. *IEEE Access* **8**:70604–70615 DOI 10.1109/ACCESS.2020.2985762.

Zhang J, Datta A. 2023. Blockchain-enabled data governance for privacy-preserved sharing of confidential Data. ArXiv arXiv:2309.04125.

Zhang Y, Deng RH, Xu S, Sun J, Li Q, Zheng D. 2020b. Attribute-based encryption for cloud computing access control: a survey. *ACM Computing Surveys (CSUR)* **53(4)**:1–41 DOI 10.1145/3398036.

Zhang L, Ren J, Kang L, Wang B. 2021. Decentralizing multi-authority attribute-based access control scheme with fully hidden policy. *International Journal of Network Security* **23(4)**:588–603 DOI 10.6633/IJNS.20210723(4).05.

Zhang L, Ren J, Mu Y, Wang B. 2020a. Privacy-preserving multi-authority attribute-based data sharing framework for smart grid. *IEEE Access* **8**:23294–23307 DOI 10.1109/ACCESS.2020.2970272.

Zhang Y, Zheng D, Chen X, Li J, Li H. 2014. Computationally efficient ciphertext-policy attribute-based encryption with constant-size ciphertexts. In: Chow SSM, Liu JK, Hui LCK, Yiu SM, eds. *Provable Security*. 8782. Cham: Springer International Publishing, 259–273 DOI 10.1007/978-3-319-12475-9_18.

Zhang Y, Zheng D, Deng RH. 2018. Security and privacy in smart health: efficient policy-hiding attribute-based access control. *IEEE Internet of Things Journal* **5(3)**:2130–2145 DOI 10.1109/JIOT.2018.2825289.

Zhao C, Xu L, Li J, Fang H, Zhang Y. 2022. Toward secure and privacy-preserving cloud data sharing: online/offline multiauthority CP-ABE with hidden policy. *IEEE Systems Journal* **16(3)**:4804–4815 DOI 10.1109/JSYST.2022.3169601.

Zhong H, Zhu W, Xu Y, Cui J. 2016. Multi-authority attribute-based encryption access control scheme with policy hidden for cloud storage. *Soft Computing* **22**:243–251 DOI 10.1007/s00500-016-2330-8.