A Blockchain-based Data Sharing Scheme using Attribute-Based Encryption and Fungible Tokens

Ashleen Daly*, Gyu Myoung Lee†, Nguyen B. Truong*
* School of Computing Science, University of Glasgow, United Kingdom
† School of Computer Science and Mathematics, Liverpool John Moores University, United Kingdomm
Emails: 2552805d@student.gla.ac.uk; G.M.Lee@ljmu.ac.uk; and Nguyen.Truong@glasgow.ac.uk

***Abstract*—Over the last few years, a number of serious data breaches have intensively occurred that have raised a significant public awareness on data privacy whilst motivated researchers to envisage novel data sharing and management schemes. Meanwhile, personal information is becoming a valuable commodity; however, the business model used by service providers rarely involves sharing the revenue made by selling personal data, with the individuals - the owners of the data. This paper proposes an alternative storage and data sharing method leveraging Blockchain technology, Smart Contracts, and Attributed-based Encryption. Blockchain's core features such as transparency and immutability of transactions is utilised to create a decentralised system that allows individuals to maintain control over their data and directly benefit from it. Smart contracts integrated with Attributed-based encryption schemes can be leveraged to allow users to securely share the data in an automate process, enabling them to receive a fair share of revenue for the use of their data by the service providers.**

***Index Terms*—Attributed-Based Encryption, Blockchain, Data Sharing, GDPR, Privacy, Fungible Token.**

I. Introduction

In recent years, there as been a significant increase in the amount personal data created and shared between individuals for a variety of applications and services. This surge in data generation has created a network of companies that view personal information as a valuable commodity. However, the business model used by these companies rarely involve sharing the revenue made by selling individuals person data, with the individuals - the true owners of the data. Moreover, the centralized nature of huge data storage and processing presents several critical privacy and security concerns such as identify theft, financial loss and privacy invasions [1]. In addition, with the introduction of Data Protection and Privacy regulations, such as the General Data Protection Regulation (GDPR) in EU/UK, companies are now being subject to stringent requirements on how they collect, store, and process personal data. While the aim of the GDPR is to protect user privacy and ensure data security, complying with it has become an increasingly expensive task for service providers.

While the centralized client-server model has enabled unprecedented connectivity and convenience for individuals around the globe, it has also raised critical issues around privacy, security and economic fairness. Addressing these concerns is critical for building a more security and user-centric digital future. These issues call out for novel systems where both users and companies have less costs, more efficiency and a better online experience. This paper proposes an alternative storage and data-sharing method leveraging Blockchain technology with Smart Contracts (SC), the Inter-Planetary File System (IPFS) distributed file system, and the Attribute-based Encryption (ABE) scheme. The propose solution prototypes a performant and GDPR-compliant decentralised data-sharing system that allows users to store and share their data over a public network securely with an automate access control to the data. Utilising blockchains core features; transparency and immutability of transactions, with the advanced features of ABE, it would be possible to create a system that allows individuals to securely maintain control over data and directly benefit from it. Moreover, SCs, implemented a Semi-Fungible Token (SFT) representing users' attributes in ABE, can be leveraged to allows users to automate control over their data, paving the way to receive a fair share of revenue for the use of their data by companies.

The rest of the paper is organised as follows: Section II provides relevant background and related work. Section III discusses how the proposed solution was formulated. Section IV describes the system implementation and performance evaluation results. Finally, Section V concludes the paper with future research directions.

II. Background and Related Work

A. Blockchain and Smart Contracts

A blockchain is fundamentally an electronic ledger of transactions recorded into a chain of blocks that operates on a Peer-to-Peer (P2P) network, where every participant in the network maintains the exact same copy of the ledger. Each block contains a cryptographic hash of the previous block, a timestamp, and data from several transactions. Each participant in a blockchain's P2P network is a computer running the blockchain software called nodes, maintains a copy of the same ledger, and is responsible for keeping the copy synchronized with all other nodes in the network. This is achieved through a consensus mechanism that ensures all transactions are validated and agreed upon by the network without a central authority [2]. Ethereum was the first blockchain built to support Turing-complete, programmable SC [3]. Unlike Bitcoin, which was designed primarily for P2P financial transactions, Ethereum introduced the ability to execute complex contractual agreements directly on the blockchain, incorporating high-level programming languages,

such as Solidity, Serpent, or LLL [4]. SCs are self-executing contracts with the terms of the agreement between the buyer and seller directly written into executable code. The code and agreements contained thereby exist across a decentralized blockchain network. The results of transactions are recorded into a ledge on Ethereum by updating the state of the SC [3].

B. IPFS

IPFS is an off-chain distributed file system that provides decentralized cloud storage using P2P networking and content addressing [5]. IPFS is used side-by-side many blockchain-based applications, such as decentralized web applications (dApps) and Non-Fungible Tokens (NFTs), because it provides an inexpensive alternative to blockchain for storing large amounts of data on a decentralized network [6]. Moreover, its use of cryptographic hashing and content addressing guarantee data integrity, which is common pain-point in decentralized applications where trust is limited. IPFS uses P2P network of nodes that work collectively to provide a distributed file storage system. Each participant in the network operates a node, which can be created by running an implementation of IPFS on a computer. The IPFS network is made up of three main components: a client, a local node, and several remote nodes. The client interacts with the local node, while the local node connects with remote nodes to share and retrieve data. IPFS uses a Distributed Hash Table (DHT) to keep track of which nodes are storing what information so that it is convenient to retrieve any file from the IPFS [7].

C. Attribute-Based Encryption Schemes

ABE is a branch of asymmetric cryptography that enables access control of encrypted data through the use of policies and attributes. Unlike traditional encryption schemes that use a single decryption key, ABE allows for a more flexible and approach to encryption and decryption. In ABE, attributes are associated with an user's private key and data is subsequently encrypted using an `access policy` that specifies which attributes are required to decrypt it. Consequently, ABE facilitates a one-to-many encryption scheme, allowing a single piece of ciphertext to be securely shared with multiple users who meet the specified attribute criteria. There are two classifications of ABE, namely Ciphertext-Policy ABE (CP-ABE) and Key-Policy ABE (KP-ABE) [8], each is suitable to different application scenarios. Our proposed solution uses CP-ABE due to its flexibility in managing access control directly using the ciphertext, allowing access control to be mobile in a data-centric system [9].

D. Related Work

Due to blockchain technology being decentralized, tamper-proof, and transparent, it is widely used in secure data sharing and management schemes. Authors in [10] have proposed a GDPR-compliant data management scheme that utilises blockchain technology and SCs to automate the process of sharing data to third parties. A similar GDPR-compliant data sharing scheme has also been proposed in [11] that made use of a consortium blockchain. In addition, Blockchain technology has also been adopted in CP-ABE data-sharing schemes to facilitate a transparent and trustful storage ledger. In these systems, SCs are used to perform automated actions on stored data, such as partially decrypting it to relieve the burden of decryption from DU, or access control based on a predefined set of conditions. The scheme in [12] depicted an efficient blockchain-assisted CP-ABE scheme for data sharing and access control; and the work in [13] took a step further with partially hidden access structures to enhance privacy.

Despite both CP-ABE schemes leverage IPFS for decentralised cloud storage and SCs for automated and distributed access control, none of them consider the centralization of the attribute authority in their system. If it were to disconnect from the network or become outdated, then the scheme would not be able to function. To remove this point of failure, blockchain utility tokens could be leveraged as attributes in a blockchain-based CP-ABE scheme. Additionally, none of the mentioned blockchain-based CP-ABE schemes provide a GDPR-compliance analysis on their system. Our proposed scheme attempts to fill in these gaps in the research of blockchain-based CP-ABE data-sharing schemes.

III. Challenges and Proposed Solutions

A. Scenarios and Challenges

In the scenarios of personal data sharing, a solution based on CP-ABE scheme typically contains the following participants: (1) `DO`: The owner who wants to share some data with those who satisfy his specific access structure. They will determine the access policy; (2) `DU`: The user who owns a set of attributes and wishes to access the data; (3) `AA`: The attribute authority who participates in public parameter and key generation and maintains the attributes of `DO` and `DU`; and (4) `Cloud Server` responsible for cloud storage management.

How these participants interact with each other is outlined in Fig. 1. First, the authority responsible for generating `Public Key (PK)` and `Master Secret Key (MSK)` will perform a setup. Once DO obtains PK from AA and formulates an access structure $\mathcal{A}$, using attributes defined and maintained by AA. With PK and $\mathcal{A}$, DO can now encrypt their message, say M.

$$\text{Encrypt}(PK, M, \mathcal{A}) = C$$

DO then sends the ciphertext C to the CS for storage. DU can now request C from CS. In order to decrypt C, DU must prove their identity to AA, in return AA will use DU's set of attributes S to generate a decryption key (SK).

$$\text{KeyGen}(\text{MK}, S) = SK$$

Finally, DU obtains their SK and uses it to decrypt C and obtain the original message M.

$$\text{Decrypt}(\text{PK}, C, \text{SK}) = M$$

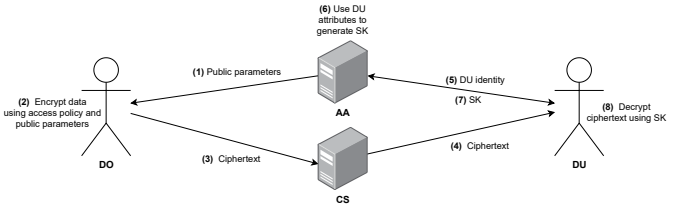

Fig. 1: An overview of a typical CP-ABE scheme.

In those scenarios, existing data sharing schemes that make use of CP-ABE, including those that integrate it with blockchain technology (e.g., in [13], [12]), still depend on a fully trusted, centralized authority `AA` for setting up the keys. This poses several security and operational risks to the overall system. Firstly, as argued by [14], a centralized authority server is vulnerable to cyber-attacks such as Denial of Service (DoS) attacks, potentially impacting the availability of the service. In addition, because the server is a single point of failure, it becomes an attractive target for external malicious actors who may seek unauthorized access of modification to user attributes [1]. Finally, a centralized model does not protect the system against internal threats, where employees with administrative privileges may try to access, modify, or delete data without detection or accountability [15]. All of these combined risks with deploying a centralized `AA` call highlight the need for a novel solution to manage attributes in a blockchain-based CP-ABE data sharing scheme.

B. Proposed Solution

Since the use of blockchain technology for decentralizing CP-ABE schemes has already been explored in the literature, this paper expands blockchain's impact on a CP-ABE scheme by fully decentralising `AA`. This could be achieved using a SC to generate attributes of users. Since SCs operate in a transparent and tamper-proof environment [3], their operations are protected from internal and external threats including DoS attacks [16].

1) Semi-fungible Token as Attributes in ABE: Our solution fully makes use of blockchain technology by representing attributes as tokens. This is a more unique and flexible approach to attribute management compared to the traditional centralised attribute management. In this solution, we utilise `ERC1155`, which is a novel token standard, that aims to combine properties from both the `ERC20` and `ERC721` to create a SFT [17]. This standard allows for greater versatility and makes it suitable for attribute management as it can create and manage a range of attribute types within a single contract. The only downside of this contract, however, is that it is more complex to set up and manage. Implementing a robust and usable management system for these tokens requires additional development effort and considerations to ensure security within the system.

2) System Architecture and Design: A high-level architecture diagram of the proposed blockchain-based data sharing scheme is depicted in Fig. 2, consisting of key components such as Client, SCs, Ethereum blockchain, Key Gen Server (KGS), and IPFS.

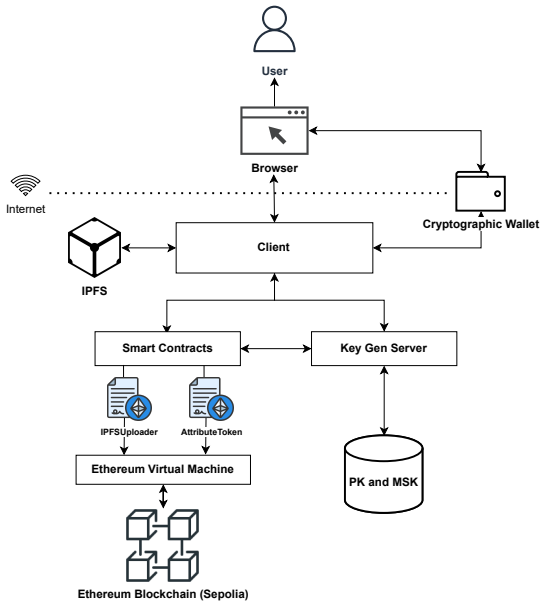

Fig. 2: A high-level architecture diagram of the proposed data sharing scheme leveraging Ethereum blockchain, SFT, and ABE.

1) Client, also known as the front-end, is the part of the system that users can see and interact with, acting as a graphical user interface (GUI).
2) Two SCs are implemented in the system, `IPFSUploader` and `AttributeToken`. `IPFSUploader` manages and keeps track of the uploaded ciphertexts on IPFS. It is also responsible for mapping each upload to a user's wallet address providing a way for Data Processor (DP) (i.e., service provider) to find any ciphertext uploaded by any DO. `AttributeToken` is responsible for implementing the ERC1155 token standard. This allows a DO to **(1)** Mint their attribute tokens, **(2)** Transfer their attribute tokens to a DP, and **(3)** Burn the attribute token transferred to DP to revoke access. This contract performs access control to restrict actions that only be performed by the token creator. This gives the DS full control of who can hold the attribute token to access their data. In addition, `AttributeToken` is responsible for generating attribute lists for users and executed by the KGS.
3) Key Gen Server: KGS is the trusted server responsible for generating the public parameters (PK) and master secret key (MSK) and users' decryption keys based on their held attributes. It maintains the PK and MSK in a centralized database and generates each users decryption key when requested by the user. KGS is the only trusted central authority we rely on in the proposed architecture.
4) IPFS is used as a distributed file storage system to hold all user data due to it having no central server node and hence, avoids the risk of a single point of failure.

As content addressing is used, it ensures that all data uploaded onto IPFS maintains its integrity and cannot be tampered with without changing the `CID` of the data.

C. Data Upload and Retrieve Processes

There are essentially two processes when sharing data in the proposed scheme, `Upload` and `Retrieve`.

1) Upload: The process flow diagram for uploading data using the proposed scheme is illustrated in Fig. 3, where blue arrows represent SC function calls. As illustrated, the system involves the same entities displayed in Fig. 2, with the addition of a new entity, DO. DO is the data owner who expects to share some data with those who satisfy their access structure. In order to encrypt their data, they will determine their own access policy based on attribute tokens they have minted. Note that KGS has already run the Setup function to generate PK and MSK and has stored them in a database; then Upload process flow is as follows:

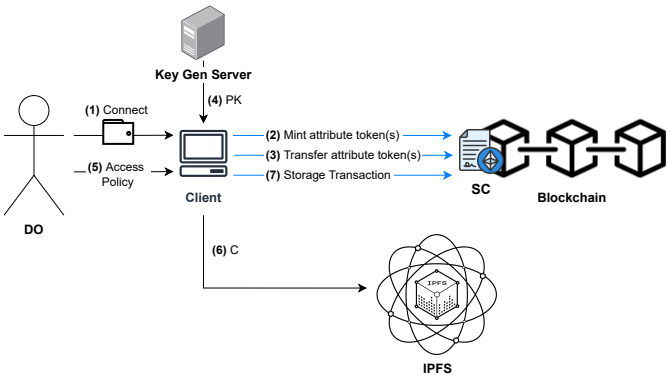

Fig. 3: Flow diagram detailing the process of uploading data to IPFS.

1) DO interacts with the client to mint new and existing attribute tokens. These tokens will then be used to define their access structure.
2) DO then transfers attribute tokens to one or multiple DP, whom they wish to share their information M with.
3) To encrypt M, DO must obtain PK from KGS.
4) Next, DO must determine their own access policy based on the tokens they have minted. DO can now use their access policy along with PK, retrieved in step **(4)**, to encrypt M using the ABE Encrypt function.
5) Ciphertext C is produced and pinned pinned to IPFS. This allows DP to retrieve C and decrypt it if their attributes satisfy the access structure defined by DO.
6) Finally, to confirm that DO has uploaded C to the IPFS network, DO will send an Upload Transaction to the blockchain using a SC, storing the CID of the uploaded C and the access policy used to encrypt it.

2) Retrieve: The process flow diagram depicting how data is retrieved using the proposed scheme is depicted in Fig. 4, where blue arrows again represent SC function calls. In this process, an addition entity is shown in the diagram, DP. DP is the data processor who possesses certain attributes and wants to access the data so they can process it. The process flow is as follows:

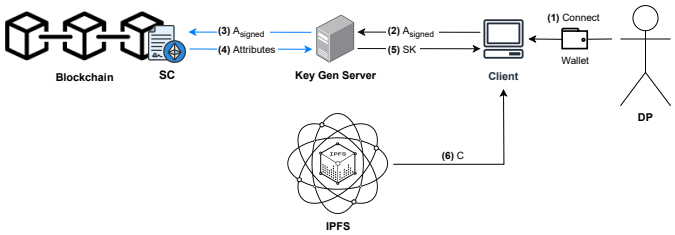

Fig. 4: Flow diagram detailing the process of retrieving data from IPFS.

1) DP obtain decryption key (SK) from KGS. To do this, they send a request to KGS while providing wallet address A, signed by wallets private key to produce A_{signed}, and their wallet's public key PK_{DP}. Their wallet address is sent to allow KGS and supporting SCs to produce the list of attributes associated with the specific wallet and the digital signature is used so that no other wallet can impersonate the DP.
2) KGS then sends A_{signed} to a SC function, whose responsibility is to generate the attribute list corresponding DP's wallet address. First, SC will verify the message is indeed signed by DP. Then, SC will generate DP's attribute list based on their balance of SFTs.
3) The SC will then send DP's attribute list back to KGS, where the server can use them along with MSK to generate a decryption key, SK, for the DP.
4) Once SK is successfully generated, it is then encrypted using DP's wallet public key PK_{DP} and sent back to DP. SK is encrypted with PK_{DP} to ensure that no eavesdropping entity can obtain SK. In order for DP to obtain their SK, they must decrypt the ciphertext using their private key provided by their wallet.
5) Once DP obtains SK, they can read C from the IPFS network and write it to their local IPFS node. Finally, DP can decrypt C using SK, provided that the DP satisfies the access structure used to encrypt M.

IV. System Implementation and Performance

A. Client and KGS Implementation

Client and KGS were implemented on the same code-base and can be refereed to as a web application. React JavaScript (React), an open-source JavaScript library was used for building user interfaces. On top of React, NextJS (Next) framework was used to enable server-side rendering and file-based routing for React-based web applications. Source-code has been public on Github for re-production and improvement[1]. Providing a interface for a user to connect their cryptographic wallet was a core feature of our scheme, as it facilitates authentication through signing transactions and messages. To simplify the problem, this project focused on implementing connections to

[1]https://github.com/nguyentb/decentralized-data-sharing-ABE

`Metamask` cryptocurrency wallet, a browser-extension wallet used to interact with the Ethereum blockchain.

1) CP-ABE: A Rust ABE library was selected to implement the algorithms necessary for CP-ABE in our system. Performing CP-ABE algorithms inside of a browser or web assembly runtime environment was essential so that we have utilised a `rabe-wasm` wrapper. This allowed the KGS to access the KeyGen function by importing the it from the public directory. Once SK has been generated using the KeyGen function, it can then be encrypted using asymmetric cryptography provided by metamask, using PK_{DP}. The function used to encrypt SK is strategically located with all other metamask-related functions inside `utils/metamask.ts`, where it can be imported into the route file containing the API endpoint.

2) SC Integration: The proposed system requires the client to interact with multiple SCs. To do this, EthersJS was imported to provide an interface interacting with Ethereum blockchain. On the other hand, due to the KGS being a server environment without access to a metamask signer, in order for the KGS to call a SC function it had to first start up its own instance of an Ethereum Provider. Once this was complete it had to manually connect to the Sepolia network using Infura, an API provider for connecting to an Ethereum network without having to start up an Ethereum node.

B. Smart Contract Implementation

Both SCs were implemented on the same code-base and reside in a separate Github repository[2].

1) IPFSUploader SC: `IPFSUploader` manages all uploads to IPFS. To implement this functionality, a data structured needed to be implemented to represent an `Upload` containing a `policyString` (access policy) and a `ipfsHash` (IPFS CID). This then made it possible to create a mapping that maps wallet addresses to an array of Upload objects. With an Upload object and a mapping from wallet addresses to an array of Uploads in place, it was straightforward to essentially create a `getter` and `setter` function for Uploads. The function takes in `policyString` and `ipfsHash`, creates an Upload object, and appends it to the mapping where the wallet address is the address of the signer sending the transaction.

2) AttributeToken SC: AttributeToken SC was more challenging to implement due to the complex nature of building an ERC1155-compliant SC. We inherit the ERC1155 and *ERC1155Burnable* contract standard from *OpenZepplin*[3]. This library provided us with all the functions necessary for a SFT contract. In order to tailor these functions to fit the needs of this contract, we first created a *tokenCreator* mapping that mapped the wallet address of initial token creator to the ID of the token. We were now able to implement our own functions including *mintNewToken*, *mintExistingToken*, and *safeTransferFrom*. Finally, a function to generate a list of attributes for a user is also integrated in *AttributeToken* SC.

[2]https://github.com/nguyentb/decentralized-data-sharing-ABE-SC
[3]https://github.com/OpenZeppelin/openzeppelin-contracts

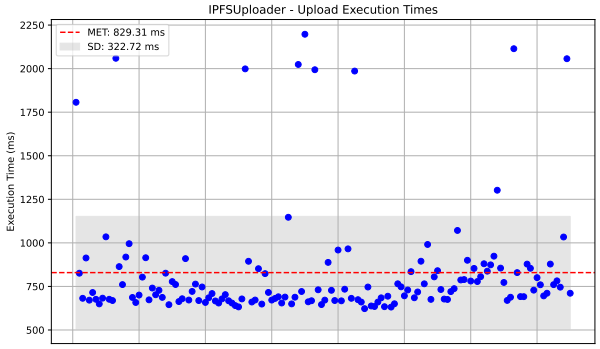

Fig. 5: The execution times of `upload` function in `IPFSUploader`.

C. System Performance

The proposed data-sharing scheme is expected to serve a large number of DOs and DPs uploading and requesting data simultaneously. Hence, the performance and scalability of the system had to be evaluated.

1) Blockchain Performance: This section evaluates the performance of SC functions used within the system. Several key SC functions were executed with varying inputs and the time they took to execute were recorded. For instance, performance of `upload` function is examined by executing the function 150 times with different access policies and IPFS URIs. The mean execution time (MET) was then calculated along with the standard deviation (SD). As can be seen in Fig. 5, the results show a MET $\pm$ SD of $829.31 \pm 322.72ms$, satisfying the same expectations of a well performing REST API which is somewhere between $100ms$ and $1000ms$. However, it is notable that during the evaluation there were some executions that took around $2000ms$ to complete. These are likely due to the Sepolia blockchain experiencing a period of high activity. As more SC functions that alter the state of the blockchain are executed, the more the network experiences increased mining demands, which can result in longer execution times.

2) CP-ABE Performance: Evaluation of the CP-ABE functions implemented in the system are examined which include: `encrypt`, `keygen` and `decrypt`. For instance, we performed several `encrypt` tests, each adding an additional attribute to the access structure, from 1 to 25, as we do not expect users to encrypt their data with more than 25 attributes. From Fig. 6, we can see that the execution time grows linearly as the number of attributes used increases. This is what is expected from CP-ABE encryption times, as shown by [18]. Analysis is also performed on `decrypt` with attributes from 1 to 25. Similar with `encrypt`, there is a strong linear correlation between the execution time and the number of attributes used, ranging from below $1ms$ to around $550ms$.

To analysis `keygen`, we recorded the time it took to generate a decryption key based on how many attributes a user holds, again from 1 to 25. As depicted in Fig. 7, when generating a decryption key, the execution time is not greatly affected by the number of attributes held by the user with some variability in the recorded times in the range of $10ms - 25ms$.

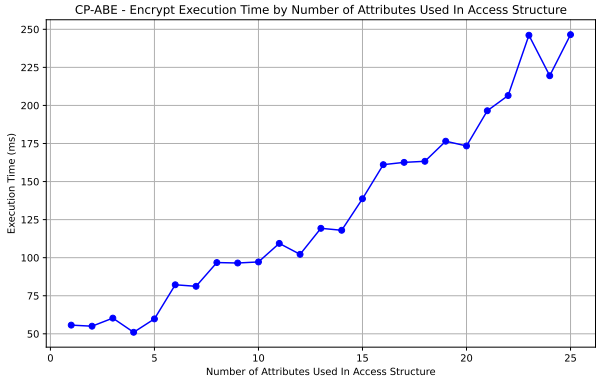

Fig. 6: CP-ABE **encrypt** execution times by the number of attributes used in the access structure.

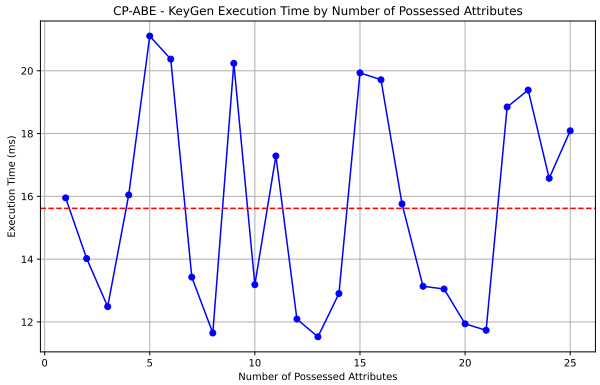

Fig. 7: CP-ABE **keygen** execution times by the number of attributes.

V. Conclusion and Future Work

In this paper, we have proposed a novel, practical, and fully-decentralised blockchain-based data sharing scheme leveraging CP-ABE and SFT on top of the IPFS distributed storage. The key idea that sets out our solution is the use of a SFT (i.e., `ERC1155` standard) as an attribute in CP-ABE so that the proposed scheme is more flexible and less centralised to attribute management compared to the existing ABE-based solutions. Following this catalyst, we have designed and implemented a holistic system that allows DOs to encrypt and upload data to IPFS, where DPs can decrypt the data only if they possess the token. System performance results have shown that such a blockchain-based system performed reasonably well under multiple network requests. It exhibited high variability in execution times; however this is likely due to the network being overcrowded with requests, providing concerns for the scalability of the system and highlighting the need for more efficient Ethereum platform. The CP-ABE encryption scheme used throughout the system was relatively performant with linear complexity in `encrypt` and `decrypt` functions, and acceptable execution time for the `decryption key generation` function.

This work can be further improved in a number of ways. The first direction is to decentralise the KGS, which is the single centralised authority within our proposed system, reducing the risk of a single point of failure within the system. Second direction is to seamlessly integrate the system with IPFS allowing users to update their files efficiently. This could be done by implementing a mechanism that tracks which file a user desires to alter, unpinning that file from IPFS and broadcasting to the network the new updated version of the file. The third direction is to extend more attributes for ABE instead of using only the token by providing more metadata to describe users. This prospective research requires privacy-preservation techniques as the metadata is publicly recorded on-chain and could be exploited in privacy attacks.

References

[1] S. Chakraborty, "Database security threats and how to mitigate them," *MOL2NET'22, Conference on Molecular, Biomed., Comput. Network Science and Engineering, 8th ed.*, 2022, 03 23 2024.

[2] V. Gramoli, "From blockchain consensus back to byzantine consensus," *Future Generation Computer Systems*, vol. 107, pp. 760–769, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17320095

[3] V. Buterin, "A next generation smart contract & decentralized application platform," 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:19568665

[4] R. M. Parizi, Amritraj, and A. Dehghantanha, "Smart contract programming languages on blockchains: An empirical evaluation of usability and security," in *Blockchain–ICBC 2018: First International Conference, Held as Part of the Services Conference Federation, SCF 2018, Seattle, WA, USA, June 25-30, 2018, Proceedings 1*. Springer, 2018, pp. 75–91.

[5] T. V. Doan, Y. Psaras, J. Ott, and V. Bajpai, "Toward decentralized cloud storage with ipfs: Opportunities, challenges, and future considerations," *IEEE Internet Computing*, vol. 26, no. 6, pp. 7–15, 2022.

[6] M. Kaur, "Ipfs: An off-chain storage solution for blockchain," *Proceedings of International Conference on Recent Innovations in Computing*, 2023.

[7] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[8] Y. Z. Dijiang Huang, Qiuxiang Dong, *Attribute-Based Encryption and Access Control*. CRC Press, 2020.

[9] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *2007 IEEE Symposium on Security and Privacy (SP '07)*, 2007, pp. 321–334.

[10] N. B. Truong, K. Sun, G. M. Lee, and Y. Guo, "Gdpr-compliant personal data management: A blockchain-based solution," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1746–1761, 2020.

[11] Y. Piao, K. Ye, and X. Cui, "A data sharing scheme for gdpr-compliance based on consortium blockchain," *Future Internet*, vol. 13, no. 8, 2021. [Online]. Available: https://www.mdpi.com/1999-5903/13/8/217

[12] L. Hong, K. Zhang, J. Gong, H. Qian, and H. Rifà-Pous, "A practical and efficient blockchain-assisted attribute-based encryption scheme for access control and data sharing," *Sec. and Commun. Netw.*, vol. 2022, jan 2022. [Online]. Available: https://doi.org/10.1155/2022/4978802

[13] Y. Ba, X. Hu, Y. Chen, Z. Hao, X. Li, X. Yan, and Q. Jiang, "A blockchain-based cp-abe scheme with partially hidden access structures," *Sec. and Commun. Netw.*, vol. 2021, jan 2021. [Online]. Available: https://doi.org/10.1155/2021/4132597

[14] K. Singh, D. K. S. Dhindsa, and B. Bhushan, "Distributed defense: An edge over centralized defense against ddos attacks," *International Journal of Computer Network and Information Security*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:53545744

[15] S. Mishra and V. Chaurasiya, "Preventing insider attacks in databases," Ph.D. dissertation, 07 2023.

[16] R. F. Ibrahim, Q. Abu Al-Haija, and A. Ahmad, "Ddos attack prevention for internet of thing devices using ethereum blockchain technology," *Sensors*, vol. 22, no. 18, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/18/6806

[17] W. Radomski, A. Cooke, P. Castonguay, J. Therien, E. Binet, and R. Sandford, "Erc-1155: Multi token standard," 2018, 03 23 2024. [Online]. Available: https://eips.ethereum.org/EIPS/eip-1155

[18] A. Mosteiro-Sanchez, M. Barcelo, J. Astorga, and A. Urbieta, "Too many options: A survey of abe libraries for developers," 2022.