Transparent Encryption for IoT using Offline Key Exchange over Public Blockchains

Mamun Abu-Tair[*], Unsub Zia[*], Jamshed Memon[*], Bryan Scotney[*], Jorge Martinez Carracedo[*], Ali Sajjad[**]

Abstract Internet of Things (IoTs) framework involves of a wide range of computing devices that rely on cloud storage for various applications. For instance, monitoring, analytics, surveillance and storing data for later processing within other applications. Due to compliance with security standards and trust issues with third-party cloud storage servers, the IoT data has to be encrypted before moving it to cloud server for storage. However, a major concern with uploading encrypted IoT data to cloud is the management of encryption keys and managing access policies to data. There are several techniques that can be used for storing cryptographic keys used for encryption/decryption of data. For instance, the keys can be stored with encrypted data on the cloud, a third-party key storage vault can be used for storing keys or the keys can stay with client so that they could download and decrypt the data by themselves. In case of encryption keys leakage, the data stored on the cloud storage could be compromised. To resolve the challenge of key management and secure access to data in third-party cloud storage, an end-to-end transparent encryption model has been proposed that securely publishes the cryptographic keys in a blockchain ledger. The data is encrypted at edge gateway before it is transmitted to cloud for storage. The user does not require cryptographic keys to access data; a seamless process involves the client proving their identity to a crypto proxy agent built upon zero trust security principles, ensuring continuous verification.

1 Introduction

The IoT technology has transformed several aspects of life by incorporating innovative and intelligent networking systems. The concept of smart devices has led to

Authors
[*]School of Computing, Ulster University, UK
[**]British Telecom, UK
e-mail: m.abu-tair@ulster.ac.uk

a revolution shift from traditional world to smart world comprising of smart home, smart cities, and industry 4.0 amongst others. Due to undeniable benefits of IoT assisted living, a rapid increase in adoption of IoT technology can be seen [1]. The recently published annual report of Cisco (2018-2023) provides insights to the increasing growth rate of the internet connecting users, devices, and machines [2]. With reference to the Ciscos report, the worldwide ratio of machine to machine (M2M) connections and devices will rise, approximately up to 14.7 billion by 2023, which is much greater in comparison to internet users. The machine to machine (M2M) communication with integrated IoT have gained popularity by contributing towards various applications for example, health care monitoring, camera scrutiny, smart asset tracking services, smart meters, and traffic control. According to the research conducted by McKinsey [3], it is expected that by 2025, the number of IoT connected devices will reach up to 1 trillion. Since IoT and its applications have become prevalent systems representing as an indispensable part of our daily lives, this raises serious concerns about privacy and security vulnerabilities. As tremendous amount of information is being shared every nano second over the heterogeneous network, indirectly creates opportunities for the cyberattacks to exploit security vulnerabilities that cannot be ignored. The IoT devices are resource constrained in terms of low powered with limited computing and storage capabilities [4]. These limited resources expose IoT framework to several security risks allowing malicious individuals to retrieve critical information from the device and collapse the network. As the developments in IoT technologies are continuously accelerating and escalating, one should be aware of the severity caused by different types of cyber-attacks.
In another report conducted by renowned company Hewlett-Packard (HP) stated that to cope with the growing demand of IoT products, security aspects are often compromised during the designing of these products. Approximately, 70% of the IoT based devices are potentially vulnerable and on average, there are 25 vulnerabilities per IoT product due to the weak credentials, un-encrypted networks, and lack of granular user access permissions [5]. This immense distribution of connected devices over the network and the increased deployment of IoT devices has raised their potential towards exploitable vulnerabilities. Consequently, in proportion to the increasing number of IoT devices and services, significant research efforts are required to minimize the rapidly emerging cyber-attacks. The number of IoT devices will continue to grow at an exponential rate, adoption of new and secure mechanisms to mitigate the threats large- scale attacks must be considered, and small-scale attacks are often ignored. No doubt, the former attacks can cause serious damage, but the latter attacks should not be ignored as they are more treacherous and may go undetected for a long period of time causing devastating effects on the reliability of the network. Therefore, adequate, and reliable security measures should be developed that can cope with the competences of constrained technology, ensuring data integrity and confidentiality [6]. Another revolution in the field of internet is the advent of Web 3.0, where blockchain would be used to host internet [7]. We utilize this opportunity to propose a novel security model for securing IoT data using blockchain to store cryptographic keys. The proposed transparent encryption architecture provides remedy for specific problem area, where the IoT data has to

be stored in an untrusted cloud storage. In such scenarios there are three main challenges, i) safe storage of data, ii) key management and iii) access policy for the data. The proposed model provides solution to above mentioned challenges by encrypting data at rest (transparent encryption) and storing the cryptographic keys to the blockchain. By doing this, the encrypted data can be stored in any untrusted cloud storage, moreover, the client can access the data without requiring the keys. This architecture has been filed as patent by British Telecom [8]. The remainder of this paper is organized as follows. Section 2 builds motivation of the paper based on the problem statement. Section 3 covers relevant prior art and inventions. The proposed idea and system architecture has been introduced in Section 3. Section 4 summarizes the contributions made by this research. The proof-of-concept implementation design and experiments is explained in Section 5. The summary of this work and future directions have been identified in Section 6.

1.1 Motivation

Identity management and secure key distribution have always been challenging goals to attain when designing cybersecurity protocols. However, with the speedy adoption of IoT and cloud technology, these goals are much more harder to achieve.

IoT framework is a complex network, comprising of different types of sensors and actuators, producing large amounts of data that is analysed using edge gateway and cloud servers [9]. Too many moving moving points in a system make it vulnerable to attacks. For instance, it is difficult to manage and store the encryption/decryption keys, in cases where the data is has to be accessed by different parties. Moreover, another problem that arises is the management of user identities. Since, each user might need access to the data for a specific time window, makes it difficult to revoke the old keys and encrypt the data using new keys.

The walled garden approach to secure networks does not suffice in the case of IoT framework, rather modern day approaches based on zero trust principles are needed [10]. In situations, where the data is stored on third party cloud storage and there are multiples users accessing the data make it challenging to ensure the integrity of data while maintaining the legitimacy of cryptographic keys used to encrypt data. There is a need of smarter solution, that could allow any number of users to access the data without the need to generate new keys every time and without the need to trust third party cloud or the users. The presently available solutions are either forced to trust the third party cloud server, or have separate key for every user, or in worst cases have to re-encrypt data with a new key every time. The existing solutions that do not require re-encrypting data mostly rely on public key infrastructure, which opens doors to quantum attacks. Thus, neither of the aforementioned approaches provide secure services, efficiency and convenience as in a single package.

1.2 Contributions

In this paper, a novel architecture has been proposed for IoT frameworks to resolve identity and key management problems, enabling secure and seamless access to data. The main contributions made by the proposed model are listed as follows:

- **Transparent Encryption** Transparent data encryption is usually referred for securing data at rest, for instance Microsoft uses it to secure SQL Server, Azure SQL Database, and Azure Synapse Analytics data files. In this paper, the proposed architecture ensures that the data stored in third-party cloud (at rest) is encrypted and the encrypted keys are kept safe.
- **Secure Key Management** Secure management of cryptographic keys is a crucial task, as compromised keys can put the entire system, including the data, at risk. This paper proposes the use of blockchains as a key escrow system to store encrypted keys. Since blockchains serve as a distributed ledger that is accessible from anywhere, key distribution is no longer a problem. The legitimate party can access the keys and thus decrypt the data stored on third-party cloud storage.
- **Seamless Access to data** The uniqueness of this study is the seamless access to data throughout the IoT data lifecycle. The IoT edge server is responsible for encrypting the data and pushing the encryption keys on to the blockchain based distributed ledger system. While on the decryption end, the crypto proxy agent ensures that the data requested by client is provided in unencrypted form, conditional to successful identity check.
- **Zero Trust Policy** The beauty of proposed architecture is that it is built upon zero trust security principles, adhering to explicit verification, assuming breach and least privileged access. Each client is verified explicitly by the crypto proxy agent and is given a limited access to data based on his eligibility conditions.

2 Related work

Several data encryption processes exist where a sender encrypts data and stores it on a cloud service and the relevant decryption key is sent to the intended recipient via a side-channel, e.g., by email, SMS or even postal service. The receiver then downloads the encrypted data from the cloud service and decrypts it using the shared key. In some variants of the above-mentioned process, the decryption key is sent to a trusted third-party key storage vault. The client must authenticate itself with this key management service to retrieve the decryption key and then download and decrypt the data from the cloud service. In [11], the authors proposed an offline data access scheme, where the client downloads data from cloud using privilege keys that are generated using hash for the particular user shared by the owner of the data. However, the limitation with this approach is the exposure to encryption/decryption keys. According to the zero trust security principles, there is no trusted entity, client and cloud both must be treated with zero trust and encryption/decryption keys must be kept secret. In another study, a scheme Cloud stash has been proposed to resolve the key management problem in cloud-based scenarios [12]. They propose to split every file into secret shares and distribute them across different cloud storage, however the file can only be reconstructed with the threshold set number of shares retrieved back. Cloud computing is mainly preferred in situations with bulk data storage; however, the proposed idea seems impractical for large amounts of data. Considering the advent of Web 3.0 [13], a hot area of research is focusing using blockchains to get rid of conventional old school key sharing schemes for securing IoT data [14]. A very basic example could be to encrypt user file and distribute it across decentralized peer blockchain network [15]. However, more advanced solutions have been already proposed to secure IoT data and access management. A study similar to our work also proposes decentralized access management of data stored in cloud but using blockchain based on temporal dimension [16]. However, the way they have used blockchain to verify user is different from our proposed solution. They have used blockchain to perform transaction using attribute-based encryption, whereas we use public blockchains just to store encrypted credentials. Another relevant prior art is a patent for titled as Transparent Proxy of Encrypted Sessions [17]. A client and a server are configured to trust a certificate of an intermediate proxy device. The proxy device intercepts the session requests between the client and the server and establishes a client-proxy session and a proxy-server session which appears transparent to the client and server. The main differences of this patent with our invention are that in our invention the client does not perform online or interactive communication with the server and the Crypto Proxy Agent does not intercept the network traffic between the client and the server. Our proposed model is novel in terms of access management for data stored in third party storage, as the client or third-party cloud never gain access to the encryption keys, yet distributed identity service (DIDS) allows session based access to the clients based on successful verification. Proxy re-encryption is another solution widely used to establish access policies for clients to access data stored in third party cloud storage. In a recently published research, a new kind of conditional proxy re-encryption has been proposed for secure cloud storage [18]. The main idea is to have a semi-trusted crypto proxy agent similar to one in our approach, however once the proxy has necessary re-encryption keys from the delegator, it can transform all ciphertexts for a delegator to ciphertexts for a delegatee. Transforming the ciphertexts and generating individual keys for every delegatee is a resource greedy task. It is equivalent to encrypting the data with new encryption keys for every client. However, in our case the encryption key for the data remains same, the crypto proxy agent just decrypts it for the clients that are able to confirm their identity via DIDS. Wang et al., recently proposed a blockchain-based access control system for secure cloud storage [19]. The proposed approach rely on smart contracts using Ethereum blockchain, that interfaces for data retrieval and storage. The data owner is in charge of developing and executing smart contracts, uploading encrypted files, specifying access control rules, appointing attribute sets to data users, and appending valid access duration. The data user accessing a cloud server-stored encrypted file, needs key from the smart contract which is needed to decrypt the encrypted file. There are few proposals for secure service guarantee from cloud storage is based on Ciphertext-Policy Attribute-Based Encryption (CP-ABE). [20] Hoewever, due to CP-inherent ABE's "all-or-nothing" decryption characteristic, implementation of the protocol may result in an unavoidable security breach known as the misuse of access credentials (i.e., decryption privileges). In a recent study, the authors proposed CP-ABE-based storage model for safe data access and storage for Internet of Things applications. [21] The proposed model adds an attribute authority management (AAM) module to cloud, that serves as an agent and offers a user-friendly access control while drastically reducing the storage overhead of public keys. This approach is completely different from our proposed architecture and despite the claimed benefits, CP-ABE approaches are yet susceptible to attacks such as chosen plain text [22] and dictionary attacks [23]. In many earlier techniques, it was impossible for the cloud provider to confirm whether a downloader could decrypt the data. Due to this, everyone with access to the cloud storage should have these data available. Another way to provide secure access control and avoid re-encryption of data, is to deploy public key infrastructure based solutions [24]. However, despite the ease of key distribution and access control public key based solutions provide, they also bring along several security weaknesses. In general, searching is much more difficult if users routinely encrypt the data. This conflict is settled using public-key encryption with keyword search (PEKS). On the other hand, because keywords have low entropy, it is susceptible to keyword guessing attacks (KGA) [25].

3 Proposed System

An integrated key escrow and data decryption scheme in which Data Encryption Keys (DEK) encrypted by per-client Key Encryption Keys (KEK) are published on a public blockchain. The blockchain stores the encrypted DEKs as assets whose ownership can be transferred to other members of the blockchain. A Crypto Proxy Agent (CPA) deployed close to a Clients network, upon authenticating various claims by the Client, re-constructs the Client-specific KEK to decrypt the encrypted DEK. Once in possession of the DEK, the CPA downloads its corresponding encrypted data from an untrusted cloud storage service, decrypts and transmits it to the Client. The proposed approach has following unique selling points:

1. As only one DEK is created and used, the data does not have to be decrypted and re-encrypted again each time it is needed to be shared with a new Client. Only a new client specific KEK is generated.
2. The Clients never get to know the DEK, so the same DEK can be used repeatedly to encrypt different data sets. Using a unique DEK for different data sets can create a large number of encryption keys, which can be difficult to manage in a Smart Home IoT environment.
3. As the IoT EG is usually not a high-powered computer/device, a create a DEK once and encrypt data once model is more suitable for it.

4. Unlike standard key escrow solutions, the Key Encryption Key or the Data Recovery Field is never attached to the encrypted data. So even if the encrypted data is subject to a ransomware attack, the keys are not lost.
5. The data owner and data consumer (Client) do not have to communicate with each other interactively (offline)
6. Blockchains allow a client to transfer the ownership of a transaction (in this case the encrypted KEKs) to other clients. Therefore, should the client want to share its KEK with another entity, it does not have to retrieve the KEK from the blockchain and send it securely to other entity using a separate protocol.

3.1 Architecture design

The identities of the Clients are managed through an external Distributed Identity Service (DIDS), which is not the scope of proposed idea, not part of proposed solution and is shown in red in the main diagram. However, the DIDS process relevant to this invention should work as below:

- A member Client (that has subscribed to the DIDS) can request the DIDS to generate temporary private/public key pairs y, g^y for it.
- The DIDS signs and publishes the latest version of its clients temporary public keys $(g^{y_1}, g^{y_2}, \ldots, g^{y_n}$ for n Clients) on the public blockchain.

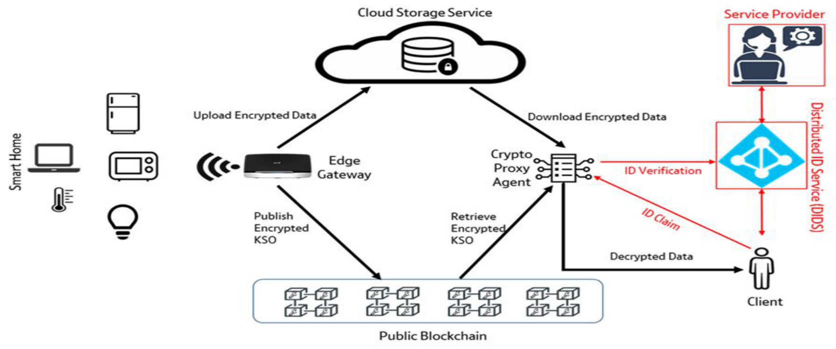


Fig. 1 The Transparent Encryption system for Edge devices in IoT environments using offline key mechanism of public blockchains.

The detailed flow description of the main process is as follows:

1. The Smart Home IoT sensors are connected to the Edge Gateway (EG); all sensors data passes through the EG.
2. The EG does the following:

a. The EG encrypts the sensor data of interest with a Data Encryption Key (DEK), i.e., $AES(DEK, M) = C$.
b. EG sends the encrypted data C to the third-party cloud storage service and receives a URI (Uniform Resource Identifier) of the uploaded encrypted data as a result of this operation.
c. A Client-specific Key Encryption Key (KEK) is generated by the EG in the following way:
   i. The EG creates a temporary private/public key pair x,g^x and publishes the temporary public key g^x on the public blockchain.
   ii. The EG queries the blockchain to get the latest public keys of all member Clients $(g^{y_1}, g^{y_2}, ..., g^{y_n})$.
   iii. The EG then generates the Client KEKs as:
   $KEK_1 = Hash(EG_{ID}||Client_{id}||g^{y_1 x})$, where EG_{ID} and $Client_{id}$ are the identities of the EG and a particular Client, and $g^{y_1 x}$ is computed by the EG using its private key x and a Clients public key g^{y_1}.
d. The EG wraps the DEK using the Client-specific KEK to generate a Key Security Object (KSO), i.e.,$AES(KEK, DEK) = KSO$
e. The EG signs and publishes the $Hash(URI)$,KSO key-value pair on the public blockchain.

3. The Client accesses the decrypted data through a Crypto Proxy Agent (CPA) according to following steps:

a. The Client verifies their identity to the CPA by sharing their Verifiable Claim with the CPA (the claim contains information like the Clients identifier/certificate, ID of the EG that is the source of the encrypted data and URI of the encrypted data stored on the cloud service that the Client wants to access).
b. The CPA communicates with the DIDS to verify the Clients claim and receives the Clients temporary private key y_1 as part of the successful verification reply.
c. After the Clients successful verification from the DIDS, the CPA retrieves
   i. The encrypted data from the cloud storage service, using the URI obtained in step a.
   ii. The EGs public key g^x from the public blockchain, using the ID of the EG obtained in step a.
   iii. The KSO associated with the hash of the URI from the public blockchain, by using the URI obtained in step a.
d. Using the Clients public key and identifier information from the claim, the CPA constructs the KEK, i.e., $KEK_1 = Hash(EG_{ID}||Client_{id}||g^{xy_1})$
e. The CPA decrypts the KSO using the KEK constructed in the previous step, and obtains the DEK as a result, i.e., $DEK = AES^{-1}(KEK_1, KSO)$
f. The CPA finally decrypts the encrypted data needed by the Client using the DEK and sends it to them, i.e., $M = AES^{-1}(DEK, C)$

4 Proof of concept implementation

The proposed idea has been implemented using MATRIX edge, which is an Edge gateway built on top of an opensource platform known as EdgeX foundry [26]. MATRIX has been designed by British Telecom Innovation Ireland Centre (BTIIC) team and is usual choice for testing or experimentation. Figure 2 presents the sequence flow of the process of encryption and storage of the encrypted data. As shown in Figure 1, the Edge gateway is responsible for encrypting all the data generated by the IoT devices, which in our case is Matric Edge. The Matrix Edge generates a unique key for the encryption. This key is generated for each device associated with a predefined encryption session/window.The Matrix Edge then forwards the encrypted data to a third-party storage service, and the key to the DLT storage. In the next step, the MATRIX Edge renews/generates a new key for each encryption session/window and use it for the encryption.

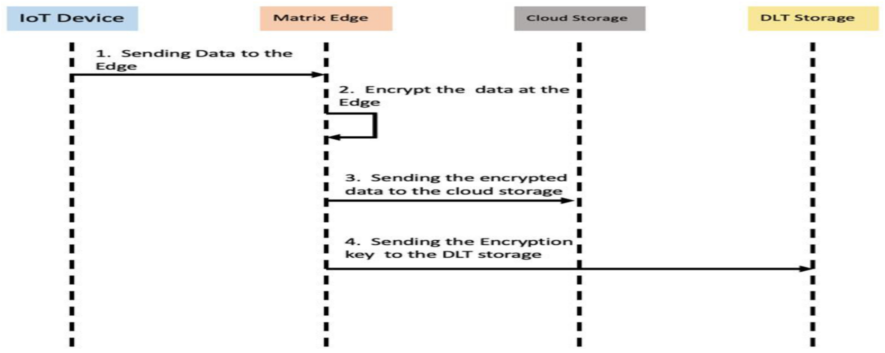

IoT Device
Matrix Edge
Cloud Storage
DLT Storage
1. Sending Data to the Edge
2. Encrypt the data at the Edge
3. Sending the encrypted data to the cloud storage
4. Sending the Encryption key to the DLT storage

Fig. 2 Sequence diagram of the encryption process.

The process of retrieving the encrypted data is presented in Figure 3. The user requests specific data from the Crypto Proxy by providing the ID of the device and the session/window of the encrypted data. The Crypto Proxy will retrieve the specific data and key to decrypt the data and display it to the user. Figure 4 shows the proposed system in operation in which the user can choose one of the sensors attached to the MATRIX edge for the encryption, and the Distributed Ledger Technology (DLT) address (to store the encrypted key). By default, we choose the Amazon AWS storage services to store all encrypted data generated by the sensors. Figure 5 shows the interface for retrieving data, in which the Crypto Proxy will consider the user input to retrieve the data from the different components of the system and display the requested data to the user.

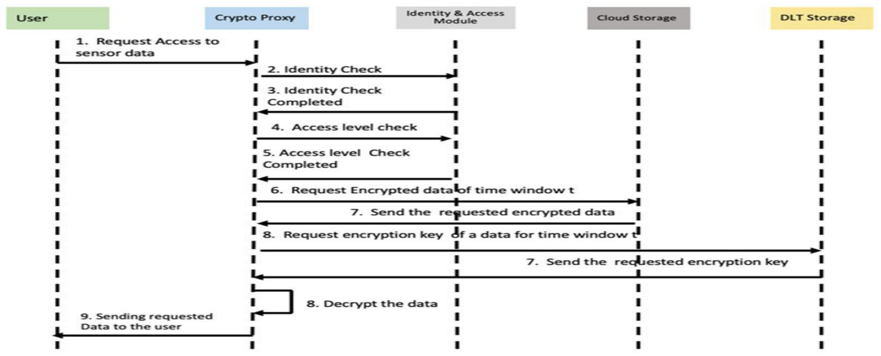
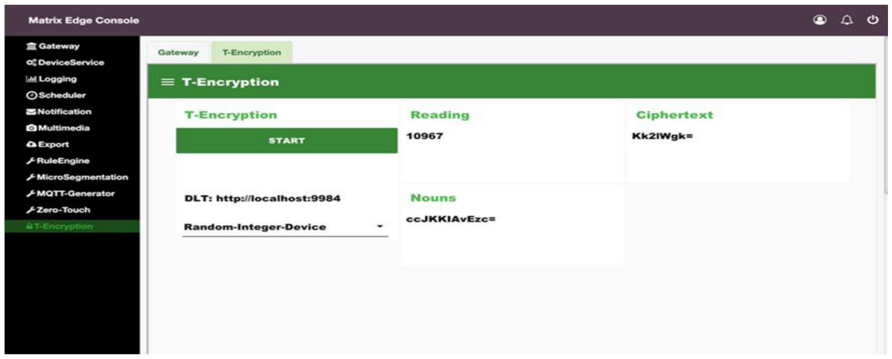

User
Crypto Proxy
Identity & Access Module
Cloud Storage
DLT Storage
1. Request Access to sensor data
2. Identity Check
3. Identity Check Completed
4. Access level check
5. Access level Check Completed
6. Request Encrypted data of time window t
7. Send the requested encrypted data
8. Request encryption key of a data for time window t
7. Send the requested encryption key
8. Decrypt the data
9. Sending requested Data to the user

Fig. 3 Sequence diagram of retrieving the data.



T-Encryption

Reading
10967

Ciphertext
Kk2IWgk=

DLT: http://localhost:9984

Random-Integer-Device

Nouns
ccJKKIAvEzc=

Fig. 4 MATRIX Edge Transparent Encryption Module.

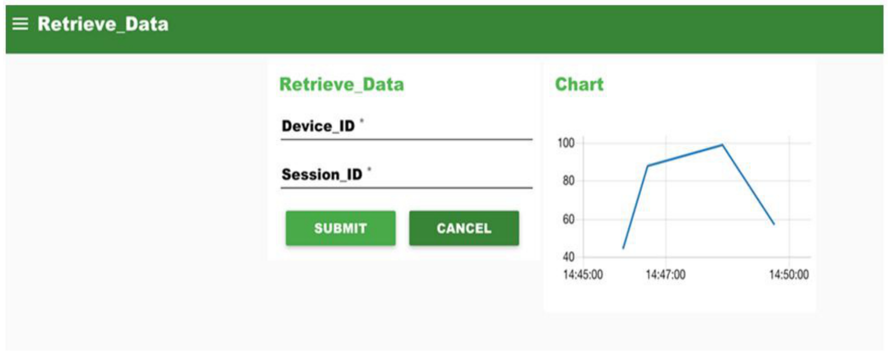


Fig. 5 Retrieving the Data Module.

4.1 Design and Experiments

The experiments were performed by setting up Matrix Edge on a desktop server with the following specifications: Intel(R) Core (TM) i5-8250U CPU @1.80 GHz, 16 GB RAM, 64-bit operating system (Ubuntu 20.04.4 LTS Focal Fossa).
In the proposed system we used ChaCha20 stream cipher [27] as the encryption algorithm and BigChainDB [28] as the DLT solution. Table 1 lists all the programming languages, libraries and technologies used in the implementation.

Table 1 Libraries and tools used in the implementation.

Programming Languages	Python 2.7, Python 3.5
Programming Tools	Node-red version 2.2 [29]
DLT	BigChainDB
Crypto Libraries	PyCryptodome version 3.14 [30]
Libraries	bigchaindb-driver
Cloud storage toolkits /APIs	Amazon Elastic Compute Cloud / boto3

Figure 4 shows the screenshot of Matrix Edge console where the data is encrypted and the encryption keys are being pushed on to the distributed ledger hosted as local host port number 9984. Figure 5 shows the GUI for the receiving end, where the device ID and session ID are required to fetch relevant data from the cloud storage.

5 Summary & Future Work

This paper presents a solution to key management problem for Edge based IoT networks that use third party cloud storage. Conventionally, the encryption keys are stored with the data or handed over to the client for accessing data. However, the in this paper a fool proof architecture based on zero trust security principles that provides for transparent encryption of IoT data, without requiring encryption keys to be shared with third party cloud or the client for accessing the data.

In the proposed model, an integrated key escrow and data decryption scheme has been presented where, Data Encryption Keys (DEK) encrypted by per-client Key Encryption Keys (KEK) are published on a public blockchain. The blockchain stores the encrypted DEKs as assets whose ownership can be transferred to other members of the blockchain. A Crypto Proxy Agent (CPA) deployed close to a Clients network, upon authenticating various claims by the Client, re-constructs the Client-specific KEK to decrypt the encrypted DEK. Once in possession of the DEK, the CPA downloads its corresponding encrypted data from an untrusted cloud storage service, decrypts and transmits it to the Client.

The proof of concept implementation of the idea has been done using MATRIX Edge platform, which is an Edge gateway designed by British Telecom Innovation Ireland Centre. This framework has been filed as a patent by British Telecom. In future the proposed idea would be extended to inclusion of smart contracts in blockchains to automate the process of authentication and asset transfer.

Acknowledgements

This research work was conducted under the BT Ireland Innovation Centre (BTIIC) project and was funded by Invest Northern Ireland and BT.

References

1. A. Sathesh, D. S. Smys, A survey on internet of things (iot) based smart systems, Journal of IoT in Social, Mobile, Analytics, and Cloud 2 (4) (2020) 181–189. doi:10.36548/jismac.2020.4.001.
2. Cisco, Unlocking the potential of the internet of things, https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf (2022).
3. J. Manyika, M. Chui, P. Bisson, J. Woetzel, R. Dobbs, J. Bughin, D. Aharon, Cisco annual internet report (20182023), https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world (Mar 2015).
4. H. Noura, R. Couturier, C. Pham, A. Chehab, Lightweight stream cipher scheme for resource-constrained iot devices, in: 2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2019, pp. 1–8. doi:10.1109/WiMOB.2019.8923144.
5. L. Garber, Melissa virus creates a new type of threat, Computer 32 (06) (1999) 16–19. doi:10.1109/MC.1999.769438.
6. A. Riahi Sfar, E. Natalizio, Y. Challal, Z. Chtourou, A roadmap for security challenges in the internet of things, Digital Communications and Networks 4 (2) (2018) 118–137. doi:https://doi.org/10.1016/j.dcan.2017.04.003.
URL https://www.sciencedirect.com/science/article/pii/S2352864817300214
7. O. Lassila, J. Hendler, Embracing ”web 3.0”, IEEE Internet Computing 11 (3) (2007) 90–93. doi:10.1109/MIC.2007.52.
8. A. Sajjad, M. Abu-Tair, U. Zia, Transparent iot edge encryption using offline key exchange over public blockchains (2022).
URL https://shorturl.at/hrKN2
9. S. Tayeb, S. Latifi, Y. Kim, A survey on iot communication and computation frameworks: An industrial perspective, in: 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), 2017, pp. 1–6. doi:10.1109/CCWC.2017.7868354.
10. S. Rose, O. Borchert, S. Mitchell, S. Connelly, Zero trust architecture (2020-08-10 04:08:00 2020). doi:https://doi.org/10.6028/NIST.SP.800-207.
URL https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=930420
11. A. S. Voundi Koe, Y. Lin, Offline privacy preserving proxy re-encryption in mobile cloud computing, Pervasive and Mobile Computing 59 (2019) 101081. doi:https://doi.org/10.1016/j.pmcj.2019.101081.
URL https://www.sciencedirect.com/science/article/pii/S1574119219301488

12. F. Alsolami, T. E. Boult, Cloudstash: Using secret-sharing scheme to secure data, not keys, in multi-clouds, in: 2014 11th International Conference on Information Technology: New Generations, 2014, pp. 315–320. doi:10.1109/ITNG.2014.119.
13. J. Hendler, Web 3.0 emerging, Computer 42 (01) (2009) 111–113. doi:10.1109/MC.2009.30.
14. M. Singh, A. Singh, S. Kim, Blockchain: A game changer for securing iot data, in: 2018 IEEE 4th World Forum on Internet of Things (WF-IoT), 2018, pp. 51–55. doi:10.1109/WF-IoT.2018.8355182.
15. M. Shah, M. Z. Shaikh, V. Mishra, G. Tuscano, Decentralized cloud storage using blockchain, 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184) (2020) 384–389.
16. M. Jemel, A. Serhrouchni, Decentralized access control mechanism with temporal dimension based on blockchain, in: 2017 IEEE 14th International Conference on e-Business Engineering (ICEBE), 2017, pp. 177–182. doi:10.1109/ICEBE.2017.35.
17. J. Wang, A. Sundaresan, V. B. Kaza, D. Calia, Transparent proxy of encrypted sessions, https://image-ppubs.uspto.gov/dirsearch-public/print/downloadPdf/8214635 (Mar 2012).
18. P. Zeng, K.-K. R. Choo, A new kind of conditional proxy re-encryption for secure cloud storage, IEEE Access 6 (2018) 70017–70024. doi:10.1109/ACCESS.2018.2879479.
19. S. Wang, X. Wang, Y. Zhang, A secure cloud storage framework with access control based on blockchain, IEEE Access 7 (2019) 112713–112725. doi:10.1109/ACCESS.2019.2929205.
20. J. Ning, Z. Cao, X. Dong, K. Liang, L. Wei, K.-K. R. Choo, Cryptcloud^{+}+: Secure and expressive data access control for cloud storage, IEEE Transactions on Services Computing 14 (1) (2021) 111–124. doi:10.1109/TSC.2018.2791538.
21. S. Xiong, Q. Ni, L. Wang, Q. Wang, Sem-acsit: Secure and efficient multiauthority access control for iot cloud storage, IEEE Internet of Things Journal 7 (4) (2020) 2914–2927. doi:10.1109/JIOT.2020.2963899.
22. H. Cui, R. H. Deng, J. Lai, X. Yi, S. Nepal, An efficient and expressive ciphertext-policy attribute-based encryption scheme with partially hidden access structures, revisited, Computer Networks 133 (2018) 157–165. doi:https://doi.org/10.1016/j.comnet.2018.01.034.
URL https://www.sciencedirect.com/science/article/pii/S138912861830046X
23. L. Li, T. Gu, L. Chang, Z. Xu, Y. Liu, J. Qian, A ciphertext-policy attribute-based encryption based on an ordered binary decision diagram, IEEE Access 5 (2017) 1137–1145. doi:10.1109/ACCESS.2017.2651904.
24. I. Sukhodolskiy, S. Zapechnikov, A blockchain-based access control system for cloud storage, in: 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2018, pp. 1575–1578. doi:10.1109/EIConRus.2018.8317400.
25. S. K. Nayak, S. Tripathy, Seps: Efficient public-key based secure search over outsourced data, Journal of Information Security and Applications 61 (2021) 102932. doi:https://doi.org/10.1016/j.jisa.2021.102932.
URL https://www.sciencedirect.com/science/article/pii/S2214212621001514
26. EdgeXFoundry, Edgex: Open source edge platform (Jan 2019).
URL https://www.edgexfoundry.org
27. Y. Nir, A. Langley, ChaCha20 and Poly1305 for IETF Protocols, RFC 7539 (May 2015). doi:10.17487/RFC7539.
URL https://www.rfc-editor.org/info/rfc7539
28. B. GmbH, Bigchaindb 2.0 the blockchain database (2018).
URL https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf
29. O. F. . Contributors, Node-red (2016).
URL https://nodered.org
30. PyPi, Pycryptodome (2022).
URL https://pypi.org/project/pycryptodome/