symmetry

MDPI

Article

An Approach for Blockchain and Symmetric Keys Broadcast Encryption Based Access Control in IoT

Miodrag J. Mihaljević [1,2,*], Milica Knežević [2], Dragan Urošević [2], Lianhai Wang [1] and Shujiang Xu [1]

[1] The Shandong Provincial Key Laboratory of Computer Networks, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China
[2] Mathematical Institute, The Serbian Academy of Sciences and Arts, 11000 Belgrade, Serbia
* Correspondence: miodragm@turing.mi.sanu.ac.rs; Tel.: +381-65-2663-257

Abstract: This paper considers the problem of data access control when the subscribers are IoT devices with initialization that cannot be updated during the entire life cycle. A generic framework and a particular instance for conditional data access control within IoT are proposed. The generic framework is based on the employment of a dedicated secret key-based broadcast encryption scheme where encrypted credentials for conditional data access is available in the blockchain and encrypted data subject to conditional access are available in an off-chain source of streaming data. Reduction of the keys management overhead in comparison with a straightforward decryption keys delivery is experimentally illustrated. An instance of the proposed framework built over the Ethereum blockchain platform is developed and experimentally evaluated.

Keywords: IoT; data access control; broadcast encryption; blockchain; Ethereum platform

Citation: Mihaljević, M.J.; Knežević, M.; Urošević, D.; Wang, L.; Xu, S. An Approach for Blockchain and Symmetric Keys Broadcast Encryption Based Access Control in IoT. *Symmetry* **2023**, *15*, 299. https://doi.org/10.3390/sym15020299

Academic Editor: Christos Volos

Received: 19 December 2022
Revised: 16 January 2023
Accepted: 17 January 2023
Published: 21 January 2023

1. Introduction

Authentication and authorization of Internet of Things (IoT) devices for conditional data access control are two critical issues. An important conditional data access scenario consists of the following: A device is initialized at the very beginning of its life cycle and should work without updates during its entire life cycle.

One version of Broadcast encryption (BE) has been developed to provide conditional data access in the scenario where an entity should be provided with certain credentials during the initialization phase and these cannot be updated during the entire life cycle. Illustrative initial applications were the following ones: a DVD player that provides conditional access to many different DVDs, or a satellite TV broadcasting system that provides conditional access to TV programs based on "pay-per-view" concept, see [1], for example. The traditional paradigm of conditional data access control employing BE is the following one: (i) encrypt data with a session secret key, (ii) append to the encrypted data as a header encrypted version of the current session key encrypted with a number of different keys so that each legitimate user can decrypt the session key and the data, (iii) make the header and the encrypted data publicly available. In particular, BE provides an opportunity for dynamic updates of the access privileges and a high reduction of the header overhead when large clusters of users should obtain the access rights or should be revoked from the set of eligible users.

On the other hand, blockchain technology provides a widely distributed medium, the ledger, that supports the distribution of different publicly available data. Blockchain technology is a highly important topic with a lot of everyday growing applications (see [2], for example). An important element of blockchain technology is so-called smart contracts (see [3], for example).

Recently, a number of advances in BE employment and combinations of BE and blockchain technology have been reported. In [4], a hierarchical identity based BE approach and blockchain have been employed for developing a medical information service platform.

A blockchain and BE have been employed for developing secure clinical data sharing as reported in [5]. An approach for the joint employment of BE and blockchain for telecare medical systems is shown in [6]. A key management scheme called dynamic contributory BE for secure channel establishment in fog computing is proposed in [7]. Securing users' data from untrusted cloud service providers employing a BE-based keys management system is considered in [8]. Employment of identity-based BE for data access control in a cloud-based storage service is reported in [9]. For secure key management and establishing secure group channels in the IoT environment, paper [10] proposes an approach based on joint employment of BE and blockchain. In all the above-considered references, the public key-based BEs have been employed.

1.1. Motivation for the Work

There is a conceptual similarity between certain traditional BE-based scenarios for conditional data access control and the ones within IoT. In both scenarios, the following restriction exists, the initial setting of a device cannot be updated during the device life cycle. Consequently, it is interesting to consider the employment of BE paradigm within IoT and the employment of the blockchain as a public media for communications that provide conditional data access. Accordingly, this paper considers a scenario where a huge number of IoT devices with initialization that cannot be updated can obtain conditional data access to streaming data employing blockchain with smart contracts. Oppositely to mainly employed public key-based BE, in order to reduce the computational overhead inherent to public key-based systems, this paper considers secret key-based BE.

1.2. Summary of the Results

This paper proposes a generic framework and a particular instance for conditional data access control within IoT. The generic framework is based on the employment of a dedicated secret key-based BE scheme where encrypted credentials for the conditional data access control are available in the blockchain, and encrypted data subject to the conditional access are available in an off-chain source of streaming data. The blockchain platform enables interactions between data users and data providers and it yields the data access control information. Reduction of the keys management overhead when BE is employed in comparison with a straightforward delivery of the decryption keys is experimentally illustrated. An instance of the proposed framework built over the Ethereum blockchain is developed and evaluated. The following procedures are proposed: (i) system initialization stage that consists of the data provider and the data user initialization; (ii) the data publication stage; and (iii) the data access control procedure that includes adding new data users and removing data users after the subscription expiration. Illustrative smart contracts are given for establishing the time-varying subscription status of devices that require streaming data access.

1.3. Organization of the Paper

A summary background on BE and blockchain technology is given in Section 2, and Section 2.2 provides a discussion on the related work. The framework for conditional data access employing dedicated BE and blockchain is proposed in Section 4 together with an experimental evaluation of the complexity. An instance of the proposed framework is reported in Section 5 together with implementation using the Ethereum platform and illustrative experimental evaluation. A discussion is given in Section 6.

2. Background

2.1. Blockchain

Blockchain technology provides: (i) an immutable distributed record, called a ledger, of updates or transactions in the corresponding network; and (ii) verified updates of the distributed ledger without the third trusted party employing the consensus protocol. The main entities of a blockchain system belong to the following two classes: entities that

support the operations of the system and users of the system. The entities that operate the system run the consensus protocol. If anybody can participate in the consensus protocol, the blockchain is permissionless or public. In permissioned blockchain systems, only selected entities can participate in the consensus protocol. One of the most widespread classes of consensus protocols is the so-called Proof-of-Work (PoW) based consensus protocols that require solving a cryptographic puzzle. redSome blockchain platforms, such as Ethereum, are characterized by programmability potential which comes in form of smart contracts, which are programs executed on the blockchain.

A simplified architecture of a public blockchain system based on PoW consensus protocol is displayed in Figure 1 where the following two important issues are highlighted: (i) the possibility of verified updates of the ledger employing a consensus protocol, and (ii) smart contracts.

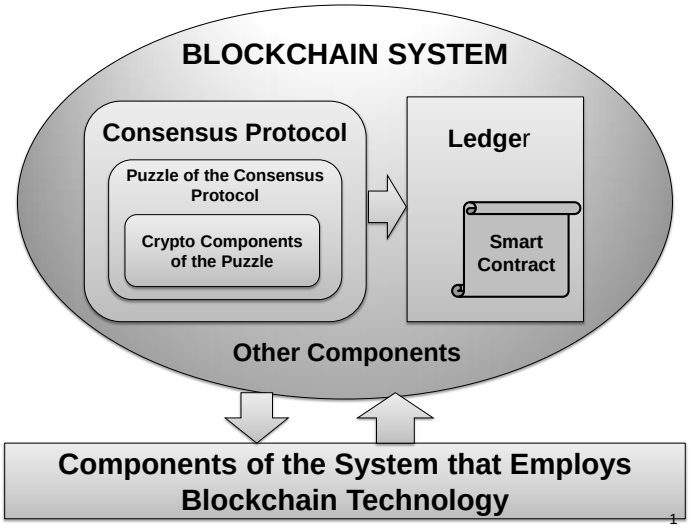

Figure 1. A simplified scheme of a public blockchain system with a consensus protocol based on PoW.

A smart contract is an executable program stored in the ledger. Smart contracts provide decentralized automation by allowing the participating nodes in a Blockchain to perform transactions without the need for a third-party entity. The most well-known blockchain platform that provides options for the creation and execution of a smart contract is Ethereum which supports the implementation of smart contracts using a programming language such as Solidity.

2.2. Broadcast Encryption

Broadcast Encryption (BE) is a technique that provides a framework for selective access to encrypted data. BE is used to distribute a certain message, in a particular scenario the key used to encrypt the data, to users in a secure way, so that only a certain subset of users, namely those who have access rights, can decrypt that message and recover the source data. There are two main classes of BE schemes: One employs secret (symmetric) key cryptography and the other is based on public (asymmetric) key cryptography. A comparison between BE and public key cryptography is reported in [11]. Secret key-based BE schemes could be with a state that is subject to updates or stateless. This paper deals with stateless BE schemes and thus the main characteristics of this BE approach are summarized here. The traditional paradigm of secret keys stateless BE is based on the following: There are two classes of secret keys: a set of static keys called the key encryption keys (KEKs) and time-varying session encryption keys (SEKs). KEKs are used for the encryption of a session key before its broadcast. A stateless BE system consists of the following: (i) BE management center; (ii) KEKs infrastructure; (iii) users; (iv) communications protocol between the management center and users. BE management center designs KEKs structure, associates

users to this structure, and provide each of the users with a subset of KEKs. During the initialization phase, BE management center establishes the infrastructure of KEKs and distributes a set of KEKs to each of the system users. Let $E_X\{Y\}$ denote encryption of Y employing a secret key X and an encryption algorithm $E_X\{\cdot\}$. In order to control access to certain data D, BE management center encrypts D with a session key SEK, encrypts SEK with a number of KEKs so that the legitimate users could perform decryption of SEK and consequently D. Finally, BE management center open to the public a sequence consisting of a prefix with ciphertexts of SEK encrypted with a number of different KEKs and index names of the employed KEKs concatenated with the data encrypted employing the considered SEK, as follows:

$$\{(ID_{KEK_i}, E_{KEK_i}(SEK))\}_{KEK_i \in \Omega} || E_{SEK}(D)$$

where ID_{KEK_i} is an index name of KEK_i and Ω is a subset of KEKs such that all legitimate users can decrypt SEK employed for encryption of D.

An illustration of KEKs infrastructure and the data access control is given in Figure 2. The considered KEKs infrastructure is designed for 32 users, and it is a binary balanced three of depth equal to 6, where 63 KEKs are associated with 63 nodes of the tree. The users are associated with the leaves of the tree. Employing this infrastructure, in the initialization phase, each user is provided with the keys in the tree nodes on the path from the root to the corresponding leaf.

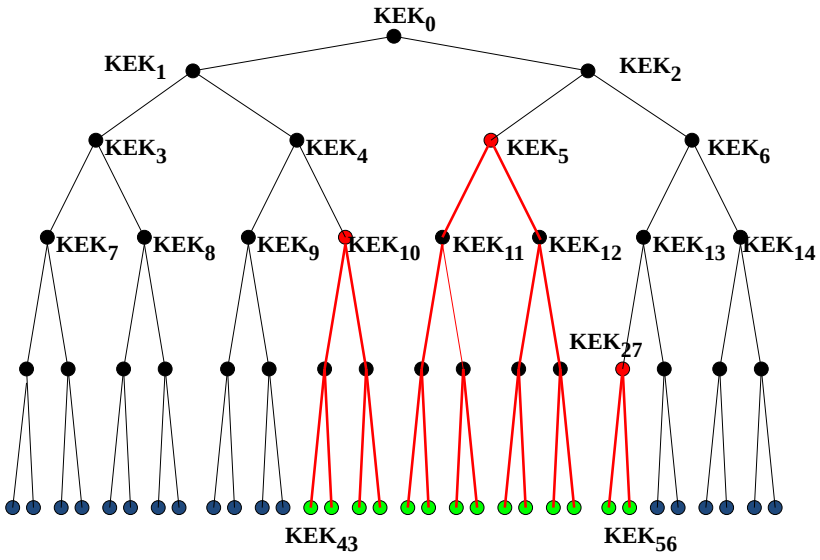

Figure 2. Basic paradigm of the broadcast encryption-based data access control.

In the example displayed in Figure 2, the goal is to open data access to the users in green. A straightforward approach is to provide each of the green users with SEK encrypted with KEKs from KEK_{43} to KEK_{56}, i.e., to generate the prefix of $E_{SEK}(D)$ consisting of 14 different ciphertexts of SEK. A much more efficient approach is to encrypt SEK just with KEK_5, KEK_{10} and KEK_{27}, i.e., to open to public the following:

$$(ID_{KEK_5}, E_{KEK_5}(SEK)) || (ID_{KEK10}, E_{KEK_{10}}(SEK)) || (ID_{KEK_{27}}, E_{KEK27}(SEK)) || E_{SEK}(D)$$

that requires a prefix of length 3 instead the straightforward approach that requires a prefix of length 14.

Regarding the consideration of secret key stateless BE versus the state-based secret key BE or public key-based BE, in particular note the following. In comparison with BE which requires the state updates at receivers, the stateless one scheme does not require any communication between the BE center and BE receivers, and in a number of scenarios, this

is an important benefit. On the other hand, also please note the following. Employment of public key cryptography requires Public Key Infrastructure (PKI), and PKI brings the problem of creating and maintaining certificates, which includes the revocation of certificates, their storage, distribution, as well as calculation costs for certificate verification. With BE, the addition of a new user, as well as the termination of user access rights, is very simple. In particular, the employment of secret key-based BE, in comparison with the public key-based ones, provides a reduction of the encryption/decryption computational costs.

Furthermore, note the following. In a public key broadcast encryption (PKBE) system, data access is controlled in a similar manner as in BE based on secret key cryptography. In PKBE, the data are encrypted employing a symmetric key encryption technique (e.g., AES block cipher) Then, SEK is encrypted with an asymmetric (i.e., public key) encryption technique, so that only the legitimate receivers can obtain the symmetric key SEK and use it for the symmetric decryption of the encrypted data. So, PKBE combines symmetric and asymmetric encryption techniques, and the employment of asymmetric cryptography requires the existence of the related public key infrastructure (PKI). In particular note the following: (i) An IoT device should access the blockchain only to read its credentials that provide access to the data, i.e., the relevant pair consisting of the device's ID and the session key ciphertext; (ii) the smart contract that provides access of the IoT device could be executed by a server that controls a group of IoT devices. Accordingly, an IoT device does not need to participate in the PKI, and in certain scenarios, it is an important restriction because the involved IoT devices are not strong enough to support PKI-related overheads including decryption of public key encryption.

3. Related Work

This section points out a number of previously reported illustrative results related to this paper and organized within the following topics: (i) BE and data access control; (ii) blockchain and data access control; (iii) joint blockchain and BE approach for data access control.

Broadcast Encryption and Data Access Control. In [9], an efficient anonymous identity-based broadcast encryption construction is proposed and its application to the data access control mechanism in cloud storage service. The paper [12] proposes a broadcast encryption scheme for tiny IoT equipment (BESTIE) suitable for large-scale IoT systems, that provides a reduction of the employed key size. The proposed BESTIE is a public key broadcast encryption scheme with the combinatorial subset difference (CSD) representation. The employed CSD is reported in [13] as a new subset representation for the broadcast encryption and it generates minimal subsets which lead to the minimal header size when applied to the broadcast encryption. Furthermore, [13] yields the CSD-based public key broadcast encryption, which can be efficiently applied to the IP multicast in IoT systems. Identity-based broadcast encryption (IBBE) is an important method to provide security and privacy protection for cloud storage services, but the side-channel attacks may lead to the disclosure of the key information of the cryptographic system, and [14] reports an IBBE with leakage resilience by state partition (LR-SP-IBBE). The paper [15] presents anonymous identity-based broadcast encryption against continual side-channel attacks in the state partition model (CLR-SS-AIBBE) where the private key is continuously updated through state division providing that the scheme can resist the continual leakage about the private key. Two new decentralized BE schemes are reported in [16], where decentralization eliminates the concern of having a single point of failure as the central authority could be attacked or become malicious, and they provide a reduction of the key size and the decryption time, as well. In [8], a cryptographic approach is reported that uses Identity-based Broadcast Encryption (IBBE) for managing the keys of the symmetric key algorithm (BDNA) by encrypting them with the particular version of IBBE. APIB-BPRE [17] supports flexible cloud data sharing which overcomes the issue of Identity-Based Broadcast Encryption applied in the outsourced data sharing environments, i.e., the problem of autonomous path

multi-hop. The authors show that APIB-BPRE can provide much better fine-grained access control to delegation broadcast receiver sets in the cloud.

Blockchain and Data Access Control within IoT. In [18], an architecture for encrypted data storing and sharing is proposed by combining proxy re-encryption with blockchain technology assuming that the data owners generate and should manage a large amount of IoT data. A blockchain-aided approach for securing remote sensing data employing Ethereum smart contracts are proposed in [19] which provides a system to share and retrieve data, as well as to use blockchain algorithms based on smart contracts to track space transactions and communications in a secure, verifiable, and transparent manner. In [20], a blockchain-based system for IoT data sharing is proposed that integrates ciphertext-policy attribute-based encryption and fully homomorphic encryption to realize secure IoT data sharing. A digital media subscription mechanism based on the Hyperledger blockchain architecture combined with proxy re-encryption is reported in [21] where symmetric and asymmetric cryptography and smart contracts are employed to design the subscription protocol so that when the licensee violates the agreement with the creator, the creator can revoke the access rights to the digital media. A decentralized, blockchain-based solution for hybrid access control in Industrial Internet-of-Thins (IIoT) is proposed in [22]. It relies on a private blockchain and smart contracts to provide transparency, reliability, and robustness in the access control mechanism for IIoT. In [23], a blockchain-based decentralized distributed storage and sharing scheme for IoT data is proposed. The solution provides end-to-end encryption and fine-grained access control using Attribute-Based Access Control (A-BAC). It employs the Ethereum blockchain with smart contracts as an auditable access control layer. FBASHI [24] framework aims at providing AAA (Authentication, Authorization and Audit Logs) services for healthcare IoT environments using fuzzy logic and Hyperledger Fabric blockchain. It is shown to be practical and effective for IoT-based distributed architectures. VO-PH-MAABE [25] is a blockchain-based multiple authority attribute-based encryption scheme for access control in Internet-of-Medical-Things (IOMT) environment. The scheme relies on four Ethereum blockchain smart contracts to achieve cross-domain distributed management of attributes and cross-domain computation of different authorities. The provided analysis of VO-PH-MAABE shows it reduces the cost of cross-domain computation and eliminates the single-point bottleneck problem of traditional Ciphertext-Policy Attribute-Based Encryption (CP-ABE) schemes. SDSM [26] is an access control scheme for IoT-based supply chains, which combines blockchain and ciphertext-based attribute cryptography. MDS^2-C^3PF [27] addresses the problem of asymmetric access control rights between doctors and patients for medical data. A method for fusing attribute-based access policies is proposed to generate an access control policy created by both medical doctors and patients. In addition, a cloud-chain cooperation retrieval method is proposed in order to achieve medical data retrieval efficiency and to be able to detect malicious feedback from cloud servers. Retrieval efficiency is improved by designing the off-chain search structure and performing an initial search on the cloud server with a secondary search on the blockchain.

Blockchain and Broadcast Encryption Approach for Data Access Control. One approach for the joint employment of blockchain technology and broadcast encryption for developing an efficient and secure information service platform has been reported in [4]. In the proposed framework, medical data are stored on distributed in the cloud after encryption, identity-based broadcast encryption is used, and an incentive mechanism is designed to encourage customers and miners to maintain the platform. In [5], broadcast encryption, blockchain and smart contract are employed to design a system for medical data sharing and integration where a patient's medical records are divided into multiple parts with different sensitivities according to personal privacy requirements. In [6], blockchain technology and broadcast encryption are employed to address the issue of secure information sharing in multiple medical servers' scenarios. In [10], a blockchain-based key management scheme is proposed for managing secure keys and establishing secure group channels in fog-based IoT systems. The proposed key management scheme uses

an improved dynamic contributory broadcast encryption (DConBE)-based key management scheme, a new PoW blockchain consensus mechanism as the key building blocks, and it also achieves the data recoverability property by using blockchain. HSE-BI [28] is a blockchain-based hierarchical searchable encryption scheme focusing on fine-grained access control challenge of multi-user searchable encryption. A hierarchical search index structure based on stepwise hierarchical key derivation is designed, and it is outsourced to the blockchain network in order to achieve search reliability. In addition, the authors propose a hierarchical authorization mechanism based on broadcast encryption to achieve fine-grained search permission granting and revoking and prevent collusion of malicious servers and users. In [29], a decentralized subscription-push service scheme inspired by broadcast encryption and multicast encryption is presented. A new encryption algorithm is designed to manage the permissions of IoT devices together with smart contracts and to protect the confidentiality of push messages. Employed blockchain smart contract enables verification of the validity of subscription services through subscription tags stored in the smart contract.

4. Framework for Data Access Control Employing Blockchain and Broadcast Encryption

The following scenario is considered for developing a system for conditional data access. Certain IoT devices submit streaming data to a provider for further "pay-per-view" distribution. The provider communicates with the service users and controls their access to the data in a highly dynamic way.

4.1. System Architecture

The main underlying ideas for the design of the system for "pay-per-view" conditional data access include the following: (i) Employment of a blockchain and smart contracts between the provider and the users for time dynamic data access control; (ii) Employment of secret key-based BE for access to the encrypted data; (iii) Employment of time segment-by-segment streaming data encryption and off-chain delivery of the encrypted data.

The main components of the system are:

- The management center that establishes and operates the system;
- A blockchain with smart contracts for subscription and revocation of the data users supporting highly dynamic data access control based on BE;
- An off-blockchain source of encrypted data that is an access point for delivering encrypted data;
- A pool of potential data users that are IoT devices.

Consequently, a basic architecture of the proposed system is displayed in Figure 3.

The management center establishes, in a general setting with IoT data providers, a cloud of data sources that are publicly accessible as encrypted segments of data streams, and an infrastructure for conditional access to these data. This infrastructure provides data users with encrypted data. The main parts of the infrastructure are a blockchain system with smart contracts where users can perform subscriptions for recovering the source data from the encrypted ones within a certain time segment. The access control is based on BE paradigm, and the management center establishes this BE-based subsystem during the initialization phase.

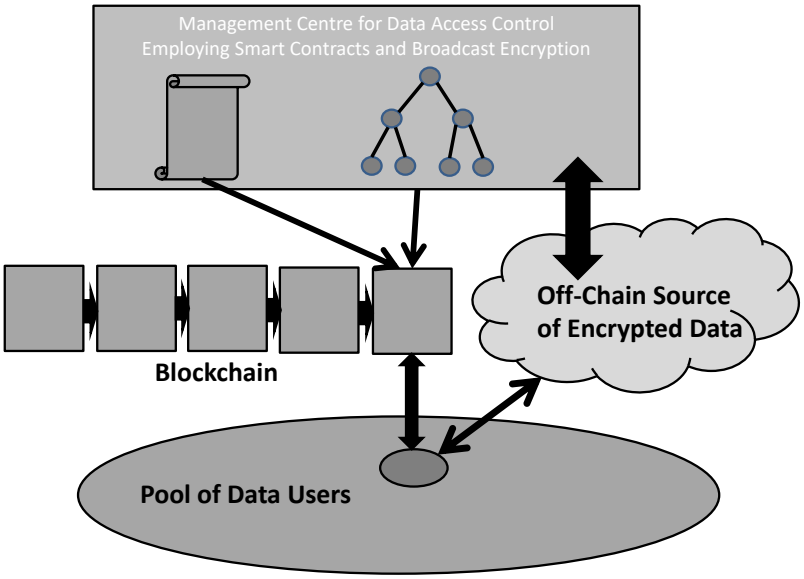

Figure 3. Basic architecture of the blockchain smart contracts and broadcast encryption-based data access control.

4.2. Framework Algorithms for the Initialization and Dynamical Data Access Control

In the initialization phase, the provider designs: (i) BE based subsystem for controlling the access to the encrypted data that follows the traditional BE paradigm with KEKs and SEKs described in Section 2.2; (ii) The access point for delivering encrypted streaming data as an off-blockchain source of encrypted data; (iii) The smart contracts for data users' subscription and dynamic data access control. The assumption is that the data available at the access point are collected in an appropriate manner from certain data sources available within IoT.

Algorithm of the System Initialization

The provider performs the following.

1. Designs cryptographic keys management structure, typically as a binary balanced tree such that number of leaves corresponds to the number of expected users;
2. Selects the encryption scheme for data encryption employing time-varying SEKs, and for encryption of SEKs employing KEKs;
3. Establishes the access point for delivering encrypted streaming data to the subscribers.
4. Prepares and deploys blockchain smart contracts that determine subscription authorization of the users to access the streaming data on the "pay-per-view" approach.

Upon the initialization of the system, the data access control is performed employing dedicated smart contract and broadcast encryption. The generic paradigm for data access control follows the next algorithm.

Algorithm for Data Access Authorization

1. A user calls the blockchain smart contract to subscribe to data and obtain access in a certain time slot.
2. Employing smart contracts, a set of eligible users that should have access to the streaming data is specified.
3. In the considered time slot, if required, a new SEK is defined—otherwise, SEK from the previous time slot is used.

4. Using the KEKs tree, the minimum subset Ω_t of KEKs is identified such that all legitimate users in the time slot t can recover SEK_t employed for encryption of data D_t and the manager delivers to the blockchain the following

$$\{(ID_{KEK_i}, E_{KEK_i}(SEK_t)\}_{i \in \Omega_t}$$

providing that each legitimate user can recover SEK_t.
5. A user accesses the information delivered in Step 4 and the encrypted data $E_{SEK_t}(D_t)$. Further on, a user decrypts SEK_t and performs data decryption employing decrypted SEK_t.

4.3. An Experimental Evaluation of the Overhead Reduction

This section provides an experimental evaluation of the gain in overhead reduction when BE is employed instead direct authorization of each user.

The binary tree structure is used to organize data users. More specifically, for simplicity reasons, the nodes in a KEK tree are enumerated as $\{0, 1, \ldots, 2^n - 1\}$, where n is the tree level. Nodes are grouped two by two (siblings) and paired nodes have a common parent node. From there, it follows that there are 2^{n-1} parent nodes and they are located at $n - 1$ tree level. Node (parent) k at penultimate level is parent of nodes $2k$ and $2k + 1$. Similarly, nodes at $n - 2$ level are paired so that node k is the parent of the nodes $2k$ and $2k + 1$ and grandparent of the nodes $4k, 4k + 1, 4k + 2$ and $4k + 3$. Similarly, node k at level $n - m$ is an predecessor of nodes $2^m k, 2^m k + 1, 2^m k + 2, \ldots, 2^m k + 2^m - 1$. If the keys should be distributed to the nodes $2^m k, 2^m k + 1, 2^m k + 2, \ldots, 2^m k + 2^m - 1$, it is sufficient to distribute the key only to the node k at $n - m$ level, i.e., to distribute a single key instead of 2^m keys. The covering algorithm is as follows:

1. Mark the leaf nodes (at level n) corresponding to Data Users that should be distributed the keys.
2. For each tree level m starting from n to 1 check if nodes $2k$ and $2k + 1$ are marked, where $k = 0, 1, 2, \ldots, 2^{m-1}$. If both nodes $2k$ and $2k + 1$, then mark their parent node k at $m - 1$ level and mark off nodes $2k$ and $2k + 1$. This means that individual keys $2k$ and $2k + 1$ do not have to be sent, but instead key k is used. This reduces the number of keys distributed. If at least one of the nodes $2k$ and $2k + 1$ is not marked, then their parent node k is not marked. Namely, since not both $2k$ and $2k + 1$ are marked, then in the subtree whose root corresponds to the not-marked node exists at least one node that should not receive the key. From this, it follows that key k should not be sent, otherwise a node that should not receive the key would have received it.

The complexity of the algorithm corresponds to the number of nodes in the tree since each node is checked in constant time. If tree height is n then there are 2^{n-1} (potential) Data Users. The total number of keys is $2^n - 1$. Consequently, the complexity of the algorithm is linear in the total number of Data Users.

The following experiments were conducted in order to demonstrate the efficiency of the algorithm. The maximal number of supported Data Users (i.e., the number of leaf nodes in the tree) is 1024, 32,768 and 1,048,576 = 2^{20}, respectively, and let 25%, 50% and 75% be the percentage of subscribers (i.e., the number of users that have access to data and that should receive the key). Twenty instances were generated for each possible combination (1024 maximal supported Data Users, 25% subscribers, 1024 maximal supported Data Users, 50% subscribers, 1024 maximal supported Data Users, 75% subscribers, 32,768 maximal supported Data Users, 25% subscribers, etc.). Table 1 shows the average number of distributed keys. The experiments were performed on a Linux 64bit platform that uses Intel® Core™ i7-4710MQ CPU @ 2.50GHz with eight cores.

Table 1. The relationship between the number of Data Users and the number of distributed keys.

Supported Number of Data Users	Number of Subscribers	Average Number of Distributed Keys	Execution Time
1024	256	211	0.001
1024	512	317	0.001
1024	768	283	0.001
32,768	8192	6717	0.01
32,768	16,384	10,162	0.01
32,768	24,576	9159	0.01
1,048,576	262,144	214,573	0.047
1,048,576	524,288	324,556	0.086
1,048,576	786,432	291,924	0.110
33,554,432	8,388,608	6,866,655	1.696
33,554,432	16,777,216	10,389,990	3.124
33,554,432	25,165,824	9,341,821	4.008

In addition, a more comprehensive evaluation was conducted in order to analyze the relationship between the percentage of Data Users and the number of distributed keys. A maximal number of Data Users is set to 1048576 and the percentage of subscribers was set to 5%, 10%, 15%, . . . , 90%, and 95% respectively. Table 2 shows the average number of distributed keys.

Table 2. The relationship between the percentage of subscribers and the number of distributed keys when the maximal number of Data Users is 2^{20} = 1,048,576.

% of Subscribes	Average Number of Distributed Keys	Execution Time
5	50,497	0.0102
10	97,247	0.0208
15	140,190	0.0300
20	179,352	0.0398
25	214,596	0.0472
30	245,701	0.0568
35	272,556	0.0644
40	294,812	0.0698
45	312,258	0.0790
50	324,564	0.0856
55	331,360	0.0922
60	332,140	0.0958
65	326,250	0.1024
70	313,216	0.1056
75	291,920	0.1100
80	261,424	0.1122
85	220,247	0.1150
90	165,919	0.1158
95	95,314	0.1142

A graphical illustration of the advantage gained through broadcast encryption in terms of the number of keys distributed is shown in Figure 4.

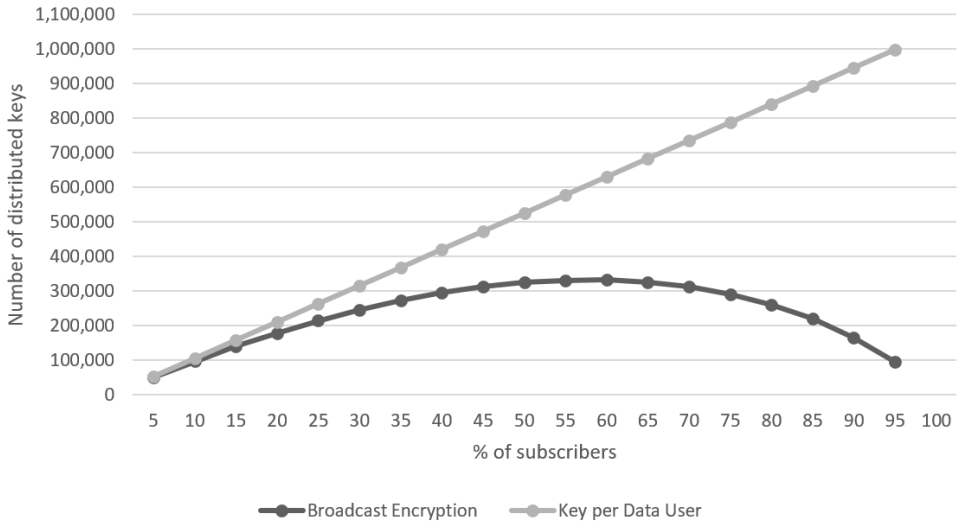

Figure 4. Comparison between the number of distributed keys when broadcast encryption is used vs. individual key per Data User when the maximal number of Data Users is set to 2^{20}.

5. An Instance of the Proposed Framework

This section provides details of the proposed approach for broadcast encryption and blockchain-based access control. The system model comprises five components (see Figure 5):

(1) **IoT network** which acts as a data creator. Data generated and collected by devices in the network is fed to the Data Provider entity and Data Users can access it in accordance with the access control rules;
(2) **Data Provider** is in charge of data encryption, data publication on Data Storage, and definition of access control rights;
(3) **Data User** can access and read data from Data Storage in accordance with access rights;
(4) **Data Storage** enables storing and accessing encrypted data;
(5) **Blockchain network** enables interactions between Data Users and Data Provider, and stores access control information.

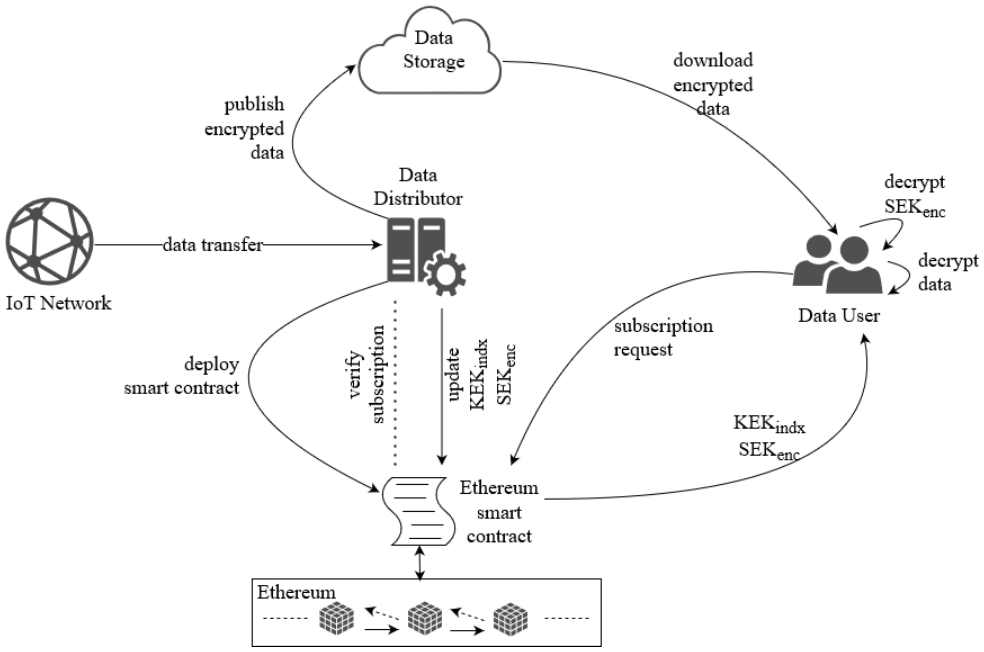

Figure 5. Model of the proposed broadcast encryption and blockchain based access control system.

5.1. System Initialization Stage

5.1.1. Data Provider Initialization

The initialization stage on the Data provider side consists of the following steps through which Data Provider:

(1) Creates a cryptographic key management structure in the form of a binary balanced tree (KEK tree) where the number of leaves corresponds to the number of expected users;
(2) Selects a data encryption scheme employing time varying SEKs for encrypting data coming from the IoT network;
(3) Establishes a connection to the IoT network to be able to receive and collect the data stemming from the IoT devices;
(4) Registers and connects to the blockchain network since communication with Data Users regarding data access relies on the blockchain, more precisely blockchain smart contracts.

5.1.2. Data User Initialization

Data User initialization stage comprises the following steps through which Data Users:

(1) Receives the KEKs assigned to them by Data Provider. Distribution of KEKs has to be performed in a secured way, e.g., a user personally picks up the keys on a USB stick while providing a proof of their identity;
(2) Registers and connects to the blockchain network in order to be able to subscribe to and read relevant data that is offered by Data Provider.

5.2. Data Publishing Stage

When the Data Provider receives new data from the IoT network, it encrypts the data with a new SEK and sends the result of the encryption to the Data Storage. Since the content on Data Storage is encrypted, Data Users who do not have access rights will not able to get the original data. Data Provider deploys the smart contract on the blockchain network and the smart contract contains essential information such as the path to the data on the Data Storage, informative description, etc. Apart from this information, the main role of the smart contract is to support and facilitate the realization of broadcast encryption. The smart contract will store access control information, i.e., the (encrypted) SEK used for data encryption, as well as the indexes of the KEKs used for SEK encryption. After the smart contract is deployed, notifications about the availability of fresh data are sent over the blockchain network, so that Data Users are informed about this.

The steps of the data publishing process are summarized as follows:

(1) The IoT network feeds data to the Data Provider;
(2) Data Provider generates a key (SEK) and encrypts the data using the key;
(3) Data Provider dispatches the encrypted data to Data Storage;
(4) Data Provider deploys the smart contract containing information about the data. The principal purpose of the smart contract will be to aid data access control;
(5) Data Users are notified about newly published content via the blockchain network.

Figure 6 shows the structure of the smart contract Data Provider deploys to the blockchain.

5.3. Data Access Control

Communication between the Data Provider and Data Users is performed through the smart contract, and as the blockchain represents a public, immutable ledger of transactions, it will permanently store information about requests and data access. In this way, the blockchain also plays the role of a reliable activity log, i.e., a data access register. By looking at the blockchain, it can be determined whether there was an irregularity in the access control process, i.e., whether, on the one hand, access to the file is enabled for users who do not have the right to do so, and, on the other hand, whether access is denied to legitimate Data Users, e.g., users who have subscribed to the data, etc.

Figure 6. The structure of the blockchain smart contract for data access control.

5.3.1. Adding New Data User

The procedure for adding a new Data User consists of the following steps:

(1) The Data User sends a request to subscribe and access data by calling the smart contract's *subscribeRequest* function. In the function call, he provides the hash value of the KEK tree leaf that is assigned to him. Hashing is necessary because, if sent as plaintext, the KEK will be visible to other blockchain participants. The hash of the KEK allows Data Provider to verify the identity of the Data User. Calling the *subscribeRequest* function emits the *NewSubscribeRequestAlert* event whose primary purpose is to notify Data Provider that a new request has arrived from a Data User;
(2) Upon receiving the request, Data Provider first verifies its validity by checking whether the KEK hash value sent by the Data User matches the hash value of a leaf from the KEK tree. If the request is valid, Data Provider finds a new KEK tree covering that includes the new Data User;
(3) When a new covering is found, Data Provider encrypts the SEK with the KEKs found in that covering and updates the values of the KEK indexes and the corresponding encrypted SEK on the smart contract. For this purpose, the Data Provider calls the *updateKeys* smart contract function and passes lists of KEKs indices and encrypted SEK as arguments. This function call changes the state and data on the smart contract in accordance with the passed arguments, and additionally, the *KeysUpdatedAlert* event is emitted, which informs Data Users about the change of valid keys;
(4) When the Data User receives the notification from the blockchain network about the key update, he reads the updated information using smart contract function *getEncSEKByKEKIndex*. As function argument, the Data User specifies the indexes of their KEKs, and the function returns the index of the KEK that is in the current covering and the corresponding SEK ciphertext;
(5) The Data User decrypts the received SEK ciphertext with the corresponding KEK;
(6) The Data User fetches encrypted data from Data Storage and reconstructs the original content, i.e., decrypts the data using the SEK obtained in Step 5.

5.3.2. Removing Data Users

Access to data can be time-limited, i.e., the Data User can subscribe to some content for a certain period of time, e.g., monthly subscription, etc. The time interval, during which the Data User has access to data on Data Storage, can be regulated in the following way:

Data Provider periodically, in accordance with the duration of subscriptions, creates a new SEK and encrypts data on Data Storage using this new key. The entire process consists of the following steps:

(1) Data Provider generates a new key (SEK) to encrypt data;
(2) Data Provider encrypts data using the new key and dispatches data to Data Storage;
(3) Data Provider finds new KEK tree cover which excludes the revoked Data Users, i.e., their KEKs;
(4) Data Provider encrypts the new SEK using current KEKs and updates the data on the smart contract accordingly using the *updateKeys*;
(5) The call to the *function* emits the *KeysUpdatedAlert* which serves the purpose of notifying Data Users about the change of valid keys. Only valid Data Users will have access to data, while the revoked Data Users will not be able to decrypt the data from Data Storage, since they will not know the current SEK;

5.4. Implementation and Illustrative Experimental Evaluation

Ethereum blockchain is used for system implementation. Ethereum is a public blockchain with support for smart contracts. The smart contract that implements the data access control mechanism is developed using the Remix Integrated Development Environment (IDE) and it was deployed on Ethereum public test network Goerli using the Metamask software (https://metamask.io/, accessed on 16 January 2023). The smart contract is written in Solidity language, which is the primary programming language for developing smart contracts on the Ethereum blockchain. Figure 7 shows the source code of the smart contract.

For illustrative experimental evaluation, a comparison regarding Ethereum gas consumption for distributing the keys in the proposed model, which employs BE, and the model without BE (a key per Data User) is performed. Encrypted SEKs are stored in the smart contract as *bytes32* data type. In order to manage access privileges, i.e., grant or revoke access to the content published off-chain, it is necessary to interact with the smart contract and change the smart contract storage accordingly. This interaction is performed through blockchain transactions. In the Ethereum blockchain, whenever a transaction is executed, the transaction fee is charged in terms of Ethers. Fees include calculations, storing or manipulating data, token transfers, etc. which consume different amounts of "gas". Ethereum "gas" is a computational unit introduced to calculate the transaction fee. The gas system is included to ensure the credibility of transactions and to keep the blockchain network safe. For transaction calls to smart contracts, gas consumption is calculated against the smart contract code execution, as blockchain miners provide their resources so that smart contracts can be deployed and executed. Whenever a smart contract transaction is performed, a number of Ethereum Virtual Machine (EVM) operational codes (opcodes) are executed. The opcodes perform corresponding stack operations implemented by EVM, each of which has some units of gas associated with it. Appendix G of the Ethereum yellow paper (https://ethereum.github.io/yellowpaper/paper.pdf, accessed on 16 January 2023) specifies the costs, in terms of gas, for EVM opcodes. Since the main purpose of the smart contract (see Figure 7) employed in the proposed model is key distribution, the focus in on storage costs. The SSTORE opcode saves values to smart contract storage. It is shown that SSTORE has a direct relationship with gas consumption, i.e., as its occurrences increase, the gas consumption increases and vice versa [30]. SSTORE is one of the opcodes that have a very strong impact on the increase of gas consumption. Based on the evaluation of storage costs provided in Section 4.3 expected gas consumption comparison, in terms of Ethereum SSTORE operations, between the proposed model and the approach based on individual keys per user is derived. This comparison is shown in Figure 8. The applicability and feasibility of a blockchain-based solution largely depends on operational costs and gas consumption and it is evident that the proposed model achieves significant improvement in terms of gas consumption.

```
pragma solidity >=0.4.0 <0.7.0;

contract DataAccessControl {

    address distributor;
    string dataURL;
    string dataDescription;
    mapping(int32 => bytes32) encSEK;
    int32[] indxsKEK;

    constructor(string memory _url, string memory _desc) public {
        distributor = msg.sender;
        dataURL = _url;
        dataDescription = _desc;

        emit NewContentAlert(dataURL, dataDescription);
    }

    event NewContentAlert(
        string dataURL,
        string dataDescription
    );

    function subscribeRequest(bytes32 leafKEKHash) public {
        emit NewSubscribeRequestAlert(msg.sender, leafKEKHash);
    }

    event NewSubscribeRequestAlert(
        address subscriber,
        bytes32 leafKEKHash
    );

    function updateKeys(int32[] memory indxs, bytes32[] memory vals) public {
        if (msg.sender != distributor) return;

        for (uint32 i=0; i<indxsKEK.length; i++) {
            delete encSEK[indxsKEK[i]];
        }

        for (i=0; i<indxs.length; i++) {
            encSEK[indxs[i]] = vals[i];
        }

        indxsKEK = indxs;

        emit KeysUpdatedAlert();
    }

    event KeysUpdatedAlert();

    function getEncSEKByKEKIndex(int32[] memory indxs) public view returns (int32, bytes32) {
        int32 res_kekIndx = -1;
        bytes32 res_encSEK = 0;

        uint i;
        for(i=0; i<indxs.length && res_encSEK==0; i++) {
            res_encSEK = encSEK[indxs[i]];
        }

        if(res_encSEK != 0) res_kekIndx = indxs[i-1];

        return (res_kekIndx, res_encSEK);

    }
}

```

Figure 7. The code of the blockchain smart contract for data access control.

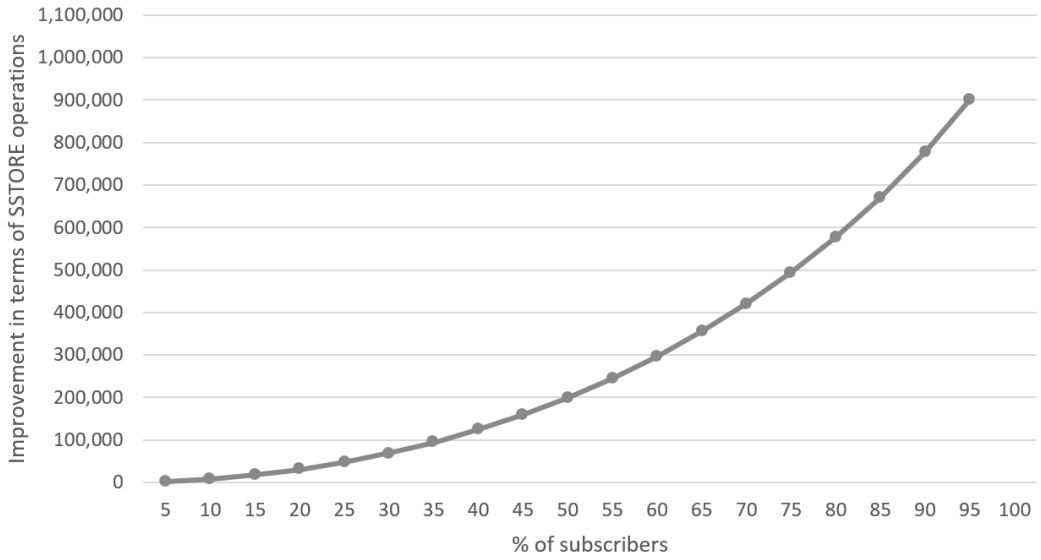

Figure 8. Improvement of the proposed model in comparison with the approach employing individual key per Data User, when the maximal number of Data Users is set to 2^{20}.

6. Discussion

Security. Regarding the security of the proposed approach, note that there are two mutually independent issues: the security of the employed BE, and security of the employed smart contracts. Regarding BE security, note that the employed BE follows the same paradigm as the traditional BE with the following two differences: (i) the public header for recovering data access privileges is available in the blockchain, not as a header of the encrypted data; (ii) the encrypted data "appears" from an IoT public source and an IoT device approaches to the encrypted data and locally, by itself, perform decryption and further data use. Obviously, these two differences have no impact on the employed BE scheme for the conditional data access, i.e., it is the same as the source one.

In particular, note that the smart contract delivers only encrypted versions of the SEK, and so the cryptographic security of this delivery depends only on the employed encryption technique.

Regarding smart contract security, recall that some of the principal blockchain characteristics are transparency, verifiability, immutability, and unforgeability. This means that in the model proposed in this paper, granting and revoking access privileges is traceable and cannot be altered or obliterated. Due to blockchain radical transparency, all blockchain content, including smart contracts, can be read by all network participants. As a consequence, storing secret or private data on blockchain represents a security breach. However, it should be noted that the secret values (SEKs) in the proposed access control model, are stored on the blockchain, i.e., in a smart contract, in an encrypted form and thus protected. Therefore, there is no conflict between the inherent transparency of blockchain and data privacy. Trustlessness is one of the biggest premises for smart contracts and decentralized applications. Smart contracts are immutable and cannot be altered. They will execute the business logic defined in the code at the time of deployment. In order for a smart contract to be trustless, its code should be made available for independent verification. Without verification, smart contracts can have exploitable vulnerabilities that compromise the safety and correctness of the system. Etherscan (https://etherscan.io/, accessed on 16 January 2023) is mostly known as an Ethereum blockchain explorer, but it also offers source code verification services for smart contracts. When the smart contract code is verified, it receives a "Verified" label and it is published on Etherscan to be audited. Etherscan is the most used tool for verifying contracts. Smart contract from Figure 7 is deployed and verified on Goerli Testnet and it is available at the address *0xd24F3296A3af5A31fc42F91320a789EF81A7de80* (https://goerli.etherscan.io/address/0xd24F3296A3af5A31fc42F91320a789EF81A7de80, accessed on 16 January 2023).

Efficiency. The implementation of the proposed approach for data access control is performed employing the basic Ethereum platform. On the other hand, regarding energy consumption, if a PoW-based consensus protocol should be involved in the employed blockchain platform, note the following. In order to reduce the energy consumption of PoW, an energy memory-based trade-off blockchain consensus protocol and accordingly modified Ethereum platform reported in [31,32] could be employed. A more detailed consideration of this issue could be a direction for further work.

Comparison with traditional BE—Differences and Benefits. In a traditional BE with stateless receivers, the access privileges, i.e., a collection of the session key SEK ciphertexts generated for different KEKs, are the header of the data encrypted with SEK. The encrypted data and the access control header are delivered through a noisy channel where the header could be corrupted so that certain legitimate receivers will not receive the subscribed service. There are the following two important differences between the employed BE in comparison with the traditional one: (i) substantially different approaches for delivering the encrypted forms of SEK and data encrypted using that SEK; (ii) trustful and transparent delivery of encrypted SEKs. One of the BE goals is to provide efficient and trustful inclusion and exclusion of subscribers into the pool of privileged ones. Blockchain achieves this goal because the privileges are distributed employing blockchain and smart contracts. In a traditional setting the inclusion information is distributed vie an untrusted communication channel. Accordingly, the immutability of blockchain provides enhanced trust in BE-based data access control. In particular, note that the blockchain-supported BE brings the following important differences in comparison with the traditional ones: (i) a new efficient and trustful technique for delivering encrypted SEKs to the privileged users; (ii) more flexible control of conditional access to the data as an implication of the smart contract employed; (iii) different channels for delivering the data access privileges (on-chain) and the data to be accessed (off-chain).

Author Contributions: Conceptualization, M.J.M.; methodology, M.J.M.; software, M.K., D.U.; validation, M.J.M., L.W. and S.X.; formal analysis, M.J.M.; writing—original draft preparation, M.J.M., M.K., D.U.; writing—review and editing, M.J.M., M.K., L.W. and S.X.; supervision, L.W. and S.X.; project administration, L.W. and S.X.; funding acquisition, L.W. and S.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Shandong Provincial Key Research and Development Program (2020CXGC010107, 2019JZZY020129), the Science, Education and Industry Integration Innovation Program of Qilu University of Technology (Shandong Academy of Science) (2020KJC-GH11); Shandong Provincial Key Research and Development Program of China (2021CXGC010107, 2020CXGC010107). D.U. was supported by the project AI4TrustBC, Science Fund of the Republic of Serbia.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lotspiech, J. Broadcast encryption's bright future. *Computer* **2002**, *35*, 57–63. [CrossRef]
2. Sunny, F.A.; Hajek, P.; Munk, M.; Abedin, M.Z.; Satu, M.S.; Efat, M.I.A.; Islam, M.J. A Systematic Review of Blockchain Applications. *IEEE Access* **2022**, *10*, 59155–59177. [CrossRef]
3. Kemmoe, V.Y.; Stone, W.; Kim, J.; Kim, D.; Son, J. Recent Advances in Smart Contracts: A Technical Overview and State of the Art. *IEEE Access* **2020**, *8*, 117782–117801. [CrossRef]
4. Du, Y.; Liu, J.; Guan, Z.; Feng, H. A Medical Information Service Platform Based on Distributed Cloud and Blockchain. In Proceedings of the 2018 IEEE International Conference on Smart Cloud, New York, NY, USA, 21–23 September 2018; pp. 34–39. [CrossRef]
5. Jin, H.; Xu, C.; Luo, Y.; Li, P. Blockchain-Based Secure and Privacy-Preserving Clinical Data Sharing and Integration. In Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing, New York, NY, USA, 2–4 October 2020; pp. 93–109.
6. Lin, H.; Zhang, H.; Yan, H.; Wang, H.; Shi, Y.; Gao, F.; Wen, Q. A Secure Online Treatment Blockchain Service. *Wirel. Pers. Commun.* **2021**, *117*, 1773–1795 . [CrossRef]

7. Zhang, L. Key Management Scheme for Secure Channel Establishment in Fog Computing. *IEEE Trans. Cloud Comput.* **2021**, *9*, 1117–1128. [CrossRef]
8. Sohal, M.; Bharany, S.; Sharma, S.; Maashi, M.S.; Aljebreen, M. A Hybrid Multi-Cloud Framework Using the IBBE Key Management System for Securing Data Storage. *Sustainability* **2022**, *14*, 13561. [CrossRef]
9. Chen, L.; Li, J.; Zhang, Y. Adaptively Secure Anonymous Identity-based Broadcast Encryption for Data Access Control in Cloud Storage Service. *KSII Trans. Internet Inf. Syst.* **2019**, *13*, 1523–1545.
10. Chen, T.; Zhang, L.; Choo, K.-K.R.; Zhang, R.; Meng, X. Blockchain Based Key Management Scheme in Fog-Enabled IoT Systems. *IEEE Internet Things J.* **2021**, *8*, 10766–10778. [CrossRef]
11. Lotspiech, J. Broadcast encryption versus public-key cryptography in content protection systems. In Proceedings of the 9th ACM Workshop on Digital Rights Management, Chicago, IL, USA, 9 November 2009; pp. 39–46.
12. Lee, J.; Kim, J.; Oh, H. BESTIE: Broadcast Encryption Scheme for Tiny IoT Equipment. *Electronics* **2020**, *9*, 1389. [CrossRef]
13. Lee, J.; Lee, S.; Kim, J.; Oh, H. Combinatorial Subset Difference—IoT-Friendly Subset Representation and Broadcast Encryption. *Sensors* **2020**, *20*, 3140. [CrossRef]
14. Yu, Q.; Li, J.; Ji, S. Identity-Based and Leakage-Resilient Broadcast Encryption Scheme for Cloud Storage Service. *Appl. Sci.* **2022**, *12*, 11495. [CrossRef]
15. Yu, Q.; Li, J.; Ji, S. Anonymous Identity Based Broadcast Encryption against Continual Side Channel Attacks in the State Partition Model. *Appl. Sci.* **2022**, *12*, 9395. [CrossRef]
16. Malluhi, Q.; Tran, V.D.; Trinh, V.C. Decentralized Broadcast Encryption Schemes with Constant Size Ciphertext and Fast Decryption. *Symmetry* **2020**, *12*, 969. [CrossRef]
17. Hu, H.; Cao, Z.; Dong, X. Autonomous Path Identity-Based Broadcast Proxy Re-Encryption for Data Sharing in Clouds. *IEEE Access* **2022**, *10*, 87322–87332. [CrossRef]
18. Chen, Y.; Hu, B.; Yu, H.; Duan, Z.; Huang, J. A Threshold Proxy Re-Encryption Scheme for Secure IoT Data Sharing Based on Blockchain. *Electronics* **2021**, *10*, 2359. [CrossRef]
19. Razzaq, A.; Mohsan, S.A.H.; Ghayyur, S.A.K.; Alsharif, M.H.; Alkahtani, H.K.; Karim, F.K.; Mostafa, S.M. Blockchain-Enabled Decentralized Secure Big Data of Remote Sensing. *Electronics* **2022**, *11*, 3164. [CrossRef]
20. Sun, S.; Du, R.; Chen, S. A Secure and Computable Blockchain-Based Data Sharing Scheme in IoT System. *Information* **2021**, *12*, 47. [CrossRef]
21. Huang, D.-C.; Liu, L.-C.; Deng, Y.-Y.; Chen, C.-L. A Digital Media Subscription Management System Combined with Blockchain and Proxy Re-Encryption Mechanisms. *Symmetry* **2022**, *14*, 2167. [CrossRef]
22. Saha, R.; Kumar, G.; Conti, M.; Devgun, T.-h.; Kim, T.; Alazab, M.; Thomas, R. DHACS: Smart Contract-Based Decentralized Hybrid Access Control for Industrial Internet-of-Things. *IEEE Trans. Ind. Inform.* **2022**, *18*, 3452–3461. [CrossRef]
23. Ullah, Z.; Raza, B.; Shah, H.; Khan, S.; Waheed, A. Towards Blockchain-Based Secure Storage and Trusted Data Sharing Scheme for IoT Environment. *IEEE Access* **2022**, *10*, 36978–36994. [CrossRef]
24. Zulkifl, Z.; Khan, F.; Tahir, S.; Afzal, M.; Iqbal, W.; Rehman, A.; Saeed, S.; Almuhaideb, A.M. FBASHI: Fuzzy and Blockchain-Based Adaptive Security for Healthcare IoTs. *IEEE Access* **2022**, *10*, 15644–15656. [CrossRef]
25. Yang, X.; Zhang, C. Blockchain-Based Multiple Authorities Attribute-Based Encryption for EHR Access Control Scheme. *Appl. Sci.* **2022**, *12*, 10812. [CrossRef]
26. Yu, C.; Zhan, Y.; Sohail, M. SDSM: Secure Data Sharing for Multilevel Partnerships in IoT Based Supply Chain. *Symmetry* **2022**, *14*, 2656. [CrossRef]
27. Pan, H.; Zhang, Y.; Si, X.; Yao, Z.; Zhao, L. MDS^2-C^3PF: A Medical Data Sharing Scheme with Cloud-Chain Cooperation and Policy Fusion in IoT. *Symmetry* **2022**, *14*, 2479. [CrossRef]
28. Li, Y.; Zhou, F.; Ji, D.; Xu, Z. A Hierarchical Searchable Encryption Scheme Using Blockchain-Based Indexing. *Electronics* **2022**, *11*, 3832. [CrossRef]
29. Deng, Y.; Wang, S.; Zhang, Q.; Zhang, D. A Secure Subscription-Push Service Scheme Based on Blockchain and Edge Computing for IoT. *KSII Trans. Internet Inf. Syst.* **2022**, *16*, 445–466. [CrossRef]
30. Khan, M.M.A.; Sarwar, H.M.A.; Awais, M. Gas consumption analysis of Ethereum blockchain transactions. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e6679. [CrossRef]
31. Mihaljević, M.J.; Wang, L.; Xu, S.; Todorović, M. An Approach for Blockchain Pool Mining Employing the Consensus Protocol Robust against Block Withholding and Selfish Mining Attacks. *Symmetry* **2022**, *14*, 1711. [CrossRef]
32. Mihaljevic, M.J. A Blockchain Consensus Protocol Based on Dedicated Time-Memory-Data Trade-Off. *IEEE Access* **2020**, *8*, 141258–141268. [CrossRef]