information

MDPI

Article

An ECDSA Approach to Access Control in Knowledge Management Systems Using Blockchain

Gabriel Nyame [1,*], Zhiguang Qin [1], Kwame Opuni-Boachie Obour Agyekum [2] and Emmanuel Boateng Sifah [2]

1 School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China; qinzg@uestc.edu.cn

2 School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China; obour539@yahoo.com (K.O.-B.O.A.); emmanuelsifah@yahoo.com (E.B.S.)

* Correspondence: gnyasane@yahoo.com; Tel.: +86-132-5827-5323

Received: 27 December 2019; Accepted: 9 February 2020; Published: 17 February 2020

Abstract: Access control has become problematic in several organizations because of the difficulty in establishing security and preventing malicious users from mimicking roles. Moreover, there is no flexibility among users in the participation in their roles, and even controlling them. Several role-based access control (RBAC) mechanisms have been proposed to alleviate these problems, but the security has not been fully realized. In this work, however, we present an RBAC model based on blockchain technology to enhance user authentication before knowledge is accessed and utilized in a knowledge management system (KMS). Our blockchain-based system model and the smart contract ensure that transparency and knowledge resource immutability are achieved. We also present smart contract algorithms and discussions about the model. As an essential part of RBAC model applied to KMS environment, trust is ensured in the network. Evaluation results show that our system is efficient.

Keywords: access control; blockchain; ECDSA; knowledge management; RBAC; smart contract

1. Introduction

The value of knowledge has risen in today's modern world due to the changes and pace of life that has created a fierce, competitive market. Business strategies have shifted from being product-based to knowledge-based due to the attention being placed on the use of knowledge by corporations. With knowledge being an intellectual property, its management should be very effective. Knowledge management (KM), often related to organizations, is a conscious effort in the provision of knowledge at the right time and place, in the right form, and finally to the right person. This helps people to capture, share and utilize knowledge so as to improve the organization's performance.

Knowledge actually represents an important resource for remaining competitive in business environments. Whether tacit or explicit knowledge [1], the success of knowledge transfer or sharing (i.e., knowledge-sharing effectiveness) is contingent on the knowledge source's capacity to identify and contribute possessed knowledge and the knowledge recipient's predisposition to adopt and re-create knowledge. Quite similar to financial value, it is an intellectual property that needs to be securely stored and shared among parties [2], and, above all, track its provenance, ownership and management. Defined by Quintas et al. [3], knowledge management efforts aim to manage knowledge, by making use of existing and acquired knowledge to meet the needs of and develop opportunities, both now and in the future.

Due to the influx of knowledge, knowledge management systems (KMSs) have been developed by organizations to share and re-use knowledge, and these systems are to ensure that institutional information about several functionalities are available for all to use. From a simple system with stored-in files to complex, heterogeneous array of systems with sophisticated options, there are several components of a KMS, and these KMSs deploy KM portals as central points of access for its component systems.

The role-based access control (RBAC) framework has been the mechanism employed by most KMSs to achieve access control [4,5]. With roles and titles or statuses, instead of users attributed to access rights, many of today's organizations adopt such a model to implement their access control mechanisms [6–10]. With this contention in the existing literature, both researchers and practitioners over the years continued to extend the core RBAC to include features that: (1) protect and wholly secure knowledge assets; (2) align with KM initiatives; and (3) maintain the determination and motivation to share and transfer knowledge. It is important to emphasize that KM initiatives embrace numerous socio-technical elements, processes, structures, and business models in a broader spectrum. Social entities such as individuals, projects teams, collaborative groups, and inter-organizational relationships use sophisticated technological tools to function in organizations. The dynamics associated with systems and human interactions in KMS environment require critical security attention as users generate, store and utilize knowledge assets. A challenge to such KMS strategic initiatives is thus contingent on the appropriateness of the extension of RBAC deployable in KMS without compromising the overall knowledge sharing or transfer agenda.

As with other information systems, users of KMS occupy roles through specific portals and platforms, and these roles are associated with permissions. At different access levels, users can access knowledge items (or objects) and perform operations on them according to their defined specific tasks. With roles associated with a set of sessions, users are allowed to share or transfer knowledge packages across different hierarchical levels. Since knowledge gives organizations competitive advantage [11], and remains a strategic intangible asset of an organization [12], it is highly essential to ensure that only verifiable and authenticated users are permitted to access knowledge objects to any particular depth of granularity. We thus posit that the RBAC model is still an essential access control model to adopt in the KMS environment.

Blockchain has been a revolutionary paradigm in systems of record and it has been seen as the emerging technology in industry and the research community. The technology plays a significant role in controlling, monitoring and, most importantly, securing systems [13–17]. With the facilitation of data sharing and other resources, including knowledge, its combination with other systems enable the automation of time-sensitive workflows in a cryptographic manner. Knowledge shared/given by their owners is always cryptographically proofed by the signature of the sender who holds a unique key pair, and therefore the integrity and authentication of the knowledge is guaranteed. Moreover, all transactions on the knowledge are recorded in a distributed ledger and can be securely traced. As a distributed, programmable and encrypted database for the transfer, storage, protection and access of knowledge from one location to the other, and the provision of a high level of security, the information or knowledge stored in the blockchain cannot be re-written or modified. This design makes the blockchain capable of having a permanent historical record. Although the blockchain can be seen as a remedy for securing most systems including KMSs, and preserving privacy of the knowledge, there are many research challenges that prevent its full incorporation.

KM is based on trust management and the blockchain's trust property can be used to manage trust in generating, storing, sharing, protecting and applying a variety of knowledge. With the help of blockchain, owners' knowledge can be integrated and collectively verified, and they will become transparent to anyone and can be used by users. Thus, users become more confident and comfortable in sharing possessed knowledge, because blockchain offers a better and more secured reporting system compared with the traditional knowledge base. In addition, it is almost impossible to alter the information or knowledge in the knowledge repository due to the security features of the blockchain. In this work, we employ Elliptic Curve Digital Signature Authentication (ECDSA) protocol, which to our best knowledge in the extant KMS literature in particular, has not been used for user verification and authentication in RBAC. To realize the effective use of RBAC model in KMS environment, this work proposes a blockchain approach to realizing access control in KMSs, to grant authorization to knowledge users in the network. Thus, our ECDSA-RBAC feature extends RBAC permission constraints to include stronger authentication in granting permissions to users to access knowledge resources for secured KMS environment. Due to the decentralized and tamper-proof nature of the blockchain, the key ideas this paper puts across are:

1. ensuring an effective user authentication and verification method for knowledge workers in an organization, and therefore providing an efficient access control to knowledge resources;
2. issuing roles and knowledge management to users, and access revocation to defaulting parties in the network; and
3. designing an architecture that seeks to achieve security requirements such as adding, updating, sharing and providing information or knowledge in the organization by making use of the blockchain technology.

The rest of this paper is organized as follows. Section 2 presents the works related to this study while Section 3 considers the background of this study. Section 4 formulates the problem and introduces the system model and its implementation. Section 5 gives the discussions while Section 6 presents the conclusions of the paper.

2. Related Works

Several works have introduced RBAC models for information security and protection. In [6], the authors were the first to present the RBAC model and they proposed the idea of roles connected with privileges, instead of the users. An RBAC model that had four different categories was also proposed by Sandhu et al. [18]. With these fundamentals, many studies have proposed extended versions of the RBAC model. Notably, Xia et al. [19] presented an RBAC model that simplified the complexity of the role hierarchy structure by using namespace. Ma et al. [20] also established a structural model that consisted of three different aspects, and had the thought of a layered management. A novel RBAC model for decentralized and distributed systems was proposed by the authors of [21], and it could be applied to dynamic assignments. We recognize these developmental extensions of RBAC, which seamlessly ensure that there is adequate information access control and management.

KMS is a task-centric information system that enables users to create, store and use knowledge to increase task performance [22,23]. It enables users to improve their knowledge-sharing capabilities for knowledge value creation through knowledge internalization [24]. Although there are varied reasons for deploying KMSs in organizations, one primary objective of all such KMSs' deployment is the ability to facilitating knowledge transfer and sharing for improved knowledge innovation across functional units [12]. Knowledge assets are available and accessible by all individuals and functional units in the organization. For instance, subject-matter experts create knowledge as "best practices" for problem solutions stored in a knowledge base and become an intellectual asset for the organization. Users can then access such knowledge objects of best practices for their specific functions. It is, however, essential to ensure that only verified and authorized users are permitted to access them. It is therefore imperative for knowledge assets to be secured, managed and used by authorized users only for their intended purposes.

For these reasons, the adoption of the RBAC model in KMS is paramount to the control and utilization of knowledge assets throughout the organization. As evidenced in [5,25–28], RBAC adoption in KMS is critical not only in the context of secured knowledge sharing but also the protection of knowledge assets. Thus, RBAC has featured in many KM initiatives as far as protection and security of knowledge assets are concerned.

To provide a secured framework for organizations to share their information or knowledge, there should be some considerations in the design of systems such as KMS with trusted user authentication and authorization. Without these security measures, the RBAC will be insecure, which will result in an unreliable access control. Digital signatures [29] have been proposed to ensure some level of security in access control but the costs involved in their implementation do not make them ideal candidates for wide adoption. Maintaining a public key infrastructure (PKI) also poses a problem [30,31]. Although PKI-based systems are well recognized, they are vulnerable to some security problems, aside their complexity and costs. With a certificate authority (CA) acting as a third party responsible for distribution and management of certificates, there is always the tendency for a single point of failure in such systems. A typical scenario is taken from DigiNotar in [32].

To mitigate the problems associated with PKI systems, decentralized systems have been studied. The authors of [33,34] presented multi-authority attribute-based encryption and signatures, respectively. While there is a requirement of a trusted setup of some parameters in [33,34], it was not in support of this setup. However, their implementation and interoperability always posed problems if several users were involved. By this notion, this work implements the blockchain technology to solve some of these issues. Due to its decentralized and tamper-proof nature, we are able to verify the creators of the model, proof of provenance, and there is a trust environment created to enhance transparency. The content of the models will be stored in timestamped blocks, and therefore the major components of a secured network platform—confidentiality, integrity and availability—will be achieved. In providing efficient access control and preventing malicious activities in the network, ECDSA is utilized to improve the security.

3. Background

In this section, we provide some background information to the technologies used in this work.

3.1. Blockchain

Blockchain first emerged as a paper written by Satoshi Nakamoto [35] in 2008, and its first implementation was in Bitcoin. Bitcoin was the first trust-less, public blockchain, and a peer-to-peer electronic cash system. The blockchain network's transactions is timestamped to continue a chain of hashed blocks that form a record that cannot be tampered with. A sequence of events illustrates the chain. To generate the longest chain, an attacker has to own at least 50% of the network. Due to the need of a huge number of computers to modify the blockchain, it is virtually impossible for the blocks to be corrupted [36,37]. To reach consensus in the network, a majority of the participants should be honest. Nodes can simply leave the network and rejoin at any time, and also accept the longest proof of work (PoW), which is a puzzle that needs to be solved by a miner (node that maintains the block).

The blockchain is a database that is distributed and consists of records that are linked sequentially, and also has the hash of the preceding block. The records are maintained for dispute sake and therefore the blockchain can achieve secure distribution between entities without the requirement of trust among the parties.

Blockchain data are immutable due to the use of cryptographic hashes, with these hashes stored as transactions in the form of a Merkle tree structure [38]. It is used to summarize the transactions in the blockchain in order to reduce transactional costs. It also offers an efficient way of verifying whether a transaction is in a block. Integrity is kept of the ledger due to the linked hash system of the block. Within each block, the data or information is linked together.

3.2. The Smart Contract

A smart contract is a programmable script that executes actions based on already laid down rules stipulated in the network. This code enables the mutual agreement of contractual terms, and ensures that there is no alteration in every recording in the network. Thus, trust is guaranteed and costly duplication is eliminated.

In our knowledge management system, the smart contracts ensure an efficient and effective exchange or transfer of knowledge. They are responsible for adding efficiency to systems by reorienting processes, and this removes humans. The introduction of smart contracts provides an additional layer of security and reduces financial costs, inefficiencies and wastage. In addition, the smart contracts monitor and manage the utilization of knowledge, and apply penalties accordingly.

4. System Model and Implementation

In this section, we first state the problem, and then dwell on the model and implementation of our system. Considering a knowledge-sharing platform, the authenticity of roles and privileges should be of utmost importance to any organization. There is the need to verify whether a particular user is the rightful owner of a particular role corresponding to a specific knowledge resource. Without a proper verification process, the system will not be a secured one and the access control mechanism will be unreliable as well. For instance, passports and ID cards are usually used as verification methods; however, in digital worlds, these cannot be utilized. In this work, we introduce an efficient verification and access control method that is based on Elliptic Curve Cryptography (ECC), as it is the mechanism most blockchain systems thrive on. Our access control system should provide an effective management of the knowledge repository, key issuance and revocation, and verification. The blockchain will also provide transparency and knowledge resource immutability.

Our system, as shown in Figure 1, comprises of the following major components: a user layer, knowledge processing and management units, a blockchain and a cloud server. The various components are explained below.

1. **User layer:** The user layer comprises of the two major entities associated with the knowledge, the knowledge owner and the knowledge user. The knowledge owner uploads its knowledge onto the cloud repository and determines which user has access to which knowledge, and what privileges the user should enjoy. The knowledge user, on the other hand, requests for use of the knowledge, and, upon a successful verification, the request is granted.
2. **Knowledge Processing Unit:** This unit consists of a key issuer, a verification unit, and a role assignment unit. The issuer is responsible for generating cryptographic keys, linked to the identities of the users, and are used for transactions on the network. Some typical transactions include knowledge requests, knowledge usage, etc. The verification unit, on the other hand, verifies all users on the network. Once the keys have been generated and given out to the users by the issuer, all users need to go through an authentication process before gaining access to the knowledge. With the help of the verification unit, an efficient access control is assured. The role assignment unit provides the rules of engagement of the system. It specifies what the roles of each entity are and the privileges each user gets to enjoy.
3. **Knowledge management center:** This unit is the heartbeat of the system. It consists of a network processing node and a smart contract unit. The processing nodes are responsible for processing all requests on the network and managing all other processes on the blockchain. It works in tandem with the smart contract, which is responsible for generating policies on the knowledge. When requests are made, the processing nodes receive the requests from the knowledge processing center and act on them. After processing, it binds a contract to the result and the final output is given to the user.
4. **Blockchain:** This is a growing list of all the knowledge records (knowledge blocks) that are cryptographically linked. In each block, there is a hash of the previous block, a timestamp, and a transaction. Once the processing nodes have completed their tasks, and with them collectively adhering to a protocol for internode communication and validation, the transactions are appended onto a block. Once recorded, the knowledge item cannot be altered without alteration of all the subsequent blocks, which requires all the nodes to reach a consensus.

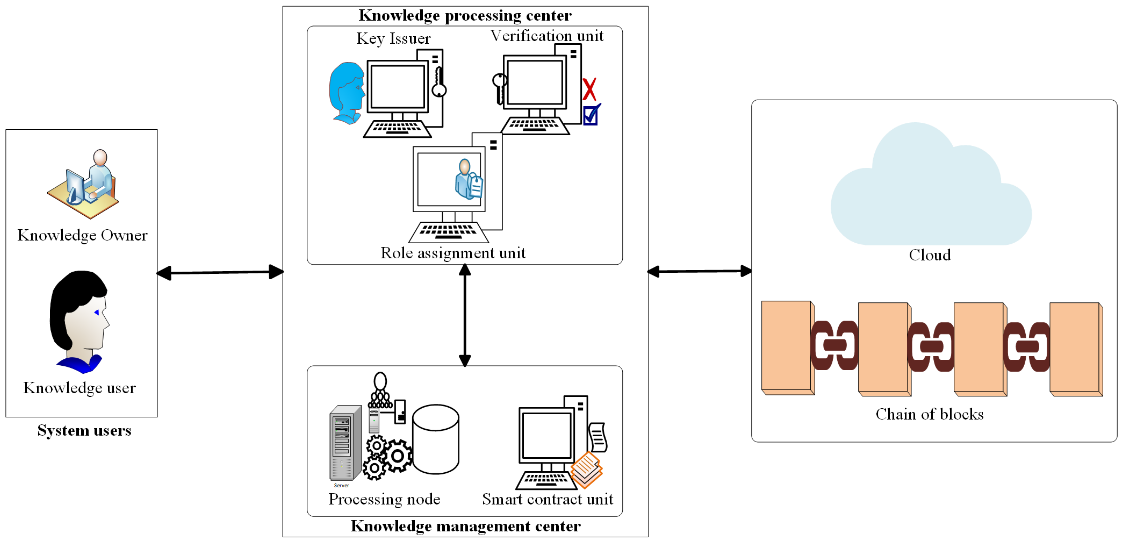

Figure 1. System architecture.

For the creation of a block, the knowledge owner has to log his knowledge into the system and this becomes a block (not verified yet), with each block having a hash value. This hash is a string of characters that is cryptographically constructed. Therefore, the blocks are reliable and each owner can trust the knowledge of its later use. Confirmation of the block is made by all the nodes in the network. All the knowledge in the network is a transaction that is stored between at least two entities involved in the process. All previous transactions related to this knowledge will be available to a user who needs knowledge. With the use of the blockchain, the knowledge will be retained and its ownership maintained and tracked in a secure environment. Figure 2 depicts the knowledge block creation process.

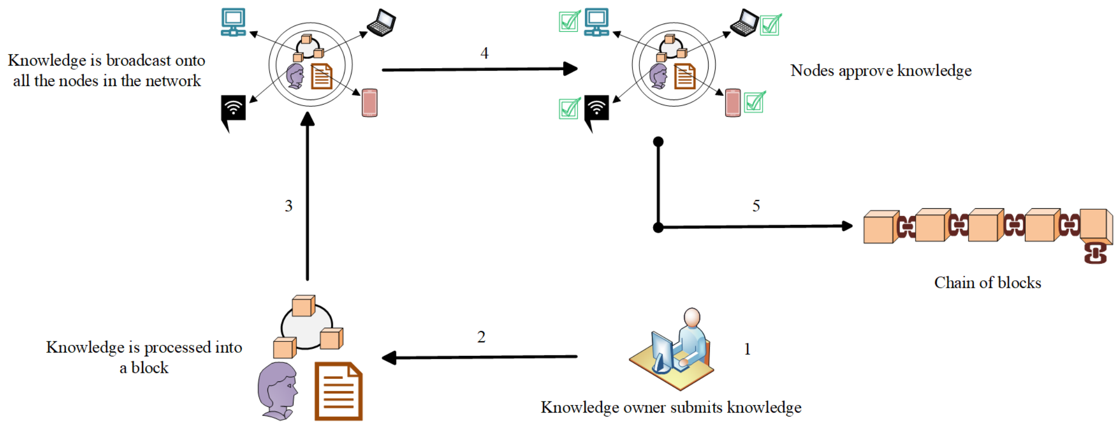
Knowledge is broadcast onto all the nodes in the network
4
Nodes approve knowledge
3
5
Chain of blocks
Knowledge is processed into a block
2
1
Knowledge owner submits knowledge

Figure 2. Creation of a knowledge block.

Figure 3 illustrates the various processes involved in the registration and authentication of a user. The various steps are outlined below.

1. A user contacts the issuer for network membership registration. The parameters needed for registration will be given by the issuer, but, as in many networks, the basic parameters include a unique ID and keying parameters.
2. The issuer generates a user membership key by fetching parameters from its key distribution center and sends the details to the verification unit.
3. The keying parameters are given to the user.
4. The user generates a private key to be used for all transactions by using the ECDSA protocol, which is given in detail in the sections that follow.

5. The verifier and the user establish mutual authentication for key and/or user validity check. Details are obtained from the database. Upon a successful verification process, the user can now access knowledge items and perform actions on the knowledge items.

The ECDSA protocol is used for authentication purposes because of its robust mathematical structure and high security compared to other schemes, and the assurance of digital data unforgeability and non-repudiation. Both discrete logarithms (DL) and ECC provide adequate security levels, but, in relation to parameters, ECC utilizes smaller parameters than DL [39]. Significant benefits of using smaller parameters include faster computations, guarantee of certificates and smaller key sizes. For a public key cryptography, every user or device participating in the communication generally employs a pair of keys (private key and public key) attached with a set of operations to perform the cryptographic operations. Unlike the public key, which is known by all users or devices participating in the communication, the private key is only known by the user in question. Mostly, a section of public key algorithms may demand a set of constants such as 'Domain parameters' that are already defined and known by all participating devices during communication. In terms of any form of shared secret, there is no such need between communicating parties by the public key cryptography as may be required normally by its counterpart—the private key cryptography. However, the private key cryptography is faster than the public key cryptography.

ECDSA, first proposed by Vanstone [39], is an elliptic curve analog to Digital Signature Algorithm (DSA) [40]. It stems from the ECC scheme that was invented by Koblitz [41] and Miller [42]. It is based on points on an elliptic curve over a finite field. The mathematical basis for the security of elliptic curve cryptosystems is the computational intractability of the elliptic curve discrete logarithm problem (ECDLP).

The fourth process in the user registration and authentication is detailed as follows. There are some key factors to consider if the ECDSA protocol has to be successfully achieved. The steps involved are the *setup phase*, *domain parameter generation and validation*, *key pair generation and public key validation*, and *signature generation and verification*.

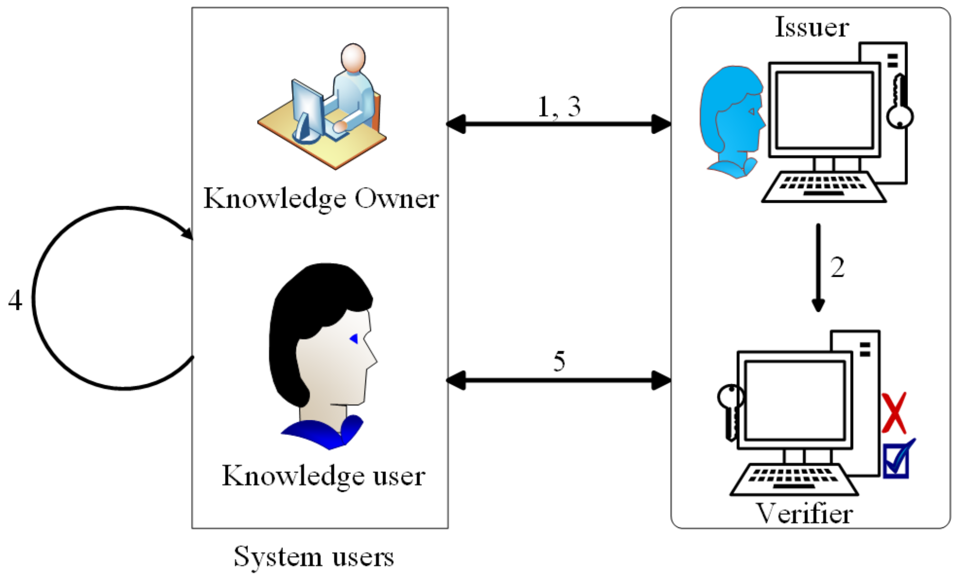

Figure 3. User registration and authentication process.

4.1. Setup Phase

To setup an ECDSA protocol, the domain parameters are made up of a field size q and an odd prime p. Then, there is a field representation $\mathbb{R}$, which is used to represent the elements of the field $\mathbb{H}_q$. Then, two elements from the field, α and β, are chosen and they define the equation of the elliptic curve $\mathbb{E}$ over the field. That is, $y^2 = x^3 + \alpha x + \beta$ if $p > 3$, or $y^2 + xy = x^3 + \alpha x^2 + \beta$, if $p = 2$. x_ξ and y_ξ are two points on $\mathbb{H}_q$ that define a finite point $\xi = \left(x_\xi, y_\xi\right)$. The finite point is of the prime order $\mathbb{E}\left(\mathbb{H}_q\right)$. δ is also the order of the point ξ, with δ either greater than 2^{160} or $4\sqrt{q}$, to ensure maximum security. A cofactor $\sigma = \#\mathbb{E}\left(\mathbb{H}_q\right)/\delta$ is also set.

4.2. Domain Parameter Generation and Validation

For the generation of the domain parameters, first, α and β are randomly selected from $\mathbb{H}_q$. In the case where $q = p$, the curve is represented as $y^2 = x^3 + \alpha x + \beta$, and also as $y^2 + xy = x^3 + \alpha x^2 + \beta$ if $q = 2^m$. Next, we compute $\Delta = \#\mathbb{E}\left(\mathbb{H}_q\right)$. Then, we verify that Δ is divisible by δ. If the verification is unsuccessful, we return to the very first step (where the random parameters are selected). Furthermore, we check that $\mu = q^\phi - 1$ is not divisible by δ, for each $\phi, 1 \leq \phi \leq 20$. If not, we still go back to step 1. Next, we verify that $\delta \neq q$. Finally, we select an arbitrary point $\psi \in \mathbb{E}\left(\mathbb{H}_q\right)$, and set $\xi = \Delta\delta^{-1}\psi$. The process is repeated until $\xi \neq 0$.

To validate the parameters, we input the domain parameters $DP = (q, \mathbb{R}, \alpha, \beta, \xi, \delta, \sigma)$, and the expected output is to either accept or reject the validity of DP. The following are the steps involved.

1. We verify that q is an odd prime or a power of 2. (i.e., $q = p$, $q = 2^m$).
2. We again verify that $\mathbb{H}$ is a valid representation of $\mathbb{H}_q$.
3. Verify that $\xi \neq 0$.
4. Again, verify that $\mathbb{H}_q$ has elements $\alpha, \beta, x_\xi, y_\xi$. These are integers in the interval $[0, p-1]$ if $q = p$, and m-bit length bit strings, if $q = 2^m$.
5. Verify that α and β accurately define the elliptic curve over $\mathbb{H}_q$ (i.e., $4\alpha^3 + 27\beta^2 \not\equiv 0 \mod p$ if $q = p$; $\beta \neq 0$, if $q = 2^m$).
6. Verify that ξ lies on the elliptic curve defined by α and β.
7. Verify that δ is a prime number.
8. Verify that $\delta > 2^{160}$ and also $\delta > 4\sqrt{q}$.
9. Verify that $\delta\xi = \emptyset$.
10. Compute $\rho = \left(1 + \sqrt{q}\right)^2 \cdot \delta^{-1}$, and verify that $\rho = \sigma$.
11. We then verify if $\mu/\delta = \emptyset$.
12. Verify that $\delta \neq q$.

If any of the verification steps fail, DP is invalid. Otherwise, DP is valid.

4.3. Key Pair Generation and Public Key Validation

The key pair is associated with a peculiar set of Elliptic Curve (EC) parameters. While the public key is a random multiple of the base point on the curve, the private key is the integer used to generate that multiple. To generate a key pair, an entity's key pair is associated with the domain parameters, DP, and the user must have the assurance that the domain parameters are valid. Because the public key is generated by the issuer and later sent to the verification unit, the entity should prove its identity to the verification unit.

To generate the keys, the already-registered user does the following: the user selects a random integer t, which is in the interval $[1, \delta - 1]$. Then, the user computes $\kappa = t\xi$. The user's public key is κ, and t is the private key.

To prevent any malice on the network, the following is computed to ensure κ is valid. The objective is to accept or reject κ, after $\kappa = (x_\kappa, y_\kappa)$ and the domain parameters DP are input.

1. Verify that $\kappa \neq \emptyset$.
2. Verify that $\mathbb{H}_q$ has elements x_κ, y_κ. These are integers in the interval $[0, p-1]$ if $q = p$, and m-bit length bit strings, if $q = 2^m$.
3. Check that κ lies on the elliptic curve characterized by α and β.
4. Verify if $\delta\kappa = \emptyset$.

If any of the checks fail, κ is invalid. Otherwise, it is a valid parameter.

4.4. Signature Generation and Verification

In order for the user to prove its identity, represented by a unique value say γ, to the verification unit, the following are the steps involved.

1. **Signature Generation**

   (a) A random integer v is selected, and it must lie in the interval $1 \leq v \leq \delta - 1$.
   (b) Compute $v\xi = (x_1, y_1)$ and convert x_1 to an integer $\check{x_1}$.
   (c) Compute $\lambda = x_1 \mod \delta$. Go to Step (a) if $\lambda = 0$.
   (d) Compute $v^{-1} \mod \delta$, and compute SHA-1(γ) and also covert this bit string to an integer ϵ.
   (e) Compute $s = v^{-1}(\epsilon + t\lambda) \mod \delta$. Go to Step (a) if $s = 0$.

   The signature of the user's identity, γ is (λ, s)

2. **Signature Verification**

   For a successful verification process, the verifier obtains a copy of the user's domain parameters DP, and the associated public key κ. The verifier then has to validate these parameters, by doing the following.

   (a) Verify that λ and s are integers in the interval $[1, \delta - 1]$.
   (b) Compute SHA-1(γ) and convert to an integer ϵ.
   (c) Compute $\varpi = s^{-1} \mod \delta$.
   (d) Compute $\varphi_1 = \epsilon\varpi \mod \delta$ and $\varphi_2 = \lambda\varpi \mod \delta$.
   (e) Compute $\mathbb{X} = \varphi_1\xi + \varphi_2\kappa$.
   (f) If $\mathbb{X} = \emptyset$, reject the signature. Otherwise, convert x_1 of $\mathbb{X}$ to an integer $\check{x_1}$ and compute $\varrho = \check{x_1} \mod \delta$.
   (g) Accept the signature iff $\varrho = \lambda$.

4.5. Smart Contract Design

When a user, after a successful authentication process, makes a request to access a specific knowledge item, the processing node checks from its servers what role(s) the user has on the knowledge. Once the role has been stated, it delivers the knowledge to the user. A smart contract code is issued and bound to the knowledge. This is done to provide effective and quality checks on the knowledge. The various smart contract codes are given in the algorithms below.

Algorithm 1 is executed by the knowledge owner to add users to the network and issue the roles, correspondingly. It adds a timestamp to denote when the contract was executed. The blockchain is consequently updated.

Algorithm 1 Algorithm for addUser

```
Require: INPUT UserAddress (UA), UserName (UN), UserClass (UC), UserRole (UR),
    RoleIssuerAddress (RA), RoleIssuerOrganization (RO), RoleIssuerNote (RN)
Ensure: OUTPUT UserDetails (UD) and RoleIssuerDetails (RD)
1: if RA is true, then
2: if RO is true, then
3: Set counter to 0;
4: Add to UserDetails (UD), UA, UN, UC, UR
5: Add RoleIssuerDetails (RD), RA, RO, RN
6: endif
7: increment counter to 1
8: end
```

To remove a user from the network, the owner again calls upon the smart contract and revokes their roles. The public key of the user is taken into account and after a successful execution of the contract, the user is removed. The blockchain is again updated. Algorithm 2 depicts the *removeUser* contract.

Algorithm 2 Algorithm for removeUser

```
Require: INPUT UserAddress (UA), RoleIssuerAddress (RA), timestamp (TM)
Ensure: OUTPUT UserAddress (UA) and RoleIssuerAddress (RA)
 1: if RA is true, then
 2: Set counter to 0;
 3: Delete from UserDetails (UD), UA
 4: Add to RemovedUser (RU), UA, TM
 5: endif
 6: increment counter to 1
 7: end
```

Next, to request for the knowledge, the smart contract is appended to the request a user makes. It first checks if the requesting party is a legitimate user of the knowledge as specified by the role and privileges the user gets to enjoy on the knowledge by the owner. Algorithm 3 illustrates this contract. Algorithm 4 also shows the provision of access to the user, once s/he has been verified and has made a request. *RecoverSigner* is the verification system in Ethereum using ECC.

Algorithm 3 Algorithm for ServiceRequest

```
Require: INPUT UserAddress (UA), ServiceOrganizationAddress (SA)
Ensure: OUTPUT UserAddress (UA) and ServiceGrantedUsers (SGU)
 1: if SA is true, then
 2: if UA is 'User', then
 3: Set counter to 0;
 4: ProvideAccess()
 5: Add to ServiceGrantedUser (SGU), UA
 6: endif
 7: increment counter to 1
 8: end
```

Algorithm 4 Algorithm for ProvidingAccess

```
Require: INPUT UserAddress (UA), Nonce (NC), Signature (XG), Hashes (HS),
    ServiceOrganizationAddress (SA)
Ensure: OUTPUT Set Access (AX) to true
 1: if RecoverSigner (UA, HS, XG, NC) is true, then
 2: Set Access (AX) to true
 3: endif
 4: end
```

5. Evaluation and Discussion

In this section, we analyze the system and its various features that can enhance the effective sharing of knowledge and role assignment. The incorporation of blockchain technology offers a host of useful features that are not common in traditional systems.

5.1. Evaluation

We show the test results of our smart contract system in this section. The costs of all transactions were measured on the Ethereum blockchain [43], which was used as the blockchain platform. The functions that were tested include *addUser()*, *removeUser()*, *serviceRequest()*, and *accessProvision()*. The tests were run using Ropsten test environment and Remix IDE alongside MetaMask, which provides access to the Ethereum platform and a host of network nodes. Solidity is the language in which our smart contract was written. An Intel Core i7 6700HQ CPU desktop with a processor speed of 2.6 GHz (8 CPUs) and a 16 GB RAM was used to perform the experiments, with a gas rate of $0.03/transfer.

The gas costs of operations (in Gwei) ertr calculated since each function on the blockchain is fixed. The efficiency of our system esd analyzed by comparing the results obtained with the ones obtained in [7], regarding the common parameters. Table 1 illustrates the cost of executing the various functions. There are relatively low costs for all functions, with an average cost of $0.0808. The costs are as a result of the prototypes deployed on our blockchain system, and, by using optimization methods, the costs can be reduced. However, these costs are significantly lower compared to setting up and maintaining a private knowledge. *serviceRequest* and *accessProvision* are not functions in [7], and hence there are no values for them in Table 1.

Table 1. Cost of different functions of the smart contract based on Gas (both Gwei and USD ($)) rates.

Function	Our System [Gas (Gwei)]	Our System [Gas ($)]	Cruz et al. [7] [Gas ($)]
Contract creation	1,560,740	0.2016	0.590
addUser	113,870	0.0147	0.047
removeUser	100,481	0.0130	0.033
serviceRequest	1,253,642	0.1619	-
accessProvision	97,213	0.0126	-

It was realized that, throughout the various operations, our system maintained a cost below $1.00. The lesser is the gas cost, the less expensive is the transaction. Comparing the results obtained for the functions shared between our system and the one in [7], we realized our system performs better. Figure 4 gives a pictorial description of the results.

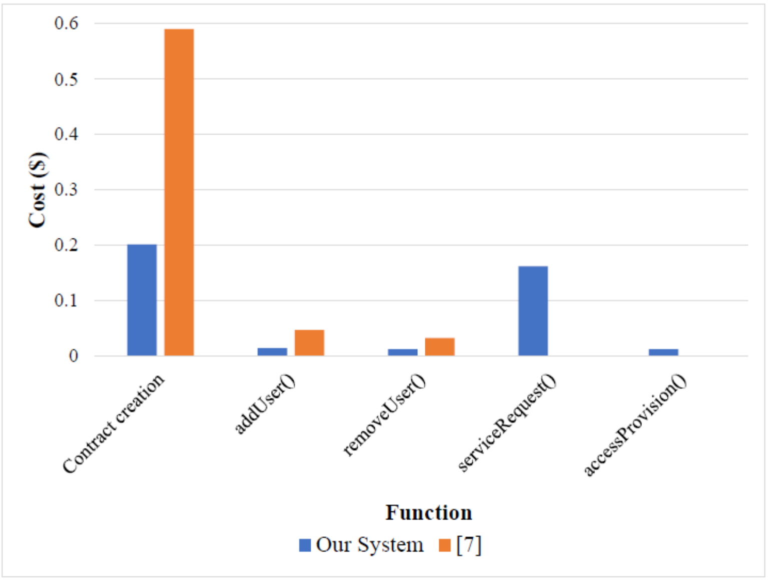

Figure 4. Smart contract function costs.

In Figure 5, we demonstrate the scalability of our blockchain system. For as low as 53 ms, our blockchain system can process 20 requests made by knowledge users, and 140 requests can be processed within 508 ms. From the test results, it was observed that there is a steady increase in transaction processing. This increase can be attributed to the fact that the transaction path and information or knowledge package flow path are separated (i.e., it reduces the delay of transaction processing). By implication, while transactions are broadcast amongst all nodes in the network, the knowledge packets are transmitted directly towards its destination along optimal paths as specified by a routing protocol such as Open Shortest Path First (OSPF) [44]. Moreover, the threshold was set to 140 transactions per second to ensure that the rate of transaction processing remains at a desirable level, which helps to maintain the scalability of our framework.

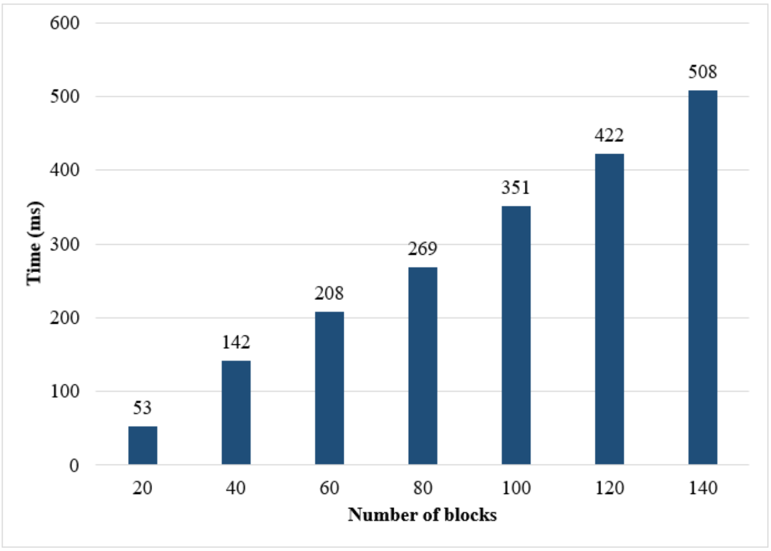

Figure 5. Block processing time.

5.2. Discussion

1. **Security and Privacy:** Security and privacy requirements are significantly important, particularly authentication and integrity, to prevent several attacks in a knowledge management system's environment. ECDSA signature is a security mechanism that alleviates attacks such as modification, spoofing, denial of service and cyber attacks. The use of the algorithm ensures that only legitimate knowledge users are authorized in the network, and the knowledge items are accessible by these users. It ensures that the identities of the users are from a trusted source, and an unknown one is ignored. With the addition of blockchain, the integrity of the knowledge is also assured. The technology ensures that the transmitted knowledge package has not been tampered with. ECDSA signatures provide mutual authentication so as to protect the knowledge repository. It is capable of preventing man-in-the-middle (MITM) attacks, eavesdropping, impersonation, replay and duplication. Furthermore, the utilization of small keys in the ECDSA algorithm provides high performance and appropriate computation costs. Knowledge owners can check the validity of roles according to the timestamp of when the role was issued to the user.
2. **Access Control:** The processing nodes in the blockchain allow for an efficient access control as they infer from the verifier the public keys of the legitimate users. Before any transaction can be completed, the node in charge of processing that request will have to confirm from the verifier if the user has its public key in the verifier's knowledge base. The node can revoke access to a malicious entity, as it must ensure that the blockchain contains only valid knowledge transactions (transactions from legitimate knowledge users). However, malicious nodes can process transactions by granting access to knowledge users without verification. However, then, the smart contract center also performs verification checks, and, if unsuccessful, the transaction is

dropped. The smart contract also permits roles and other information or knowledge package to legitimate knowledge users in the network efficiently and effectively. Users can also be revoked when the *removeUser* function is invoked. This function is very important in cases where a user with malicious intention has been found.

3. **Transparency:** The system achieves transparency as all functions are executed by the smart contract and are logged on the blockchain. A secret action cannot be performed without the awareness of the other entities. In addition, entities will not be able to deny any action committed because every action is linked to their private keys.

6. Conclusions

In this work, an effective and efficient user authentication and authorization protocol for knowledge management systems (KMS) based on elliptic curve cryptography is proposed. In conjunction with the RBAC model, the ECDSA algorithm in the explicitly described system model provides a secure mechanism for accessing knowledge by verifying users before the knowledge is shared, transferred or stored in the knowledge repository. Blockchain is utilized in the work to provide transparency and knowledge immutability, and, with the smart contract added to the KMS, we are able to achieve a secured utilization of roles in an organization. Test results also indicate that our system performs significantly better than the results in [7]. As a future work, we will dwell on the formal security analysis and use some optimization techniques, such as Proof of Search, Proof of Stake, and other consensus protocols, to create more efficient functions.

Author Contributions: Conceptualization, G.N. and K.O.-B.O.A.; methodology, E.B.S.; validation, E.B.S. and Z.Q.; writing—review and editing, G.N. and K.O.-B.O.A.; supervision, Z.Q.; and funding acquisition, Z.Q. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the NSFC-Guangdong Joint Fund (Grant No. U1401257), National Natural Science Foundation of China (Grant Nos. 61300090, 61133016, and 61272527), science and technology plan projects in Sichuan Province (Grant No. 2014JY0172) and the opening project of Guangdong Provincial Key Laboratory of Electronic Information Products Reliability Technology (Grant No. 2013A061401003).

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

Abbreviations

The following abbreviations are used in this manuscript:

RBAC	Role-Based Access Control
ECDSA	Elliptic Curve Digital Signature Authentication
KM	Knowledge Management
KMS	Knowledge Management Systems
PKI	Public Key Infrastructure
CA	Certificate Authority
PoW	Proof of Work
ECC	Elliptic Curve Cryptography
DSA	Digital Signature Algorithm
ECDLP	Elliptic Curve Discrete Logarithm Problem
DL	Discrete Logarithm
EC	Elliptic Curve
OSPF	Open Shortest Path First
MITM	Man-In-The-Middle

References

1. Nonaka, I.; Takeuchi, H. *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*; Oxford University Press: New York, NY, USA, 1995; pp. 3–19.
2. Nemetz, M. A meta-model for intellectual capital reporting. In *International Conference on Practical Aspects of Knowledge Management*; Springer: Berlin/Heidelberg, Germany, 2006.

3. Quintas, P.; Lefrere, P.; Jones, G. Knowledge management: A strategic agenda. *Long Range Plan.* **1997**, *30*, 385–391. [CrossRef]
4. Ogunseye, O.S.; Folorunso, O.; Zhang, J. Preventing Social Engineering and Espionage in Collaborative Knowledge Management Systems (KMSs). *Int. J. E-Adopt.* **2011**. [CrossRef]
5. Chen, T.Y. A multiple-Layer knowledge management system framework considering user knowledge privileges. *Int. J. Softw. Eng. Knowl. Eng.* **2009**. [CrossRef]
6. Ferraiolo, D.; Cugini, J.; Kuhn, D.R. Role based access control: Features and motivations. In Proceedings of the 11th Annual Conference on Computer Security Applications, New Orleans, LA, USA, 11–15 December 1995; IEEE Computer Society Press: Los Alamitos, CA, USA, 1995; pp. 241–248.
7. Cruz, J.P.; Kaji, Y.; Yanai, N. RBAC-SC: Role-Based Access Control Using Smart Contract. *IEEE Access* **2018**, *6*, 12240–12251. [CrossRef]
8. Ferraiolo, D.F.; Barkley, J.F.; Kuhn, D.R. A role-based access control model and reference implementation within a corporate intranet. *ACM Trans. Inf. Syst. Secur.* **1999**. [CrossRef]
9. Gupta, A.; Kirkpatrick, M.S.; Bertino, E. A formal proximity model for RBAC systems. *Comput. Secur.* **2013**. [CrossRef]
10. Malik, A.K.; Dustdar, S. Sharing and Privacy-Aware RBAC in Online Social Networks. In Proceedings of the 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing, Boston, MA, USA, 9–11 October 2011; pp. 1352–1355. [CrossRef]
11. Nonaka, I.; Toyama, R.; Konno, N. SECI, Ba and Leadership: A Unified Model of Dynamic Knowledge Creation. *Long Range Plan.* **2000**. [CrossRef]
12. Ipe, M. Knowledge sharing Sharing in organizations: A conceptual framework. *Hum. Resourc. Dev. Rev.* **2003**, *2*, 337–359 [CrossRef]
13. Panarello, A.; Tapas, N.; Merlino, G.; Longo, F.; Puliafito, A. Blockchain and IoT Integration: A Systematic Survey. *Sensors* **2018**, *18*, 2575. [CrossRef]
14. Kshetri, N. Can Blockchain Strengthen the Internet of Things? *IT Prof.* **2017**, *19*, 68–72. [CrossRef]
15. Joshi, A.D.; Gupta, S.M. Evaluation of design alternatives of End-Of-Life products using Internet of things. *Int. J. Prod. Econ.* **2019**, *208*, 281–293. [CrossRef]
16. Queiroz, M.M.; Samuel, F.W. Blockchain adoption challenges in supply chain: An empirical investigation of the main drivers in India and the USA. *Int. J. Inf. Manag.* **2019**, *46*, 70–82. [CrossRef]
17. Wang, Y.; Singgih, M.; Wang, J.; Rit, M. Making sense of blockchain technology: How will it transform supply chains? *Int. J. Prod. Econ.* **2019**, *211*, 221–236. [CrossRef]
18. Sandhu, R.S.; Coyne, E.J.; Feinstein, H.L.; Youman, C.E. Role-based access control models. *Computer* **1996**, *29*, 38–47. [CrossRef]
19. Xia, L.; Jing, J. An administrative model for role-based access control using hierarchical namespace. In Proceedings of the Pacific Asia Conference on Information Systems (PACIS), Kuala Lumpur, Malaysia, 6–9 July 2006; Volume 30.
20. Lilin, M.; Li, H. A permission model of Saas system based on RBAC. *Comput. Appl. Softw.* **2010**, *27*, 42–44.
21. Li, Q.; Xu, M.; Zhang, X. Towards a group-based RBAC model and decentralized user-role administration. In Proceedings of the 2008 28th International Conference on Distributed Computing Systems Workshops, Beijing, China, 17–20 June 2008.
22. Alavi, M. Knowledge management and knowledge management systems: Conceptual foundations and research issues. *MIS Q.* **2001**, *25*, 107–136. [CrossRef]
23. Feng, K.C.; Chen, E.T.; Liou, W.C. Implementation of knowledge management systems and firm performance: An empirical investigation. *J. Comput. Inf. Syst.* **2004**, *45*, 92–104.
24. Kostova, T. Transnational Transfer of Strategic Organizational Practices: A Contextual Perspective. *Acad. Manag. Rev.* **1999**, *24*, 308. [CrossRef]
25. St. Rose, V. An Empirical Study of the Characteristics of the Role Based Access Control (RBAC) Model in Securing Knowledge Management (KM) and Knowledge Management Systems (KMS). Ph.D. Thesis, Colorado Technical University, Colorado Springs, CO, USA, 2015; ProQuest LLC; ISBN 978-0-3558-2647-0.
26. Bakar, A.; Abdullah, R. A framework of secure KMS with RBAC implementation. *ARPN J. Eng. Appl. Sci.* **2015**, *10*, 1051–1059.

27. Fill, H.-G.; Felix, H. Knowledge Blockchains: Applying Blockchain Technologies to Enterprise Modeling. Available online: https://scholarspace.manoa.hawaii.edu/handle/10125/50398 (accessed on 17 February 2020). [CrossRef]
28. Wang, H.; Guo, X.; Fan, Y.; Bi, J. Extended Access Control and Recommendation Methods for Enterprise Knowledge Management System. *IERI Procedia* **2014**, *10*, 224–230. [CrossRef]
29. Farrell, S.; Housley, R. An Internet Attribute Certificate Profile for Authorization; RFC 3281; 2002. Available online: https://tools.ietf.org/html/rfc3281 (accessed on 17 February 2020).
30. Ellison, C.; Frantz, B.; Lampson, B.; Rivest, R.; Thomas, B.; Ylonen, T. RFC 2693: SPKI Certificate Theory. The Internet Society. 1999. Available online: https://tools.ietf.org/html/rfc2693 (accessed on 17 February 2020).
31. Gutmann, P. Simplifying public key management. *Computer* **2004**, *37*, 101–103. [CrossRef]
32. Charette, R. DigiNotar certificate authority breach crashes e-Government in The Netherlands. IEEE Spectrum. Available online: https://spectrum.ieee.org/riskfactor/telecom/security/diginotar-certificate-authority-breach-crashes-egovernment-in-the-netherlands (accessed on 17 February 2020).
33. Lewko, A.; Waters, B. Decentralizing attribute-based encryption. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, 15–19 May 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 568–588.
34. Okamoto, T.; Katsuyuki, T. Decentralized attribute-based signatures. In Proceedings of the International Workshop on Public Key Cryptography, Nara, Japan, 26 February–1 March 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 125–142.
35. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: http://bitcoin.org/bitcoin.pdf (accessed on 17 February 2020).
36. Reyna, A.; Martín, C.; Chen, J. Enrique Soler, and Manuel Díaz. On blockchain and its integration with IoT. Challenges and opportunities. *Future Gener. Comput. Syst.* **2018**, *88*, 173–190. [CrossRef]
37. Sifah, E.B.; Xia, Q.; Agyekum, K.O.-B.O.; Amofa, S.; Gao, J.; Chen, R.; Xia, H.; Gee, J.C.; Du, X.; Guizani, M. Chain-based big data access control infrastructure. *J. Supercomput.* **2018**, *74*, 4945–4964. [CrossRef]
38. Merkle, R.C. Protocols for Public Key Cryptosystems. In Proceedings of the 1980 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 14–16 April 1980; p. 122.
39. Vanstone, S. Responses to NIST's Proposal. *Commun. ACM* **1992**, *35*, 50–52.
40. National Institute of Standards and Technology. *Digital Signature Standard*; FIPS Publication: Gaithersburg, MD, USA, 1994; p. 186.
41. Koblitz, N. Elliptic curve cryptosystems. *Math. Comput.* **1987**, *48*, 203–209. [CrossRef]
42. Miller, V. Uses of elliptic curves in cryptography. In *Advances in Cryptology—Crypto '85*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1986; Volume 218, pp. 417–426.
43. Wood, G. Ethereum: A Secure Decentralised Generalised Transaction Ledger. Available online: https://gavwood.com/paper.pdf (accessed on 17 February 2020).
44. Dorri, A.; Kanhere, S.S.; Jurdak, R.; Gauravaram, P. LSB: A Lightweight Scalable Blockchain for IoT security and anonymity. *J. Parall. Distribut. Comput.* **2019**, *134*, 180–197. [CrossRef]