proceedings

MDPI

Proceedings

Blockchain Technologies for Private Data Management in AmI Environments †

Tomás Robles [1], Borja Bordel [1], Ramón Alcarria [2,*] and Diego Sánchez-de-Rivera [1]

[1] Department of Telematics Systems Engineering, Universidad Politécnica de Madrid, Madrid 28040, Spain; tomas.robles@upm.es (T.R.); bbordel@dit.upm.es (B.B.); diego.sanchezderivera@upm.es (D.S.d.R.)
[2] Department of Geospatial Engineering, Universidad Politécnica de Madrid, Madrid 28031, Spain
* Correspondence: ramon.alcarria@upm.es; Tel.: +34-91-06-73922
† Presented at the 12th International Conference on Ubiquitous Computing and Ambient Intelligence (UCAmI 2018), Punta Cana, Dominican Republic, 4–7 December 2018.

Published: 19 October 2018

Abstract: Blockchain enables the creation of distributed ledgers as a type of database that is shared, replicated, and synchronized among the members of a network. In this paper we analyze how distributed ledgers can be used for empowering end-users to self-manage their own data, enabling third parties to access those data under a cryptographic management model. We propose a use case where both blockchain and smart contracts are employed by using cryptographic technology to enable user empowerment of data management in AmI. Finally, we analyze strengths and weaknesses of the proposed scenario.

Keywords: blockchain; AmI; data management; user empowerment

1. Introduction

In the Big Data era, the amount of data is continuously increasing [1], from user's behavior [2] to other personal data as a new key economical resource [3]. While there are a lot of clear benefits concerning the use of Big Data technologies and the data themselves as a key resource for economy, there is a growing concern about user's privacy and the misuse of user's information [4]. Organizations collect large amounts of personal and sensitive data [5]. Specifically, Ambient Intelligence (AmI) environments require such information for providing context-aware [6] and other personalized services to optimize corporate decisions.

Individuals have little control over the data about them that is stored and how it is used [5]. In recent years, several controversial incidents related to privacy have been discussed on public media. Some of the known examples are government surveillance [7], and large-scale usage of Facebook's data used for influencing public opinion [4], and publishing location data without suitable controls [8].

In this context *blockchain* technology emerged as one promising technology for enabling a suitable data management using the concept of public ledger updated independently by each participant in the network. In this paper we explore how to combine the building characteristics of blockchain technology to define, create and deploy a user-centric private data management system in AmI environments, providing a data management preliminary solution in Ethereum and an abstract model for authorization of IoT devices and third parties to data. In particular, we contribute to the following objectives:

- Definition of mechanism for user empowerment on data management scenarios, by deploying some mechanisms to store data provided by IoT devices in AmI environments, and to enable the end-user to generate and to revoke data usage authorization for others.

- Definition of secure procedures in blockchain for storing and reading data for IoT devices. Data stored on blockchain network will be provided and accessed by IoT devices with the corresponding credentials, to avoid unauthorized usage of such data storage.

The rest of the paper is organized as follows: Section 2 introduces *blockchain* technology and requirements for private data management in AmI environment; Section 3 provides a security management scenario for AmI environments using blockchain and smart contracts; Section 4 provides an implementation for the data management functionality as a smart contract; Section 5 proposes an abstract security management model and finally Section 6 analyses the proposed solutions and describes future developments.

2. Blockchain Technology and Data Management in AmI

In this section we analyze two main technologies for data management using blockchain technologies. First, we review the facilities offered by blockchain technologies and finally we go through technologies and requirements related to data management in AmI environments.

2.1. Blockchain Technologies

Blockchain emerged as a new technology with the success of cryptocurrencies [9]. A blockchain is a tamper-evident, shared digital ledger that records information (transactions and data) in a public or private peer-to-peer system. Distributed to all member nodes in the system, the ledger records, in an immutable sequential chain of cryptographic hash-linked blocks, the history of asset exchanged that take place between the peers (users, programs or contracts) in the network, preserving the validated list of transactions and the resulting data values associated to these transactions.

All the confirmed and validated transaction blocks are linked and chained from the beginning of the chain to the most current block, using the proof of work as the consensus mechanism for preserving the chain integrity. *Blockchain* offers several key characteristics that can be used for data management in AmI environment [10]:

Data storage: blockchain packs data and transactions in a single structure. A block maintains the record of all transactions occurred across the system. It prevails as an immutable information in the blockchain.

Most relevant benefits of blockchain are fraud protection (prevents the unauthorized changes or malicious tampering, as changes are not possible), easy management (all accepted transactions can be found in the network in a sequential way), ownership (security is maintained by the means of *keys* and *signatures*), and lack of mediators (decreasing the settlement times of many transactions and speeding up the process).

Data Distribution: the owner has direct control of the data by using his/her private key. The key enables owner authentication and the owner can provide access rights to the data to whomever he wants.

Most relevant benefits of blockchain regarding data distribution are fault tolerance and attack resistance (decentralized networks are less likely to fail because of their lack of single point of failure), lower transaction costs (by the elimination of intermediaries and overhead costs for exchanging assets) and transparency and auditability (as all transactions are immutable and public).

Data integrity and authenticity: Anyone trying to alter a transaction stored inside a block would need an almost impossible amount of resources (memory, computing power and bandwidth) to alter sufficient number of nodes. Thus, the system cannot be easily corrupted. Guaranteeing the authenticity of data is very important; organizations and individuals have sensitive documents, assets or contracts that must be protected [11]. When data, in any from, is stored, a hash is created for this information piece. This mechanism that works for a single piece of information (data, file, document, etc.) can be extended to a large amount of data using technologies such as the Merkle tree or hash tree.

Smart contracts: Smart contracts [12] are cryptographic “boxes” containing programming logic that is executed if certain conditions are met. They offer more power than Bitcoin scripting because

of the added powers of Turing-completeness, value-awareness, blockchain-awareness and state [13]. Smart contracts can function as 'multi-signature' accounts, so that funds are spent only when a required percentage of people agree, and also can store data and provide utility to other contracts (similar to how a software library works).

2.2. Data Management in AmI Environments

Data management is a key element in any AmI architecture [14]. A central element regarding data management in these scenarios is privacy, as the right of every person to control access to his/her own personal information [15].

Jøsang et al. [16] explain that the fundamental privacy protection principle is that exposure of personal information should be minimized. Translating this concept to personal data protection, this means that the fewer parties involved in the management of the identity information the better.

Possible weakness in security of a wireless system should be recognized so that the right measures can be taken to improve the user's confidence. AmI systems require data privacy, security, and physical security. A very small carelessness in the AmI security could really have a big impact to everyone involved [16]. Problems like authorization, authentication, and accounting are important while considering the data security. Different devices and standards for communications should be studied properly. Security requirements in AmI systems are studied from the data integrity, authentication and confidentiality point of view.

Data integrity requirement should ensure that the transmitted data from source to destination is unaltered by any means. The data could be intercepted in transit and can be modified [17]. Therefore, data integrity checks (by fingerprinting) should be performed so that the receiver could confirm that the data is not altered.

Data authentication is the process ensuring that the provider is truly the provider of the data [17,18]. The provider should be authenticated so that the attacker pretending to be the sender would not be able to fake the communication.

Data confidentiality is the process of hiding the information so that only the recipients could know of what's being provided by the provider. It can be achieved by using the data encryption algorithms defined in symmetric [16,18] (shared common key) and asymmetric (public-private key pairs) models.

3. Security Management Scenario for User Data in AmI Environments

In this section we present a security management scenario for user data in AmI Environments. The scenario proposed in Figure 1 integrates several actors: end-users, IoT devices controlled by the user, the third parties willing to access the private information provided by IoT devices, all around the blockchain infrastructure that is going to provide secure access to the data controlled by end-users.

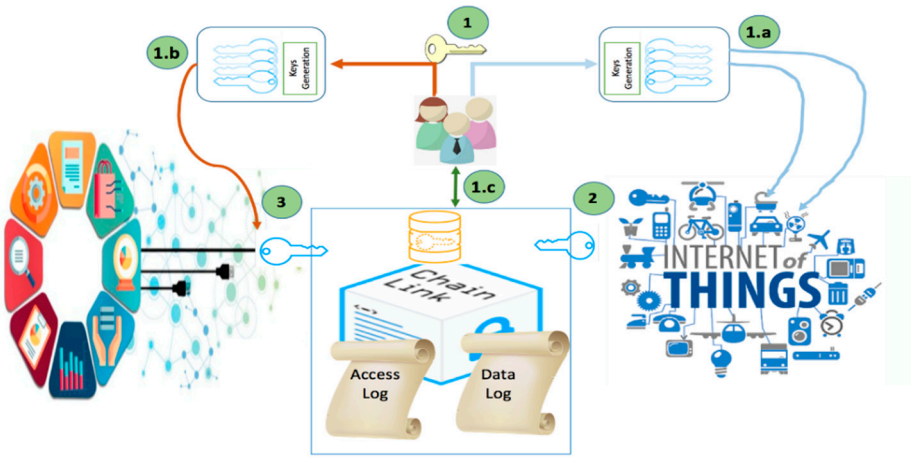

Figure 1. Security management scenario for user data in AmI environments.

In this scenario users manage their own security keys (1), so they can generate suitable credentials, using cryptographic techniques, to be provided to the IoT devices (1.a), and they also are going to generate credentials (1.b) for third parties in order to enable them to access data of the user under strict conditions. Both credentials should be translated to the blockchain infrastructure so they can enforce the application of the rules defined by the end-user regarding data storage and data access. Cryptographic techniques suitable for supporting this are under development. A security model for such proposal is presented in Section 5.

IoT devices, by using security keys, are able to store information into the blockchain (2). End-users feed the blockchain with suitable information, then the smart contract can apply security restrictions to store and retrieve data from the blockchain, while it provides a full *data log*. Third parties requesting end-user authorization for using private data can access those data stored in the blockchain (3). In this case blockchain is going to provide *access log* based on blockchain characteristics.

4. Implementing Data Management on Blockchain

In this section we present an implementation for the data management functionality in blockchain. We use Ethereum, an open software platform based on blockchain technology that enables developers to build and deploy decentralized applications. Ethereum allows us the implementation of smart contracts, written in Solidity, a contract-oriented programming language.

We provide an implementation of a data management procedure to handle user's private data in Ethereum, by using the smart contract capabilities.

The smart contract of Figure 2 identifies several key elements:

The first line (1) of the contract defines the version of Solidity used for coding this smart contract.

This defines the name of the smart contract as the name of the class in most object-oriented programming languages.

As one of the attributes of the class we can define data structures (5). In this case we define an array of uint (a short of integer) values for defining the data storage structure.

Another attribute (7) is defined for storing the address of authorized persons for using this contract. In this case for the sake of simplicity only the address of the owner is included in this list of authorized people, although a function can be added to include and remove authorized persons.

The function to generate events to be sent out of the contract is defined (9). Events are the natural way for sending information to users and other external applications.

One specific method (11) is used for checking authorization. In this simple case we checked that the sender of a request is included in the list of authorized users defined at Point 4.

We define the constructor of the smart contract (19).

Function in line 23 enables a data reading. The data of the position pos is provided to anybody that requests the data. In this case no filter is applied.

Function in line 28 enables the storage of a data. A new data is stored in the next position of the array data. In this case only users meeting the requirements imposed by authorized function can use this function (in this example only the owner of the contract).

As a summary, this smart contract defines a data storage structure, with a list of authorized addresses (i.e., external users) that can use the functions *getData* and *setData* of this contract. An *authorized* function is defined for evaluating the access right of each invocation to these functions. This simple contract provides some solutions to common problems in AmI environments regarding data management, such as integrity (data storage structure in unaltered), authentication (the sender of a request must be included in the list of authorized users) and confidentiality (smart contract providers and consumers are only recognized by their public key).

```
pragma solidity ^ 0.4.23;

contract UsersData {

    uint[] data; //Array if stored data. Data model not specified

    address owner; //Contract owner

    event Request (string msg, address who);

    modifier onlyOwner {
        require(
            msg.sender == owner //"Only owner can call this function."

        );
        _;
    }

    constructor() public {
        owner = msg.sender;
    }

    function getData (uint pos) public returns (uint) {
        emit Request ("You've got a data", msg.sender);
        return data[pos];
    }

    function setData (uint newData) public onlyOwner returns(string) {
        data.push(newData);
        return "You sucessfully changed the data";
    }

}
```

Figure 2. Smart contract for data management.

5. An Abstract Model for Authorization of IoT Devices and Third Parties to Access a Smart Contract

In this section we present a general cryptographic model that can be used for generating credentials to enable IoT devices and third parties to store and to read data from the smart contract managed by the end-user. This model can be implemented for generating the credential, providing suitable information to the smart contract to enable them through the modifiers function to filter the access to the data repository.

In the proposed scenario, where keys must be distributed among an undefined (and probably unlimited) number of entities (including devices, third party users and stakeholders), encryption schemes based on symmetric keys have been proved to be highly inefficient and costly. Public Key Infrastructures (PKI), on the other hand, can be the answer to these problems, but the identity assurance of the key owner then becomes a new issue to solve.

Nonetheless, the proposed technique describes a high-level application which should be supported by existing communication technologies, so among all components in the scenario it is supposed a robust, reliable and highly available communication infrastructure. Therefore, problems associated to the private information protection during transmissions and to mechanisms to guarantee the identity of the different entities exchanging keys (which are addressed at link, network and transport level) are assumed to be resolved by the underlying communication infrastructure.

By default, blockchain networks are transparent and all data stored in the smart contracts are accessible by every user in the system. The use of these networks, however, enables all the community sharing information to maintain an agreed and unmodifiable record about the accesses and transactions made over these data. Nevertheless, organizations and owners (usually) need to stablish controls about who can obtain their information or data (systems must be permissioned) [19]. Therefore, incorporating a certain level of privacy in blockchain is required.

In this first work and use case, AmI data are stored in the smart contracts, but in future and advances works this information could be maintained in a database whose access in controlled by a smart contract. In that way, a record about the transactions involving the AmI data will be created, but information could be modified more dynamically and efficiently than in solution using only

blockchain technologies. In conclusion, smart contracts only should manage the authorization of users to obtain data, which is the solution to be developed in this section.

In our application scenario, smart contracts, the blockchain network, third-party entities and sensitization devices typically communicate using TCP/IP technologies and REST-like petitions (a REpresentational State Transfer architecture). Thus, for all elements it is possible to stablish a secure communication channel with any other component in the system through protocols such as Hypertext Transport Protocol Secure (HTTPS), which includes solution for both privacy information protection and identity certification.

Therefore, in this section we are only discussing about the "intrusion tolerance" and "active protection" policies at application level. In other words, we are only describing a solution to guarantee that only authorized entities are allowed to communicate (though the described secure channels) with the proposed deployment.

Various special characteristics of blockchain networks and smart contract must be considered:

- First, all attributes, parameters and variables in a smart contract are public. In that way, any user, program or device may access to any information about authorized entities that is stored in the smart contract.
- Besides, sensitization devices are usually resource constraint and cannot perform complex algorithms. On the other hand, devices are also typically unmanaged and geographically sparse, so it is not safe to store information enough to break the security of the system in the ROM memory of these devices (as it could be physically accessed by unauthorized people) [20].
- Finally, in PKI, keys cannot be sent through public networks (even if secure channels are stablished), except if a piece of information is not transmitted to maintain the security of the entire solution.

In general, a final user U who deploys a smart contract to manage information in AmI environments, employs an original and key k_u. Then, it is proposed a digital signature scheme (DSS) described by the 3-tuple $(\mathcal{G}_{sig}, \mathcal{S}_{sig}, \mathcal{V}_{sig})$. The first element is a key generator, the second one is the signature function, and the third one is the verification procedure. From this DSS we only require to be homomorphic.

Homomorphic encryption is a special encryption type where applying any algebraic operation on the original information is equivalent to apply also an algebraic operation (not necessary the same) on the encrypted information (1). Many existing DSS schemes and encryptions algorithms are homomorphic (such ElGamal technique [21]).

$$\mathcal{S}_{sig}(m_1 \circledast m_2) = \mathcal{S}_{sig}(m_1) \odot \mathcal{S}_{sig}(m_2) \quad (1)$$

being $\circledast$,$\odot$ algebraic operations.

The proposed security model is showed on Figure 3. As can be seen, using the signature generator, the private user key k_u and a n-tuple of additional parameters to increase the generator entropy, it is constructed a user operation key k_u^*. This user operation key is a 2-tuple, including (as any PKI) two different keys: a private user operation key k_{u-pv}^* and a public user operation key k_{u-pb}^* (2).

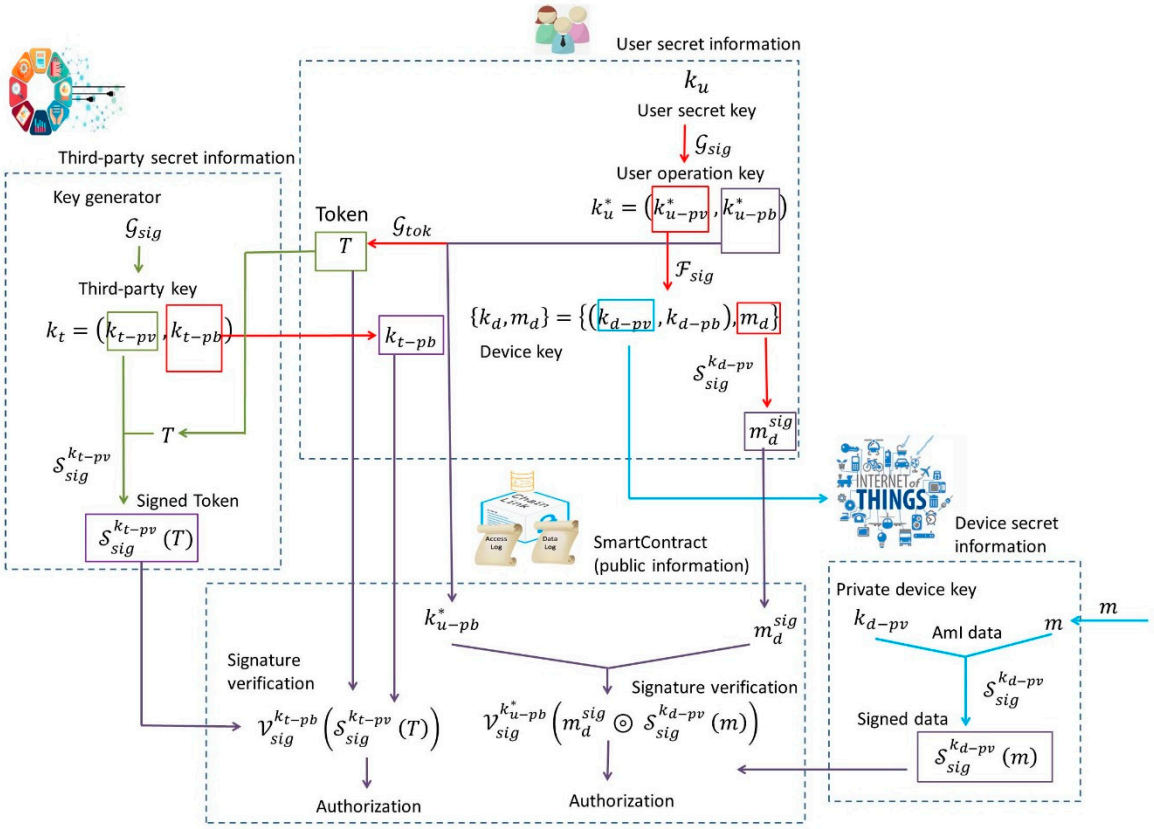

Figure 3. Security model.

$$\mathcal{G}_{sig}(k_u, \{p_1, \ldots, p_n\}) = k_u^* = \left(k_{u-pv}^*, k_{u-pb}^*\right) \tag{2}$$

In order to authorize an AmI device to interact with a deployed smart contract, a device key k_d is constructed. This device key is generated using a second generator $\mathcal{F}_{sig}$ that presents vector structure (3). It receives as parameters the private user operation key k_{u-pv}^* and a n-tuple of additional parameters to increase the generator entropy. As result, the generator produces the device key and a piece of information m_d, which is unique for each device, and is called "homomorphic constant".

$$\mathcal{F}_{sig}\left(k_{u-pv}^*, \{r_1, \ldots, r_n\}\right) = \{k_d, m_d\} = \left\{\left(k_{d-pv}, k_{d-pb}\right), m_d\right\} \tag{3}$$

As in the previous case, the device key is composed of a public device key k_{d-pv} and a private device key k_{d-pb}. In this proposal, it is required the private device key, the private user operation key and the homomorphic constant to be related through an equivalence in the digital signature algorithm of message m (4). Although this relation may be difficult to fulfil, it must be noted that employed private keys during the digital signature process are different; thus, both signature functions are slightly different and the homomorphic can be obtained and found even if collision-resistant hash functions are used during the signature procedure. The resistance to the birthday attack, any case, is preserved. Any case, the calculation of this constant may be costly, so it is quite impossible to obtain several of them in a row to attack an entire AmI system.

$$s_{sig}^{k_{u-pv}^*}(m) = s_{sig}^{k_{d-pv}}(m_d \circledast m) \tag{4}$$

being $\circledast$ an algebraic operation.

At this point, and considering that the selected encryption scheme is homomorphic, it is possible to rewrite the signature operation of message m in terms of a new parameter that is unique for each

AmI device, m_d^{sig}; and that corresponds to the signed homomorphic constant using the private device key (5).

$$\mathcal{S}_{sig}^{k_{u-pv}^*}(m) = \mathcal{S}_{sig}^{k_{d-pv}}(m_d \circledast m) = \mathcal{S}_{sig}^{k_{d-pv}}(m_d) \odot \mathcal{S}_{sig}^{k_{d-pv}}(m) = \\ = m_d^{sig} \odot \mathcal{S}_{sig}^{k_{d-pv}}(m) \tag{5}$$

being $\circledast$,$\odot$ algebraic operations

Then, in the smart contract (blockchain network) only two parameters are stored: the public user operation key k_{u-pb}^*, and the m_d^{sig} parameter for each authorized device. On the other hand, to the authorized device, it is only sent the private device key k_{d-pv}. In order to the smart contract to accept a data from an AmI device, it must send the corresponding information m together with the signature $\mathcal{S}_{sig}^{k_{d-pv}}(m)$. Then, in the blockchain network, and using the properties of homomorphic encryption, the signature is verified using only data stored in the smart contract (6). Only if the AmI device employs a valid key during the digital signature procedure, the signature could be validated using the public user operation key and parameter m_d^{sig}. In that way, if signature is verified, the received information is accepted as the remote device is authorized to store data.

$$\mathcal{V}_{sig}^{k_{u-pb}^*}\left(\mathcal{S}_{sig}^{k_{u-pv}^*}(m)\right) = \mathcal{V}_{sig}^{k_{u-pb}^*}\left(m_d^{sig} \odot \mathcal{S}_{sig}^{k_{d-pv}}(m)\right) \tag{6}$$

This approach presents several important advantages. First, the smart contract does not store any private key or device key, which would be publicly accessible. Besides, as the authorization procedure is based on the signature of the collected data, pattern recognition techniques or statistical attacks are not successful as hash functions are not invertible. On the other hand, devices do not store a complete key, only the private key of a KPI, so some types of cyberattacks (such as Denial of Service) are also impossible, as nobody can encrypt a message using the private device key which is protected by the user. As keys are generated by the user, strong long-length keys could be employed, as resource constraint devices are not a problem to employ complex key generators.

Algorithm 1 shows the entire authorization mechanism for AmI devices.

Algorithm 1 Authorization mechanism for AmI devices

user executes:

$(k_{u-pv}^*, k_{u-pb}^*) \leftarrow \mathcal{G}_{sig}()$

$\{(k_{d-pv}, k_{d-pb}), m_d\} \leftarrow \mathcal{F}_{sig}()$

$m_d^{sig} \leftarrow \mathcal{S}_{sig}^{k_{d-pv}}(m_d)$

user stores k_{u-pb}^*, m_d^{sig} in the smart contract

user sends k_{d-pv} to AmI device

AmI device executes:

AmI device collects information m

$s \leftarrow \mathcal{S}_{sig}^{k_{d-pv}}(m)$

AmI device sends m, s to the smart contract

smart contract executes:

$auth \leftarrow \mathcal{V}_{sig}^{k_{u-pb}^*}\left(m_d^{sig} \odot \mathcal{S}_{sig}^{k_{d-pv}}(m)\right)$

if auth **then**

m is stored

end if

Finally, we must discuss how to authorize third-party entities to access to the data stored in the smart contract. The same strategy previously presented for AmI devices could be also employed, if desired. However, in this case, as these entities will be remote, keys are not recommended to be

directly computed by the user. Instead of that a token-based solution is proposed. The user will produce for each third-party a token T using a specific token generator $\mathcal{G}_{tok}$ which receives as parameters the public user operation key and a n-tuple of additional parameters to increase the generator entropy (7). The token is sent to the authorized third-party.

$$\mathcal{G}_{tok}\left(k^{*}_{u-pb}, \{q_1, \ldots, q_n\}\right) = T \tag{7}$$

On the other hand, using the key generator, the third-party creates a new key k_t named as "third-party key" (8). The user receives, then, from the third-party the public third-party key k_{t-pb}, and stores it (with the corresponding token) in the smart contract.

$$\mathcal{G}_{sig}(\{s_1, \ldots, s_n\}) = k_t = \left(k_{t-pv}, k_{t-pb}\right) \tag{8}$$

Then, to access to certain data in the smart contract, the third-party should send (with the data request) the authorization token signed with the private third-party key k_{t-pv}. If this signature is verified, then, the third-party is authorized to access to data (9).

$$\mathcal{V}_{sig}^{k_{t-pb}}\left(\mathcal{S}_{sig}^{k_{t-pv}}(T)\right) \tag{9}$$

Algorithm 2 shows the entire authorization mechanism for third-party. As can be seen, in this algorithm only required information to manage privacy and user authorizations is stored in smart contracts. The storage and recuperation of AmI data should be developed in a different algorithm. If data are stored in blockchain networks, they should be encrypted before being written, in order to preserve their privacy.

Algorithm 2 Authorization mechanism for third-party

User executes:
- $T \leftarrow \mathcal{G}_{tok}()$
- user waits for the public third-party key k_{t-pb}
- user receives the public third-party key k_{t-pb}
- user stores T, k_{t-pb} in the smart contract

Third-party executes:
- $\left(k_{t-pv}, k_{t-pb}\right) \leftarrow \mathcal{G}_{sig}()$
- Third-party sends k_{t-pb} to user
- $T_{sig} \leftarrow \mathcal{S}_{sig}^{k_{t-pv}}(T)$
- Third-party sends T, T_{sig} to the smart contract

Smart contract executes:
- $auth \leftarrow \mathcal{V}_{sig}^{k_{t-pb}}(T_{sig})$
- **if** auth **then**
  - m is stored
- **end if**

6. Conclusions and Future Work

In this paper we analyzed the problems related to data management by end-users in AmI environments and developed a user-centric private data management infrastructure.

Based on the analysis of blockchain, smart contracts and AmI environments, we designed one scenario where end-users can use their own secure keys for deriving cryptographic information that can be translated to both IoT devices and third parties. Using this cryptographic information, which is also managed by the smart contract, the IoT devices and third-parties are able to use the data repository according to explicit permissions granted by the end-users. Those permissions can be changed dynamically, when required, by the end-user.

The proposed contract is only an example of how to provide some solutions to common problems in AmI environments regarding data management, such as integrity (data storage structure

in unaltered), authentication (the sender of a request must be included in the list of authorized users) and confidentiality (smart contract providers and consumers are only recognized by their public key).

Although the proposed smart contract fulfills many conditions of our scenario, it offers some relevant drawbacks, based on blockchain's basic design principles: every data stored in the smart contract is publicly readable when it appears in a public transaction, the cost of storing data is very high, and the list of authorized parties is also publicly available. For solving these limitations, we provide a security approach for user authorization so that smart contracts do not store any private key or device key avoiding security problems related to the public availability of the information.

We can conclude from the initial work presented in this paper, that blockchain technology combined with smart contract enable the end-users to own and manage the data generated by IoT devices in AmI environment. However, some problems arise in the provision of a public blockchain infrastructure such as the one provided by Ethereum. The main problem is that public networks do not provide means to effectively restrict read access to data, as all persistent data in a contract is readable to anyone even without the provision of access functions in Solidity. Thus, it is still required to deploy complex cryptographic solutions for avoiding problems of transparency related to blockchain and to link the blockchain with external devices in order to automatically store information in the system. Therefore, external data repositories are required to create hybrid solutions where benefits of blockchain networks (traceability of data transactions) are combined with the efficient and private access and storage of data. Other contributions related to these limitations, such as external storage capabilities and integrity (Merkle tree [22]) and data anonymity (Kademilia [23]) techniques will be studied in future works.

Author Contributions: The contributions described in this work are distributed among the authors in the way that follows: T.R. proposed and developed the paper's idea, B.B. proposed the security model, R.A. contributed to the theoretical formalization and paper redaction, and D.S. implemented smart contract for data management function.

Funding: Borja Bordel has received funding from the Ministry of Education through the FPU program (grant number FPU15/03977). Additionally, the research leading to these results has received funding from the Ministry of Economy and Competitive-ness through SEMOLA project (TEC2015-68284-R) and from the Autonomous Region of Madrid through MOSI-AGIL-CM project (grant P2013/ICE-3019, co-funded by EU Structural Funds FSE and FEDER).

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

References

1. Carron, C.; Morris, A.; Percival, R. Big data and the internet of things. In *Magazine Motion 2016 of Norton Rose Flubright*; Norton Rose Flubright: London, UK, 2016.
2. Schroepfer, M. Chief Technology Officer of Facebook. An Update on Our Plans to Restrict Data Access on Facebook. Available online: https://newsroom.fb.com/news/2018/04/restricting-data-access/ (accessed on 4th April 2018).
3. Schwab, K.; Marcus, A.; Oyola, J.O.; Hoffman, W.; Luzi, M. Personal data: The emergence of a new asset class. In *An Initiative of the World Economic Forum*; World Economic Forum: Cologny, Switzerland, 2011.
4. The Verge. The Cambridge Analytical Scandal, Understanding Facebook's Data Privacy Debacle. Available online: https://www.theverge.com/2018/4/10/17165130/facebook-cambridge-analytica-scandal (accessed on 16th October 2018).
5. Martin, K.D.; Borah, A.; Palmatier, R.W. Data Privacy: Effects on Customer and Firm Performance. *J. Mark.* **2017**, *81*, 36–58.
6. Acampora, G.; Cook, D.J.; Rashidi, P.; Vasilakos, A.V. A Survey on Ambient Intelligence in Health Care. *Proc. IEEE* **2013**, *101*, 2470–2494, doi:10.1109/JPROC.2013.2262913.
7. Ball, J. Nsa's prism surveillance program: How it works and what it can do. *The Guardian*, 2013. Available online: https://www.theguardian.com/world/2013/jun/08/nsa-prism-server-collection-facebook-google (accessed on 16th October 2018).
8. Hern, A. Fitness Tracking App Strava Gives Away Location of Secret US Army Bases. In BBC News. Available online: https://www.bbc.com/news/technology-42853072 (accessed on 16th October 2018).

9. Narayanan, A.; Bonneau, J.; Felten, E.; Miller, A.; Goldfeder, S. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*; Princeton University Press: Princeton, NJ, USA, 2016.
10. Techracers, "4 Key Features of Blockchain". Available online: http://www.techracers.com/blockchain-key-features (accessed on 16th October 2018).
11. Tapscott, D.; Tapscott, A. *Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World*; Penguin Publishing Group: London, United Kingdom, 2017.
12. Ethereum Community. *Ethereum Homestead Documentation*; Release 0.1. Available online: http://www.ethdocs.org/en/latest/ (accessed on 16 October 2018).
13. Dameron, M. Beigepaper: An Ethereum Technical Specification. Available online: https://github.com/chronaeon/beigepaper/ (accessed on 16 October 2018).
14. Murabet, A.E.; Abtoy, A.; Touhafi, A.; Tahiri, A. Ambient Assisted living system's models and architectures: A survey of the state of the art. *J. King Saud Univ. Comput. Inf. Sci.* **2018**, doi:10.1016/j.jksuci.2018.04.009.
15. Dey, N.; Ashour, A.S.; Amira, S. Ambient Intelligence in Healthcare: A State-of-the- Art. *Glob. J. Comput. Sci. Technol.* **2017**, *17*, 19–28.
16. Jøsang, A.; Zomai, M.A.; Suriadi, S. Usability and Privacy in Identity Management Architectures. In Proceedings of the Fifth Australasian Symposium on ACSW Frontiers, Ballarat, Australia, 30 January–2 February 2007; pp. 143–152.
17. Qu, H.; Cheng, J.; Cheng, Q.; Wang, L.Y. WiFi-Based Telemedicine System: Signal Accuracy and Security. *Int. Conf. Comput. Sci. Eng.* **2009**, *2*, 1081–1085.
18. Adekunle, A.A.; Woodhead, S.R. On Efficient Data Integrity and Data Origin Authentication for Wireless Sensor Networks Utilising Block Cipher Design Techniques. In Proceedings of the Third International Conference on in Next Generation Mobile Applications, Services and Technologies (NGMAST '09), Cardiff, Wales, UK, 15–18 September 2009; pp. 419–424.
19. Alcarria, R.; Bordel, B.; Martín, D.; De Rivera, D.S. Rule-based monitoring and coordination of resource consumption in smart communities. *IEEE Trans. Consum. Electron.* **2017**, *63*, 191–199.
20. Mareca, M.P.; Bordel, B. Improving the complexity of the Lorenz dynamics. *Complexity* **2017**, *2017*, doi:10.1155/2017/3204073.
21. Schnorr, C.P.; Jakobsson, M. Security of signed ElGamal encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*; Springer: Berlin/Heidelberg, Germany, 2000; pp. 73–89.
22. Merkle, R.C. A Digital Signature Based on a Conventional Encryption Function. Advances in Cryptology—CRYPTO '87. *Lect. Notes Comput. Sci.* **1988**, *293*, 369.
23. Maymounkov, P.; Mazieres, D. Kademlia: A peer-to-peer information system based on the xor metric. In *Peer-to-Peer Systems*; Springer: Berlin, Germany, 2002; pp. 53–65.