Towards a Novel Privacy-Preserving Access Control Model Based on Blockchain Technology in IoT

Aafaf Ouaddah, Anas Abou Elkalam and Abdellah Ait Ouahman

Abstract Access control face big challenges in IoT. Unfortunately, it is hard to implement current access control standards on smart object due to its constrained nature while the introduction of powerful and trusted third party to handle access control logic could harm user privacy. In this work we show how blockchain, the promising technology behind Bitcoin, can be very attractive to face those arising challenges. We therefore propose FairAccess as a new decentralized pseudonymous and privacy preserving authorization management framework that leverages the consistency of blockchain technology to manage access control on behalf of constrained devices.

Keywords Internet of things · Security · Privacy · Access control · Blockchain · Bitcoin · Cryptocurrency

1 Introduction

We believe that the concept of a distributed IoT is a promising approach to release [1]. As devices increase their computational capacity, there are more opportunities to bring intelligence, mainly security and access control logic, on devices themselves. Actually, with this edge intelligence principle, users have more control over the granularity of the data they produce. However, as side effect of this approach, end-users are not expected to be experts to use security mechanisms. A simple mistake or a misconfiguration can lead to huge breaches in their privacy. For this reason, access control, mainly within the decentralized approach, have to be enough usable for ordinary people. Furthermore, the decentralized approach faces the following challenges: implementing current security standards and access control solutions on the device's side is more complicated. It requires intensive and

A. Ouaddah (✉) · A.A. Elkalam · A.A. Ouahman
OSCARS Laboratory, ENSA of Marrakesh, Cadi Ayyad University,
BP 575, 40000 Marrakesh, Morocco
e-mail: aafafouaddah@gmail.com

Á. Rocha et al. (eds.), *Europe and MENA Cooperation Advances in Information and Communication Technologies*, Advances in Intelligent Systems and Computing 520, DOI 10.1007/978-3-319-46568-5_53

computational capabilities which is not always available, especially in devices like sensors, actuators or RFID tags etc. While, relieving those devices from the burden of handling a vast amount of access control-related information by outsourcing these functionalities to a powerful entity prevents end-to-end security to be achieved. In addition, delegating the authorization logic to an external service requires a strong trust relationship between the delegated entity and the device. Moreover, all communications between them must be secured and mutually authenticated, so that the delegated entity security level is at least as high as if the authorization logic were implemented internally. Hence, we believe that IoT needs a new access control framework suitable to its distributed nature, where users may control their own privacy and, rather than being controlled by a centralized authority, and at the same time, the need arises for centralized entity handling authorization function to hardly constrained IoT devices. Then, the goal of this paper is to introduce FairAccess framework as a balance solution and equilibrium that solve the dilemma of centralized and decentralized access control management challenges highlighted above by leveraging the block chain technology.

Contribution: we introduce FairAccess as a novel Distributed Privacy Preserving Access Control framework in IoT scenario that combines, for the first time, access control models and cryptocurrency blockchain mechanisms. In FairAccess, we propose the use of SmartContract [2] to express fine-grained and contextual access control policies to make authorization decisions. We opt for authorization tokens as access control mechanism, delivered through emergent cryptocurrency solutions. We use blockchain firstly to ensure evaluating access policies in distributed environments where there is no central authority/administrator, and guarantee that policies will be properly enforced by all interacting entities and secondly to ensure token reuse detection.

Organization: The rest of this paper is structured as follows: in Sect. 2, we review related work and discuss the benefit of a decentralized peer-to- peer architecture. In Sect. 3, we show how blockchain can be used in distributed and transparent access control. We then introduce our proposed framework in Sect. 4 and finally Sect. 5 concludes the paper.

2 Related Work

In one hand, numerous efforts have emerged in adapting traditional access control model such as The Role Based Access Control (RBAC) model [3] that was extended to a new model named context based access control by the introduction of context which is provided by the web service. In this model the permission is assigned to the role according to the characteristics and contextual information collected from the environment of the physical object however its feasibility in constrained devices has not been demonstrated. The Capability-based access control model (CapBAC) was also chosen in [4] where it was directly implemented on resource-constrained devices, within a fully distributed security approach but the

model introduced was coarse grained and not user-driven. Another generic Authorization Framework for the Internet-of-Things is proposed in [5]. It supports fine-grained and flexible access control for any objects with low power and memory resources. Based on current Internet standards and access control solutions such as XACML and Security Assertion Markup Language (SAML). But it introduces a Trusted Third Party as an authorization engine to handle access control logics.

In other hand, across the industry, many companies implement their own proprietary authorization software based on the OAuth protocol [6], in which they serve as centralized trusted authorities. For instance, EU project CALIPSO [7] adopts a centralized approach where the authorization logic is outsourced from the smart and constrained device to a more powerful server called IoT-OAS. However, it has demonstrate in [8] the impossibility to run all OAuth logic in a constrained device due to its heavy communication and processing overheads.

Unfortunately, those typical security and access control standards today are built around the notion of trust where a centralized trusted entity is always introduced. However, significant drawbacks arise when centralized approaches are considered on a real IoT deployment. On one hand, the inclusion of a central entity for each access request clearly compromises end-to-end security properties. On the other hand, the dynamic nature of IoT scenarios with a potential huge amount of devices complicates the trust management with the central entity, affecting scalability. In addition, they are built around a single logical server and multiple clients. As a consequence, access control is often done within the server side application, once the client has been authenticated. IoT reverses this paradigm by having many devices serving as servers and possibly many clients, taking part in the same application. More importantly, servers are significantly resource-constrained, which results in the minimization of the server side functionality. Subsequently, access control becomes a distributed problem.

We therefore turn our attention to blockchain, the technology behind Bitcoin protocol, to conceive our new FairAccess authorization framework as ultimate solution and equilibrium that solve all IoT authorization challenges previously highlighted above. Actually, the blockchain is the first technology that has successfully overcome the problem related to how consensus can be reached in distributed anonymous participants, some of whom may be behaving with malicious intent without the intervention of any centralized party. It is a universal digital ledger that functions at the heart of decentralized financial systems such as Bitcoin, and increasingly, many other decentralized systems such as Storj,[1] a decentralized peer-to-peer cloud storage network. Onename,[2] a distributed and secured identity platform. IBM's Adept, an Internet of things architecture [9], Enigma [10] to

[1] http://www.storj.io.

[2] http://www.onename.com/.

enhance user privacy and many others. However, to our knowledge, the use of a blockchain in access control filed has never been explored yet. In the next section the characteristics of the blockchain that will help provide a decentralized privacy preserving access control model are described.

3 How Blockchain Can Be Used in Distributed and Transparent Access Control

3.1 Background

Cryptocurrency and blockchain: Cryptocurrencies are a new form of virtual currency, first introduced with creation of Bitcoin, developed by Nakamoto [10]. A cryptocurrency is a decentralized digital currency built on cryptographic protocols providing an open, self-regulating alternative to classical currencies managed by central authorities such as banks. It is the first technology to successfully overcome the problem related to how consensus can be reached in a group of anonymous participants, some of whom may be behaving with malicious intent without the intervention of any centralized party. Actually, specific nodes known as miners are responsible for collecting transactions, solving challenging computational puzzles (proof-of-work) to reach consensus, and adding the transactions in form of blocks to a distributed public ledger known as the blockchain.

The blockchain: The blockchain technology provides everyone with a working proof of a decentralized trust. All cryptocurrencies utilize what can best be described as a public ledger that is impossible to corrupt. Every user or node has the exact same ledger as all of the other users or nodes in the network. This ensures a complete consensus from all users or nodes in the corresponding currencies blockchain.

Transactions: A transaction records the transfer of a value (altcoin) from some input address to output addresses. Transactions are generated by the sender and distributed amongst the peers in the network. Transactions are only valid once they have been accepted into the public history of transactions, the blockchain. Actually, the fundamental building block of a cryptocurrency transaction is an unspent transaction output, or UTXO. UTXO are a value of the currency locked to a specific owner, recorded on the blockchain, and recognized as currency units by the entire network. The UTXO consumed by a transaction are called transaction inputs, and the UTXO created by a transaction are called transaction outputs. The recipient is identified through their public key, so cryptocurrency transactions can be traced throughout the blockchain, to the beginning of the creation of the cryptocurrency. This forms the mechanism for checking the ownership of cryptocurrency bitcoins. Publicly verifiable transactions by any node avoids double spending and provides a high degree of certainty to the participants of the cryptocurrency ecosystem.

Scripting language and smart contract: Each UTXO has to specify a person (or several persons) eligible for spending virtual money associated with it. To accomplish this, the Bitcoin protocol introduces a scripting language. The language describes the execution of a certain program on a stack machine. Each transaction output contains a script which locks, or encumbers, the money associated with the *UTXO*. This script is commonly referred to as **scriptPubKey**. To spend this money, a user of the Bitcoin network must demonstrate the proof of ownership in the form of an unlocking script **scriptSig**. Hence, a script is a part of each input and each output of a transaction. When we generalize this scripting language computation to arbitrary Turing complete logic, we obtain an expressive smart contract system. The interest in smart contract applications steadily risen since 2014 due to the appearance of Bitcoin-like technologies, such as Etherum [2] and many other works designed specifically to decentralized smart contract system.

3.2 Blockchain in FairAccess

Most cryptocurrencies solutions are designed with a currency in mind. In our FairAccess framework, we define an Authorization Token rather than a bitcoin. This token is simply a digital signature that represents the access right or the entitlement defined by the creator of the transaction to its receiver in order to access a specific resource designed by its address. Blockchain specifications vary from cryptocurrency to cryptocurrency to meet the purpose of specific applications. Our FairAccess Framework uses a custom blockchain transaction specification that includes additional fields tailored to the requirements of a granular access control model. FairAccess provides several useful mechanisms using the blockchain. In fact, in FairAccess, the blockchain is considered as a database that stores all access control policies for each pair (resource, requester) in form of transactions, it serves also as logging databases that ensures auditing functions. Furthermore, it prevents forgery of token through transactions integrity checks and detects token reuse through the double spending detection mechanism.

4 FairAccess: A Token-Based Access Control Model Enforced by the Blockchain Technology

4.1 Technical Description

Preliminaries: We will denote key pairs using the capital letters (e.g. A), and refer to the private key and the public key of A by: A.Sk and A.pk, respectively then: $A = (A.sk, A.pk)$. In addition, we will use the following convention: if A = (A.sk, A.pk) then $\mathrm{sig}_{\mathrm{A}}(m)$ denote a signature on a message m computed with A.Sk and let

Table 1 FairAccess main interacting entities

Acronym	Its meaning
IDx	The index of the current transaction Tx where x = H(Tx), H is a hash function
rs	The address of requested resource
rq	The address of the requester who is the receiver of the current transaction
πx	Locking script (access control policies written in scripting language)
$TKN_{rq,rs}$	Encrypted access token associated to couple (rs, rq)
ref	Point to the previous transaction output

$\text{check}_A(m, \sigma)$ denote the result (true or false) of the verification of the signature σ on the message *m* with respect to the public key A.pk and finally we note H as a hash instantiated by a SHA-256 implementation.

The acronyms we use to describe the functionalities of our proposed framework and their meanings are summarized in Table 1.

Our authorization Framework is based on the following main authorization functionalities: (1) Registering a new resource with a corresponding address. (2) Grant access. (3) Request access. (4) Delegate access. (5) Revoke access. We describe each function in this paragraph.

FairAccess work flow: Typical FairAccess based access control works as follows. A subject (e.g., a device A, identified with the address rq) wants to perform an action (e.g., modify) on a protected resource (e.g., Device B temperature, identified with address rs). The subject submits this request to the authorization management point (AMP = wallet) acting as a policy Enforcement Point (PEP) that manages the protected resource. The PEP formulates such a request to a GetAcess transaction. Then, the PEP broadcasts this transaction to the network nodes till it reach miners, those later act as distributed Policy Decision Point, and evaluate the transaction. The PDP checks the request with the defined policy, by comparing the unlocking script of this transaction to the locking script of the previous GrantAccess transaction. Then determines whether the request should be permitted or denied. Finally, if it is permitted the transaction is valid and it will be recorded in the blockchain, else the transaction will be rejected and a notification will be sent to its sender.

Phase 1: Reload access control policy to the blockchain trough: Grant access transaction

Before requesting access to device B, the Smart device A needs to obtain an access token. For this purpose, it sends a request to device B owner (RO) indicating the address of the target resource rs and the action to be performed on. Then, the device B owner defines his access control policy and reloads it to the blockchain through a GrantAccess transaction. This later is created by the RO wallet. The GrantAccess transaction encapsulates the defined access control policy in form of locking script in its output, the address rq of device A as receiver and the access token signed as UTXO. Then the wallet broadcasts the GrantAccess transaction to the peer to peer nodes. The peer to peer nodes verify the transaction and record it in the block chain in case of success validation. At this stage, the network witnesses that the device B owner has entitled the device A to get access to the specific

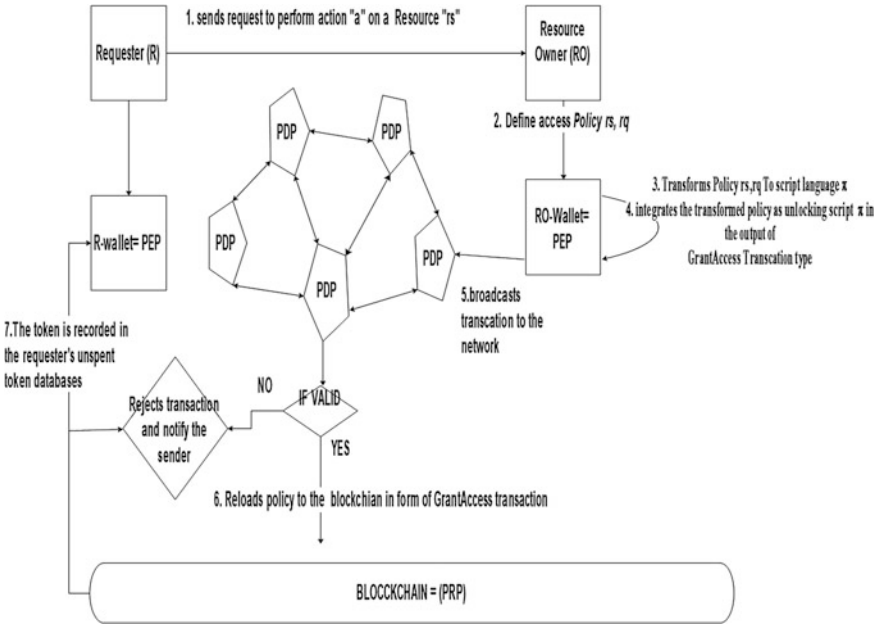

Fig. 1 GrantAccess transaction process

service provided by device B. But device A could not get yet access to the target service till he meets the access control policy and unlock the output of the transaction which is the token.

The sequence of GrantAccess transactions are illustrated in Fig. 1 and described as follows:

1. The RO defines for the couple (Resource *rs*, Requester *rq*) an access control policy $POLICY_{rs,rq}$
2. The wallet transforms this access control policy to a scripting language $POLICY_{rs,rq} \rightarrow \pi_x$
3. The RO, generates a Token $TKN_{rs,rq}$ encrypted with the requester public key.
4. The wallet generates a **GrantAccess** Transaction in the following form:

$$T_x = (m, sig_{rs}(m)) \ where \ m = (ID_x, input(rs), \ output(rq, \pi_x, TKN_{rs,rq})$$

5. Each node verifies the transaction within the transaction validation process.
6. If the transaction is valid the unspent transaction output:$TKN_{rs,rq}$ is recorded in the blockckain and shown in the requester's wallet as part of the available $TKN_{rs,rq}$. Else, the transaction will be rejected.

At the end of this phase, if the transaction appears in the blockchain, it means that a new *TKN* is added to the requester *available unspent TKN* database. Meaning that the network witnesses that the Resource owner had entitled the requester to

access that resource but the requester could not access yet till he unlocks the access condition then could spend the *TKN* To do so, the requester has to prove to the network that he fulfills really the access conditions in a new transaction called **GetAccess** transaction which is the objective of the second phase.

Phase 2: GetAccess

In this phase, the device A will create a new transaction, that we call a GetAccess transaction. The GetAccess transaction redeems the GrantAccess transaction to use the token and access to a service being hosted on the device B. Actually, GetAccess transaction input is an unspent output ($UTXO$ = token) of its previous GrantAccess transactions recorded on the blockchain. The inclusion of this transaction into the block chain enables the delivery of the encrypted access token $TKN_{rs,rq}$ to device A. When device A tries to access to device B. This later can check whether the token is valid or not by checking the signature in one hand and in the other hand checking either the transaction redeeming this token is included in the blockchain. It could also check the requested action against the access rights already defined in the transaction as unlocking script. Finally, since the device have the final say, it can check the current context, like for instance the temperature level, before allowing device A to access its resource. If all those conditions are fulfilled, the request is accepted and the service is provided to the device A.

The sequence of GetAccess transactions are illustrated in Fig. 2 and described as follows:

1. The requester will, first, scan his available TKN database $ScanTKN(rq) \rightarrow TKN_{rs,rq}$
2. The wallet decrypts the token $decrypt(TKN_{rs,rq})$
3. The wallet gets the locking script $GetLockingscript(TKN) \rightarrow \pi_x^{'}$
   Where: $\pi_x^{'}$ is the locking script in the corresponding GrantAcces transaction.
4. The requester fulfills access control condition placed in $\pi_x^{'}$ and generate an unlocking script ψ

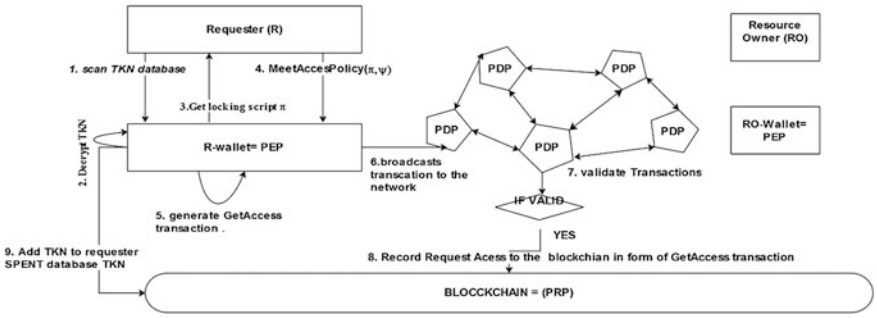

Fig. 2 GetAccess transaction process

$$MeetAccessControlPolicy(\pi_x^{'}) \rightarrow \psi$$

5. The wallet generates a GetAccess transaction type in the following form:

$$T_x = (ID_x, input(ref, rs, \psi), output(rq, TKN_{rq,rs})$$

6. The wallet broadcasts the transaction to the network
7. The network nodes verify and validate the transaction if it was valid it will be included in the blockchain else it will be rejected and a notification is sent to its sender.
8. Once the transaction appears in the blockchain. It means the network witnesses that the requester has full filled the access condition (unlocking script) then the Token is now valid and could be spent.
9. The requester device sends the token to the target device
10. The target device checks the validity of the token by checking the inclusion of GetAcess transaction in the blockchain. If it was valid, the access is allowed else the access is denied.

Delegate access through *DelegateAccess Transaction type*:

Consider Device A again as example. This later can delegate access rights or part of his granted rights over the service *rs* provided by Device B to another Device C identified with *C.pk* to access resource *rs* through this transaction.

1. Device A owner's wallet generate the following transaction:

$$T_x = (m, sig_A(m)) \ where \ m = (ID_x, input(ref, rs, \psi), \ output(C.pk, \pi_x, TKN_{C.pk,rs})$$

2. The wallet broadcasts the transaction
3. The network nodes validate transaction
4. If the transaction is valid, the unspent transaction output: $TKN_{C.pk,rs}$ is recorded in the blockchain and showed in the device C owner's wallet as part of the available $TKN_{C.pk,rs}$
5. When device C wants to access to that resource *rs*, it creates a GetAccess transaction that releases the encumbrance, unlocking the output by providing an unlocking script meeting the access conditions.

Revoke/Update access through a new GrantAccess Transaction type:

The Resource Owner could revoke or update the permissions granted to a requester at any time by simply issuing a GrantAccess transaction with a new set of permissions, including revoking access to previously designed resource. This new transaction will override all rights granted by all previous transactions. Since transaction are recorded in a chronological way in the blockchain.

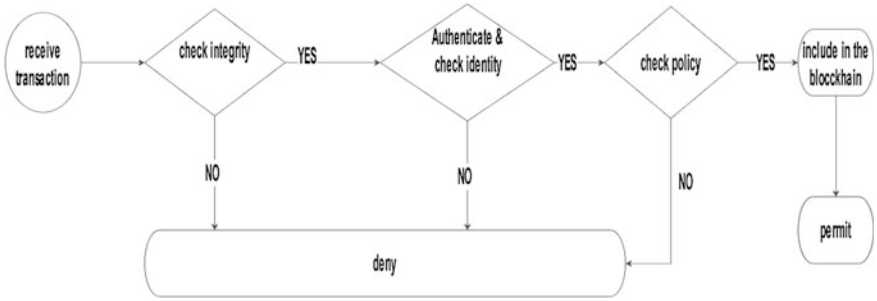

Fig. 3 Authorization evaluation process

Evaluation the access control policy by validation transaction:
Each peer to peer node receives a signed transaction in the following form:

$$(T_x, sig_A(T_x)) \text{ where } T_x = (ID_x, input1(ref, A.pk, \psi), output(B.pk, \pi_x, TKN_{A.pk,B.pk})$$

To validate the transaction and evaluate the access control policy, the node executes the following functions as illustrated in Fig. 3:

1. **CheckIdentity**: checking the signature of the owner prove the following properties: (1) Authenticate the owner. (2) Prove his ownership to the resource (3) prove his non repudiation. This function is ensured by executing this procedure: $Check_A(T_x, \sigma) = True$.
2. **CheckIntegriy** (T_x): Hash the transaction and compare to its ID_x to ensure that the transaction has not been altered during its propagation in the network. This function is ensured by executing this procedure: Compare $(\mathcal{H}(T_x).ID_x) = True$
3. **CheckPolicy**: check if the sender fulfill access control policy which is simply the unlocking scripts for each input must validate against the corresponding output locking scripts. To do so, for each input in the transaction, the validation function CheckPolicy will first retrieves the output of the previous transaction referenced by the input of the current transaction. This output contains a locking script defining the conditions required to get the TKN. The validation function will then take the unlocking script contained in the input that is attempting to get this TKN and execute the two scripts. We note that the $\pi_x^{'}$ is permanently recorded in the blockchain, and therefore is invariable and is unaffected by failed attempts to spend it by reference in a new transaction. This function is ensured by executing these two procedures:
   a. $\text{GetOutputFromRef(ref)} \rightarrow \pi_x^{'}$
   b. $\text{Compare}(\psi, \pi_x^{'}) = \text{True}$

If any result other than "True" remains after execution of described function, then transaction is considered as invalid. The access is denied and a notification is sent to its sender.

Hence we have shown that access control process can be easily achieved using the blockchain, and through the same mechanism, additional integrity and security protections can be added. Furthermore, blockchain scripts allows for intelligent programming of actions within a transaction. This enables FairAccess to implement more granular access control policies, expressed by any access control model as soon as this model could be transcoded to a script language.

5 Conclusion

In this paper, we have inaugurated a new applicability domain of blockchain that is access control through our FairAccess framework. Our framework leverages the consistency offered by blockchain-based cryptocurrencies to solve the problem of centralized and decentralized access control in IoT highlighted in the beginning of this paper. In FairAccess, we provide a stronger and transparent access control tool. We have explained, the main building blocks and functionalities of our framework. In future work, we will implement FairAccess with RaspberryPI IoT device and bitcoin blockchain as example.

References

1. Vermesan, P., Friess, P., Guillemin, S., Gusmeroli, H., Sundmaeker, A., Bassi, I.S., Jubert, M., Mazura, M., Harrison, M.D.: Internet of things strategic research roadmap. In: Cluster of European Research Projects on the Internet of Things, CERP-IoT (2011)
2. SZABO, Nick: Formalizing and securing relationships on public networks. First Monday, **2** (9) (1997)
3. Zhang, G., Tian, J.: An extended role based access control model for the Internet of Things. In: 2010 International Conference on Information Networking and Automation (ICINA), pp. V1-319–V1-323. IEEE, (2010)
4. Hernández-Ramos, J.L., Jara, A.J., Leandro, M., et al.: Dcapbac: embedding authorization logic into smart things through ecc optimizations. Int. J. Comput. Math. no ahead-of-print, 1–22 (2014)
5. Seitz, L., Selander, G., Gehrmann, C.: Authorization framework for the internet-of-things. In: 2013 IEEE 14th International Symposium and Workshops on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), pp. 1–6. IEEE (2013)
6. Hardt, D. (ed.): The OAuth 2.0 authorization framework. In: IETF, RFC6749, October 2012
7. Connect All IP-Based Smart Objects (CALIPSO)—FP7 EU Project [Online]. http://www.ict-calipso.eu/. Accessed 15 Oct 2014
8. Cirani, S., Picone, M., Gonizzi, P., Veltri, L., Ferrari, G.: Iot-oas: an OAuth-based authorization service architecture for secure services in IoT scenarios. IEEE Sens. J. **15**(2), 1224–1234 (2015)
9. Sanjay, P., Sumabala, N., Paul, B., Pureswaran, V.: ADEPT: an IoT practitioner perspective, Draft copy for advance review. IBM (2015)
10. Zyskind, G., Nathan, O.: Decentralizing privacy: using blockchain to protect personal data. In: Security and Privacy Workshops (SPW), 2015 IEEE, pp. 180–184. IEEE (2015). Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system (2008)