Semantic IoT Gateway: Towards Automated Generation of Privacy-Preserving Smart Contracts in the Internet of Things

Faiza Loukil[1(✉)], Chirine Ghedira-Guegan[2], Khouloud Boukadi[3], and Aïcha Nabila Benharkat[4]

[1] University of Lyon, University Jean Moulin Lyon 3, CNRS, LIRIS, Lyon, France
faiza.loukil@liris.cnrs.fr
[2] University of Lyon, University Jean Moulin Lyon 3, iaelyon School of Management, CNRS, LIRIS, Lyon, France
chirine.ghedira-guegan@liris.cnrs.fr
[3] Mir@cl Laboratory, Sfax University, Sfax, Tunisia
khouloud.boukadi@fsegs.usf.tn
[4] University of Lyon, INSALyon, CNRS, LIRIS, Lyon, France
nabila.benharkat@liris.cnrs.fr

Abstract. The Internet of Things paradigm has brought opportunities to meet several challenges by interconnecting IoT resources, such as sensors, actuators, and gateways on a massive scale. The IoT gateways play an important role in the IoT applications to bridge between sensor networks and the external environment through the Internet. Typically, the IoT gateways collect and send the data collected from sensors and actuators to external platforms where they will be remotely analyzed. However, the users desire a more adapted IoT gateway that can improve the IoT data privacy preservation before sending them to these external platforms. Thus, an IoT gateway that enables a better control over the set of private IoT resources and protects the collected personal data and their privacy is required. For this purpose, we propose a Semantic IoT Gateway that helps implement a dynamic and flexible privacy-preserving solution for the IoT domain. First, it enables to match between the data consumer's terms of service and the data owner's privacy preferences by generating an adapted privacy policy. Second, it converts the privacy policy into a custom smart contract. Finally, it connects a set of private IoT resources to a distributed network using the blockchain technology to host the generated smart contracts. A smart contract is an executable code that runs on top of the blockchain to facilitate, execute and enforce an agreement between untrusted parties without the involvement of a trusted third party. Our proposal, which is highlighted through an example and experimentation on a real-world use-case, has given the expected results.

H. Panetto et al. (Eds.): OTM 2018 Conferences, LNCS 11229, pp. 207–225, 2018.
https://doi.org/10.1007/978-3-030-02610-3_12

1 Introduction

The Internet of Things (IoT) is a novel paradigm, the main strength of which is its high impact on several aspects of everyday's life and behavior of potential users. From the point of view of the private user, the most obvious effects of the IoT introduction will be visible in both working and domestic fields, such as assisted living, e-health, and enhanced learning. Similarly, from the perspective of the business users, the most apparent consequences will be equally visible in fields, such as automation and industrial manufacturing, business/process management, and intelligent transportation of people and goods [3].

Actually, many challenging issues related to the IoT resource characteristics still need to be addressed. In fact, they have a low computation and an energy capacity to protect personal data and the user's privacy. In order to overcome this problem, another IoT resource type is proposed. This is called IoT Gateway, the role of which is to collect and send the collected data from IoT sensors and actuators to external platforms to be remotely analyzed. Moreover, those external platforms gather the IoT data and use them to personalize services, optimize decision-making processes, and predict future trends. However, the IoT data raise security and privacy concerns. In fact, the users have a little or no control over the collected data about themselves [10]. For instance, sharing the collected data by wearable devices with service providers leads to lose the IoT data control and ownership [10]. Moreover, users have no guarantee that the service provider will respect the licensing agreement concerning privacy and security protection [10]. Moreover, the used IoT gateways are generic, with basic settings, and do not preempt the user's requirements especially concerning the privacy issue.

Motivated by the actual basic role and the need to have a more flexible gateway that enables to better control the set of private IoT resources, we propose a Semantic IoT Gateway as a core component of our proposed end-to-end privacy-preserving framework for the IoT data based on the blockchain technology, called PrivBlockchain [9]. The reason behind using the blockchain technology is that the blockchain is an immutable public record of data secured by a network of peer-to-peer participants that hosted smart contracts, which are executable codes that run on top of the blockchain to facilitate, execute and enforce an agreement between untrusted parties without the involvement of a trusted third party. Nevertheless, the Semantic IoT Gateway is intended to convert the data owner's privacy preferences into smart contracts that will be published within the blockchain. Moreover, our Semantic IoT Gateway is based on an IoT privacy ontology, called LIoPY, which is a European legal compliant ontology to preserve privacy for IoT and defined in our previous work [8]. Thanks to LIoPY use, a privacy policy can be inferred according to the data owner's privacy preferences and the data consumer's terms of service. This policy is a set of conditions that the consumer needs to fulfill in order to handle specific shared IoT data. Those conditions are hosted in a smart contract. Thus, the use of a smart contract will prevent any privacy violation attempts by enforcing the data privacy requirements and ensuring that the shared data will be handled as expected in the whole IoT data lifecycle, collection, transmission, storage and processing phases.

This paper is organized as follows. Section 2 deals with the existing researchers who studied how privacy is preserved in the IoT scope. Section 3 presents an overview of the PrivBlockchain framework. Section 4 identifies the framework's core components. Section 5 defines the proposed Semantic IoT Gateway and explains its components and main functionalities. Section 6 validates our solution in a healthcare scenario. Section 7 concludes the paper and presents some future endeavors.

2 Related Work

There are many researchers, who have studied the integration of IoT technology and semantic modeling or blockchain technology for preserving privacy.

Semantic-based privacy preservation solutions are based on ontologies and inference rules for developing smart applications. For instance, Celdran et al. [5] proposed a solution called SeCoMan, in which an ontology is employed to model the description of entities, reason over data to obtain useful knowledge, and define context-aware policies. However, privacy protection is fulfilled in a location-limited level. For their part, Wang et al. [13] proposed an Ontology-based Resource Description Model to describe resources in the IoT environment. They defined a Privacy class that protects the device from illegal access or control. However, ORDM did not offer fine-grained access control to the sensed data.

Blockchain-based privacy preservation solutions are based on the blockchain technology for enabling users to preserve their IoT data privacy while eliminating the need to trust a centralized regulator. For instance, Hashemi et al. [6] proposed a distributed data storage system, which used blockchain to maintain data access control and data storage model. For their part, Zyskind et al. [14] proposed a decentralized personal data management system, which used blockchain to keep track of both data and access transactions. However, the IoT devices have not sufficient resources to store the whole blockchain.

To the best of our knowledge, the combination of semantic modeling and blockchain technology for preserving IoT data privacy has never been explored.

3 PrivBlockchain Overview

Considering the legal rights imposed by the GDPR [11], it is necessary to ensure the privacy requirement compliance to preserve privacy during the whole data lifecycle, covering the collection, transmission, storage and processing phases. In our previous work [9], we have proposed PrivBlockchain, an end-to-end privacy-preserving framework for the IoT data based on the blockchain technology. PrivBlockchain aims at enforcing these privacy requirements and obligations for the IoT environment.

PrivBlockchain is based on the main following principles. **User-driven and transparency:** The user is the master of his own data since he has a full control over the data he shared in the network. **Fairness:** Using the blockchain in

our end-to-end privacy-preserving framework improves fairness because nobody could systematically be enforced to lose control over his own data. **Distributed architecture and the lack of a central authority:** Each node in the network directly shares its data with other nodes, without the intervention of any third or trusted entity to manage the whole network. **Fine-granularity:** The use of a smart contract enables the user to implement expressive and granular privacy policies over our framework.

Figure 1 depicts PrivBlockchain, the proposed architecture that includes two types of network: first, the private IoT network, which can be a smart home, smart building, etc. This network includes the IoT resources owned by a data owner, which can be an individual or an organization. The second network is the public IoT network, which represents the external domain of the private IoT network. Moreover, we distinguish three IoT network node types, namely private, public, and storage nodes. Both public and storage IoT nodes belong to the public network. The private node (i.e., Semantic IoT Gateway or private IoT resource) is an IoT node that belongs to both the private and public IoT networks.

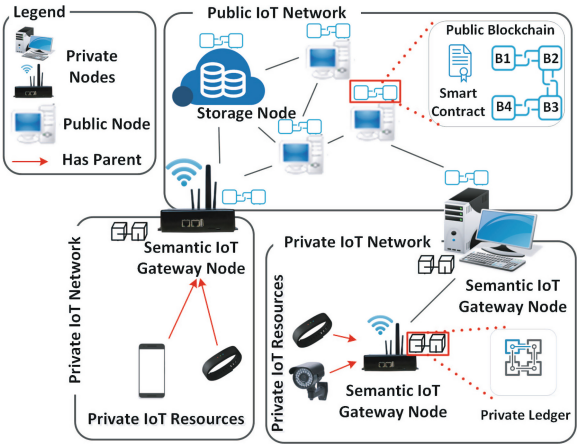

Fig. 1. PrivBlockchain architecture

In the private IoT network, each data owner has one or more high resource devices, known as the "Semantic IoT Gateway", which is responsible for the other owned IoT resources. The communication between the owned IoT resources by the data owner (i.e., the private IoT resources) is stored in a private lockchain called the "private ledger". The communication between the private IoT nodes and the other nodes of the public IoT network is stored in a "public blockchain".

We outline the proposed framework core components in the following section.

4 PrivBlockchain Core Components

This section discusses the main blockchain-based solution components. Indeed, the PrivBlockchain framework consists of nine core components, such as smart contract, transaction, private IoT network, private ledger, Semantic IoT Gateway, local storage, public IoT network, public blockchain, and storage node.

4.1 Smart Contract

Two parties can share a set of conditions by signing a common agreement. This kind of published agreement within the blockchain is known as a smart contract, which contains a code and defines a set of functions. For instance, the smart contract can define the constructor function that enables to create the smart contract itself. The sender of the transaction (i.e., network node) that invokes the constructor function becomes the smart contract owner.

A self-destruct function is another example of the functions that can be defined in a smart contract. Usually, only the smart contract owner can destruct the contract by invoking this function. A smart contract is likely to be a class that contains state variables, functions, function modifiers, events, and structures [4]. Besides, it can even call other smart contracts. We represent the smart contract, which is denoted as SC, as a tuple that has the following form:

$$SC = <states, functions>$$

- **States:** they are variables that hold some data or the owner's Ethereum wallet address (i.e., the address in which the smart contract is deployed). We can distinguish between two state types, namely *constant states*, which can never be changed, and *writable states*, which save states in the blockchain.
- **Functions:** they are pieces of code that can read or modify states. We can distinguish between two function types, namely *read-only functions*, which are marked as constant in the code and do not require *gas* to run and *write functions* that require *gas* because the state transitions must be encoded in a new block of the blockchain.

In order to invoke one smart contract function, a transaction needs to be created.

4.2 Transaction

Communication between IoT resources and network nodes is known as a transaction. In our work, we define a set of transaction types. T_{Add} and T_{Remove} transactions are generated by a Semantic IoT Gateway to add a new private IoT resource or to remove it from the network. $T_{LocalStore}$ transaction is generated by IoT resources to locally store the data. T_{Store} transaction is generated by IoT resources to store data on a Storage Node that can be a cloud storage provider. T_{Access} transaction is generated by an IoT resource, gateway node or IoT network node to access a shared data. $T_{Monitor}$ transaction is generated by an IoT network node to periodically receive near real-time collected data by IoT resources.

Moreover, $T_{GetPermission}$, $T_{GrantPermission}$, and $T_{GetSharedResource}$ transactions are used to ask for permissions to (i) access a specific IoT resource output, (ii) define a set of permissions to a data consumer by a data owner, and (iii) handle the shared data by an allowed data consumer. Each transaction contains a set of parameters, as depicted in Fig. 2, such as the previous transaction identifier to chain transactions, the current transaction identifier, and the transaction type.

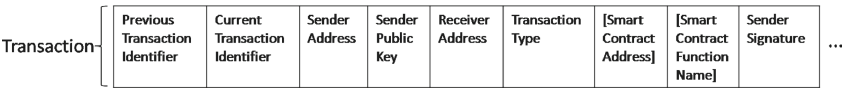

Fig. 2. Transaction structure

Lightweight cryptography, such as AES Encryption [7], is used by IoT resources to secure the transactions during the communication. It should be noted that all the transactions between the IoT resources and the Semantic IoT Gateway occur in a private IoT network.

4.3 Private IoT Network

The private IoT network is an area, like a smart home or a smart building, where its owner can control a set of owned IoT resources. Indeed, the private IoT network includes a set of private IoT resources and Semantic IoT Gateway nodes, which are high resource devices that validate communication between the private IoT resources and link these private resources with the public IoT network. In our work, we distinguish between two node types, namely *full nodes*, which process every transaction and store the entire blockchain, and *light nodes*, which only store the relevant information, such as the gateway node and smart contract addresses due to their limited resources. In our private IoT network, the gateway nodes are full nodes while the private IoT resources are light nodes. Each private IoT network maintains a private ledger.

4.4 Private Ledger

A private ledger is a local private blockchain that enables the data owner to control his own IoT resources. This blockchain contains the data owner's private IoT resource communication and has a set of smart contracts that enforce the data owner's privacy preferences on how his IoT resources must behave. Transactions are chained together in a block. Each block in the private ledger contains a block header, which is the hash of the previous block to keep the blockchain immutable. Besides, each block contains a list of transactions (see Fig. 1).

The private ledger is kept and managed by a set of Semantic IoT Gateway nodes.

4.5 Semantic IoT Gateway Node

A Semantic IoT Gateway is a device with high memory and storage capabilities. Each gateway node is responsible for a set of private IoT resources, generates their keys and adds them to the IoT network. In our proposal, IoT resources with low memory and storage capabilities, such as a Beaglebone or an Arduino board, can delegate complicated treatments to the Semantic IoT Gateway. Moreover, it validates the incoming and the outgoing transactions before adding them to the private ledger. On the other hand, the Semantic IoT Gateway is considered as a public node in the public IoT network. In fact, it communicates with both the public and storage nodes, and stores a copy of the public blockchain to benefit from the IoT applications that are offered by the public IoT network nodes. For this purpose, it uses another couple of public and private keys different from the couple used in the private IoT network to reduce the linkability problem.

Furthermore, the Semantic IoT Gateway manages a local storage.

4.6 Local Storage

Local storage is a storing device, which is used to store data locally. It saves the collected data by IoT resources for a long-term storage before sending them to the external storage center, which is the storage node in our case. Each data block is stored using its Data Block ID. In case of a data center failure, the data can be restored from the local storage using the unique data identifiers. The local storage provides an additional capability to the Semantic IoT Gateway to belong as a public node to the public IoT network.

4.7 Public IoT Network

The proposed public IoT network is a peer-to-peer network (P2P) that contains several nodes with different memory and storage capabilities. The public nodes can be a gateway, a storage, or a public node. These nodes require a high memory and storage capabilities to store the public blockchain. Each IoT network node has a unique pair of public (PK) and private (SK) keys. The former, which is known by the other public nodes in the IoT network, is used as a unique node identifier to communicate (send/receive) transactions from the other nodes in the public IoT network while the latter, which is kept secret to the node, is used to sign transactions before sending them. Then, the signature is verified using the node's PK in the transaction. The digital signature, which is the hash of a digital asset (i.e., a transaction), improves transaction sender's authentication (i.e., proves that the transaction sender has the appropriate couple of public key and blockchain address), non-repudiation (i.e., the sender cannot deny having sent a transaction), and integrity (i.e., proves that a transaction is not altered while transmitted). Only valid transactions can be added to the public blockchain and distributed between all the public IoT network nodes. The public IoT network maintains a public blockchain.

4.8 Public Blockchain

The public blockchain can be seen as the history of all the transactions that are sent by the public nodes to access or share IoT data in the public IoT network. In fact, it can ensure auditing functions. Hence, our solution offers a non-repudiation principle compliance, which consists in preventing any public IoT network node from denying actions that are performed by itself. Furthermore, the public blockchain contains smart contracts that enforce the data owner's privacy preferences on how his data must be handled. In fact, the smart contract can be considered as data owner's privacy policy that specifies obligations for handling the shared IoT data. The public blockchain is stored on public and storage nodes.

4.9 Storage Node

The storage node is proposed as a public IoT network node that offers a storing service for both public blockchain and data collected by the IoT resources. For instance, the storage node can be a cloud storage service. Each data owner has the choice whether to use a different storage node for each of his IoT resources or the same storage node to store all of his collected IoT data. It is worth noting that the use of separate storage nodes can reduce privacy hurdles, especially the linkability issue [12]. Thus, separate databases must be created in such a way that common attributes are avoided.

After presenting an overall design architecture of PrivBlockchain, we focus on the core component, which is the Semantic IoT Gateway in the following section.

5 Semantic IoT Gateway

Typically, gateways collect and send the collected data from IoT resources like sensors and actuators to external platforms in order to be remotely analyzed. In order to enable a better control over their private IoT resources, a more flexible gateway is required by the users. For this purpose, we propose a Semantic IoT Gateway that aims at converting the data owner's privacy preferences into smart contracts that will be published within the blockchain to be enforced. As aforementioned, our Semantic IoT Gateway is based on an IoT privacy ontology, called LIoPY and defined in our previous work [8].

The architecture of the Semantic IoT Gateway is shown in Fig. 3. It includes four core components, which are: (i) the Semantic Rule Manager, which aims at matching the data owner's privacy preferences and the data consumer's terms of service in order to generate an adapted privacy policy, (ii) the Smart Contract Factory, which converts the privacy policy into a custom smart contract that will be hosted in the blockchain to enforce the privacy requirements. Moreover, it generates three smart contract types, namely *PrivacyPermissionSetting*, *Ownership*, and *PrivacyPolicy*. The two first smart contracts are published in the private ledger while the third is published in the public blockchain, (iii) the MQTT Client, which enables the Semantic IoT Gateway to subscribe to the data

consumer's terms of service and publish the custom smart contract parameters using the MQTT standard, which is a lightweight publish/subscribe messaging protocol, and (iv) the Blockchain Client, which is considered as an access point to the blockchain network to receive a blockchain address and access this latter.

All the Semantic IoT Gateway components interact among them and with the external network in order to preserve the IoT data privacy. We detail below those components, the associated processes/workflows, and an example of a smart contract generation protocol.

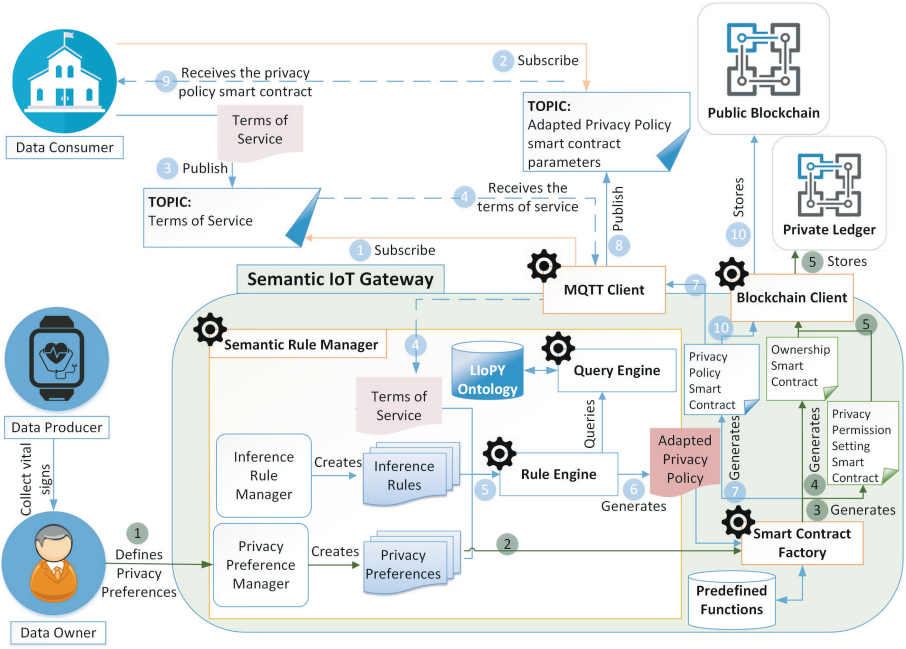

Fig. 3. Architecture of Semantic IoT Gateway

5.1 Semantic Rule Manager: From Privacy Preferences to Privacy Policy

The Semantic Rule Manager provides the Semantic IoT Gateway with semantic capabilities that enable to infer additional knowledge from the defined concepts in the European Legal compliant IoT Privacy-preserving ontologY (LIoPY) [8]. The main purpose of our LIoPY ontology is to enable inferring a privacy policy that aims at protecting privacy during the whole process of collecting, transmitting, storing, and processing the collected data by smart devices.

Figure 4 provides an overview of LIoPY. In order to cover the whole privacy aspects, the LIoPY contains three main modules, namely IoT resource management, IoT description, and IoT resource result sharing management. Each

module includes a set of sub-modules. We referred to the data owner's privacy preferences by the Privacy_Rule class and to the data consumer's terms of service by the Terms_of_Service class. Both of these two classes are associated to a set of privacy requirements depicted by the Privacy_Attribute class, which has a set of subclasses, namely Consent, Purpose, Retention, Operation, Condition, and Disclosure. These subclasses specify for what reason, for how long, how, under which conditions the owner's data will be handled, and to whom they can be disclosed. The Semantic Rule Manager is based on our matching algorithm described in [8] to provide the appropriate policy that matches the privacy requirements of both Privacy_Rule and Terms_of_Service classes using a set of predefined inference rules.

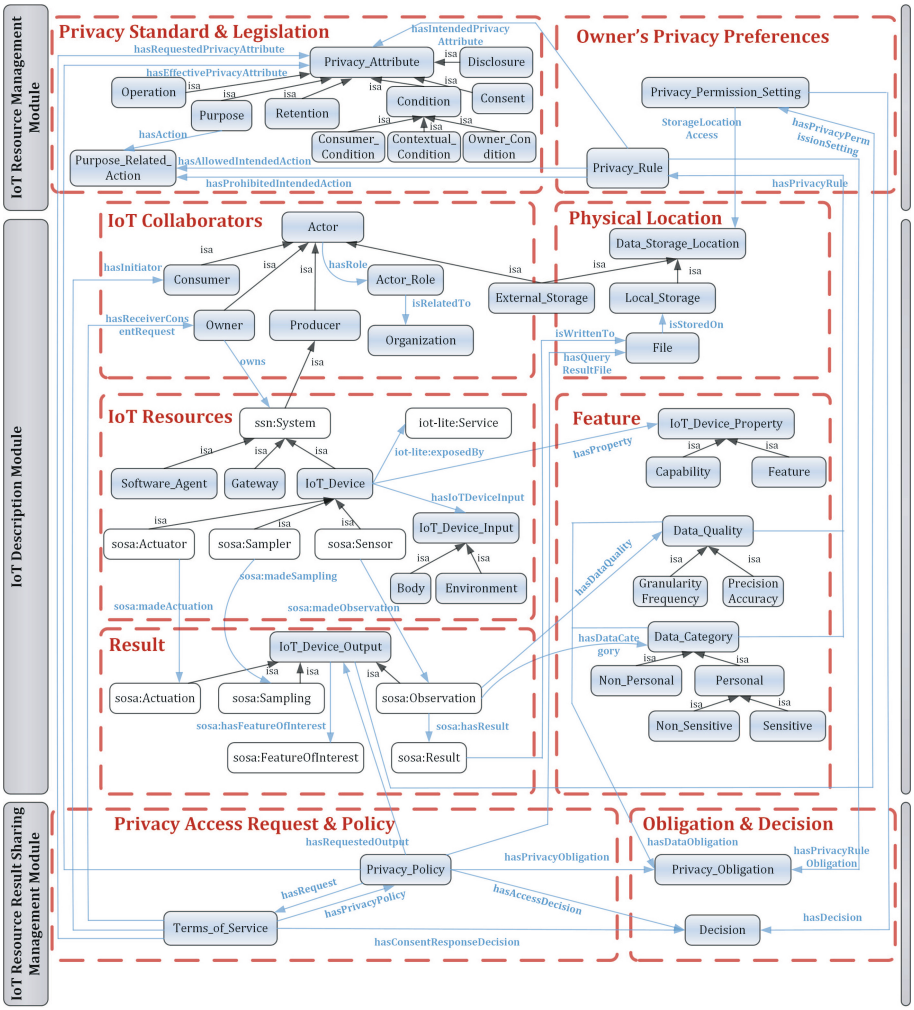

Fig. 4. LIoPY overview

Once a privacy policy is generated by the Semantic Rule Manager, it is converted to a smart contract by the Smart Contract Factory that is detailed below.

5.2 Smart Contract Factory: From Privacy Policy to Smart Contract

As aforementioned, three smart contract types are proposed, namely *PrivacyPermissionSetting*, *Ownership*, and *PrivacyPolicy*. The first and second contracts enforce the data owner's privacy preferences on how his IoT resources must behave according to each data output while the third enforces the data owner's privacy preferences and requirements on how his data must be handled once shared.

PrivacyPermissionSetting Smart Contract. In order to add a new *PrivacyPermissionSetting* smart contract, the Semantic IoT Gateway creates the contract and deploys it in the private ledger. Each IoT resource that knows the smart contract address can use it by invoking its defined functions.

The *PrivacyPermissionSetting* smart contract is designed to store the permission for each IoT resource concerning a specific IoT resource output according to the data owner's privacy preferences. This smart contract defines a set of functions, namely: (i) **LocalStore** function that enables to verify the IoT resource permission to locally store its collected data, (ii) **ExternalStore** function that verifies if the IoT resource has the permission to send the collected data to be stored on an external storage node, (iii) **Read** function that verifies if the IoT resource has the permission to request data from other internal or external IoT resources after verifying the IoT resource permissions, (iv) **Write** function that enables an IoT resource to add and/or modify a requested data collected by other internal or external IoT resources if the IoT resource is permitted, and (v) **Monitor** function that enables to verify the IoT resource permission to receive periodic data from another IoT resource. Furthermore, the *PrivacyPermissionSetting* smart contract includes a self-destruct function. Only the Semantic IoT Gateway can invoke this function to destruct the smart contract in order to revoke the granted privacy permissions for all the IoT resources associated with this contract. It is worth noting that when destructing the smart contract, it will be inoperable, but its history remains in the private ledger.

Ownership Smart Contract. The Semantic IoT Gateway creates an *Ownership* smart contract in order to store its own IoT resource addresses. For each IoT resource, a set of outputs is added. A *PrivacyPermissionSetting* contract is associated with each IoT resource output. Thus, the smart contract address is stored on the *Ownership* smart contract and sent to the appropriate IoT resource according to its data outputs and granted permissions.

The *Ownership* smart contract is designed to enforce data owner's control over his IoT resources and their outputs. It defines a set of functions, namely: (i) **addNewIoTResource** function, which enables to add a new IoT resource by

indicating an IoT resource address, an IoT resource output, and the address of the *PrivacyPermissionSetting* smart contract, which is associated with the IoT resource output, (ii) **modifyIoTResource** function, which enables to modify the description of an existing IoT resource except for the set of its outputs, (iii) **removeIoTResource** function, which enables to remove an existing IoT resource, (iv) **addIoTResourceOutput** function, which enables to add a new output to an existing IoT resource by indicating a description output and the associated *PrivacyPermissionSetting* smart contract address, (v) **modifyIoTResourceOutput** function, which enables to modify the description of an existing IoT resource output, and (vi) **removeIoTResourceOutput** function, which enables to remove an existing IoT resource output from an existing IoT resource.

PrivacyPolicy Smart Contract. A *PrivacyPolicy* smart contract is created when a data owner wants to share a new IoT resource output with consumers. A set of subscribers can be added to the allowed consumer's list. This smart contract is designed to enforce the data owner's privacy preferences on how his IoT resource outputs must be handled once shared. The *PrivacyPolicy* smart contract contains many data fields, such as data hash, data path hash (while the data path is sent in a private transaction over HTTPS), creation date, and a set of consumer's addresses. Each consumer is defined with various permissions. The defined **addConsumer** function enables to add a new consumer by indicating its address, and a set of permissions relevant to the privacy requirements, such as the allowed action according to the chosen purpose, operation, retention duration, disclosure limitation, etc. Moreover, a set of conditions can be added to each existing consumer's permission, such as the allowed location consumer's address, time of day for handling the shared data, and the allowed role of the consumer's address. To this end, an **addCondition** function is defined. When the retention duration ends, the consumer's address is automatically removed from the consumer's list by invoking the **removeConsumer** function. Besides, this function is used when the data owner wants to revoke the permissions of a specific consumer. In case of the file content modification, the **updateFile** function needs to be invoked in order to keep consistency between the file hash that is stored on blockchain and the off-blockchain stored file content. Similar to the smart contract owner, a consumer with a *write permission* can invoke this function in order to change the hash of the file content. It should be noted that the use of data hash enables the data integrity.

After presenting the two core components of the Semantic IoT Gateway, we define an example of a smart contract generation in the following subsection.

5.3 *PrivacyPolicy* Smart Contract Generation Protocol

Figure 5 depicts the business process of generating and adding a new *PrivacyPolicy* smart contract to the public blockchain. The parameters of such a smart contract are only generated in case of a match between the data consumer's terms of service and the data owner's privacy preferences.

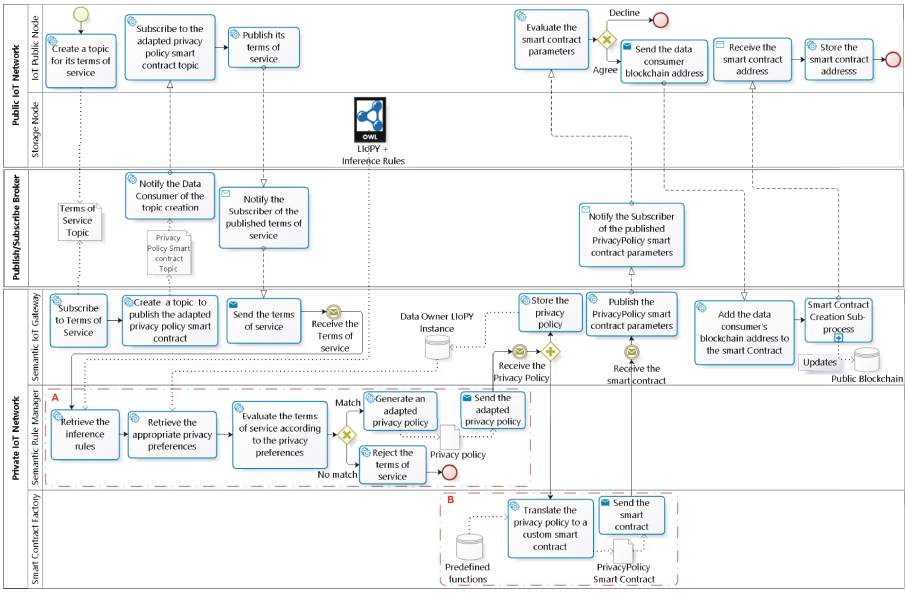

Fig. 5. Adding a new *PrivacyPolicy* smart contract business process notated in BPMN

Each IoT public node (i.e., data consumer in our case) publishes its terms of service by creating the appropriate topic according to the data that will be used. An MQTT broker manages these topic and broadcasts the published messages to the appropriate subscribers. The Semantic IoT Gateway subscribes on the data consumer's terms of service topic and creates a new topic in which it will publish a set of *PrivacyPolicy* smart contract parameters. The Broker notifies the data consumer of the topic creation to subscribe. Once the data consumer publishes new terms of a service, the privacy policy generation sub-process starts (see Fig. 5A). When the Semantic IoT Gateway node receives the terms of service, it communicates with the Semantic Rule Manager, which is responsible for reasoning about the received request and then taking a decision whether to create or not a privacy policy. First, the Semantic Rule Manager retrieves the predefined set of inference rules, which are stored on the storage node and shared between all the involved parties. These inference rules help the Semantic Rule Manager to retrieve the appropriate data owner's privacy preferences from the "Data Owner's LIoPY Instance" base, which is an instance of LIoPY ontology. Then, the Semantic Rule Manager matches the terms of service with the privacy preferences to infer an adapted privacy policy. In case of a match, a privacy policy is created and sent to the Semantic IoT Gateway that will store the privacy policy on the "Data Owner's LIoPY Instance" base and starts the smart contract generation sub-process. Otherwise, the Semantic Rule Manager rejects the terms of service and the process stops. Figure 5B shows the smart contract generation Sub-Process in details. In order to generate a new smart contract,

the Semantic IoT Gateway begins by sending the privacy policy to the Smart Contract Factory, which will transform the privacy policy into a smart contract using the set of predefined functions. In fact, each privacy policy parameter is presented by a function on the smart contract. Moreover, some functions are automatically added to any smart contract regardless of the privacy policy. For instance, the constructor function, which enables the creation of the smart contract itself is an example of these default functions. Once the smart contract is created, the Smart Contract Factory sends it to the IoT Gateway.

Once the Semantic IoT Gateway receives the *PrivacyPolicy* smart contract, it publishes its parameters. When the data consumer receives the smart contract parameters, it evaluates them and decides whether to accept or reject them. In case of a rejection, the process will stop. In case of an agreement, the data consumer communicates its blockchain address to the Semantic IoT Gateway that starts the smart contract creation sub-process. Thus, it adds the received data consumer's blockchain address to the *PrivacyPolicy* smart contract and broadcasts a signed transaction that invokes the smart contract constructor to host it on the blockchain. Once the smart contract is hosted, the Semantic IoT Gateway sends its address to the data consumer and the process is finished with success.

6 Prototype and Validation

We implemented our proposed smart contracts using the Solidity language [2] and deployed it to the Ethereum test network. Because our system does not rely on the currency transfer, there is no difference between the real Ethereum network and the Ethereum test network. Ethereum is currently the most common blockchain platform for developing smart contracts [4].

We applied our solution to the following scenario from the healthcare domain in order to validate our solution:

A patient named Alice needs to follow a healthcare protocol, which consists in practicing some sport activities and eating healthy meals. Alice owns a wearable device that collects the user's heart rate. This IoT resource is connected to Alice's Semantic IoT Gateway. Alice regularly goes to a modern gym. During the training, the wearable device collects Alice's vital parameters and sends them to her gateway. The latter receives Alice's sensitive data and decides what information to send to the hospital to be stored on Alice's medical base, which is regularly checked by her doctor. These stored data are analyzed to propose personalized recommendations for patients. Hence, a need for a break or water notifications could be sent to Alice when necessary. Moreover, Alice wants to use a "Healthy Eating" application, which is offered by the gym with the aim of proposing a set of healthy meals according to the needed calories for each specific user.

However, Alice is afraid of losing the control over her data once shared. For this purpose, she uses her Semantic IoT Gateway that checks if the "Healthy Eating" application's terms of service match her privacy preferences. We assume that Alice's gateway and the IoT application are connected to the PrivBlockchain framework and each of them is identified by Ethereum addresses.

The depicted steps by the business process (see Fig. 5) are implemented to validate our proposal. First, the data consumer (i.e., "Healthy Eating" application administrator) creates an MQTT message that includes the requested data, and a set of privacy requirements, such as the use purpose, disclosure, retention duration, the requested operation, etc. The reason behind the use of JSON format for the MQTT message's content is to ensure an easy matching between the data consumer's terms of service and the data owner's privacy preferences, which are defined on RDF format. Listing 1.1 is an example of MQTT message content.

Listing 1.1. MQTT message content notated in JSON's format

```
{
        "className": "Terms_of_Service",
        "individualName": "eatHealthy_TersmsOfService",
        "objectProperties": {
                "hasRequestedPrivacyAttribute": ["Treatment_Purpose",
                "Write", "Retention_180_days", "With_Requester"]
        },
        "dataProperties": {
                "requested_data_name": "Heartrate"
        }
}
```

When the Semantic IoT Gateway received the "Healthy Eating" application terms of service, it communicated with the Semantic Rule Manager that decided if the received terms of service match Alice's privacy preferences that are already stored on "Alice's LIoPY instance" base. In our case, the Semantic Rule Manager succeeded in inferring an adapted privacy policy that is depicted in Fig. 6.

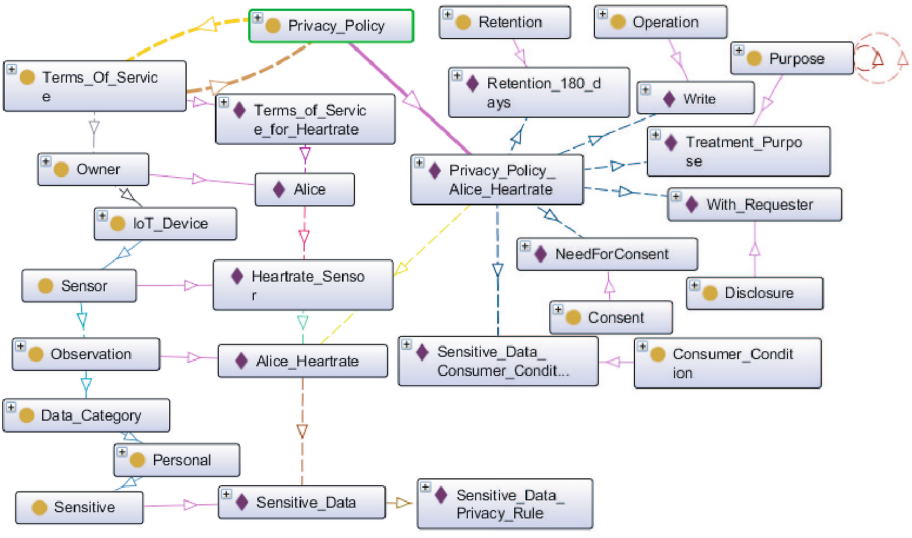

Fig. 6. The generated privacy policy by the Semantic IoT Gateway

Once the privacy policy is created, the Semantic IoT Gateway stored it on "Alice's LIoPY instance" base and sent it to the Smart Contract Factory, which converted the policy into a smart contract using a set of predefined functions. For instance, if in the privacy policy, the permitted operation is 'Write', then the *updateFile* function is added to the custom smart contract. Moreover, the Smart Contract Factory added a modifier, called *AllowedToWrite* that aims at verifying a set of conditions before executing the 'Write' operation. Listing 1.2 shows an example of code that is included in the generated *PrivacyPolicy* smart contract.

Listing 1.2. Example of a *PrivacyPolicy* smart contract including two Solidity predefined functions: *updateFile* function and *AllowedToWrite* modifier

```
contract PrivacyPolicy {
...
modifier AllowedToWrite(address _account) {
        require(  owner == _account ||
                ( isConsumer[_account] && isAllowedToWrite[_account]
                && now < (now + retentionDuration[_account]) ) );
        _;
    }
function updateFile(string new_data_file_name,
                string new_data_file_path, bytes new_data_hash) public
        AllowedToWrite(msg.sender) {
        require(msg.sender == owner);
        sharedFile = File(sharedFile.dataFileId, new_data_file_name,
        new_data_file_path, new_data_hash, sharedFile.createdDate,
        now, sharedFile.consumers, sharedFile.storageNode);
    }
...
}
```

Once the Semantic IoT Gateway received the *PrivacyPolicy* smart contract from the Smart Contract Factory, it published its parameters. Figure 7 depicts the process of publish/subscribe between the Semantic IoT Gateway and the data consumer. Both Semantic IoT Gateway and the "Healthy Eating" application interacted with a publish/subscribe message broker as MQTT clients, which we developed in Java using Eclipse Paho [1]. We chose Eclipse Mosquitto as a broker because it is an open-source message broker that implements the MQTT standard, which is a lightweight publish/subscribe messaging protocol.

When the data consumer received the smart contract parameters, it evaluated them and decided to agree or reject them. In case of a rejection, Alice cannot use the "Healthy Eating" application because it did not match her privacy preferences. In case of an agreement, the application communicated its blockchain address to the Semantic IoT Gateway that hosted the smart contract on the blockchain and sent its address to the "Healthy Eating" application as depicted in Fig. 8.

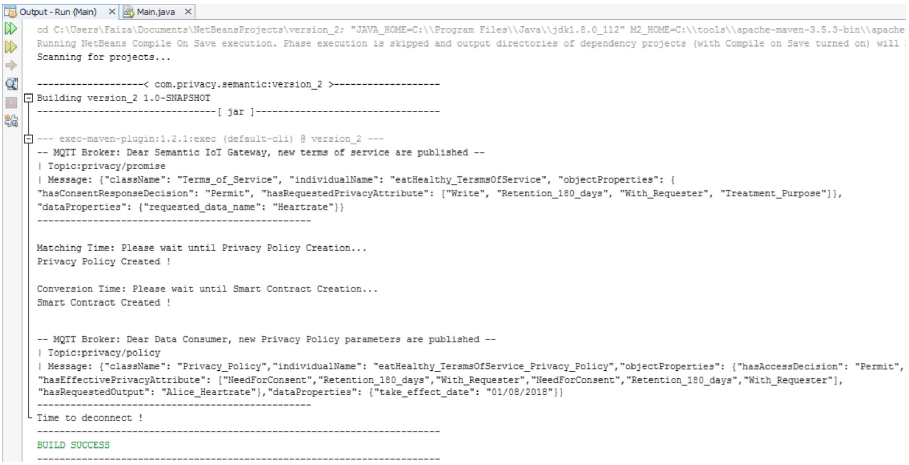

Fig. 7. Prototyping of the Publish/Subscribe process

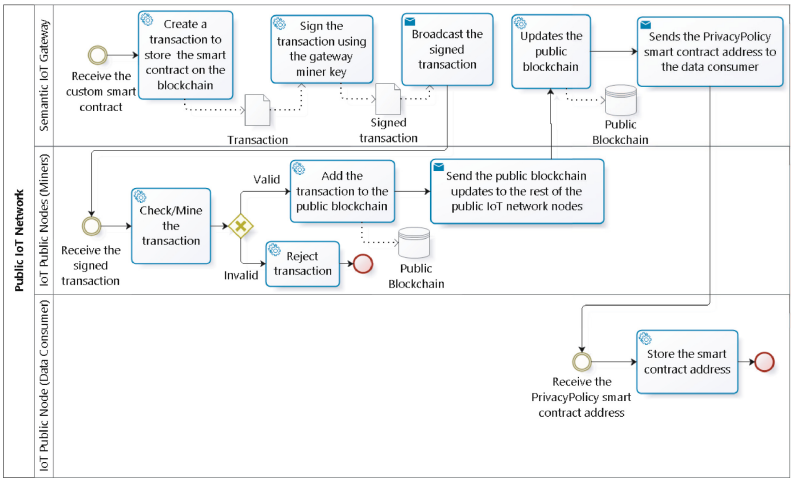

Fig. 8. Storing the *PrivacyPolicy* smart contract on the public blockchain

7 Conclusion

The blockchain technology is a distributed database that records all the transactions that have ever occurred in the network. The main feature of the blockchain is that it allows deploying smart contracts. In fact, a smart contract is an executable code that runs on top of the blockchain to facilitate, execute and enforce an agreement between untrusted parties without the involvement of a trusted third party. Hosting a smart contract in the blockchain can enforce privacy-preserving in the IoT domain. For this purpose, we defined a Semantic IoT Gateway that acts as a bridge between the IoT sensors, actuators and the blockchain

network. For this purpose, we considered the Semantic IoT Gateway as a core component of our proposed end-to-end privacy-preserving framework for the IoT data based on the blockchain technology, called PrivBlockchain [9]. The main functionalities of our proposed Semantic IoT Gateway are: first, to match the owner's preferences and the consumer's requirements in order to infer a privacy policy using LIoPY, a European legal compliant ontology to preserve privacy for IoT as well as a set of inference rules [8]. Second, to convert the inferred privacy policy into a custom smart contract using a set of predefined set of functions written in the Solidity language. Indeed, the use of smart contract aims at enforcing the privacy requirements when sharing the IoT data. Our experimentation on a real-world use-case has given the expected results and the custom smart contracts are generated and added to the blockchain with success.

It is possible that the data consumer use or share the data without executing the smart contract. To overcome this problem, we intend to incorporate a set of penalties by proposing a new smart contract type. For instance, a payment would be automatic in case of breaking a contract by sharing data illicitly.

References

1. Eclipse paho. http://www.eclipse.org/paho/
2. Solidity language. http://solidity.readthedocs.io/en/develop/
3. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. Comput. Netw. **54**(15), 2787–2805 (2010)
4. Buterin, V., et al.: A next-generation smart contract and decentralized application platform. White paper (2014)
5. Celdrán, A.H., Clemente, F.J.G., Pérez, M.G., Pérez, G.M.: SeCoMan: a semantic-aware policy framework for developing privacy-preserving and context-aware smart applications. IEEE Syst. J. **10**(3), 1111–1124 (2016)
6. Hashemi, S.H., Faghri, F., Rausch, P., Campbell, R.H.: World of empowered IoT users. In: 2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI), pp. 13–24. IEEE (2016)
7. Landman, D.: Arduino Library for AES Encryption (2017). https://github.com/DavyLandman/AESLib
8. Loukil, F., Ghedira-Guegan, C., Boukadi, K., Benharkat, A.N.: LIoPY: a legal compliant ontology to preserve privacy for the internet of things. In: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), pp. 701–706. IEEE (2018)
9. Loukil, F., Ghedira-Guegan, C., Boukadi, K., Benharkat, A.N.: Towards an end-to-end IoT data privacy-preserving framework using blockchain technology. In: 19th International Conference on Web Information Systems Engineering (WISE) (2018)
10. Maddox, T.: The dark side of wearables: how they're secretly jeopardizing your security and privacy. https://www.techrepublic.com/article/the-dark-side-of-wearables-how-theyre-secretly-jeopardizing-your-security-and-privacy/
11. Regulation, General Data Protection: Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46. Official Journal of the European Union (OJ), **59**, 1–88 (2016)

12. Spiekermann, S., Cranor, L.F.: Engineering privacy. IEEE Trans. Softw. Eng. **35**(1), 67–82 (2009)
13. Wang, S., Hou, Y., Gao, F., Ma, S.: Ontology-based resource description model for internet of things. In: 2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), pp. 105–108. IEEE (2016)
14. Zyskind, G., Nathan, O., et al.: Decentralizing privacy: using blockchain to protect personal data. In: 2015 IEEE Security and Privacy Workshops (SPW), pp. 180–184. IEEE (2015)