JEAS | Journal of Economics and Administrative Sciences
E-ISSN: 2148-1792 P- ISSN: 1302-2024
Volume 5, Supplement Issue 2
www.jeasweb.org

Privacy preserving protocol with trust model for IoT environments[1]

Zahra Dehrouyeh,
Master of Information Technology Engineering, Department of Computer Engineering, Faculty of Engineering, Alzahra University, Tehran, Iran

Reza Azmi Ph.D.
Associate Professor of Electrical Engineering- Electronic,Department of Computer Engineering, Faculty of Engineering,Alzahra University,Tehran,Iran

ABSTRACT

Purpose: *The utilization of IoT technology increases uninterruptedly. Many of the IoT applications are using individual information of users for providing services for them. One of the most significant challenges in IoT environment is security issues. If users can't trust IoT application, they won't have any motivations to use them. The most important security issues in IoT environments consist of confidentiality, integrity and privacy preserving. In this paper, a protocol has been proposed for privacy preserving in IoT environments using blockchain technology and smart contract concept.*

Methodology: *Blockchain provides a condition that non-trusting members can correlate with each other without a trusted intermediary. Smart contract is a computer code that executes the conditions of a contract itself. Smart contract helps us to exchange our valuable things in a transparent way without any need to a middleman.*

Findings: *The key issue in this work is a secure transmission of the shared key between two agents in IoT environment. The researchers have implemented a prototype of the proposed protocol using pyethereum, and evaluated its security via AVISPA tool.*

Originality/value: *Using the proposed protocol provides a variety of security benefits for developers and users, and provides a good level of security for sensitive data. Also a trust model has been defined according to the security levels of service providers in IoT application according to the transmitter's grade using defined computing model and its previous interacts with them in the proposed protocol.*

Keywords: Blockchain, Internet of Things (IoT), Privacy, Security, Smart Contract , Trust Model

[1] **Acknowledgement**

This work was supported in part by Cloud Computing and Value added services Lab (CCVL), Alzahra University, Tehran, Iran.

Introduction

Internet of Things (IoT) generally refers to scenarios where network connectivity and computing capability extend to individuals, wireless identifiable objects, sensors, sensor embedded-smart tiny devices and everyday items enabling these to generate, exchange and consume data with minimal human intervention (R. Das and I. Das, 2016; Rose *et al.*, 2015). The fundamental fact that IoT consists of a ubiquitous array of devices having sensing and actuating abilities being confined to the Internet depicts the scenario that the relationships between objects and people are tightly intertwined (Brandt, 2015; R. Das and I. Das, 2016).

Two important challenges for IoT technology are security and user privacy. The data collection may be out of the user's knowledge, and data transmitting may be in plaintext, since the massive collected data is shared among different departments (Yao *et al.*, 2015), which may be accessed by unauthorized users to cause serious problems or even be used to harm the owners of the data if no security restriction is made on it (Yao *et al.*, 2015).

Security solutions designed for IoT environments have to deal with heterogeneous IoT entities with various hardware specifications (Azmi and Kordian, 2011). In IoT, the most spread devices are usually resource- constrained devices because of their low cost (Azmi and Kordian, 2011).

In the IoT environment as we connect more devices to the internet, then the new opportunities for using from security vulnerabilities are increased. One of the usual security issues is confidentiality of information between devices, and if users are not sure that their communication devices are safe against misusing, so they can't trust global using of IoT. Such as traditional networks (wired or wireless), data security in IoT environments consisting of confidentiality, integrity, authentication and privacy. In IoT environments, because data is broadcast for dynamic storage and sharing, encryption is important to obtain confidentiality and privacy.

One of our goals of defining the new protocol for maintaining security and privacy in the environment of the internet of things (IoT) is providing maximum security for users' personal data with minimum computational costs and one of the reasons for using blockchain technology in this protocol is to benefits from the security advantages of this platform.

The studies on using blockchain technology in IoT environment have not merely for preserving privacy in the IoT environment and can have various applications. However, our purpose in our proposed protocol is to provide a plan to preserve the privacy of sensitive data of the users in IoT environment. Many of the previous studies using blockchain technology to maintain security and privacy have used the electronic money feature present in the blockchain context. Nonetheless, in our proposed protocol, regardless of the concept of electronic money and using blockchain platform for its exchange and doing financial transactions, we have used data exchange feature in this platform to reach our purpose.

Blockchain and Smart Contract

Blockchain

Blockchains have recently attracted the interest of stakeholders across a wide span of industries: from finance and healthcare, to utilities, real estate, and the government sector (Christidis and Devetsikiotis, 2016).

The reason for this explosion of interest is that with a blockchain in place, applications that could previously run only through a trusted intermediary, can now operate in a decentralized fashion, without the need for a central authority, and achieve the same functionality with the same amount of certainty (Christidis and Devetsikiotis, 2016).

Blockchain is a distributed database that keeps a list of growing records in the form of blocks, and keeps them safe against intermediary and review risks. Every block consists of a time stamped and has a link to the previous block. In fact, every block has an encrypted signature from previous block and obtains a safe record. Blockchain is resistant against data changing.

Blockchain is a new technology that flips the traditional model of a ledger upside down. Rather than having multiple separate silos, a blockchain (in its purest form) can act as a unified database that's accessible (on a read and write basis) by everyone (it is in effect "permissionless") (Diffenthal *et al.*, 2016).

The heavy use of cryptography, a key characteristic of blockchain networks, brings authoritativeness behind all the interactions in the network (Christidis and Devetsikiotis, 2016).

Why do we use blockchain in our proposed protocol?

We need a database in our proposed protocol to store collected data from sensors and then transmit them to the service providers. If we use a common database, some security techniques should be done on our data before putting them in the database. For instance, to preserve data confidentiality, integrity and privacy, we should use some techniques such as encryption and hashing, because databases aren't secure against hacking. Using blockchain technology as a database in our proposed protocol can eliminate the objections of usual databases. Using blockchain instead of common databases has some advantages, and the most important of them have been listed below:

- Common databases keep all data in a centralized, way and this matter increases the risk of hacking. But in blockchain technology, network is in a distributed shape, and hacking of it is impossible.
- With a blockchain, there is no need for a central trusted authority or for intermediaries (Diffenthal *et al.*, 2016). The disintermediation of intermediaries could redefine the value chain in a wide range of industries, from financial services to media, and puts the power and value of data back in the hands of the people creating that data (Diffenthal *et al.*, 2016). Blockchains can be public (such as the Bitcoin blockchain or the Ethereum blockchain) – these are effectively permissionless, or they can be private (where access is restricted to a selected group of users) (Diffenthal *et al.*, 2016).
- Blockchains are weighty, nonrepudiation, faster, cheaper and more secure than traditional systems, and because of this, banks and governments have tendency to use it.
- Blockchains are designed as secure and a sample of distributed systems. A decentralized consensus can be got by a blockchain. This makes the blockchain suitable for recording events, medical records, management, recognition, transaction process and document's source activities.
- Blockchain is a distributed network that surveys the transaction's integrity and balance of relative accounts, so it makes impossible the most intelligent attacks. Blockchain technology is apparent, and transactions aren't changeable.
- Blockchain is resistant against modifying and data changing, so integrity of storing data in the blockchain is guaranteed. Therefore, by using blockchain instead of usual databases, there is no need to use security techniques for data integrity, and the performance is improved.

In fact, blockchain obtains more security for storing data in comparison with usual databases because of its security features, so the need of using security techniques is decreased, and the performance is also increased.

Smart Contract

Smart contracts _self-executing scripts that reside on the blockchain_ integrate these concepts and allow for proper, distributed, heavily automated workflows (Christidis and Devetsikiotis, 2016). This should

make blockchains enticing to researchers and developers working in the Internet of Things (IoT) domain (Christidis and Devetsikiotis, 2016).

Because of Turing-completeness of in-built contract programming language, and this reality that computations are executed on every node of network, a program can make an intolerable loop, for instance a contract can paralyze a network. For preserving against this issue, the programmable computations in Etehereum is executed by units of gas. If a transaction doesn't have enough gas for executing, it will be failed automatically.

Despite the expressiveness and power of the blockchain and smart contracts, the present form of these technologies lacks transactional privacy (Kosba *et al.*, 2016). The entire sequence of actions taken in a smart contract is propagated across the network and/or recorded on the blockchain, and therefore it is publicly visible (Kosba *et al.*, 2016).

Related Works

There are different privacy preserving methods used in IoT environments. In Li and Cao (2012), a privacy preserving data aggregation scheme based on homomorphic encryption for sensor data collection by an untrusted aggregation was proposed. In Evans and Eyers (2012), the use of data tagging is proposed. The scheme controls the flow of information based on the tag it received at creation time.

In Sicari *et al.* (2015), a lightweight privacy-preserving trust model was proposed based on the observation that a large class of applications can be provisioned based on simple threshold detection. The main algorithm in this proposed model has been an uniformization scheme that uses a combination of sensor aliases to hide the identity of the sensing source and perfunction initialization vector to reveal information only to relevant service providers. This proposal has some advantages. First, processing on sensor is the least and it can be easily implemented on resource constrained devices (Appavoo *et al.*, 2016). Second, since the data store and service provider are separate entities, service provisioning is now open to external third party providers, and a user have the flexibility to dynamically configure the service he/she wants (Appavoo *et al.*, 2016).

The researchers of this study used the proposed model in Appavoo *et al.* (2016).

In Kosba *et al.* (2016), a blockchain model of encryption and smart contract for privacy preserving was proposed. Hawk is a decentralized smart contract system that doesn't store financial transactions visible on the blockchain, so it preserves transactional privacy. A Hawk programmer can write a private smart contract in a direct way without any encryption, and then the existent compiler creates an encryption protocol where the participants of the contract interact with blockchain automatically using encryption preliminaries such as zero-knowledge proofs. In fact, Hawk is a platform for composition of privacy preserving smart contracts.

In Ouaddah *et al.* (2017), it was how blockchain, the promising technology behind Bitcoin, can be very attractive to face those arising challenges. Therefore, it proposed 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.

Proposed Protocol

The main purpose of this proposed protocol is to use blockchain technology and smart contract for safe transmission of a shared key between two elements in IoT environments. Suppose that we have several sensors in IoT environment and want to transmit the collected data from sensors between a transmitter and a receiver in a way that preserves the privacy of data. For instance, one of the utilization of this protocol is in the healthcare applications that patient's data is collected using sensors and transmitted to the doctor; in this scenario, the privacy preserving is a very important issue.

There are two general stages prior to transmitting key data concerning the shared key: generating the shared key and transferring it securely.

First, shared keys should be generated according to the data that is collected from sensors and then the data should be encrypted with these shared keys. The authors used the proposed model in Appavoo *et al.* (2016) for generating shared keys but they used blockchain technology for exchanging the shared keys between transmitter and receiver. In fact the main idea, in this protocol has been using blockchain for secure transmission of shared keys.

In our proposed protocol for scenarios in IoT environment, we consider transmitting and receiving agents as users of blockchain and build some blocks for them.

There have been three main agents in this protocol: transmitter, administrator and receiver. Every agent has a block in blockchain and every block has two keys.

This helps enhance security too as the keys produced by each block are produced by blockchain itself and would not be able to be hacked. The blocks public key is used to address them in transactions and the private key is used to sign the sent message.

The authors used their keys for transmission of data in blockchain. Because of the transparency feature of blockchain, important data of confidentiality should be encrypted before inputting them in the blockchain.

Figure 1 illustrates a totality image of this protocol.

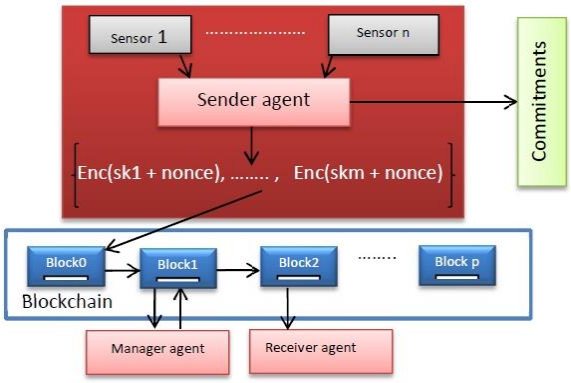

Fig .1.Image of proposed protocol

The proposed protocol has some summery symbols. They have been described in Table 1.

Table 1. Summary symbols of the protocol

Summary symbol	Description
X	**Reading data from sensors**
F	**Function/condition**
P	**The probability function**
Maxaliases	**Maximum number of sensor aliases**
Num-aliases	**Number of sensor aliases**
Alias	**Sensor alias**
Cx	**Index of encrypted outcome from the function**
Ivx	**Index of initialization vector**
Sk	**Shared key**
Nonce	**Nonce number for every shared key**
S	**transmitter agent**

M	**administrator agent**
R	**Receiver agent**
Ksr	**Shared key of transmitter and receiver**
Ksm	**Shared key of transmitter and administrator**
krm	**Shared key of receiver and administrator**

Shared key generation

The authors used the proposed model in Appavoo *et al.* (2016) for generating shared keys. The main motivation in Appavoo *et al.* (2016) was that the great classes of applications are obtained based on the recognition that sensor's data are more than some threshold values or not. This model has three main elements: sensor, service provider, database (insecure). The sensors are configured beforehand to generate triggers according to sensor values passing thresholds. The sensor transmits information to an uncertain data store whether triggers are activated or not. This database just stores data and can't do any computation. Ultimately, a user commits to various services by telling related servers where to get the data and how to extract the trigger information. In the proposed protocol of this study, the authors used blockchain instead of insecure database for data storage and secure transmission of generated shared keys according to the sensor's collective data. First, the data were gathered from sensors, some functions were applied on them, and the probabilities of every function for every sensor were calculated. Every sensor has a MAX_ALIASES parameter, and depends on the resources available on the sensor platform (Appavoo *et al.*, 2016). Then, according to the probabilities of functions on a sensor and MAX_ALIASES of it, the number of alias names for that sensor is calculated. Generating alias names from sensor readings process has been shown in figure 2. The idea of using alias names is for user's privacy preserving. We then collect the number of aliases of all physical sensors and calculate the total number of aliases for a given function. Finally, if we assume that is an alias for function i, j will generate a sk_{ij} shared key for each pair of function i and j. The generated sk_{ij} involves three parameters - $alias_j$, cx_{ij} and ivx_{ij}. $Alias_j$ is the alias in question obtained randomly. Then, a shared key is generated as a flow for every couple of a function and an alias name:

$SK[i][j] = (alias_j , cx_{ij} , ivx_{ij})$

ivx is generated randomly, and cij = fi(x)xor… (the XORed outcome of ivij individual bits). The number of bits in the iv must be at least 2, so as to ensure that cij is equally likely to appear as 0 or 1, whether fi(x) is either 0 or 1 (Appavoo *et al.*, 2016)

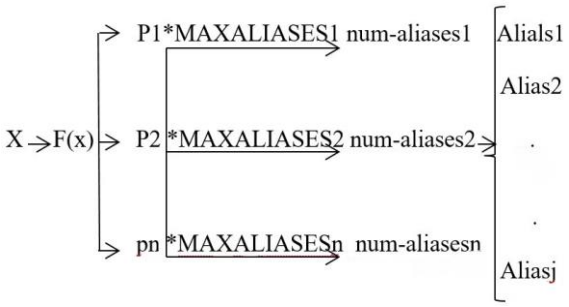

Fig.2.Generating alias names from sensor readings process

The number of alias names for every sensor depends on two parameters:

1. Probability of being true of a function for received data from a sensor
2. MAX_ALIASES constant for that sensor

Ultimately, sk_{ij} is obtained. This generated shared key has to be first exchanged between the transmitter and the recipient, which is the server, and then used to transmit user data. Hence, the secure exchange of this shared key guarantees the security of the user's personal data as well.

Shared key transmission via blockchain

In blockchain, a block is created for every user or entity in the IoT environment, and exchanging between users via blockchain is done between their blocks. Smart contracts are used for data transmission between blocks in blockchain.

We consider three main factors in our proposed protocol: 1) transmitter, 2) administrator, and 3) receiver.

Two main paths exist to exchange a shared key, and the security of the transmitted data has to be fully maintained:

The path between transmitter and administrator agents

This path has two parts. The first part is the transmission of data from a transmitter agent to the blockchain and putting it in the block of transmitter in the blockchain. This part of path is out of blockchain, and can't use security advantages of blockchain. For instance, data integrity isn't guaranteed in this part, so some techniques are needed for this issue. The technique that we use for data integrity is using hash function.

A nonce number is generated randomly for every shared key. A data commitment is generated for every couple of shared key and its nonce for verifying the integrity of transitive data, and it will be opened for all agents. The certificate value is obtained using a hash function. We have used sha3-256 function to generate a hash in the implementation. This function is a rapport function between transmitter and administrator agents that can be used with any other hash function. Data should be encrypted before inputting in the block of bockchain, and then the encrypted data will be transmitted between blocks. The data in the first part of the first path from the transmitter to the corresponding block in the blockchain is encrypted with shared key and nonce values.

The significant point concerning blockchain is that due to its transparency, all the transactions and data exchanged between the blocks in the blockchain can be seen by all users, which is in contradiction with preserving the confidentiality of data sent. Thus, a solution must be provided to this. Thus, to protect the security of data that is important for privacy and which they wish to exchange through the blockchain, they must be encrypted and encrypted in the corresponding block before being placed on the blockchain. Thus, during the transactions that transmit data between the blocks even though all users and members of the blockchain can see the data they are not aware of the original data content as they do not have the decryption key, which will preserve data privacy.

Likewise, we encrypt the shared key and the nonce in the first path and its first part prior to placing it on the blockchain and place its encrypted form in the block for the transmitter in our proposed protocol.

Shared key and nonce encryption

The data is encrypted and sent in the first path between the two agents - transmitter and administrator - so that only the administrator can decode them. We have used AES symmetric algorithm for encryption. We have used the private keys of their respective blocks on the blockchain to generate the shared key between the two transmitting and administrating agents, and obtained a shared key for secure data exchange between them using Diffie-Hellman algorithm.

Diffie-Hellman algorithm is used to create a secure data exchange channel between two points. Using this algorithm is based on the premise that both sides have a private value, based on which some values and constants are considered in the algorithm without the two parts being aware of the other's private values and they obtained an agreed value as a shared key. We have used the blocks private keys as private values for both parties.

Then the shared key generated by Diffie-Hellman algorithm is used as the symmetric key of AES encryption algorithm. The transmitter agent encrypts the shared key and nonce key parameters with AES algorithm and encrypts its own key with administrator, and then places the encrypted data on its own block in the blockchain.

The data in the transmitter block is sent to the block address of the administrator via a transaction as a data field.

Now, the administrator agent has all the shared keys and their associated nonce as encrypted. It first decodes the received data using AES symmetric algorithm and its shared key with the transmitter based on their own private keys and Diffie-Hellman algorithm. For examining the validity of the received data, it must then calculate the value of the certificate for each pair of shared keys and its corresponding nonce using the hash function. The transmitter and administrator agents already agree on a specific hash function to calculate the certificates of authenticity. As already stated, we have used sha3-256 function here. Then the administrator agent compares the calculated hash values with the values in the certificates of accuracy that the transmitter calculated and provided to all agents. If the compared values are equal, then the received data is maintained and the data is correctly managed by the administrator. This step is done once for every set of collected data from sensors, and the administrator will have the shared keys.

The path between administrator and receiver agents

In this path, transfers happen between the blocks related to the administrator and the recipient, and as it is entirely in blockchain platform, the accuracy of the data transmitted is fully guaranteed. Thus, there is no need to transmit a nonce for each shared key sent by the administrator and calculate the verification certificates by the recipient for verification of the received data. Moreover, the only thing that is sent from the administrator to the intended recipient is the shared key as a data field as transactions.

However, the point is preserving the confidentiality of the shared key sent by the administrator to the recipient that is why there is no guarantee in blockchain for preserving the confidentiality of important transactions and sent data. Thus, the data must be encrypted transacted in an encrypted way in this path.

In several scenarios of IoT environment, there will be several servers that have the role of recipients in our protocol. Various services may only need certain servers to have access for the data received from the sensors. Indeed, only some servers receive a shared key and can exchange data directly with the transmitter. Thus, after the administrator receiving and verifying the authenticity of the shared keys and wanting to transmit them to various recipients, it should examine to see which particular recipient is authorized to receive the shared key and transmit the shared key to that particular recipient in way that the confidentiality of it is preserved from others. Thus, the solution to this, like the first path, is to use cryptography. As in the first path, AES symmetric algorithm is used in this path. AES algorithm key is obtained using Diffie-Hellman algorithm as well and based on the private keys, the blocks related to the intended administrator and recipient are obtained and thus, the secure channel for secure data transfer between two factors emerges.

Every receiver that wants to have a shared key or we want to transmit a shared key to it just need the compromise between the administrator and the receiver for a symmetric key according to their block's private keys, and then the administrator encrypts the shared key with symmetric key and transmits it as a data field of a transaction to the address of the receiver's block. The reason of encryption of shared key is confidentiality, since maybe in a special scenario just some receivers should have the shared key, and in blockchain, a sent transaction appears for all members which is not coherent with confidentiality.

The important point is that transmitting data in step of transmitting shared key from administrator agent to the receivers is just a shared key and doesn't need to transmit a nonce for every shared key, because blockchain guarantees the integrity of transmitted data. This point can improve the performance of the system, since the cost of computation of commitment per receiver is omitted. In many scenarios, the shared keys transmission between transmitter and administrator is done just once, but between administrator and receivers it should be done many times so this matter can be very useful for the improvement of performance.

When a receiver gets the shared key, decrypts it, and afterwards, it can exchange data with transmitter agent directly using a received shared key. Replay attack is prevented by using blockchain technology, smart contract is used for data transmission in blockchain, and the amount of gas is defined for every smart contract. The gas is a cost for doing that smart contract. Every transaction for doing its operation should call a smart contract and also there must be enough gas for doing that operation, otherwise that transaction is known as invalid. Using gas concept in blockchain prevents replay attack, because every transaction has determinate amount of gas and also can be executed for a bound round.

Implementation

In this study, Ubuntu 14.04 operating system was used as a virtual machine in vmware workstation pro 12 for implementation of the proposed protocol.

For blockchain technology, Ethereum was used. Ethereum is a decentralized cryptocurrency that uses its built-in currency, Ether, as the fuel to power the programmable "smart contracts" that live on its blockchain (Delmolino *et al.*, 2015). Think of a "contract" as a program that provides services such as: voting systems, domain name registries, financial exchanges, crowdfunding platforms, company governance, self-enforcing contracts and agreements, intellectual property, smart property, and distributed autonomous organizations (Delmolino *et al.*, 2015).

For using Ethereum on a virtual machine, pyethereum and serpent tools are needed. Pyethereum is the program that allows interacting with the blockchain and testing the contracts (Delmolino *et al.*, 2015). Serpent 2.0 will allow compiling our serpent code into the stack-based language that is actually executed on the blockchain (Delmolino *et al.*, 2015).

All operations in agents such as encryption, decryption, and hash functions are implemented using python language.

The authors built a block on the blockchain for every agent and used utils in ethereum to attach a public and a private key to every block. Public key of a block is used for addressing in transactions. The researchers should state GAS_LIMIT, STARTGAS and GASPRICE parameters for preservation against replay attack in blockchain. Then, in pyethereum tester, a chain was defined and the smart contracts were written via serpent language. The defined smart contracts were called in the functions, and transactions were done by them in the blockchain. The defined smart contracts were for creating a new block, the transmission between two blocks, and getting value and data fields of a block. The main functions of the protocol were as follow:

- Makesk: was done via transmitter agent and its outputs were shared keys, nonce and commitments. The hash function for production of commitments was sha3_256 and was compromised between transmitter and administrator agents. Also the commitments output of makesk function were visible for the administrator agent.
- Generatekey: converted the private key of blocks of agents into integer. After that, these integer values were used in diffi function for the generation of a shared key between transmitter and administrator agents.
- Difii: created a shared key for every two agents using their private keys.

- Encsk: encrypted transmitting items using AESCipher.
- Sendingitem: transmitted encrypted items between blocks of agents using smart contracts.
- Decsk: decrypted the received items using AESCipher function.
- Checkcommitment: was done just in an administrator agent, and computed the commitments of received items using adaptive hash function to verify the integrity of them.
- Checksk: confirmed that the received items were accurate.
- AESCipher: was a class that consisted of __init__(), encrypt(), decrypt(), pad() and unpad() functions. init () was for calculating theshared key of AES algorithm. Encrypt() and decrypt(0 functions were respectively for encryption and decryption of data using shared key of AES algorithm. CFB mode was used in foe encryption and decryption operations by AES symmetric algorithm. Sometimes the size of input data wasn't suitable for AES algorithm, and padding operation was needed, so pad () and unpad() functions were used in encryption and decryption operations; respectively.

The encsk function was done in transmitter and administrator agents, and decsk function was done in administrator and receiver agents. Computing and verifying the commitments were done just in the administrator agent, hence in the transmission between administrator and receiver, just the shared keys were sent, and there was no need for nonce. The pseudocode of makesk, sendingitem, encsk and decsk functions have been shown in Figures 3, 4, 5, 6.

```
for all sensors
   for all x  from 1 to n
      num-aliases = =p(f(x)=1)*MAXALIASES
totalaliases  ==numaliases1 + ... + numaliasesn
for all aliasname from 0 to totalalises
   alias = randint()
   ivx = randint()
   cx = ivx xor f(x)
   sk = alias , cx , ivx
   nonce = makenonce()
   datacommitment = sha3_256(sk , nonce)
return (sk , nonce , datacommitment)
```

Fig.3.The pseudocode of makesk function

```
for all item in encsk
   x.senddata(public key of sender's block , public key of destination's block , item , startgas)
   receiveditem = x.ask(public keyof a block , startgas)
return (receiveditem)
```

Fig.4.The pseudocode of sendingitem function, senddata and ask are the functions that are defined in the smart contract via serpent language.

```
aes  = AESCipher(key)
for all sk
   encsk  = aes.encrypt(sk)
return(encsk)
```

Fig.5.The pseudocode of encsk function, AESCipher is a class that consists of encryption and decryption functions via AES algorithm. The key is generated by diffi function.

```
aes = AESCipher(key)
for all encsk
  decsk = aes.decrypt(encsk)
return(decsk)
```

Fig.6.The pseudocode of decsk function, AESCipher is a class that consists of encryption and decryption functions via AES algorithm. The key is generated by diffi function.

In the blockchain, users use a pair of public and private keys to interchange with blockchain. The private key is used for signing the transactions, and public key is used for addressing. Using asymmetric encryption brings confidentiality and integrity preserving.

When using insecure database instead of blockchain, there is no guarantee that the transmitted data in the path isn't changed, and integrity isn't guaranteed. But blockchain itself is resistant against modifying of data, and the transmitted data will be safe.

According to the experiment result and comparing two states of using insecure database and blockchain technology, it can be concluded that in the first state, the probability of unsafe transmission of data exists, and some techniques should be used for the certainty of the correctness of data. One technique is using hash function. In this method, transmitter should generate a commitment for every transmitted data and its unique nonce number using a compromised hash function. Also, receiver should compute commitment for every data and its nonce number, and then compare them with the transmitter's commitments. If the commitments are equal, the integrity of data is insured. But by using blockchain, there is no need to transmit nonce numbers for every data and compute hash values, and this issue improves the performance.

The reason of integrity preservation in the blockchain is that the verified and signed data are stored in the blocks. And data for exchanging between blocks are signed by private keys, so there is no risk for invention of data.

There were two main paths in this proposed protocol, one path was between transmitter and administrator agents, and the other was between the administrator and receiver agents. In the first path, since a part of the path was out of blockchain, so hash function and commitment technique should have been used, but in the second path, because of using blockchian, there was no need to compute hash values, and this matter could improve the performance. It could be concluded that using blockchain in the proposed protocol can improve the performance of integrity in the system.

Evaluation

The evaluation of our proposed protocol is done in two stages:

- Examining the validity and reliability of the proposed protocol
- Optimality of the proposed protocol compared to previous methods and its advantages

Evaluating the performance accuracy and security of the proposed protocol

In this study, AVISPA tool was used for evaluating the performance accuracy and security of the proposed protocol. AVISPA denotes Automated Validation of Internet Security Protocols and Applications (The AVISPA Team, 2006). By this tool, the protocol was simulated and the data transmission was checked

in it, also some attacks could be simulated for that, and its safety against threats was examined. The other option of this tool was simulation of the intruder in the network and the resistance of protocol to it.

In this study, for using AVISPA tool, SPAN tool was installed on the virtual machine. AVISPA tool obtained a set of programs for formal models of constructing and analyzing security protocols. Protocol models have been written in HLPSL language.

For evaluation of the protocol in AVISPA, there were three main agents consisting of a transmitter, an administrator and a receiver. And the main purpose of the protocol was that the shared key between transmitter and receiver remained secret. First, the shared key of the transmitter and receiver was encrypted by the shared key of transmitter and administrator, and exchanged between the transmitter and administrator agents, the administrator decrypted it, and then, it was encrypted by the shared key of itself and receiver, and was sent to the receiver. The messages in cas+ were as follows:

1. T -> A: {Ktr}Kta

2. A -> R: {Ktr}Kra

The protocol was written by hlpsl language, and simulated via span tool. Attack simulation should have been done to verify the security of the protocol. The AVISPA Tool comprised of four back-ends: OFMC, CLAtSe, SATMC, and TA4SP. OFMC and ATSE tools were used to approve the security of the protocol. As can be seen in figure 7, the tool called the OFMC, it can be seen that OFMC found no attacks. In other words, the stated security goals were satisfied for a bounded number of sessions as specified in the environment role. The goal that was defined for the protocol was that Ksr was kept safe, and the results showed that it was reached.

```
SUMMARY
 SAFE
DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
 /home/reyhane/avispa/span//testsuite/results/c.if
GOAL as specified
BACKEND OFMC
STATISTICS
 TIME 32 ms
 parseTime 4 ms
 visitedNodes: 21 nodes
 depth: 5 plies
```

Fig.7. Result of attack simulation of protocol via OFMC

Attack simulation was also done by ATSE tool. ATSE was used with both Depth First and Breadth First algorithms, and in two conditions the protocol was safe. The results of simulation by ATSE tool have been shown in Figures 8 and 9.

```
SUMMARY
 SAFE

DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL

PROTOCOL
 /home/reyhane/avispa/span//testsuite/results/c.if

GOAL
 As specified

BACKEND
 CL-AtSe

STATISTICS

 Analysed  : 2 states
 Reachable  : 2 states
 Translation: 0.01 seconds
 Computation: 0.00 seconds
```

Fig.8. Result of attack simulation of protocol via ATSE by Breadth First algorithm

```
SUMMARY
 SAFE

DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL

PROTOCOL
 /home/reyhane/avispa/span//testsuite/results/c.if

GOAL
 As specified

BACKEND
 CL-AtSe

STATISTICS

 Analysed  : 2 states
 Reachable  : 2 states
 Translation: 0.00 seconds
 Computation: 0.00 seconds
```

Fig.9. Result of attack simulation of protocol via ATSE by Depth First algorithm

For intruder simulation, when the protocol was defined in hlpsl file, the measure of information should be determined that can have providing existence in the network. And then, in the simulation, it was examined that the intruder accesses information more than defined limitation or not. The result of intruder simulation for the proposed protocol has shown that even if any intruder is existence in the network, it can't access to the information more than the level that has been determined for it, and it means that the protocol is resistant against intruder.

Optimization of the proposed protocol over previous methods and the advantages of using it

The security advantages of the proposed protocol and replacing blockchain instead of common and unsecured database

Using the protocol presented in IoT environments provides a variety of security benefits for developers and users, and provides a good level of security for sensitive data against users. The most significant security advantages of the proposed protocol are as follows:

- Sensor data anonymity: Given the uniformization process and using anonymity aliases the data received from the sensors is provided and the main source of data cannot be identified. This issue is very important for preserving the privacy of users. For instance, in scenarios like a patient health screening program not disclosing user personal information and hiding the source of the received data.

- Avoiding inference attack: Using an uniformization approach, the probability of triggering by any of the sensors is equal, regardless of their values. Moreover, the number of sensor aliases is expected to increase significantly compared to the number of physical sensors.

- Unhackability of blockchain: Blockchain is virtually unhackable given the distribution and dispersion of data between various nodes that enhances the security of our proposed protocol.

- Maintaining data integrity: To ensure data integrity in the transitional path, the hash function technique has been used outside blockchain to generate a certificate of authentication for the transmitted data that has verified and validated the data when received. Inside the blockchain, this technology is inherently resistant to data change that will ensure that the data exchanged within the blockchain remains unchanged and guaranteed. However, in a common and unreliable database, it

is not guaranteed that the data will not be altered or tampered with by the transmitter in question. Indeed, data integrity will not be certain, so we must take measures to verify the accuracy of the data received. This will increase the computational overhead.

- Preserving confidentiality and preventing spoofing: From among the three key elements defined in the protocol, we have used the encryption technique to exchange the shared key. This is necessary even in the part of the transmission path done in the context of blockchain as it is transparent and the transactional data is visible to the public in the form of transactions. In this protocol, we have used AES symmetric algorithm for encryption. Using this encryption technique preserves confidentiality and prevents spoofing of the transmitted data, as no one except the two parties have the symmetric key relationship of the AES encryption algorithm according to the private keys of their respective blocks in the blockchain and using Diffie-Hellman algorithm. Thus, no one can decode or modify the exchanged data.

- Prevention of replay attack: Using blockchain technology, a concept called smart contract is used to transfer data across the blockchain environment, where some gas is defined as the cost of executing that contract for each smart contract. Any transaction intending to happen in blockchain environment to perform actions such as data transfer between blocks must call for smart contracts and have sufficient gas to do so; otherwise, that transaction will be considered as invalid. Using gas concept in blockchain will virtually prevent a replay attack and stop events like creating an infinite loop and disruption of the network, since each transaction has a limited amount of gas and a limited number of applications. Non-use of blockchain may expose the transmit data to the malicious person in the path between the transmitter and the receiver and be transmitted several times to the recipient. This can disable the recipient, which is actually our target server, by transmitting large volumes of data or disrupting the entire network.

- Prevention of man in the middle attack: this attack cannot happen because of using encryption technique in transferring the shared key between the agents defined in the protocol.

- Non-repudiation: the transmitted data is exchanged as transactions between various blocks in blockchain. These transactions are signed by the origin block that prevents the problem of non-repudiation.

- Lack of risk of database hacking: Common and insecure databases have the risk of being hacked and may get controlled by malicious people compromising the users' confidential data. However, the blockchain is practically unhackable given its distribution and dispersion properties between various nodes.

Examining the computational overhead of the proposed protocol and its advantages

Given the use of blockchain technology, data integrity is guaranteed. Moreover, in the section of the data transfer path done in blockchain context, there is no need for any techniques to generate data certification and verifying the integrity of the received data, which drastically reduces computational overhead and enhances system performance.

Let us imagine we have n servers and want to transmit the data collected from a set of sensors to them. The first step is to transmit a shared key generated based on the data collected. We study this scenario in two various states using the conventional method and applying the proposed protocol and compare the results.

First case: Using the common method and using uncertain data storage to share a shared key: In this case, for each generated shared key and transmitted to the desired server, the shared key must first be sent to the untrusted data storage, where the following operations have to be conducted:

• Cryptography operation: As the path between sensors and data storage is uncertain and no security is guaranteed, encryption must be done on the exchanged data to preserve its confidentiality.

Applying hash function: As the communication platform between the sensors and the data storage is unreliable and the data can be modified and manipulated along the way, it is randomly assigned to the generated key in the transmitter of a nonce number. We then obtain a certificate value based on the value of the shared key and its nonce using a hash function. This way the recipient side makes sure that the received data is the same as the transmitted data.

The significant point in this approach that leads to computational overhead is that as unreliable data storage has no guarantee for the secure storage of the received data, a separate shared key has to be sent for every server that needs a shared key. Indeed, the encryption process is applying the hash function and transmitting the shared key to the unreliable data storage must be done separately for each server. Moreover, in scenarios where the number of servers is high, this leads to a high computational overhead and lacks the needed efficiency.

In the second part of the path, from unreliable data storage to servers, as the transmission path is unreliable, the shared key again needs encryption and applying the hash function.

It is clear that when encryption is done at the start of a transient path, decryption should be done at the destination. Moreover, hash function computation process has to be done at the origin and destination.

Second case: Using the proposed protocol and applying blockchain technology to transfer the shared key: In this case, we consider three main agents and with a block considered for each in the blockchain. Servers are considered as recipients and if we have several servers, one block is considered for each. The shared key generated according to the data received from the sensors has to be sent to the transmitter agent first. As transmitting is outside the blockchain in this path, data exchange requires encryption and certificate validation based on the hash function. Now the desired data is in the blockchain transmitter block.

Given the transparency of blockchain, and as the confidentiality of the transmit data is important, data must be encrypted in the blockchain context. Encryption and authentication must be done using the hash function to transmit data from the transmitter to the administrator. The block related to administrator actually has the role of unreliable data storage, except that it ensures the security of the received data.

The significant point is that in this case, if we have n servers, we do not need to transmit a separate shared key as was the case in the previous case. However, it only transmits a shared key once to the administrator and the administrator transmits it to as many servers as needed. This is because the administrator is on the blockchain platform and guarantees that data security is preserved.

Moreover, if blockchain technology is used as the public and private keys of the blocks used in cryptography operation are generated by the blockchain itself, they provide much higher security than usual.

When the administrator has received the shared key, it has to transmit it to the appropriate servers. In the path between the administrator and the servers, only the cryptography is done and there is no need to apply the hash function and the certificate authentication computation that diminishes a lot of computational overhead.

The computations needed for the two common methods and the proposed protocol and comparing them for n servers:

Common approach:

Transmitting the shared key from the sensor to unsafe data storage: n Enc + n Hash

Receiving the shared key in uncertain data storage: n Dec + n Hash

Transmitting the shared key from uncertain data storage to the servers: n Enc + n Hash

Receiving the shared key on servers: n Dec + n Hash

Total: 2n Enc + 2n Dec + 4n Hash

The proposed protocol:

Transmitting the shared key from the sensor to the transmitter: Enc + Hash

Receiving the shared key in the transmitter: Dec + Hash

Transmitting shared key from transmitter to the administrator: Enc + Hash

Receiving the shared key in the administrator: Dec + Hash

Transmitting the shared key from the administrator to the recipient agents (servers): nEnc

Receiving a shared key in the receiving agents (servers): nDec

Total:

(n+2)Enc + (n+2)Dec + 4 Hash

As is seen, the computation is reduced in case of using the proposed protocol and applying blockchain technology instead of storing uncertain data that shows the proposed protocol is optimal.

Trust Model for Proposed Protocol

Suppose we have several service providers in our IoT scenario, and every ones gets different services. For some services, just one service provider is needed, and for some of them, multiple of them are needed. It was intended to define a trust model for service providers and get a number as their trust level according to their security level by using some special parameters. Trust level can be a range of numbers, and every number shows a special security level to us. For instance by increasing the number of trust level, the trustworthiness of those service providers increases.

There are two main states for getting services from service providers:

1. The state that only one service provider is needed for the service, in this state, the number of trust level of that service provider is examined and it is decided about the communication with it.
2. The state that a set of service providers are needed for getting a special service, in fact, it's essential to connect to the chain of service providers. In this condition, the trust level numbers of all service providers should be examined, and the minimum number of them should be calculated, after that, it can be decided about the security level of that set of service providers and communication with them.

For getting the trust level of a service provider, its previous communications should be used. As was stated, there are three main agents that have been in the proposed protocol, and service providers are as a receiver agent. The administrator agent is used for transmission of a shared key between transmitter and receiver agents. After that, the transmitter and receiver agents can interact with each other directly. Then, the transmitter agent gives a grade to every receiver according to its satisfaction. These grades have been stored in a table.

Then for every connection, the transmitter agent can check the grade's table of receivers, and according to them, it decides about its correlation with them. In this way, a transmitter can use from satisfaction of other transmitters before its communication.

This grading is done based on a computing model according to the past communications of users and service providers.

Grading has been done by some parameters. Four main parameters have been listed below:

1. Result: The result is the satisfaction that a peer receives in a transaction and shows that the other peer how much does its tasks well (Azmi and Kordian, 2011).
2. The number of results: This parameter is a prominent factor for the comparison of the results. A simple way of reputation calculating can be the proportion of total satisfaction numbers of a special peer to the total results of it that shows the average satisfaction from that peer in every transaction (Azmi and Kordian, 2011).
3. Result reliability: It's possible that a peer because of negative motivations or insufficient experiment gives a false result about the other peer (Azmi and Kordian, 2011). So, the result of a peer with higher reliability degree is more important than a result of a peer with the lower reliability degree (Azmi and Kordian, 2011).
4. Transaction background: Transaction backgrounds such as value, time, and negative grade coefficient are impressive elements in a transaction (Azmi and Kordian, 2011).

In this trust model, direct reputation was used in the computing model, it means that a transmitter only based on its communication and satisfaction from service providers can determine its trust level, and indirect reputation and others' opinions aren't used here.

In this method, the amount of trust of transmitter i to the receiver j in time t (now) has been defined as follows:

T(i, j, t)

In the first transaction of i with j, the amount of its trust based on direct reputation is zero (Azmi and Kordian, 2011). But after the first transaction, the amount of trust based on the direct reputation is gotten by equation (1), in this equation, Z is a total number of transactions of i with j and $M_z(i, j, t)$ is the grade that i gives to j after the transaction (Azmi and Kordian, 2011). C(z) is the transaction background factor that affects negatively on grade coefficient and transaction size in the equation (Azmi and Kordian, 2011). Negative grade coefficient is a number more than 1 that makes the influence of transactions with a negative result (Azmi and Kordian, 2011). So, the influence of negative manner is more than the positive manner (Azmi and Kordian, 2011).

$$T(i, j, t) = \frac{\sum_{z=1}^{z} M_z(i, j, t) * C(z)}{Z} \quad (1)$$

The cost of grade that i gives to j is decreased by passing the time, for example a negative grade that was given 2 months ago isn't as valuable as now, so this influence of time should be calculated and the cost of grade is updated. Because of this, if $M_z(i, j, t)$ about j is old, it should be updated before Transmitting to the other users. This updating has been done by equation (2):

$$M(i, j, t) = N + (M(i, j, t_0) - N)\, e^{\frac{(t-t_0)}{r}} \quad (2)$$

In this equation, N is an inactive amount, both trusty and trustless (Azmi and Kordian, 2011). Time changing is exponential and t0 is the last time that Mz(i, j, t) was updated (Azmi and Kordian, 2011). And r is an experiment constant that determines the time that information is being invalid and as it's bigger, the

information is being invalid in a shorter time (Azmi and Kordian, 2011). The amount of the trust level for every service provider that is calculated by this computing model is used in the proposed protocol.

In the proposed protocol for every agent, there has been a block in the blockchain. And transactions have been used for the interaction between blocks. One of the fields that is existent in the transaction is the value. In this study, it was intended to use the value field for adding trust model to the proposed protocol and store the number of trust level of the receiver agents in this field.

Grading the transmitter agent to the receiver agents and storing these grades have been done by a table that consists of transmitter, receiver and grade fields. Transmitter and receiver agents were known by the address of their blocks in the blockchian. Then, the existent grade for every receiver agent was set as a value field in its transaction.

When a transmitter wants to connect with a receiver and needs to know its grade can read the value field of that receiver. A smart contract can be defined in the serpent_code part for reading of the value field of a block and then calling it in the transaction.

By this trust model in the IoT environment, the users can know about service providers, their grades and trust level before their communications, and then, they can decide about their connections and security techniques.

Conclusion

The proposed protocol in this study was about using a method for privacy preserving in IoT environments. In shared key generation step, because of using alias names and uniformization operation, the main origin of collected data is uncertain and it helps in privacy preserving issue. In the step of shared key transmission by blockchain, first, the shared key was encrypted and then it was put on the blockchain, thus the confidentiality of the shared key was guaranteed. Also, the blockchain ensured the integrity of transmitted data, so there was no need to verify the integrity of data and this point could cause decreasing the computation costs and improving the performance. The authors had also shown that the proposed protocol can be useful for security and privacy challenges of IoT environments through implementation and evaluation. By adding trust model to the proposed protocol, the users can know about service providers and their grades before their communications and then decide about connections and security techniques in the IoT environments.

References

Appavoo, P., Chan, M.C., Bhojan, A. and Chang, E.-C. (2016), "Efficient and Privacy-Preserving Access to Sensor Data for Internet of Things (IoT) based Services", paper presented at the 8th International Conference on Communication Systems and Networks (COMSNETS), 2016, place of conference, available at: URL

Azmi, R. and Kordian, M. (2011), "Trust model based on reputation in peer to peer electronic commerce society with ability of recognizing and opposition against malicious attacks", *Journal Name*, volume issue, pages.

Brandt, J. (2015), "50 billion connected IoT devices by 2020. Available", available at: http://www.smartgridnews.com/story/50-billion-connected-iot-devices-2020/2015-04-21 (accessed 12 November 2016).

Christidis, K. and Devetsikiotis, M. (2016), "Blockchains and Smart Contracts for the Internet of Things", *IEEE*, Vol. 4, pp. 2292-2303.

Das, R. and Das, I. (2016), "Secure Data Transfer in IoT environment: adopting both Cryptography and Steganography techniques", *Journal Name*, volume issue, pages.

Delmolino, K., Arnett, M. and Kosba, A. (2006), "A Programmer's Guide to Ethereum and Serpent", *Journal Name*, volume issue, pages.

Diffenthal, R., Lewis Rinaudo Cohen, L. and Maxwell, W. (2016), "Tree examples of blockchain smart contracts – Internet of Things, commercial paper and daos", available at: https://www.hlmediacomms.com/2017/01/10/three-examples-of-blockchain-smart-contracts-internet-of-things-commercial-paper-and-daos/ (accessed 12 November 2016).

Evans, D. and Eyers, D.M. (2012), "Efficient data tagging for managing privacy in the internet of things. ", In Green Computing and Communications (GreenCom)", in *IEEE International Conference on*, IEEE, 2012, pp. 244–248.

Kosba, A., Miller, A., Shi, E., Wen, Z. and Charalampos Papamanthou, Ch. (2016). "Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts*",* paper presented at the IEEE Symposium on Security and Privacy, 2016, place of conference, available at: URL

Li, Q. and Guohong Cao, G. (2012). "Efficient and privacy-preserving data aggregation in mobile sensing.", In Network Protocols (ICNP), in *20th IEEE International Conference on*, IEEE, 2012, pp. 1–10.

Ouaddah, A., Elkalam, A.A. and Ouahman, A.A. (2017), "Towards a Novel Privacy-Preserving Access Control Model Based on Blockchain Technology in IoT", in: Rocha, Á., Serrhini, M. and Felgueiras C. (eds.), *Europe and MENA Cooperation Advances in Information and Communication Technologies*, Advances in Intelligent Systems and Computing, vol. 520. Springer, Cham.

Rose, K., Eldridge, S. and Chapin, L. (2015), *The Internet of Things: An Overview; Understanding the Issues and Challenges of a More Connected World*, The Internet Society (ISOC).

Sicari, S., Rizzardi, A. and Grieco, L.A. (2015), "A Coen-Porisini, Security, privacy and trust in internet of things: The road ahead", *Computer Networks*, Vol. 76, pp. 146–164.

The AVISPA Team. (2006), "AVISPA v1.1 User Manual", Document Version: 1.1 June 30.

Yao, X., Chen, Zh. and Tian, Y. (2015), "A lightweight attribute-based encryption scheme for the Internet of Things", *Future Generation Computer Systems*, Vol. 49, pp. 104–112.