Using Smart Contract for Privacy Preserving in Internet of Things (IoT) Environments

Zahra Dehrouyeh and Reza Azmi

***Abstract*—The utilization of IoT technology increases uninterrupted. The sensors in IoT environments collect great amounts of data and give them to the service providers. This generous content of user's individual data for IoT applications introduces new security challenges. Indeed privacy preserving is a very significant issue for IoT applications. In this paper, we propose a protocol for privacy preserving in IoT environments using blockchain technology and smart contract concept. 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. The key issue in this work is a secure transmission of the shared key between a sender and a receiver using smart contract and blockchain technology in IoT environments. The secure exchange of shared key can result privacy preserving. We have implemented a prototype of the proposed protocol using pyethereum, and evaluate its security via AVISPA tool.**

***Index Terms*—Blockchain, Internet of Things (IoT), privacy, security, smart contract.**

I. Introduction

Internet of Things (IoT) generally refers to scenarios where network connectivity and computing capability extends to individuals, wireless identifiable objects, sensors, sensor embedded-smart tiny devices and everyday items (not normally considered computers) enabling these to generate, exchange and consume data with minimal human intervention [1], [2]. 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. Additionally, the deployed proportion of IoT devices will get doubled (50 billion devices) by 2020 is the forecast predicted by Cisco's Internet Business Solutions Group [2], [3]. Some examples of IoT applications lies in EHealth, Smart Transportation, Smart Home Experience, Environmental Monitoring, Precision Agriculture, Retail and Logistics and numerous such domains [2].

Two important challenges for IoT technology are security and user privacy. For instance, in the ubiquitous IoT application such as smart city, data is usually gathered from various sources owned by different administrative domains (e.g., smart phones, and public or private transportation providers) [4]. 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 [4].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 [4].

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 [5]. The reason for this explosion of interest: 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 [5].

Smart contracts _self-executing scripts that reside on the blockchain_ integrate these concepts and allow for proper, distributed, heavily automated workflows [5]. This should make blockchains enticing to researchers and developers working in the Internet of Things (IoT) domain [5].

The goal of this paper is to propose a protocol for privacy preserving in IoT environments using blockchain technology and smart contract. Our proposed protocol is using blockchain technology for transmission of shared key in IoT applications.

The rest of this paper is organized as follow. Section II presents related work, in Section III we examine what a blockchain is and how it works. We present our protocol and its specifications in Section IV and details of the implementation in Section V. Section VI presents evaluation of our implementation and we conclude in Section VII.

II. Related Work

There are different privacy preserving methods are used in IoT environments. In [6] a privacy preserving data aggregation scheme based on homomorphic encryption for sensor data collection by an untrusted aggregation was proposed. In [7] the use of data tagging was proposed. The scheme controls the flow of information based on the tag it received at creation time. Authors in [8] proposed a model where access to sensitive data is restricted to authorized users and devices.

In [9] proposed a lightweight privacy-preserving trust model 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 is 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 minimum and can

Manuscript received March 20, 2018; revised May 10, 2018. This work was supported in part by Cloud Computing and Value added services Lab (CCVL), Alzahra University, Tehran, Iran.

Zahra Dehrouyeh and Reza Azmi are with the Alzahra university, Tehran, Iran (e-mail: reyhane.dehrouyeh@gmail.com, azmi@alzahra.ac.ir).

doi: 10.18178/jacn.2018.6.1.257

be easily implemented on resource constrained device [9]. 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[9].

We use proposed model in [9] and its functions for generating shared keys in our proposed protocol. But instead of the data store that was used in [9] we use blockchain technology.

III. Blockchain and Smart Contract

Blockchain is a new technology that flips the traditional model of a ledger upside down. Rather than have 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") [10]. The ledger stored on a blockchain is shared amongst a distributed network of computers, and the use of cryptography enables users to modify the master ledger without the need for a central authority [10].

One of the best things about blockchain is that a decentralized system is among all permissive participants and there is no need for intermediates. Blockchain is a distributed database that consists of blocks and keeps a list of records and preserves them from intermediary risk. Every block has a public and private key that the public key is used for addressing in transactions.

The blockchain enables trustless networks, because the parties can transact even though they do not trust

Each other and the absence of a trusted intermediary mean faster reconciliation between transacting parties [5]. The heavy use of cryptography, a key characteristic of blockchain networks, brings authoritativeness behind all the interactions in the network [5]. Smart contracts _self-executing scripts that reside on the blockchain_ integrate these concepts and allow for proper, distributed, heavily automated workflows, and this matter should make blockchains enticing to researchers and developers working in the Internet of Things (IoT) domain [5].

Smart contracts can encrypt the rules of a contract into computer code and that code is replicated among blockchain nodes and executed.

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 matter, the programmable computations in Etehereum are executed by units of gas. If a transaction doesn't have enough gas for executing will be failed automatically.

Despite the expressiveness and power of the blockchain and smart contracts, the present form of these technologies lacks transactional privacy [11]. The entire sequence of actions taken in a smart contract are propagated across the network and/or recorded on the blockchain, and therefore are publicly visible, Even though parties can create new pseudonymous public keys to increase their anonymity, the values of all transactions and balances for each (pseudonymous) public key are publicly visible[11].

IV. Protocol and its Specification

The main purpose of our proposed protocol is to use blockchain technology and smart contract for secure transmission of shared key between two points in IoT environments. Suppose that we have several sensors in IoT environment and want to transmit collected data from sensors between a sender and a receiver in a secure way that preserves user's privacy. For instance one of the using of this protocol is in the healthcare applications that patient's data is collected using sensors and transmitted to a doctor, emergency or other service providers, in this scenario patient's privacy preserving is a very important issue [9].

For secure transmission of data, shared keys are generated and used but secure exchange of these shared keys is a challenge that should be solved and in this proposed protocol we present a solution for this problem.

First we should generate shared keys according to the data that is collected from sensors and then encrypt data with these shared keys. We use the proposed model in [9] for generating shared keys but for exchanging the shared keys between sender and receiver use blockchain technology. In fact the main idea in our protocol is using blockchain for secure transmission of shared keys.

There are three main agents in our protocol: sender, manager and receiver. Every agent has a block in blockchain and every block has a public and private key. For data transmission between agents applying transactions between blocks of blockchain, so we can use blockchain's security advantages.

Blockchain is a set of blocks that every block has its public and private keys. We use these keys for secure transmission of shared keys in blockchain. Because of transparency feature of blockchain all transitive data and transactions are revealed for all members of blockchain since blockchain has a global public ledger that is open for all members and because of this matter we should encrypt important data of confidentiality before input them in the blockchain.

We illustrate a general image of our protocol in Fig. 1.

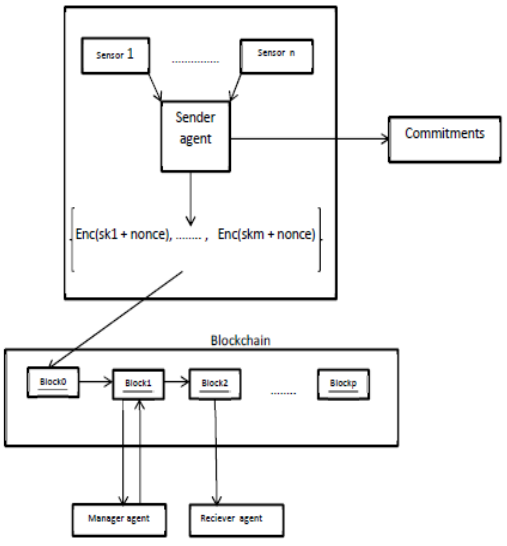

Fig. 1. General image of proposed protocol.

The proposed protocol has some summery signs that are

used in implementation. They are described in Table I.

TABLE I: SUMMARY SIGNS OF THE PROTOCOL

Summary sign	Description
X	Reading data from sensors
F	Function/condition
P	The probability function
MAXALIASES	Maximum number of possible 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	Sender agent
M	Manager agent
R	Receiver agent
Ksr	Shared key of sender and receiver
Ksm	Shared key of sender and manager
krm	Shared key of receiver and manager

A. Shared Key Generating

For generating of shared keys we use proposed model in [9]. First data are gathered from sensors, some functions are applied on them and probabilities of every function for every sensor are calculated. Every sensor has a MAX_ALIASES parameter that is maximum number of possible sensor aliases and depends on the resources available on the sensor platform [9]. Then according to the probabilities of every function 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 is shown in Fig. 2. The idea of using alias names is for uniformization operation and user's privacy preserving. Since the real origin of received data is unknown therefore privacy is preserved. Then for every couple of a function and an alias name a shared key is generated as flow:

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

ivx is generated randomly, and $c_{ij} = f_i(x)xor\ldots$ (the XORed outcome of iv_{ij} individual bits). The number of bits in the iv must be at least 2 so as to ensure that c_{ij} is equally likely to appear as 0 or 1, whether $f_i(x)$ is either 0 or 1 [9].

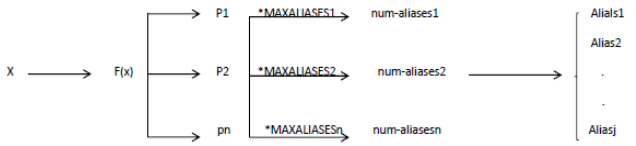

Fig. 2. Generating alias names from sensor readings process.

B. Shared Key Transmission Via Blockchain

A nonce number is generated randomly for every shared key. For verifying the integrity of transitive data a commitment is generated for every couple of shared key and its nonce, and it will be opened for all agents. Data should be encrypted before input in the block of bockchain and then the encrypted data is transmitted between blocks. We use private keys of blocks for encryption of data, and use Diffie-Hellman algorithm to generate symmetric key, and encrypt shared key using AES algorithm via generated symmetric key and transmit it in blockchain. In other words the sender encrypts shared key via AES algorithm and its generated shared key with manager and put it as data field in its block. And then using a transaction to send that data to the manager's block. The manager receives enc(sk+nonce), decrypts it, calculates the commitment of recipient data according to the hash function that compromises with sender, then compares the calculating value with commitment value that sender got and granted to all agents, if two values are equal thus the manager finds that the received data is correct and its integrity is preserved.

This step is doing once for every set of collected data from sensors and the manager will have the shared keys. After that every receiver that wants to have a shared key or we want to send a shared key to it just need that the manager and the receiver compromise for a symmetric key according to their block's private keys and then the manager encrypts the shared key with symmetric key and sends it as a data field of a transaction to the address of receiver's block. The cause 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 sending transaction is appear for all members and this manner has incoherence with confidentiality.

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

When a receiver gets the shared key, decrypts it and afterwards can exchange data with sender agent directly using received shared key.

C. The Advantages of Using Blockchain Technology in Proposed Protocol

- Blockchains allow us to have a distributed peer-to-peer network where non-trusting members can interact with each other without a trusted intermediary, in a verifiable manner [5].
- Blockchain guarantees the integrity of transmission data and this point can help to increase the performance of the system.
- Replay attack isn't possible because of limited amount of gas for every transaction in blockchain, and a transaction that use more than its gas limit automatically will be failed.
- It's not any requirement for intermediate because of using smart contract and any pair can correlate with each other directly.

- Hacking of the network is impossible because of distributed nature of blockchain.

V. IMPLEMENTATION

We use Ubuntu 14.04 operating system as a virtual machine in vmware workstation pro 12 for implementation of our proposed protocol.

For blockchain technology we use Ethereum. Ethereum is a decentralized cryptocurrency that uses its built-in currency, Ether, as fuel to power the programmable "smart contracts" that live on its blockchain [12]. Ethereum is more than a cryptocurrency (even though mining is involved) [12]. 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 [12].

For using Ethereum on a virtual machine need to pyethereum and serpent tools. Pyethereum is the program that allows for us to interact with the blockchain and test our contracts [12]. Serpent 2.0 will allow for us to compile our serpent code into the stack-based language that is actually executed on the blockchain [12].

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

We build a block on the blockchain for every agent and using utils in ethereum to attach a public and private key to every block. Public key of a block is used for addressing in transactions. We should state GAS_LIMIT, STARTGAS and GASPRICE parameters to preservation against replay attack in blockchain. Then in pyethereum tester define a chain and write our smart contract via serpent language. The main functions of our protocol are as follow:

- Makesk: is done via sender agent and its outputs are shared keys, nonce and commitments. The hash function for producing commitments is sha3_256 and is compromised between sender and manager agents. Also the commitments output of makesk function are visible for manager agent.
- Generatekey: converts private key of blocks of agents into integer.
- Difii: creates a shared key for every two agents using private key of their block.
- Encsk: encrypts sending items using AESCipher.
- Sendingitem: sends encrypted items between blocks of agents using smart contracts that are defined.
- Decsk: decrypts received items using AESCipher function.
- Checkcommitment: is done just in manager agent. And computes the commitments of received items using adaptive hash function to verify the integrity of them.
- Checksk: confirms that received items are correct.
- AESCipher: is a class that implements encryption and decryption of sending and received items using AES algorithm.

The encsk function is done in sender and manager agents and decsk function is done in manager and receiver agents. Computing and verifying the commitments are done just in manager agent, hence in transmission between manager and receiver just shared keys are sent and there is no need for nonce. The pseudocode of makesk, sendingitem, encsk and decsk functions are shown in Fig. 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 using a pair of public and private keys interchange with blockchain. The private key is used for signing the transactions and public key is used for addressing. Using of asymmetric encryption brings confidentiality and integrity preserving.

When using insecure database instead of blockchain, there is no guarantee that transmission data in the path isn't changed by malicious person. In other words integrity isn't guaranteed and we must use some methods to authenticate the integrity of received data. So these methods increase computation costs. But blockchain itself is resistant against changing of data and transmission data will be safe.

According to the experiment result and comparing two states of using insecure database and blockchain technology for storing and exchanging of shared key, we can result that in the first state, probability of unsafe transmission of data exists, and some techniques are needed for certainty of correctness of data. One technique is using hash function. In this method sender should generate a commitment for every transmission 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 sender's commitments. If the commitments are equal so the integrity of data is guaranteed. But by using blockchain there is no need to sending nonce numbers for every data and computing hash values, and this issue can improve the performance of the system.

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

There are two main paths in our proposed protocol, one path is between sender and manager agents and another is between manager and receiver agents. In the first path since a part of the path is out of blockchain, so we use hash function and commitment technique, but in the second path because of using blockchian there is no need to compute hash values, and this matter improves performance of the application. We can result that using blockchain in our proposed protocol can improve performance of integrity in the system.

VI. Evaluation

We use the AVISPA tool for evaluation of our proposed protocol. AVISPA stands for Automated Validation of Internet Security Protocols and Applications [13].

For evaluation the protocol in AVISPA we have tree main agents consist of sender, manager and receiver. And the main purpose of the protocol is that the shared key of sender and receiver is remained secret. First the shared key of sender and receiver is encrypted by shared key of sender and manager and exchange between sender and manager agents, the manager decrypts it, then the manager encrypts the shared key by shared key of itself and receiver and sends it to the receiver. The messages in cas+ are as follow:

1. S -> M : {Ksr}Ksm
2. M -> R : {Ksr}Krm

We write the protocol by hlpsl language and simulate it via span tool. We should do attack simulation for verifying the security of the protocol and see that the protocol is safe or not. The AVISPA Tool comprises four back-ends: OFMC, CLAtSe, SATMC, and TA4SP. We use OFMC and ATSE tools to examine the security of protocol. As we see in Fig. 7 the tool calls the OFMC back-end when called with the default options. We can see that OFMC found no attacks. In other words, the stated security goals were satisfied for a bounded number of sessions as specified in environment role. The goal that we define for protocol is secret of Ksr.

```
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 tool.

Also we do attack simulation by ATSE tool. We use ATSE tool with both algorithms, Depth First and Breadth First, and in tow conditions the protocol is safe. The results of simulation by ATSE tool are shown in Fig. 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 tool 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 tool by depth first algorithm.

Also the result of intruder simulation of the proposed protocol via AVISPA tool shows that if an intruder exists in the network, its knowledge is limited to the parameters that we defined for it and it can't access to the more information. So this point can improve confidentiality and integrity preservation of the transmission data in the protocol.

VII. Conclusion

Our proposed protocol is a usage 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 unknown and it helps for privacy preserving issue. In step of shared key transmission via blockchain and smart contract, first the shared key was encrypted and then 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 improve the performance.We have also shown through our implementation and evaluation that the proposed protocol is safe against attacks and can be useful for security challenges

of IoT environments. In the future we want to add some specifications to our proposed protocol for improvement of privacy preserving. One of the things that we want to do for improvement of our protocol is that designate different security levels to the various service providers, And this operation can help for increasing the security in the IoT environments. For implementation of this specification, we can use value parameter in the blockchain and give special value to the every service provider according to its trust level. So our effort for security preserving of transmission data between sender and every service provider will be according to the service provider's value parameter and its trust level.

ACKNOWLEDGEMENT

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

REFRENCES

[1] K. Rose, S. Eldridge, and L. Chapin, "The internet of things: An overview; understanding the issues and challenges of a more connected world," *The Internet Society (ISOC)*, 2015.

[2] R. Das and I. Das, *Secure Data Transfer in IoT Environment:* Adopting *both Cryptography and Steganography Techniques*.

[3] J. Brandt, 50 billion connected IoT devices by 2020. [Online]. Available: http://www.smartgridnews.com/story/50-billion-connected-iot-devices-2020/2015-04-21

[4] X. Yao, Z. Chen, and Y. Tian, "A lightweight attribute-based encryption scheme for the internet of things," *Future Generation Computer Systems*, vol. 49, pp. 104-112, 2006.

[5] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE, Digital Object Identifier*.

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

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

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

[9] P. Appavoo, M. Choon Chan, A. Bhojan, and E. Chang, "Efficient and privacy-preserving access to sensor data for internet of things (IoT) based services," in *Proc. 2016 8th International Conference on Communication Systems and Networks (COMSNETS).*

[10] R. Diffenthal, L. Rinaudo Cohen, and W. Maxwell, "Tree examples of blockchain smart contracts – Internet of Things, commercial paper and daos," *Global Media and Communications* **Quarterly** *Autumn*, 2016.

[11] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Proc. 2016 IEEE Symposium on Security and Privacy*.

[12] K. Delmolino, M. Arnett, and A. Kosba, *A Programmer's Guide to Ethereum and Serpent*, 2015.

[13] The AVISPA team, "AVISPA v1.1 user manual," *Document Version*, 2006.

[14] The AVISPA team, "HLPSL tutorial, a beginner's guide to modelling and analysing internet security protocols," *Document Version*, 2006.

[15] T. Genet, "A short SPAN+AVISPA tutorial," 2015.

[16] G. Zyskind, O. Nathan, and A. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," in *Proc. 2015 IEEE CS Security and Privacy Workshops.*

[17] X. Wang, J. Zhang, E. M. Schooler, and M. Ion, "Performance evaluation of attribute-based encryption: Toward data privacy in the IoT," *Communication and Information*.

[18] A. Ouaddah, A. Abou Elkalam, and A. Ait Ouahman, *Towards A Novel Privacy-Preserving Access Control Model Based on Blockchain Technology in IoT*, Springer International Publishing AG 2017.

[19] K. Delmolino, M. Arnett, and A. Kosba, "Lab: Step by step towards programming a safe smart contract," 2015.

[20] K. Delmolino, M. Arnett, and A. Kosba, "Step by Step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab," 2015.

Zahra Dehrouyeh receceived her BS degree in information technology engineering from Mazandaran University, Babolsar, Mazandaran, Iran in 2015. Now she is a master candidate in information technology engineering in Alzahra University, Tehran, Iran. Her research interests include internet of things (IoT), network security and privacy protection.

Reza Azmi received his BS degree in electrical engineering from Amirkabir University of technology, Tehran, Iran in 1990 and his MS and Ph.D degrees in electrical engineering from Tarbiat Modares University, Tehran, Iran in 1993 and 1999 respectively. Since 2001, he has joined Alzahra University, Tehran, Iran. He was an expert member of security information technology and systems working groups in ITRC (From 2006 to 2008). He was project manager and technical member of many industrial projects. Azmi is founder and director of cloud computing and value added services lab (CCVL), OCR and Image Processing Lab (OIPL) and Open Source Security Laboratory (OSSL) in Alzahra University.