# Ethereum-based Emergency Service for Smart Home System: Smart Contract Implementation

Yu Nandar Aung[1], Thitinan Tantidham[2]
Faculty of ICT, Mahidol University, Thailand
**yunandar.aun@student.mahidol.ac.th[1], thitinan.tan@mahidol.ac.th[2]**

*Abstract*— **Emergency service call for public service providers has become an important role for smart home applications in order to support safety and security in the household building. Blockchain has been a promising solution with cryptography and incentive distributed mechanisms to support the verification, execution and recording of transactions between untrusted parties. In this paper, we present a Smart Home System (SHS) based on Ethereum with smart contract infrastructure for handling an emergency service sending from SHS to Home Service Providers (HSP) when there are unusual environmental conditions. Our SHS testbed consists of three domains: (1) Smart Home Sensor Manager (SM) or IoT devices to gather environmental sensor data and send an emergency call to HSP, (2) Home Service Provider (HSP) system deployed with Ethereum Virtual Machine (EVM) and smart contract, and (3) decentralize Meteor framework to interface between Ethereum and web based applications for homeowners (HO) and HSP staffs. To achieve homeowner privacy and security, we enable digital signature coupling with InterPlanetary File System (IPFS) for handling the emergency call from SM to HSP and One Time Passcode (OTP) produced by HSP for HSP staffs to verify themselves for further access control when they go to service homeowner's house. Each smart contract transaction in solidity is described. Finally, security and privacy issues for our proposed work are discussed.**

*Keywords*— **Internet of Things (IoT), Smart Home, Emergency, Ethereum, Private Blockchain, Smart Contract, Solidity, IPFS, PKI, One Time Passcode**

## I. INTRODUCTION

With the rapid growth of technology, Smart Home System (SHS), deployed Internet of Things (IoT) technologies for 24/7 monitoring and controlling security and safety for a variety of threats and attacks like accidental injuries, crimes and life threatening, has been increasingly developed in market and research [1]. An emergency call is sent from a SHS to regarding public service providers like police office, fire department, hospital, or household equipment maintenance when the SHS detects unusual situations. Homeowner information, such as their home location, telephone number and email address, is transferred to those public services and it needs to be protected from various cyber-attacks. Furthermore, we also need to ensure and verify a public service staff to get an access control when the staff goes and gives the services at a resident house.

Due to cyber-attacks from Distributed Denial of Service (DDoS) on IoT system [2], man in the middle intercepting the transactions by malicious users during transmission and rogue of an IoT system [3], consequently trustiness, transaction handling and management between SHS coupling with IoT devices and public service providers as well as the privacy of homeowner information according to GDPR [4] and CBPR [5] are considered in our proposed system.

Our proposed work, we explore the design and implementation of Home Service Provider (HSP) for emergency call services based on a cluster of Ethereum Miners (EMs) and Meteor Decentralized Application framework. Ethereum with smart contract is used to handle transactions between SHS's IoT devices, HSP, Homeowners (HOs), and HSP staffs while the Meteor provides web-based interface for HOs to register, update, and view their own information as well as for HSP staffs to monitor and respond emergency calls. We leverage the encrypted message for homeowner profile during transferring to HSP, deploy digital signature coupling with IPFS for verifying an emergency call sent from SHS to HSP in order to protect DDoS attack from rogue IoT devices and apply One Time Passcode (OTP) as QR code with lifetime limitation generated by HSP for HSP staffs to verify themselves for any further access control at a homeowner's house.

The rest of the paper is organized as follow. The literature review and our proposed system architecture as well as implementation results with smart contract are presented in Section II and III respectively. Section IV discusses security and privacy for our proposed work. Finally, conclusions and future work are given in Section V.

## II. LITERATURE REVIEW

### *A. The Use of Blockchain in IoT*

Blockchain provides distributed infrastructure and uses public-key cryptography to create and record an immutable chain of blocks of transactions to protect against alteration and modification. Blockchain has been a significant for IoT applications to enhance security and data privacy without the need for trusted centralized authority [6]- [11].

Private (Permissioned) network can be restricted to a certain group of participants and only the specific members are allowed to deal with the distributed blocks. Public (Permission-less) network is open for anyone to join in and the distributed blocks can be accessed by every P2P member [12].

Dorri, et al. [13] proposed a combination of private and public Blockchain. Their approach consists of three tiers: smart home systems (SHS), overlay network and cloud storage. Private Blockchain was employed to handle data flow in SHS, whereas public Blockchain was to manage data flow over cloud storage.

Shrobe, et al. [14] presented Enigma, a high decentralized computation platform of secure multi-party computation for guarantee privacy. The aim of their research is to allow developers to build 'end-to-end decentralized applications', 'privacy by design', without a trusted third party.

According to Seyoung, et al. [15], Ethereum Blockchain platform where smart contracts are used to manage IoT devices by tracking meter usage and setting policies to control switching on and off air conditioners and light bulbs in order to save energy consumption.

Furthermore, Han, et al. [16] developed a blockchain-based smart door lock system. They mentioned an intrusion detection algorithm to control the smart door lock system and blockchain network is used to broadcast an alarming message when intrusion has happened.

In our proposed work, we implement the Ethereum based private blockchain with smart contract for handling an emergency call from SHS coupled with asymmetric key encryption, digital signature and OTP to enhance security and privacy issues.

### *B. Ethereum Blockchain*

Ethereum as our previous studies [18] is an open and programmable blockchain platform where developers can deploy smart contracts and build decentralized applications. Smart contract is a legal agreement between parties which can be developed by using Turing complete language such as Solidity, Serpent and Low-Level Lisp Like Language (LLL) [19]- [20]. In our proposed work, the Ethereum JavaScript console called geth [21] is used to run and stop the mining process. As described in Figure 1, geth can be started like "`geth attach ipc: /home/yunandar/Ethereum/miner1/geth.ipc`".

```
yunandar@ubuntu:~/Ethereum/watchdog_service$ geth attach ipc:/home/yunandar/Ethereum/miner1/geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/miner1/v1.8.15-stable-89451f7c/linux-amd64/go1.10.1
coinbase: 0x228ecdb45489780c8147fb62f44f4c0c6d60541a
at block: 5843 (Fri, 30 Nov 2018 20:03:54 +07)
 datadir: /home/yunandar/Ethereum/miner1
 modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

>
```

**Figure 1.** Geth JavaScript console

## III. PROPOSED WORK AND IMPLEMENTATION

As shown in Figure 2, our testbed implementation consists of (1) HSP with two Ethereum Miner (EM) nodes represented on separate Ubuntu 18.04LTS virtual machines running on CPU Intel i3-5005@ 2.00GHz and 12 GB RAM, SHS with Raspberry Pi 3 model B acting as a sensor manager (SM) equipped with sensors to sense the surrounding conditions and send an emergency call to HSP, and (3) web application interfaces for HSP staffs and homeowners. The following subsections will present the system configurations and implementation results for each component.

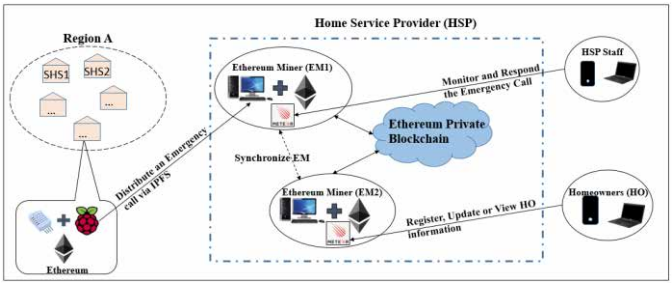


**Figure 2.** Proposed System Architecture

### *A. Ethereum Installation and Configuration*

We installed Ethereum package obtained from [22], [23] on two EM nodes and SM and described in details in [17]. A specific data directory folder is needed to create in order to store database and wallet for each member. We setup two nodes and SM in one Ethereum private blockchain network. All of them start with the same "genesis" file with the parameters described in Table 1. To synchronize the chain data, every node exchanges the information with each other by using "`node.admin.Info`" and "`admin.addPeer(NodeInfo)`".

**TABLE 1.** PARAMETER DESCRIPTION

| Parameters | Description | Examples |
|---|---|---|
| identity | The name of EM node | -- identity "miner1" |
| netwrok id | An arbitrary value used to pair all nodes of P2P network. | --networkid 22 |
| datadir | Folder where our private blockchain stores its data | --datadir "~/Ethereum/miner1" |
| rpc and rpcport | Enabling HTTP-RPC server and giving its listening port. | --rpcport 8545 |
| port | Network listening port number, on which EM nodes connect with each other. | --port 1234 |
| mine | To do the mining process. | --mine |
| unlock | Identity of the default Ethereum account to mine. | --unlock 0 (Ethereum account of miner1) |
| password | Path of the file containing the password of the default account. | --password ~/Ethereum/miner1/password.sec |
| ipcpath | Path where to store the file for IPC socket. | --ipcpath "~/Ethereum/miner1/geth.ipc" |

### *B. Meteor Installation and Web Interface*

At HSP, the Meteor Framework is installed and configured according to the guidelines as described in [24]. Meteor project can be created by using "`meteor create <project name>`" that MongoDB is built in automatically. Web service is started with "`meteor –port <host>:<port>`" as described in Figure 4. "Web3.js" [25] is added into the Meteor by "`meteor add ethereum: web3`" for interacting with the Ethereum JavaScript API. And the Ethereum can be started and interacted through web service with the command:

```
web3=new Web3(new Web3.providers.HttpProvider
("http://192.168.43.63:8545")),
```

where 192.168.43.63 is currently IP address of our EM system and 8545 is the rpcport number as described in Table I.

```
yunandar@ubuntu:~/Ethereum/watchdog_service$ meteor --port 192.168.43.63:3000
[[[[[ ~/Ethereum/watchdog_service ]]]]]

=> Started proxy.
=> A patch (Meteor 1.8.0.1) for your current release is available!
   Update this project now with 'meteor update --patch'.
=> Started MongoDB.
=> Started your app.

=> App running at: http://192.168.43.63:3000/
```

**Figure 3.** Ethereum running on meteor web service

When a user, HO or HSP staff registers to our system via web application, every user will obtain an Ethereum account via "`web3.personal.newAccount (passphrase)`", where passphrase is used to encrypt the account with AES-128 [26].

### *C. IPFS Installation and Configuration*

We use InterPlanetary File System (IPFS) to manipulate an emergency call among SM and EMs. Therefore, our testbed with two EMs and SM need to install and configure IPFS based on the guidelines as found in [27]. IPFS service is started with the command "`ipfs daemon`" on each node.

### *D. Authentication and Key Management*

Authentication is applied to identify the legitimate users and IoT end devices. There are numerous authentication methods like user name-password, biometric, one-time password token, digital signature, multiple factors authentication and Public Key Infrastructure (PKI) [28]- [33]. In our proposed work, we choose the following methods as follows.

- During the registration phase of our implementation, EMs generate 1024-bit RSA asymmetric key pair with JSEncrypt library [34] written in JavaScript, whereas SM produces the key pair Crypto.PublicKey module [35] with python.

  For EM:

  ```
  crypt = new JSEncrypt ({default_key_size: 1024});
  crypt.getKey();
  var publicKey=crypt.getPublicKey();
  var privateKey=crypt.getPrivateKey();
  ```

  For SM:

  ```
  pubkey, privkey) = rsa.newkeys(1024);
  privkey = privkey.save_pkcs1();
  pubkey = (pubkey.n, pubkey.e);
  ```

- User name and password is applied for user authentication like HOs and HSP staffs, via web interface.
- Digital signature is applied to verify the integrity of an emergency call and the correctness from SM. When there are any unusual conditions, SM attaches the digital signature with:

  ```
  signature = hexlify (rsa.pkcs1.sign(HoInfo, privkey, 'SHA-256'))
  ```

  where pkcs1 stands for Public Key Cryptographic standard [36].
- Time limited QR code is generated via one-time password token (OTP) for HSP staffs. Each staff has this code on his phone to be used as a pass code for getting into the HO house.

### *E. Transaction Types*

In our scenario, the following five transactions are referred function calls of defined smart contracts by the Ethereum accounts.

- User and Device Registration: To specify HOs, HSP staffs, EMs and SM with unique identification like Ethereum account, and MAC address.
- Emergency Type Identification: To identify the emergency types e.g., Fire alarm, medical emergencies and home invasion.
- HSP Staff Call: After verifying the incident type, the EM gives a call for HSP staffs.
- HSP staff Response: The HSP staff responds to the EM according to the emergency call for each homeowner.
- QR Code Generation: After the EM verifies the HSP staff, a QR code for the HSP staff will be generated as a pass code as an access control for giving the emergency service at HO's house.
- Service Fee Transfer: The EM makes a service fee balance between the HO and the HSP staff.

Users information are passed through from web interface as parameters to the Registration contract for HOs, HSP staffs as illustrated in Figure 4 of line 8. HOs require to fill token that will be converted to Ether for service fee payment. EMs and SM also need to register with MAC address of their own system, Ethereum account and RSA asymmetric key pair to construct PKI as described in Figure 5 of line number 8. In our proposed work, public key can be queried with its associated MAC address

Figure 6 shows the content of an emergency call from SM via IPFS hash file that consists of emergency type (line 2), digital signature (line 3), and HO's privacy information (lines 4-6). At EM, peer identity of each SM is obtained from "`ipfs swarm peers`" and the file content of each SM's is opened by "`ipfs name resolve <IPFS Hash File>`" to identify the emergency type. Then, EM forwards the emergency call to HSP staffs as illustrated in Figure 7, line 4. When a HSP staff responds this emergency call via web interface, the staff's Ethereum account and phone number will be transferred as input variables as shown in Figure 8 in order to generate QR code at EM as described in Figure 9. This QR code will be also transmitted to SM to verify the HSP staff at SM's house.

When the HSP staff is verified with the QR code, SM will send a report to EM in order to process a service fee payment, where the fee is withdrawn from HO's Ethereum wallet and deposited to HSP staff's wallet as shown in Figure 10 in line 4, and 6 respectively.

In our proposed work, all smart contracts from Figure 4 to 10 are created in Solidity and compile at Remix [37] to obtain JSON Application Binary Interface (ABI). We will use this ABI with "`web3.eth.Contract (abi).at (contract account)`" to instantiate the defined smart contract on the Ethereum [20] as mentioned in Figure 11.

```
//Home Owner Registration
1. contract HORegistration{
2.     string HOName;
3.     string HOEmail;
4.     uint HOTelephone;
5.     string HOAddress;
6.     uint HOToken;
7.
8. function homeOwnerRegistration(string memory _HOName, string memory
   _HOEmail, uint _HOTelephone,string memory _HOAddress, uint _HOToken)
   public {
9.         HOName=_HOName;
10.        HOEmail=_HOEmail;
11.        HOTelephone=_HOTelephone;
12.        HOAddress=_HOAddress;
13.        HOToken=_HOToken;
14.    }
15.}
```

**Figure 4.** Homeowner (HO) registration

```
//Ethereum Miner(EM) and Sensor Manager (SM) Registration
1. contract EMSMRegistration{
2. struct keyHolder {
3.         bool isSet;
4.         string publicKey;
5.         address account;
6.     }
7.     mapping (string => keyHolder) keys;
8.     function addKey(string memory mac, string memory
   _publicKey)public{
9.         require(!keys[mac].isSet);
10.        keys[mac]=keyHolder(true,_publicKey, msg.sender);
11.    }
12.
13.    function getKey(string memory mac)public view returns (string
   memory){
14.        return keys[mac].publicKey;
15.    }
16.}
```

**Figure 5.** Ethereum Miner (EM) and Sensor Manager (SM) registration

```
1. pi@raspberrypi:~/python $ python ./emergencyCall.py
2. Fire Service

3. signature =
   9c6db255d7376aeeec0ae680acf8992487a9259adab6d78c7e2c19ef2a3d50fe033de42b1d
   b3aca90f8bf96e68e75007c12fb3cf760026e0c013ce9783c789ef1d35ddcf340b3cf4afe2
   0b90c3c21f231257352a93bfc41d27ee30d3050fba7b0efc7e4d54f19ef5b2569d45b86f07
   be8038278494d4d731f6f4c4dea373b742

4. HO_Info={"Name": "Alice",
5.       "Home Location": "Salaya",
6.       "Ethereum account": "0xb23b22fc76a98510675fc437ccf001a3fa11b2ec"
7.           }
8. 2018-11-29 17:25:43.842698
```

**Figure 6.** The content of an emergency call from SM

```
//An Emergency Call to HSP Staffs
1. contract AnEmergencyCall {
2. string public message;
3.     mapping (address => string) message;

4. function sendemergencycall(string memory _emergencyType) public{
5.     message[_ HSPStaff] = _emergencyType;
6.     }
7. Function rademergencycall() returns (string memeory){
8.     return message[msg.sender];
9.     }
10.}
```

**Figure 7.** An emergency call to HSP staff

```
//An HSP staff respond to the emergency call
1. contract RespondEmergencyCall {
2. string public message;
3.     mapping (address => string) message;

4. function respondemergencycall(address memory _HSPStaff, address
   _HSPAccount, uint _mobilePH, string memory _respondcall)public{
5.          message[_ HSPStaff] = _emergencycall;
6.    HSPaccount=_HSPAccount;
7.    MobilePH= _mobilePH
8.     }
9. }
```

**Figure 8.** The response from an HSP staff to the emergency call

```
//EM generates QR code as Passcode to access to the HO's House
1. contract QRCode{
2.     string public QRCode;
3.     event stringChanged(string indexed changedString);
4.     function setMessage(string memory _ QRCode) public {
5.          QRCode =_ QRCode;
6.          emit stringChanged(QRCode);
7.     }
8.     function getMessage() public view returns (string QRCode){
9.         return QRCode;
10.    }
11.}
```

**Figure 9.** The operations of EM to generate a QR code as a passcode for a HSP staff in order to access the HO's house

```
1. Contract SendServiceFee{
2. mapping (address => uint256) public balanceOf;
3. function transfer(address _to, uint256 _value) public returns
   (bool success) {
      // Check homeowner's balance
4.          require(balanceOf[msg.sender] >= _value);
      // withdrawl homeowner's balance
5.          require(balanceOf[_to] + _value >= balanceOf[_to]);
6.    balanceOf[msg.sender] -= _value;
7.    // deposit to HSP's staff balance
8.          balanceOf[_to] += _value;
9.          return true;
10.         }
11.}
```

**Figure 10.** Service fee payment

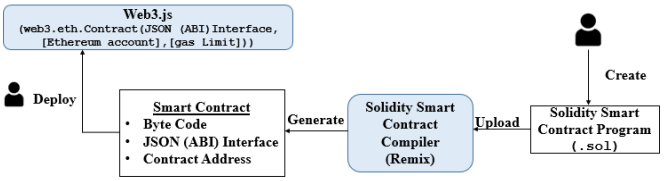


**Figure 11.** Create and deploy smart contract

## IV. SECURITY AND PRIVACY DISCUSSIONS

In this section, we discuss security and privacy of our proposed work in each domain as follows.

### *A. Smart Home System (SHS)*

To get access of Raspberry Pi (RPi), we need user name and strong password. Moreover, we encrypt homeowner information and store it on SD card. Therefore, the attacker cannot easily access and do hijacking RPi.

### *B. Privacy Information*

Personal data of HOs distributed to EM or HO peers via IPFS for an emergency call is digitally signed with the private key of SM and decrypted with the public key of EM. For this reason, an attacker cannot intercept the emergency call to get HO's personal information.

Moreover, all outgoing transactions are signed with the private key of sender side and is verified with the public key

at receiver side. If any attacker generates forged message, it can be detected and discarded. Therefore, our proposed system can prevent DDoS attack.

### C. Web Service and MongoDB

We restrict our users of HOs and HSP staffs to use strong password which contains special characters, capital letters, numbers and eight-character minimum length for registration.

### D. Ethereum Account

Untrusted users cannot do illegal activities and transactions with the use of Ethereum accounts since we record these accounts together with the user information.

### E. Smart Contract Vulnerabilities

To explore the vulnerabilities of our solidity smart contract, we use "SmartCheck" [38] as a static code analyser by running through solidity source code in remix. Three recommendations found in highlights of Figure 12 are (1) 'compiler version not fixed', (2) 'upgrade code to solidity 0.5.0' and (3) 'implicit visibility level' in lines 1, 5, 8-12 and 14-16.

```
1  pragma solidity ^0.4.18;
2
3  //Sensor Manager sends notification about Emergency Incidents
4  contract DetectIncidents {
5      string IPFS;
6      event stringChanged(string indexed changedString);
7
8      function setIncidentsType(string memory _IPFS)  {
9          IPFS=_IPFS;
10
11        emit stringChanged(IPFS);
12     }
13
14     function getIncidentsType()  view returns (string memory){
15         return IPFS;
16     }
17 }
```

**Figure 12.** Suggestions obtained from "SmartCheck"

```
1  pragma solidity 0.5.0;
2
3  //Sensor Manager sends notification about Emergency Incidents
4  contract DetectIncidents {
5      string public IPFS;
6      event stringChanged(string indexed changedString);
7
8      function setIncidentsType(string memory _IPFS) public {
9          IPFS=_IPFS;
10
11        emit stringChanged(IPFS);
12     }
13
14     function getIncidentsType() public view returns (string memory){
15         return IPFS;
16     }
17 }
```

**Figure 13.** After correcting the suggestions by "SmartCheck"

The first recommendation means that we have to remove "^" because we may have risks of undiscovered bugs. The second one means that our contract's compiler version should be upgraded with Solidity v0.5.0. The last one is that our contract's functions and variables need to declare explicitly in order to prevent illegal access. They should be specified as 'external', 'public', 'internal' or 'private' [39]. As illustrated in Figure 13, lines 5, 8 and 14, we define our contract's functions and variables as 'public' to allow internal calls and to deploy the contracts with web3.js.

### F. EM Performance Results

Based on our testbed system, the average elapsed time for each transaction ranges between 5 to 11 seconds from running experimental 10 times. During the mining process, the minimum and maximum CPU usage ranges from 63% to 100%. The performance can be improved if we have a high speed computing system.

## V. Conclusions and Future Work

In this paper, we presented an emergency service for Smart Home System design and implementation based on the Ethereum private blockchain with solidity smart contract to enable transactions from untrusted parties. We analysed the security and privacy based on our testbed components where smart home system based on Raspberry Pi can be represented as an IoT edge device and privacy information is focused on user information. Future work should include secure file system and MongoDB for handling all user passwords and key management. Our testbed system can be further applied for any smart emergency call service applications. House access control mechanisms with OTP can be further improved. As smart contract security is an issue in Blockchain development, we plan to analyse our testbed and compare the results with other security analysis tools as shown in [40].

## Acknowledgement

This research project was partially supported by Faculty of Information and Communication Technology, Mahidol University.

## References

[1] J. Bleja, U. Grossmann and H. Langer, "A Collaborative System Business Model for Ambient Assisted Living Systems," 2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), Lviv, 2018, pp. 78-81.

[2] E. Fernandes, "Security Risks in the Age of Smart Homes," 2016 [Online]. Available: http://scitechconnect.elsevier.com/security-risks-age-smart-homes/ [Accessed: 1- Dec - 2018].

[3] Ali B, Awad AI. Cyber and Physical Security Vulnerability Assessment for IoT-Based Smart Homes. Sensors. 2018 Mar 8;18(3):817. [Online]. Available: https://www.mdpi.com/1424-8220/18/3/817/htm [Accessed: 1- Nov - 2018].

[4] Team IP. EU general data protection regulation (GDPR): an implementation and compliance guide. IT Governance Ltd; 2017 Aug 31.

[5] Maxwell WJ. Global Privacy Governance: A comparison of regulatory models in the US and Europe, and the emergence of accountability as a global norm. Cahier de prospective. 2014 Feb;63.

[6] C. Natoli and V. Gramoli, "The Blockchain Anomaly," 2016 IEEE 15th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, 2016, pp. 310-317.

[7] J. Yli-Huumo, D. Ko, S. Choi, S. Park, K. Smolander. Where is current research on blockchain technology? —a systematic review. PloS one.

2016 Oct 3;11(10): e0163477, [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0163477 [Accessed: 9- Dec - 2018].
[8] C. Dannen, *Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginner*, Brooklyn, New York, USA, ISBN-13 (electronic), 2017.
[9] D. Patel, J. Bothra and V. Patel, "Blockchain exhumed," 2017 ISEA Asia Security and Privacy (ISEASP), Surat, 2017, pp. 1-12, 2017.
[10] Z. Zheng, S. Xie, H. Dai, X. Chen and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," 2017 IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, USA, 2017, pp. 557-564.
[11] X. Xu et al., "A Taxonomy of Blockchain-Based Systems for Architecture Design," 2017 IEEE International Conference on Software Architecture (ICSA), Gothenburg, 2017, pp. 243-252.
[12] G. Gabison, "Policy considerations for the blockchain technology public and private applications," SMU Science and Technology Law Review, 2016, pp. 330-335.
[13] A. Dorri, S. S. Kanhere, R. Jurdak and P. Gauravaram, "Blockchain for IoT security and privacy: The case study of a smart home," 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Kona, HI, 2017, pp. 618-623.
[14] H. Shrobe, D. L. Shrier, A. Pentland, *CHAPTER 15 Enigma: Decentralized Computation Platform with Guaranteed Privacy in New Solutions for Cybersecurity*, MITP, 2018, pp.425-454.
[15] S. Huh, S. Cho and S. Kim, "Managing IoT devices using blockchain platform," 2017 19th International Conference on Advanced Communication Technology (ICACT), Bongpyeong, 2017, pp. 464-467.
[16] D. Han, H. Kim and J. Jang, "Blockchain based smart door lock system," 2017 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, 2017, pp. 1165-1167.
[17] A. Yunandar and T. Tantidham. "Emergency Service for Smart Home System Using Ethereum Blockchain: System and Architecture," accepted, to be appeared in Proceedings of the 3rd International Workshop on Smart Edge Computing and Networking (SmartEdge), 2019 Mar 11-15, Kyoto, Japan.
[18] A. Yunandar and T. Tantidham. "Review of Ethereum: Smart Home Case Study," in Proceedings of the 2nd International Conference on Information Technology, 2017 Nov 2-3, Nakhon Pathom, Thailand, 2017, pp. 1-4.
[19] E. Daniel, "An Introduction to LLL for Ethereum Smart Contract Development," [Online]. Available: https://media.consensys.net/an-introduction-to-lll-for-ethereum-smart-contract-development-e26e38ea6c23, 2017.
[20] C. Dannen, "Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginner", Brooklyn, New York, USA, ISBN-13 (electronic), 2017.
[21] C. Dannen, *Introducing Ethereum and Solidity*, Apress, Berkeley, CA, 2017, pp. 111-137.
[22] E. Said, "Creating a private blockcahin with IoT Devices," [Online]. Available: https://chainskills.com/2017/02/24/create-a-private-ethereum-blockchain-with-iot-devices-16/, 2017 Feb 24. [Accessed: 20-Dec-2017]
[23] F. Lange, "Installation Instructions for Ubuntu," [Online]. Available: https://github.com/ethereum/go-ethereum/wiki/Installation-Instructions-for-Ubuntu [Accessed: 20-Dec-2017]
[24] D. Turnbull, *Your first meteor application,* Kindle Amazon, 2014.
[25] "Web3.js – Ethereum Javascript API," [Online]. Available: https://web3js.readthedocs.io/en/1.0/ [Accessed: 1-Aug-2018]
[26] E. Jincharadze, "Hybrid Encryption Model of Symmetric and Asymmetric Cryptography with AES and Elgamal Encryption Algorithms," Scientific & practical cyber security journal, ISSN, pp. 2587-4667, 2018.
[27] "IPFS is the Distributed Web," [Online]. Available: https://ipfs.io/ [Accessed: 1-Aug-2018]
[28] Thakur M. Authentication, Authorization and Accounting with Ethereum Blockchain, 2017.
[29] Merkle RC. A digital signature based on a conventional encryption function. InConference on the theory and application of cryptographic techniques 1987 Aug 16 (pp. 369-378). Springer, Berlin, Heidelberg.
[30] Simmons GJ. Symmetric and asymmetric encryption. ACM Computing Surveys (CSUR). 1979 Dec 1;11(4):305-30.
[31] Somani U, Lakhani K, Mundra M. Implementing digital signature with RSA encryption algorithm to enhance the Data Security of cloud in Cloud Computing. InParallel Distributed and Grid Computing (PDGC), 2010 1st International Conference on 2010 Oct 28 (pp. 211-216). IEEE.
[32] Sun H, Sun K, Wang Y, Jing J. Trustotp: Transforming smartphones into secure one-time password tokens. InProceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security 2015 Oct 12 (pp. 976-988). ACM.
[33] "How to transfer between two remote servers," [Online]. Available: https://stackoverflow.com/questions/28831329/how-to-transfer-a-file-between-two-remote-servers-using-scp-from-a-third-local [Accessed 15-Aug-2018]
[34] A. Lapets, E. Dunton, K. Holzinger, F. Jansen, & A. Bestavros, "Web-based multi-party computation with application to anonymous aggregate compensation analytics," Computer Science Department, Boston University, 2015.
[35] W. Hu, P. Corke, W. C. Shih, & L. Overs, A public key technology platform for wireless sensor networks. In European Conference on Wireless Sensor Networks (pp. 296-311). Springer, Berlin, Heidelberg, February, 2009.
[36] B. Kaliski, "PKCS# 1: RSA encryption version 1.5," 1998.
[37] "Remix," [Online]. Available: https://github.com/ethereum/remix [Accessed: 1- Aug – 2018].
[38] A. Seleznev, "Connect Job Smart Contract Security Audit," [Online]. Available: https://blog.smartdec.net/connectjob-smart-contracts-security-audit-9d4b20f7e83f, 2017. [accessed: 1 Dec 2018].
[39] S. Tikhomirov, E. Voskresenskaya, et al., "SmartCheck: Static Analysis of Ethereum Smart Contracts," 2018.
[40] R. M. Parizi, A. Dehghantanha, K-K R. Choo. And A. Singh. "Empirical vulnerability analysis of automated smart contracts security testing on blockchains," In Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering, October 2018.

**Yu Nandar Aung** received her M.E. degree in Information Technology from Yangon Technological University, Myanmar. She is currently a Master student in Cybersecurity and Information Assurance program at Faculty of ICT, Mahidol University, Thailand.

Thitinan Tantidham received her B.Eng. degree in Computer Engineering from Kasetsart University, Thailand, her M.Sc. in Computer Science from Mahidol University, Thailand and Ph.D. degree in Computer Science from RWTH Aachen University, Germany. Her research projects and interests include embedded systems, Internet of Things applications and security, energy consumption monitoring and management, and energy efficiency.