Emergency Service for Smart Home System Using Ethereum Blockchain: System and Architecture

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

Abstract—**Due to emerging disruptive technologies, Internet of Things (IoT) play a vital role for smart living domains, for examples, elderly and disabilities healthcare services and home safety and security monitoring and automation control services. These systems will send automatically an emergency call with home user information and location as a privacy data to public services like hospitals, police offices, or fire departments. This paper introduces emergency service for a Smart Home System (SHS) based on Ethereum blockchain with smart contract for decentralized handling access control among untrusted public services so called Home Service Providers (HSPs) and smart home IoT devices. Our SHS testbed consists of (1) smart home sensor manger equipped with Raspberry Pi (RPi) represented as an edge IoT gateway for gathering environmental sensor data, (2) HSP miners deployed Meteor and Ethereum platform, and (3) web-based applications for home users and HSP staffs. Furthermore, our contribution includes the integration of digital signature for the IoT device authentication, the One Time Passcode with QR code for HSP staff access control, and IPFS for manipulating emergency call from SHS in peers. Our implementation results focusing on HSP miners will be presented and analysed.**

Fig. 1. ***Keywords— Internet of Things (IoT), Ethereum, Private Blockchain, Smart Home, Emergency, Smart Contract, IPFS, One Time Passcode***

I. INTRODUCTION

Establishing safe and secure living environment goes beyond protecting residents from a variety of threats like accidental injuries, crimes and life-threatening situations. Therefore, 24/7 monitoring and controlling safety and security services for residents such as elderly people, disabilities and independent living families increasingly demand for smart home Internet of Things (IoT), which require to send automatically an emergency call to public services like hospitals, police stations, fire departments, or household maintenances according to any abnormal situations [1]. User information and home location transferred to those public services need to be protected from various of cyber-attacks according to GDPR [2] and CBPR [3].

Due to various types of cyber-attacks such as Botnets, Distributed Denial of Service (DDoS), data and identity theft, home users are wary of using Smart Home System (SHS) [4]. Hence, in this paper, we propose an Ethereum blockchain based Home Service Provider (HSP) with smart contract for decentralized handling transaction control among HSPs and SHSs. Our proposed SHS is composed of Sensor Manager (SM) coped with sensors to collect environmental data and send an emergency call automatically to HSP when any sensor data values are over than the predefined threshold. HSP deploys a cluster of Ethereum Miners (EMs) coupled with Meteor Decentralized Application Framework to provide web-based applications for home users and HSP staffs to register, view and edit user information, and for HSP staffs to respond an emergency call. Furthermore, our contribution with authentication techniques will be integrated as follows. Interplanetary File System (IPFS) with digital signature and PKI is applied for emergency call service when a SHS sends an emergency call to HSP peers in order to protect DDoS attack from any rogue IoT device. QR code based on HSP staff phone number is employed as one time pass code in order for verification and accessing control at the home owner's house or building.

The rest of the paper is organized as follow. Section II explains literature review. Section III presents our proposed system architecture design and implementation. Section IV presents performance results and analysis. Finally, conclusions and future work are given in Section V.

II. LITERATURE REVIEW

Blockchain is a distributed database and able to maintain continuous records of transactions as a block within Peer-to-Peer (P2P) network nodes. These blocks are linked with each other as a chain by using the cryptographic hash function. Consequently, attackers cannot change these records since they are stored by using cryptographic hash and distributed within P2P nodes. Many blockchain platforms are available to allow developers to make and program an agreement, policies and rules for users within a distributed blockchain network. These programs are called smart contract. Judgement made by third parties is eliminated with the use of smart contract. Due to its several advantages, blockchain technology can be applied for other domains beyond cryptocurrencies including identity management, data storage management, data trading and so on [5].

A. Ethereum Blockchain

According to our previous studies [6] Ethereum blockchain is chosen as an open source platform, and compatible with many decentralized application frameworks such as Meteor, Truffle, and Ganache [7, 8] and provides programmable smart contracts to trigger transactions automatically with specified conditions. It also provides both private and public blockchain. Private blockchain allows only the specified members of the P2P network to deal with the distributed records. Moreover, different roles of members are allowed only read permission in distributed records [9]. Therefore, we choose Ethereum private blockchain for our proposed work due to homeowner's privacy information and the immutability of data transaction, auditability, integrity and authorization.

B. *Smart Contract*

In 2013, Vitalik Buterin proposed a built-in Turning-complete language to develop smart contract on Ethereum blockchain. According to our previous studies [10], smart contracts are written in solidity scripting language [11] and compiled at Remix [12] to get JSON Application Binary Interface (ABI). This ABI with the Ethereum contract address are used to call the smart contract function as a transaction within the Ethereum private blockchain (EPB).

C. *Mining Process*

In the blockchain network, the most important point is to make sure that the distributed data to every P2P node should be identical, immutable and tamper-proof. For this reason, blockchain requires consensus algorithm to achieve an agreement on one consistent state of transaction among distributed P2P nodes. There are several well-known cnsensus algorithms in the blockchain technology such as Proof-of-Concept (PoC), Proof-of-Work (PoW), Proof-of-Stake (PoS), and Byzantine Fault Tolerance (PBFT) [13].

PoC is a demonstration to prove that the blockchain has significant benefits for the real-world applications. In our proposed work, we use PoC to convince that the private Ethereum blockchain with solidity smart contract plus decentralized application (DApp) can manipulate emergency service for smart home system without trust.

D. *InterPlanetary File System (IPFS)*

IPFS [14] is an open-source, content-addressable, and peer-to-peer (P2P) method of storing and sharing hypermedia in a distributed system. IPFS addresses file location with the content itself. This is done by using a cryptographic hash on the file. At our proposed work, we combine IPFS and Ethereum in distributed miner systems. The IPFS will be used to send an emergent incident from each home SM to a regional miner.

E. *Asymmetric Encryption*

Encryption is to protect user data or sensitive information from eavesdropper during data-in-transit. In our proposed work, we use RSA asymmetric encryption algorithm and the key length is 1024-bit. RSA can be used for both "data encryption" and "digital signature" [15].

F. *Authentication Methodologies*

Authentication is a process to verify the identity of users or IoT devices to be able to access computing system, resources or applications. It is used to make sure that only the trusted members or IoT devices can make a transaction with SHS. Our proposed work, we deploy authentication mechanisms for edge users and devices as follows and more implementation details can be found in our work [10].

User name- password login will be used to authenticate our users from web application interface.

Our proposed Public Key Infrastructure (PKI) [16] is built based on the RSA asymmetric (private-public) key together with the Ethereum account and MAC address of each EM and SM system to verify the emergency call.

Digital signature [17, 18] is used to verify the emergency call (message) from each SM via IPFS. The SM uses its own "private key" to sign the message which contains emergency type, and HO information. This emergency message is encrypted with EM's public key and broadcasted throughout the IPFS peer. When the content is received by the trusted EM node, it can be decrypted with EM's private key.

One-time password token [19] is generated in the form of QR code for each HSP staff based on his/her phone number. Each HSP staff obtains the QR code from EM on his/her smart phone and uses it as a pass code to get into the HO house.

III. PROPOSED DESIGN AND IMPLEMENTATION RESULTS

The system architecture of our proposed work consists of Smart Home System (SHS), Ethereum miner (EM) for Home Service Provider (HSP), and web based application for homeowners (HO) and HSP staffs as described in Figure 1.

Each SHS is composed of Raspberry Pi 3 (RPi) as a Sensor Manager (SM) for detecting the surroundings and sending an emergency call according to predefined conditions to HSP's EM miner peer via IPFS.

The HSP's EMs have three parts: Meteor Decentralized Application (DApp) Framework [20], Geth [21, 22] and IPFS[14]. The Meteor is a web service framework for creating web user interfaces like user registration and monitoring transactions for HO and HSP staff. Geth is the Ethereum JavaScript console for HSP's admin to monitor transactions and to do mining process. Solidity smart contract is implemented, compiled and deployed on the Ethereum private blockchain (EPB) via Remix IDE [12]. All of these components are shown in Figure 2. The installation and configuration of each system will be described below.

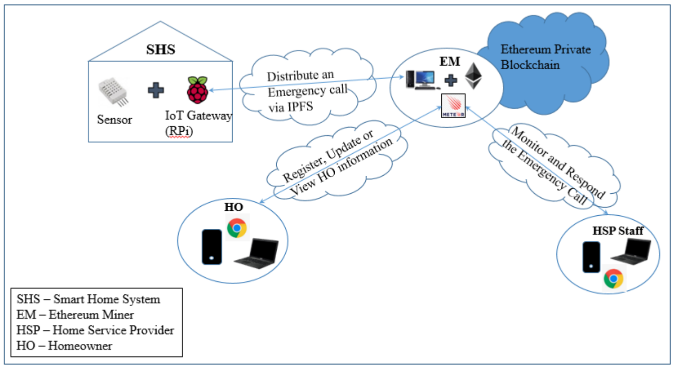
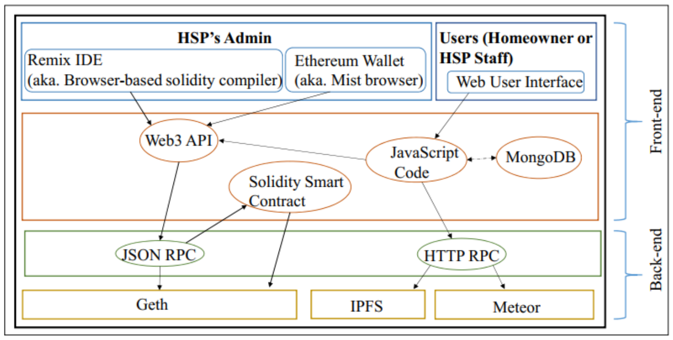

SHS
Sensor
IoT Gateway (RPi)
Distribute an Emergency call via IPFS
EM
Ethereum Private Blockchain
Register, Update or View HO information
Monitor and Respond the Emergency Call
HO
HSP Staff
SHS – Smart Home System
EM – Ethereum Miner
HSP – Home Service Provider
HO – Homeowner

Fig. 1. Proposed system architecture



Remix IDE – https://github.com/ethereum/remix-ide
Ethereum Wallet – https://github.com/ethereum/mist/releases
IPFS – https://docs.ipfs.io/introduction/install/
Meteor – http://meteortips.com/first-meteor-tutorial/getting-started/

Fig. 2. Software components of EM

A. *Sensor Manager Installation and Configuration*

We deploy Raspberry Pi (RPi) as the sensor manager, IoT gateway and Ethereum client, and apply IPFS to distribute an emergency call. We install python IPFS [23] and geth [24] on RPi.

To start the Ethereum private blockchain, a "genesis" file with the specified parameters, shown in Table I, is initialized with the command "`geth --datadir ~/Ethereum/rpi/ init genesis.json`".

The service of IPFS is started with "ipfs daemon" and configured its default port number of HTTP RPC API from the python script file as follows.

"`ipfsAPI=ipfsApi.Client('a.b.c.d',5001, {protocol: 'http'})`", where a.b.c.d is the current IP address of SM.

When the SM sends an emergency call to EM, the operations will be as follows

- SM encrypts the emergency call with the RSA public key of EM node like

  "`encrypted_emergencyCall=rsa_EMpublicKey.encrypt('EmergencyCall.json',32)`" with the attachment of digital signature as:

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

  where pkcs1 performs RSA algorithm to generate Public Key Cryptographic Signature [25].

- SM uploads the encrypted emergency call as: "`ipfsLoadedFile=api.add('encrypted_encrypted_emergencyCall.json')`", to the IPFS peers.

- EM receives the encrypted emergency call from the hash distributed by SM and decrypts with its own RSA private key to retrieve the type of emergency.

B. *HSP's EM Installation and Configuration*

To set up EM node, the Ethereum package is obtained from [24] and we need to create a specific data directory folder to store database and wallet of the EPB. The configuration is the same as SM described in Table I, but EM requires to apply "`--mine`" parameter in "genesis" file to do the mining process.

We install and build the Meteor DApp framework according to [25]. We can create meteor project with the command "`meteor create <project name>`" and MongoDB is automatically built. For the web service, we can start with the command "`meteor -port a.b.c.d:3000`", where a.b.c.d is the current IP address of EM node and 3000 is the port number of HTTP RPC API. "Web3.js" is added into the Meteor by "`meteor add ethereum:web3`" for interacting with the solidity smart contracts functions. EM is also required to install and configure IPFS to receive the emergency call from an SM.

TABLE I. TABLE TYPE STYLES

Parameters	Descriptions	Examples
identity	Name of an HSP's EM node.	`"miner1"`
networkid	A defined value. We use it to pair all HSP's EM nodes in the same private blockchain network.	`22`
datadir	Folder where our EPB stores its data.	`"~/Ethereum/miner1"`
rpc and rpcport	HTTP-RPC port	`8545`
port	Network listening port to connect HSP's EM nodes with each other.	`1234`
mine	To do the mining process. This parameter is only for EM.	`--mine`
unlock	Default Ethereum account to do mining.	`etherbase (default etheruem account)`
password	File path contains the password of the etherbase.	`"~/Ethereum/miner1/password.sec"`
ipcpath	File path which stores configuration file.	`"~/Ethereum/miner1/geth.ipc"`
rpcapi	Lists of remote procedure call API protocols running on EPB.	`"db,net,admin,eth,personal,miner,net,web3"`
rpccorsdomain	To allow meteor web service to get access to EPB.	`"http://192.168.43.63:3000"`
rpcaddr	IP Address of HTTP-RPC server.	`192.168.43.63`
syncmode	Synchronization mode using for HSP's EM nodes	`"fast"`
cache	Cache size of the HSP's EM nodes during blocks synchronization	`1024`

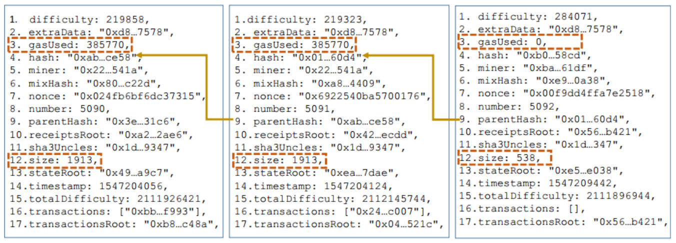
C. *Transaction Types*

In our proposed system, we handle the following types of transactions as functions to invoke solidity smart contract via JSON RPC. The details of each smart contract below can be found in [10].

- Solidity Smart Contract Creation.
- Emergency Type Identification.
- Emergency call for HSP staffs.
- HSP staff response for the emergency call.
- QR code generation for the HSP staff to get access control at HO’s house.
- Service fee transfer from HO’s Ethereum wallet to HSP staff’s.

D. *Gas Limit*

In our proposed work, every transaction is charged a gas fee according to its transaction size in bytes. Figure 3 shows an example of three blocks in sequence, where gas usage for each transaction and transaction size are illustrated at line 3 and 12 respectively.



```
1.  difficulty: 219858,
2.  extraData: "0xd8…7578",
3.  gasUsed: 385770,
4.  hash: "0xab…ce58",
5.  miner: "0x22…541a",
6.  mixHash: "0x80…c22d",
7.  nonce: "0x024fb6bf6dc37315",
8.  number: 5090,
9.  parentHash: "0x3e…31c6",
10.receiptsRoot: "0xa2…2ae6",
11.sha3Uncles: "0x1d…9347",
12.size: 1913,
13.stateRoot: "0x49…a9c7",
14.timestamp: 1547204056,
15.totalDifficulty: 2111926421,
16.transactions: ["0xbb…f993"],
17.transactionsRoot: "0xb8…c48a",
```

```
1.difficulty: 219323,
2.  extraData: "0xd8…7578",
3.  gasUsed: 385770,
4.  hash: "0x01…60d4",
5.  miner: "0x22…541a",
6.  mixHash: "0xa8…4409",
7.  nonce: "0x6922540ba5700176",
8.  number: 5091,
9.  parentHash: "0xab…ce58",
10.receiptsRoot: "0x42…ecdd",
11.sha3Uncles: "0x1d…9347",
12.size: 1913,
13.stateRoot: "0xea…7dae",
14.timestamp: 1547204124,
15.totalDifficulty: 2112145744,
16.transactions: ["0x24…c007"],
17.transactionsRoot: "0x04…521c",
```

```
1.  difficulty: 284071,
2.  extraData: "0xd8…7578",
3.  gasUsed: 0,
4.  hash: "0xb0…58cd",
5.  miner: "0xba…61df",
6.  mixHash: "0xe9…0a38",
7.  nonce: "0x00f9dd4ffa7e2518",
8.  number: 5092,
9.  parentHash: "0x01…60d4",
10.receiptsRoot: "0x56…b421",
11.sha3Uncles: "0x1d…347",
12.size: 538,
13.stateRoot: "0xe5…e038",
14.timestamp: 1547209442,
15.totalDifficulty: 2111896944,
16.transactions: [],
17.transactionsRoot: "0x56…b421",
```

Fig. 3. Comparison of gas usage and transaction size among three sequential blocks

IV. Performance Results and Analysis

Our testbed system consists of two HSP’s EM nodes: EM1, and EM2 running on Ubuntu are illustrated in Table II.

TABLE II. System Specifications of HSP’s EMs

	EM1	EM2
CPU Model	Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz	Westmere E56xx/L56xx/X56xx (IBRS update)
CPU MHz	1995.383	2659.998
RAM	3072 KB	16384 KB

A. *Transaction Operations*

One of our solidity smart contracts called Emergency Type Identification (ETI) is illustrated in Figure 4. After deploying this contract in Remix environment, we got JSON ABI and contract address to call ETI functions: setEmergencyType() and getEmergencyType().

These both functions create new pending transactions which can be viewed at Remix terminal denoted as “A” and “B” shown in Figure 5. When the EM node does the mining process with the command “`miner.start()`” in geth console, these pending transactions are successfully sealed at the block number 5077 as highlighted in Figure 6 and the details of this block content shown in Figure 7.

B. *HSP’s EM Performance Results*

To add any new EM node, we have to synchronize all EM nodes as the following steps.

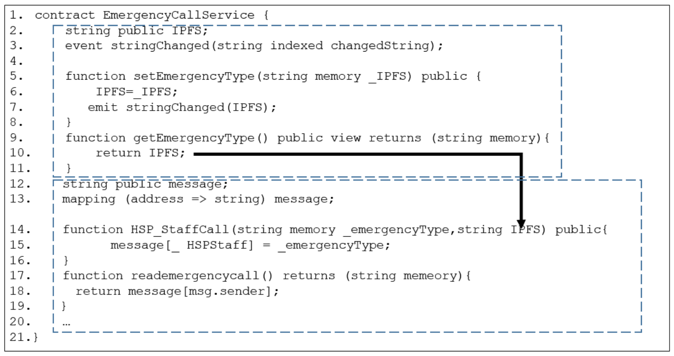
- Node information of each EM can be retrieved from geth console with the command “`admin.nodeInfo`”.
- As shown in Figure 8, EM 1 adds the node information of EM2 as its peers with the command “`admin.addPeer(<Node Information of EM2>)`” and vice versa for EM2.

According to [26], we can setup synchronization mode as “`—syncmode fast`” in the genesis file in order to speed up the synchronization time. Figure 9 shows the output results after EM synchronization, where the new block 5537 is successfully imported to EM1 after this block is mined by EM2.

The mining processes for each transaction on EM1 and EM2 have been done about 10 times and the average elapsed time is given in Table III. As these results, relied on the system performance of EM2, our system design can be applied for any other emergency services.

C. *Security Analysis*

In our prototype, we apply private Ethereum blockchain to handle transactions among untrusted parties between smart home IoT devices and public service providers. Furthermore, PKI and digital signature with IPFS for emergency calls can prevent DDOS and man-in-the middle attacks and protect the privacy information of the home owner with RSA encryption. We employ strong user password policy for web-based applications for edge users as homeowner and public service providers to prevent attack from web interfaces and apply one time pass code for generating QR code based on service provider staff’s phone number to verify and control the staff to access at the homeowner house.



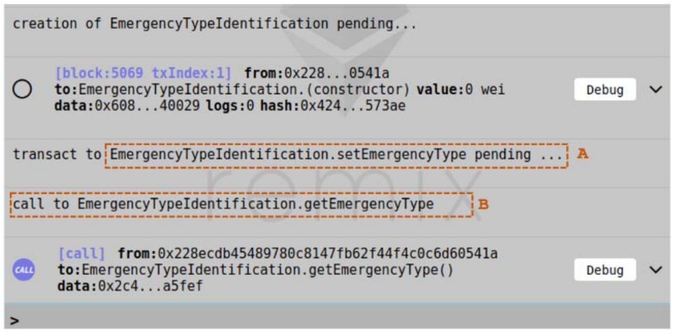
```
contract EmergencyCallService {
    string public IPFS;
    event stringChanged(string indexed changedString);

    function setEmergencyType(string memory _IPFS) public {
        IPFS=_IPFS;
       emit stringChanged(IPFS);
    }
    function getEmergencyType() public view returns (string memory){
        return IPFS;
    }
    string public message;
    mapping (address => string) message;

    function HSP_StaffCall(string memory _emergencyType,string IPFS) public{
          message[_ HSPStaff] = _emergencyType;
    }
    function reademergencycall() returns (string memeory){
      return message[msg.sender];
    }
    …
}
```

Fig. 4. Comparison of gas usage and transaction size among three sequential blocks

V. Conclusions and Future Work

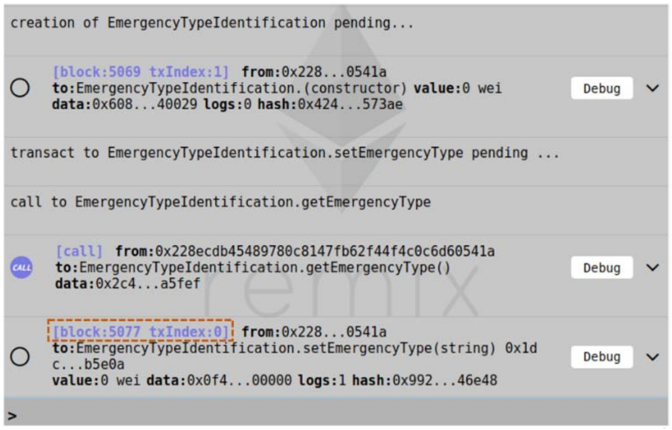
In this paper, we presented our proposed work architecture design and implementation results for emergency service for Smart Home System (SHS) based on the Ethereum private blockchain. We implemented smart contract using Solidity language to handle and record the transactions between the homeowners, Ethereum miners and Home Service Provider

(HSP) Staffs. We described installation and configuration of our Ethereum miner systems and also showed how to build PKI, IPFS with digital signature when a sensor manager from SHS sends an emergency call to HSPs in order to prevent DDoS attack from any rogue IoT device. The mining process results in Remix were described. Future work, we should include an access control mechanism with lifetime limitation QR code and design gas usage for each transaction as well as ethereum wallet charge for each emergency call service.



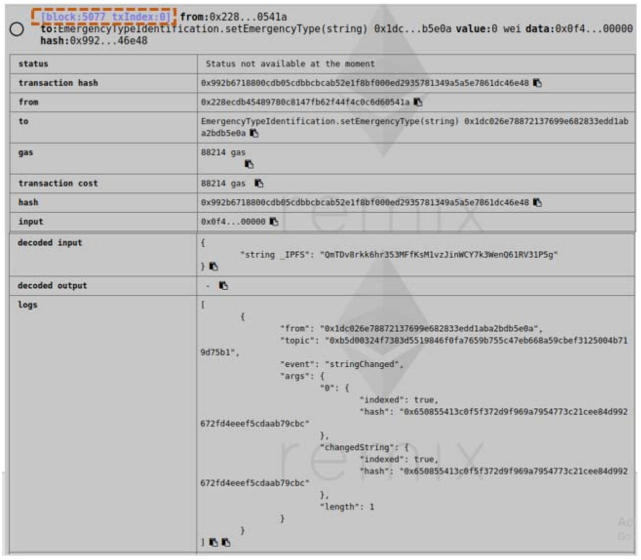
```
creation of EmergencyTypeIdentification pending...

[block:5069 txIndex:1] from:0x228...0541a
to:EmergencyTypeIdentification.(constructor) value:0 wei
data:0x608...40029 logs:0 hash:0x424...573ae

transact to EmergencyTypeIdentification.setEmergencyType pending ...

call to EmergencyTypeIdentification.getEmergencyType

[call] from:0x228ecdb45489780c8147fb62f44f4c0c6d60541a
to:EmergencyTypeIdentification.getEmergencyType()
data:0x2c4...a5fef
>
```

Fig. 5. Pending transactions are appeared after calling functions from Figure 3



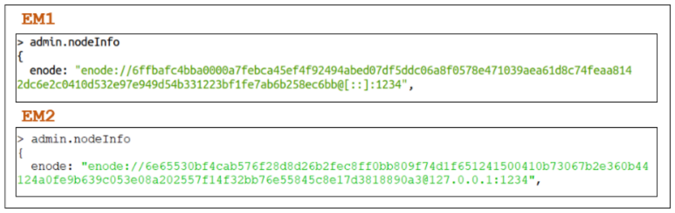
```
creation of EmergencyTypeIdentification pending...

[block:5069 txIndex:1] from:0x228...0541a
to:EmergencyTypeIdentification.(constructor) value:0 wei
data:0x608...40029 logs:0 hash:0x424...573ae

transact to EmergencyTypeIdentification.setEmergencyType pending ...

call to EmergencyTypeIdentification.getEmergencyType

[call] from:0x228ecdb45489780c8147fb62f44f4c0c6d60541a
to:EmergencyTypeIdentification.getEmergencyType()
data:0x2c4...a5fef

[block:5077 txIndex:0] from:0x228...0541a
to:EmergencyTypeIdentification.setEmergencyType(string) 0x1d
c...b5e0a
value:0 wei data:0x0f4...00000 logs:1 hash:0x992...46e48
>
```

Fig. 6. The new block 5077 is retrieved after mining the pending transactions of Figure 4



```
[block:5077 txIndex:0] from:0x228...0541a
to:EmergencyTypeIdentification.setEmergencyType(string) 0x1dc...b5e0a value:0 wei data:0x0f4...00000
hash:0x992...46e48

status               Status not available at the moment
transaction hash     0x992b6718800cdb05cdbbcbcab52e1f8bf000ed2935781349a5a5e7861dc46e48
from                 0x228ecdb45489780c8147fb62f44f4c0c6d60541a
to                   EmergencyTypeIdentification.setEmergencyType(string) 0x1dc026e78872137699e682833edd1ab
                     a2bdb5e0a
gas                  88214 gas
transaction cost     88214 gas
hash                 0x992b6718800cdb05cdbbcbcab52e1f8bf000ed2935781349a5a5e7861dc46e48
input                0x0f4...00000
decoded input        {
                             "string _IPFS": "QmTDv8rkk6hr353MFfKsM1vzJinWCY7k3WenQ61RV31P5g"
                     }
decoded output        -
logs                 [
                             {
                                     "from": "0x1dc026e78872137699e682833edd1aba2bdb5e0a",
                                     "topic": "0xb5d00324f7383d5519846f0fa7659b755c47eb668a59cbef3125004b71
                     9d75b1",
                                     "event": "stringChanged",
                                     "args": {
                                             "0": {
                                                     "indexed": true,
                                                     "hash": "0x650855413c0f5f372d9f969a7954773c21cee84d992
                     672fd4eeef5cdaab79cbc"
                                             },
                                             "changedString": {
                                                     "indexed": true,
                                                     "hash": "0x650855413c0f5f372d9f969a7954773c21cee84d992
                     672fd4eeef5cdaab79cbc"
                                             },
                                             "length": 1
                                     }
                             }
                     ]
```

Fig. 7. The details of the block number 5077



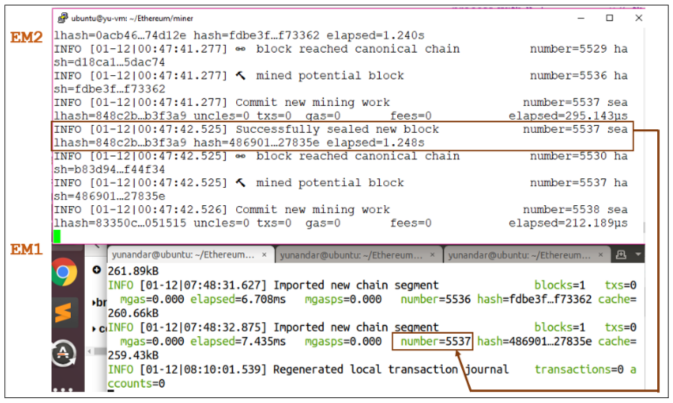
EM1

```
> admin.nodeInfo
{
  enode: "enode://6ffbafc4bba0000a7febca45ef4f92494abed07df5ddc06a8f0578e471039aea61d8c74feaa814
2dc6e2c0410d532e97e949d54b331223bf1fe7ab6b258ec6bb@[::]:1234",
```

EM2

```
> admin.nodeInfo
{
  enode: "enode://6e65530bf4cab576f28d8d26b2fec8ff0bb809f74d1f651241500410b73067b2e360b44
124a0fe9b639c053e08a202557f14f32bb76e55845c8e17d3818890a3@127.0.0.1:1234",
```

Fig. 8. Node information of EM1 and EM2



```
lhash=0acb46…74d12e hash=fdbe3f…f73362 elapsed=1.240s
INFO [01-12|00:47:41.277] ⚯ block reached canonical chain          number=5529 ha
sh=d18ca1…5dac74
INFO [01-12|00:47:41.277] ⛏ mined potential block                  number=5536 ha
sh=fdbe3f…f73362
INFO [01-12|00:47:41.277] Commit new mining work                    number=5537 sea
lhash=848c2b…b3f3a9 uncles=0 txs=0  gas=0       fees=0        elapsed=295.143µs
INFO [01-12|00:47:42.525] Successfully sealed new block             number=5537 sea
lhash=848c2b…b3f3a9 hash=486901…27835e elapsed=1.248s
INFO [01-12|00:47:42.525] ⚯ block reached canonical chain          number=5530 ha
sh=b83d94…f44f34
INFO [01-12|00:47:42.525] ⛏ mined potential block                  number=5537 ha
sh=486901…27835e
INFO [01-12|00:47:42.526] Commit new mining work                    number=5538 sea
lhash=83350c…051515 uncles=0 txs=0  gas=0       fees=0        elapsed=212.189µs
```

EM1
yunandar@ubuntu: ~/Ethereum...
yunandar@ubuntu: ~/Ethereum...
yunandar@ubuntu: ~/Ethereum...

```
261.89kB
INFO [01-12|07:48:31.627] Imported new chain segment               blocks=1   txs=0
  mgas=0.000 elapsed=6.708ms   mgasps=0.000   number=5536 hash=fdbe3f…f73362 cache=
260.66kB
INFO [01-12|07:48:32.875] Imported new chain segment               blocks=1   txs=0
  mgas=0.000 elapsed=7.435ms   mgasps=0.000   number=5537 hash=486901…27835e cache=
259.43kB
INFO [01-12|08:10:01.539] Regenerated local transaction journal    transactions=0 a
ccounts=0
```

Fig. 9. Output results after synchronization between EMs

TABLE III. AVERAGE ELAPSED TIME FOR EACH TRANSACTION

Transaction Types	**Average elapsed time (ms)**	
	EM1	**EM2**
Solidity Smart Contract Creation	7460	7.19
Emergency Type Identification	8330	2.18
HSP Staffs Call	5240	2.00
HSP Staff Respond	10470	2.55
QR Code generation for the HSP staff	3630	1.53
Service Fee Payment	9860	2.38

REFERENCES

[1] X. Wen and Y. Wang, "A Collaborative System Business Model for Ambient Assisted Living Systems," The 4th IEEE International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems, September 2018.

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

[3] 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.

[4] NRI, "Survey on blockchain technologies and related services," Tech. Rep., 2015. [Online]. Available: http://www.meti.go.jp/english/press/ 2016/pdf/0531 01f.pdf. [Accessed: 1- Apr - 2018].

[5] 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.

[6] 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.

[7] C. Dannen, "Introducing Ethereum and Solidtty: Foundations of Cryptocurrency and Blockchain Programming for Beginner", Brooklyn, New York, USA, ISBN-13 (electronic), 2017.

[8] Hanada Y, Hsiao L, Levis P. Smart Contracts for Machine-to-Machine Communication: Possibilities and Limitations, 2018.

[9] D. Patel, J. Bothra and V. Patel, "Blockchain exhumed," 2017 ISEA Asia Security and Privacy (ISEASP), Surat, 2017, pp. 1-12, 2017.

[10] A. Yunandar and T. Tantidham. “Ethereum-based Emergency Service for Smart Home System: Smart Contract Implementation,” accepted, to be appeared in Proceedings of the 21st International Conference of Advanced Communications Technology, 2019 Feb 17-20, Phoenix Park, PyeongChang, Korea (s).

[11] Park JS, Youn TY, Kim HB, Rhee KH, Shin SU. Smart Contract-Based Review System for an IoT Data Marketplace. Sensors. 2018 Oct 22;18(10):3577.

[12] “Remix,” [Online]. Available: https://github.com/ethereum/remix [Accessed: 1- Aug – 2018].

[13] 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, pp. 557-564, 2017.

[14] “IPFS is the Distributed Web,” [Online]. Available: https://ipfs.io/ [Accessed: 1-Aug-2018]

[15] Simmons GJ. Symmetric and asymmetric encryption. ACM Computing Surveys (CSUR). 1979 Dec 1;11(4):305-30.

[16] Thakur M. Authentication, Authorization and Accounting with Ethereum Blockchain, 2017.

[17] 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.

[18] 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

[19] 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.

[20] D. Turnbull, Your first meteor application, Kindle Amazon, 2014.

[21] “Ethereum”, [Online] Available. https://www.ethereum.org/ [Accessed: 1- Dec - 2018].

[22] C. Dannen, “Introducing Ethereum and Solidtty: Foundations of Cryptocurrency and Blockchain Programming for Beginner”, Brooklyn, New York, USA, ISBN-13 (electronic), 2017.

[23] “IPFS”, [Online] Available. https://docs.ipfs.io/introduction/install/ [Accessed: 1- Oct - 2018].

[24] “Installation Instructions for Ubuntu”, [Online] Available. https://github.com/ethereum/go-ethereum/wiki/Installation-Instructions-for-Ubuntu [Accessed: 10- Oct - 2017].

[25] D. Turnbull, Your first meteor application, Kindle Amazon, 2014.

[26] Singla A, Bertino E. Blockchain-based PKI solutions for IoT. In2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC) 2018 Oct 18 (pp. 9-15). IEEE.