Managing IoT Devices using Blockchain Platform

Seyoung Huh*, Sangrae Cho*, Soohyung Kim*
*ETRI, Daejeon, South Korea
one@etri.re.kr, sangrae@etri.re.kr, lifewsky@etri.re.kr

Abstract — **Since the start of Bitcoin in 2008[1], blockchain technology emerged as the next revolutionary technology. Though blockchain started off as a core technology of Bitcoin, its use cases are expanding to many other areas including finances, Internet of Things (IoT), security and such[2]. Currently, many private and public sectors are diving into the technology[3]. Aside from that, as software and hardware improve, we would see the beginning of IoT. And those IoT devices need to communicate and synchronize with each other. But in situations where more than thousands or tens of thousands of IoT devices connected, we expect that using current model of server-client may have some limitations and issues while in synchronization. So, we propose using blockchain to build IoT system. Using blockchain, we can control and configure IoT devices. We manage keys using RSA public key cryptosystems where public keys are stored in Ethereum and private keys are saved on individual devices. Specifically, we choose Ethereum as our blockchain platform because using its smart contract, we can write our own Turing-complete code to run on top of Ethereum. Thus, we can easily manage configuration of IoT devices and build key management system. Even though we can simply use account as a key management system, which most of blockchain platform supports, we decide to use Ethereum because we can manage the system in a more fine-grained way. For the proof of a concept, we use a few IoT devices instead of a full system of IoT system, which consists of thousands of IoT devices. But in our later study, we would like to build a fully scaled IoT system using blockchain.**

Keywords— **Blockchain, IoT, Ethereum, Key Management, Information & Network Security, Authentication, Smart Contract**

I. Introduction

Since the emergence of Bitcoin in 2008 initiated by Satoshi Nakamoto[4], many people invested or speculated on Bitcoin. While Bitcoin itself has an economic, philosophical, and technical significance, all of those revolutions could not be possible without blockchain, distributed ledger, which is a pivotal part of Bitcoin. After success of Bitcoin, many other cryptocurrencies emerged[5], which obviously is built on top of blockchain technology. Not only cryptocurrencies, but also other fields are adapting blockchain[6]. Also, while Bitcoin was a great success, it has some limitations[7]. First of all, its block generation period is about ten minutes[8], which is relatively long to make transactions. Second, while it can keep track of transactions as UTXO[9](Unspent Transaction Outputs) and supports scripting[10], it cannot use loops. In other words, it is not Turing complete[10,11]. With these limitations in mind, Ethereum comes into play. Ethereum with approximately 12 second block period[11] lets developers write smart contract[12]. In other words, developers can write a program that can run on top of Ethereum. Given that it is running on blockchain, developers and users can assume that it cannot be modified without permission and is transparent across Ethereum. Simply put, we can think of Ethereum as a massive shared computing system.

Using Ethereum, we can configure IoT devices. We can manage public key infrastructure in order to authenticate. IoT devices can rely on Ethereum for updating their behaviour. As the era of IoT started, many domains are adapting IoT[13]. Places where more than hundreds of devices need to be connected such as factories are attempting to use such technology[14]. But there are some issues. First, given there are more than hundreds of devices interconnected, it is a hassle to synchronize all of devices. Second of all, just like any other server-client model, if server is vulnerable, all of relying devices will be in trouble. However using blockchain on IoT, IoT devices can synchronize easily with other devices because of its distributed ledger. Also, using its consensus algorithm, it will be hard to forge data in the blockchain or accomplish denial of service (DOS) attack unless there are innate issues such as problems with OPCODE[15].

Thus we propose using Ethereum on managing IoT devices. Using smart contract on Ethereum, we can write a code that defines behaviours of IoT devices. We can also build public key infrastructure on smart contract so malicious attackers cannot control over the management system on Ethereum. In the beginning, we start a proof of concept, which includes a few IoT devices using Raspberry Pi and a smartphone. Once we complete the model, we would like to build fully-scaled IoT system using Ethereum.

II. Related Work

A. Ethereum

Proposed by Vatalik Buterin in 2013, Ethereum is a public blockchain-based distributed computing platform[7,16,18]. Unlike previous blockchain such as Bitcoin, it can work as a computer even though the performance will be slower than most of current PCs since it has a transaction time of around 12 seconds. But, because it has its own language such as Solidity or Serpent[13], it lets developers write and compile a program. Once compiled, it can run on Ethereum Virtual Machine[14]. Just like any other computing environment, once it gets compiled, compiled code gets translated to opcode and then binary, which will be executed on Ethereum Virtual

Machine environment. Thus Ethereum is unique in a sense that it combines computing system with blockchain. It is ground-breaking because it gives developers flexibility to write a code that can run on blockchain. Because it will be difficult to maliciously manipulate or tamper the code, users who rely on the written code are almost guaranteed that it will behave as they expect it to. Even though attacks such as DAO or computational denial of service happened recently, it was due to vulnerabilities of smart contract code or opcode gas price not vulnerabilities on fundamentals of blockchain or Ethereum itself. Thus once system is stabilized and matures, it would become a stronger system.

Given that the system is stabilized, it can be used in wide ranges of domain. Due to its transparency because people can look at its publicly available logic or code of smart contracts, betting or gambling service can be implemented and used. Voting service can easily be implemented with strong certainty that the result is not manipulated or forged. Thus there are many companies, industries, and people who are trying to find their own use cases of Ethereum[17].

III. Managing IoT Devices using Ethereum

In this section, we would like to discuss how we can manage IoT devices using Ethereum.

A. Scenario

As mentioned previously, we used a few IoT devices instead of hundreds of devices in order to make a proof of a concept. More specifically, we use a smart phone, and three Raspberry Pis. For three Raspberry Pis, we treat each of them as a meter to keep track of electricity usage, an air conditioner, and a lightbulb since using actual device such as air conditioner would require too much overhead. Using smart phone, user can set up the policy. For example, user can set devices to turn on energy saving mode when electricity usage hits 150 KW. When the user sets up the configuration via smartphone, the data is sent to the Ethereum network. In the meantime, devices such as lightbulb or air conditioner are retrieving values of policy periodically from Ethereum. Also meter keeps track of electricity usage and updates it on Ethereum. Thus three different processes are happening concurrently. The below image is shown as a diagram of our scenario.

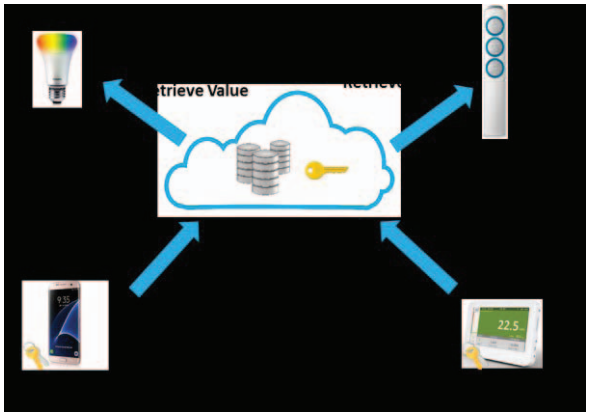

Figure 1. Diagram of Ethereum IoT Scenario

B. Ethereum Model

Unlike server-client model, Ethereum is a distributed computing platform[7,16,18], which means all of participating entities contain parts of blockchain of Ethereum. Though Figure 1 looks similar to server-client model for simplicity, actual model looks different in a sense that each contributing entity of blockchain contains blockchain partially or entirely. Instead of sending to server, each device updating or making transactions contains Ethereum as shown in figure 2.

Figure2. Diagram of Devices connected to Ethereum

Because blockchain is partially contained in contributing devices, we know that transactions are executed and stored via consensus algorithm, which means attackers cannot forge or tamper data easily. Leveraging this characteristic lets us to build IoT system, which is strong enough to stand against many denial of service attacks and forgery attacks, if not all. Even though we are given only a few devices for this paper, we believe it will be possible to synchronize hundreds of devices.

C. Smart Contract

One of the most significant aspects of Ethereum lies in smart contract. The concept first introduced by Nick Szabo in 1994[19], smart contract brought innovation to blockchain. Ethereum uses smart contract on top of blockchain so that developers can write a program on blockchain. In other words, using smart contract, we can use Ethereum as a computing platform. There are programming languages such as Solidity, Serpent, and LLL in Ethereum[20]. At this point, solidity is the most widely used language and compiler[20]. This high level language once developed is compiled into byte codes. And those byte codes are deployed onto Ethereum. And since byte codes are simply a list of opcodes, Ethereum nodes follow those instructions in the code once the corresponding contract is executed from valid account.

For our experiment, we wrote three smart contracts. We wrote one contract tracking the value of meter. And we wrote one for saving policy values of each air conditioner and lightbulb. In order to authenticate valid account, we also added signature and public key on both contracts. Thus, if malicious attackers try to manipulate the storage in smart contracts, computing systems in air conditioner or lightbulb can detect such attack and simply ignore them.

Meter Contract: On a smart contract, meter periodically saves electricity use. Precisely, Raspberry Pi containing Ethereum account acts as an IoT device to monitor meter and sends the value to Ethereum. In order to show the identity of sender, sender needs to send its public key and signature along with electricity use.

```
contract Meter{
  int value;
  bytes publicKey;
  bytes signature;
  update(int _value, bytes _publicKey, bytes _signature){
    value = _value;
    publicKey = _publicKey;
    signature = _signature;
  }
}
```

Figure3. Simple Pseudo code of Meter Contract

As seen in Figure 3, logic is straightforward. Meter device which contains Ethereum blockchain can simply sign and send input values of *value*, *publicKey*, and *signature*. Since each contract has its own unique address, we know where to send data. In order to use *update* method, we need to encode input values so it can run on Ethereum Virtual Machine. For example, if we want to run *update*(300, "deadbeef", "babebabe"), we need to encode it into binary. If we encode it, we would get the following:

```
f1de8ccb0000000000000000000000000000000000000000000000000
0000000000000000012c0000000000000000000000000000000000
0000000000000000000000000000000000600000000000000000
0000000000000000000000000000000000000000000000000a0000
0000000000000000000000000000000000000000000000000000000
0000000004deadbeef0000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000
00000000000000000000000000000004babebabe0000000
0000000000000000000000000000000000000000000000000000
```

Figure4. Encoded value of update(300, "deadbeef", "babebabe")

Even though the full explanation of encoding is out of scope in this paper, we can briefly go over in order to understand the general structure of Ethereum. To run *update(int,bytes,bytes)*, we need to hash it using Keccak-256 and take first four bytes of hashed output[21]. In other words, we need first four bytes Keccak256("*update(int,bytes,bytes)*"), which is f1d38ccb. 12c signifies 300. And "deadbeef" and"babebabe" are our bytes inputs. Values in between signify offsets and length of bytes.

Thus using meter contract, meter that is attached to Ethereum account can send values periodically to blockchain, which other entities such as air conditioner or lightbulb can retrieve.

Policy Contract: Along with meter, we need to set up policies. As mentioned in the scenario, we would like to set up policies for air conditioner and lightbulb. Given that we set policy of air conditioner as 150KW, once the meter reaches 150KW, air conditioner simply switches from normal mode to energy saving mode. The same scenario applies to lightbulb too. Similar to meter contract, policy needs to be registered. But instead of meter, we use smartphone to register. Smartphone that contains Ethereum account can send data to smart contract that manages policies.

```
contract ACPolicy{
  int acLimit;
  bytes publicKey;
  bytes signature;
  update(int _acLimit, bytes _publicKey, bytes _signature){
    acLimit= _acLimit;
    publicKey = _publicKey;
    signature = _signature;
  }
}
```

Figure5. Simple Pseudo code of Air Conditioner Policy Contract

Just like Figure3, logic of Figure5 is almost identical. And we omit light bulb policy since it is almost identical as well. Once Ethereum account encodes policy with public key and signature, and sends them to Ethereum, those values are registered in contract.

Key Management:

Given that all of those contracts are running on Ethereum, IoT devices such as air conditioner or light bulb need to retrieve values from both meter contract and policy contract. On values coming from meter contract, they check if inputs are valid using public key and signature. To be precisely, we are using RSA algorithm, which smart phone and meter keep their secret keys. And on values from policy contracts, they also check if inputs are valid using public key and signature. If the scenario where electricity use surpasses policy while periodically retrieving values, devices simply switch to energy saving mode. Even though it is possible to use Ethereum accounts as public key infrastructure since it is based on ECDSA[22], we decide to use our own because of fine-granularity.

IV. Simulation Results and Discussion

We deployed smart contracts on Ethereum. Once we deployed contracts, we started to provide inputs after encoding. Once we have successfully updated/registered values on Ethereum, we were able to retrieve values from Ethereum. Our finished prototype looks as the following.

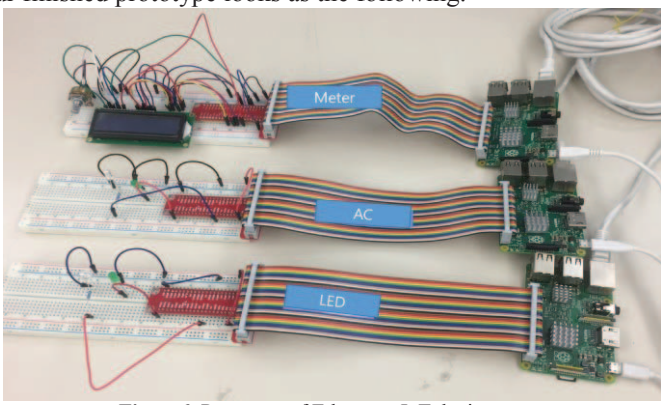

Figure6. Prototype of Ethereum IoT devices

As seen in the Figure 6, we used Raspberry Pis to simulate IoT system. We set up meter, which updates to Ethereum network periodically. We used a smartphone to set up the policies of air conditioner and LED. And those two devices respond according to policies given from Ethereum.

In the process of development, we find that there are some weaknesses on Ethereum blockchain. First of all, even though it has around 12 second transaction time, it still is not fast enough for some domains. For time sensitive domain it may be difficult to use such technology. Second of all, since light client is not supported on Ethereum at this point, either we need to use a proxy or have a large storage to save entire blockchain. Using proxy may be easy. But we compromise security because there is a third party involved. Second solution, while it does not compromise security, may require large storage, which would be too expensive or infeasible for small IoT devices. Thus we would need to investigate further on solutions for those weaknesses.

V. Conclusion

In this paper, we propose a way to manage IoT devices using Ethereum, blockchain computing platform. We write smart contracts to save data coming from meter and smart phone. Using Ethereum account, meter constantly sends electricity use and smart phone sends policies for air conditioner and light bulb. And air conditioner and lightbulb constantly checks the values on Ethereum to update their devices. When necessary, they switch their mode from normal to energy-saving. As proof of concept, we are starting with small number of devices. Since we found that it is feasible to build such a system, in further studies, we would like to build fully-scaled IoT system which contains multiple of devices. With the start of this experiment, we hope to see improvements on IoT where users of the technology do not need to worry about synchronization and denial of service attacks while serving them efficiently and fast.

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP)
(No. B0717-16-0114, Development of Biometrics-based Key Infrastructure Technology for On-line Identification)

References

[1] S. Nakamoto, *Bitcoin: Peer-to-Peer Electronic Cash System*, 2008
[2] "17 Blockchain Disruptive Use Cases." Everis NEXT. Everis NEXT, 02 June 2016. Web. 03 Jan. 2017.
[3] A. Shelkovnikov, *Blockchain applications in the public sector*, Deloitte, 2016
[4] J. Brito, *Bitcoin A Primer for Policymakers*, 2013
[5] Coinmarketcap.com, *https://coinmarketcap.com*, 2016/10/27
[6] G. Greenspan, Four Geuine Blockchain Use Cases, Coindesk.com. May, 2016. Available: http://www.coindesk.com/four-genuine-blockchain-use-cases/
[7] Ethereum White Paper, https://github.com/ethereum/wiki/wiki/White-Paper, accessed 2016/10/31
[8] Bitcoinwiki, https://en.bitcoin.it/wiki/Block, accessed 2016/10/31
[9] Bitcoin Developer Guide, https://bitcoin.org, accessed 2016/10/31
[10] Bitcoinwiki, https://en.bitcoin.it/wiki/Script, accessed 2016/10/31
[11] V. Buterin, *Toward a 12-second Block Time*, Ethereum Blog, 2014
[12] G. Wood, Ethereum: A Secure Decentralised Generalised Transaction Ledger, http://gavwood.com/paper.pdf, accessed 2016/10/31
[13] J Bughin, *An Executive's guide to the Internet of Things*, McKinsey, 2016
[14] PwC, *The Internet of Things has arrived in America's factories*, 2015
[15] J. Wilcke, *The Ethereum network is currently undergoing a DoS attack*, Sep 2016. Available: https://blog.ethereum.org/2016/09/22/ethereum-network-currently-undergoing-dos-attack/
[16] Ethereum, Writing a Contract, https://github.com/ethereum/go-ethereum/wiki/Contracts-and-Transactions, accessed 2016/10/31
[17] P. Rizzo, *Thomson Reuters Demos New Ethereum Blockchain Use Cases*, Sep 2016. Available: http://www.coindesk.com/thomson-reuters-blockchain-ethereum-devcon2/
[18] G. Wood, Ethereum: A Secure Decentralised Generalised Transaction Ledger, http://gavwood.com/paper.pdf, accessed 2016/10/31
[19] M. Gord, *Smart Contracts Described by Nick Szabo 20 Years ago Now Becoming Reality*, Bitcoin Magazine, 2016
[20] Ethereum Frontier Guide, https://ethereum.gitbooks.io/frontier-guide, accessed 2016/10/31
[21] Ethereum Contract ABI, https://github.com/ethereum/wiki/wiki/Ethereum-Contract-ABI, accessed 2016/10/31
[22] H. Mayer, ECDSA Security in Bitcoin and Ethereum: a Research Survey, 2016, Available: http://blog.coinfabrik.com/ecdsa-security-in-bitcoin-and-ethereum-a-research-survey/

Seyoung Huh is a researcher of Authentication Research Team in ETRI, South Korea. His research interests include blockchain, authentication, cryptography, and any security-related fields.

Sangrae Cho is a senior researcher of Authentication Research Team in ETRI, South Korea. He graduated from Imperial College London in 1996 obtained BEng Computing degree and studied MSc in Information Security in Royal Holloway, University of London in 1997. He started his career as a researcher in LG Corporate Technology Institute in 1997 and has worked in ETRI for over 15 years as a security researcher. During that time, he has actively involved in constructing national PKI infrastructure project until 2001. From 2004, he has done several projects relating to Digital Identity Management including SAML v2.0 and Authentication technology based on FIDO (Fast Identity Online) specifications.