ICE Block: Emergency Contacts

Jeongmin Oh
M.S. in Computer Engineering
Columbia University
New York, United States
jeongmin.oh@columbia.edu

***Abstract*—I present concept and prototype of In Case of Emergency Block(ICE Block), an Emergency Contacts on Blockchain. Through the ICE Block, (1) Users can encrypt and store their emergency medical information and contact in Public Blockchain. (2) Whenever the information are necessary, anyone with a key pair can decrypt and view the information. (3) It is provided as a service of Public Blockchain, so that anyone can have a copy of the entire Databse, (4) and the information can be read through the copy even if the internet connection is temporarily impossible.**

***Index Terms*—Internet Service, Blockchain, Emergency**

I. Background

Having more information about patients is crucial to prescribe medical treatments. If it is possible, medical staffs will be able to treat the patients more accurately and quickly. However, it is not easy for them to obtain the information without having patient's concent. In an emergency situation, the problem can be further deepened. For example, suppose you have a sudden accident at road. And, Fortunately, someone reported the accident. First respondents will carry you to the nearest hospital. In bad situation, they need to follow remote direction from medical staff who is located in remote. What if they are tyring to give an injection when you are in coma? What if you have a specific response to the injection, or a history of a particular chronic disease they need to know? If you are lucky, you body doesn't react to emergency prescriptions, or even worse.

We need to save our basic medical information and contacts that is easily accessible by anyone who can help people who is in emergency situation. There are several solution is available today for this. First, We can use local storage like cell phone that can show your basic medical information and contact list without unlock it. Second, Remote online server can be an alternative. Third one is leaving physical notes somewhere near us. All of these solutions have shortcomings and problems that is ICE Block solved. ICE Block resolves limitations of existing solutions and potentially can be extended its service scope beyond its original purpose.

II. Previous Work

A. Using Local Storage

Mobile devices like cell phone provide easy way of saving information in local. People carry cell phone almost everywhere they go. So, it will be most likely found every accident scene with patients. Some of the cell phone maker providing functionality of emergency medical information using local storage. For example, Apple's Medical ID [1] provide local storage which is available without unlock the device. Their solution seems pausible but has some limitation. It is not available when the local storage is not accessible because of physical damage or no power. To provide high availability, the information should be stored in remote storage.

B. Using Remote Server

Some of the goverment provide emergency contact service. Illinois's Emergency Contact Database is good example. User voluntarily save their information on the database and it can be used in emergency. I assume their solution built on conventional architecture that is expansive to scale. Prodiving high reliability with great scalability is burden for any organization.

Using Cloud repository might be a solution but it also has some limitation. Store and backup a few of line of characters on global cloud service like Goole Docs might be simple and economic solution but user need to think it will be really available everywhere. The service may not cross border because there might be political issues.

III. Design Principle

To overcome previous solutions limitation.

a) High Availability: Mobile device can fail -¿ remote server centralized db can fail -¿ replica

b) Sensitive Information: Encryption

c) Low Cost: To ran by any body even in cell phone

d) Scalability: Lower down

A. Design

B. Server

ICE Block Server gets transactions from muliple clients. It needs to validate the transaction and create a new block. The server also needs to keep data consistancy throughout the network. It use consensus algorithm to select a chain from neighbors.

a) Validation of Transaction: Client send a transaction that is containing account_ID, encrypted record and signature. Server needs to validate the transaction whether it is created by owner of private key or not because public key is also used account_ID in ICE Block.

Columbia University COMS E6181 Advanced Internet Services Fall 2018 by Professor Henning Schulzrinne

Algorithm 1 Algorithm for Validation of Transaction

Input: account_ID, record, signature

Initialisation :

validation = **false**
public_key = importKey(account_ID)
verifier = PKCS1_v1_5.new(public_key)
h = hash(account_ID, record)

Check Validity :

validation = verifier.verify(h, signature)
if $validation\ is\ true$ **then**
    Push the transaction in the buffer
end if

b) Proof of Work Algorithm: Every nodes need to prove that they invested some amount of effort to create a new block as nonce. nonce is number of try the node run hash function to find the hash ending with several cascaded zeros. Proof of work algorithm prevents divergence of chain with increasing mining difficulty. [4]

Algorithm 2 Algorithm for Proof of Work

Input: last_hash, transactions
Output: nonce

Initialisation :

test
nonce = 0
valid_proof = **false**

Loop over to find valid proof :

while $valid\_proof\ is\ false$ **do**
    guess_hash = hash(transactions, last_hash, nonce)
    nonce += 1
    if $guess\_hash[:difficulty] ==' 0' * difficulty$ **then**
        valid_proof = **true**
    else
        valid_proof = **false**
    end if
end while
return $nonce$

c) Conflict Resolution Algorithm: Server needs to resolve conflict when they add neighbors. The neighbors may have diffent chain. Valid and longest chain should be selected in the network. And every other nodes should update their chain using the chain. Propagation of the chain in the network is important to keep data integrity. In my prototype conflict resolution will be excuted when server refresh their view of chain. It needs to be optimized futher, it is getting whole set of chain from neighbor. Rathern than sending whole blocks of chain, sending just freshly created block seems more reasonablbe. Of course checking up whole chain's validity is also important for security reason.

C. Client

Client provide funtionality of creating pair of public and private key. public key is used for account_ID, encryption and signature validation. private key is used for decryption and sign transaction. I assume that user can provide the private key when they need it. It might be a type of QRCode or RFID.

Algorithm 3 Algorithm for Conflict Resolution

Input: neighbors

Loop over to find longer chain :

for $neighbor\ in\ neighbors$ **do**
    Get the chain from neighbor
    if $the\ chain >= self\_chain$ **and** $valid\_chain$ **then**
        Replace self_chain with the chain
    end if
end for

a) Account ID and Password: Account_ID is string representation of public key that is also being used for encryption and signature validation. I chose to RSA 1024 bits to randomly create a key pair. Created binary representation of key pair converted into hexa decimal representation and than decoded using ascii code. User can create a key pair whenever they need it.

b) Encryption and Decryption: Client can encrypt their record that is user provided sensitive information and contact. To cipher the information, ICE Block use public key that is same as account_ID. Decryption works similar way of its reverse order. Client can retrieve encrypted information from the server and use private key to decrypt the information.

c) Signature: Technically, Everyone who knows account_ID can spoil it's records by creating invalid trasactions. To prevent this problem, ICE Block provides signature that is used in validation algorithm at server side. I chose to use PKCS1 to create the sign and verification.

IV. Prototype

I implemented prototype of ICE Block using technology below.

- OS: macOS Mojave
- Laguage: Python 3.7.0
- Virtual Environment: pipenv
- Microframework: Flask
- Front-end: Bootstrap, jQuery, DataTables
- Encryption: PyCrypto(SHA, RSA, PKCS1_v1_5)
- Testing: Postman, Safari

Source code of ICE Block is provided on GitHub. [3]

a) Managing Packages: I used pipenv to create and manage python package. It provides pacakges in two categories production and development. The list of package requirement is scripted in Pipfile and Pipfile.lock. These files are used when user install the packages as a reference.

b) Running Program: Both server and client program runs on localhost using ip address of 127.0.0.1. And also, get an argument of port using -p flag from command line. Server use port 5000 and Client use 8080 as default.

V. Demonstration

A.

VI. Experience from prototyping

Throughout the implementation, I learned something new I didn't know before.

a) RSA has maximum bytes to encrypt: When I was tring to encrypt whole information which is user provided, I realized it is not possble. [5] I created 1024 bits of key that means it can encrypt 128 bytes including 11 bytes of padding for PKCS1. I decided encrypt it's value by value. Increasing number of bits of key is not good solution because ICE Block using public key as ID. ID had to be short enough to be represneted in QR Code for good recognition.

b) Hash cares order: ICE Block gets user input as html form and it is treated in Python dictionary. After encryption, the information delivered using JSON format. The problem I was encountred is when creating a Hash for the information. It had to be ordered so I used Python OrderedDict to construct and re-construct.

VII. Futher Work

To provide better interface to user, key representation should be considered. The prototype only shows string representation of Account_ID and private key. Better way to representation technology should be applied for this. QR Code or RFID can be an option. To apply QR Code, User can decide it's visibility But key should be short enough to be easily scanned.

To provide better scalability, Applying difficulty algorithm should be considered because if it is too easy, The chain may not be converged. Notifying new block creation is also should be considered. I presume asynchronous way of consensus can be helpful.

VIII. Conclusions

ICE Block provide scalable database for emergency contact. I implemented prototype and showed its service concept in Demo. It is a light weight public blockchain network anyone can run. User doesn't have to worry about lost or damage of their local storage. And It can be accepted global standard of emergency contact because it will be maintained by participating nodes not the government or organization. It means that first respondent doesn't have to spend more time to identify patients. I hope ICE Block save more lives

References

[1] Apple's Medical ID
https://support.apple.com/en-us/HT207021

[2] Illinois's Emergency Contact Database
https://www.cyberdriveillinois.com/departments/drivers/ECD/home.html

[3] ICE Block GitHub Repository
https://github.com/battlerhythm/iceblock

[4] Bitcoin: A Peer-to-Peer Electronic Cash System, Satoshi Nakamoto
https://bitcoin.org/bitcoin.pdf

[5] RSA Maximum bytes to encrypt
https://security.stackexchange.com/questions/33434/rsa-maximum-bytes-to-encrypt-comparison-to-aes-in-terms-of-security