연구 보고서

작성자	김한호	작성일자	2021.06.13.

1. 연구 계획

- 최종프로젝트 발표 준비

2. 논문 연구 진행

```
h2kim@h2kim-VBox:~/fabric-samples/test-network$ ./network.sh deployCC -ccn enterbasic -ccl go -ccp ..
/e2-enter-basic/chaincode-go/
deploying chaincode on channel 'mychannel'
executing with the following
- CHANNEL_NAME: mychannel
- CC_NAME: enterbasic
- CC_SRC_PATH: ../e2-enter-basic/chaincode-go/
- CC_SRC_LANGUAGE: go
- CC_VERSION: 1.0
- CC_SEQUENCE: 1
- CC_END_POLICY: NA
- CC_COLL_CONFIG: NA
- CC_INIT_FCN: NA
- DELAY: 3
- MAX_RETRY: 5
- VERBOSE: false
```

기존에 만들어 놓은 암호화를 하지 않은 e2-enter-basic을 체인코드를 피어에 배포함.

```
h2kim@h2kim-VBox:~/fabric-samples/test-network$ docker ps
CONTAINER ID        IMAGE
                                                                                          COMMAND
                 CREATED             STATUS              PORTS
 NAMES
7404f52301e9        dev-peer0.org2.example.com-enterbasic_1.0-b31793fa0c09fc0605f25d6e9a7420413bd1e0c
ce5bff4b9c883dfa895be632e-1ee6b60192f5028ab32ccf68f037c27cd5d981fb77bdde3bc35f00fa14cdfa6f   "chainco
de -peer.add…"   50 seconds ago      Up 47 seconds
 dev-peer0.org2.example.com-enterbasic_1.0-b31793fa0c09fc0605f25d6e9a7420413bd1e0cce5bff4b9c883dfa895
be632e
153d5ec35ba7        dev-peer0.org1.example.com-enterbasic_1.0-b31793fa0c09fc0605f25d6e9a7420413bd1e0c
ce5bff4b9c883dfa895be632e-790f75032528b02768cebb3b36b8ac13d3b01848c9ccb98ee216522f84c5de7a   "chainco
de -peer.add…"   50 seconds ago      Up 47 seconds
 dev-peer0.org1.example.com-enterbasic_1.0-b31793fa0c09fc0605f25d6e9a7420413bd1e0cce5bff4b9c883dfa895
be632e
```

체인코드가 컨테이너에 올라간 것을 확인함.

```
h2kim@h2kim-VBox:~/fabric-samples/test-network$ export PATH=${PWD}/../bin:$PATH
h2kim@h2kim-VBox:~/fabric-samples/test-network$ export FABRIC_CFG_PATH=$PWD/../config/
h2kim@h2kim-VBox:~/fabric-samples/test-network$ export CORE_PEER_TLS_ENABLED=true
h2kim@h2kim-VBox:~/fabric-samples/test-network$ export CORE_PEER_LOCALMSPID="Org3MSP"
h2kim@h2kim-VBox:~/fabric-samples/test-network$ export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizatio
ns/peerOrganizations/org3.example.com/peers/peer0.org3.example.com/tls/ca.crt
h2kim@h2kim-VBox:~/fabric-samples/test-network$ export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/p
eerOrganizations/org3.example.com/users/Admin@org3.example.com/msp
h2kim@h2kim-VBox:~/fabric-samples/test-network$ export CORE_PEER_ADDRESS=localhost:11051
```

org3를 올린 컨테이너의 변수를 설정함.

```
h2kim@h2kim-VBox:~/fabric-samples/test-network$ peer lifecycle chaincode package enterbasic.tar.gz --
path ../e2-enter-basic/chaincode-go/ --lang golang --label enterbasic_1.0
h2kim@h2kim-VBox:~/fabric-samples/test-network$ peer lifecycle chaincode install enterbasic.tar.gz
2021-06-07 15:18:22.550 KST [cli.lifecycle.chaincode] submitInstallProposal -> INFO 001 Installed rem
otely: response:<status:200 payload:"\nOenterbasic_1.0:b31793fa0c09fc0605f25d6e9a7420413bd1e0cce5bff4
b9c883dfa895be632e\022\016enterbasic_1.0" >
2021-06-07 15:18:22.550 KST [cli.lifecycle.chaincode] submitInstallProposal -> INFO 002 Chaincode cod
e package identifier: enterbasic_1.0:b31793fa0c09fc0605f25d6e9a7420413bd1e0cce5bff4b9c883dfa895be632e
h2kim@h2kim-VBox:~/fabric-samples/test-network$
```

org3에 체인코드를 올리는 명령을 실행함.

```
h2kim@h2kim-VBox:~/fabric-samples/test-network$ peer lifecycle chaincode package enterbasic.tar.gz --
path ../e2-enter-basic/chaincode-go/ --lang golang --label enterbasic_1.0
h2kim@h2kim-VBox:~/fabric-samples/test-network$ peer lifecycle chaincode install enterbasic.tar.gz
2021-06-07 15:18:22.550 KST [cli.lifecycle.chaincode] submitInstallProposal -> INFO 001 Installed rem
otely: response:<status:200 payload:"\nOenterbasic_1.0:b31793fa0c09fc0605f25d6e9a7420413bd1e0cce5bff4
b9c883dfa895be632e\022\016enterbasic_1.0" >
2021-06-07 15:18:22.550 KST [cli.lifecycle.chaincode] submitInstallProposal -> INFO 002 Chaincode cod
e package identifier: enterbasic_1.0:b31793fa0c09fc0605f25d6e9a7420413bd1e0cce5bff4b9c883dfa895be632e
h2kim@h2kim-VBox:~/fabric-samples/test-network$ peer lifecycle chaincode queryinstalled
Installed chaincodes on peer:
Package ID: enterbasic_1.0:b31793fa0c09fc0605f25d6e9a7420413bd1e0cce5bff4b9c883dfa895be632e, Label: e
nterbasic_1.0
h2kim@h2kim-VBox:~/fabric-samples/test-network$ export CC_PACKAGE_ID=enterbasic_1.0:b31793fa0c09fc060
5f25d6e9a7420413bd1e0cce5bff4b9c883dfa895be632e
h2kim@h2kim-VBox:~/fabric-samples/test-network$ peer lifecycle chaincode approveformyorg -o localhost
:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile ${PWD}/organizations/ordererOrg
anizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --chann
elID mychannel --name enterbasic --version 1.0 --package-id $CC_PACKAGE_ID --sequence 1
2021-06-07 15:24:22.770 KST [chaincodeCmd] ClientWait -> INFO 001 txid [e678b306cf0da26eae8c40e13d3d0
5884e7ecd0b0390f4eedd01a392485ce183] committed with status (VALID) at
h2kim@h2kim-VBox:~/fabric-samples/test-network$ peer lifecycle chaincode querycommitted --channelID m
ychannel --name enterbasic --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/or
derer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
Committed chaincode definition for chaincode 'enterbasic' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: tr
ue, Org2MSP: true, Org3MSP: true]
h2kim@h2kim-VBox:~/fabric-samples/test-network$
```

체인코드가 올라가는 로그를 확인했을 때 정상적으로 올라간 것을 확인할 수 있음.

```
h2kim@h2kim-VBox:~/fabric-samples/test-network$ docker ps
CONTAINER ID        IMAGE
                                                                                                  COMMAND
                  CREATED             STATUS              PORTS
 NAMES
cd31623a0f2c        dev-peer0.org3.example.com-enterbasic_1.0-b31793fa0c09fc0605f25d6e9a7420413bd1e0c
ce5bff4b9c883dfa895be632e-7db74015c64af7b242bfd604dd55f826dd23536a4f0df933ec4cca30fab9581e   "chainco
de -peer.add…"   3 minutes ago       Up 3 minutes
 dev-peer0.org3.example.com-enterbasic_1.0-b31793fa0c09fc0605f25d6e9a7420413bd1e0cce5bff4b9c883dfa895
be632e
7404f52301e9        dev-peer0.org2.example.com-enterbasic_1.0-b31793fa0c09fc0605f25d6e9a7420413bd1e0c
ce5bff4b9c883dfa895be632e-1ee6b60192f5028ab32ccf68f037c27cd5d981fb77bdde3bc35f00fa14cdfa6f   "chainco
de -peer.add…"   17 minutes ago      Up 17 minutes
 dev-peer0.org2.example.com-enterbasic_1.0-b31793fa0c09fc0605f25d6e9a7420413bd1e0cce5bff4b9c883dfa895
be632e
153d5ec35ba7        dev-peer0.org1.example.com-enterbasic_1.0-b31793fa0c09fc0605f25d6e9a7420413bd1e0c
ce5bff4b9c883dfa895be632e-790f75032528b02768cebb3b36b8ac13d3b01848c9ccb98ee216522f84c5de7a   "chainco
de -peer.add…"   17 minutes ago      Up 17 minutes
 dev-peer0.org1.example.com-enterbasic_1.0-b31793fa0c09fc0605f25d6e9a7420413bd1e0cce5bff4b9c883dfa895
be632e
```

기존 org1, org2에 올라있는 체인코드에 더해서 org3에 체인코드가 올라간 것을 확인할 수 있음.

```
h2kim@h2kim-VBox:~/fabric-samples/test-network$ ./network.sh deployCC -ccn enterencrsa -ccl go -ccp .
./e2-enter-encrsa/chaincode-go/
deploying chaincode on channel 'mychannel'
executing with the following
- CHANNEL_NAME: mychannel
- CC_NAME: enterencrsa
- CC_SRC_PATH: ../e2-enter-encrsa/chaincode-go/
- CC_SRC_LANGUAGE: go
- CC_VERSION: 1.0
- CC_SEQUENCE: 1
- CC_END_POLICY: NA
- CC_COLL_CONFIG: NA
- CC_INIT_FCN: NA
- DELAY: 3
- MAX_RETRY: 5
- VERBOSE: false
Vendoring Go dependencies at ../e2-enter-encrsa/chaincode-go/
```

공개키 시스템을 이용한 암호화한 데이터를 저장하는 체인코드를 네트워크에 배포함.

```
h2kim@h2kim-VBox:~/fabric-samples/test-network$ peer lifecycle chaincode package enterencrsa.tar.gz --path ../e2-enter-encrsa/chaincode-go/ --lang golang --label enterencrsa_1.0
h2kim@h2kim-VBox:~/fabric-samples/test-network$ peer lifecycle chaincode install enterencrsa.tar.gz
2021-06-07 16:22:32.537 KST [cli.lifecycle.chaincode] submitInstallProposal -> INFO 001 Installed remotely: response:<status:200 payload:"\nPenterencrsa_1.0:7a21777b8b51579011cd8c603c73b2d780b10c8b80b715538c13b8735c08024b\022\017enterencrsa_1.0" >
2021-06-07 16:22:32.537 KST [cli.lifecycle.chaincode] submitInstallProposal -> INFO 002 Chaincode code package identifier: enterencrsa_1.0:7a21777b8b51579011cd8c603c73b2d780b10c8b80b715538c13b8735c08024b
h2kim@h2kim-VBox:~/fabric-samples/test-network$ export CC_PACKAGE_ID=enterencrsa_1.0:7a21777b8b51579011cd8c603c73b2d780b10c8b80b715538c13b8735c08024b
h2kim@h2kim-VBox:~/fabric-samples/test-network$ peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --channelID mychannel --name enterencrsa --version 1.0 --package-id $CC_PACKAGE_ID --sequence 1
2021-06-07 16:24:17.278 KST [chaincodeCmd] ClientWait -> INFO 001 txid [c29ddc83d5a2ef5b3afb6cc6eb2ea41cd3c607363cf5835590fbc4d93302aaba] committed with status (VALID) at
h2kim@h2kim-VBox:~/fabric-samples/test-network$ peer lifecycle chaincode querycommitted --channelID mychannel --name enterencrsa --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
Committed chaincode definition for chaincode 'enterencrsa' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true, Org3MSP: true]
h2kim@h2kim-VBox:~/fabric-samples/test-network$ peer chaincode query -C mychannel -n enterencrsa -c '{"Args":["GetAllShops"]}'

h2kim@h2kim-VBox:~/fabric-samples/test-network$
```

위의 org3에 체인코드를 올리는 과정도 동일하게 반복함. 여기까지 연구를 진행하고 다음주 화요일에 있을 블록체인과 보안 발표를 위해 보여줄 결과를 내기 위해 암호화된 데이터를 구현을 보여줄 수 있는 화면을 보여주는 구현물을 만들어 내기로 함.

```
h2kim@h2kim-VBox:~/fabric-samples/test-network$ peer chaincode query -C mychannel -n enterencrsa -c '{"Args":["CreateShop", "shop1", "Burger King Gal town-ro, Yeongtong-gu, Suwon-si, Gyeonggi-do, Republic of Korea"]}'

h2kim@h2kim-VBox:~/fabric-samples/test-network$ peer chaincode query -C mychannel -n enterencrsa -c '{"Args":["GetAllShops"]}'

h2kim@h2kim-VBox:~/fabric-samples/test-network$ peer chaincode query -C mychannel -n enterencrsa -c '{"Args":["ReadShop","shop1"]}'
Error: endorsement failure during query. response: status:500 message:"the shop shop1 does not exist"
h2kim@h2kim-VBox:~/fabric-samples/test-network$ peer chaincode query -C mychannel -n enterencrsa -c '{"Args":["CreatePKI", "pki1", "tst1", "test2"]}'

h2kim@h2kim-VBox:~/fabric-samples/test-network$ peer chaincode query -C mychannel -n enterencrsa -c '{"Args":["ReadPKI", "pki1"]}'
Error: endorsement failure during query. response: status:500 message:"the pki pki1 does not exist"
h2kim@h2kim-VBox:~/fabric-samples/test-network$
```

기존에 fabric 1.4 버전에서 사용하던 peer chaincode query를 이용하여 화면에 데모를 실행하려 했으나 작동하지 않는 것을 확인할 수 있었음.

```
h2kim@h2kim-VBox:~/fabric-samples/e2-enter-encrsa/application-javascript$ node app.js

--> Fabric client user & Gateway init: Using Org1 identity to Org1 Peer
Loaded the network configuration located at /home/h2kim/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/connection-org1.json
Built a CA Client named ca-org1
Built a file system wallet at /home/h2kim/fabric-samples/e2-enter-encrsa/application-javascript/wallet/org1
An identity for the admin user already exists in the wallet

--> Fabric client user & Gateway init: Using Org2 identity to Org2 Peer
Loaded the network configuration located at /home/h2kim/fabric-samples/test-network/organizations/peerOrganizations/org2.example.com/connection-org2.json
Built a CA Client named ca-org2
Built a file system wallet at /home/h2kim/fabric-samples/e2-enter-encrsa/application-javascript/wallet/org2
An identity for the admin user already exists in the wallet
Successfully registered and enrolled user appUser2 and imported it into the wallet

--> Test Generate Public/Private Key Print out fmt.Print
  result: {
  "ID": "pki1",
```

전에 작성을 해 놓았던 javascript-application을 통해 작동을 확인하고 이를 통해 체인코드의 구동을 확인할 수 있음을 보여주기로 함.


```
  "_rev": "1-285bbab0b81e7d719e52323e004a15a2",
  "ID": "pki1",
  "privateKey": "-----BEGIN RSA PRIVATE KEY-----\nMIIEpAIBAAKCAQEAtJMi0dmuk/a+cv/WcFMg4oyilZLJ
dKTZrNc\nRI1+9s/kYTT1p41pINmvdiKb44yZBIN097nUHYngA5z++/T7KzPM3ZqI/uww6fom\n0IdDFw4QeWLXibC7J/H
k+yQF74+RamMbs1m3UEtglJEYYZFnyob\nhGNaubbDH2Lgm4IEg8bUu46eK6mbt+iFm4PHuQIDAQABAoIBAG/X8mkn+wmK
o4G7z2rJaarW8nhDdBEh99O5XiX6sZUdZ3IqY0s7Et6ixLAxvwz5dhylU\ngs4tO0RQcX/LbeOBCVDo9KsNXG0Hs6sq570
l8yxjP/5Ty2QJksL\n7CuwCUak2UvvEYbohEbmPAGm2fgIi8UocKqVgNeRb3mHGU42saK/61xIN7Z3P8kT\nrXEbmUECgY
n+EEJJo4taTxpblAIUWtTMjVWD22H5iqCHd8UD8tC\nsxylwhgkFIjExyIHFaWl2vnp/NwUokLSZM3BAGY82DSyODW26Xw
F\nNvjN1Lu/t6VjJy/cd9PIfLRDH8uEXP1KfpmnB2zK6OWsaJ8jtPeI73OlyfAmekPl\nbuUO9aBnUaWMFNcfOj2HzxwvC
N42sBmBQvt6UCaMFlcCqQTQS6E\ngjA3WPBlQmv2KMHBMbpuWXgJXMuBFzmzKXApOCSmUE85/qioDoJhCSzVgbNlY35t\n
jvLHjwRLU3KV1+OtL8fHNztuphsn/sDSwM0d1QaK+alp6ugB2J1\n8HFwBM4e0h9mm2+MCSw+aoTUyfJ3HrCzq2QhGnKq9
iCFQOx2YlH\nqR7PW8JJslNdJYyw4gBu4u9Ec+rrd25K11o/B38Lpg3lcXYgHR9QtDB4lZ8d5trS\nU9zpGfNPCeQrZ1nH
  "publicKey": "-----BEGIN RSA PUBLIC KEY-----\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAtJ
os21g\npZw6RcDksYbkwZXE9Cm09dKTZrNcRI1+9s/kYTT1p41pINmvdiKb44yZBIN097nU\nHYngA5z++/T7KzPM3ZqI/
jrSA+mcVdD0JT38UNdRy9FgXL9k+yQ\nF74+RamMbs1m3UEtglJEYYZFnyobhGNaubbDH2Lgm4IEg8bUu46eK6mbt+iFm4
  "~version": "CgMBCAA="
 }
}
```


localhost:5984/_utils에 로그인을 하여 couchdb를 확인해본 결과 제대로 값이 입력된 것을 확인할 수 있음.

```go
    "github.com/hyperledger/fabric-contract-api-go/contractapi"
)

// SmartContract provides functions for managing an Asset
type SmartContract struct {
    contractapi.Contract
}

// Civil describe basic details of what makes up a simple human
type Civil struct {
    ID          string `json:"civilID"`
    Name        string `json:"civilName"`
    PhoneNumber string `json:"civilphoneNumber"`
    Address     string `json:"civilAddress"`
    Status      string `json:"civilStatus"`
}

// Shop is describe basic information of shop
type Shop struct {
    ID        string `json:"shopID"`
    Name      string `json:"shopName"`
    Telephone string `json:"shopTelephone"`
    Address   string `json:"shopAddress"`
}

// StoreVisitList are recorded when citizen visit the store visits the store.
type StoreVisit struct {
    ID        string `json:"visitID"`
    ShopID    string `json:"shopID"`
    CivilID   string `json:"civilID"`
    VisitTime string `json:"visitTime"`
}

type EncryptVisit struct {
    ID      string `json:"encvisitID"`
    EncData []byte `json:"encData"`
}

```

EncryptVisit 구조체에 암호화하여 저장을 하고 그 값을 보여줌으로 보안성을 향상할 수 있음을 보여줄 수 있다고 계획함.

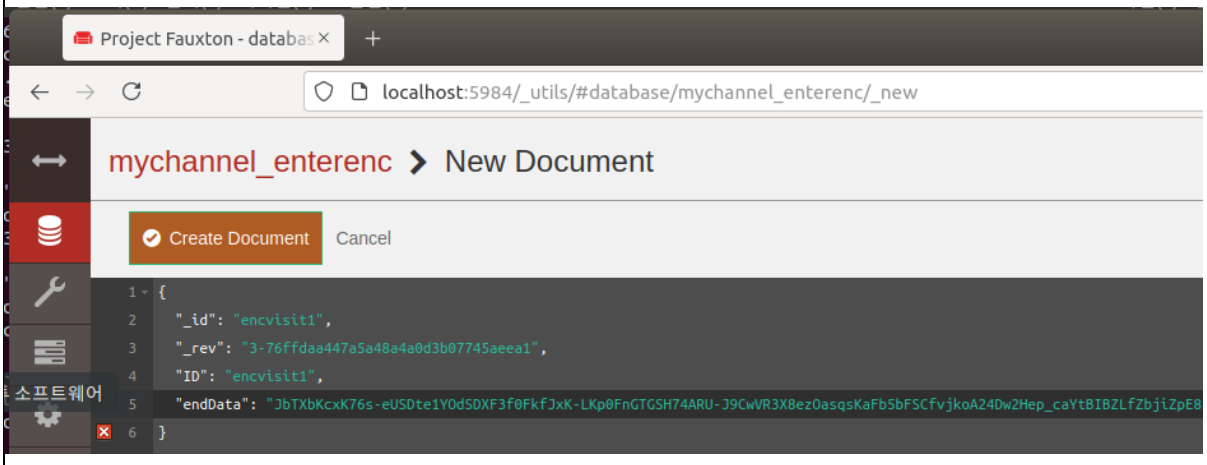
```
  "publicKey": "-----BEGIN RSA PUBLIC KEY-----\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA2zpgZg+1M
mlVd5IaXvf9\nWXeQ3wPUbCSg5vMrl5ZGItDO6DZgJonuonaIeb8LPMOcUgGpLI0Qd9C5OO54Za+T\nsXrj6Y0R2ovsfNlNj8gLT6
mcq4fRamIzguoDD2MKM5Ccoye1K+Bn23R6livWhCuz\nkJnTy7aEVHFRlTO6nfjmD9alT9+FZ7WrYgNhqX+X4Zj+Ez1wB/L1lcSSB
Eu1Qt7L\nCYNHwMTW085EwX611nOGGXF6vGx5q3ppqkd22VrdRNn0BYkDzUZ6dHnrv5J1VwO5\nVWEgaB3iiueJ4hKqwTe8aenSC/
i5Fed7FXt27DuB5WroQVXC/QuNTaFO+Lls/EGt\nKQIDAQAB\n-----END RSA PUBLIC KEY-----\n"
}

2021-06-10T03:43:45.441Z - error: [DiscoveryHandler]: compareProposalResponseResults[undefined] - rea
d/writes result sets do not match index=1
2021-06-10T03:43:45.469Z - error: [DiscoveryHandler]: compareProposalResponseResults[undefined] - rea
d/writes result sets do not match index=1
2021-06-10T03:43:45.470Z - error: [DiscoveryHandler]: compareProposalResponseResults[undefined] - rea
d/writes result sets do not match index=1
2021-06-10T03:43:45.470Z - error: [Transaction]: Error: No valid responses from any peers. Errors:
    peer=undefined, status=grpc, message=Peer endorsements do not match
Error in transaction: Error: No valid responses from any peers. Errors:
    peer=undefined, status=grpc, message=Peer endorsements do not match
Error: No valid responses from any peers. Errors:
    peer=undefined, status=grpc, message=Peer endorsements do not match
    at newEndorsementError (/home/h2kim/fabric-samples/e2-enter-encrsa/application-javascript/node_mo
dules/fabric-network/lib/transaction.js:49:12)
    at getResponsePayload (/home/h2kim/fabric-samples/e2-enter-encrsa/application-javascript/node_mod
ules/fabric-network/lib/transaction.js:17:23)
    at Transaction.submit (/home/h2kim/fabric-samples/e2-enter-encrsa/application-javascript/node_mod
ules/fabric-network/lib/transaction.js:212:28)
    at process._tickCallback (internal/process/next_tick.js:68:7)
```

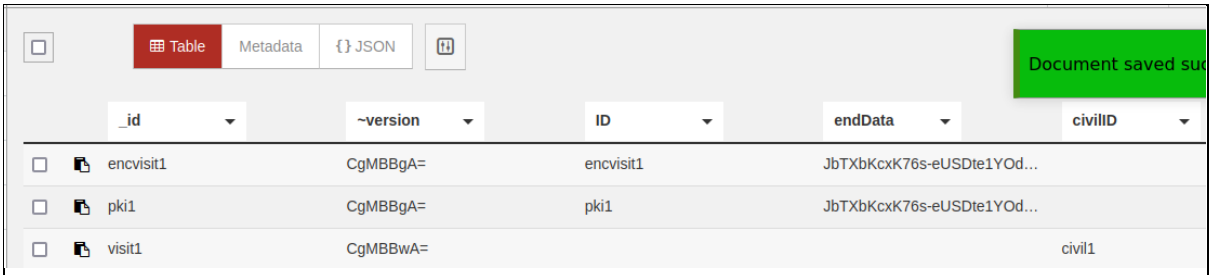
공개키를 이용한 암호화를 작동시키면 그 값이 데이터베이스에 들어갈 수 없다는 에러를 발생시키고 정지하는 것을 확인함. 이후 에러를 발생시킨 자바스크립트의 코드를 읽고 확인해본 결과 정해지지 않은 형태를 저장하려 했다는 정도로 에러를 발생시킨다는 수준정도로만 표현되어 있었음.

```
chaincode/e2entervisit.go:52:31: cannot use encodedData (type string) as type []byte in argument to c
tx.GetStub().PutState
"
Chaincode installation on peer0.org1 has failed
Deploying chaincode failed
```

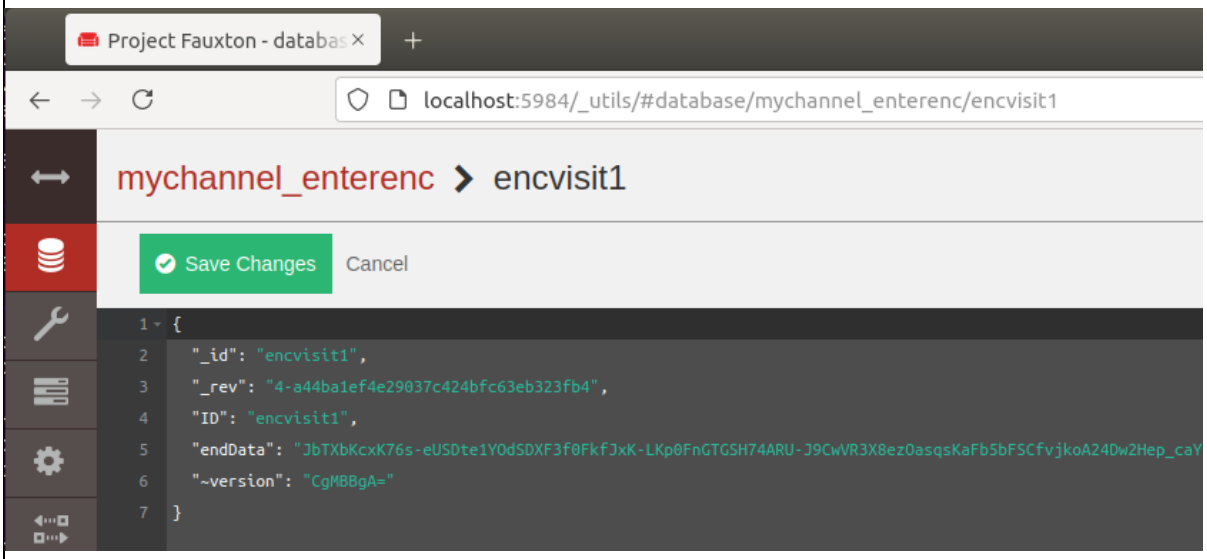
데이터 타입에 따라서 저장이 되지 않는 것인가 싶어서 변형을 해보았지만 자료형태를 수정해보고 저장을 해보았지만 변경할 때에 받아들이는 값이 정확하게 들어가지 않는 상황이 발생함.



암호화를 지나고 난 값을 couchdb에 암호화된 값을 데이터베이스에서 저장되지 않고 롤백하는 것은 아닌가 싶어서 couchdb 대시보드에서 도큐먼트에 직접 입력해서 저장하기로 계획함



도큐먼트 저장을 해보니 암호화된 값이 생성되는 것을 확인함.



endData 입력한 값이 저장되어 있음을 확인할 수 있음. 데이터베이스에서 값이 들어가는 것을 제한하는 것은 아니라는 판단을 할 수 있었음.

```
log.Printf("visitJSON: %v\n", string(visitJSON))
//cipherText, err := rsa.EncryptOAEP(sha256.New(), rand.Reader, pubKey, visi
//encodedData := base64.URLEncoding.EncodeToString(cipherText)
//log.Printf("encodedData: %v\n", encodedData)
encID := "enc" + visitID
encVisit := EncryptVisit{
        ID:      encID,
        EncData: "Test",
}
encVisitJSON, err := json.Marshal(encVisit)
log.Printf("encVisitJSON: %v\n", string(encVisitJSON))
return ctx.GetStub().PutState(encID, encVisitJSON)
```

값을 암호화된 값이 아니라 테스트 값을 넣어서 제대로 체인코드가 작동하는 것을 확인하기로 함.

```
  "doc": {
    "_id": "encvisit",
    "_rev": "1-387fc5f1f357b0ac366e68b57afff5d2",
    "ID": "encvisit",
    "encData": "Test",
    "~version": "CgMBBwA="
  }
}
```


테스트 값을 저장하는 것을 확인할 수 있었음. 값을 저장하는 형태의 문제인지 아니면 다른 체인코드 작동에서 문제인지 확인하기로 함.

```
31          cipherText, err := rsa.EncryptOAEP(sha256.New(), rand.Reader, pubKey, visitJ
32          encodedData := base64.URLEncoding.EncodeToString(cipherText)
33          log.Printf("encodedData: %v\n", encodedData)
34          encID := "enc" + visitID
35          encVisit := EncryptVisit{
36                  ID:      encID,
37                  EncData: "Test",
38          }
39          encVisitJSON, err := json.Marshal(encVisit)
40          log.Printf("encVisitJSON: %v\n", string(encVisitJSON))
41          return ctx.GetStub().PutState(encID, encVisitJSON)
42  }
```

위의 주석처리 되어 있는 부분을 다시 작동되게 하여 함수가 작동을 하는 것에 따라서 데이터베이스에 값이 추가되지 않을 수도 있겠다는 생각이 들어서 주석을 삭제하고 체인코드를 실행함.


```
[
  "_id": "encvisit",
  "_rev": "1-387fc5f1f357b0ac366e68b57afff5d2",
  "ID": "encvisit",
  "encData": "Test",
  "~version": "CgMBBwA="
}
```


주석 처리했던 함수를 작동하더라도 값이 들어가는 것을 확인할 수 있었음.

```
33          log.Printf("encodedData: %v\n", encodedData)
34          encID := "enc" + visitID
35          encVisit := EncryptVisit{
36                  ID:      encID,
37                  EncData: "0VJrpYQFy53qQgwrgx6OK_s89yzjFIZfsYedEXTeq-2flzkaHABlyhhgqx
38          }
39          encVisitJSON, err := json.Marshal(encVisit)
40          log.Printf("encVisitJSON: %v\n", string(encVisitJSON))
41          return ctx.GetStub().PutState(encID, encVisitJSON)
```

암호화해서 생성된 값을 받아들이지 않는 형태가 될 수도 있다는 생각이 들어서 직접 대입하여 생성을 하여 값 추가여부를 확인하기로 함.


```
{
  "_id": "encvisit",
  "_rev": "1-457ba668195627459ecfe75a9edb1143",
  "ID": "encvisit",
  "encData": "0VJrpYQFy53qQgwrgx6OK_s89yzjFIZfsYedEXTeq-2flzkaHABlyhhgqxloy4ftW6yRFqxwBFUDoMeKXGRKD_NE6UbOS
  "~version": "CgMBBwA="
}
```


암호화를 한 값을 제대로 저장하고 있음을 확인할 수 있었음. 입력 값을 가지고 삽입 제한 판단을 하지 않는다는 것을 확인할 수 있었음.

```
30          log.Printf("visitJSON: %v\n", string(visitJSON))
31          cipherText, err := rsa.EncryptOAEP(sha256.New(), rand.Reader, publ
32          encodedData := base64.URLEncoding.EncodeToString(cipherText)
33          log.Printf("encodedData: %v\n", encodedData)
34          encID := "enc" + visitID
35          encVisit := EncryptVisit{
36                  ID:      encID,
37                  EncData: string(cipherText),
38          }
39          encVisitJSON, err := json.Marshal(encVisit)
40          log.Printf("encVisitJSON: %v\n", string(encVisitJSON))
41          return ctx.GetStub().PutState(encID, encVisitJSON)
42  }
```

데이터 값을 공개키로 암호화한 이후 문자열로 인코딩 이후에 데이터베이스에 저장이 되는 것을 테스트하기로 함.

```
2021-06-11T04:39:36.654Z - error: [DiscoveryHandler]: compareProposalResponseResults[undefined] - rea
d/writes result sets do not match index=1
2021-06-11T04:39:36.655Z - error: [Transaction]: Error: No valid responses from any peers. Errors:
    peer=undefined, status=grpc, message=Peer endorsements do not match
Error in transaction: Error: No valid responses from any peers. Errors:
    peer=undefined, status=grpc, message=Peer endorsements do not match
Error: No valid responses from any peers. Errors:
    peer=undefined, status=grpc, message=Peer endorsements do not match
    at newEndorsementError (/home/h2kim/fabric-samples/e2-enter-enc/application-javascript/node_modul
es/fabric-network/lib/transaction.js:49:12)
    at getResponsePayload (/home/h2kim/fabric-samples/e2-enter-enc/application-javascript/node_module
s/fabric-network/lib/transaction.js:17:23)
    at Transaction.submit (/home/h2kim/fabric-samples/e2-enter-enc/application-javascript/node_module
s/fabric-network/lib/transaction.js:212:28)
    at process._tickCallback (internal/process/next_tick.js:68:7)
h2kim@h2kim-VBox:~/fabric-samples/e2-enter-enc/application-javascript$
```

암호화한 이후 저장한 처음 체인코드와 같은 에러를 발생하는 것을 확인함.

```
32          log.Printf("visitJSON: %v\n", string(visitJSON))
33          cipherText, err := rsa.EncryptOAEP(sha256.New(), rand.Reader, publ
34          //encodedData := base64.URLEncoding.EncodeToString(cipherText)
35          //log.Printf("encodedData: %v\n", encodedData)
36          encID := "enc" + visitID
37          var Buf bytes.Buffer
38          Buf.Write(cipherText)
39          encodedStr := Buf.String()
40          Buf.Reset()
41          encVisit := EncryptVisit{
42                  ID:      encID,
43                  EncData: encodedStr,
44          }
45          encVisitJSON, err := json.Marshal(encVisit)
46          log.Printf("encVisitJSON: %v\n", string(encVisitJSON))
47          return ctx.GetStub().PutState(encID, encVisitJSON)
```

인코딩 방식을 문자열로 바꾸는 방식이 아니라 버퍼를 이용하여 변경 이후 저장을 하는 방식으로 변경하여 체인코드를 작동하도록 함.

```
2021-06-11T04:49:08.224Z - error: [DiscoveryHandler]: compareProposalResponseResults[undefined] - read/writes result sets do not match index=1
2021-06-11T04:49:08.230Z - error: [Transaction]: Error: No valid responses from any peers. Errors:
    peer=undefined, status=grpc, message=Peer endorsements do not match
Error in transaction: Error: No valid responses from any peers. Errors:
    peer=undefined, status=grpc, message=Peer endorsements do not match
Error: No valid responses from any peers. Errors:
    peer=undefined, status=grpc, message=Peer endorsements do not match
    at newEndorsementError (/home/h2kim/fabric-samples/e2-enter-enc/application-javascript/node_modules/fabric-network/lib/transaction.js:49:12)
    at getResponsePayload (/home/h2kim/fabric-samples/e2-enter-enc/application-javascript/node_modules/fabric-network/lib/transaction.js:17:23)
    at Transaction.submit (/home/h2kim/fabric-samples/e2-enter-enc/application-javascript/node_modules/fabric-network/lib/transaction.js:212:28)
    at process._tickCallback (internal/process/next_tick.js:68:7)
```

위와 같이 동일한 에러를 발생하는 것을 확인함.

```
2021/06/11 05:54:53 encodedStr: uf[illegible](PR[illegible]J[illegible]:햪[illegible]SXb[illegible]R[illegible](VOW[illegible]5L[illegible]z[illegible]-[illegible]#[illegible]A[illegible]C[illegible]eC[illegible]mf[illegible]E:[illegible]W[illegible]R[illegible]][illegible]9[illegible]@[illegible]n[illegible]b[illegible]Nw[illegible]K[illegible]$[illegible].[illegible]Vot[illegible]2D[illegible]S[illegible]P^T[illegible]E[illegible]
[illegible]e0m[illegible]|[illegible]7:J^[illegible]~\[illegible]G/[illegible]
                                MX[illegible]l%[illegible]O[illegible][illegible]b>m[illegible]e[illegible]6[illegible]#$(l[illegible]IZ[illegible]^
2021/06/11 05:54:53 encVisitJSON: {"ID":"encvisit1","encData":"uf\u0010\ufffd\ufffd\ufffd\ufffd\ufffd\u003e\ufffd\ufffd\u001a\u0013m(PR\ufffd\ufffd\ufffdJ\u0018i\u0017\ufffd:햪\ufffd\ufffd\u0014\ufffd\u001f\ufffd\u0014\ufffdSXb\ufffd\u0005R\ufffd\ufffd\u0011f\ufffd\ufffd(VOW\ufffd\ufffd5L\ufffd\ufffdz\ufffd\ufffd\u00184-\ufffd\ufffd\ufffd\u0017\ufffd#\ufffd\u003eA\ufffd\ufffdC\ufffd\u001b\u001b\ufffd\ufffd\ufffd\ufffdeC\ufffd\ufffd\ufffd\u001a\ufffd\ufffdmf\ufffd\u0026\ufffd\ufffdE:\ufffd!\ufffd\ufffdW\u0007\ufffd\ufffd\ufffdR\ufffd\ufffd\ufffd]g\ufffdi\ufffd\ufffd9\u0007\u001cv\ufffd@\ufffd\ufffd\ufffdn\ufffdb\ufffd\u0002\ufffdNw\u0013\ufffd\ufffdK\ufffd$\ufffd\u0000\ufffd\ufffd\u0001\ufffd.\u0017\ufffd\ufffdVot\u0015\ufffd2D\ufffdS\u001b\u000f\u001a\ufffd\u0002P^T\ufffd\ufffd\u000c\ufffdE\ufffdl\ufffd\ufffd\ufffde0m\ufffd|\u001c\u001f\ufffd\u0008B\u0011\ufffd7:J^\ufffd]\ufffd~\\\ufffd\ufffd\ufffdG/\ufffd\ufffd\ufffd\u000cMX\ufffd\ufffdl\u0005%\ufffd\u001e\ufffdO\u0003q\ufffd\ufffd\u001c\ufffd[illegible]|b\u003em\ufffd\ufffd]e\ufffd\ufffd6\ufffd\ufffdo\ufffd#$(l\ufffdIZ\ufffd^"}
2021/06/11 05:54:53 result <nil>
```

$docker ps 명령어를 실행하면 버퍼를 통해 출력되는 값을 확인해본 결과 인코딩까지는 작동하였으나 데이터베이스에 입력되는 순간 피어에서 에러를 발생한 것으로 판단됨.

```go
// StoreVisitList are recorded when citizen visit the store visits the st

func (s *SmartContract) CreateEncVisit(ctx contractapi.TransactionContext
        //pki, err := s.ReadPKI(ctx, pkiid)
        //pubKey, err := ParseRsaPublicKeyFromPemStr(pki.PublicKey)
        visit, err := s.ReadVisit(ctx, visitID)
        visitJSON, err := json.Marshal(visit)
        if err != nil {
                return err
        }
        log.Printf("visitJSON: %v\n", string(visitJSON))
        //cipherText, err := rsa.EncryptOAEP(sha256.New(), rand.Reader, pu
        encodedData := base64.StdEncoding.EncodeToString(visitJSON)
        log.Printf("encodedData: %v\n", encodedData)
        //var Buf bytes.Buffer
        //Buf.Write(cipherText)
        //encodedStr := Buf.String()
        //Buf.Reset()
        log.Printf("encodedStr: %v\n", encodedData)
        encID := "enc" + visitID
        encVisit := EncryptVisit{
                ID:      encID,
                EncData: encodedData,
        }
        encVisitJSON, err := json.Marshal(encVisit)
        log.Printf("encVisitJSON: %v\n", string(encVisitJSON))
        result := ctx.GetStub().PutState(encID, encVisitJSON)
        log.Printf("result %v", result)
```

인코딩을 표준 인코더를 통해서 저장을 하는 방식으로 변경하여 암호화를 하는 과정을 실행하지 않고 진행하면 데이터베이스에 저장되지 않을까 싶어서 변경하여 테스트를 진행함.


```
"doc": {
  "_id": "encvisit1",
  "_rev": "1-a2a6dd89d25b49cba2a2d925f4a6f42d",
  "ID": "encvisit1",
  "encData": "bnVsbA==",
  "~version": "CgMBBwA="
 }
}
```


암호화를 하지 않고 인코딩만을 통과한 값이 데이터베이스에 입력되는 것을 확

인할 수 있었음.

```
22  func (s *SmartContract) CreateEncVisit(ctx contractapi.TransactionContext
23          pki, err := s.ReadPKI(ctx, pkiid)
24          pubKey, err := ParseRsaPublicKeyFromPemStr(pki.PublicKey)
25          visit, err := s.ReadVisit(ctx, visitID)
26          visitJSON, err := json.Marshal(visit)
27          if err != nil {
28                  return err
29          }
30          log.Printf("visitJSON: %v\n", string(visitJSON))
31          cipherText, err := rsa.EncryptOAEP(sha256.New(), rand.Reader, pub
32          encodedData := base64.StdEncoding.EncodeToString(cipherText)
33          log.Printf("encodedData: %v\n", encodedData)
34          //var Buf bytes.Buffer
35          //Buf.Write(cipherText)
36          //encodedStr := Buf.String()
37          //Buf.Reset()
38          log.Printf("encodedStr: %v\n", encodedData)
39          encID := "enc" + visitID
40          encVisit := EncryptVisit{
41                  ID:      encID,
42                  EncData: encodedData,
43          }
44          encVisitJSON, err := json.Marshal(encVisit)
45          log.Printf("encVisitJSON: %v\n", string(encVisitJSON))
46          result := ctx.GetStub().PutState(encID, encVisitJSON)
47          log.Printf("result %v", result)
```

인코딩을 한 값은 제대로 들어가는 것은 확인하였기 때문에 암호화를 통해서 저장을 하면 그 값이 들어가지 않을까 싶어서 다시 공개키를 통한 암호화를 하고 다시 그 값을 인코딩하여 데이터베이스에 저장할 수 있는지를 테스트를 함.

```
--> Print pki values Public/Private Key ~~
pki not exists so create PKI
2021-06-11T06:42:03.842Z - error: [DiscoveryHandler]: compareProposalResponseResults[undefined] - rea
d/writes result sets do not match index=1
2021-06-11T06:42:03.842Z - error: [Transaction]: Error: No valid responses from any peers. Errors:
    peer=undefined, status=grpc, message=Peer endorsements do not match
Error in transaction: Error: No valid responses from any peers. Errors:
    peer=undefined, status=grpc, message=Peer endorsements do not match
Error: No valid responses from any peers. Errors:
    peer=undefined, status=grpc, message=Peer endorsements do not match
    at newEndorsementError (/home/h2kim/fabric-samples/e2-enter-enc/application-javascript/node_modul
es/fabric-network/lib/transaction.js:49:12)
    at getResponsePayload (/home/h2kim/fabric-samples/e2-enter-enc/application-javascript/node_module
s/fabric-network/lib/transaction.js:17:23)
    at Transaction.submit (/home/h2kim/fabric-samples/e2-enter-enc/application-javascript/node_module
s/fabric-network/lib/transaction.js:212:28)
    at process._tickCallback (internal/process/next_tick.js:68:7)
```

동일하게 트랜잭션을 하는 과정 속에서 에러가 발생하는 것을 확인할 수 있었음.

```
22  func (s *SmartContract) CreateEncVisit(ctx contractapi.TransactionContext
23          pki, err := s.ReadPKI(ctx, pkiid)
24          pubKey, err := ParseRsaPublicKeyFromPemStr(pki.PublicKey)
25          visit, err := s.ReadVisit(ctx, visitID)
26          visitJSON, err := json.Marshal(visit)
27          if err != nil {
28                  return err
29          }
30          log.Printf("visitJSON: %v\n", string(visitJSON))
31          //cipherText, err := rsa.EncryptOAEP(sha256.New(), rand.Reader, pu
32          cipherText, err := rsa.EncryptPKCS1v15(rand.Reader, pubKey, visit.
33          encodedData := base64.StdEncoding.EncodeToString(cipherText)
34          log.Printf("encodedData: %v\n", encodedData)
35          //var Buf bytes.Buffer
36          //Buf.Write(cipherText)
37          //encodedStr := Buf.String()
38          //Buf.Reset()
39          log.Printf("encodedStr: %v\n", encodedData)
40          encID := "enc" + visitID
41          encVisit := EncryptVisit{
42                  ID:      encID,
43                  EncData: encodedData,
44          }
45          encVisitJSON, err := json.Marshal(encVisit)
46          log.Printf("encVisitJSON: %v\n", string(encVisitJSON))
47          result := ctx.GetStub().PutState(encID, encVisitJSON)
```

암호화하는 함수의 문제일 수도 있겠다는 생각이 들어서 EncryptOAEP 함수를 EncryptPKCS1v15함수로 변경하여 암호화하는 과정을 거쳐서 제대로 값이 입력되는 것을 확인하기로 함.

```
--> Print pki values Public/Private Key ~~
pki not exists so create PKI
2021-06-11T06:53:05.473Z - error: [DiscoveryHandler]: compareProposalResponseResults[undefined] - rea
d/writes result sets do not match index=1
2021-06-11T06:53:05.473Z - error: [Transaction]: Error: No valid responses from any peers. Errors:
    peer=undefined, status=grpc, message=Peer endorsements do not match
Error in transaction: Error: No valid responses from any peers. Errors:
    peer=undefined, status=grpc, message=Peer endorsements do not match
Error: No valid responses from any peers. Errors:
    peer=undefined, status=grpc, message=Peer endorsements do not match
    at newEndorsementError (/home/h2kim/fabric-samples/e2-enter-enc/application-javascript/node_modul
es/fabric-network/lib/transaction.js:49:12)
    at getResponsePayload (/home/h2kim/fabric-samples/e2-enter-enc/application-javascript/node_module
s/fabric-network/lib/transaction.js:17:23)
    at Transaction.submit (/home/h2kim/fabric-samples/e2-enter-enc/application-javascript/node_module
s/fabric-network/lib/transaction.js:212:28)
    at process._tickCallback (internal/process/next_tick.js:68:7)
```

암호화하는 함수의 문제가 아님을 알 수 있었음. 변경하는 함수의 문제가 아니라면 다른 문제가 무엇이 있을지 고민을 했던 중에 체인코드를 한 곳에서 처리를 하여 코드 실행 중에 처리하는 과정에 문제가 생길 수도 있다는 생각을 함. 코드

를 분리하여 외부의 함수를 불러와서 암호화를 처리하고 코드를 진행하는 과정을 하기로 함.

```
154  func RSA_OAEP_Encrypt(secretMessage string, key rsa.PublicKey) string {
155          label := []byte("OAEP Encrypted")
156          rng := rand.Reader
157          ciphertext, err := rsa.EncryptOAEP(sha256.New(), rng, &key, []byt
158          CheckError(err)
159          return base64.StdEncoding.EncodeToString(ciphertext)
160  }
161
```

외부에 파일에서 암호화하는 과정을 분리하여 다른 체인코드에서 암호화하는 함수를 불러올 수 있게 함.

```
34      encodedData := RSA_OAEP_Encrypt("test", *pubKey)
35      //utf8encStr := []rune(encodedData)
36      //log.Printf("encodedData: %v\n", encodedData)
37      //var Buf bytes.Buffer
38      //Buf.Write(cipherText)
39      //encodedStr := Buf.String()
40      //Buf.Reset()
41      //log.Printf("encodedStr: %v\n", encodedData)
42      encID := "enc" + visitID
43      encVisit := EncryptVisit{
44              ID:      encID,
```

기존에 바로 함수를 불러오는 것이 아니라 외부에 있는 함수를 통해서 불러옴으로 체인코드에서 바로 불러오는 것이 아니라 한번 외부에 있는 과정을 거치게 함.

```
--> Print pki values Public/Private Key ~~
pki not exists so create PKI
2021-06-11T06:42:03.842Z - error: [DiscoveryHandler]: compareProposalResponseResults[undefined] - rea
d/writes result sets do not match index=1
2021-06-11T06:42:03.842Z - error: [Transaction]: Error: No valid responses from any peers. Errors:
    peer=undefined, status=grpc, message=Peer endorsements do not match
Error in transaction: Error: No valid responses from any peers. Errors:
    peer=undefined, status=grpc, message=Peer endorsements do not match
Error: No valid responses from any peers. Errors:
    peer=undefined, status=grpc, message=Peer endorsements do not match
    at newEndorsementError (/home/h2kim/fabric-samples/e2-enter-enc/application-javascript/node_modul
es/fabric-network/lib/transaction.js:49:12)
    at getResponsePayload (/home/h2kim/fabric-samples/e2-enter-enc/application-javascript/node_module
s/fabric-network/lib/transaction.js:17:23)
    at Transaction.submit (/home/h2kim/fabric-samples/e2-enter-enc/application-javascript/node_module
s/fabric-network/lib/transaction.js:212:28)
    at process._tickCallback (internal/process/next_tick.js:68:7)
```

차이는 없이 에러를 발생시키는 것으로 여태까지 생각했던 방식에서 저장을 할 수 있는 과정을 거친 것을 기존에 있는 마셜링 된 값을 인코딩하여 저장하는 과정 밖에 없었음. 공개키 암호화된 형식이 아니라 다른 과정의 암호화 과정을 선택하기로 함.

```
35          block, err := aes.NewCipher([]byte(pki.PublicKey))
36          CheckError(err)
37          cipherText := AES_Encrypt(block, visitJSON)
38          //block.Encrypt(cipherText, visitJSON)
39          log.Printf("cipherText: %v\n", cipherText)
40          //encodedData := RSA_OAEP_Encrypt("test", *pubKey)
41          //utf8encStr := []rune(encodedData)
42          //log.Printf("encodedData: %v\n", encodedData)
43          //var Buf bytes.Buffer
44          //Buf.Write(cipherText)
45          //encodedStr := Buf.String()
46          //Buf.Reset()
47          //log.Printf("encodedStr: %v\n", encodedData)
48          encID := "enc" + visitID
49          encVisit := EncryptVisit{
50                  ID:      encID,
51                  EncData: cipherText,
52          }
```

공개키 암호화 방식이 아니라 대칭키 암호화 방식을 통해서 암호화를 하고 암호화된 데이터를 데이터베이스에 저장하기로 함.

```
165 func AES_Encrypt(b cipher.Block, plaintext []byte) string {
166         if mod := len(plaintext) % aes.BlockSize; mod != 0 {
167                 padding := make([]byte, aes.BlockSize-mod)
168                 plaintext = append(plaintext, padding...)
169         }
170         ciphertext := make([]byte, aes.BlockSize+len(plaintext))
171         iv := ciphertext[:aes.BlockSize]
172         if _, err := io.ReadFull(rand.Reader, iv); err != nil {
173                 fmt.Println(err)
174                 return ""
175         }
176         mode := cipher.NewCBCEncrypter(b, iv)
177         mode.CryptBlocks(ciphertext[aes.BlockSize:], plaintext)
178         return base64.StdEncoding.EncodeToString(ciphertext)
179 }
```

대칭키로 암호화하는 함수는 외부에서 선언하여 부족한 값은 패딩값을 넣어서 암호화하는 값에 나눠질 수 있도록 하고 암호화한 값을 문자열 값으로 리턴하게 하는 함수를 작성함.

```
panic: runtime error: invalid memory address or nil pointer dereference
[signal SIGSEGV: segmentation violation code=0x1 addr=0x18 pc=0x514bc6]

goroutine 55 [running]:
crypto/cipher.NewCBCEncrypter(0x0, 0x0, 0xc0000269a0, 0x10, 0x70, 0x10, 0x10)
        /usr/local/go/src/crypto/cipher/cbc.go:46 +0x26
github.com/hyperledger/fabric-samples/asset-transfer-basic/chaincode-go/chaincode.AES_Encrypt(0x0, 0x
0, 0xc0005d2780, 0x60, 0x60, 0xae4c60, 0xc00038a7a8)
        /chaincode/input/src/chaincode/e2enterenc.go:176 +0x19d
github.com/hyperledger/fabric-samples/asset-transfer-basic/chaincode-go/chaincode.(*SmartContract).Cr
eateEncVisit(0xc000255950, 0x7f58211ab1c8, 0xc0003a8300, 0xc0004cdc48, 0x4, 0xc0004cdc80, 0x6, 0x0, 0
x0)
```

암호화를 하지 못하고 메모리 에러를 발생시킨 것을 확인할 수 있음.

```
35          block, err := aes.NewCipher([]byte(pki.PublicKey[:15]))
36          CheckError(err)
37          cipherText := AES_Encrypt(block, visitJSON)
38          //block.Encrypt(cipherText, visitJSON)
39          log.Printf("cipherText: %v\n", cipherText)
```

암호화할 때 넘겨주는 암호화 대칭키의 크기를 줄여서 대칭키 암호화하는 함수에 전달함.

```
2021-06-12 04:18:58.641 UTC [endorser] SimulateProposal -> ERRO 088 failed to invoke chaincode entere
nc, error: chaincode stream terminated
github.com/hyperledger/fabric/core/chaincode.(*Handler).Execute
        /go/src/github.com/hyperledger/fabric/core/chaincode/handler.go:1184
github.com/hyperledger/fabric/core/chaincode.(*ChaincodeSupport).execute
        /go/src/github.com/hyperledger/fabric/core/chaincode/chaincode_support.go:272
github.com/hyperledger/fabric/core/chaincode.(*ChaincodeSupport).Invoke
        /go/src/github.com/hyperledger/fabric/core/chaincode/chaincode_support.go:202
github.com/hyperledger/fabric/core/chaincode.(*ChaincodeSupport).Execute
        /go/src/github.com/hyperledger/fabric/core/chaincode/chaincode_support.go:155
github.com/hyperledger/fabric/core/endorser.(*SupportImpl).Execute
        /go/src/github.com/hyperledger/fabric/core/endorser/support.go:126
github.com/hyperledger/fabric/core/endorser.(*Endorser).callChaincode
        /go/src/github.com/hyperledger/fabric/core/endorser/endorser.go:119
github.com/hyperledger/fabric/core/endorser.(*Endorser).SimulateProposal
        /go/src/github.com/hyperledger/fabric/core/endorser/endorser.go:187
github.com/hyperledger/fabric/core/endorser.(*Endorser).ProcessProposalSuccessfullyOrError
```

체인코드가 정상적으로 작동하지 않고 에러를 발생하고 정지되는 것을 확인할 수 있었음.

```
2021/06/12 04:18:54 result <nil>
2021/06/12 04:18:58 visitJSON: {"visitID":"visit1","shopID":"shop1","civilID":"civil1","visitTime":"2
021-05-21 17:45"}
0x906da0
panic: runtime error: invalid memory address or nil pointer dereference
[signal SIGSEGV: segmentation violation code=0x1 addr=0x18 pc=0x514bc6]

goroutine 55 [running]:
crypto/cipher.NewCBCEncrypter(0x0, 0x0, 0xc000026a80, 0x10, 0x70, 0x10, 0x10)
        /usr/local/go/src/crypto/cipher/cbc.go:46 +0x26
github.com/hyperledger/fabric-samples/asset-transfer-basic/chaincode-go/chaincode.AES_Encrypt(0x0, 0x
0, 0xc0001da720, 0x60, 0x60, 0xae4c60, 0xdfee18)
        /chaincode/input/src/chaincode/e2enterenc.go:176 +0x19d
github.com/hyperledger/fabric-samples/asset-transfer-basic/chaincode-go/chaincode.(*SmartContract).Cr
eateEncVisit(0xc0002578c0, 0x7fef2ae76758, 0xc0002e3ee0, 0xc0003bd4d8, 0x4, 0xc0003bd510, 0x6, 0x0, 0
```

작동은 하고나서 메모리 에러를 발생하며 정지한 것을 로그를 통해 확인할 수 있었음.

```
48 func main() {
49     key := `-----BEGIN RSA PUBLIC KEY-----\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvMt3b6a/G
   Fi91S6M5EYo\nu43aiOM/TdB39YiM30531Ktp65DwTuiEen7570aO63+s80X1eI6AmC8MwucYGp/L\n95XjH88QJadlXgUGwm
   0EwD8rLeJBNERycdHJ2pC4x0Ya47MLkVxGmel8XC6lGPXE\nz7FhqiP9QGbcB5/XoK2Nkjl7FDmth+O42+q+F+bvpDNtngOC8
   nnKbTkW9PqAuMu1\ndILYwaYnSWRtdF61SP93lv4tvhtfBD7QAbFthO+mUNntYG6SCHAkj86HffrcaeUC\nj+No2ITF08I4ve
   P1k5q6cCHBNSXUc18Nzy/ssgQLOv5rBDrmCmwXg9wPMqQx8q7y\nIQIDAQAB\n-----END RSA PUBLIC KEY-----\n` //
   16바이트
50
51     s := `alpha bravo charlie delta echo foxtrot golf hotel india juliett kilo lima mike november
    oscar papa quebec romeo sierra tango uniform victor whiskey x-ray yankee zulu`
52
53     block, err := aes.NewCipher([]byte(key[:16])) // AES 대칭키 암호화 블록 생성
54     if err != nil {
55         fmt.Println(err)
56         return
57     }
58
59     ciphertext := encrypt(block, []byte(s)) // 평문을 AES 알고리즘으로 암호화
60     fmt.Printf("%x\n", ciphertext)
61
62     plaintext := decrypt(block, ciphertext) // AES 알고리즘 암호문을 평문으로 복호화
63     fmt.Println(string(plaintext))
64
65 }
```

테스트용으로 소스코드를 작성해서 동작을 확인함.

```
h2kim@h2kim-VBox:~/test$ go build aes.go
h2kim@h2kim-VBox:~/test$ ./aes
99d0c93ff5a0c61e73a17b69f84a7f0bb94bdc6336f30ddcc9353b456c5f3093fdd1ccd58039d4cfbccaaf7e784451477cd6d
bd4df4d8784604489b126a93f7b30689e05778960639b8f46a9c6d4c5dfe9a1b1d7a87b3ca2db38f566eccc9cdb8a47100590
4fe1043eb7f3b1c3f4a4e226000cb83d9a20fde478c033e1539158a516ea90da60488b8118909f0060865b930340405a73e28
925978172b977565aee485387755296d04f7fecac497863f118405af58ce062d552313e13b078c9dd
alpha bravo charlie delta echo foxtrot golf hotel india juliett kilo lima mike november oscar papa qu
ebec romeo sierra tango uniform victor whiskey x-ray yankee zulu
h2kim@h2kim-VBox:~/test$
```

코드를 빌드하고 동작을 확인하면 암호화되는 것을 확인하고 다시 복호화 한 값을 출력하는 것을 확인할 수 있었음.

```
35          //log.Printf("encodedData: %v\n", stdVisit)
36          block, err := aes.NewCipher([]byte(pki.PublicKey[:16]))
37          CheckError(err)
38          cipherText := AES_CBC_Encrypt(block, visitJSON)
39          //block.Encrypt(cipherText, visitJSON)
40          log.Printf("cipherText: %v\n", cipherText)
41          encData := base64.StdEncoding.EncodeToString(cipherText)
```

소스코드에서 동작한 것을 적용하여 수정을 한 이후

```
h2kim@h2kim-VBox:~/fabric-samples/test-network$ docker logs -f 4ffa498da3af
2021/06/12 04:48:32 result <nil>
2021/06/12 04:48:37 visitJSON: {"visitID":"visit1","shopID":"shop1","civilID":"civil1","visitTime":"2
021-05-21 17:45"}
2021/06/12 04:48:37 cipherText: [218 226 118 38 174 170 48 149 31 134 135 185 56 237 231 77 102 106 9
6 21 25 164 108 151 88 232 106 108 179 184 200 162 161 68 40 181 43 12 110 242 25 150 64 243 209 213
166 157 252 195 10 44 13 41 213 181 220 92 208 99 239 84 203 175 227 22 186 172 38 131 89 38 105 142
251 55 165 12 143 66 219 172 36 153 12 50 77 214 41 84 218 64 207 183 209 133 192 83 200 40 228 67 15
0 235 181 92 151 16 221 41 191 228]
2021/06/12 04:48:37 encVisitJSON: {"ID":"encvisit1","encData":"2uJ2Jq6qMJUfhoe5OO3nTWZqYBUZpGyXWOhqbL
O4yKKhRCi1Kwxu8hmWQPPR1aad/MMKLA0p1bXcXNBj71TLr+MWuqwmg1kmaY77N6UMj0LbrCSZDDJN1ilU2kDPt9GFwFPIKORDluu
1XJcQ3Sm/5A==\"}
2021/06/12 04:48:37 result <nil>
```

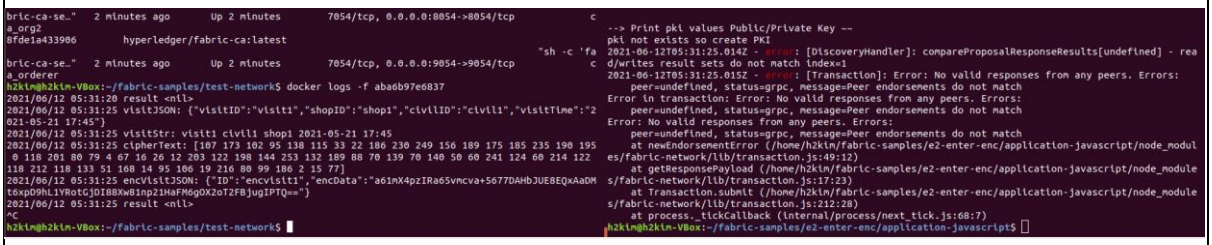
체인코드를 변경하여 작동을 확인하고 로그를 통해 대칭키 암호화를 통해 데이터가 암호화를 거친 값을 보이는 것을 확인할 수 있었음.

```
2021-06-12T04:48:37.433Z - error: [DiscoveryHandler]: compareProposalResponseResults[undefined] - rea
d/writes result sets do not match index=1
2021-06-12T04:48:37.434Z - error: [Transaction]: Error: No valid responses from any peers. Errors:
    peer=undefined, status=grpc, message=Peer endorsements do not match
Error in transaction: Error: No valid responses from any peers. Errors:
    peer=undefined, status=grpc, message=Peer endorsements do not match
Error: No valid responses from any peers. Errors:
    peer=undefined, status=grpc, message=Peer endorsements do not match
```

로그를 통해 대칭키 암호화가 동작한 것은 확인할 수 있었지만 데이터베이스에 저장되지 않았다는 것을 확인할 수 있었음.

```
24      pki, err := s.ReadPKI(ctx, pkiid)
25      //pubKey, err := ParseRsaPublicKeyFromPemStr(pki.PublicKey)
26      visit, err := s.ReadVisit(ctx, visitID)
27      visitJSON, err := json.Marshal(visit)
28      if err != nil {
29              return err
30      }
31      log.Printf("visitJSON: %v\n", string(visitJSON))
32      visitStr := visit.ID + visit.CivilID + visit.ShopID + visit.VisitTime
33      //cipherText, err := rsa.EncryptPKCS1v15(rand.Reader, pubKey, visitJSON
34      //var stdVisit []byte
35      //base64.StdEncoding.Encode(stdVisit, visitJSON)
36      //log.Printf("encodedData: %v\n", stdVisit)
37      block, err := aes.NewCipher([]byte(pki.PublicKey[:16]))
38      CheckError(err)
39      cipherText := AES_CBC_Encrypt(block, []byte(visitStr))
40      //block.Encrypt(cipherText, visitJSON)
41      log.Printf("cipherText: %v\n", cipherText)
42      encData := base64.StdEncoding.EncodeToString(cipherText)
```

데이터 형태에 문제가 있을 수도 있지 않을까 싶어서 데이터를 변경하여 적용하는 방식도 고려하여 테스트를 함.



대칭키를 이용한 암호화는 되었지만 데이터베이스에 저장되지 않는 과정은 변화가 없음을 확인할 수 있었음.

```
51              encID := "enc" + visitID
52              encVisit := EncryptVisit{
53                      ID:      encID,
54                      EncData: visitJSON,
55              }
56              encVisitJSON, err := json.Marshal(encVisit)
57              log.Printf("encVisitJSON: %v\n", string(encVisitJSON))
58              result := ctx.GetStub().PutState(encID, encVisitJSON)
59              log.Printf("result %v", result)
```

암호화하는 과정 없이 데이터베이스에서 읽어온 json 값을 다시 저장하여 실제로 작동하는지 확인하기로 함.


```
"doc": {
 "_id": "encvisit1",
 "_rev": "1-e0e95bf87cbd44bfe173269498f3d412",
 "ID": "encvisit1",
 "encData": "eyJ2aXNpdElEIjoidmlzaXQxIiwic2hvcElEIjoic2hvcDEiLCJjaXZpbElEIjoiY2l2aWwxIiwidmlzaXRUaW1lIjoiMjAyMS0wNS0yMSAxNzo0NSJ9",
 "~version": "CgMBCAA="
}
}
```


json값이 doc에 저장되는 것을 확인할 수 있었음. 체인코드 영역에서 암호화를 하게 되면 트랜잭션이 되는 것을 피어에서 거부하여 데이터베이스에 저장되는 것이 중지되는 것으로 판단됨.

```
42          encData := base64.StdEncoding.EncodeToString(visitJSON)
43          //encodedData := RSA_OAEP_Encrypt("test", *pubKey)
44          //utf8encStr := []rune(encodedData)
45          //log.Printf("encodedData: %v\n", encodedData)
46          //var Buf bytes.Buffer
47          //Buf.Write(cipherText)
48          //encodedStr := Buf.String()
49          //Buf.Reset()
50          //log.Printf("encodedStr: %v\n", encodedData)
51          encID := "enc" + visitID
52          encVisit := EncryptVisit{
53                  ID:      encID,
54                  EncData: encData,
55          }
56          encVisitJSON, err := json.Marshal(encVisit)
57          log.Printf("encVisitJSON: %v\n", string(encVisitJSON))
58          result := ctx.GetStub().PutState(encID, encVisitJSON)
59          log.Printf("result %v", result)
60          return nil
61  }
```

문자열 인코딩을 하게 되면 변화가 있을까 싶어서 체인코드를 변경하고 테스트를 해보기로 함.


```
"doc": {
 "_id": "encvisit1",
 "_rev": "1-e0e95bf87cbd44bfe173269498f3d412",
 "ID": "encvisit1",
 "encData": "eyJ2aXNpdElEIjoidmlzaXQxIiwic2hvcElEIjoic2hvcDEiLCJjaXZpbElEIjoiY2l2aWwxIiwidmlzaXRUaW1lIjoiMjAyMS0wNS0yMSAxNzo0NSJ9
 "~version": "CgMBCAA="
}
}
```


위의 값과 변화가 없다는 것을 확인함. Byte배열과 문자열 인코딩을 통한 데이터 변환은 없다는 것을 확인할 수 있었음. 체인코드 영역에서 암호화하는 것을 벗어나서 어플리케이션 영역에서 암호화를 하게 되면 데이터를 입력 가능할 것인지 테스트하기로 함.

```
172             result = await contractOrg1.evaluateTransaction('ReadVisit', "visit1")
173             console.log('\n'+result)
174             encmsg = crypto.publicEncrypt(pki.publicKey, Buffer.from('test', 'utf8') ).toString('base64');
175             result = await contractOrg1.submitTransaction('CreateEncVisit', 'encvisit1', encmsg);
176             try {
177                 //try delete pki.ID ReadPKI Error not exists
178                 console.log('--> Attempt Transaction DeletePKI ' + pki.ID);
179                 result = await contractOrg1.evaluateTransaction('ReadPKI', pki.ID);
180                 console.log('******** FAILED : expected to return an error');
181             } catch (error) {
182                 console.log(`  Successfully caught the error: \n    ${error}`);
183             }
184         } finally {
185             // Disconnect from the gateway peer when all work for this client identity is complete
```

nodejs에서 제공하는 crypto패키지를 이용하여 공개키 암호화를 한 값이 입력이 되는지 'test'라는 문자열을 통해 확인하기로 함.

```
{"visitID":"visit2","shopID":"shop2","civilID":"civil1","visitTime":"2021-05-21 17:45"}
Error in transaction: Error: error:0D0680A8:asn1 encoding routines:asn1_check_tlen:wrong tag
Error: error:0D0680A8:asn1 encoding routines:asn1_check_tlen:wrong tag
    at Object.publicEncrypt (internal/crypto/cipher.js:44:12)
    at main (/home/h2kim/fabric-samples/e2-enter-enc/application-javascript/app.js:183:33)
    at process._tickCallback (internal/process/next_tick.js:68:7)
```

암호화를 작동하게 되면 기존에 go에서 생성한 pki.publickey 값을 읽으면 제대로 인코딩이 되어 있지 않다고 하면서 에러를 발생시킴.

```
180             result = await contractOrg1.evaluateTransaction('ReadVisit', "visit1")
181             console.log('\n'+result)
182             console.log('\n'+pki.publicKey)
183             var PUBKEY = '-----BEGIN RSA PUBLIC KEY-----\n'+
184 'MIIBCgKCAQEAuErwCGWyq3nIASTKhgiHGAaURZ9rs5EBR9L1YUJh1ftYgQStOgUz\n'+
185 'ab6hyW/upgM4Z4Mu6pA+KZCHncKc4SHJXdJvYnj1L6/RRinXQp+R3MmGypoB/r1c\n'+
186 'kM1aEt75/IOm7Dlatj58f56Z21yHsJNbtf1LAD981a3kR6kaIb7Uc5bsUDwObF2r\n'+
187 '5xAenQDtaZoWgaErHWwiqzJJQebcAVVqlq2/+f1kzRVqHsasosXOD6hrDz9oC2Sv\n'+
188 'BKWVrmOPF4D+mwwaChEKAhFDvCKj5NYwprmoOXWK6t5WroYvsVo5Sa039DiCPXMs\n'+
189 'ug9MidhQLB7SpW7Bi+xeuwvObWppgGZ8FQIDAQAB\n'+
190 '-----END RSA PUBLIC KEY-----\n';
191             var encmsg = crypto.publicEncrypt(PUBKEY, Buffer.from(result, 'utf8') ).toString('base64');
192             console.log('\n'+encmsg)
193            //  .toString('base64');
194             //console.log('\n'+encmsg)
195             result = await contractOrg1.submitTransaction('CreateEncVisit', 'encvisit1', encmsg);
196             try {
197                 //try delete pki.ID ReadPKI Error not exists
198                 console.log('--> Attempt Transaction DeletePKI ' + pki.ID);
199                 result = await contractOrg1.evaluateTransaction('ReadPKI', pki.ID);
200                 console.log('******** FAILED : expected to return an error');
201             } catch (error) {
202                 console.log(`  Successfully caught the error: \n    ${error}`);
```

nodejs 공개키 암호화를 가지고 찾아낸 예제 중에서 공개키를 하드코딩하고 테스트를 한 경우가 있어서 그 값을 코드에 넣고 실행시킴.

```
  "ID": "encvisit1",
  "encData": "s+3iOXM0WZ2pZ1lawPiPK62EL60wzohy/RbLnNiCyl26erdXrgeqD+Pzrxu9jwaYSyWFufTpj7E223p0xzJIWZShcgx0X3TDOVEqFZY6vrE0
C6tsKlqSujUAoBX5n/7QZ/fjoXqnLIP5OjVgDk4zQ67h68T2ypO1ruVdqkdtKsrZN90wYPGtYXlWaJudnuEwWXILb1igejOLBYBiq+PiH7a7Dr9u/mRc9/kyKgu
dbbTDn1x+lNoUF6YfACDeSluMTqqQ2635mkGTZAVAKttTy30qNKBsMwxfUZAZA==",
  "~version": "CgMBCAA="
 }
}
```

값이 제대로 endData에 입력된 것을 볼 수 있음. nodejs에서 생성하는 공개키

값과 go에서 생성하는 공개키가 다르다는 것을 알 수 있었음. 자바와 nodejs에서 호환성에 관한 패딩을 더해서 작동하는 과정을 설명한 예제를 보고 실제로 둘 간의 데이터를 확인해본 결과 nodejs의 공개키 값이 더 길다는 것을 확인할 수 있었음.

```
161         var pkiExists = await contractOrg1.evaluateTransaction('PKIExists', "pki2");
162         if (pkiExists=='false') {
163             console.log('pki not exists so create PKI');
164             const { privateKey, publicKey } = crypto.generateKeyPairSync('rsa', {
165                 modulusLength: 4096,
166                 publicKeyEncoding: {
167                     type: 'pkcs1',
168                     format: 'pem',
169                 },
170             privateKeyEncoding: {
171                     type: 'pkcs1',
172                     format: 'pem',
173                     cipher: 'aes-256-cbc',
174                     passphrase: '',
175                 },
176             })
177             result = await contractOrg1.submitTransaction('CreatePKI', "pki2", privateKey, publicKey);
178         }
179         var result = await contractOrg1.evaluateTransaction('ReadPKI', "pki2");
180         var pki = JSON.parse(result);
181         //var keyBundle = JSON.parse(result);
182         //console.log('\n--> Get chaincode pki values Public/Private Key ~~');
183         //console.log('\n' + keyBundle.ID);
```

nodejs의 공개키를 생성하고 그 값을 체인코드를 통해 블록체인에 저장하는 과정을 실행함.

```
194         result = await contractOrg1.evaluateTransaction('ReadVisit', "visit2")
195         console.log('\n'+result)
196         //console.log('\n'+escape(pki.publicKey))
197         var encmsg = crypto.publicEncrypt(pki.publicKey, Buffer.from(result, 'utf8') ).toString('base64');
198
199         console.log('\n'+encmsg)
200        //  .toString('base64');
201         //console.log('\n'+encmsg)
202         result = await contractOrg1.submitTransaction('CreateEncVisit', 'encvisit2', encmsg);
203         try {
204             //try delete pki.ID ReadPKI Error not exists
205             console.log('--> Attempt Transaction DeletePKI ' + pki.ID);
206             result = await contractOrg1.evaluateTransaction('ReadPKI', pki.ID);
207             console.log('******** FAILED : expected to return an error');
208         } catch (error) {
209             console.log(`  Successfully caught the error: \n    ${error}`);
210         }
211     } finally {
212         // Disconnect from the gateway peer when all work for this client identity is complete
213         gatewayOrg1.disconnect();
214         gatewayOrg2.disconnect();
215     }
216 } catch (error) {
```

공개키 값을 읽어오고 그 값을 통해서 암호화를 하고 암호화한 값을 블록체인에 저장하는 과정을 실행함.

id "encvisit2"


```
{
 "id": "encvisit2",
 "key": "encvisit2",
 "value": {
  "rev": "1-e1690821749d9499ba3bbe36c3739900"
 },
 "doc": {
  "_id": "encvisit2",
  "_rev": "1-e1690821749d9499ba3bbe36c3739900",
  "ID": "encvisit2",
  "encData": "MH9/Kcf8XQc3ET0KILL/lrxPoOeAI14bh0FzoaGGZmzfIxUUX8FtLThn7Ppe0oqwpGDNSaznDcgmV9qTDHCR3uoaPIR9K3tStx3tJKIt2rLWSm52jDND9f/cBW2/3OWXXLqDN
oAxEiSZFhMzRUDS3kUIS0ZJBddeojRJEpaYqgNVXuVyYEu4o+8VivRrTRrJ1KFWaDZ/JBdWcPoTusVdN1/a9ZnfPuYjNn14nLGdnKhiG2RKrUeFuoB+Myd2rEegSn46E9nigkuJtOGNVCC15mE/
vN2OSA2KJ519/EgoeQJ+0v/KVxqDx3I7S9tcYg1CfpAYH3Y+/mdRGbDEgyGBE2U+Z85I1QIjzm2+KHLTeESPlLVq6cXdNCx9TO8gJxHpOOYTxcO6QyP0obNHjRHyi4hYVxrs84nejKxF904aLGd
aRttIA/NJEDxKx/8yq6Xab3mJBwCAL9HAr4wHRoO7k7kspVVzup6YsB2LvBiVL/zvtqxfm3i9sxw6/cxSvJpznovPTrkwI20w4gQBoJLbiBaOgpsmixAYF6BVA1F0q93ErAJaI71N1q+fgciMXG
46TjareVMNz+m+GwMQdHSU31XQfCr27E84pEKALVuNxbzwlN8/Ur371QE4KjBZnIzhLJgum/JOMc2TWpAt5GfWIF1Uq5ZoB3ESM5gYFDWC3tw=",
  "~version": "CgMBDAA="
 }
}
```


공개키 값을 읽어와서 데이터를 암호화하여 저장되는 것을 확인할 수 있었음. 기존에 연구에서 생각했던 체인코드 영역에서 암호화를 진행하는 것은 할 수 없다는 것을 알게 됨. 어플리케이션 영역에서 암호화를 하여 입력 해야 하이퍼레저 블록체인에 암호화된 데이터를 저장 가능하다는 결과를 얻을 수 있었음.

3. 차주 계획

- 블록체인과 보안 Term paper 작성 및 제출

4. 참고 자료

[1] http://pyrasis.com/book/GoForTheReallyImpatient/Unit53/03

[2] http://pyrasis.com/book/GoForTheReallyImpatient/Unit53/02

[3] http://blog.securekim.com/2018/01/nodejs-rsa.html

[4] https://www.sohamkamani.com/nodejs/rsa-encryption/

[5] https://charlie-choi.tistory.com/251

[6] https://awesometic.tistory.com/12