

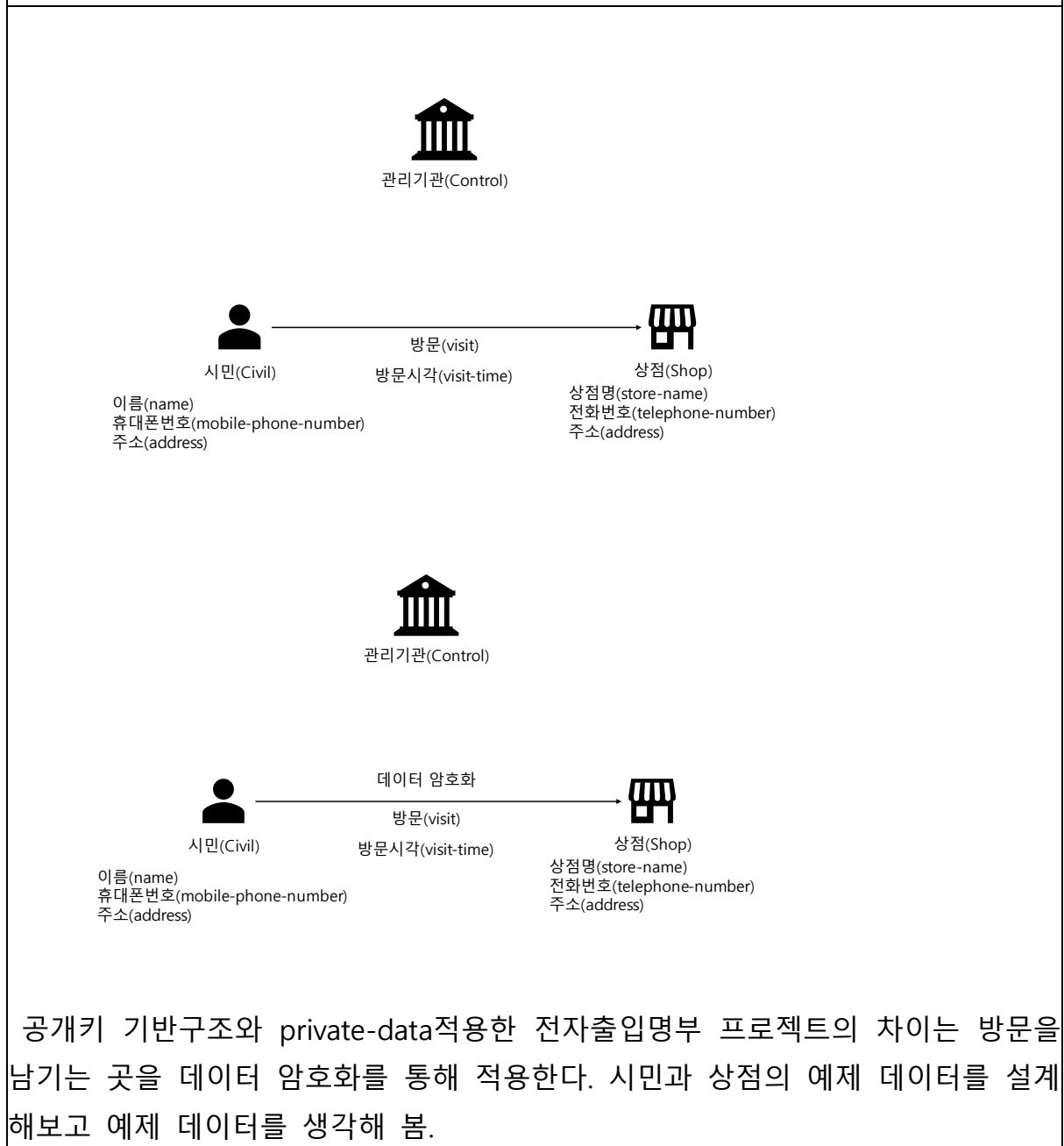
연구 보고서

작성자	김한호	작성일자	2021.04.25.
-----	-----	------	-------------

1. 연구 계획

- Hyperledger fabric asset-transfer-private-data를 참고하여 전자출입명부 개념적 설계
- 공개키 기반 구조를 체인코드 적용방안 구상 후 테스트

2. 논문 연구 진행





시민(Civil)

일련번호(ID)	이름(name)	휴대폰 번호(mobile-number)	주소(address)
1	홍길동	010-4685-9875	경기도 수원시 장안구
2	김철수	019-787-5513	서울특별시 관악구
3	이랑	016-884-7952	대전광역시 중구
...			



상점(shop)

일련번호(ID)	상점명(store-name)	전화번호(telephone-number)	주소(address)
1	버거킹 광고아브뉴프랑점	031-212-0360	경기 수원시 영통구 센트럴타운로 85
2	역전우동 0410 아주대점	031-217-0411	경기 수원시 영통구 중부대로 259 에스프 라자 107호
3	태화장 본점	031-213-6998	경기 수원시 팔달구 아주로 13번길 22
...			

많은 데이터를 넣어서 실제 세상에 적용하는 방법보다는 단순화하여 구현을 쉽게 하기로 계획했다. 시민(Civil)은 일련번호(ID), 이름(name), 휴대폰번호(mobile-number), 주소(address)의 속성(attribute)를 가지고 있음. 상점(shop)은 일련번호(ID), 상점명(store-name), 전화번호(telephone-number), 주소(address)를 정보를 저장하고 있음. 방문이라는 상황이 발생할 때에 저장하는 내용의 차이가 있기 때문에 들어갈 내용을 예측해서 설계를 구상함.

방문(visit)

일련번호(ID)	방문자번호(civil-number)	상점번호(shop-number)	방문시각(visit-time)
1	1	3	2021-04-22 11:30:00
2	2	2	2021-04-24 18:50:24
3	3	1	2021-04-25 19:24:33
...			

방문(visit)

일련번호(ID)	상점번호(shop-number)	방문정보
1	3	!343299fjidRieis88d)(43!!@#ghdjfd
2	2	*783hfuhud(*847372hfjds94uhdsav
3	1	(878fhejr89s89*&(*848302fjdhsjfkfds
...		

private-data와 공개키를 사용한 정보를 데이터 저장 차이는 방문자 번호를 가지고 있는 private-data와 방문자번호와 방문시각을 암호화하여 저장한 방문정보의 데이터 저장 차이가 있을 것 같다는 예상이 들었음.

```

func generateKeyPair(bits int) (*rsa.PrivateKey, *rsa.PublicKey) {
    // This method requires a random number of bits.
    privateKey, err := rsa.GenerateKey(rand.Reader, bits)
    if err != nil {
        fmt.Println("Error: ", err)
    }

    // The public key is part of the PrivateKey struct
    return privateKey, &privateKey.PublicKey
}

func (s *SmartContract) TestGenPubPrivKey(ctx contractapi.TransactionContextInterface) error {
    privateKey, publicKey := generateKeyPair(2048)
    log.Printf("Private Key: %v\n", privateKey)
    log.Printf("Public Key: %v\n", publicKey)
    return nil
}

func (s *SmartContract) EnDecryptTest(ctx contractapi.TransactionContextInterface, pubKeyPath string,
    , privKeyPath string, message string) error {
    pubKeyFilePath := readKeyPathFromDir(pubKeyPath)
    privKeyFilePath := readKeyPathFromDir(privKeyPath)
    log.Println(pubKeyPath, privKeyPath, message)
    pubKeyPEM := readKeyFromFile(pubKeyFilePath)
    log.Printf("readKeyFrom Public KeyFile")
    pubKeyFile := exportPEMStrToPubKey(pubKeyPEM)
    log.Printf("exportPemStringPubKey")
    cipherText, _ := rsa.EncryptOAEP(sha256.New(), rand.Reader, pubKeyFile, []byte(message), nil)
    log.Printf("exchange message to ciphertext")
    fmt.Printf("Encrypted: %v\n", cipherText)

    privKeyPEM := readKeyFromFile(privKeyFilePath)
    log.Printf("readKeyFrom Private KeyFile")
    privKeyFile := exportPEMStrToPrivKey(privKeyPEM)
    log.Printf("exportPemStringPrivKey")
    decMessage, _ := rsa.DecryptOAEP(sha256.New(), rand.Reader, privKeyFile, cipherText, nil)
    log.Printf("exchange ciphertext to message")
    fmt.Printf("Original: %s\n", string(decMessage))
    return nil
}

```

체인코드에 공개키 적용을 위해 테스트하는 EnDecryptTest 함수를 만들어서 asset-transfer-private-data의 application app.js에서 호출하여 테스트하기로 계획함.

```

statefulTxn.setTransient({
  asset_delete: tmapData
});
result = await statefulTxn.submit();

let data2ForDelete = { assetID: assetID2 };
try {
  //Non-owner Org2 should not be able to DeleteAsset. Expect an error from DeleteAsset
  console.log('--> Attempt Transaction: as Org2 DeleteAsset ' + assetID2);
  statefulTxn = contractOrg2.createTransaction('DeleteAsset');
  tmapData = Buffer.from(JSON.stringify(data2ForDelete));
  statefulTxn.setTransient({
    asset_delete: tmapData
  });
  result = await statefulTxn.submit();
  console.log('***** FAILED : expected to return an error');
} catch (error) {
  console.log(' Successfully caught the error: \n    ${error}');
}
// Delete Asset2 as Org1
console.log('--> Submit Transaction: as Org1 DeleteAsset ' + assetID2);
statefulTxn = contractOrg1.createTransaction('DeleteAsset');
tmapData = Buffer.from(JSON.stringify(data2ForDelete));
statefulTxn.setTransient({
  asset_delete: tmapData
});
result = await statefulTxn.submit();

console.log('\n--> Evaluate Transaction: ReadAsset ' + assetID2);
result = await contractOrg1.evaluateTransaction('ReadAsset', assetID2);
console.log(' result: ' + prettyJSONString(result.toString()));

console.log('\n~~~~~ As Org2 Client ~~~~~');
// Org2 can ReadAssetPrivateDetails: Org2 is owner, and private details exist in new owner's Collection
console.log('\n--> Evaluate Transaction as Org2: ReadAssetPrivateDetails ' + assetID1 +
' from ' + org2PrivateCollectionName);
result = await contractOrg2.evaluateTransaction('ReadAssetPrivateDetails', org2PrivateCollectionName, assetID1);
console.log(' result: ' + prettyJSONString(result.toString()));

await contractOrg1.evaluateTransaction('TestGenPubPrivKey')
//console.log('\n--> Test CryptoSuite public/private encrypt test');
/*let Org1PubKey = path.join(Org1PubKeyDir + '/cert.pem');
*let Org1PrivKey = path.join(Org1PrivKeyDir + '/'
*console.log('\nOrg1PublicKey:'+Org1PublicKey+'~~~~~');
*const fileExists = fs.existsSync(Org1PublicKey);

```

app.js에서 TestGenPubPrivKey함수를 호출하여 실제로 공개키 생성을 할 수 있는지 실행함.

```

h2kim@h2kim-VBox:~/fabric-samples/test-network$ docker ps
CONTAINER ID        IMAGE                                     COMMAND
NAMES              CREATED             STATUS              PORTS              COMMAND
f05f9bfd9024        dev-peer0.org1.example.com-private_1.0-75ed6ae2ecb5ea950953d5c91e48dce1556251b3c5e28afaa960a89bab1d1136-8ab7295b5e6d111973db4c2e4da34e2d82c3889fd85e878b91ca772318e79b78 "chaincode -peer.add..." 5 minutes ago Up 5 minutes dev-peer0.org1.example.com-private_1.0-75ed6ae2ecb5ea950953d5c91e48dce1556251b3c5e28afaa960a89bab1d1136
046b2ef5f598        dev-peer0.org2.example.com-private_1.0-75ed6ae2ecb5ea950953d5c91e48dce1556251b3c5e28afaa960a89bab1d1136-425abdc9b3a9d5df0064a186089c9de10256f3d9dbc910ec4b73d9e23cbcb12a "chaincode -peer.add..." 5 minutes ago Up 5 minutes dev-peer0.org2.example.com-private_1.0-75ed6ae2ecb5ea950953d5c91e48dce1556251b3c5e28afaa960a89bab1d1136
0c049fa37f23        hyperledger/fabric-peer:latest        "peer node start" 8 minutes ago Up 8 minutes 0.0.0.0:7051->7051/tcp peer0.org1.example.com
88abd5cfb0dd        hyperledger/fabric-peer:latest        "peer node start" 8 minutes ago Up 8 minutes 7051/tcp, 0.0.0.0:9051->9051/tcp peer0.org2.example.com
ac27a9ce93e4        couchdb:3.1                            "tini -- /docker-ent..." 8 minutes ago Up 8 minutes 4369/tcp, 9100/tcp, 0.0.0.0:7984->5984/tcp couchdb1
ad01bd6be81c        couchdb:3.1                            "tini -- /docker-ent..." 8 minutes ago Up 8 minutes 4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp couchdb0
fcc59a440297        hyperledger/fabric-orderer:latest      "orderer" 8 minutes ago Up 8 minutes 0.0.0.0:7050->7050/tcp orderer.example.com
e5839ac27608        hyperledger/fabric-ca:latest           "sh -c 'fabric-ca-se..." 8 minutes ago Up 8 minutes 0.0.0.0:7054->7054/tcp ca_org1
1caddc561fd0        hyperledger/fabric-ca:latest           "sh -c 'fabric-ca-se..." 8 minutes ago Up 8 minutes 7054/tcp, 0.0.0.0:9054->9054/tcp ca_orderer
66ee336df5ee        hyperledger/fabric-ca:latest           "sh -c 'fabric-ca-se..." 8 minutes ago Up 8 minutes 7054/tcp, 0.0.0.0:8054->8054/tcp ca_org2
h2kim@h2kim-VBox:~/fabric-samples/test-network$

```

체인코드에 올려서 확인함.

```
2021/04/20 08:54:28 ReadAsset: collection assetCollection, ID asset1
2021/04/20 08:54:28 ReadTransferAgreement: collection assetCollection, ID asset1
2021/04/20 08:54:28 TransferAsset Put: collection assetCollection, ID asset1
2021/04/20 08:54:30 ReadAsset: collection assetCollection, ID asset1
2021/04/20 08:54:31 ReadAssetPrivateDetails: collection Org1MSPPrivateCollection, ID asset1
2021/04/20 08:54:31 AssetPrivateDetails for asset1 does not exist in collection Org1MSPPrivateCollection
2021/04/20 08:54:31 ReadAsset: collection assetCollection, ID asset2
2021/04/20 08:54:33 Deleting Asset: asset2
2021/04/20 08:54:35 ReadAsset: collection assetCollection, ID asset2
2021/04/20 08:54:35 asset2 does not exist in collection assetCollection
2021/04/20 08:54:35 Private Key: &{{2069638300773909134103069510771083583946361503392817875624315376
2206416151135545016037574909463321082692717397883816420442339646891887156340950862561208061842298359
7018813024428001445543063080428163580528691166860784137515425111821200883842138356513367122216939753
2570630882406370814142149644444464961805395408398241721755085279940487088077219813828857822188570669
5520241436625441752522169583735564954673298445848628218981592687759794037033058652023255446160758607
8646273085541825462783497910889236221463573774325102583308869357461733786327597743204774899749836646
33817627456980648189137032216436464202920125386438541 65537} 170846135880294313372562156541672065995
5416899362977357390107295932629070260208714722420621331409235353580437349153101829394959941485107890
2687667494108002283722801774139470255817138721454617469770739744880284613528879624615863633561145331
4306194781164247859017664086044723954689200062697858865138096699073637845702044687646937491636888990
247933854481728398981278944531138280638349455518877300051652592667733502537244913863518604298221308
1820486006865421769953827444820450954449844298302098155705525931639210635853015899215011047401078730
1826399598324088953608769618610348676421504735720260974219651696100783216673 [1527804090572199462608
1290806080652876558579060460903802875307777744371399322236746834315504191247053848464049343968376068
1581279854107335426850474163229470240040720091374924525406778873864899015589141743546674544632442048
806517006210817158500834084948283085064747077177072737588122341266581693240706078279721 135464901131
3210778095912108834021436783249992846972929917459568024382979843118093868132074809741984710320499399
7446467222241007143508312451818058851254828754600622073264533994558950746773020089450503887857727874
0286763955896907897529000137068401143727326216969393464449349655125806837337288585081716850356421] {
1383105981257130996483517682481394253112295278348801785626318117328491623238458132336969862617554492
4314957536163963009477452763070851867206120265853036306924854076205839867970905571189688884841532117
3987894230751072566439655319193409643133098165711887660946288225644886334826450757564541347511481622
46706953 1141603443777234711296477192592016783702929598622737002293991057600846422276506589015928933
1483291852777972928551489515612316261417763463336953345291282850788357604972861135122442144891348000
3591917745659151671648473960866139527643629196905111430110747520133103736836314411423357772123016592
89491528787776653 1489189932296157962243281410009390614357308500751268639599030460451062683618072431
5411503982227003489812261571273309965161668938676198793656889153590632126960093427029003715029640046
4107722440149842513569529085231764689756994559862906840915108768201097046558010821744494040225569115
771099869440384831192945204 [ ]}}
2021/04/20 08:54:35 Public Key: &{{206963830077390913410306951077108358394636150339281787562431537622
0641615113554501603757490946332108269271739788381642044233964689188715634095086256120806184229835970
1881302442800144554306308042816358052869116686078413751542511182120088384213835651336712221693975325
7063088240637081414214964444446496180539540839824172175508527994048708807721981382885782218857066955
2024143662544175252216958373556495467329844584862821898159268775979403703305865202325544616075860786
4627308554182546278349791088923622146357377432510258330886935746173378632759774320477489974983664633
817627456980648189137032216436464202920125386438541 65537}
```

docker logs -f <container-number>를 확인을 통해 private-key와 public-key를 출력하는 것을 확인할 수 있음.

```

func generateKeyPair(bits int) (*rsa.PrivateKey, *rsa.PublicKey) {
    // This method requires a random number of bits.
    privateKey, err := rsa.GenerateKey(rand.Reader, bits)
    if err != nil {
        fmt.Println("Error: ", err)
    }

    // The public key is part of the PrivateKey struct
    return privateKey, &privateKey.PublicKey
}

func ExportRsaPrivateKeyAsPemStr(privkey *rsa.PrivateKey) string {
    privkey_bytes := x509.MarshalPKCS1PrivateKey(privkey)
    privkey_pem := pem.EncodeToMemory(
        &pem.Block{
            Type: "RSA PRIVATE KEY",
            Bytes: privkey_bytes,
        },
    )
    return string(privkey_pem)
}

func ExportRsaPublicKeyAsPemStr(pubkey *rsa.PublicKey) (string, error) {
    pubkey_bytes, err := x509.MarshalPKIXPublicKey(pubkey)
    if err != nil {
        return "", fmt.Errorf("Error: MarshalPKCS1PublicKey")
    }
    pubkey_pem := pem.EncodeToMemory(
        &pem.Block{
            Type: "RSA PUBLIC KEY",
            Bytes: pubkey_bytes,
        },
    )
    return string(pubkey_pem), nil
}

func (s *SmartContract) TestGenPubPrivKey(ctx contractapi.TransactionContextInterface, pkiID string)
(*PKI, error) {
    privKey, pubKey := generateKeyPair(2048)
    log.Printf("Private Key: %v\n", privKey)
    privateKey := ExportRsaPrivateKeyAsPemStr(privKey)
    log.Printf("Public Key: %v\n", pubKey)
    publicKey, err := ExportRsaPublicKeyAsPemStr(pubKey)
    if err != nil {
        return nil, fmt.Errorf("Error: fun:ExportRsaPublicKeyAsPemStr")
    }
}

```

TestGenPrivKey 함수는 새로운 키를 만들어내고 키를 통해 암호화를 할 수 있는 상황을 테스트할 수 있도록 구동한다.


```

210 func (s *SmartContract) GetPubPrivKey(ctx contractapi.TransactionContextInterface, privStr string
, pubStr string) (*rsa.PrivateKey, *rsa.PublicKey, error) {
211     privKey, err := ParseRsaPrivateKeyFromPemStr(privStr)
212     if err != nil {
213         return nil, nil, fmt.Errorf("Error: func:GetPubPrivKey Get privateKey")
214     }
215     pubKey, err := ParseRsaPublicKeyFromPemStr(pubStr)
216     if err != nil {
217         return nil, nil, fmt.Errorf("Error: func:GetPubPrivKey Get publicKey")
218     }
219     return privKey, pubKey, nil
220 }
221
222 func (s *SmartContract) EnDecryptTest(ctx contractapi.TransactionContextInterface, privStr string
, pubStr string, message string) error {
223     privKey, pubKey, err := s.GetPubPrivKey(ctx, privStr, pubStr)
224     if err != nil {
225         return fmt.Errorf("Error: func:EnDecryptTest")
226     }
227     cipherText, _ := rsa.EncryptOAEP(sha256.New(), rand.Reader, pubKey, []byte(message), nil)
228     log.Printf("exchange message to ciphertext")
229     fmt.Printf("Encrypted: %v\n", cipherText)
230
231     decMessage, _ := rsa.DecryptOAEP(sha256.New(), rand.Reader, privKey, cipherText, nil)
232     log.Printf("exchange ciphertext to message")
233     fmt.Printf("Original: %s\n", string(decMessage))
234     return nil
235 }

```

위에서 만들어진 비밀키와 공개키를 가지고 평문을 암호문으로 만드는 테스트를 하는 EnDecryptTest 함수를 작성함. 위의 함수를 app.js에 적용하여 실제로 암호화하는지 확인하기로 했음.

```

h2kim@h2kim-VBox:~/fabric-samples/asset-transfer-private-data/chaincode-go/chaincode$ docker ps
CONTAINER ID        IMAGE                                     COMMAND                  CREATED
STATUS            PORTS                                     NAMES
30943f6797a7       hyperledger/fabric-peer:latest         "peer node start"       3 minutes ago
Up 3 minutes      7051/tcp, 0.0.0.0:9051->9051/tcp         peer0.org2.example.com
2e0e5bdecbaa       hyperledger/fabric-peer:latest         "peer node start"       3 minutes ago
Up 3 minutes      0.0.0.0:7051->7051/tcp                 peer0.org1.example.com
e0a5c1a49bfc       couchdb:3.1                             "tini -- /docker-ent..." 3 minutes ago
Up 3 minutes      4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp couchdb0
ca5622c49236       hyperledger/fabric-orderer:latest      "orderer"               3 minutes ago
Up 3 minutes      0.0.0.0:7050->7050/tcp                 orderer.example.com
44ba3a770256       couchdb:3.1                             "tini -- /docker-ent..." 3 minutes ago
Up 3 minutes      4369/tcp, 9100/tcp, 0.0.0.0:7984->5984/tcp couchdb1
1a09f4c888b2       hyperledger/fabric-ca:latest           "sh -c 'fabric-ca-se..." 3 minutes ago
Up 3 minutes      0.0.0.0:7054->7054/tcp                 ca_org1
f24c2ff69b69       hyperledger/fabric-ca:latest           "sh -c 'fabric-ca-se..." 3 minutes ago
Up 3 minutes      7054/tcp, 0.0.0.0:9054->9054/tcp         ca_orderer
9d8f15892043       hyperledger/fabric-ca:latest           "sh -c 'fabric-ca-se..." 3 minutes ago
Up 3 minutes      7054/tcp, 0.0.0.0:8054->8054/tcp         ca_org2
h2kim@h2kim-VBox:~/fabric-samples/asset-transfer-private-data/chaincode-go/chaincode$

```

체인코드가 컨테이너에 올라가 있지 않은 것을 확인함.

```

2021-04-23 05:05:13.713 UTC [committer.txvalidator] Validate -> INFO 071 [mychannel] Validated block [5] in 0ms
2021-04-23 05:05:13.728 UTC [lifecycle] update -> INFO 072 Updating cached definition for chaincode 'private' on channel 'mychannel'
2021-04-23 05:05:13.736 UTC [lifecycle] update -> INFO 073 Chaincode with package ID 'private_1.0:bbc4c71504cc2a5164b6f9a30e394f5982d7366058f3e6165e4c889d9612f9e6' now available on channel mychannel for chaincode definition private:1.0
2021-04-23 05:05:14.511 UTC [couchdb] createDatabaseIfNotExist -> INFO 074 Created state database mychannel_private$$paset$collection
2021-04-23 05:05:14.837 UTC [couchdb] createIndex -> INFO 075 Created CouchDB index [indexOwner] in state database [mychannel_private$$paset$collection] using design document [_design/indexOwnerDoc]
2021-04-23 05:05:14.837 UTC [statecouchdb] ProcessIndexesForChaincodeDeploy -> INFO 076 successfully submitted index creation request present in the file [META-INF/statedb/couchdb/collections/assetCollection/indexes/indexOwner.json] for chaincode [private$$paset$collection] on channel [mychannel]
2021-04-23 05:05:14.837 UTC [cceventmgmt] HandleStateUpdates -> INFO 077 Channel [mychannel]: Handling deploy or update of chaincode [private]
2021-04-23 05:05:15.047 UTC [kvledger] CommitLegacy -> INFO 078 [mychannel] Committed block [5] with 1 transaction(s) in 1334ms (state_validation=1162ms block_and_pvtdata_commit=9ms state_commit=154ms) commitHash=[ff2d475944fac41f5772e26279f66b9049bda1f1ed6dfee79e46b63bc575abfa]
2021-04-23 05:05:15.060 UTC [comm.grpc.server] 1 -> INFO 079 streaming call completed grpc.service=protos.Deliver grpc.method=DeliverFiltered grpc.request_deadline=2021-04-23T05:05:41.699Z grpc.peer_address=192.168.32.1:59452 error="context finished before block retrieved: context canceled" grpc.code=Unknown grpc.call_duration=3.361584212s
2021-04-23 05:05:15.980 UTC [lifecycle] Work -> WARN 07a could not launch chaincode 'private_1.0:bbc4c71504cc2a5164b6f9a30e394f5982d7366058f3e6165e4c889d9612f9e6': chaincode registration failed: container exited with 2

```

로그를 확인해본 결과 체인코드가 등록되지 않은 것을 확인할 수 있었음.

```

/*
func (s *SmartContract) GetPubPrivKey(ctx contractapi.TransactionContextInterface, privStr string
, pubStr string) (*rsa.PrivateKey, *rsa.PublicKey, error) {
    privKey, err := ParseRsaPrivateKeyFromPemStr(privStr)
    if err != nil {
        return nil, nil, fmt.Errorf("Error: func:GetPubPrivKey Get privateKey")
    }
    pubKey, err := ParseRsaPublicKeyFromPemStr(pubStr)
    if err != nil {
        return nil, nil, fmt.Errorf("Error: func:GetPubPrivKey Get publicKey")
    }
    return privKey, pubKey, nil
}

func (s *SmartContract) EncryptTest(ctx contractapi.TransactionContextInterface, privStr string
, pubStr string, message string) error {
    privKey, pubKey, err := s.GetPubPrivKey(ctx, privStr, pubStr)
    if err != nil {
        return fmt.Errorf("Error: func:EncryptTest")
    }
    cipherText, _ := rsa.EncryptOAEP(sha256.New(), rand.Reader, pubKey, []byte(message), nil)
    log.Printf("exchange message to ciphertext")
    fmt.Printf("Encrypted: %v\n", cipherText)

    decMessage, _ := rsa.DecryptOAEP(sha256.New(), rand.Reader, privKey, cipherText, nil)
    log.Printf("exchange ciphertext to message")
    fmt.Printf("Original: %s\n", string(decMessage))
    return nil
}
*/

```

이번에 새로 작성한 함수를 주석하여 다시 체인코드를 올리는 프로세스를 진행함.

```

h2kim@h2kim-VBox:~/fabric-samples/asset-transfer-private-data/chaincode-go/chaincode$ docker ps
CONTAINER ID        IMAGE                                     COMMAND                  CREATED             STATUS              PORTS                               COMMAND NA
MES
2daad501b36e        dev-peer0.org2.example.com-private_1.0-e32bf5d9c54e711a396450ef0a561e630ca9ecf4f2 "chaincode de
-peer.add..."    22 seconds ago      Up 18 seconds
v-peer0.org2.example.com-private_1.0-e32bf5d9c54e711a396450ef0a561e630ca9ecf4f2bafaad8bee941aad98021f
9ed87df0a9e1        dev-peer0.org1.example.com-private_1.0-e32bf5d9c54e711a396450ef0a561e630ca9ecf4f2 "chaincode de
-peer.add..."    22 seconds ago      Up 19 seconds
v-peer0.org1.example.com-private_1.0-e32bf5d9c54e711a396450ef0a561e630ca9ecf4f2bafaad8bee941aad98021f
ad401e5ed7d1        hyperledger/fabric-peer:latest
start"            2 minutes ago        Up 2 minutes              0.0.0.0:7051->7051/tcp
er0.org1.example.com
9a927e44dab4        hyperledger/fabric-peer:latest
start"            2 minutes ago        Up 2 minutes              7051/tcp, 0.0.0.0:9051->9051/tcp
er0.org2.example.com
93011b62d76c        hyperledger/fabric-orderer:latest
derer.example.com   2 minutes ago        Up 2 minutes              0.0.0.0:7050->7050/tcp
bb991dd0ed39        couchdb:3.1
ocker-ent..."    2 minutes ago        Up 2 minutes              4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp
uchdb0
0cb368b275d5        couchdb:3.1
ocker-ent..."    2 minutes ago        Up 2 minutes              4369/tcp, 9100/tcp, 0.0.0.0:7984->5984/tcp
uchdb1
84b0f4687524        hyperledger/fabric-ca:latest
ric-ca-se..."    2 minutes ago        Up 2 minutes              7054/tcp, 0.0.0.0:8054->8054/tcp
_org2
4e0e48a20650        hyperledger/fabric-ca:latest
ric-ca-se..."    2 minutes ago        Up 2 minutes              7054/tcp, 0.0.0.0:9054->9054/tcp
_orderer
2dd56a624aff        hyperledger/fabric-ca:latest
ric-ca-se..."    2 minutes ago        Up 2 minutes              0.0.0.0:7054->7054/tcp
_org1
h2kim@h2kim-VBox:~/fabric-samples/asset-transfer-private-data/chaincode-go/chaincode$

```

제대로 도커에 체인코드를 올라가 있는 것을 확인함. 새로 작성한 코드에 문제가 발생한 것으로 추측됨. 차주에는 독립된 프로젝트로 공개키 암호화한 부분만 구현하여 테스트하는 것으로 계획을 변경.

3. 차주 계획

- 테스트 프로젝트 구현시작
- 공개키 기반 구조 체인코드에 적용

4. 참고 웹페이지

[1] <https://ichi.pro/ko/go-eseoui-rsa-amhohwa-gaideu-28694008034141>