J-PASS

DID 탈중앙화 신원인증 전자 신분증

최종장박봉

CH.1

최종장박봉

팀원 소개
역할 소개

CH.2

개발동기

CH.3

블록체인

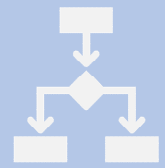
기존 방식 문제
블록체인 소개

CH.4

본론

DID
개발환경 도구
서비스 구성도
개발 내용

CH.5



결론

기대효과
Q&A

CH.1

최종장박봉

팀원 소개
역할 소개

이현종 (팀장) 총괄
백엔드 및 블록체인

박주형 (팀원) DB 개발
프론트엔드 개발

이강봉 (팀원) DB 개발
프론트엔드 개발

장예진 (팀원) 프론트엔드 개발

최유진 (팀원) 프론트엔드 개발

CH.2

개발동기

J-PASS 개발개요

현 주민등록증은 사용자의 개인정보가 그대로 노출되어 타인에게 쉽게 개인정보가 유출되기도 하며, 각종 첨단 장비를 통한 위 변조를 통하여 이를 방지하기 어려워지면서 이를 방지하기 위해 탈중앙화 신원증명(Decentralized Identity, DID)기반 모바일 신분증을 기획하게 되었다.

CH.3

블록체인

기존 방식 문제

기존 방식 문제

기존의 회원가입 방식은 중앙제어 관리 방식으로 웹사이트 자체에 개인정보를 입력하여, 회원가입을 하였다.
사용자는 각 웹사이트마다 자신의 개인정보를 입력 해야 하는 번거로움을 겪었다. 이를 극복하기 위해 기존 가입된 홈페이지를 통해 다른 웹사이트에 간편 회원가입이 가능하게 만든 SSO(Single Sign On, 통합 로그인)방식이 생기기도 하였다. (구글, NAVER 등)
이러한 방식은 중앙 관리자가 개인정보를 유출하게 될 위험성이 있다.
(실제로 페이스북과 구글은 수백만명의 개인정보를 유출한 사례가 있다.)

CH.3

블록체인

불록체인 소개

블록체인 소개

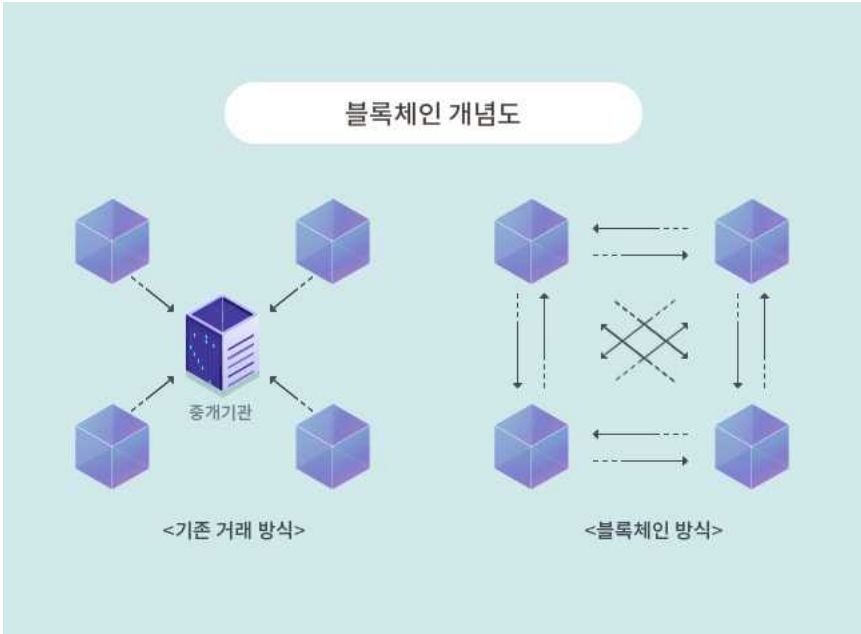
P2P(Peer to Peer) 네트워크를 통해서 관리되는 분산 데이터베이스의 형태로 분산처리와 암호화 기술을 동시에 적용하여 높은 보안성을 확보하는 한편 신속성과 투명성을 특징으로 한다.

블록 단위의 소규모 데이터들이 체인 형태로 무수히 연결되어 있어 누구도 임의로 수정할 수 없고, 누구나 변경의 결과를 열람할 수 있다는 것이 특징이다.

CH.3

블록체인

불록체인 소개



기존 중앙 기관으로 인해 관리 되었던 중앙집중 서버 방식에서 벗어나 탈중앙화 된 구조를 가지게 되어 안전한 거래를 가능하도록 만들었다.

CH.4

본론

DID

DID

블록체인을 기반으로 한 탈중앙화 신원증명(Decentralized Identifier, DID)은 블록체인의 특장점을 활용하여 중앙 시스템에 의해 통제되던 기존 신원확인 방식과 달리 개개인이 자신의 정보에 대한 완전한 통제권을 갖도록 하는 기술이다.

SSI(Self-Sovereign Identity, 자기주권 신원증명)에 사용된다.

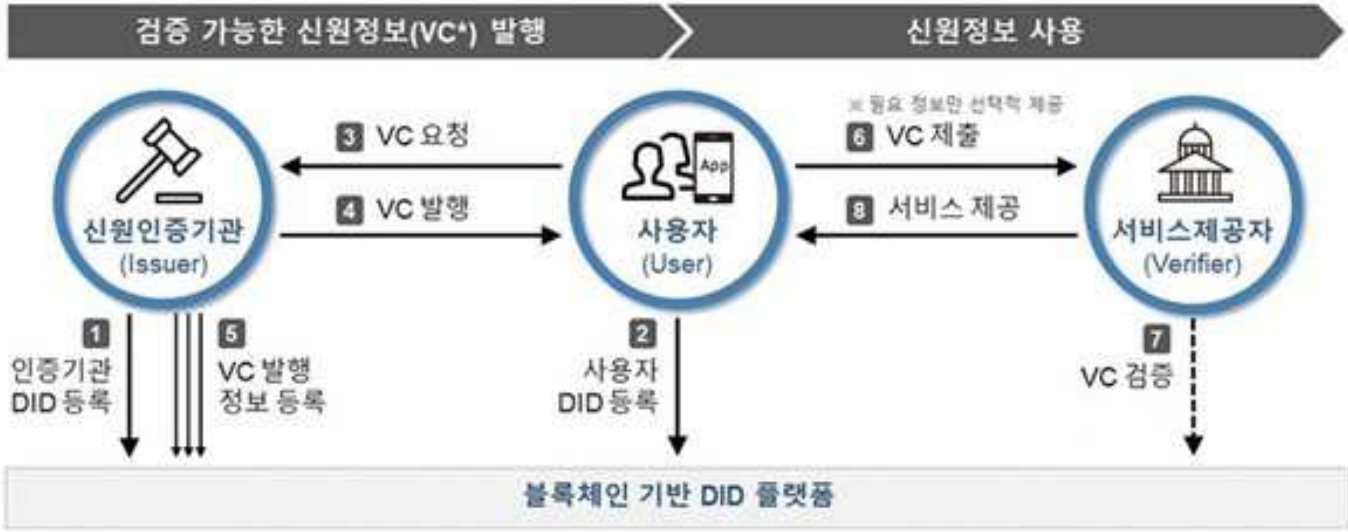
CH.4

본론

DID

SSI 구성요소

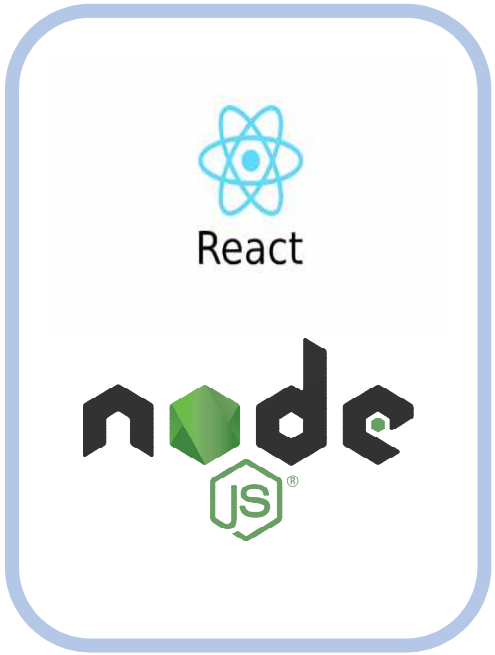
DID, DID Document, Verifiable Credential(VC),
Verifiable Presentation(VP)
VC : (보관용 ID), VP : (제출용 ID)
ZKP(Zero-Knowledge Proof)을 사용하면 속성을 알려주지 않고 검증 가능



CH.4

본론

개발환경 도구



CH.4

본론

개발환경 도구

HYPERLEDGER INDY

① 사용자 자신 외 아무도 아이덴티티를 변경/제거/삭제 할 수 없다. 자기 주권이 가능

② 인디에서 만든 아이덴티티는 여러 응용프로그램과 도메인에서 사용할 수 있고, 호환가능

③ DID를 사용하기 때문에 DID는 단일 사용자가 고유하게 소유하고 있기 때문에 ID 도용 문제 해결

④ RBFT 합의 알고리즘을 사용하여 효율적인 합의

⑤ ZKP(영지식증명)을 사용하여 다른 정보를 공개하지 않고 필요한 정보만 공개

Validation

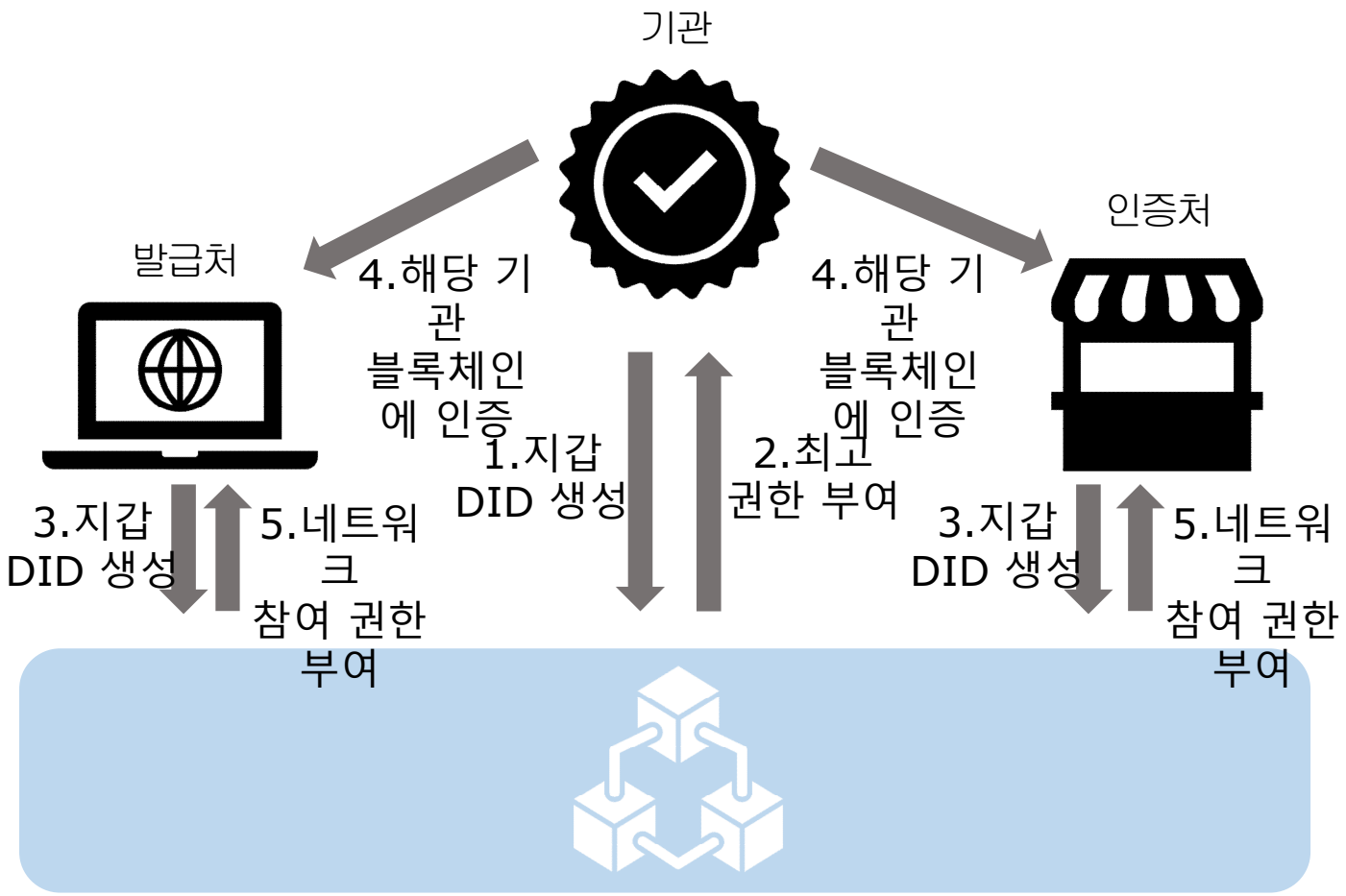
Access	Permissionless	Permissioned
Public	Bitcoin, Ethereum	Hyperledger Indy, Ripple
Private	Hollochain, LTO Network	Hyperledger Fabric, R3 Corda

CH.4

본론

서비스 구성도

초기 설정

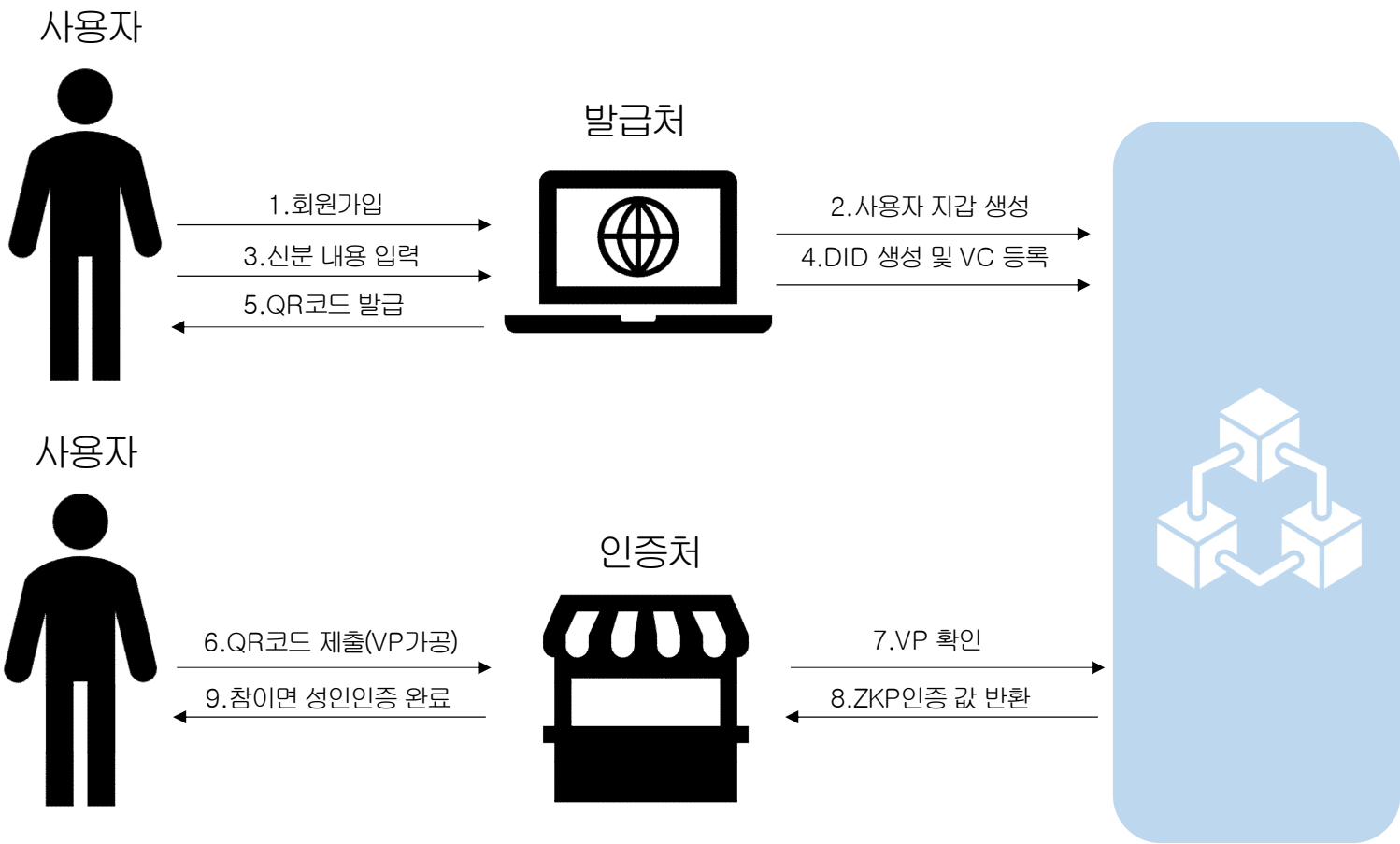


CH.4

본론

서비스 구성도

서비스 시나리오

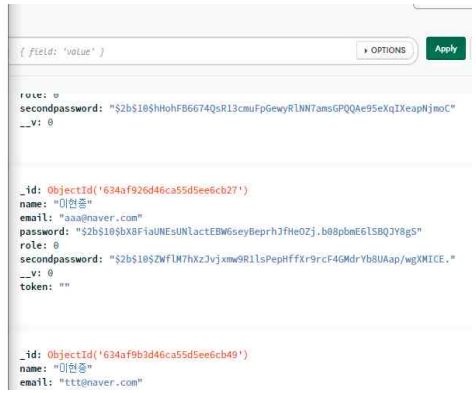


CH.4

본론

개발 내용

개발 내용 -회원가입




```
const { User } = require("../../models/User");
const indy = require("../../indy/index");

module.exports = {
  post: async (req, res) => {
    const user = new User(req.body);
    try {
      await user.save();
      let userData = await User.findOne({email: req.body.email});
      await indy.wallet.newRegister(null, userData.email, userData.password);
      return res.status(200).json({
        success: true
      })
    } catch (e) {
      return res.status(400).json({
        success: false, e
      })
    }
  },
};
```

회원가입시 사용자의 정보를 입력하고,
비밀번호, 2차비밀번호는 해시화하여 DB에 저장

```
exports.newRegister = async function (didInfoParam, walletName, walletKey) {
  await sdk.createWallet(
    { id: walletName },
    { key: walletKey }
  );
  return
}
```

블록체인에 사용자의 지갑을 생성

CH.4

본론

개발 내용

개발 내용 -로그인

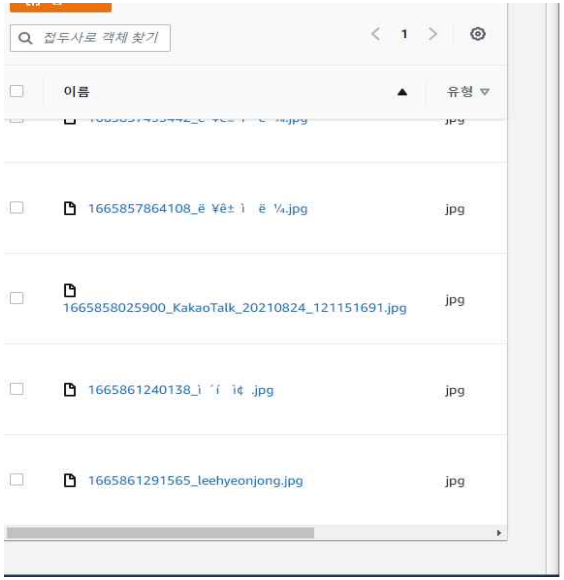
```
post: (req, res) => {
  User.findOne({ email: req.body.email }, (err, user) => {
    if (!user) {
      return res.json({
        loginSuccess: false,
        message: "제공된 이메일에 해당하는 유저가 없습니다.",
      });
    }

    user.comparePassword(req.body.password, (err, isMatch)
      if (!isMatch)
        return res.json({
          loginSuccess: false,
          message: "비밀번호가 틀렸습니다.",
        });

      user.generateToken((err, user) => {
        if (err) return res.status(400).send(err);

        res.cookie("x_auth", user.token).status(200).json(
          loginSuccess: true,
          userId: user._id,
          userToken: user.token,
        });
      });
    });
  });
},
```

CH.4

본론

개발 내용

개발 내용 – 발급



블록체인에 사용자의 DID에 대한 VC 생성

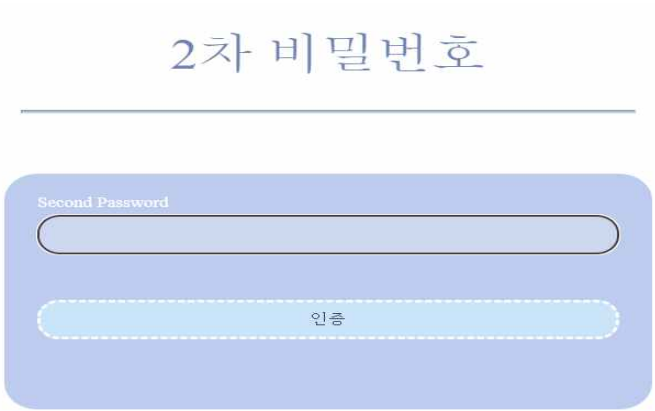
```
exports.CreateCredentialProcess = async (walletName, walletKey, value) => {
  let seedInfo = await indy.utils.walletKeyHash(walletName, walletKey);
  console.log(seedInfo);
  let proverWallet = await indy.wallet.get(walletName, walletKey);
  let issuerWallet = await indy.wallet.get(process.env.COMMUSERVEICECENTER_WALLET_NAME, process.env.COMMUSERVEICECENTER_WALLET_KEY);
  let [userDid, userVerkey] = await indy.did.createDid(seedInfo, proverWallet);
  let credOffer = await exports.sendCredOffer(issuerWallet)
  let [credReq, credReqMetaData] = await exports.sendCreateCredReq(proverWallet, credOffer)
  let [credential, revId, revRegDelta, credId] = await indy.credentials.acceptRequestCreateCredential(proverWallet, issuerWallet, credOffer, credReq, credReqMetaData, value);
  await sdk.closeWallet(issuerWallet);
  return [proverWallet, userDid, userVerkey, credential, revId, revRegDelta, credId]
}
```

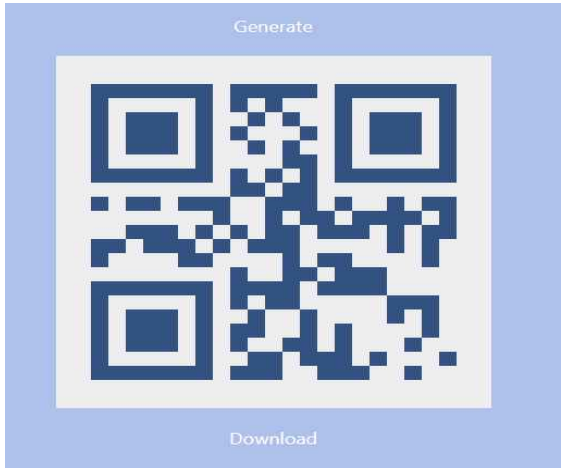
CH.4

본론

개발 내용

개발 내용 – QR





VP에 대한 사용자 VC 정보 가공

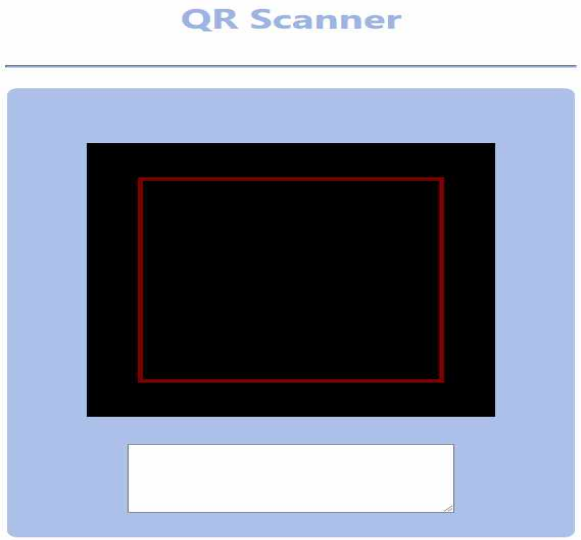
```
exports.ProverSubmitPresentation = async (proverWallet) => {
  let issuerWallet = await indy.wallet.get(
    process.env.COMMUSERVEICECENTER_WALLET_NAME,
    process.env.COMMUSERVEICECENTER_WALLET_KEY)
  let verifierWallet = await indy.wallet.get(
    process.env.STORE_WALLET_NAME,
    process.env.STORE_WALLET_KEY)
  let issuerDid = await indy.did.getDidFromWallet(issuerWallet);
  let revRegDefId = await indy.did.getEndpointDidAttribute(issuerWallet, 'revocation_registry_id')

  let [proverRevRegDelta, timestampOfDelta] = await indy.ledger.getRevRegDelta(await indy.pool.get(), issuerDid, revRegDefId[0], 0, indy.utils.getCurrentTimeInSeconds())
  let [proofRequest, credsForProof, requestedCreds] = await exports.createVerificationPresentation(issuerWallet, proverWallet, timestampOfDelta)

  let message = [proverWallet, proverRevRegDelta, timestampOfDelta, proofRequest, credsForProof, requestedCreds]
  console.log(message)
  let authCryptMessage = await indy.crypto.authCrypt(proverWallet, verifierWallet, message)
  await sdk.closeWallet(issuerWallet)
  await sdk.closeWallet(verifierWallet);
  return authCryptMessage;
```

CH.4

본론

개발 내용

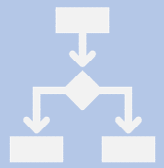
개발 내용 – QR




```
exports.verifyProof = async (encryptedMessage) => {
  let verifierWallet = await indy.wallet.get(process.env.STORE_WALLET_NAME, process.env.STORE_WALLET_KEY);
  let verifierDid = await indy.did.getDidFromWallet(verifierWallet);
  let decryptedMessage = await indy.crypto.authDecrypt(verifierWallet, encryptedMessage)
  let userData = JSON.parse(decryptedMessage["message"]);

  console.log(typeof userData);
  let proverWallet = userData[0];

  let proverDid = await indy.did.getDidFromWallet(proverWallet);

  let masterSecretId = await indy.crypto.getMasterSecretId(proverWallet);
  let [provSchemas, provCredDefs, provRevocStates] = await indy.ledger.proverGetEntitiesFromLedger(proverWallet, proverDid, userData[4], userData[1], userData[2]);

  let proof = await sdk.proverCreateProof(proverWallet, userData[3], userData[5], masterSecretId, provSchemas, provCredDefs, provRevocStates);
  let [schemas, credDefs, revRegDefs, revRegs] = await indy.ledger.verifierGetEntitiesFromLedger(verifierDid, proof["identifiers"])
  const result = await sdk.verifierVerifyProof(userData[3], proof, schemas, credDefs, revRegDefs, revRegs);

  await sdk.closeWallet(proverWallet);
  await sdk.closeWallet(verifierWallet);
  return result
}
```

인증처는 VP에 대한 사용자 인증

CH.5



결론

기대효과

기대효과

① 필요한 정보만을 제출하기 때문에 안전한 제출 가능

② 블록체인을 사용하기 때문에 무결성과 보안성을 갖춤

③ 사용자 개인이 자신의 신원 정보를 관리하기 용이

향후 계획

Q&A