JSON-LD BSS+ 테스트 진행 현황

작성일	버전	변경안	비고
22.11.16	v.0.1	초안작성	
22.11.17	v.0.2	추가 테스트 방안 작성	
22.11.22	v.0.3	테스트 9번에 대한 결과 작성	
22.12.13	v.0.4	테스트 6, 8번에 대한 결과 작성	p.14-16
23.01.02	v.0.5	파일 구조 수정, 테스트12 추가	p2, p10-p11, p20

JSON-LD BSS+ 테스트 진행도

목표

: JSON-LD 기반으로 작성된 VC에 대해 BBS 서명을 통한 선택적 증명 과정을 구현하고 검증하자

검증 리스트

1 ☑ 올바른 기능 동작 : VC에 대해서 선택적 제시가 올바르게 되고 있는가?
2 ☑ 발급 기관의 유효성 : 발급된 VC가 정말 해당 Issuer가 발급한 것인가?
3 ☑ 데이터의 무결성1 : Issuer가 발급한 VC 데이터는 변질되지 않았는지 Holder가 확인가능한가?
4 ☑ 데이터의 무결성2 : Holder가 제출한 VP 데이터는 변질되지 않았는지 Verifier가 확인가능한가?
~~5 ☐ proof request 전송 과정에서 올바른 Verifier에게 수신됨을 Holder가 확인하는 메커니즘의 종류?~~
6 ☑ 코드에 정의된 키 이외로 BBS+ 서명 기법에 이용된 방식으로 새로운 키를 생성할 수 있는가?
7 ☐ Issuer의 공개키가 변형된 경우 올바르게 처리하는가?
8 ☑ Issuer의 DID가 잘못된 경우 (존재하는 다른 Issuer의 DID로 작성된 경우) 올바르게 처리하는가?
9 ☑ Holder가 VC를 발급받은 경우, 하나의 요소가 변형이 아닌 누락된 경우 올바르게 처리하는가?
10 ☐ 키를 두 개만들고, Issuer를 두 개 만들어서 두 VC의 결합해 VP를 제출해보자.
~~11 ☐ 와세다 대학의 공개키/비밀키가 있다면, 각 이슈어마다의 공개/비밀키가 있을 텐데 가져다써보자.~~
> 하드코딩 되었다면 가져오고, 코드화 되어있다면 사용하자. 이걸로 Issuer 두 개를 만들면 되겠다.
12 ☑ 데이터의 무결성3 : Holder가 proof를 생성하기 전에 변형을 시킨 경우 올바르게 검증하는가?

각 검증 방안

1 ☑ Issuer가 발급한 VC의 개인 정보 속성과 Verifier 검증 시의 VP의 개인 정보 속성을 비교한다.
2 ☑ 검증을 위해 작성된 Issuer의 DID를 다른 혹은 존재하지 않는 Issuer의 DID로 설정한다.
3 ☑ 발급된 VC를 검증 함수를 이용해 검증한다.
4 ☑ 발급된 VP를 검증 함수를 이용해 검증한다.
~~5 ☐ proof Request에 대한 Verifier의 서명 메커니즘에 적용한다.~~
6 ☑ bbs 서명에 대한 문서의 key-generation-operations 내용을 참조하여 새로운 키를 생성한다.
7 ☐ VC와 VP에 대해 key파일의 공개키를 변형시킨 후 테스트를 진행한다.
8 ☑ 두 개의 Issuer를 생성해, 검증 시 다른 Issuer로 접근하여 검증하도록 한다.
9 ☑ 요소에 대한 각각 서명이 된 경우에 된 것에 대한 검증으로, 요소를 누락 시킨 후 검증함수를 동작한다.
10 ☐ 키를 두 개만들고, Issuer를 두 개 만들어서 두 VC의 결합해 VP를 제출해보자.
~~11 ☑ 와세다 대학의 공개키/비밀키가 있다면, 각 이슈어마다의 공개/비밀키가 있을 텐데 가져다써보자.~~
12 ☑ proof 생성 전에, VC에 대해 변형을 시킨 후 proof를 생성해 제출한다.

기반 코드

> JSONLD 기반 BBS+ 서명 : https://github.com/mattrglobal/jsonld-signatures-bbs
> 키 생성 : https://github.com/oMFDOo/bbs-signature/blob/main/draft-irtf-cfrg-bbs-signatures.md

시나리오

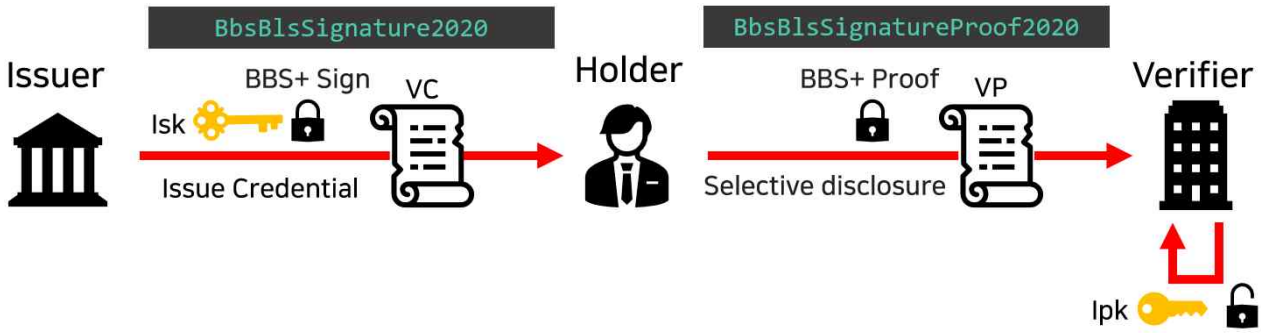
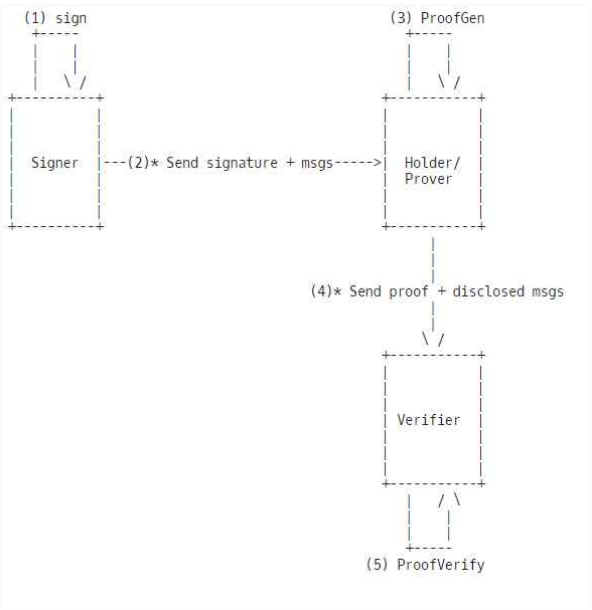
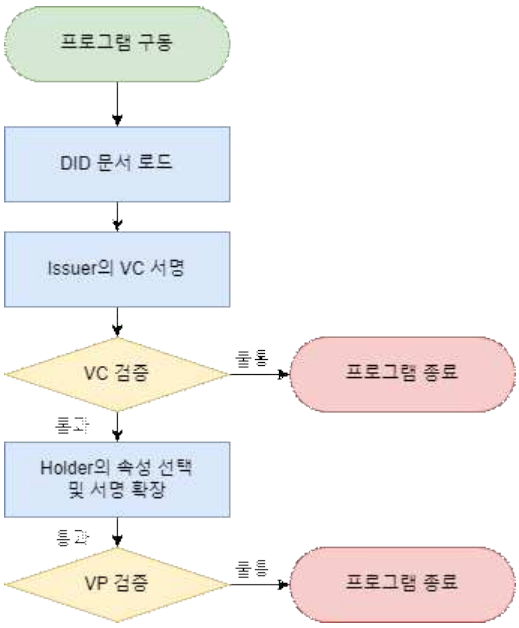


그림 1 VC 발급/선택/검증 과정에서의 서명 시나리오

1) Issuer가 Holder에게 VC를 자신의 비밀키를 통해 BBS+ 서명을 해 Holder에게 전달한다.
2-1) Issuer가 발급한 VC를 검증한다.
2-2) Holder는 VC의 속성 중 일부를 선택 후, BBS+ Proof를 생성함으로 암호화 하여 Verifier에게 전달한다.
3) Verifier는 Issuer의 공개키를 이용해 해당 내용을 검증한다.

세부적인 시나리오는 CFRG에서 작성한 BBS 서명에 관한 문서를 참조할 수 있다.





* ISK/IPK : 이슈어 공개키/비밀키 Issuer Secret Key, Issuer Public Key
* https://identity.foundation/bbs-signature/draft-irtf-cfrg-bbs-signatures.html#name-introduction

전송 데이터 명세

그림 4 발급될 정보, VC, VP의 형태

시나리오에서 Credential의 형태는 *W3C의 데이터 무결성 문서의 형태를 기반으로 한다. 해당 문서에서는 4가지 형태(Holder에게 발급할 데이터, Issuer의 서명이 포함된 VC, Holder가 제출한 VP, VC/VP 검증 결과)의 데이터를 보일 것이다.

위의 그림 2는 시나리오를 기반으로한 데이터 생성, VC, VP의 데이터의 변천을 보이고 있다. 모든 데이터는 JSON-LD 형태를 띄기 때문에, 진한 남색으로 표시 된 데이터는 각 타이틀을 의미하고 연한 파란색의 사각형은 내부 데이터를 의미한다. 사용자의 DID를 포함한 개인 정보는 "credentialSubject"에 Key/Value 형태로 저장된다. "Issuer"에는 Issuer의 DID가 정의되어 있다. 이후 VC 발급을 위해 Issuer가 서명을 하게 되면, proof란이 추가되어 서명 기법과 서명 값 등이 추가된다. 그림 2의 마지막 데이터 형태는 Holder가 VC에서 필요 데이터를 선택 후 최종 proof를 제출한 내용이다. "credentialSubject"는 Holder가 선택한 내용만이 표시되고, proof의 증명값이 변화되며, nonce값이 추가된다. 이 때, proofValue값은 항상 유일해야 하므로, 같은 내용일지라도 매번 변화된다.

* Credential 형태 : https://w3c.github.io/vc-data-integrity

아래는 VC, VP, 검증 결과에 대한 형태를 보인다.

- 예제) 발급하고자 하는 정보

```
{  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://w3id.org/citizenship/v1",
    "https://w3id.org/security/bbs/v1"
  ],
  "id": "https://issuer.oidp.uscis.gov/credentials/83627465",
  "type": [
    "VerifiableCredential",
    "PermanentResidentCard"
  ],
  "issuer": "did:example:489398593", // Issuer DID
  "identifier": "83627465",
  "name": "Permanent Resident Card",
  "description": "Government of Example Permanent Resident Card.",
  "issuanceDate": "2019-12-03T12:19:52Z", // 발급일, 폐기일
  "expirationDate": "2029-12-03T12:19:52Z",
// 이후 위의 데이터는 모두 동일하기 때문에 생략 함. //
  "credentialSubject": { // 실제 개인 정보
    "id": "did:example:b34ca6cd37bbf23",
    "type": [
      "PermanentResident",
      "Person"
    ],
    "givenName": "Jinju",
    "familyName": "Hwang",
    "gender": "Female",
    "image": "",
    "residentSince": "2015-01-01",
    "lprCategory": "C09",
    "lprNumber": "999-999-999",
    "commuterClassification": "C1",
    "birthCountry": "Bahamas",
    "birthDate": "1958-07-17"
  }
}
```

W3C는 물류 관리와 시민권이라는 두 가지 종류의 context에 대해 제공하고 있다. 현 예제는 시민권 context를 참조하여 작성되었다.

> 물류 : https://w3c-ccg.github.io/traceability-vocab/#Product
> 시민권 : https://w3c-ccg.github.io/citizenship-vocab/#abstract

- 예제) Issuer가 발급한 VC

```
{
 // 전략 //
  "credentialSubject": {
    "id": "did:example:b34ca6cd37bbf23",
    "type": [
      "PermanentResident",
      "Person"
    ],
    "givenName": "Jinju",
    "familyName": "Hwang",
    "gender": "Female",
    "image": "",
    "residentSince": "2015-01-01",
    "lprCategory": "C09",
    "lprNumber": "999-999-999",
    "commuterClassification": "C1",
    "birthCountry": "Bahamas",
    "birthDate": "1958-07-17"
  },
  "proof": { // 추가된 내용
    "type": "BbsBlsSignature2020",  // 사용한 서명 기법
    "created": "2022-11-15T15:19:46Z",
    "proofPurpose": "assertionMethod",
    "proofValue":  // 검증값
"hHrtv3+3boi2NFDOlSpLrK9J48cHpbVQeZUhT2YrovveH2V+7pVdV523qj3KAaCsYyGAUQEcMZkSnRHgUGMAQvxSmVEjKdyP+0O3nr1FAZxdsExzEsz
2k5fZMSAkgnmwrotnigMVlKES30kEyv7Ghw==",
    "verificationMethod": "did:example:489398593#test" // 증명을 위한 함수의 Endpoint
  }
}
```

Issuer가 서명을 한 VC의 내용이다. 앞선 데이터와 동일하되, 증명을 위한 내용이 추가됨을 확인 할 수 있다.

- 예제) Issuer가 발급한 VC에 대한 검증

```
{  "verified": true, // 검증 결과 : 성공
  "results": [
    {
      "proof": {
        "@context": "https://w3id.org/security/v2",
        "type": "sec:BbsBlsSignature2020",
        "created": "2022-11-15T15:19:46Z",
        "proofPurpose": "assertionMethod",
        "proofValue":
"hHrtv3+3boi2NFDOlSpLrK9J48cHpbVQeZUhT2YrovveH2V+7pVdV523qj3KAaCsYyGAUQEcMZkSnRHgUGMAQvxSmVEjKdyP+0O3nr1FAZxdsExzEsz
2k5fZMSAkgnmwrotnigMVlKES30kEyv7Ghw==",
        "verificationMethod": "did:example:489398593#test"
      },
      "verified": true
    }
  ]
}
```

Holder가 수신한 VC에 대해 검증이 통과된다면, 위와 같이 검증 결과가 true가 나오게된다. 실패하는 경우에 대해서는 이후 서술할 것이다.

- 예제) proof Request

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://w3id.org/citizenship/v1",
    "https://w3id.org/security/bbs/v1"
  ],
  "type": ["VerifiableCredential", "PermanentResidentCard"],
  "credentialSubject": {
    "@explicit": true,
    "type": ["PermanentResident", "Person"],
    "givenName": {}, // 두 가지 속성이 요구됨을 볼 수 있다.
    "familyName": {}
  }
}
```

Verifier가 Holder에게 전송하는 proof Request의 형태로, 현재 성과 이름 두 가지 속성에 대해 요구한다.

- 예제) Holder의 VC 속성 선택 및

```
{
 // 전략 //
  "credentialSubject": {
    "id": "did:example:b34ca6cd37bbf23",
    "type": [
      "Person",
      "PermanentResident"
    ], // proof Request를 기반으로 Holder에 의해 선택된 내용
    "familyName": "Hwang",
    "givenName": "Jinju"
  },
  "proof": {
    "type": "BbsBlsSignatureProof2020",
    "created": "2022-11-15T15:19:46Z",
    "nonce": "rp5TfJsyI3AAuymYQSNS7KD6ndgb4cK8OP2bw/91LNVTyfEdzxRM2G06L681vA+dOKw=", // Holder 서명 이후 추가 됨
    "proofPurpose": "assertionMethod",
    "proofValue": // Holder 서명 이후 변경된 검증 값
"ABkB/wavkrN9BuGz2yVoi4xF2FgNKFoyT4gGbce+C7Xd/rY1SE96gH/2BccyzDu5csuRsjWskZ4aKddP583AyCc5nAEQm1pj83VI3GPXy0py9gUm6ni
srd1G8Zuz+ekzaCfLgsc7jfP4s67zge2Jc39rQ9+V4q0rr0jbzN1jiH92+Mrb4s7IyRGvnmDan4E+fYa11wTeAAAAdLZgSIEqltg5GlEhsi252En6ZhR
veFGQ4bQDwFqMaRou17eMK4/S2MZa/JdYhv3RuQAAAAJpBCQGqpeXUZlWhAZKveBPrvXMy6DqIgb+0pvRpOkKkEFFmKuu4C9NsIustzJ2jqbv6DTqYfA
ieftezUD0PbGurHciSy5LBCLXOqMhlsuiFBb9RwguGeUfJQy+n/aVSDVxEVbuMFs1qSad8ENGnZ0EAAAAClLLR9/+QE01RHpoG0WNORBDvux0y/G9ddG
cKZmlqY8mUsno2voCC8xXet/CAXVfU/LmUCSbOOqdndQA5/Y1srAOwbqiv6r/r7lfSvKsnD4LMmtufu/JVziKW8UdiajgtAgBarv8jN7PE9BLgJBG8JB
3aztRHNWBmt/G2dpMKAAwGlxs3HMc3g+r8bBf7mx+sBbXPjTgOJ/WTlztAh5r+LsUek9OPY7+Fd6Zf+lDeSTl6OeNqFlMH2K3ewZ4hJnrnHKcy5QNd6l
ffT6sPQfacptUTO8cGh//Ko09IVVxh9iZF8WFB7JJVCj5SnMFhXToR5K2Kca0MS/C54mE+c5iZ6IY/KqAZ/0Bdau612eVG4Y9aDilaoIvFuJt/sVjCtO
yYlj3zOtsnhMEJVltJ2q24YOs9FIqMzXyI6yOC/B+UxxV",
    "verificationMethod": "did:example:489398593#test"
  }
}
```

Holder의 선택으로 "credentialSubject"의 속성이 Verifier가 요청한 두 가지 속성으로 줄어들었다. 또한, proof의 “proofValue”가 변경되었으며, nonce값이 추가되었다. 해당 검증값들은 고유해야 한다는 특성이 있어, 같은 내용일지라도 검증을 요청할 때마다 변경된다.

- 예제) Verifier의 VP 검증


```
{  "verified": true, // 검증 결과 : 성공
  "results": [
    {
      "proof": {
        "@context": "https://w3id.org/security/v2",
        "type": "https://w3id.org/security#BbsBlsSignature2020",
        "created": "2022-11-15T15:19:46Z",
        "nonce": "rp5TfJsyI3AAuymYQSNS7KD6ndgb4cK8OP2bw/91LNVTyfEdzxRM2G06L681vA+dOKw=",
        "proofPurpose": "assertionMethod",
        "proofValue":
"ABkB/wavkrN9BuGz2yVoi4xF2FgNKFoyT4gGbce+C7Xd/rY1SE96gH/2BccyzDu5csuRsjWskZ4aKddP583AyCc5nAEQm1pj83VI3GPXy0py9gUm6ni
srd1G8Zuz+ekzaCfLgsc7jfP4s67zge2Jc39rQ9+V4q0rr0jbzN1jiH92+Mrb4s7IyRGvnmDan4E+fYa11wTeAAAAdLZgSIEqltg5GlEhsi252En6ZhR
veFGQ4bQDwFqMaRou17eMK4/S2MZa/JdYhv3RuQAAAAJpBCQGqpeXUZlWhAZKveBPrvXMy6DqIgb+0pvRpOkKkEFFmKuu4C9NsIustzJ2jqbv6DTqYfA
ieftezUD0PbGurHciSy5LBCLXOqMhlsuiFBb9RwguGeUfJQy+n/aVSDVxEVbuMFs1qSad8ENGnZ0EAAAAClLLR9/+QE01RHpoG0WNORBDvux0y/G9ddG
cKZmlqY8mUsno2voCC8xXet/CAXVfU/LmUCSbOOqdndQA5/Y1srAOwbqiv6r/r7lfSvKsnD4LMmtufu/JVziKW8UdiajgtAgBarv8jN7PE9BLgJBG8JB
3aztRHNWBmt/G2dpMKAAwGlxs3HMc3g+r8bBf7mx+sBbXPjTgOJ/WTlztAh5r+LsUek9OPY7+Fd6Zf+lDeSTl6OeNqFlMH2K3ewZ4hJnrnHKcy5QNd6l
ffT6sPQfacptUTO8cGh//Ko09IVVxh9iZF8WFB7JJVCj5SnMFhXToR5K2Kca0MS/C54mE+c5iZ6IY/KqAZ/0Bdau612eVG4Y9aDilaoIvFuJt/sVjCtO
yYlj3zOtsnhMEJVltJ2q24YOs9FIqMzXyI6yOC/B+UxxV",
        "verificationMethod": "did:example:489398593#test"
      },
      "verified": true
    }
  ]
}
```


Verifier가 수신한 VP에 대해 검증이 통과된다면, 위와 같이 검증 결과가 true가 나오게된다. 실패하는 경우에 대해서는 이후 검증에서 서술할 것이다.

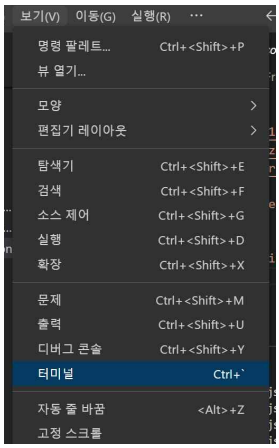
테스트 방법

앞선 시나리오 및 이후 진행 될 오류 케이스에 대한 테스트 방법에 대해 기술합니다.
테스트를 위해서는 아래 안내된 링크의 코드를 다운받아 진행 할 수 있습니다.
본격적인 코드는 "jsonld-signatures-bbs/sample/browser/" 경로에 존재합니다.

1) 코드 다운로드

- 테스트 코드 : https://github.com/oMFDOo/jsonld-signatures-bbs

2) vscode의 터미널을 열어주고 테스트 코드 경로로 이동해줍니다.



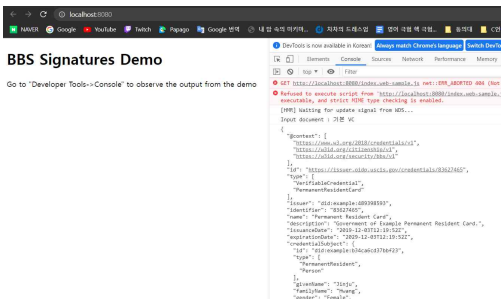

```
cd .₩sample₩browser
```

3) 종속성 다운로드 및 실행

```
C:\Users\jinjo\OneDrive\바탕 화면\jsonld-signatures-bbs\sample\browser>yarn demo
yarn run v1.22.19
$ webpack serve
(node:18996) [DEP_WEBPACK_DEV_SERVER_CONSTRUCTOR] DeprecationWarning: Using 'compiler' as the first argument
is deprecated. Please use 'options' as the first argument and 'compiler' as the second argument.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:18996) [DEP_WEBPACK_DEV_SERVER_LISTEN] DeprecationWarning: 'listen' is deprecated. Please use the async
 'start' or 'startCallback' method.
<i> [webpack-dev-server] Project is running at:
<i> [webpack-dev-server] Loopback: http://localhost:8080/
<i> [webpack-dev-server] On Your Network (IPv4): http://10.80.1.183:8080/
<i> [webpack-dev-server] Content not from webpack is served from 'C:\Users\jinjo\OneDrive\바탕 화면\jsonld-si
gnatures-bbs\sample\browser\public' directory
```

```
yarn install —frozen-lockfile
yarn demo
```

4) 결과 확인

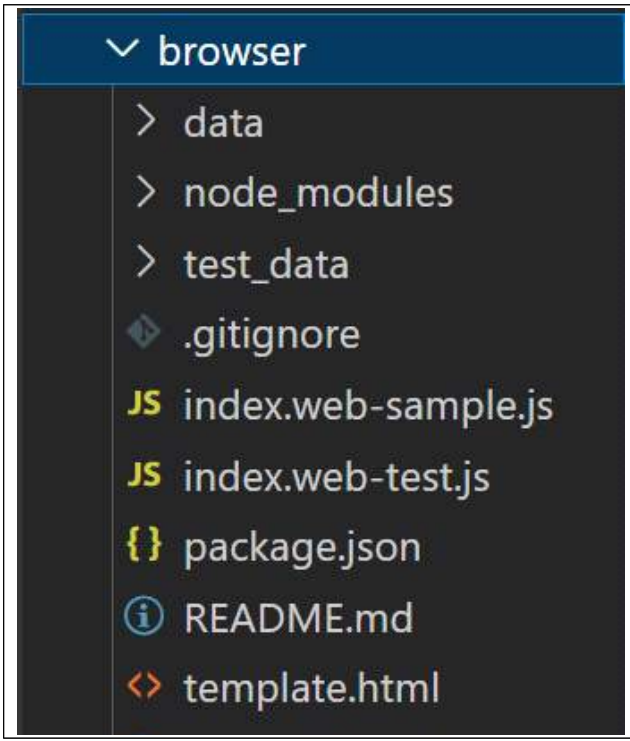


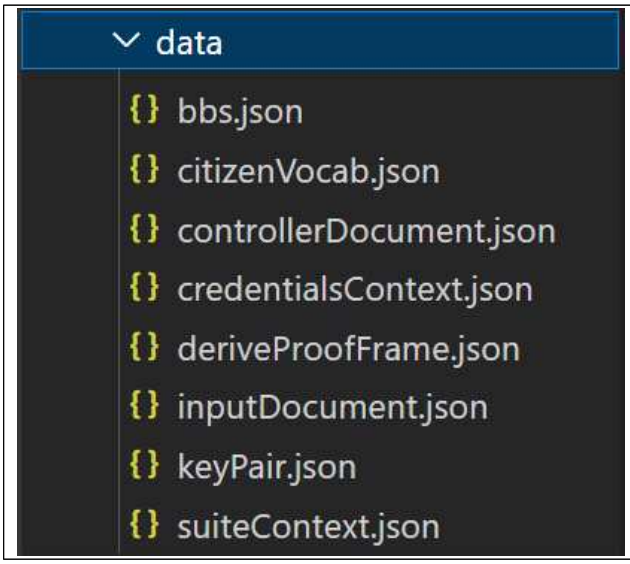

```
yarn demo
```

> http://localhost:8080/ 접속 -> f12를 눌러 개발자 모드 실행 -> Console로 들어가 로그 확인

파일 구조

임시로 작성한 테스트 파일과 실제 테스트를 위한 파일이 공존해 혼동을 유발할 수 있어 정의합니다.
실제 테스트를 위해 변경하거나 확인하는 주요한 파일에는 색을 칠해 표시했습니다.

	browser
	실제 테스트에 사용되는 데이터 (사용O)
	종속성
	추가 테스트를 위한 임시 작성 데이터 (사용X)
	github용
	Issuer/Holder/Verifier가 상호작용하는 코드 (사용O)
	임시 데이터에 대한 상호작용 테스트용 (사용X)
	종속성
	github용
	웹 페이지 UI

	browser/data
	bbs 서명에 관한 context 정의
	시민권에 관한 context 정의
	did 문서 정보 정의 : Issuer
	credential에 관한 context 정의
	proof Request : Holder 속성 선택에 영향
	Holder VC에 작성될 데이터 : VC 발급에 영향
	서명에 사용되는 키 쌍
	검증에 관한 context 정의

v0.4의 테스트 8번 이후로 Issuer가 두 명이 필요해짐에 따라 controllerDocument, deriveProofFrame, inputDocument, keyPair json 파일은 이름 끝에 “2”를 붙여 두 개 존재하게 됩니다.

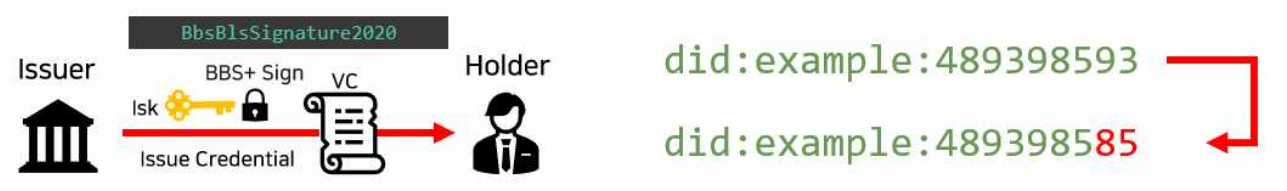
검증 테스트

아래 서술할 검증 테스트는 모두 주석처리 되어 있습니다. 주석을 해제하면 해당 테스트를 진행할 수 있습니다.

```
95    // VC 변형 테스트 1
96    // let change = JSON.stringify(signedDocument, null, 2).toString().replace(/Jinju/g,
97    // console.log(change);
98    // change = JSON.parse(change);
99    // signedDocument = change;
```

검증 번호	작성 라인	테스트 내용
3	157-160	VC 데이터 변질 테스트
9	163-167	VC 데이터 누락 테스트
12	198-202	Proof 생성 전 VP 변형 테스트
12	205-209	Proof 생성 전 VP 누락 테스트
4	238-241	Proof 생성 후 VP 변형 테스트
9	244-248	Proof 생성 후 VP 누락 테스트
8	251-254	Issuer DID 변형 테스트

검증1) 발급된 VC가 정말 해당 Issuer가 발급한 것인가?



처음 발급된 VC의 Proof가 변경되는데, 이럼에도 Verifier가 Holder의 VP를 검증할 수 있는가? 혹은 이 경우에 Verifier는 Issuer를 코드상으로 참조하고 있는가?

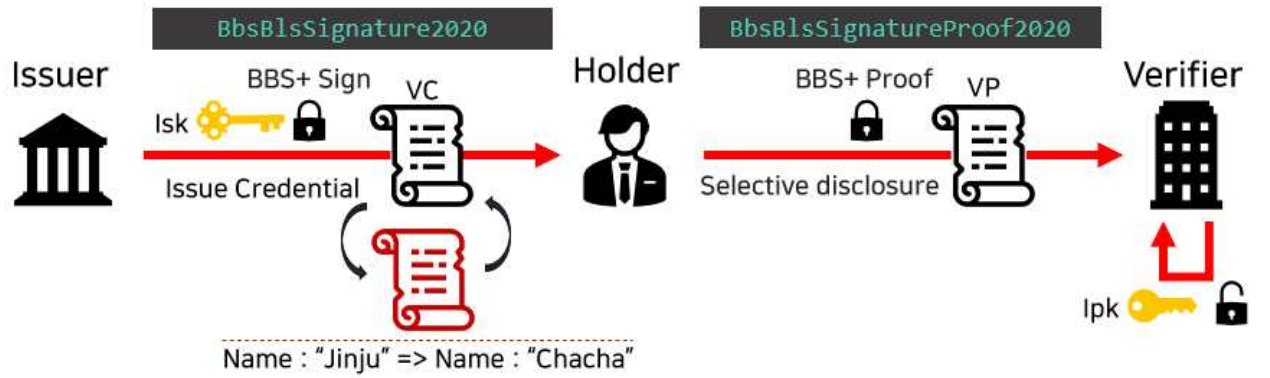
- 가정 상황 : VC의 "Issuer"란의 DID가 존재하지 않는 DID로 변질된 경우
- 기대 결과 : Holder에서 VC 검증 시 "verified": false가 되어야 함.
- 검증 방법 : VC수신 시 Issuer DID를 존재하지 않는 DID로 교체 후 검증함.
- 검증 결과 : (통과) Holder에서 VC 검증 시 "verified": false가 도출됨.


```
  "verified": false,
  "results": [
    {
      "proof": {
        "@context": "https://w3id.org/security/v2",
        "type": "https://w3id.org/security#BbsBlsSignature2020",
        "created": "2022-11-10T04:39:53Z",
        "nonce": "nJs/gApmboK4EAQO7ITsrlmp1kuMOUzrB5JcaE6qdjOtsLjE8Ska0M/ZAv/TLhQ4evw=",
        "proofPurpose": "assertionMethod",
        "proofValue":
"ABkB/wbvsgIZyMoAZ6z0xE5P1LTnQ2mC33Cn6krdF86+qgs0YuJFqb4FhKuwMn8BSQUsvzlstgJlb5cBHz9dGK9ogPWghVwiu/r0hhQorADJD8JWjVTqMsih3Br1Ef2pCB5
hnyXeI7R9ETETB5/DDe4P08JasnYdcm7NUfhfuetQWrXQMT5wR0gRnXG1UbX+R4tAAAAdI9NR1rtuIssVwlhOHcU27yV2QQ1TMFZQBzYnABtI1OXglcb6aFGOlcKSAfe6y++
AIRQazKgsrjBPrCRr/kVdTybepgrXqIYBhFPlLIYUCtdFFiKJwMYxt3DgC3DFbfrExuP/1cqb+MEj3iegQvqnZwjyqY13xBRX8JAHcux88dNMipG/PNmJFx83ssVigdiR9TD
ZgVrt/6mxrl0PiAAAACQhdfE2o9HX5KIsw4s4NBDk0xwkVdm6ED5JM3wvAI4KBa/okAJX+8HM7UoFve+82oN+ffx2fyX2j9jucWuR6Mu9QrJ920twCwEV1ZEWSKsFIIF4OY
hDsrUpvEhy29GVDCnSD1B9YSzk1QMz67RBBtx+g8oNxNXC0heJE0tI/wUHwctufSVj51SFaC11CRruYHV4guk9/24HE67eskgpCibq8mHbzzIv2M6c0mWOBoLJDAnKqkbVxF
AVw7WgUWD00pU5a1Kb7oo9vtrR0wsFNLNT/zMnQE2iGSC+scmdjiE8AZhUoLEYHx//UVWf5J2cwI0Cpn7b8d0JVeF7d1QA9r5EKLknO63MdSAbG03CqnuHJ7rTEsFBAIKA==
        "verificationMethod": "did:example:489398585#test"
      },
      "verified": false,
      "error": {
        "name": "Error",
        "message": "Attempted to remote load context : 'did:example:489398585#test', please cache instead",
        "stack": "Error: Attempted to remote load context : 'did:example:489398585#test', please cache instead\n    at customDocLoad
ebpack:///./index.web-sample.js?:80:9)\n    at eval (webpack:///./node_modules/jsonld-signatures/lib/documentLoader.js?:47:12)\n
eval (webpack:///./node_modules/jsonld-signatures/lib/documentLoader.js?:47:12)\n    at jsonld.get (webpack:///./node_modules/jsonld
jsonld.js?:876:27)\n    at jsonld.expand (webpack:///./node_modules/jsonld/lib/jsonld.js?:309:36)\n    at jsonld.frame (webpack:///.
_modules/jsonld/lib/jsonld.js?:474:33)\n    at async BbsBlsSignatureProof2020.getVerificationMethod (webpack:///./node_modules/@matt
al/jsonld-signatures-bbs/lib/BbsBlsSignatureProof2020.js?:313:24)\n    at async BbsBlsSignatureProof2020.verifyProof (webpack:///./n
odules/@mattrglobal/jsonld-signatures-bbs/lib/BbsBlsSignatureProof2020.js?:197:40)\n    at async Promise.all (index 0)\n    at async
_verify (webpack:///./node_modules/jsonld-signatures/lib/ProofSet.js?:325:11)"
      }
    }
  ]
```


> 존재하지 않는 Issuer이기 때문에 해당 문서를 찾을 수 없다는 메시지가 도출된다.
> 만약 다른 존재하는 Issuer에게 요청했다면, 어떤 결과를 도출하는지 추가 검증이 필요하다.
> == Issuer가 존재하되, 공개키가 변형된 경우

=> Verifier는 검증을 위해 Issuer에 대해 참조하고 있다.

검증2) Issuer가 발급한 VC 데이터는 변질되지 않았는지 Holder가 확인가능한가?




```
"givenName": "Jinju",
"familyName": "Hwang",
"gender": "Female",
"image": "",
"residentSince": "2015-01-01",
"lprCategory": "C09",
"lprNumber": "999-999-999",
"commuterClassification": "C1",
"birthCountry": "Bahamas",
"birthDate": "1958-07-17"
```

```
"givenName": "Chacha",
"familyName": "Hwang",
"gender": "Female",
"image": "",
"residentSince": "2015-01-01",
"lprCategory": "C09",
"lprNumber": "999-999-999",
"commuterClassification": "C1",
"birthCountry": "Bahamas",
"birthDate": "1958-07-17"
```


Issuer가 발급한 VC가 변형되었을 때, Holder는 해당 데이터가 변질됨을 알 수 있는가에 대한 검증이다.

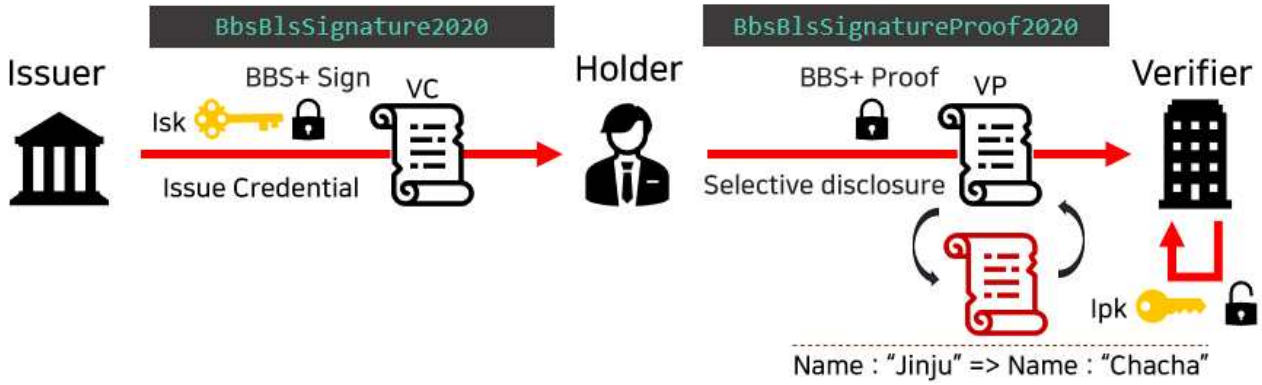
- 가정 상황 : VC의 "credentialSubject"란의 데이터가 변질된 경우
- 기대 결과 : Holder에서 VC 검증 시 "verified": false가 되어야 함.
- 검증 방법 : VC수신 시 "credentialSubject"란의 “givenName”의 데이터를 Jinju에서 Chacha로 변경함.
- 검증 결과 : (통과) Holder에서 VC 검증 시 "verified": false가 도출됨.


```
"verified": false,
"results": [
  {
    "proof": {
      "@context": "https://w3id.org/security/v2",
      "type": "sec:BbsBlsSignature2020",
      "created": "2022-11-10T03:16:16Z",
      "proofPurpose": "assertionMethod",
      "proofValue":
"hIB7CfBVPekzdDbmsKJwq01c0aV8jp32VsF3Yio8xF0zr3lXz33/6fQ5q9lmfEtlPsSFJk4WMkJGdWw3kVocmws8VfdUo5FhrnlAOqrGFpk+NWozVOruLUk/RSi3iuTsS07qqBT+
pJvjQ/C4tWEkIA==",
      "verificationMethod": "did:example:489398593#test"
    },
    "verified": false,
    "error": {
      "name": "Error",
      "message": "Invalid signature.",
      "stack": "Error: Invalid signature.\n    at BbsBlsSignature2020.verifyProof (webpack:///./node_modules/@mattrglobal/jsonld-signat
ures-bbs/lib/BbsBlsSignature2020.js?:178:23)\n    at async Promise.all (index 0)\n    at async _verify (webpack:///./node_modules/jsonld-
signatures/lib/ProofSet.js?:325:11)\n    at async ProofSet.verify (webpack:///./node_modules/jsonld-signatures/lib/ProofSet.js?:253:23)\n
at async verify (webpack:///./node_modules/jsonld-signatures/lib/jsonld-signatures.js?:38:18)\n    at async main (webpack:///./index.web-
sample.js?:119:18)"
    }
  }
],
```


> 검증 과정에서의 오류임이 도출된다.

=> Issuer의 VC 내용이 변질됨을 Holder는 알 수 있다.

검증3) Holder가 제출한 VP 데이터는 변질되지 않았는지 Verifier가 확인 가능한가?




```
"givenName": "Jinju",
"familyName": "Hwang",
"gender": "Female",
"image": "",
"residentSince": "2015-01-01",
"lprCategory": "C09",
"lprNumber": "999-999-999",
"commuterClassification": "C1",
"birthCountry": "Bahamas",
"birthDate": "1958-07-17"
```

```
],
"familyName": "Hwang",
"gender": "Female",
"givenName": "Chacha"
},
```


VC서명 후 개인 정보가 변형되었을 때, Verifier는 해당 데이터가 변질됨을 알 수 있는가에 대한 검증이다.

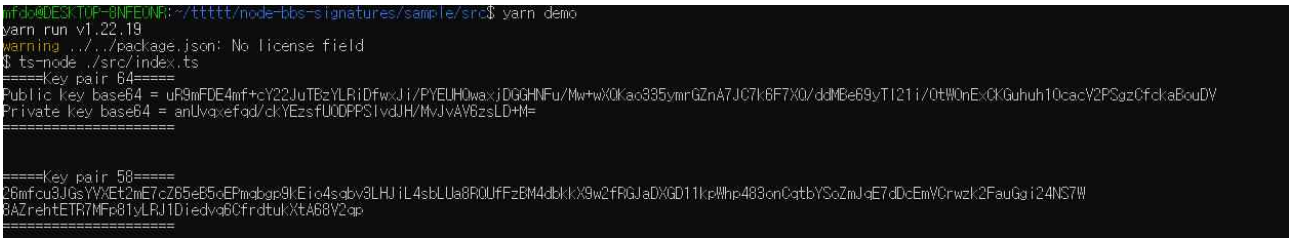
- 가정 상황 : VP의 선택 과정이 끝나고, "credentialSubject"란의 데이터가 변질된 경우
- 기대 결과 : Holder에서 VC 검증 시 "verified": false가 되어야 함.
- 검증 방법 : VP서명 후 "credentialSubject"란의 “givenName”의 데이터를 Jinju에서 Chacha로 변경함.
- 검증 결과 : (통과) Verifier에서 VP 검증 시 "verified": false가 도출됨.


```
Verification result : derivedProof 검증 결과                                    index.web-sample.js:160
{                                                                               index.web-sample.js:161
  "verified": false,
  "results": [
    {
      "proof": {
        "@context": "https://w3id.org/security/v2",
        "type": "https://w3id.org/security#BbsBlsSignature2020",
        "created": "2022-11-10T03:24:40Z",
        "nonce": "z+5EAeLT6+rrk3oDROL+LyWCR0+hCQ4yk6TkjQG6lnf/fIT7bjutrXdyu8yxd54kkrU=",
        "proofPurpose": "assertionMethod",
        "proofValue":
"ABkB/wbv1SlZC90r6ogzzOlwNLv5KHAS4/9xpPY3amI0z5kU/xpyidDCWMV11PmY55SonjxwhxN72LKDiwyahQcQkaRqDcPP/FTKUbQ1JzYWB4BRmSec1UoDeJkib6ZruORDOj0G
rWWTE/6zVWzmTs0KNcpUl2kL+aPnUuQZ/YXoi88oH+Ct+VyQwIk4T+emZYqig9s6AAAAdJdTFCvXgz55NTYL/1bagt1TsUqt1WmOlvm5ylLDPcpLpZiksUAzccpwUNDN/L75xQAAA
AIe6O7wPDxyOBN3SY66+df+2ne6FfsiYCx0C33sam8ILWqFOMKskkYdCx5O5UM8E11nyx2sdmfUuQBrSF8f51h7mP/CwOj5Mmq8lwcY08r6kdNyq+vHkwtx3ORLtAvCXbVfzj2Svz
LjW1Pb60MFP45xAAAACVSvtAubYA4Q3JNKx6hSNsw4EX0+oUfAlLUUPxOxTBaIAbznZpf7+ybwl2cCONDIqJ+MpSM/1yNu7Hbbry+uG1tNO1YqEG0L/4LdDh2afxjNlBkPjDz8h1q
EVLqgwNTFtyu4bv+geYdHwq+EkfAVLprHlb0LSqTN85GqRujbeBp5PCL8o9sfNMZbvGUOfDFEp7zBzYawrMuef2yoFue09S0+irTbqSUk0R8TIWIWibN/wnE5LMGwezzYXsT8FFJD
pDkSnR90FgfXgQv9ePU+B60E8GqVxQkixzID6ot8CUIgIfqiDtkf+wEYXs5hmY2FqviuctFTZFVj6Tt8qp+jMk8EVLN1F2EbgGZzWArdGab8JSiDsYTiXyU6njYxz+CDSg==",
        "verificationMethod": "did:example:489398593#test"
      },
      "verified": false
    }
  ]
}
```


> 검증 과정에서의 오류임이 도출된다.

=> Holder의 VP 내용이 변질됨을 Verifier는 알 수 있다.

검증6) 코드에 정의된 키 이외로 BBS+ 서명 기법에 이용된 방식으로 새로운 키를 생성할 수 있는가?



JSON-LD기반 BBS서명 코드에 적용될 수 있는 유효한 키를 생성하고자 하는 테스트이다.

- 가정 상황 : node-bbs-signatures에 정의된 generateBls12381G2KeyPair()함수를 이용해 키를 생성함.
- 기대 결과 : 유효한 키가 생성됨
- 검증 방법 : node-bbs-signatures의 sample 코드를 이용해 해당 함수를 실행함.
- 검증 결과 : base64기반의 키가 생성됨.

설치과정

구동 환경 : Windows10, WSL

1) 환경설정

```
sudo apt update
sudo apt install yarn
sudo apt install nodejs
sudo apt install npm
sudo apt get install git
```

3) node 다운로드

```
npm i -g n  // global로 n 설치
n 16 // node버전 변경
```

- yarn.lock에 "@tsconfig/node16@^1.0.2"라 선언되어있기 때문에 버전을 맞추어줌

4) 파일 다운로드 버전 설정

```
git clone https://github.com/oMFDOo/node-bbs-signatures.git
cd node-bbs-signatures
git reset --hard 668c998
```

5) bs58설치

Node BBS 구동을 위한 종속성 설치

```
yarn add @mattrglobal/node-bbs-signatures
```

6) bs58설치

base64로 키를 생성하는 Node BBS코드와 달리, JSON-LD BBS 코드의 KeyPair는 base58를 기반으로 하기 때문에 변경을 위해 다음 종속성을 추가 설치함.

```
npm i --save bs58
```

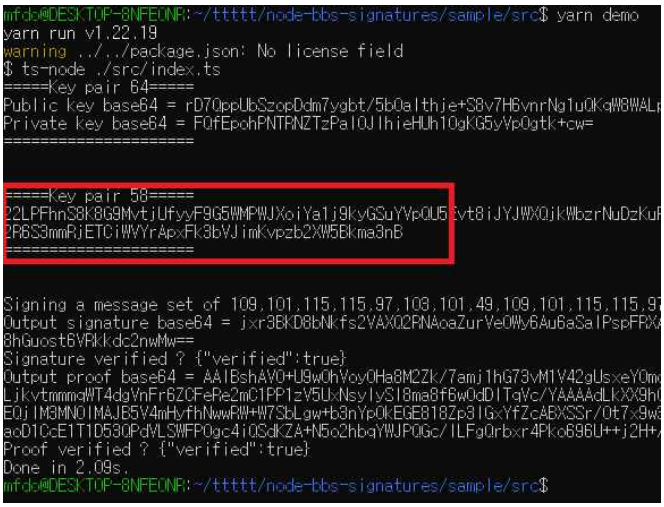
7) lockfile 기반 환경 설정

```
yarn install --frozen-lockfile
```

8) 프로그램 실행

```
yarn demo
```

출력 내용



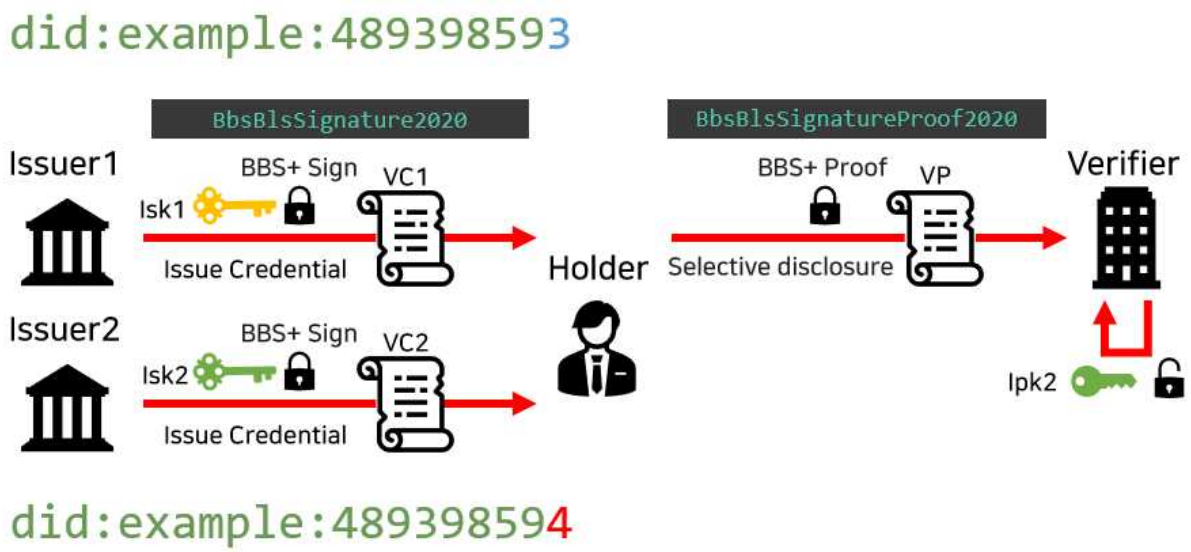
1. Base64 인코딩 키(공개키, 비밀키)
2. Base58 인코딩 키(공개키, 비밀키)
3. 서명 메시지 버퍼 내용
4. 서명 검증 결과
5. proof 생성 결과
6. proof 검증 결과

과정 2의 공개키 비밀키를 JSON-LD BBS 코드의 keyPair.json 내용에 키를 복사해 넣어주어 사용 가능하다.

검증 결과

- 새로 적용한 키를 이용해서도 검증이 올바르게 되는 모습.

검증8) Issuer의 DID가 잘못된 경우 (존재하는 다른 Issuer의 DID로 작성된 경우) 올바르게 처리하는가?



- 가정 상황 : 2명의 Issuer가 존재할 때, 다른 Issuer의 정보를 통해 검증하는 경우
- 기대 결과 : Holder/Verifier에서 VC/Proof 검증 시 "verified": false가 되어야 함.
- 검증 방법 : 다른 DID, Key를 가진 2명의 Issuer가 2개의 VC를 발급, 문서 로드 시 타 Issuer의 정보 이용
- 검증 결과 : (통과) 검증 시 "verified": false가 도출됨.

Issuer1 (VC1)	Issuer2 (VC2)
DID : 489398593], "givenName": "Jinju", "familyName": "Hwang", "gender": "Female", "image": ""	DID : 489398594], "residentSince": "2015-01-01", "lprCategory": "C09", "lprNumber": "999-999-999", "commuterClassification": "C1", "birthCountry": "Bahamas", "birthDate": "1958-07-17"

Issuer1과 Issuer2는 각각 93, 94로 끝나는 DID로 구분되고 있으며, 각기 다른 정보를 제공한다.

Issuer1 (VP1)	Issuer2 (VP2)
], "familyName": "Hwang", "gender": "Female", "givenName": "Jinju" }, "expirationDate": "2029-12-03T12:19:52Z", "issuanceDate": "2019-12-03T12:19:52Z", "issuer": "did:example:489398593",	], "birthDate": "1958-07-17", "birthCountry": "Bahamas" }, "expirationDate": "2029-12-03T12:19:52Z", "issuanceDate": "2019-12-03T12:19:52Z", "issuer": "did:example:489398594",

Holder는 선택을 통해 VC1을 통한 VP1, VC2를 이용한 VP2 총 2개의 VP를 만들어 냈다.

이후 VP1과 VP2 모두 VC2의 발급정보를 따라 검증을 진행한다. 93, 94 Issuer의 발급에의해 생성된 모든 VP를 94 Issuer의 정보로 검증을 하는 것이다.

Verification result : VP1의 derivedProof 검증 결과

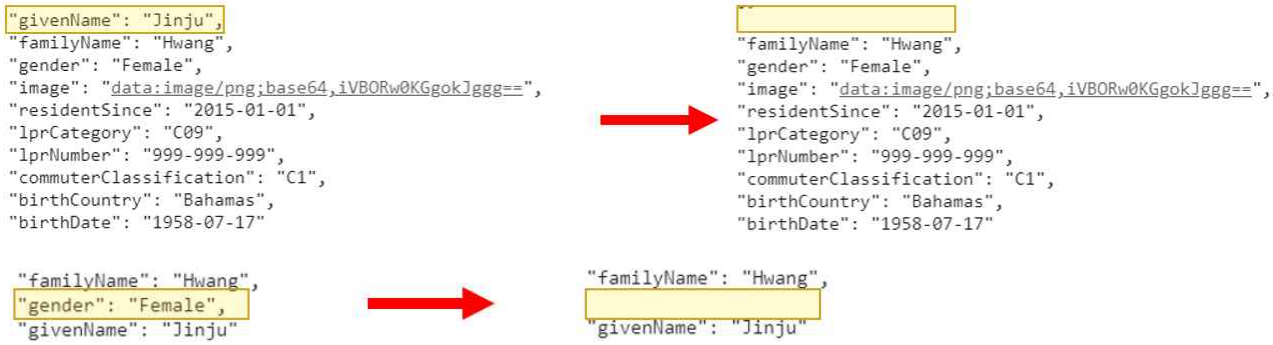

```
{
  "verified": false,
  "results": [
    {
      "proof": {
        "@context": "https://w3id.org/security/v2",
        "type": "https://w3id.org/security#BbsBlsSignature2020",
        "created": "2022-12-13T21:11:57Z",
        "nonce": "i1cwBcv1zFsEasMUUIUQariNzyYLZTRX4v3Gl/atuWxfU9fqDLmiYWgRNdawqPTyR1c=",
        "proofPurpose": "assertionMethod",
        "proofValue":
"ABUF8P+PF2IFZ05lygZqFUPGh4dsLaNtKZ0YjaGHARw3uAM1wa8aArTP3Vxmy1TP19+fM16LV2sJ2o11fOrDnZF6JMT9M3n1BypDxOnrGTd
QMDmpN8N/Y1YYS1/UUQP+M0Rx3obxSwAAAB0uEIL9VzYLwP1D+9X2z8urS4FpERPV2ICmchOB5W+8cVYAhq4WmO1AR34jegSpVzoAAAAAk58
gSm7pfTCI+y+QwvbE68cwenE4ILctcGK5RMX1nj+YL4mI8Zggs/TnypmOSvUPaDnUWLE6mo1L9/8eCQH/KizCgAAAAGQ4r92zthHVa3oUTCc
h3V+gbeVFQc5Ma+hXbETSkqVL41xXqf6m+Jg7xTC6ko07eyyaLB1GGDv71LkuPddsGRl1cEF7BPnWTMPjD7dbzLEeK1e1JGhWLyeVio0m+5
5/uOvRSjL",
        "verificationMethod": "did:example:489398594#test"
      },
      "verified": false,
      "error": {
        "name": "Error",
        "message": "Attempted to remote load context : 'did:example:489398594#test', please cache instead",
        "stack": "Error: Attempted to remote load context : 'did:example:489398594#test', please cache inste
ad\n    at eval (webpack:///./node_modules/jsonld-signatures/lib/documentLoader.js?:47:12)\n    at ev
al (webpack:///./node_modules/jsonld/lib/jsonld.js?:876:27)\n    at jsonld.
oader.js?:47:12)\n    at jsonld.get (webpack:///./node_modules/jsonld/lib/jsonld.js?:474:33)\n    at async BbsBlsSignatu
6)\n    at jsonld.frame (webpack:///./node_modules/jsonld/lib/jsonld.js?:313:24)\n    at async BbsBlsSignatureProof2020.
@mattrglobal/jsonld-signatures-bbs/lib/BbsBlsSignatureProof2020.js?:197:40)\n    at async Promise.all (index 0)\n    at asy
onld-signatures-bbs/lib/BbsBlsSignatureProof2020.js?:325:11)"
      }
    }
  ]
}
```


Verification result : VP2의 derivedProof 검증 결과


```
{
  "verified": true,
  "results": [
    {
      "proof": {
        "@context": "https://w3id.org/security/v2",
        "type": "sec:BbsBlsSignature2020",
        "created": "2022-12-13T21:11:57Z",
        "proofPurpose": "assertionMethod",
        "proofValue":
"tCQl6YPRdiMLTQKlocK/mAAzBLK1UkzPfOHn+ynyxjOL498U3+fYMm85pT
        "verificationMethod": "did:example:489398594#test"
      },
      "verified": true
    }
  ]
}
```


그 결과 Issuer 93에 의해 발급된 VP1은 검증에 실패하였지만, Issuer 94에게 발급된 VP2는 검증에 성공하는 모습을 보인다. 추가적으로, VC에서 Issuer가 발급하는 데이터가 동일한 경우에도 올바르게 작동하였다.

검증9) VC/VP 검증 시, 요소가 누락된 경우 올바르게 처리하는가?



VC/VP의 정보에 대해 각각 서명이 되어있다면, 요소가 누락된 경우 검증을 올바르게 하는 가에 대한 검증이다. 첫 번째 사진은 VC의 요소가 삭제된 모습, 두 번째는 VP의 요소가 삭제된 모습이다.

- 가정 상황 : VC/VP의 제출 시, "credentialSubject"란의 데이터가 누락된 경우
- 기대 결과 : VC/VP 검증 시 "verified": false가 되어야 함.
- 검증 방법 : VC/VP 제출 시, "credentialSubject"란의 데이터를 하나 삭제함.
- 검증 결과 : (통과) VC/VP 검증 시 "verified": false가 도출됨.

index.web-sample.js:140


```
{
  "verified": false,
  "results": [
    {
      "proof": {
        "@context": "https://w3id.org/security/v2",
        "type": "sec:BbsBlsSignature2020",
        "created": "2022-11-23T09:37:35Z",
        "proofPurpose": "assertionMethod",
        "proofValue":
"rDywI/C72SoMLvbuTYjX5eopu2VjVWdoeWVcusokRdZ4sLpgndQ5/eTWnN7zDYSaMpQvnyyEQO37dZ9A12p5nu0OYU7hXBmWwLCd6pRfqrYHFEhbb7W07qP6Oa+ezaZ98CyqIYG
5EB4RtlOcjjYIng==",
        "verificationMethod": "did:example:489398593#test"
      },
      "verified": false,
      "error": {
        "name": "Error",
        "message": "Invalid signature.",
        "stack": "Error: Invalid signature.\n    at BbsBlsSignature2020.verifyProof (webpack:///./node_modules/@mattrglobal/jsonld-signa
tures-bbs/lib/BbsBlsSignature2020.js?:178:23)\n    at async Promise.all (index 0)\n    at async _verify (webpack:///./node_modules/jsonl
d-signatures/lib/ProofSet.js?:325:11)\n    at async ProofSet.verify (webpack:///./node_modules/jsonld-signatures/lib/ProofSet.js?:253:2
3)\n    at async verify (webpack:///./node_modules/jsonld-signatures/lib/jsonld-signatures.js?:38:18)\n    at async main (webpack:///./i
ndex.web-sample.js?:132:18)"
      }
    }
  ],
```


> VC검증 과정에서의 오류임이 도출된다.

Verification result : derivedProof 검증 결과 index.web-sample.js:181

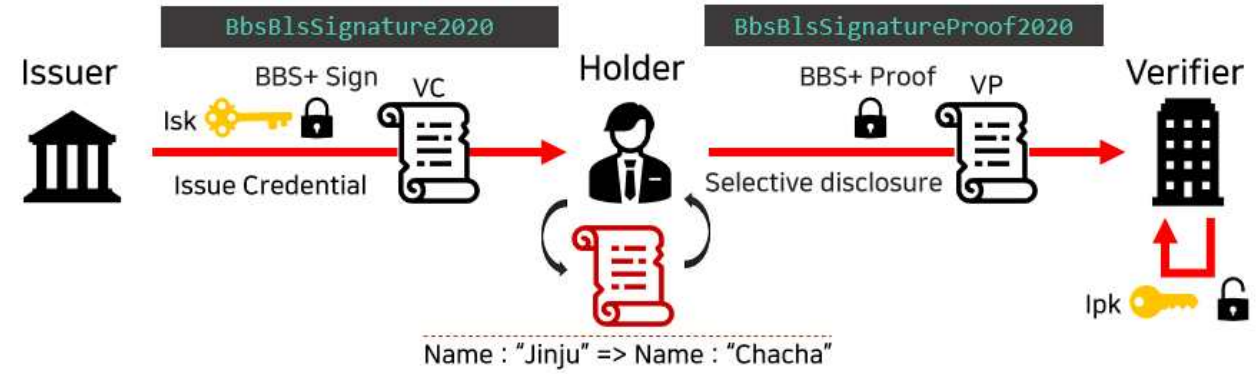
index.web-sample.js:182


```
{
  "verified": false,
  "error": {
    "name": "VerificationError",
    "errors": [
      {
        "name": "TypeError",
        "message": "Object.defineProperty called on non-object",
        "stack": "TypeError: Object.defineProperty called on non-object\n    at Function.defineProperty (<anonymous>)\n    at _addToJSON
(webpack:///./node_modules/jsonld-signatures/lib/ProofSet.js?:347:10)\n    at eval (webpack:///./node_modules/jsonld-signatures/lib/Proo
fSet.js?:338:7)\n    at Array.map (<anonymous>)\n    at _verify (webpack:///./node_modules/jsonld-signatures/lib/ProofSet.js?:333:8)\n
at async ProofSet.verify (webpack:///./node_modules/jsonld-signatures/lib/ProofSet.js?:253:23)\n    at async verify (webpack:///./node_m
odules/jsonld-signatures/lib/jsonld-signatures.js?:38:18)\n    at async main (webpack:///./index.web-sample.js?:175:14)"
      }
    ]
  }
}
```


> VP검증 과정에서의 오류임이 도출된다.

=> Holder의 VP 내용이 변질됨을 Verifier는 알 수 있다.

검증 12) Holder가 proof를 생성하기 전에 변형을 시킨 경우 올바르게 검증하는가?




```
"credentialSubject": {
  "id": "did:example:b34ca6cd37bbf23",
  "type": [
    "PermanentResident",
    "Person"
  ],
  "givenName": "Jinju",
  "familyName": "Hwang",
  "gender": "Female",
  "image": ""
},
```



```
"credentialSubject": {
  "id": "did:example:b34ca6cd37bbf23",
  "type": [
    "PermanentResident",
    "Person"
  ],
  "givenName": "Chacha",
  "familyName": "Hwang",
  "gender": "Female",
  "image": ""
},
```


VC를 받은 Holder가 Proof를 생성하기 전에 개인 정보가 변형되었을 때, Verifier는 해당 데이터가 변질됨을 알 수 있는가에 대한 검증이다.

- 가정 상황 : VP의 선택 전, "credentialSubject"란의 데이터가 변질된 경우
- 기대 결과 : Holder에서 VC 검증 시 "verified": false가 되어야 함.
- 검증 방법 : VC 발급 후, "credentialSubject"란의 "givenName"의 데이터를 Jinju에서 Chacha로 변경함.
- 검증 결과 : (통과) Holder에서 VP 생성 시 "Failed to create proof"가 도출되며 proof가 생성되지 않음.


```
▼Uncaught (in promise) Error: Failed to create proof
    at throwErrorOnRejectedPromise (wasm_module.js:40:13)
    at async module.exports.blsCreateProof (wasm_module.js:115:10)
    at async BbsBlsSignatureProof2020.deriveProof (BbsBlsSignatureProof2020.js:151:29)
    at async deriveProof (deriveProof.js:50:20)
    at async main (index.web-sample.js:250:22)
throwErrorOnRejectedPromise                @ wasm_module.js:40
await in throwErrorOnRejectedPromise (async)
eval                                       @ index.web-sample.js:315
./index.web-sample.js                      @ jsonld-signatures-bbs.min.js:1002
__webpack_require__                        @ jsonld-signatures-bbs.min.js:790
fn                                         @ jsonld-signatures-bbs.min.js:101
eval                                       @ client:3
0                                          @ jsonld-signatures-bbs.min.js:6001
__webpack_require__                        @ jsonld-signatures-bbs.min.js:790
(anonymous)                                @ jsonld-signatures-bbs.min.js:857
(anonymous)                                @ jsonld-signatures-bbs.min.js:860
```


> proof 생성 오류가 도출된다.

=> Holder의 VC 내용이 변질됨을 proof 생성 시 알 수 있다.