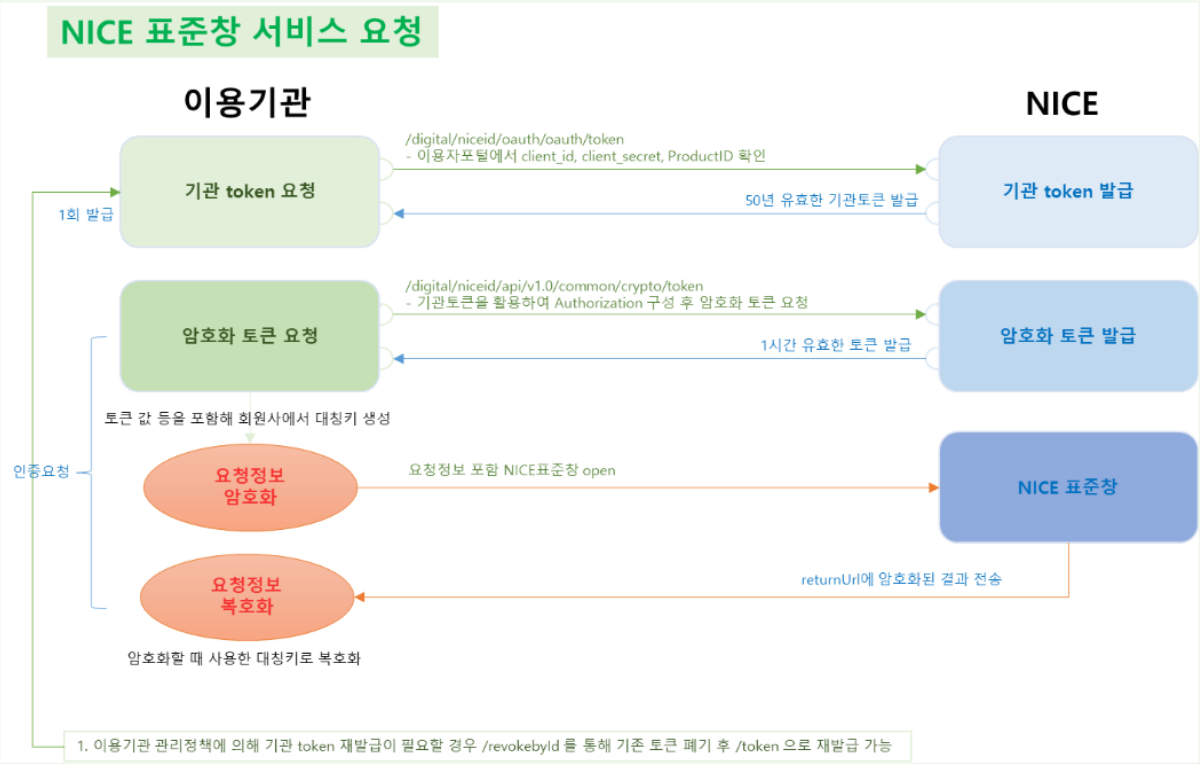
스터디 진행

본인 인증 시스템 관련 스터디로 내용 변경 (다음주는 인공지능 챗봇 관련하여 찾아볼 예정,,)

NICE API (링크)

- 휴대폰, 공인인증서 본인확인, PASS 인증서 본인인증을 통합하여 제공하는 서비스
- NICE 통합인증창 호출 API를 통해 간편하게 전체 인증수단 적용 가능
- 사용자 식별 값(CI/DI)을 제공받아 회원관리(회원가입, 비밀번호 변경 등) 가능 (PASS 인증서 이용 시 DI 제공불가)



API 기본 정보

본인확인 서비스 이용을 위해서는 암호화 토큰 발급이 필요

▼ **token 요청 과정**

1. API를 사용하기 위해 기관 token이 필요하며 “token 요청” API를 통해 token을 발급받아야 함
2. token의 유효기간은 50년임(반영구토큰)
3. token이 만료 및 폐기된 상태에서 API를 요청하면 "GW_RSLT_CD":"1800"을 받게되며 이때는 "token 요청" API로 신규 token을 다시 받아서 이용해야함
4. 발급된 token이 만료되기전 "token 요청"을 하게되면 신규 token 발급이 아닌 현재 token의 "expires_in"에 잔여시간이 내려감
   따라서, 신규 token발급이 필요할 경우에는 "토큰폐기(revokeById)"를 통해 현재 token 폐기 후 "token 요청" API로 신규 token 발급필요
5. token 만료 및 폐기후 재발급할 경우 폐기된 token을 이용한 API요청일 경우 오류가 발생하기 때문에 폐기 및 발급시점에 DB처리 또는, AP상 동기화 처리 등의 구현이 필요할 수 있음

API 명	설명
토큰발급	API 이용을 위한 access_token을 발급해주는 API입니다.
토큰폐기	access_token을 폐기해주는 API입니다.
암호화 토큰 발급	본인확인/아이핀 표준창 호출을 위한 암호화 토큰을 발급합니다.

API 목록

https://svc.niceapi.co.kr:22001/digital/niceid/oauth/oauth/token 토큰발급 API

기본정보

[HTTP Header]
- Content-Type: application/x-www-form-urlencoded
- Authorization: 생성 가이드

[POST data]
grant_type=client_credentials&scope=default

Example

```
POST /digital/niceid/oauth/oauth/token HTTP/1.1
Host: svc.niceapi.co.kr:22001
Content-type: application/x-www-form-urlencoded;charset=utf-8
Authorization: Basic Base64Encode(client_id:client_secret)
```

"Basic " + Base64Encoding(${client_id}+":" + ${client_secret})

- client_id : APP등록 시 생성값
- client_secret: APP등록 시 생성값

https://svc.niceapi.co.kr:22001/digital/niceid/oauth/oauth/token/revokeById 토큰폐기 API

기본정보

[HTTP Header]
- Content-Type: application/x-www-form-urlencoded
- Authorization: 생성 가이드

Example

```
POST /digital/niceid/oauth/oauth/token/revokeById HTTP/1.1
Host: svc.niceapi.co.kr:22001
Content-type: application/x-www-form-urlencoded;charset=utf-8
Authorization: Basic Base64Encode(access_token:current_timestamp:client_id)
```

- access_token: 토큰 발급 API를 통해 발급 받은 토큰 값(유효기간 존재)
- current_timestamp: 현재시간 Timestamp (예: new Date().getTime()/1000)
- client_id: APP등록 시 생성 값

https://svc.niceapi.co.kr:22001/digital/niceid/api/v1.0/common/crypto/token 암호화 토큰 발급

[HTTP Header]
- Content-Type: application/json
- Authorization: 생성 가이드
- client_id: [APP 등록시 발급]
- productID: 2101979031

기본정보

Parameters

Parameter 명	입력값	설명	필수여부	최대길이	자료형
req_dtim		요청일시	Y	14	string
req_no		요청고유번호	Y	30	string
enc_mode		암복호화구분(1:AES128/CBC/PKCS7)	Y	1	string

API TEST

Responses

Parameter 명	설명

"bearer " + Base64Encoding(${access_token}+":"+${current_timestamp}+":"+ ${client_id})

- access_token: 토큰 발급 API를 통해 발급 받은 토큰 값(유효기간 존재)
- current_timestamp: 현재시간 Timestamp (예: new Date().getTime()/1000)
- client_id: APP등록 시 생성 값

나중에 더 참고해볼 글 → [Node.js] NICE 본인인증 모듈 크로스 도메인 환경에서 연동하기 (서버)

영지식 증명

참고 : https://youtu.be/usSZKfm39CE?feature=shared, https://hyun-jeong.medium.com/h-3c3d45861ced

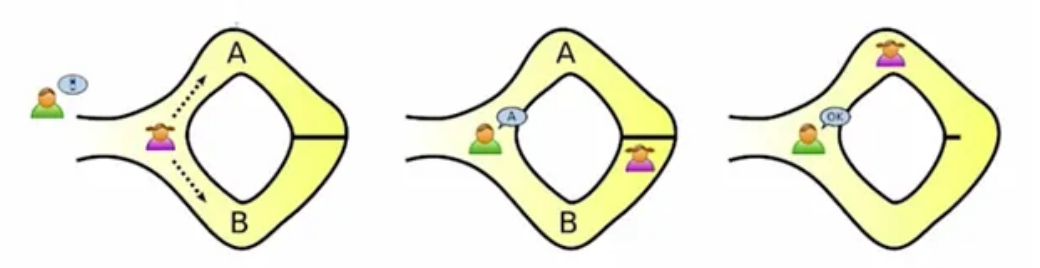
암호학에서 누군가 상대방에게 어떤 상태가 참이라는 것을 증명할 때, 그 **문장의 참, 거짓 여부를 제외한 어떤 것도 노출되지 않도록 하는 절차**이다.

영지식 증명을 활용한 프로토콜의 가장 큰 특징은 **정보를 공개하지 않고 정보의 '유효성'을 증명**할 수 있는 방법이라는 것이다.

영지식 증명에는 상태의 유효성을 증명하고자 하는 **Prover**와 이를 검증하고자 하는 **Verifier**가 참여한다.

- Prover : 자신이 가지고 있는 정보가 무엇인지 공개하지 않고, Verifier에게 '정보를 알고 있다'는 것을 증명하고 싶은 참여자
- Verifier : Prover가 해당 정보를 가지고 있음을 검증하고 싶은 참여자
- Secret : Porver가 가지고 있음을 증명하고 싶은 정보 (모두에게 숨기고자 하는 정보)
- Challenge : Verifier가 Prover가 Secret을 가지고 있는지 확인하기 위해 문제를 내는 과정
- Statementg is true : Verifier가 Prover가 Secret을 가지고 있음을 검증한 상태

예시) **The Ali Baba Cave**



Prover : 분홍 / Verifier : 초록 / Secret : 동굴 문을 열 수 있는 주문 / Challenge : Verifier가 Prover에게 나올 방향을 요구하는 과정

해당 동굴은 A와 B 방향의 두 갈래길이 있고, 가운데는 문으로 막혀있다. 동굴의 문은 주문을 통해 열 수 있고, 주문을 알지 못하면 다른 방향으로 나올 수 없다. Prover는 동굴의 문을 열 수 있는 주문을 알고 있고, 이를 Verifier에게 알리지 않은 채 자신이 주문을 알고 있다는 것을 증명하고 싶어 한다.

Prover는 자신의 주문(Secret)을 Verifier에게 알리지 않고 주문을 알고 있다는 사실을 어떻게 증명할 수 있을까?

1. Verifier는 동굴 밖에서 기다리고, Prover는 A 또는 B 방향의 길 중에서 가고 싶은 곳으로 먼저 들어감
2. Verifier는 Prover에게 A(또는 B)로 나오라고 함
3. Prover는 Verifier가 요구한 A(또는 B)로 나옴
4. 이 과정을 반복함

Prover는 동굴의 문을 여는 주문을 알고 있기 때문에, B로 들어갔어도 A로 나올 수 있음

But Prover가 주문을 모르면 B로 들어갈 경우 A로 나올 수 없고, Verifier는 Prover가 주문을 모른다고 의심할 수 있음

(Prover는 주문을 몰라도 Verifier에게 주문을 알고 있다고 속일 수 있음)

Prover가 B로 들어갔는데 운좋게 Verifier가 B로 나오라고 할 경우, Prover는 주문을 몰라도 B로 나올 수 있고, 이에 따라 Verifier를 속일 수 있음

ex)

Prover가 주문 없이 Verifier가 요구하는 방향으로 나올 수 있는 확률은 1/2.

이는 Verifier가 Prover가 들어간 방향으로 나오라고 요구할 확률과 일치함.

해당 과정을 20번 정도 반복하면 Prover가 주문을 모른 채 20번 모두 성공할 확률은 (1/2)^20임.

= 시행 횟수에 따라 성공 확률이 줄어들게 됨

= 우선 성공하면 Verifier는 Porver가 주문을 알고 있다고 '확률적'으로 확신할 수 있게 됨.

추가적인 예시와 관련하여 GPT에게 물어봄

영지식 증명의 예시 - **암호화된 메시지의 무결성 증명**

상황 : 앨리스와 밥이 안전한 통신을 하려고 함. 앨리스는 밥에게 메시지를 보내는데, 중간에 변조가 되지 않았는지를 확인하려고 함.

영지식 증명 :

1. 앨리스는 원본 메시지 M을 가지고 있음
2. 앨리스는 해시 함수를 사용하여 M을 해싱*함.
   (해시 함수는 임의 길이의 데이터를 고정된 길이의 해시 값으로 변환하는 함수)
3. 앨리스는 해싱된 메시지를 디지털 서명 알고리즘을 사용하여 암호화 함.
   디지털 서명은 앨리스의 개인 키를 사용하여 암호화되며, 이를 통해 메시지 무결성을 보장함
4. 앨리스가 암호화된 메시지와 함께 밥에게 전송.
5. 밥은 앨리스의 공개 키를 사용하여 암호화 된 메시지의 복호화를 시도.
   이를 통해 메시지가 앨리스에 의해 암호화 되었다는 것을 확인 가능.
6. 밥은 암호화된 메시지를 해시 함수를 사용하여 해싱함.
7. 밥은 앨리스의 공개 키를 이용하여 암호화 된 디지털 서명을 해독.
8. 밥은 해시된 메시지와 디지털 서명에서 얻은 결과를 비교함.
   (두 값이 일치한다면 메시지가 변조되지 않았다는 것을 영지식으로 증명한 것)

해싱 : 임의의 길이 데이터를 고정된 길이의 값으로 변환하는 과정을 말함. (=해시 값)

해시 함수를 사용하여 생성되고, 해시 함수는 데이터를 입력으로 받아서 불규칙한 알고리즘을 통해 고정된 길이의 비트열로 매핑하는 작업을 수행함.

해싱의 특징:

1. **고정된 길이의 출력**: 어떤 크기의 데이터도 해시 함수에 입력되더라도 항상 고정된 길이의 해시 값이 생성됨. ex) 256비트의 해시 함수는 항상 256비트의 출력을 생성함.
2. **일방향 함수**: 해시 함수는 일방향 함수로서, 해시 값으로부터 원본 데이터를 복구하는 것은 거의 불가능.
   즉, 해시 값으로부터 원본 데이터를 역추적하는 것 어려움.
3. **다양한 입력에 대한 고유한 출력**: 약간의 입력 값만 바꿔도 해시 값은 완전히 다른 결과를 나타내며, 약간의 입력 차이로도 해시 값은 전혀 다른 값이 됨.

해싱의 활용:

1. **데이터 무결성 검사**: 해싱은 데이터가 변조되지 않았음을 확인하는 데 사용될 수 있음.
   원본 데이터를 해시화하여 해시 값으로 저장한 뒤, 나중에 데이터가 변조되지 않았는지 확인할 때 저장된 해시 값과 실제 해시 값을 비교한다.
2. **비밀번호 보안**: 사용자 비밀번호를 저장할 때 해싱을 사용.
   원본 비밀번호를 저장하는 대신에 해시된 비밀번호를 저장하여 보안을 강화함.
3. **데이터 검색**: 해시는 데이터 구조에서 빠르게 검색하기 위해 사용될 수 있음.
   해시 테이블 등의 자료구조에서 사용됨.
4. **암호화**: 암호화에서도 해시 함수가 사용될 수 있음.
   패스워드 등을 해싱하여 저장하거나, 메시지 무결성을 검증하는 데 활용.