TTA Standard

정보통신단체표준(국문표준)

TTAK.KO-12.0429

제정일 2025.12.05.

본인확인서비스 제공을 위한 본인확인기관-이용기관 간 API 규격

API Specification for Identity Verification Services between Identity Verification Service Providers and Relying Parties

TTA 한국정보통신기술협회
Telecommunications Technology Association

표준초안 검토 위원회 개인정보보호/ID 관리, 블록체인 보안 프로젝트그룹(PG502)

표준안 심의 위원회 정보보호 기술위원회(TC5)

	성명	소 속	직위	위원회 및 직위
표준(과제) 제안	권다운	한국인터넷진흥원	선임	PG502 위원
	조경민	넥스원소프트(주)	상무	
	신상우	넥스원소프트(주)	이사	
	김용성	넥스원소프트(주)	담당	
	이수진	넥스원소프트(주)	프로	
	문현철	드로닉스	이사	
	박주선	드로닉스	책임	
	송승규	드로닉스	선임	
	김주성	드로닉스	선임	
표준 초안 에디터	권다운	한국인터넷진흥원	선임	PG502 위원
사무국 담당	오흥룡	TTA	수석	-

본 표준 발간 이전에 접수된 지식재산권 확약서 정보는 본 표준의 '부록(지식재산권 확약서 정보)'에 명시하고 있으며 이후 접수된 지식재산권 확약서는 TTA 웹사이트에서 확인할 수 있습니다. 준용표준인 경우 해당 표준화기구 또는 단체의 웹사이트에서 이를 확인해야 합니다.
본 표준과 관련하여 접수된 확약서 외의 지식재산권이 존재할 수 있습니다.

발행인 : 한국정보통신기술협회 회장
발행처 : 한국정보통신기술협회
13591, 경기도 성남시 분당구 분당로 47
Tel : 031-724-0114, Fax : 031-724-0109
발행일 : 2025. 12. 05.

서 문

1 표준의 목적

본 표준은 본인확인서비스 제공을 위해 본인확인기관과 이용기관 간 연동에 이용되는 표준 API(Application Programming Interface) 규격과 API 통신 및 인증 방법, 전송 데이터 보호 방법에 대해 정의하고 이를 통해 CI(Connecting Information) 기반 본인확인서비스의 신뢰도 향상과 개인정보 보호를 위한 조치 방법을 제시하는 것을 목적으로 한다.

2 주요 내용 요약

본 표준은 대한민국에서 i-PIN(Internet-Personal Identification Number), 휴대폰, 카드, 인증서를 이용하여 본인확인서비스를 제공하는 본인확인기관과 이용기관 간의 연동 API 규격을 정의하며 이용자의 개인정보를 안전하고 일관되게 전달하기 위한 메시지 형식, 통신 절차, 및 보안 요구사항을 포함한다. REST(Representational State Transfer) API 기반 구조를 채택하여 본인확인 요청, 인증 수행, 결과 수신 등 전체 절차를 표준화함으로써 서비스 간 연동의 효율성과 보안성을 강화한다. 표준의 주요 내용은 다음과 같다:

- 첫째, 본인확인 표준 API 규격,
- 둘째, 본인확인 표준 API 통신 및 인증 방법,
- 셋째, 전송 데이터 보호 방법

본 표준은 본인확인기관과 이용기관 간의 API 연동 구간에 적용되며 신뢰성 있는 본인확인 체계 구현을 위한 기술적 기반을 제공한다.

3 인용 표준과의 비교

해당 사항 없음

Preface

1 Purpose

This standard defines the specifications for standardized APIs(Application Programming Interface), communication and authentication methods, and transmission data protection mechanisms used in the interconnection between Identity Verification Service Providers and relying parties, in order to provide Identity verification services. The objective is to enhance the reliability of CI(Connecting Information) based identity verification services and present protective measures to safeguard personal information.

2 Summary

This standard defines the API specifications for the integration between relying parties and accredited Identity Verification Service Providers, using i-PIN(Internet-Personal Identification Number), mobile phone, card, and certificate authentication methods designated as official alternatives in the Republic of Korea. It includes message formats, communication procedures, and security requirements necessary to safely and consistently transmit users' personal information.

By adopting a REST(Representational State Transfer) API-based structure, this standard standardizes the entire process—including identity verification request, authentication execution, and result retrieval—enhancing both interoperability and security between systems.

The main components of this standard are as follows:

- First, standardized API specifications for Identity Verification Service Providers,
- Second, communication and authentication methods for secure API usage,
- Third, data protection mechanisms for transmitted information.

This standard applies to the API interaction scope between relying parties and Identity Verification Service Providers, including API access control, verification request, and result handling. It provides a technical foundation for implementing reliable identity verification systems.

3 Relationship to Reference Standards

None

목 차

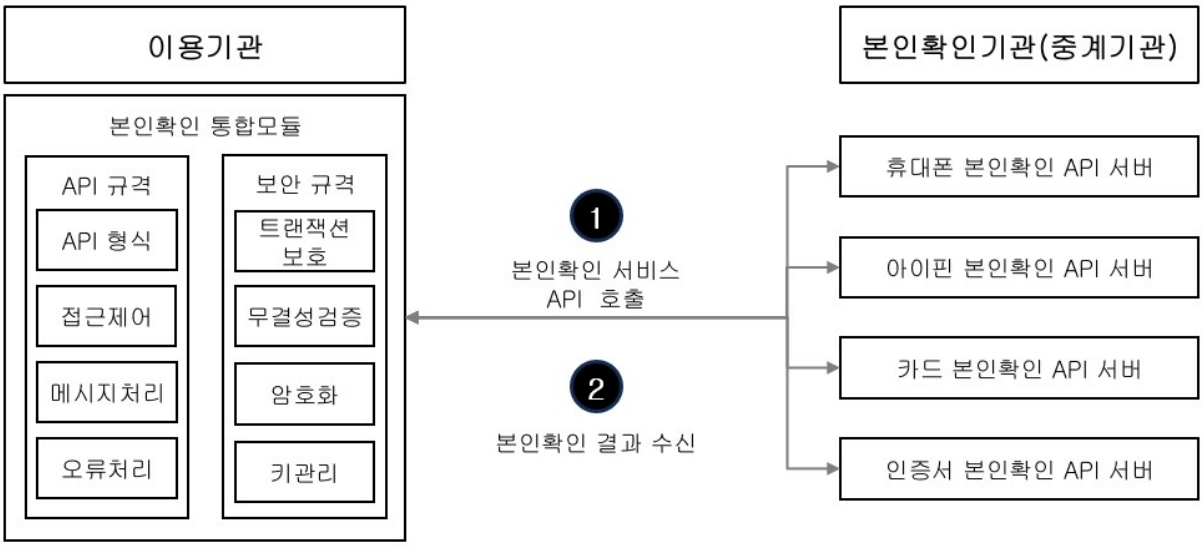
본인확인서비스 제공을 위한 본인확인기관-이용기관 간 API 규격 (API Specification for Identity Verification Services between Identity Verification Service Providers and Relying Parties)

1 적용 범위

본 표준은 본인확인서비스 제공을 목적으로, 본인확인기관과 이용기관 간 연동되는 서버 간 통신에 대한 기술 규격을 정의하며 구체적으로 다음 사항을 포함한다.

- 본인확인 표준 API 규격 (요청/응답 파라미터, 데이터 형식 등)
- 본인확인 표준 API 통신 및 인증 방법 (접근 토큰 발급 및 검증 절차 등)
- 전송 데이터 보호 방법 (암호화, 무결성 검증, 키 관리 등)

본 표준의 범위는 본인확인기관(혹은 중계기관)과 이용기관 간의 API 연동 구간으로 한정하며 이용자와 본인확인기관 간의 인증 방식, 중계기관과 본인확인기관 간의 연동 규격은 본 표준의 범위에서 제외한다.



(그림 1-1) 본인확인 서비스 API 규격

2 인용 표준

본 표준에서는 아래의 기술문서를 인용하여 기술 규격을 정의하며, 각 기능별 세부 내용은 해당 항목에서 상세히 서술한다.

- RFC 2119: Key words for use in RFCs to Indicate Requirement Levels
- RFC 4648: The Base16, Base32, and Base64 Data Encodings
- RFC 8259: The JavaScript Object Notation (JSON) Data Interchange Format

- RFC 6749: The OAuth 2.0 Authorization Framework

3 용어 정의 및 효력

3.1 용어의 정의

3.1.1 API(Application Programming Interface)

운영 체제, 프로그래밍 언어 등에 있는 라이브러리를 응용 프로그램 개발 시 이용할 수 있도록 규칙들을 정의해 놓은 인터페이스.

[출처: TTA정보통신용어 사전]

3.1.2 본인확인서비스

이용자가 본인확인기관이 제공하는 대체수단(i-PIN, 휴대전화, 카드, 인증서 등)을 통해 자신의 신원을 증명하는 서비스. 본인확인 입력정보를 이용하여 안전하게 이용자를 식별·인증하기 위한 수단임.

[출처: 정보통신망법]

3.1.3 이용자

본인확인서비스를 통해 본인확인을 받는 주체로 본인확인기관에 개인정보를 제공하고 대체수단을 부여받은 자.

[출처: 정보통신망법]

3.1.4 본인확인기관

이용자의 주민등록번호를 사용하지 않고 본인을 확인할 수 있는 수단(대체수단)을 제공하며 방송통신위원회로부터 지정받은 기관. 본인확인서비스를 운영·제공하는 역할을 수행함.

[출처: 정보통신망법]

3.1.5 중계기관

본인확인기관이 본인확인서비스를 직접 제공하지 않고 이를 대신하여 이용자 요청을 중계하여 본인확인기관과 이용기관 간의 기술적·행정적 연계를 수행하는 기관.

3.1.6 트랜잭션(transaction)

트랜잭션은 조화 또는 다단계 상호 작용에서 결과 또는 작업의 조화를 지원하는 아키텍처의 기능. 트랜잭션의 근본적 특성은 여러 작업을 같은 작업 단위로 결합시켜 해당 작업들이 한 단위로 성공하거나 실패하도록 하는 기능임.

[출처: TTA정보통신용어 사전]

3.1.7 JSON(JavaScript Object Notation)

속성-값 쌍 또는 "키-값 쌍"으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용하는 개방형 표준 포맷을 나타냄.

[출처: TTA정보통신용어 사전]

3.1.8 클라이언트 아이디(Client Identifier)

클라이언트 식별자는 클라이언트 등록 정보를 나타내는 고유 문자열로 본인확인 API 호출 주체인 이용기관을 식별하기 위한 고유 식별 값임.

[출처: RFC 6749 - The OAuth 2.0 Authorization Framework]

3.1.9 클라이언트 시크릿(Client Secret)

클라이언트를 식별하고 인증하기 위해 부여받는 값으로 클라이언트와 권한 서버만이 알고 있는 비밀 값임.

[출처: RFC 6749 - The OAuth 2.0 Authorization Framework]

3.1.10 연계정보(Connecting Information)

정보통신서비스 제공자의 온·오프라인 서비스 연계를 위해 본인확인기관이 이용자의 주민등록번호와 본인확인기관간 공유 비밀정보를 이용하여 생성한 정보로 서로 다른 인터넷 서비스 간에도 동일한 이용자인지 구분을 가능하게 해주는 연계정보.

[출처: 방송통신위원회 고시(본인확인기관 지정 등에 관한 기준)]

3.1.11 중복가입확인정보(Duplication Information)

웹사이트에 가입하고자 하는 이용자의 중복가입 여부를 확인하는 데 사용되는 정보로서 본인확인기관이 이용자의 주민등록번호, 웹사이트 식별번호 및 본인확인기관간 공유비밀정보를 이용하여 생성한 정보

[출처: 방송통신위원회 고시(본인확인기관 지정 등에 관한 기준)]

3.1.12 공개키암호 표준(Public-Key Cryptography Standards)

디지털 서명과 같은 암호화된 데이터를 표현하는 문법에 대한 표준으로 RFC 2315에 정의됨.

[출처: TTA정보통신용어 사전]

3.1.13 메시지 인증 코드(The Keyed-Hash Message Authentication Code)

미국 국립표준기술연구소(NIST, National Institute of Standard and Technology)에서 표

준으로 정하고 있는 키 기반의 메시지 인증 알고리즘.

[출처: TTA정보통신용어 사전]

3.1.14 유도키 함수(Key Derivation Function)

키 유도 함수로 공유된 비밀값을 이용하여 비밀키를 생성하는 함수.

[출처: TTA정보통신용어 사전]

3.1.15 표준창

본인확인기관이 이용기관에 제공하는 인증창.

3.1.16 민감 정보(Sensitive Information)

누출이나 훼손되었을 때 정보의 소유자에게 부정적 영향이 발생하며 시스템의 계속적 운영이 불가능해지고 상당한 양의 자원을 다시 생성해야 하는 상황을 유발하는 정보. 이름, 성명, 전화번호, 주민등록번호, CI, DI 등 개인을 식별하거나 유추할 수 있는 정보를 의미함.

[출처: TTA정보통신용어 사전]

3.1.17 콜백(callback)

호출될 함수를 알려 주어 다른 프로그램 또는 다른 모듈에서 함수를 호출하게 하는 방법. 즉, 함수에 전달되는 “또 다른 함수”로 작업이 끝났을 때 또는 특정 이벤트가 발생했을 때 호출됨.

[출처: TTA정보통신용어 사전]

3.1.18 ticket

메시지 암·복호화 시 필요한 키를 유도하기 위해 발급받는 데이터.

3.1.19 cpCode

본인확인기관이 본인확인서비스 이용기관을 구분하기 위해 생성한 웹사이트 식별번호.

3.2 용어의 효력

본 표준에서 사용된 다음의 용어들은 본인확인서비스를 제공 및 처리하는 데에 따라야할 구현 정도를 의미한 것으로 RFC 2119를 준용하여 다음과 같은 의미를 갖는다.

- 해야한다, 필수이다, 강제한다 (기호 : M (MUST))
  - 반드시 준수해야 한다.
- 조건부로 적용한다 (기호 : C (CONDITIONAL))
  - 특정 조건이 충족되는 경우에만 반드시 준수해야 하며
    조건이 충족되지 않으면 준수하지 않아도 된다. 적용 조건은 별도로 명시한다.
- 할 수 있다, 쓸 수 있다 (기호 : O (OPTION))
  - 주어진 상황에 따라 선택적으로 적용 가능하다.

4 약어

API	Application Programming Interface
CI	Connecting Information
DI	Duplication Information
EncK	Encryption Key
HMAC	Hash-based Message Authentication Code
HmacK	HMAC Key
HTTP	Hypertext Transfer Protocol
i-PIN	Internet Personal Identification Number
IV	Initial Vector
JSON	JavaScript Object Notation
JWT	JSON Web Token
KDF	Key Derivation Functions
TLS	Transport Layer Security
URL	Uniform Resource Locator
UUID	Universally Unique Identifier

5 본인확인서비스 플로우

5.1 본인확인서비스 인증구조

본인확인서비스의 인증 과정에서 발생할 수 있는 문제점과 그에 대한 개선 방안을 설명한다.

본인확인서비스는 주민등록번호를 사용하지 않고 대체 수단을 통해 이용기관이 이용자를 식별할 수 있도록 제공되는 서비스이다. 이 과정에서 이용자의 개인정보를 보호하기 위해 본인확인기관은 대체 수단을 활용하여 본인확인을 수행하며 모든 중요 데이터는 암호화를 통해 안전하게 전송된다.

현재의 본인확인서비스는 데이터 보호를 위해 본인확인 결과 데이터를 암호화하여 전달하고 있으나 암호화된 데이터를 표준창으로 전달하고 표준창이 이를 다시 이용기관 업무 서버에 전달하는 구조는 잠재적인 취약점을 내포하고 있다. 이러한 구조는 데이터가 안전하게 보호되더라도 통신 과정에서 본인확인기관이나 이용기관이 아닌 제3자(이용자)가 해당 데이터를 확인할 수 있는 구조이기 때문이다.

일반적인 이용자가 자신의 데이터를 확인하는 것은 큰 문제가 되지 않지만 악의적인 의도를 가진 이용자는 본인확인 과정에서 발생하는 데이터를 수집할 수 있으며 만약 암복호화 알고리즘 및 키 데이터가 탈취될 경우 해당 데이터를 변조하여 전송할 수 있다. 중간에 데이터가 변조되었더라도 이용기관이 해당 데이터를 정상적으로 복호화하고 무결성 검증 값이 일치한다고 판단하면 이를 정상 데이터로 처리할 수 있다.
일부 기관에서는 이러한 구조적 문제를 보완하기 위해 다양한 방식으로 개선하여 서비스를 운영 중이다.

이러한 잠재적 위험성이 존재하는 현행 본인확인서비스 인증 구조를 개선하는 방안으로 API 방식을 통한 본인확인서비스에 대한 내용을 설명한다. 또한, 개선된 구조에서 본인확인서비스 제공을 위해 반드시 지켜야 할 조건들을 제시함으로써 보다 본인확인서비스 제공을 목표로 한다.

5.2 API 방식 본인확인 플로우

본 절에서는 API 방식을 통한 본인확인서비스 플로우에 관해 설명한다.

본 서비스 플로우는 이용기관이 이용자가 완료한 본인확인 결과를 확인하고자, 본인확인기관에 인증 결과를 요청하는 본인인증 서비스 절차이다.

아래 인증 절차는 API 방식을 이용하여 본인확인기관을 통해 이용자의 본인확인을 진행

하는 과정을 도식화한 것이다.

(그림 5-1) 본인확인서비스 인증 흐름

① 이용자는 이용기관이 제공하는 본인확인 요청 화면에 접속한다.
② 이용자는 이용기관에서 제공하는 본인확인 수단 중 하나를 선택한다.
③ 이용기관은 본인확인기관으로부터 발급받은 클라이언트 아이디와 클라이언트 시크릿을 사용해 접근토큰을 요청한다.
④ 본인확인기관은 이용기관의 클라이언트 아이디와 시크릿을 검증한 뒤 접근토큰을 발급한다.
⑤ 이용기관은 본인확인 거래에 필요한 데이터를 본인확인기관으로 전송한다.
⑥ 본인확인기관은 수신한 데이터를 검증한 후 본인확인 표준창 URL을 반환한다.
⑦ 이용기관은 본인확인기관에서 전달받은 표준창 URL을 이용자에게 제공한다.
⑧ 이용자는 표준창에 접속하여 개인정보를 입력하고 본인확인을 진행한다.

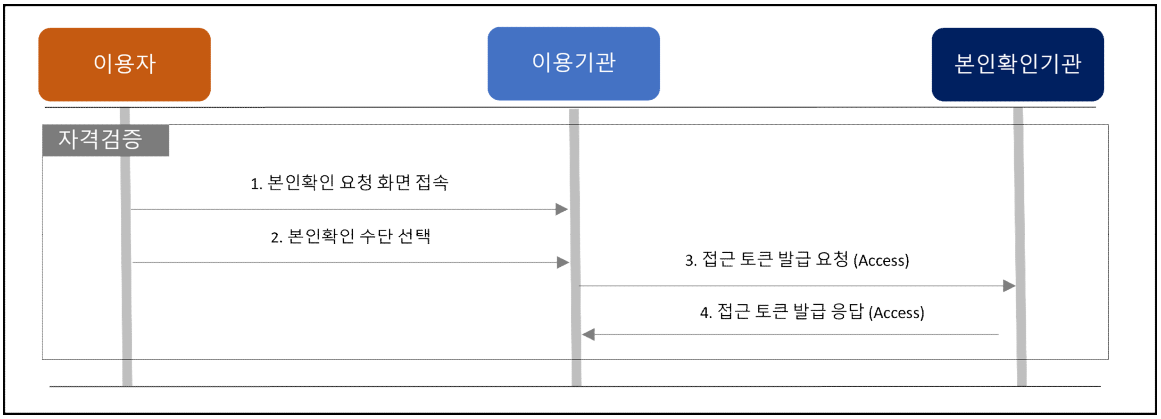
- 트랜잭션 ID 수신 - Type 1 (외부 데이터 접근을 허용하는 경우)
⑨ 본인확인기관은 이용기관에게 트랜잭션 ID를 Callback 방식으로 전달한다.

- 트랜잭션 ID 수신 - Type 2 (외부 데이터 접근을 불허하는 경우)
⑨ 본인확인기관은 트랜잭션 ID를 표준창 내에 포함하여 이용자에게 전달한다.
⑩ 표준창은 해당 트랜잭션 ID를 이용기관에 전달한다.

⑪ 이용기관은 본인확인기관에 트랜잭션 ID와 함께 이용자 본인확인 결과를 요청한다.
⑫ 본인확인기관은 트랜잭션 ID를 검증한 후 본인확인 결과를 암호화하여 제공한다.

위와 같이 이용자가 서비스 사용을 위해 본인확인을 12단계 절차에 따라 진행하는 과정에서 API 및 기타 데이터가 전달되는 과정은 아래와 같다. 아래의 설명은 인증 과정의 이해를 위해 예시로 설명된 내용으로 각 요소에 대한 상세한 설명은 문서 각 항목에 상세 설명한다.

5.2.1 자격검증

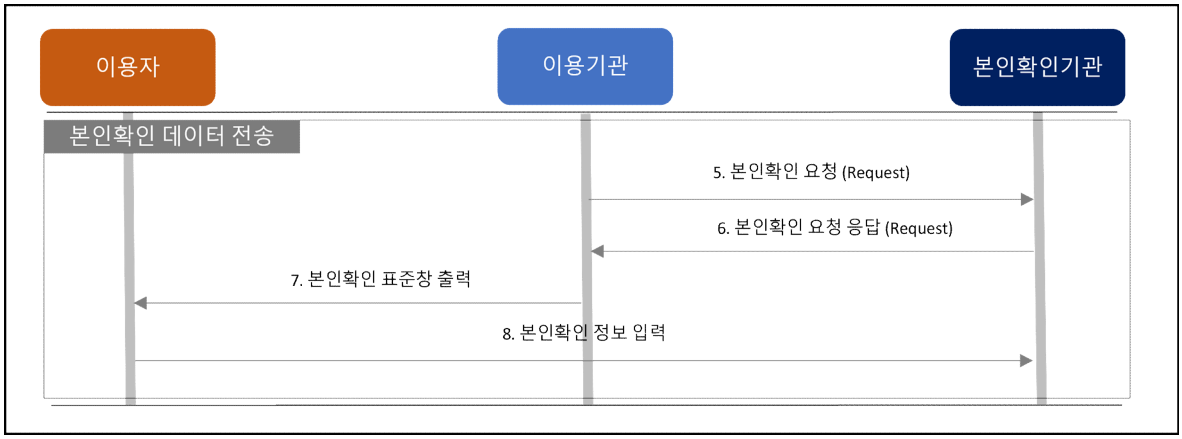


(그림 5-2) 접근토큰 발급

본인확인기관은 이용기관에게 클라이언트 아이디와 클라이언트 시크릿을 발급하며, 해당 정보는 제3자를 거치지 않고 외부에 노출되지 않는 6.2.2의 언급된 방식을 통하여 전달한다.

이를 기반으로, 이용기관은 본인확인기관에게 접근토큰 발급을 요청하여 사전에 전달받은 정보를 제공한다. 본인확인기관은 이를 검증하여 정당한 이용기관임을 확인한 후 접근토큰을 발급한다. 발급된 접근토큰은 API 통신 시에만 사용되며, 외부에 노출되지 않도록 안전하게 관리되어야 한다.

5.2.2 본인확인 데이터 전송



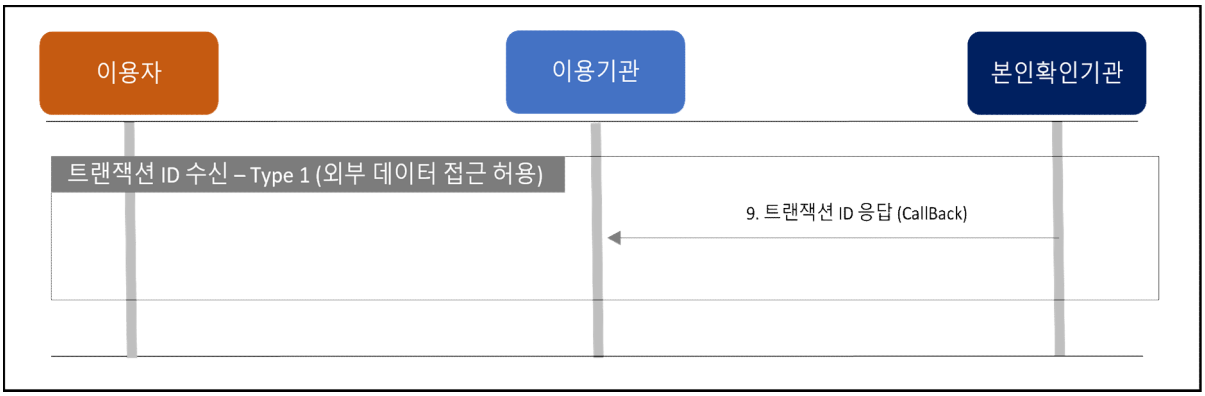
(그림 5-3) 본인확인 데이터 전송 및 표준창 출력

본인확인서비스를 이용자에게 제공하기 전에 안전한 서비스 제공을 위해 이용기관은 본인확인기관으로 사전 데이터(cpCode, 요청 시간, 요청 구분, Callback URL 등)를 요청 API를 통해 전송한다.

본인확인기관은 전달받은 사전 데이터를 바탕으로 트랜잭션을 생성하고 표준창 URL과 함께 해당 트랜잭션 정보를 이용기관에 전달한다.

이용기관은 전달받은 표준창을 이용자에게 제공하여 본인확인서비스를 진행한다.

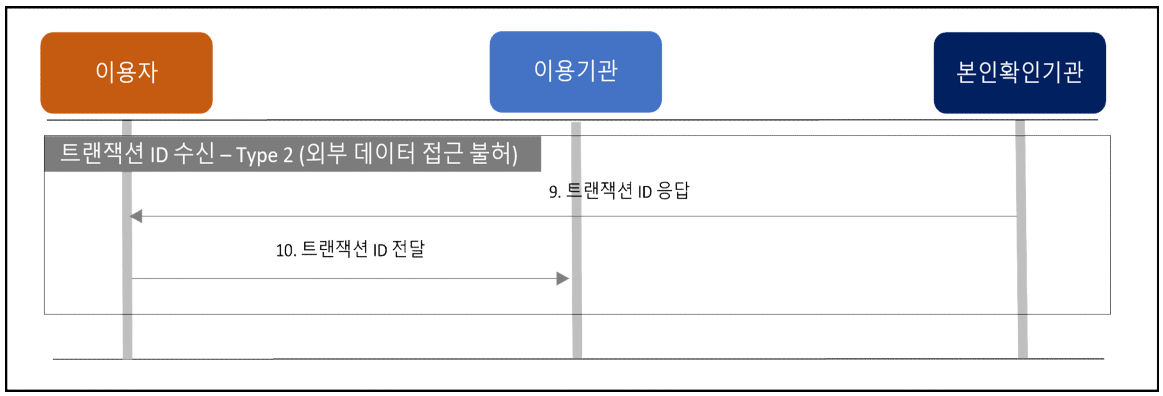
5.2.3 트랜잭션 ID 수신(외부 데이터 접근 허용하는 경우)



(그림 5-4) 트랜잭션 ID 수신 Type1

이용기관의 시스템 환경이 본인확인기관으로부터 데이터를 수신할 수 있는 환경인 경우 본인확인기관은 결과 조회를 위한 트랜잭션 ID를 이용기관에서 요청한 Callback URL로 전달한다.

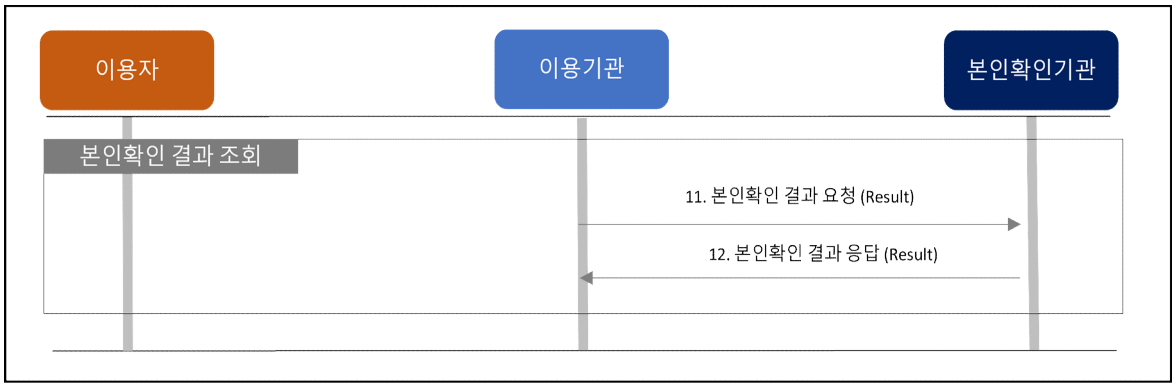
5.2.4 트랜잭션 ID 수신(외부 데이터 접근 불허하는 경우)



(그림 5-5) 트랜잭션 ID 수신 Type2

이용기관의 시스템 환경이 본인확인기관으로부터 데이터를 수신할 수 없는 환경(방화벽 등)인 경우 본인확인기관은 결과 조회를 위한 트랜잭션 ID를 표준창으로 전달하고 표준창은 트랜잭션 ID를 이용기관으로 전달한다.

5.2.5 본인확인 결과 조회



(그림 5-6) 본인확인 결과 조회

본인확인 결과는 이용자가 성공적으로 본인확인을 마친 이후 전달된 고유 값을 이용기관에서 직접 본인확인기관에 직접 통신을 요청하여 직접 결과를 요청하고 받아오는 방식으로 진행한다.

이용자 정보를 이용기관에 전달 시 암호화 데이터, 트랜잭션값을 이용해 암호화하며 무결성 검증을 위한 HMAC 키로 사용되며 각 키 유도방식, 암호 알고리즘에 관한 내용은 7.1에서 상세 설명한다.

6 본인확인서비스 요구사항

본 절에서는 본인확인서비스의 보안성 및 신뢰성을 확보하기 위한 요구사항을 기술한다. 특히, 본인확인기관과 이용기관 간 Server-to-Server 연동 구조에서 발생할 수 있는 보안 위협을 식별하고, 이에 대한 기능별 대응 방안을 제시한다. 또한, API의 호출 흐름과 구성 원칙, 명세 설계 기준을 설명하며, 구현 시 유의사항을 서술한다.
정의된 요구사항은 서비스 연동 전반의 보안성과 일관성 확보를 목적으로 하며, 운영 환경에 따라 추가적인 보안 조치가 필요할 수 있다.

6.1 보안 위협

본 절에서는 본인확인서비스 제공을 위협할 수 있는 보안 위협 요소를 식별하고 정의한다. 본 표준은 공개된 네트워크 환경을 기반으로 하는 Server to Server 연동 구조이므로 네트워크를 통한 접근뿐만 아니라 내부 시스템 설정, 인증 흐름, 암호화 구성 등 다양한 측면에서 보안을 저해할 수 있다. 이에 따라 공격자는 일반적으로 제한된 기술 수준과 자원 및 동기를 가진 외부 또는 내부 실체로 가정하였으며 본 표준은 이들이 취약점을 악용하여 무단 접근, 정보 유출, 데이터 위변조, 인증 우회 등의 행위를 시도할 수 있다는 전제하에 보안 요구사항을 도출한다.

<표 6-1> 보안 위협 및 요구사항

번호	위협명	상세 내용	보안 요구사항	설명 위치
1	가짜 접근토큰	● 정의 : 위조된 접근토큰 생성 및 사용 ● 설명 : OAuth 토큰 구조를 모방하여 위조된 토큰 생성 후 API 접근 시도	접근토큰의 유효성 검증 및 서명 검증	6.3.5, 6.3.7, 7.2
2	만료된 토큰사용	● 정의 : 만료 토큰 사용 ● 설명 : 만료된 토큰의 만료 검증 누락으로 접근 허용	접근토큰의 유효시간 확인 및 만료 처리	6.2.8, 7.2
3	자격증명 노출	● 정의 : 클라이언트 아이디 및 클라이언트 시크릿 노출 ● 설명 : 평문 전송·저장 또는 Header 노출로 인증정보 탈취	자격증명에 대한 암호화 및 안전한 저장	6.2.2, 6.2.1, 6.2.3, 6.2.5.1

번호	위협명	상세 내용	보안 요구사항	설명 위치
4	비인가API 접근	● 정의 : 비인가된 IP 및 API에 대한 접근 ● 설명 : 등록되지 않은 IP에서 본인확인기관 API 호출 시도	IP 화이트리스트 및 접근 토큰을 통한 접근제어	6.2.5.1, 6.3.6
5	평문전송	● 정의 : 민감 정보 평문 전송 ● 설명 : 암호화 없이 Header 또는 Body에 민감 정보 포함	암호화 전송 및 API 호출 시 TLS 보안 채널 사용	6.2.4, 6.2.5.2
6	무결성위조	● 정의 : 메시지 무결성 우회 ● 설명 : HMAC 위조로 위변조된 메시지를 무결한 것으로 처리	HMAC 기반 무결성 검증	6.2.5.3, 7.1.4
7	데이터변조	● 정의 : 전송 데이터 위변조 ● 설명 : 암호화되지 않은 본문이 변조되어 수신자 오작동 유발	AES256 암호화, TLS 보호	6.2.4, 6.2.5.2, 7.1.3
8	암호화 알고리즘 취약점	● 정의 : 암호 알고리즘의 보안 수명 초과 ● 설명 : 구식 또는 취약 알고리즘 사용 시 해독 위험	안전한 표준 암호 알고리즘 사용	6.2.5.2
9	트랜잭션ID 재사용	● 정의 : 트랜잭션 ID 재사용 ● 설명 : 트랜잭션 ID의 유효기간 초과 후 재요청하여 결과 반복 조회	트랜잭션 ID 유일성 보장, 만료 정책 적용	6.2.6, 7.1.5
10	결과재요청	● 정의 : 결과 반복 요청 ● 설명 : 동일 트랜잭션 ID으로 반복 조회하여 개인정보 수집	결과 1회 발급 및 응답 만료 정책 적용	7.1.5
11	인증우회	● 정의 : 인증 절차 우회 또는 변조 ● 설명 : 인증 절차를 거치지 않거나 변조하여 결과 요청 시도	트랜잭션 완료 여부 확인, 재요청 차단	7.1.5, 7.4

번호	위협명	상세 내용	보안 요구사항	설명 위치
12	세션 하이재킹	● 정의 : 인증 중 세션 탈취 ● 설명 : 접근토큰 또는 트랜잭션 ID 탈취로 결과 요청 위장	접근토큰 만료, HTTPS, 재사용 방지	6.2.4, 6.2.6, 6.2.8
13	키유도공격	● 정의 : 키 유도 함수의 파라미터 노출 ● 설명 : ticket, 트랜잭션 ID가 예측 가능 시 KDF 유출	무작위 ticket 생성, UUID 기반 트랜잭션 ID 사용	6.2.6, 6.2.7, 7.1.2
14	시스템 시간조작	● 정의 : 트랜잭션 만료 우회 ● 설명 : 시스템 시간을 조작하여 트랜잭션 ID 만료 우회 시도	서버기반 시간검증, 유효시간 엄격 적용	6.2.6
15	권한초과 scope	● 정의 : 정의되지 않은 scope 접근 ● 설명 : scope 외 인증수단 요청하여 서비스 오남용	scope 검증 및 계약된 서비스만 허용	6.3.7
16	암호화키 재사용	● 정의 : 키 재사용에 따른 정보유출 ● 설명 : 동일 ticket 사용 시 복호화 키 유추 가능	데이터 암호화 시 무작위성 보장	6.2.7, 6.2.8

6.2 주요 기능별 보안대책

본 절에서는 6.1에서 식별된 보안 위협에 대응하기 위해 본인확인서비스에 적용되어야 할 주요 기능별 보안대책을 정의한다. 각 기능에 대한 요구사항은 관련 표준 및 기술 가이드라인을 근거로 도출하였다. 기반이 된 연관 표준 및 기술 가이드라인은 부록 II를 참고하도록 한다.
각각의 기능에 대한 세부 내용은 이후 항목에서 설명한다.

6.2.1 키 종류

- 클라이언트 아이디, 클라이언트 시크릿
  RFC 6749에 따라, 클라이언트 아이디와 클라이언트 시크릿은 API를 사용하는 이용기관을 식별하기 위해 접근토큰 발급 시 사용되는 고유한 인증 정보이다. 이 값들은 본인확인기관에서 자체적으로 생성하며 생성 시 각 이용기관의 cpCode와 같은 고유 식

별 정보를 포함하여 다른 이용기관과 중복되지 않는 유일한 값으로 구성되어야 한다. 각 이용기관이 사용할 클라이언트 아이디와 클라이언트 시크릿의 구성 방식, 문자열 길이, 형태 등은 본인확인기관이 자율적으로 정할 수 있다. 단, 생성된 값은 다음과 같은 조건을 충족해야 한다.

- 쉽게 유추할 수 없는 복잡한 값일 것
- 중복되지 않을 것
- RFC 4648에 따라 정의된 Base64로 인코딩 시에 데이터의 손상, 깨짐, 손실, 불일치 등의 문제가 발생하지 않을 것

6.2.2 키 분배 방식

키는 본인확인기관에서 이용기관에 유일한 채널을 통해 안전하게 전달되어야 한다.
클라이언트 시크릿은 본인확인기관에서 생성하며 제 3자에게 공유되거나 노출되어서는 안 된다.

※ “유일한 채널로 안전하게 배포한다”는 것은 계약된 본인확인기관과 이용기관 간의 채널을 제외한 제 3자가 데이터를 조회, 수집, 탈취할 수 없는 방식으로 전달한다는 의미한다.

6.2.3 키 사용

본인확인기관의 API 사용 권한을 얻기 위해 클라이언트(이용기관)는 사전에 본인확인기관으로부터 6.2.2의 언급된 방식에 의해 공유받은 인증 키(클라이언트 아이디 및 클라이언트 시크릿)를 사용해 접근토큰을 발급받는다.

이때 클라이언트 시크릿은 약속된 방식에 따라 변환되어 본인확인기관에 전달되며 본인확인기관은 이를 검증한 후 유효한 접근토큰을 발급한다.

- 접근토큰
  접근토큰에는 본인확인 과정 중 민감 정보 보호 및 통신 메시지의 무결성을 위한 ticket 값이 포함된다.
  ticket은 각 거래마다 새로운 암호키를 생성하기 위한 데이터이며 접근토큰 갱신 시 함께 갱신되어야 한다.

- 세부 정보

<표 6-2> 키 종류별 세부 정보

키 종류	생성 / 배포대상	용도	유효기간
클라이언트 시크릿 키	본인확인기관 / 이용기관	API 사용 및 이용자 식별을 위해 필요한 접근토큰을 발급하는 용도로 사용한다.	-

키 종류	생성 / 배포대상	용도	유효기간
접근토큰	본인확인기관 / 이용기관	API 사용승인 및 암·복호화 및 무결성 검증을 위한 키 생성 시 사용된다.	최대 1일

6.2.4 통신 구간 보호

본인확인기관, 이용기관, 이용자나 이용자 간의 모든 통신 구간은 TLS 1.2 이상 보안 통신을 수행해야 한다(국가 보안요구사항 및 NIST SP 800-52 Rev.2).

※ 국가 보안요구사항(2024.04)에 따르면, 안전한 암호통신을 위해서 표준 프로토콜을 사용하여 기밀성과 무결성을 제공해야 한다. (안전한 암호통신 프로토콜은 HTTPS(TLS을 이용하여 구현), TLS(TLS 1.2 RFC 5246 이상), SSH(SSH V2 - RFC 4251, 4254) 등이 있다.)

※ 미국 국립표준기술연구소(NIST)의 NIST SP 800-52 Rev.2(2019)에 따르면, 최소 TLS 1.2를 사용해야하며 2024년 1월 1일까지 TLS 1.3을 지원해야 한다. 비정부 시스템과 상호 운용성이 필요한 경우 TLS 1.1 및 TLS 1.0이 지원될 수도 있다. (In particular, it requires that TLS 1.2 be configured with cipher suites using NISTapproved schemes and algorithms as the minimum appropriate secure transport protocol andrequires support for TLS 1.3 by January 1, 2024.1 When interoperability with non-governmentsystems is required, TLS 1.1 and TLS 1.0 may be supported.)

6.2.5 데이터 보호

안전한 API 사용 인증을 위해, 본인확인기관은 API 요청을 수행하는 클라이언트가 사전에 등록된 이용기관인지 여부를 반드시 검증해야 한다.

기본적인 접근 통제는 이용기관의 IP와 발급된 접근토큰 확인을 통해 이루어진다.
이외의 추가적인 접근 통제 수단은 각 운영기관의 정책에 따라 자율적으로 강화할 수 있다.

6.2.5.1 API 접근통제

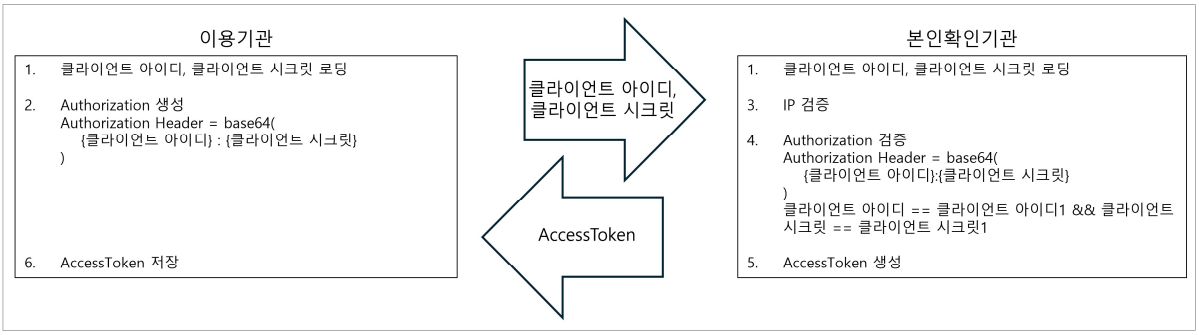
- 본인확인기관 접근 통제
  본인확인기관은 API 서버에 접근할 수 있는 이용기관의 IP를 사전에 등록하고, 등록된 IP만 접근 가능하도록 설정해야 한다. 이는 접근토큰이 외부에 탈취되었을 경우에도, IP 기반 이중 검증을 통해 불법 접근을 방지하기 위함이다.

- 불필요한 접근 차단
  본인확인기관은 API 서버에 대해 본인확인서비스 수행과 무관한 이용자 또는 시스템의 접근을 차단해야 한다.

필요한 관계자 외의 모든 접근은 원천적으로 제한되어야 한다.

- 이용기관 보안 준수 사항
  - 이용기관은 본인확인기관으로부터 사전 공유받은 클라이언트 시크릿 및 암호화 토큰을 평문으로 전송해서는 안 된다.
  - 특히 Header 또는 Body에 평문 형태로 포함되어서는 안 되며 반드시 보안 처리된 방식으로 API 호출이 이루어져야 한다.
  - API 요청 시, 이용기관은 발급받은 접근토큰을 HTTP Header에 포함하여 전송해야 하며 본인확인기관은 이를 통해 정상적인 이용기관 여부를 검증한다.

아래 그림의 경우 클라이언트 아이디, 클라이언트 시크릿을 이용해 접근토큰 발급을 요청 및 발급하는 과정이다.



(그림 6-1) 접근토큰 발급

6.2.5.2 중요 정보 암호화

다음과 같은 방법으로 매 통신 시 다른 키를 생성하는 방법을 제안하며 암호 데이터 상호 운용성 확보를 위해 키 유도 및 암호 알고리즘을 7.1.3와 같이 정의한다. 사용된 암호 알고리즘의 강도는 국내외 표준과 암호 알고리즘 보안 수명 기간을 고려하여 선정하였다.

- 보안강도

  장기적인 보안성과 운용을 고려하여, 256bit 이상의 보안강도를 보장하는 표준 알고리즘의 사용을 권고한다(국가 보안요구사항, NIST SP 800-131A Rev.2 및 NIST SP 800-57 Part 1. Rev.5).

※ 보안강도(Security strength)는 암호화된 데이터를 해독하기 위해 수행해야 하는 연산량(작업 복잡도)를 나타내며 비트(bit)로 표현한다. 이는 알고리즘이 제공하는 암호학적 안전성 수준을 정량적으로 나타내는 지표로 사용된다.

※ 국가 보안요구사항 (2024.04), 과학기술정보통신부·KISA 암호 알고리즘 및 키 길이 안내서 (2018.12)에 따르면, 2030년 이후부터는 보안강도가 128bit 이상인 표준 알고리즘을 사용하도록 권고한다. 중장기적으로는 196bit 또는 256bit 수준의 강도 확보가 가능한 알고리즘 채택이 요구된다.

※ 미국 국립표준기술연구소(NIST)의 NIST SP 800-131A Rev.2 (2019) 및 NIST SP 800-57 Part 1 Rev.5 (2020)에 따르면, 암호 알고리즘 및 키 길이의 보안강도는 적용 시기에 따라 상이하게 요구되며, 2031년부터는 최소 128bit 이상의 보안 강도를 만족해야한다고 명시되어 있다. (Table 4 provides a projected time frame for applying cryptographic protection at a minimum security strength (e.g., at least 128 bits in 2031).)

- 민감 정보 암호화 적용 기준
  - 본인확인서비스의 통신 과정에서 발생하는 민감 정보 및 보안이 요구되는 데이터는 반드시 암호화되어야 한다.
  - 암호화에는 보안강도 256bit 이상을 충족하는 안전한 표준 암호 알고리즘을 사용해야 한다. 이때, 안전한 표준 암호 알고리즘의 세부사항은 암호 알고리즘 및 키 길이 이용 안내서, 소프트웨어 암호모듈 검증기준, NIST SP 800-131A Rev.2 및 NIST SP 800-57 Part 1 Rev.5를 참고한다.
  - 암호화에 사용되는 키는 유효기간 만료 전 반드시 갱신되어야 하며 재사용을 방지해야 한다.

- 키 유도 방식
  - 암호화를 위한 키는 다음 정보를 기반으로 유도(KDF)된다:
    1. 접근 토큰에 저장된 암호화 데이터
    2. 본인확인 요청 시 생성되는 고유 트랜잭션 값
  - 해당 데이터를 이용하여 키(Key)와 IV(Initialization Vector)를 생성하고, 이를 통해 민감 정보를 암호화한다.
  - 생성된 키는 거래별로 서로 다른 키여야 하며 한 건의 키가 다른 요청에 영향을 미쳐서는 안 된다.

- 암호화 적용 시점

  개인정보보호법에 따라, 정보통신서비스 제공자 등은 전송 시 개인정보, 인증정보 등과 같은 기타 보안이 필요한 데이터를 암호화해야 한다. 따라서, 아래의 시점에 암호화를 적용해야 한다(개인정보보호위원회·KISA 개인정보의 암호화 조치 안내서 및 NIST SP 800-52 Rev.2).
  - 본인확인 결과 요청 시
  - 본인확인 결과 응답 시
  - 기타 보안이 필요한 데이터 전송 시

※ 개인정보보호위원회·KISA의 개인정보의 암호화 조치 안내서(2020.12)에 따르면, 개인정보처리자 및 정보통신서비스 제공자 등을 대상으로, 암호화 적용 대상과 적용 시점에 대해 명시되어 있다. (개인정보보호법에 따라 암호화 대상 개인정보를 저장·전송하는 개인정보처리자 및 정보통신서비스 제공자등을 대상으로 한다.)

※ NIST SP 800-52 Rev.2(2019)에 따르면, 민감 정보 등과 같은 데이터를 다룰 때 이를 적절하게 보호해야 한다고 명시되어 있다. (Any network service that handles

sensitive or valuable data, whether it is personally identifiable information(PII), financial data, or login information, needs to adequately protect that data.)

- 알고리즘의 보안 수명 기간
  알고리즘 보안 수명 기간(Algorithm security lifetime)은 키가 손상되지 않았다는 가정 하에 특정 암호 알고리즘으로 보호되는 데이터가 안전하게 유지되는 예상기간을 뜻하는 개념이다. 알고리즘의 보안 수명 이후에도 데이터 보호가 필요한 경우, 해당 알고리즘을 암호화에 사용해서는 안 된다(NIST SP 800-57 Part 1 Rev.5).

※ NIST SP 800-57 Part 1 Rev.5(2020)에 따르면, 알고리즘 보안 수명기간(Algorithm security lifetime), 알고리즘 송신자 사용기간(Algorithm originator-usage period), 데이터 보안 수명(Security life of data)가 명시되어 있다. (The estimated time period during which data protected by a specific cryptographic algorithm(and key size) remains secure is called the algorithm security lifetime. During this time, the algorithm may be used to both apply cryptographic protection (e.g., encrypt data) and process the protected information (e.g., decrypt data), although the period allowed for applying protection (the originator-usage period) colud be shorter than the algorithm security lifetime.)

6.2.5.3 메시지 무결성

API 상호 운용성 확보를 위해 키 유도 및 MAC 검증알고리즘을 7.1.4과 같이 정의한다.

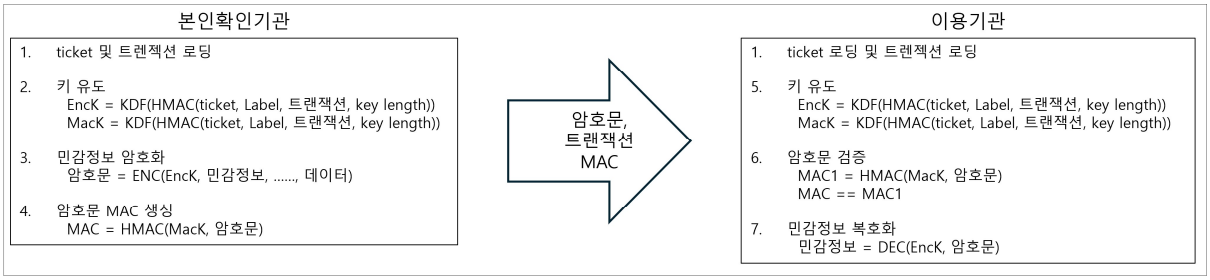
본인확인기관과 이용기관 사이에 발행하는 송/수신 메시지 또는 민감 정보의 무결성 확보를 위하여 메시지 또는 민감 정보에 대한 무결성 검증 값 (HMAC)을 추가한다.

본인확인 완료 후 이용자의 본인확인 결과 데이터 응답 시 무결성 검증을 위해 접근토큰을 통해 발급한 암호화 데이터와 본인확인 요청 시마다 생성되는 트랜잭션을 이용해 HMAC 해시값을 생성 후 전송해야 한다. 이때 생성한 HMAC 해시값은 암호화 데이터와 함께 전송한다.

HMAC 해시값을 생성 및 검증하기 위한 비밀키는 각 기관 간 사전 공유된 암호화 데이터값과 트랜잭션을 조합하여 사용한다. 수신자는 사전에 약속된 무결성 검증 방법을 사용하여 수신한 메시지에 대한 무결성 검증을 수행한다.

메시지 및 민감 정보 무결성 보장을 위해 SHA-2 이상 해시, AES 256bit 이상으로 암호화하여 전송하며 전송된 데이터 무결성 검증을 진행할 수 있어야 한다.

아래 그림의 경우 민감 정보 보호 및 메시지 무결성을 위한 키 유도 및 HMAC 해시 검증알고리즘을 정의한다.



(그림 6-2) 민감 정보 보호와 메시지 무결성 확보

6.2.6 트랜잭션 보호

트랜잭션 ID는 각 이용기관에서 본인확인 요청이 이루어질 때, 본인확인기관이 생성하는 고유 식별 값이다.
이 값은 인증이 진행되는 각 거래를 식별하기 위한 고유한 ID로 사용되며 거래 단위의 추적과 검증에 활용된다.

- 생성 방식
  트랜잭션 ID는 반드시 중복되지 않는 고유한 값으로 생성되어야 한다.
  생성 방식은 각 본인확인기관의 자율에 따르되, 다음과 같은 조합을 활용하는 것이 일반적이다.

  예시)
  cpCode + 요청시간 + 서버시간을 기반으로 한 UUID 생성
  cpCode + 요청시간 + 서버시간을 기반으로 한 고유한 문자열 생성

- 요구사항
  - 유일성 보장
    각 본인확인 요청에 대해 하나의 고유 트랜잭션 ID만 생성되어야 한다.
  - 재사용 방지 및 만료 처리
    1. 인증 정보의 재사용을 방지하기 위해, 트랜잭션 ID는 생성된 후 최대 10분까지만 유효해야 한다.
    2. 10분이 경과된 트랜잭션 ID는 만료된 것으로 간주하며 해당 트랜잭션을 통한 결과 요청은 처리하지 않는다.
  - 환경 고려
    시스템 시간 차이, 통신 지연 등 시스템 환경 요소를 고려한 유효 시간 관리가 필요하다.

6.2.7 암호화 데이터

암호화 데이터는 접근토큰 구성요소에 포함된 데이터로 암·복호화 및 무결성 검사 시 사용되는 키(80Byte), IV 값을 생성하는 데 필요한 데이터이다.

- 암호화 데이터는 다른 이용기관과 중복되지 않는 고유의 문자열이며 생성 방식은 각 본인확인기관의 자체적인 구현 방식에 따라 생성한다.
- 각 본인확인기관에 의해 생성된 암호화 데이터는 각 본인확인기관에서만 사용이 가능하며, 무작위성을 보장하기 위해 안전한 난수 생성기를 사용하여 고유한 문자열을 생성해야 한다.
- 암호화 데이터는 접근토큰에 종속되어 있으며 토큰 갱신 시 암호화 데이터 또한 갱신하여 암호키 노출에 대한 위험성을 최소화한다.

6.2.8 암호키 교체 시나리오

본인확인서비스는 연중무휴(24/365)로 이용자에게 제공되어야 하는 대국민 서비스이다. 이때 접근토큰은 만료 기간이 존재하므로, 만료되기 전에 토큰을 갱신해야 한다.

접근토큰의 갱신은 실시간으로 이루어지며 본인확인서비스가 진행되는 도중 토큰이 갱신되면 기존의 접근토큰은 자동으로 만료된다. 이 경우, 이미 진행 중인 서비스에서 토큰이 변경되면 세션이 끊기거나 서비스가 중단되어, 이용자가 처음부터 다시 본인확인을 진행해야 하는 문제가 발생할 수 있다.

이러한 문제를 방지하기 위해, 본인확인 결과 응답 시 트랜잭션 ID를 함께 포함하여 제공한다. 이를 통해 이용기관은 본인확인 요청 당시 사용된 암호화 키 정보(토큰이 유효하던 시점)를 기준으로 데이터를 복호화할 수 있도록 한다.

예시) 키 교체 시나리오
- 11:59 : 이용자 A가 본인확인을 완료하고, 트랜잭션 ID(Tx_id_ABC123)를 이용기관에 전달함
- 11:59 : 본인확인기관이 트랜잭션 ID(Tx_id_ABC123)를 기반으로 본인확인 데이터를 암호화
- 00:00 : 이용기관에서 접근토큰이 자동 갱신됨 (토큰 정보 및 키 변경)
- 00:01 : 이용기관이 트랜잭션 ID(Tx_id_ABC123)로 결과 조회 API를 호출
  본인확인기관이 트랜잭션 ID(Tx_id_ABC123)와 함께 암호화된 본인확인 결과를 전송
  이용기관은 트랜잭션 ID(Tx_id_ABC123)를 기반으로 이전 토큰 시점의 키를 복원하여 정상적으로 복호화 수행

6.2.9 암호화 모듈

민감 정보 및 암호화 데이터를 사용하는 본인확인기관 또는 이용기관의 암호화 모듈은 민감 정보의 위·변조, 노출, 탈취 등에 대한 위험 대응을 위한 보다 높은 보안성을 통해 보호해야 한다.

6.3 표준 API 구성

본인확인서비스를 위해 본인확인기관은 표준 API를 통해 이용기관 등 본인확인서비스 관계자와 통신한다.

6.3.1 API 형식

본 절에서는 API 통신 시 사용되는 HTTP 및 JSON과 같은 기술 표준 및 프로토콜을 기반으로 하는 웹서비스 API를 정의한다.

서비스 API는 보편화된 HTTP/1.1 이상을 사용하여 API 통신을 진행한다. HTTP 2와 HTTP 3의 경우 본인확인서비스 표준 API에 사용할 수 있지만, 서비스를 사용하는 이용기관과 서비스를 제공하는 본인확인기관의 특성상 보편화된 방식을 활용하여 많은 기관과 이용자에게 서비스를 제공해야 하므로 HTTP/1.1 버전을 사용하는 것을 권장한다.

6.3.2 본인확인서비스 Base URL 및 표준 API 명

서비스 URL은 기관별/표준으로 구성되며 각각은 아래와 같이 구성한다.

<표 6-3> 서비스 URL 구성 정보

요약	상세	비고
서비스 URL	{기관 URL}/{표준 API URL}	
기관 URL	기관도메인/{API 이름}/{기관 API 버전}	기관별 정의
표준 API URL	/ident/{표준 버전}/{서비스명}	
표준 버전	v1.0	
서비스명	access	접근토큰 발급
	request	본인확인 요청
	result	본인확인 결과요청

6.3.3 데이터 타입

<표 6-4> 데이터 타입

타입	포맷	비고
string	문자열	UTF-8(문자열 인코딩)
number	숫자	
Boolean	true or false	
time	unix time	Unix Time stamp
date	yyyyMMddHHmmss	string으로 표현 예시)20230908043932

6.3.4 HTTP 헤더

RFC 8259에 따라, 표준 API 사용 시 적용되는 HTTP 요청 및 응답 헤더의 형식은 다음과 같다.

<표 6-5> HTTP 헤더

필드명	데이터	설명
Content-Type	application/JSON;charset=utf-8	기본 정보
Authorization	access_token	이용기관이 발급받은 API 접근토큰

6.3.5 API 사용 승인

이용기관이 API를 사용하기 위해 이용기관 인증방식은 자격증명 확인 후 본인확인기관으로부터 접근토큰을 발급받아 이용 권한을 검증하는 방식으로 상세 내용은 아래와 같다.

- 접근토큰은 본인확인기관이 이용기관에 발급한 클라이언트 아이디, 클라이언트 시크릿을 본인확인기관에서 검증한 후 서버가 발급하는 토큰을 나타낸다.
- 이용기관이 본인확인기관에 API를 요청하는 경우 사용한다.
- 본인확인기관은 발급하는 접근토큰의 사용 기간을 “최대 1일” 이내로 설정할 것을 권고한다.
- 이용기관은 접속 IP를 사전에 본인확인기관 시스템에 등록하여 지정된 이용기관이 아닌 이용자의 접근을 차단한다.
- 표준 API 사용승인 방법은 ~/ident/version/access API를 통해 발급받은 접근토큰을 API 통신 시 Header에 사용하여 다른 API의 사용승인을 얻는 방식이다.

- ~/ident/version/access API를 통한 인증을 위해 이용기관은 사전에 약속된 자격증명 방법을 사용한다.
- 발급받은 접근토큰은 안전하게 관리해야 하며 접근토큰의 유효기간 동안 사용할 수 있다.
- 접근토큰의 갱신은 유효기간이 만료하기 전에 사전에 갱신해야 한다.
  예시) 1일의 경우 만료 1시간 전, 1시간의 경우 만료 10분 전 등.
- 발급받은 접근토큰은 API 사용 시에만 사용해야 한다.

```
접근토큰 인증

예시)
[Request]
 POST /request
 Content-Type: application/JSON; charset=UTF-8
 Authorization: {access_token}
{
 …
}
```

6.3.6 클라이언트 자격증명

클라이언트 인증을 통해 접근토큰을 발급받는 방법으로 Header에 클라이언트 아이디와 클라이언트 시크릿을 담은 Basic Base64{client id:client secret} 형식의 값을, JSON 파라미터로 client_credentials을 전송한다. 기본적으로 클라이언트는 접근토큰을 API 사용 시 헤더에 포함하여 전송하여 접근토큰으로 인가된 이용기관임을 증명한다.

```
클라이언트 인증

예시)
[Request]
 POST ~/ident/version/access
 Content-type: application/JSON; charset=UTF-8
 Authorization: Basic base64({client id}:{client secret})
…
{
 “grant_type“: ”{client_credentials}”
}

[Response]
 HTTP/1.1 200 OK
 Status: 200 OK
 Content-Type: application/JSON; charset=utf-8
```

```
...
{
 “code”: 200,
 “message”: “발급완료”,
 “access_token”: “{접근토큰}”,
 “expires_in”: {time},
 “token_type”: “Bearer”
}
```

6.3.7 접근토큰 구성

클라이언트 자격증명 시 사용되는 접근토큰의 구성은 다음과 같다.

```
접근토큰 구성

예시)
JWT.builder(
   useOrganization : {이용기관},
   scope : [{계약된 서비스 리스트}],
   ticket : {암·복호화에 사용되는 데이터},
   iat : {접근토큰 생성 시간},
   exp : {접근토큰 만료 시간}
   {etc} : {기타 각 기관에서 추가할 정보}
)SignatureAlgorithm
```

발급된 접근토큰은 JWT(JSON Web Token) 형식으로 생성되며 해당 토큰 내에는 만료 시간을 포함한 다양한 정보가 Map 형식으로 포함한다. 이를 통해 토큰을 소지한 경우, 만료 기간을 확인하고 만료된 토큰의 재발급을 요청할 수 있다. JWT는 이용기관, 발급기관 등의 기본 정보를 포함하여 구성되며 필요시 각 기관에서 필요한 정보를 추가로 포함할 수 있다.

본인확인서비스 부정 사용을 방지하기 위해 접근토큰 생성 시 scope 값에 계약된 본인확인서비스 코드를 포함하며 scope 값을 검증해 허용된 본인확인서비스만 제공해야 한다. 추가로 각 기관에서 토큰에 추가로 담아야 하는 정보는 각 기관의 요구에 따라 토큰에 데이터를 추가할 수 있다.

접근토큰에 포함된 scope별 본인확인서비스 허용 권한은 다음과 같다.

<표 6-6> scope별 본인확인서비스 허용 권한

scope 값	설명
I	i-PIN 본인확인서비스가 이용기관과 계약된 경우
M	휴대폰 본인확인서비스가 이용기관과 계약된 경우
C	카드 본인확인서비스가 이용기관과 계약된 경우
S	공동 인증서 본인확인서비스가 이용기관과 계약된 경우
F	금융 인증서 본인확인서비스가 이용기관과 계약된 경우
A	모바일 인증서 본인확인서비스가 이용기관과 계약된 경우
여러 개의 본인확인서비스를 제공하는 경우 예시) [I, M, C, S, F, A]	
단일 본인확인서비스를 제공하는 경우 예시) [I]	

6.3.8 암호화 데이터 발급

본 절에서 제시하는 암호화 데이터는 메시지 암·복호화 시 필요한 키를 유도하기 위해 발급받는 데이터를 나타낸다. 암호화 데이터는 접근토큰 내부에 있는 ticket 필드에 저장되며 각 데이터는 이용기관별로 겹치지 않는 고유의 데이터를 생성하여 발급한다.

발급된 데이터는 키 유도를 위해 필요한 암호 키로 활용되며 접근토큰이 재발급될 경우 암호화 토큰 또한 기존의 데이터와 다른 새로운 암호화 데이터를 생성하여 접근토큰 내부에 포함되어야 한다.

6.4 구현 시 주의사항

본 표준은 본인확인서비스 구현을 위한 표준 API 규격을 제시하며 6.1 및 6.2의 보안위협과 주요 기능별 보안 대책은 일반적인 보안 수준과 시스템 구성 환경을 고려하였다. 실제 시스템에 본 표준을 적용하는 경우 적용 환경의 특성, 정책적 요건 등을 충분히 고려하여야 하며 실제 구현 및 적용 환경에 따라 별도의 보안 검토 및 강화가 필요할 수 있다.

7 본인확인서비스 API 명세

7.1 API 공통

7.1.1 필요 데이터

- 이용기관
- 본인확인기관별 발급받은 자격증명 정보 : 클라이언트 아이디, 클라이언트 시크릿

7.1.2 HMAC 기반 키 유도 함수

본 절에서는 본인확인이 발생할 때마다 사용되는 암호화 키, IV 값을 유도하는 방식에 관해 설명한다.

암호키 유도방식은 TTAK.KO-12.0334, 패스워드 기반 키 유도함수(Part2 해시 함수 SHA-2)를 준수한 방식으로 키를 유도한다. 해당 표준은 국제표준 ISO/IEC 10118-3을 인용한 방식이다.

* secret인 암호화 데이터는 JWT 특성상 base64 인코딩되어 있으므로, 반드시 base64 디코딩하여 사용해야 한다.

- HMAC-SHA-256 기반의 키 유도 함수

KDF(Key, Label, Context, Key length)

- Context : 매 거래 요청 시 본인확인기관에서 생성한 트랜잭션 값 사용
- key : 본인확인기관 API 서버에서 발급받은 암호화 데이터 사용
- Label : 키 생성 시, 생성 목적 식별용으로 사용
- Key length : 80Byte
- 생성 알고리즘 운영 모드
  - 카운터 모드(CTR)
  - Counter는 키 길이에 따라 1부터 N까지
- 암·복호화 키 : 32byte(생성된 데이터 왼쪽부터 32byte)
- IV : 16byte(암·복호화 키 이후 16byte)
- 무결성 키 : 32byte(IV 값 이후 32byte)
- 해시 알고리즘 : HMAC-SHA256

예시)

```
key(ticket) = 03e550b77bf43da54170be12672822cd18e44b47bf3fbd0eb3
context(tx_id) = A001.03dc7e22-0573-421f-96c0-fd1d5b869e1b
label = keycreate
```

key length(출력 키 길이) = 80byte (구현 시 16진수 비트열로 표현)

1. Init

- Key와 Context 값 설정
  - Key = “962abde1035d8ffd498d669a63c951841923f7062c1f8bcca8cccbaefdbb8e40”
  - Context = “A001.cad800ed-40e1-4876-a16a-177676d0d83a”
  - label = “keycreate”
  - key length = 0x0280

2. 1R HMAC 출력

- Input data = (byte array)[Label+0x00+Context+Key length]
  - input data =₩x6b₩x65₩x79₩x63₩x72₩x65₩x61₩x74₩x65₩x00...₩x02₩x80
  - date = counter(₩x01) + input data

  * 입력값의 각 변수(key, context, label, key length)가 설정에 따라 문자열 혹은 hexstring이 섞여 있을 수 있다. String의 경우 hexstring으로 변경해야 하며 이후 hexstring을 byte 형식으로 변환이 필요하다.

  가이드에 제공된 데이터의 형식과 변환되는 예시는 다음과 같다.

  Key = hexString
  Label = String
  0x00 = Byte
  Context = String
  Key length = hexString
  제공된 데이터는 String, hexString, Byte가 혼용된 형식이다.
  최종 입력 데이터를 만들기 위해 각 데이터를 Byte 형식으로 변환과정이 필요하다.
  먼저 String의 경우 hexString으로 변환하면 다음과 같다.

  Label : keycreate -> 6b6579637265617465
  Context : A001.cad800ed-40e1-4876-a16a-177676d0d83a
  -> 413030312e6361643830306564342d343065312d343837362d613136612d313737363736643064383361
  마지막으로 hexString은 Byte형식으로 변환 뒤 각 데이터를 합치는 과정을 거치게 된다.

  Label : 6b6579637265617465 -> ₩x6b₩x65₩x79₩x63₩x72₩x65₩x61₩x74₩x65
  Context의 경우 Label과 동일한 방식으로 Byte 형식으로 변환과정을 수행한다.
  최종적으로 생성된 bytearray 형식의 input data 앞에 1라운드를 나타내는 counter ₩x01을 추가하여 data를 생성한다.

- 1R HMAC 결과
  - Hmac-sha-256(key, data)

- 결과 : 725c59018693ef0b87bf4a1fdb8ec6224168bb07358b2a562d5dad724b585ecf

3. 2R HMAC 출력

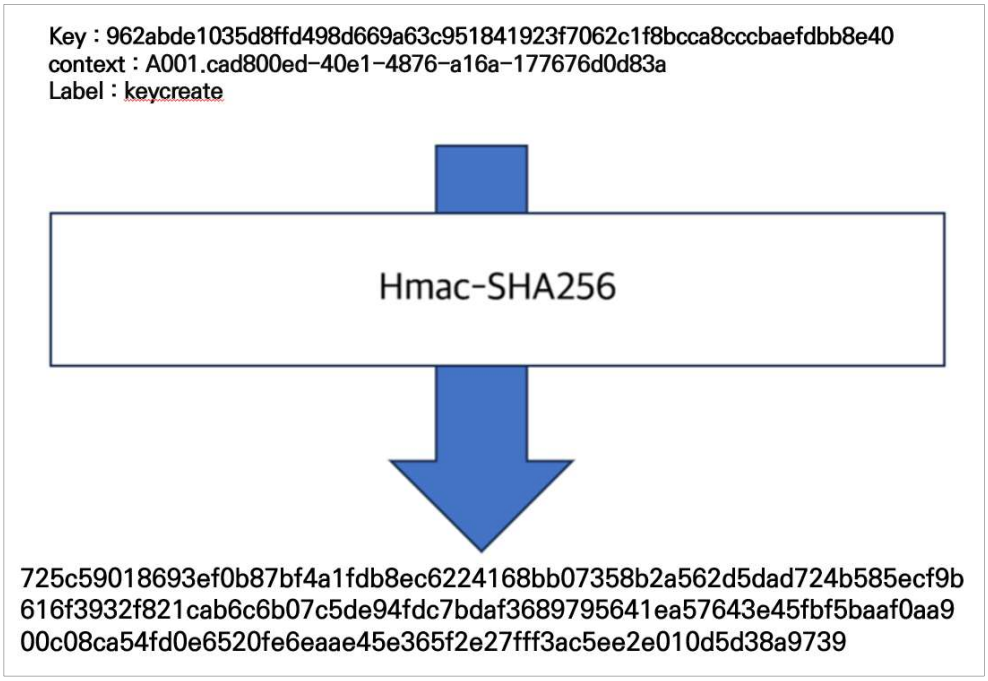
- Input data = (byte array)[Label+0x00+context+key length]
  - input data = ₩x6b₩x65₩x79₩x63₩x72₩x65₩x61₩x74₩x65₩x00...₩x02₩x80
  - date = counter(₩x02) + input data
- 2R HMAC 결과
  - Hmac-sha-256(key, data)
  - 결과 : 9b616f3932f821cab6c6b07c5de94fdc7bdaf3689795641ea57643e45fbf5baa

4. 3R HMAC 출력

- Input data = (bytearray)[Label+0x00+context+key length]
  - input data = ₩x6b₩x65₩x79₩x63₩x72₩x65₩x61₩x74₩x65₩x00...₩x02₩x80
  - date = counter(₩x03) + input data
- 3R HMAC 결과
  - Hmac-sha-256(key, data)
  - 결과 : f0aa900c08ca54fd0e6520fe6eaae45e365f2e27fff3ac5ee2e010d5d38a9739

5. 결과

- 각 라운드 별 생성 값
  - 1R HMAC 결과: 725c59018693ef0b87bf4a1fdb8ec6224168bb07358b2a562d5dad724b585ecf
  - 2R HMAC 결과: 9b616f3932f821cab6c6b07c5de94fdc7bdaf3689795641ea57643e45fbf5baa
  - 3R HMAC 결과: f0aa900c08ca54fd0e6520fe6eaae45e365f2e27fff3ac5ee2e010d5d38a9739



(그림 7-1) HMAC-SHA-256 기반 키 유도 함수

- 최종 출력 키
  - 1R+2R+3R HMAC 결과에서 왼쪽부터 80byte
  - 725c59018693ef0b87bf4a1fdb8ec6224168bb07358b2a562d5dad724b585ecf9b616f3932f821cab6c6b07c5de94fdc7bdaf3689795641ea57643e45fbf5baaf0aa900c08ca54fd0e6520fe6eaae45e365f2e27fff3ac5ee2e010d5d38a9739
- 암·복호화 키 (왼쪽부터 32byte) : 725c59018693ef0b87bf4a1fdb8ec6224168bb07358b2a562d5dad724b585ecf
- IV (암·복호화 키 이후 16byte)) : 9b616f3932f821cab6c6b07c5de94fdc
- 무결성 키 (IV 이후 32byte) : 7bdaf3689795641ea57643e45fbf5baaf0aa900c08ca54fd0e6520fe6eaae45e

※ 각 출력값은 JAVA 기준 bytearray 형식으로 한다.

7.1.3 데이터 암호화 및 복호화

본 절에서는 본인확인서비스를 수행하는 과정에서 발생하는 주요 데이터 및 개인정보 암호화 및 복호화에 대해 아래와 같이 정의한다. 아래 표시된 KDF의 상세 정보는 7.1.2에 서술되어 있다.

- 대칭 알고리즘 키 및 IV 값 생성

EncK = KDF(Key, Label, Context, Key length)

- Context : 매 거래 요청 시 본인확인기관에서 생성한 트랜잭션 값 사용
- Key : 본인확인기관 API 서버에서 발급받은 암호화 데이터 사용
- Label : 키 생성 시, 생성 목적 식별용으로 사용
- Key length : 80Byte
- EncK 생성 :
  - EncK = KDF(Key, Label, Context, Key length)
  - 암호화 키 : 32byte(생성된 키 데이터의 왼쪽부터 32byte)
  - IV : 16byte(암호화 키 이후 16byte)
- 해시 알고리즘 : HMAC-SHA256
- 암호화 알고리즘 : AES256-CBC / PKCS#7Padding

* 송/수신자는 같은 방법으로 복호화 키 유도

예시) 본인확인 결과 암·복호화 키 유도

context(tx_id) = A001.cad800ed-40e1-4876-a16a-177676d0d83a
Key(ticket) = 962abde1035d8ffd498d669a63c951841923f7062c1f8bcca8cccbaefdbb8e40
Label = keycreate

key length = 80byte

키 유도 이후

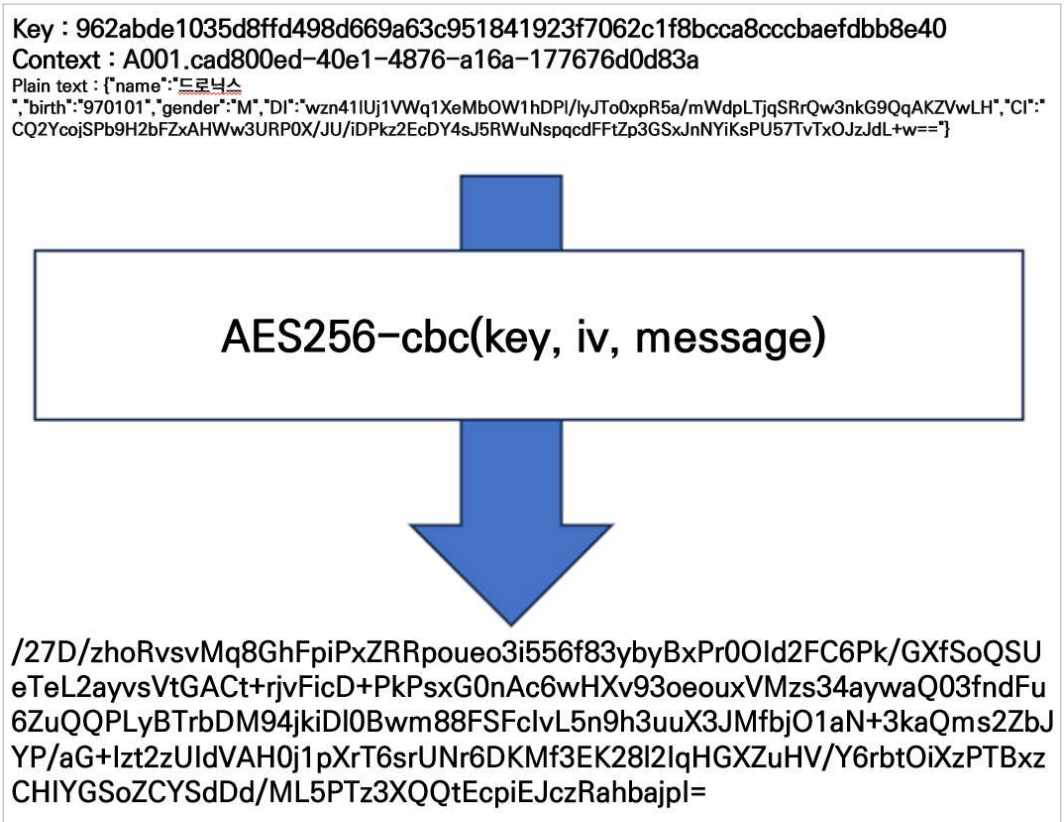
key : 725c59018693ef0b87bf4a1fdb8ec6224168bb07358b2a562d5dad724b585ecf

IV : 9b616f3932f821cab6c6b07c5de94fdc

Plain text = {"name":"드로닉스","birth":"970101","gender":"M","DI":"wzn41IUj1VWq1XeMbOW1hDPI/lyJTo0xpR5a/mWdpLTjqSRrQw3nkG9QqAKZVwLH","CI":"CQ2YcojSPb9H2bFZxAHWw3URP0X/JU/iDPkz2EcDY4sJ5RWuNspqcdFFtZp3GSxJnNYiKsPU57TvTxOJzJdL+w=="}

Encryption(aes256-cbc(key,iv,message))

->/27D/zhoRvsvMq8GhFpiPxZRRpoueo3i556f83ybyBxPr0OId2FC6Pk/GXfSoQSUeTeL2ayvsVtGACt+rjvFicD+PkPsxG0nAc6wHXv93oeouxVMzs34aywaQ03fndFu6ZuQQPLyBTrbDM94jkiDI0Bwm88FSFcIvL5n9h3uuX3JMfbjO1aN+3kaQms2ZbJYP/aG+Izt2zUIdVAH0j1pXrT6srUNr6DKMf3EK28I2IqHGXZuHV/Y6rbtOiXzPTBxzCHIYGSoZCYSdDd/ML5PTz3XQQtEcpiEJczRahbajpI=



(그림 7-2) 대칭 알고리즘 키 및 IV 값 생성

Decryption(aes256-cbc(key,iv,message))

->{"name":"드로닉스","birth":"970101","gender":"M","DI":"wzn41IUj1VWq1XeMbOW1hDPI/lyJTo0xpR5a/mWdpLTjqSRrQw3nkG9QqAKZVwLH","CI":"CQ2YcojSPb9H2bFZxAHWw3URP0X/JU/iDPkz2EcDY4sJ5RWuNspqcdFFtZp3GSxJnNYiKsPU57TvTxOJzJdL+w=="}

7.1.4 메시지 무결성 검증

본 절에서는 본인확인기관과 이용기관 간 전송하는 데이터 중 민감 정보의 무결성 검증을 위해 아래와 같이 검증하도록 정의한다.

무결성 검증에 사용되는 해시 알고리즘은 HMAC-SHA256을 사용하며 HMAC에 사용될 키는 암·복호화에 사용되는 방식과 같은 방식으로 유도한다. 아래 표시된 KDF의 상세 정보는 7.1.2에 서술되어 있다.

- 무결성 검증 키 생성

HmacK = KDF(Key, Label, Context, Key length)

- Context : 매 거래 요청 시 본인확인기관에서 생성한 트랜잭션 값 사용
- Key : 본인확인기관 API 서버에서 발급받은 암호화 데이터 사용
- Label : 키 생성 시, 생성 목적 식별용으로 사용
- Key length : 80Byte
- HmacK 생성 :
  - MacK = KDF(Key, Label, Context, Key length)
  - 32byte(생성된 키 데이터의 오른쪽 32byte)
- 해시 알고리즘 : HMAC-SHA256
- 유도하는 키 길이를 32byte일 경우 암·복호화 키와 동일한 키 값이 유도되기 때문에 80byte를 유도한 뒤 오른쪽 32byte를 사용하여 암·복호화키 및 IV와 겹치지 않은 데이터를 사용한다.

송/수신자는 같은 방법으로 무결성 검증에 사용될 검증키를 유도한다.

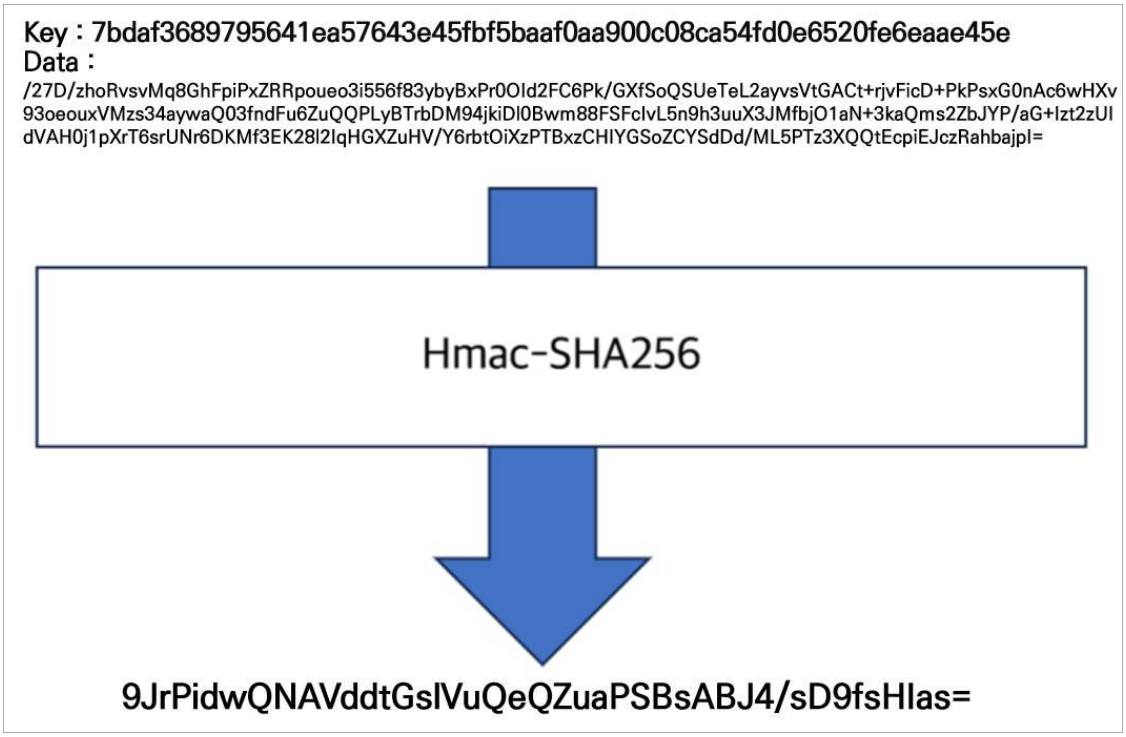
예시)

Context(tx_id) = A001.cad800ed-40e1-4876-a16a-177676d0d83a
Key(ticket) = 962abde1035d8ffd498d669a63c951841923f7062c1f8bcca8cccbaefdbb8e40
Label = keycreate
key length = 80byte

키 유도 이후
Hmac-key : 7bdaf3689795641ea57643e45fbf5baaf0aa900c08ca54fd0e6520fe6eaae45e
input Hmac : 9JrPidwQNAVddtGslVuQeQZuaPSBsABJ4/sD9fsHIas=
message : /27D/zhoRvsvMq8GhFpiPxZRRpoueo3i556f83ybyBxPr0Old2FC6Pk/GXfS
oQSUeTeL2ayvsVtGACt+rjvFicD+PkPsxG0nAc6wHXv93oeouxVMzs34aywaQ03fndFu6
ZuQQPLyBTrbDM94jkiDl0Bwm88FSFclvL5n9h3uuX3JMfbjO1aN+3kaQms2ZbJYP/aG+I

zt2zUIdVAH0j1pXrT6srUNr6DKMf3EK28l2IqHGXZuHV/Y6rbtOiXzPTBxzCHIYGSoZCYS
dDd/ML5PTz3XQQtEcpiEJczRahbajpI=

generate hmac = hmac_sha256(HmacK, message)



(그림 7-3) 무결성 검증 키 생성

generate Hmac : 9JrPidwQNAVddtGslVuQeQZuaPSBsABJ4/sD9fsHIas=

7.1.5 응답 재사용 여부 검증

응답 값을 재사용하는지 확인 및 검증해야 하며 이에 관련하여 아래를 충족해야 한다.

- 이용기관은 결과요청 값에 정당한 이용자임을 증명할 수 있도록 지정된 데이터를 반드시 포함하여 전송해야 한다.
- 본인확인기관은 이용기관이 요청하는 본인확인 결과가 정당한 이용자가 요청한 값인지 반드시 확인해야 한다.
- 인증 결과는 최초 1회만 결과를 발급하며 발급 이후 해당 인증을 만료해 결과 재요청을 방지한다.
- 본인확인 인증 건에 대한 응답은 10분에 한해 만료 시간을 유지하며 만료 시 해당 본인확인 인증 건을 만료시켜 본인확인 인증에 대한 보안성을 강화한다.

7.1.6 API 요약

<표 7-1> API 요약표

API	API 이름	설명
access	접근토큰 발급	인가된 이용기관 검증 및 API 접근토큰 발급
request	본인확인 요청	본인확인 발생 시 이용기관에서 본인확인기관으로 본인확인서비스 요청
result	본인확인 결과요청	이용자의 본인확인 결과 데이터 요청

7.2 접근토큰 발급 (access)

<표 7-2> 접근토큰 발급 API 설명

API 명	접근토큰 발급 (access)		
URL	~/ident/v1.0/access	Method	POST
설명	• API 사용 시 인가된 이용자임을 검증하기 위해 접근토큰을 발급한다. • 결과에 포함된 접근토큰은 이용기관에서 API 사용 시 Header에 Authorization에 삽입하여 API를 요청한다. • 인가된 이용자임을 확인할 수 있도록 최초 인증기관에서 발급받은 클라이언트 아이디, 클라이언트 시크릿을 사용한다. 예시) Authorization : Basic Base64({client id:client secret}) • 수신된 접근토큰은 유효기간 동안 유효하며 이후 만료를 통해 새로운 접근토큰 발급을 유도해야 한다. • 접근토큰 내부에 암호화 데이터가 포함되어 있어 API 사용 이외에 데이터를 외부로 노출되는 것을 금지한다.		

- Request Body (JSON)

<표 7-3> 요청 데이터(Request Body)

항목	타입	필수 여부	암호화 여부	설명
grant_type	String	M	N	“client_credentials” 고정 API 사용 자격증명 발급을 위해 사용
scope	String	O	N	발급한 토큰 권한 관리 필요할 때 사용

- Response body (JSON)

<표 7-4> 응답 데이터(Response Body)

항목	타입	필수 여부	암호화 여부	설명
code	String	M	N	응답 코드 : 응답 코드 정의 참조
message	String	M	N	응답 메시지 : 응답 코드 정의 참조
access_token	String	M	N	본인확인기관에서 이용기관 인증 이후 API 사용을 위해 생성한 토큰
expires_in	time	M	N	접근토큰 만료 시간 (Unix Time)
token_type	String	M	N	토큰 유형, “bearer” 고정

- Sample

● Request

```
Request POST /ident/v1.0/access HTTP/1.1
content-length: 55
Authorization: Basic RHJvbm14Q2xpZW50SU9GRHJvbm14U2VjcmV0
Content-Type: application/json

{
  "grant_type": "client_credentials",
  "scope" : ""
}
```

● Response

```
Response HTTP/1.1 200
Content-Type: text/plain;charset=UTF-8
Content-Length: 384
Date: Mon, 03 Jun 2024 14:17:36 GMT
Connection: close

{
  "code":200,
  "message" : "응답성공",
  "access_token" : "eyJhbGciOiJIUzI1NiJ9.ew0GICAgInVzZXJpZCI6IjAxYphdGlvb1i6ICJECcm9uaXlgLoq
ICAgInNjb3BlIjogbyAiSSIsICJ1NiIgMiwgIlAiIF9sCiAgICAidGJav2l0gijIAZzU1MGJnN2ZmNDkYTU0MTcwYmuXMj
Y3MjgyMmNkMTlhNDNiRDNid2jNmYmQwZWIzIgogICAgImhlbCI6IDEM3ATzODI2NDUsCiAgICAiZXhwIjogMTc0
MTcwODY0NswKICAgICJldGMiOiAiIg9.fMV6j5GUcEVTb_Sf7sIsqo5hqA2_g91y4rKU6VU1bJpA",
  "expires_in": 1714918645,
  "token_type": "Bearer"
}
```

7.3 본인확인 요청 (request)

<표 7-5> 본인확인 요청 API 설명

API 명	본인확인 요청 (request)		
URL	~/ident/v1.0/request	Method	POST
설명	• 본인확인서비스에 대한 최초 트랜잭션 발생 단계이다. • 이용기관은 본인확인기관으로 거래를 식별할 수 있는 데이터와 필요에 따라 인가된 이용기관을 식별할 수 있는 데이터를 함께 전송한다. • 본인확인기관은 해당 거래를 식별할 수 있는 트랜잭션 ID(tx_id)를 생성하고 표준창 URL과 같이 반환한다.		

- Request Body (JSON)

<표 7-6> 요청 데이터(Request Body)

항목	타입	필수 여부	암호화 여부	설명
site_tx	String	M	N	이용기관에서 자체적으로 생성한 고유 요청 번호. 이용기관의 고유 정보 (cpCode, 요청 시간 등)을 활용하여 트랜잭션을 부분 하기 위한 값
service_type	String	M	N	본인확인 인증수단을 구분하기 위한 식별자. 각 타입에 맞춰 본인확인 표준창 URL 반환 service_type = {I|M|C|S|A} - I : i-PIN 본인확인서비스 표준창 URL - M : 휴대폰 본인확인서비스 표준창 URL - C : 카드 본인확인서비스 표준창 URL - S : 공동 인증서 본인확인서비스 표준창 URL - F : 금융 인증서 본인확인서비스 표준창 URL - A : 모바일 인증서 본인확인서비스 표준창 URL
req_code	String	M	N	이용기관에서 필요한 본인확인 결과 코드. req_code = {none|CI|DI|ALL} - none : 본인확인 결과를 요청하지 않는 경우, 이 경우 응답 메시지를 암호화해서 전달 한다. - CI : CI만 요청하는 경우, 본인확인 결과 데이터에 DI가 포함되지 않는다. - DI : DI만 요청하는 경우, 본인확인 결과 데이터에 CI가 포함되지 않는다. - ALL : CI/DI를 모두 요청하는 경우, 본

항목	타입	필수 여부	암호화 여부	설명
				인확인 결과 데이터에 CI/DI가 모두 포함된다.
callback	String	M	N	이용자 본인확인 이후 결과 데이터 전송 경로
callback_type	String	M	N	이용자 본인확인 이후 트랜잭션 ID 전송 방식. callback_type = {T1|T2} - T1 : 외부 데이터 접근 허용하는 경우 (Server To Server) - T2 : 외부 데이터 접근 불허하는 경우 (Server To Browser)
auth_type	String	O	N	APP Push를 통한 본인확인서비스 이용 시 사용
temp_data	String	O	N	본인확인기관별 추가로 이용기관에게 전송받을 데이터

- Response body (JSON)

<표 7-7> 응답 데이터(Response Body)

항목	타입	필수 여부	암호화 여부	설명
code	String	M	N	응답 코드 : 응답 코드 정의 참조
message	String	M	N	응답 메시지 : 응답 코드 정의 참조
tx_id	String	M	N	이용기관에서 요청한 트랜잭션을 식별하기 위해 본인확인기관에서 생성한 고유 정보
auth_url	String	M	N	이용자 인증을 위한 표준창 URL

- Sample

● Request

```
Request POST /ident/v1.0/request HTTP/1.1
content-length: 78
Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.
ewogICAgInVzZXJpZCI6IjAxYphdGlvb1i6ICJECGcm9uaXlgLoqICAgInNjb3BlIjogbyAiSSIsICJ1NiI
gMiKiLCAiUyIsICJ1GilwIlAilFhscsCiAgICIAidGJav2l0gijlAZzU1MGJnN2ZmNDkYTU0MTcwYmuX
MjY3MjgyMmNkMTlhNDNiRDNid2jNmYmQwZWlzIgogICAgImhlbCI6IDE3MTA2ODI2NDUsCiAgICC
IAZixNzXhjogMTc0MTcwODY0NswKICAgICJldGMiOiAilgpg9.
fMV6j5GUcEVTb_Sf7sIsqo5hqA2_g91y4rKU6VU1bJpA
Content-Type: application/json

{
```

```
  "site_tx":"20240624145005",
  "service_type":"M",
  "req_code":"CI",
  "callback":"http://{CPUrl}/Sample/Phone/result",
  "callback_type":"T1",
  "temp_data":"test"
}
```

- Response

```
Response HTTP/1.1 200
Content-Type: application/json
Date: Mon, 24 JUN 2024 14:50:05 GMT
Connection: close
Content-Length: 141

{
  "code":"200",
  "message":"응답성공",
  "auth_url":"http://{표준창 Url}",
  "tx_id":"A001.25998660-c751-4e17-b05e-3b65d57296d2"
}
```

7.4 본인확인 결과요청 (result)

<표 7-8> 본인확인 결과요청 API 설명

API 명	본인확인 결과요청 (result)		
URL	~/ident/v1.0/result	Method	POST
설명	• 이용자가 인증 완료한 본인확인서비스에 대한 결과 요청이다. • 각 인증수단을 통해 이용자가 본인확인을 완료하여 이용기관으로 전달된 트랜잭션 ID(tx_id)값을 본인확인기관에 전송하여 암호화된 개인정보를 반환한다. • 결과를 생성하는 본인확인기관은 암호화 데이터가 생성된 접근토큰의 시간을 결과 정보에 포함하여 복호화 시 같은 암호화 데이터를 바탕으로 키 유도할 수 있도록 한다. • 요청하는 트랜잭션의 본인확인이 아직 완료되지 않았을 때 진행 중을 나타내는 상태 값을 반환한다. • 응답 값의 encdata와 hmac의 결과는 데이터 손실을 방지하기 위해 암호화 이후 base64 인코딩하여 전송한다.		

- Request body (JSON)

<표 7-9> 요청 데이터(Request Body)

항목	타입	필수 여부	암호화 여부	설명
tx_id	String	M	N	트랜잭션을 통해 필요한 본인확인 결과요청

- Response body (JSON)

<표 7-10> 응답 데이터(Response Body)

항목	타입	필수 여부	암호화 여부	설명
code	String	M	N	응답 코드 : 응답 코드 정의 참조
message	String	M	N	응답 메시지 : 응답 코드 정의 참조
tx_id	String	M	N	트랜잭션을 통해 키 유도하기 위한 callback
encdata	String	M	Y	JSON 포맷의 암호화된 이용자의 본인확인 결과(base64encode) user_name : 이름 birth : 생년월일 {YYYYMMDD} gender : 성별 {M|F} M : 남자, F : 여자 national_info : 내/외국인 구분 L : 내국인, F : 외국인 ci : 연계 정보 di : 중복가입 방지 정보 ci2 : ci 대체 연계 정보
hmac	String	M	Y	암호화 데이터의 무결성 검증을 위한 검증값(base64encode)

- Sample

● Request

```
Request POST /ident/v1.0/result HTTP/1.1
content-length: 54
Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.ewoJICAgInVzZXJpZCI6IjAxYphdGlvb1i6ICJECGcm9u
aXlgLoqICAgInNjb3BlljogbyAiSSIsICJ1NiIgMiKiLCAiUyIsICJ1GilwIIAiIFhscsCiAgICIAidGJav2l0gijIAZzU
1MGJnN2ZmNDkYTU0MTcwnYmuXMjY3MgyMmNkMTIhNDNiRDNid2jNmYmQwZWlzIgogICAgImhlbCI
6ICJED3MzOTA2NDNUSCIgICAgICIAZxNwIjogMTc0MTcwODY0NswKICAgICJcldGMiOiAiIgpg9.fMV6j
5GUcEVtb_Sf7sIsoqShqA2_g91y4rKU6VU1bJpA
Content-Type: application/json

{
  "tx_id":"A001.25998660-c751-4e17-b05e-3b65d57296d2"
}
```

- Response

```
Response HTTP/1.1 200
Content-Type: application/json
Date: Mon, 03 Jun 2024 14:17:36 GMT
Content-Length: 536

{
  "code":"200",
  "message":"응답성공",
  "tx_id":"A001.25998660-c751-4e17-b05e-3b65d57296d2",
  "encData":"mHhNJawjjhY0Uvo0QhG8lubg6xcpY+2duKfFx+EvDzh1cSKBtrOAyNr1i6Qko2rq
OK7Xi/WkEGPoSknluz8FS1u7F9x6OLi8yUQZAf7nfvFzDZnEZ0JynCvGYQxIPBK+qXgn9uesyhLA
TcSdDxR0URMbLz/dKsp9kKtjNCcy79QiuI9ZetZDX0gn+tES07NbgY4CDv9u3ZbStj6mOtAE0mpi
oi7GeJE8joscSRU3RyU2wDbISUi5ngfHYLaoCVcYBqF9px0VXGYIbB1vUSEKNmg9j48S7mJZ1I0
6B3ms5IcYjc1TCQ84CQciKPx29X4/W6FD9BM0e2XkPOEtGWMZiHy/td70Ve90fenRXxqKRA=",
  "HMAC":"qM4OWebMsyDKTDM9No+GmbJDrenWFAz9DfJ+FLfcISY="
}
```

7.5 응답 코드

<표 7-11> 응답 코드 표

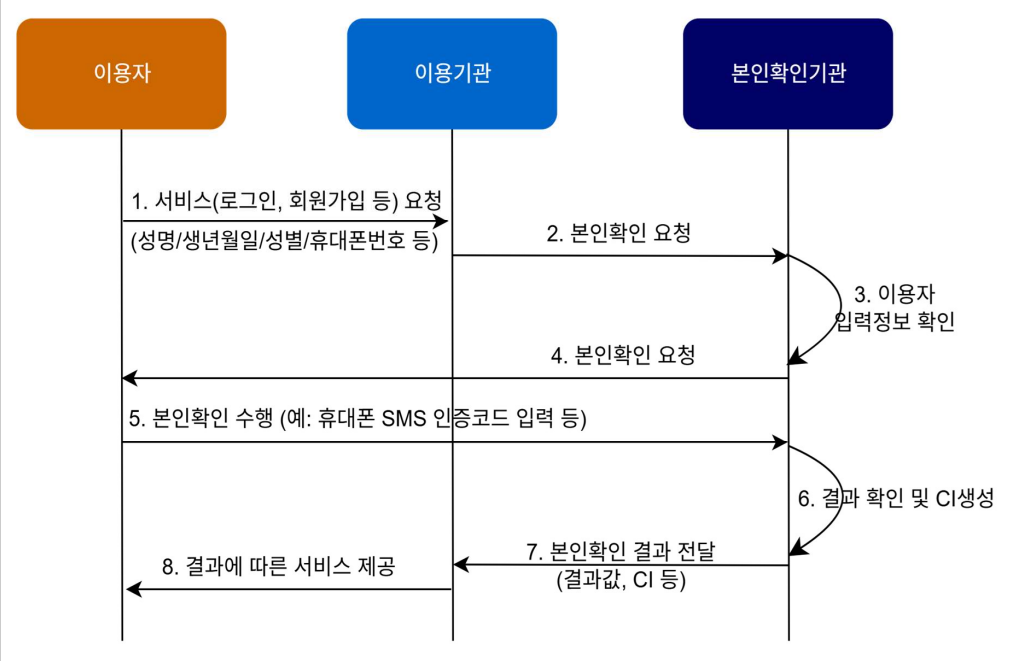
HTTP	CODE	MESSAGE	설명
200	200	SUCCESS	응답성공
400	001	AUTHORIZATION_HEADER_ERROR	헤더 오류
400	002	INVALID_PARAMETER	파라미터 오류
400	003	TOKEN_EXPIRATION_ERROR	토큰 만료 오류
400	004	EXPIRATION_TIME_ERROR	결과조회 시간 만료 오류
400	005	EXPIRATION_COUNT_ERROR	결과조회 횟수 만료 오류
400	006	SERVICE_FAILURE_ERROR	서비스 장애
400	007	ACCESS_DENIED	접근 거부
400	008	INVALID_USER_ERROR	잘못된 이용자
500	500	INTERNAL_SERVER_ERROR	서버 오류
500	999	UNKNOWN_ERROR	알 수 없는 오류

부 록 Ⅰ

(본 부록은 표준을 보충하기 위한 내용으로 표준의 일부는 아님)

기존의 본인확인 프레임워크

본 부록에서는 본 표준의 적용 범위와 목적을 보다 명확히 이해할 수 있도록 기존 본인확인 프레임워크에서 사용된 일반적인 인증 절차를 서술한다.



(그림 Ⅰ.1-1) 인증절차

① 이용자는 이용기관이 제공하는 웹 또는 앱 환경에서 서비스 요청한다.

② 이용기관은 전달받은 이용자 정보를 포함하여 본인확인기관에게 본인확인 요청을 전송한다.

③ 본인확인기관은 전달받은 이용자 정보에 대해 일치여부를 확인한다.

④ 본인확인기관은 이용자 정보가 일치한 경우, 해당 이용자에게 본인확인 요청을 수행한다.

⑤ 이용자는 본인확인기관으로부터 요청받은 본인확인 절차를 수행한다. (예를 들어, 휴대폰 SMS를 통해 전달받은 인증코드 입력 등)

⑥ 본인확인기관은 본인확인 결과를 확인하고, 유효한 경우 CI를 생성한다.

⑦ 본인확인기관은 이용기관에게 생성한 CI를 포함하여 본인확인 결과를 전달한다.

⑧ 이용기관은 전달받은 결과를 확인하고, 그에 따라 이용자에게 최종 서비스를 제공한다.

부 록 II

(본 부록은 표준을 보충하기 위한 내용으로 표준의 일부는 아님)

기능별 연관 표준 목록

본 부록에서는 6.2의 주요 기능별 보안대책에서 참고한 관련 표준 및 기술 가이드라인에 대해 서술한다.

6.2에서 서술한 주요 기능별 보안대책에 대해 참고한 국내외 가이드라인 및 표준 목록은 다음과 같다.

<표 II.1-1> 기능별 연관 표준 목록

번호	내용	구분	표준명	적용 위치
1	Base 인코딩	국외 기술문서	RFC 4648: The Base16, Base32, and Base64 Data Encodings	6.2.1
2	JSON		RFC 8259: The JavaScript Object Notation(JSON) Data Interchange Format	6.3.4
3	클라이언트 아이디 및 시크릿		RFC 6749: The OAuth 2.0 Authorization Framework	6.2.1
4	TLS	국내 가이드라인	(2024.04) 국가정보원, “국가용 보안요구사항”	6.2.4
		국외(미국) 가이드라인	(2019) NIST SP 800-52 Rev.2 “Guidelines for the Selection, Configuration, and Use of Transport Layer Security(TLS) Implementations”	
5	암호 보안강도	국내 가이드라인	(2018) 과학기술정보통신부, 한국인터넷진흥원, “암호 알고리즘 및 키 안내서”	6.2.5.2
			(2024.04) 국가정보원, “국가용 보안요구사항”	
		국외(미국) 가이드라인	(2019) NIST SP 800-131A Rev.2 “Transitioning the Use of Cryptographic Algorithms and Key Length”	
			(2020) NIST SP 800-57 Part1 Rev.5 “Recommendation for Key Management:Part 1 - General”	
6	암호화 적용 시점	국내 가이드라인	(2020.12) 개인정보보호위원회, 한국인터넷진흥원, “개인정보의 암호화 조치 안내서”	6.2.5.2
		국외(미국) 가이드라인	(2019) NIST SP 800-52 Rev.2 “Guidelines for the Selection, Configuration, and Use of Transport Layer Security(TLS) Implementations”	

번호	내용	구분	표준명	적용 위치
7	알고리즘의 보안 수명 기간	국외(미국) 가이드라인	(2020) NIST SP 800-57 Part1 Rev.5 “Recommendation for Key Management:Part 1 - General”	6.2.5.2
8	HMAC	국내 표준문서	(2018) TTAK.KO-12.0334-Partr2: 패스워드 기반 키 유도 함수 - 제2부:해시함수 SHA-2	7.1.2
		국외 표준문서	(2018.10) ISO/IEC 10118-3, “IT Security - Hash-functions-Part 3: Dedicated hashfunctions”	

부 록 III

(본 부록은 표준을 보충하기 위한 내용으로 표준의 일부는 아님)

Message Schema (JSON-SCHEMA)

본 부록에서는 7절의 본인확인서비스 API에 대한 메시지 스키마 예시를 소개한다.

III.1 Access

접근토큰 발급에 대한 요청 및 응답 스키마 예시는 다음과 같다.

- Request Schema

```
{
    "$schema": "http://JSON-schema.org/schema#",
    "type": "object",
    "properties": {
        "grant_type": {
            "type": "string",
            "description" : "API 사용 자격증명을 발급받기 위해 사용 ('client_credentials' 고정)"
        },
        "scope": {
            "type": "string",
            "description" : "발급한 토큰 권한 관리 필요시 사용"
        }
    },
    "required": [
        "grant_type"
    ]
}
```

- Response Schema

```
{
    "$schema": "http://JSON-schema.org/schema#",
    "type": "object",
    "properties": {
        "access_token": {
            "type": "string",
            "description" : "API 사용을 위해 생성한 접근토큰"
        },
```

```
        "code": {
            "type": "integer",
            "description" : "응답 상태에 대한 코드"
        },
        "message": {
            "type": "string",
            "description" : "응답 상태에 대한 메세지"
        },
        "token_type": {
            "type": "string",
            "description" : "토큰 유형 ('bearer' 고정)"
        },
        "expires_in": {
            "type": "integer",
            "description" : "access_token 만료 시간 (UNIXTIME)"
        },
    },
    "required": [
        "access_token",
        "code",
        "expires_in",
        "message",
        "token_type"
    ]
}
```

III.2 Request

본인확인 요청에 대한 요청 및 응답 스키마 예시는 다음과 같다.

- Request Schema

```
{
    "$schema": "http://JSON-schema.org/schema#",
    "type": "object",
    "properties": {
        "site_tx": {
            "type": "string",
            "description" : "이용기관에서 자체적으로 생성한 고유 요청 번호."
        },
        "service_type": {
            "type": "string",
```

```
            "description" : "본인확인 인증수단을 구분하기 위한 식별자"
        },
        "auth_type": {
            "type": "string",
            "description" : "APP Push를 통한 본인확인서비스 이용 시 사용하는 값"
        },
        "req_code": {
            "type": "string",
            "description" : "이용기관에서 필요한 본인확인 결과 코드"
        },
        "call_back": {
            "type": "string",
            "description" : "이용자 본인확인 이후 결과 데이터 전송 경로"
        },
        "temp_data": {
            "type": "string",
            "description" : "본인확인기관별 추가로 이용기관에게 전송받을 데이터"
        }
    },
    "required": [
        "req_code",
        "service_type",
        "site_tx",
        "call_back",
        "auth_type",
        "temp_data"
    ]
}
```

- Response Schema

```
{
    "$schema": "http://JSON-schema.org/schema#",
    "type": "object",
    "properties": {
        "code": {
            "type": "string",
            "description" : "응답 상태에 대한 코드"
        },
        "auth_url": {
            "type": "string",
            "description" : "이용자 인증을 위한 표준창 URL"
        },
```

```
        "message": {
            "type": "string",
            "description" : "응답 상태에 대한 메시지"
        },
        "tx_id": {
            "type": "string",
            "description" : "이용기관에서 요청한 트랜잭션을 식별하기 위해 본인확인기관에
                            서 생성한 고유정보"
        }
    },
    "required": [
        "auth_url",
        "code",
        "message",
        "tx_id"
    ]
}
```

III.3 Result

본인확인 결과요청에 대한 요청 및 응답 스키마 예시는 다음과 같다.

- Request Schema

```
{
    "$schema": "http://JSON-schema.org/schema#",
    "type": "object",
    "properties": {
        "tx_id": {
            "type": "string",
            "description" : "이용기관에서 요청한 트랜잭션을 식별하기 위해 본인확인기관에
                            서 생성한 고유 정보"
        }
    },
    "required": [
        "tx_id"
    ]
}
```

- Response Schema

```
{
    "$schema": "http://JSON-schema.org/schema#",
    "type": "object",
    "properties": {
        "code": {
            "type": "string",
            "description" : "응답 상태에 대한 코드"
        },
        "HMAC": {
            "type": "string",
            "description" : "암호화 데이터의 무결성 검증을 위한 검증 값"
        },
        "encData": {
            "type": "string",
            "description" : "본인확인 결과의 암호화 데이터"
        },
        "message": {
            "type": "string",
            "description" : "응답 상태에 대한 메시지"
        },
        "tx_id": {
            "type": "string",
            "description" : "이용기관에서 요청한 트랜잭션을 식별하기 위해 본인확인기관에
                            서 생성한 고유 정보"
        }
    },
    "required": [
        "HMAC",
        "code",
        "encData",
        "message",
        "tx_id"
    ]
}
```

부 록 IV-1

(본 부록은 표준을 보충하기 위한 내용으로 표준의 일부는 아님)

지식재산권 확약서 정보

해당 사항 없음

부 록 IV-2

(본 부록은 표준을 보충하기 위한 내용으로 표준의 일부는 아님)

시험인증 관련 사항

해당 사항 없음

부 록 IV-3

(본 부록은 표준을 보충하기 위한 내용으로 표준의 일부는 아님)

본 표준의 연계(family) 표준

해당 사항 없음

부 록 IV-4

(본 부록은 표준을 보충하기 위한 내용으로 표준의 일부는 아님)

참고 문헌

아래 기재된 참고 문헌의 발간일이 기재된 경우, 해당 표준(문서)의 해당 버전에 대해서만 유효하며 연도를 표시하지 않은 경우에는 해당 표준(권고)의 최신 버전을 따른다.

[1] Bradner, S. (1997). RFC 2119: Key words for use in RFCs to Indicate Requirement Levels.

[2] Josefsson, S. (2006). RFC 4648: the base16, base32, and base64 data encodings.

[3] Bray, T. (Ed.). (2017). RFC 8259: The JavaScript object notation (JSON) data interchange format.

[4] Hardt, Dick. (2012). RFC 6749: The OAuth 2.0 Authorization Framework.

[5] NIST SP 800-131A Rev.2 Transitioning the Use of Cryptographic Algorithms and Key Lengths (2019)

[6] NIST SP 800-52 Rev.2 Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations (2019)

[7] NIST SP 800-57: Recommendation for Key Management (2020)

[8] NIS, 국가용 보안요구사항 (2024.04)

[9] NIS, 소프트웨어 암호모듈 검증기준

[10] 과학기술정보통신부, KISA, 암호 알고리즘 및 키 길이 안내서 (2018.12)

[11] 개인정보보호위원회, KISA, 개인정보의 암호화 조치 안내서 (2020.12)

[12] TTAK.KO-12.0334-Part2 : 패스워드 기반 키 유도 함수-제2부: 해시함수 SHA-2 (2018)

[13] ISO/IEC 10118-3: IT Security - Hash-functions-Part 3: Dedicated hash functions (2018.10)

[14] NIST SP 800-63-3: Digital Identity Guidelines (2020)

[15] NIST SP 800-63B: Authentication and Lifecycle Management (2020)

[16] TDIF(Trusted Digital Identity Framework) - Australian Government Architecture https://architecture.digital.gov.au/trusted-digital-identity-framework-tdif-0

부 록 IV-5

(본 부록은 표준을 보충하기 위한 내용으로 표준의 일부는 아님)

영문표준 해설서

해당 사항 없음

부 록 IV-6

(본 부록은 표준을 보충하기 위한 내용으로 표준의 일부는 아님)

표준의 이력

판수	채택일	표준번호	내용	담당 위원회
제1판	2025.12.05	TTAK.KO-12.0429	-	개인정보보호/ID관리, 블록체인 보안 (PG502)