International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 16, Number 2, pages 118–146, June 2026

Proposal of an Identity Verification System Using Trusted Execution Environment in the Japan's Digital Authentication App

Tomoki Yasui
Graduate School of Advanced Science and Engineering, Hiroshima University 1-4-1 Kagamiyama, Higashi-Hiroshima, 739-8527, Japan

Hidenobu Watanabe
Information Media Center, Hiroshima University 1-4-1 Kagamiyama, Higashi-Hiroshima, 739-8527, Japan

Received: February 20, 2026
Revised: May 3, 2026
Accepted: June 4, 2026
Communicated by Yasuyuki Nogami

Abstract

As digital identity systems continue to evolve, ensuring the secure handling of personal data and authentication tokens on the service provider side has become increasingly critical. Japan's My Number Card is a national identification card that contains an IC chip storing personal information such as name, address, and date of birth, and is increasingly used for digital services. The Digital Authentication App provided by Japan's Digital Agency enables online identity verification using the My Number Card and provides APIs that allow service providers to access user information with the user's consent. Although the Digital Agency mandates the secure management of such data, its guidelines do not specify concrete technical countermeasures for server-side environments that process and store tokens and personal data.

To address this issue, we propose a practical server-side architecture that integrates AWS Nitro Enclaves, a Trusted Execution Environment (TEE), with Japan's Digital Authentication App APIs. Our system confines OpenID Connect (OIDC) token validation and user information access entirely within a TEE, thereby resisting insider threats and compromised operating systems. Furthermore, audit logs are protected with digital signatures and hash chaining, enabling verifiable integrity and non-repudiation.

To validate the feasibility and performance of the proposed architecture, we implemented a prototype demo system and conducted measurement experiments. Preliminary measurement results indicate that the integration of TEE introduces limited performance overhead while enhancing security guarantees. This architecture demonstrates a practical and scalable approach to strengthening trust in digital authentication services from the service provider's perspective.

Keywords: TEE, Identity Verification, Japan's Digital Authentication App

1 Introduction

In recent years, the demand for online identity verification has rapidly increased. With the advancement of the digital society, traditional face-to-face identity verification methods are becoming

inadequate in many situations, and secure verification technologies in remote environments are gaining importance. Particularly in financial institutions and universities, there is a growing need for secure and efficient identity verification when users access services online.

To address these needs, the Digital Agency of Japan has developed the Digital Authentication App[2], which enables online identity verification using the My Number Card (Japanese national ID card). This infrastructure simplifies authentication and facilitates integration into various online services. Government agencies and private organizations can utilize the APIs provided by the app to implement secure authentication mechanisms based on the My Number Card.

Universities are also facing the need to verify the identity of external users, such as incoming students and guests. Traditionally, identity verification for new students, such as the issuance of student IDs, was conducted in person. However, due to various constraints, in-person verification has become difficult for some students. Furthermore, universities often need to complete initial device configuration and credential issuance before the beginning of the academic term. These preparations necessitate secure identity verification in advance, even before the student physically arrives on campus. By introducing online identity verification, universities can streamline these processes and reduce the burden on both students and administrative staff.

At Hiroshima University, an online verification process based on information submitted in advance has been adopted. However, the system relied on a combination of examinee ID and dates of birth as passwords, which posed a risk of unauthorized access by individuals familiar with the applicants.

Based on this background, we define the central research question of this study as "How can service providers securely process identity verification tokens based on My Number Cards without trusting the host OS or administrators?". To answer this question, we design and implement a system architecture that executes the entire OIDC flow inside AWS Nitro Enclaves, a type of TEE. By confining critical operations within the TEE and protecting logs with tamper-proof signatures and hash chains, we provide a feasible path for server-side protection of the Digital Authentication App.

Our main contributions are as follows:

- We present a server-side architecture that integrates Trusted Execution Environments (TEEs) with the Digital Authentication App APIs, and define a clear threat model in which the service provider's host operating system and administrators are not trusted. This architecture demonstrates a practical direction for Enclave-based OpenID Connect processing and potential scalability to university-level enrollment scenarios.
- We implement a prototype system that confines core OpenID Connect processing, including authentication token validation and sensitive user data handling, within AWS Nitro Enclaves, while keeping non-sensitive components such as database access outside the Enclave. This design explicitly clarifies the security boundary of the TEE and mitigates insider threats, token leakage, and OS-level compromises.
- We also design a tamper-evident audit logging mechanism that combines TEE-based log signing, WORM storage, and blockchain commitments, enabling integrity, non-repudiation, and ex-post verification even under partial system compromise.
- Through a prototype-based evaluation, we demonstrate that the proposed architecture incurs an additional overhead of approximately 2 ms due to TEE-based processing compared with a standard OpenID Connect flow without TEE, and a total end-to-end overhead of approximately 5 ms when including audit logging and baseline OIDC processing. These results indicate that the security benefits of Enclave-based processing can be achieved with practical performance costs in real-world deployments.

The remainder of this paper is organized as follows. Section 2 describes the Digital Authentication App and presents the threat analysis and system requirements. Section 3 explains the architecture of the proposed system. Section 4 discusses related work. Section 5 presents the evaluation methodology and experimental results. Section 6 provides a discussion of the results. Finally, Section 7 concludes this paper.

2 Digital Authentication App

2.1 Overview

The authentication and authorization processes of the Digital Authentication App are based on the OAuth 2.0 authorization flow and OIDC. Figure 1 illustrates the authentication sequence, adapted from the Digital Agency's documentation[3]. The diagram describes the flow of interactions among the user, browser, the service provider(Relying Party, RP), and the Digital Agency's API Server.

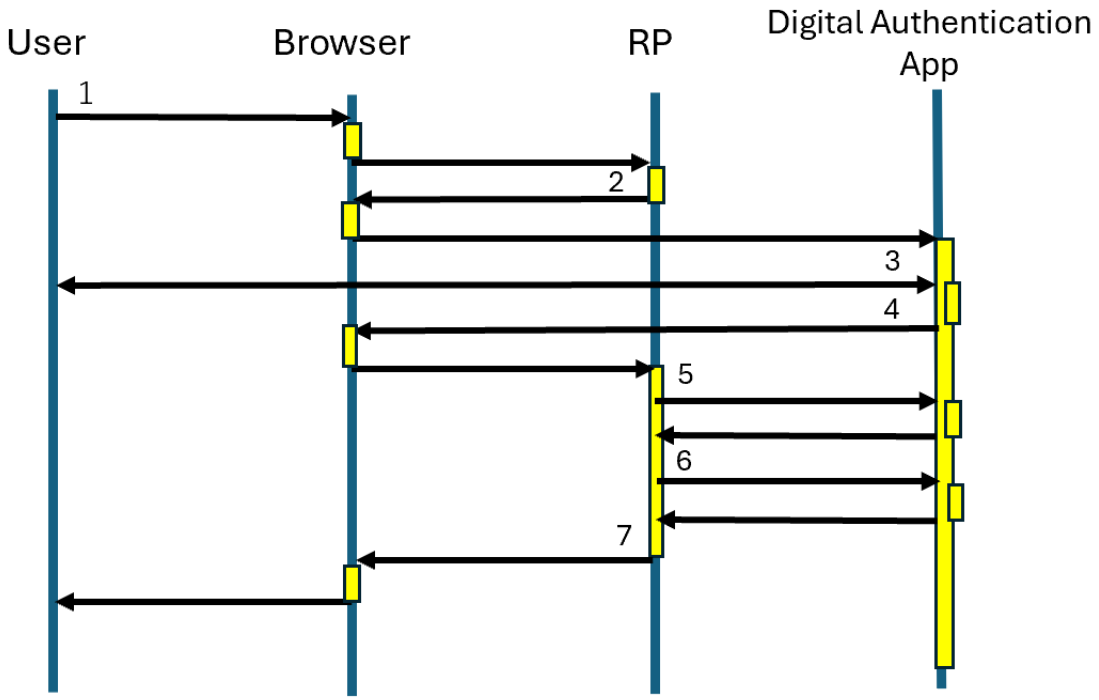

Figure 1: Authentication Process Flow of the My Number System

1. The user sends an authentication request to the RP via a browser.

2. The RP redirects the authorization request to the Digital Agency's API Server via the browser.

3. The authentication and authorization process is conducted between the user and the Digital Agency using the digital certificate embedded in the user's My Number Card.

4. The result of the authentication is returned to the RP through a redirect. If successful, an authorization code is issued.

5. The RP uses the received authorization code to request tokens from the Digital Agency's server. The obtained tokens include an ID token, access token, and refresh token.

6. The RP sends a request to the UserInfo endpoint using the access token to obtain the user's personal information (name, address, date of birth, and gender).

7. A response confirming successful authentication is returned to the user, and the service proceeds to post-authentication processing.

The Digital Authentication App employs multi-factor authentication, combining knowledge-based factors (a 4-digit PIN) and possession-based factors (confirmation of the My Number Card via NFC on a smartphone).

The types of personal information that can be obtained are limited to the so-called "basic four information": name, address, date of birth, and gender. As such, facial images or bank account information are not included. Therefore, manual facial comparison is not part of this authentication process.

2.2 Problem Analysis

When service providers (e.g., universities) adopt a user verification system based on the Digital Authentication App, they need to temporarily retain and process sensitive data such as authentication tokens and user information. These data are highly confidential, and their leakage could lead to serious problems from the standpoint of personal information protection.

In addition, the following security threats exist in API communications and server-side processing:

- **Malicious insider threats**: Unauthorized access by system administrators or operational personnel
- **External threats**: Attacks by third parties exploiting operating system or middleware vulnerabilities
- **Accidental or design-related risks**: Token leakage due to mismanagement, tampering with processing logs caused by insufficient safeguards, or inability to ensure non-repudiation due to flawed operational practices

2.3 System Requirement

To address these threats, we defined the following security requirements:

- **Confidentiality of authentication tokens**: Tokens must not be stored, reused, or output in plaintext unnecessarily.
- **Trustworthiness of the execution environment**: Even if the operating system or administrators are compromised, the system must ensure secure execution of sensitive processes.
- **Log integrity**: Processing results and validation data must be stored in a tamper-evident manner to support future audits.
- **Non-repudiation**: Actions performed by administrators or system components must be provable, ensuring accountability and preventing denial of performed operations.

3 Proposed System Architecture

In this study, we propose a secure identity verification system and audit logging system based on TEE to fulfill the above requirements. While TEEs such as Intel SGX and AWS Nitro Enclaves have been widely studied, their suitability for server-side authentication processing differs significantly. Yasui et al. [10] designed the system using Intel SGX. However, SGX requires additional drivers and libraries, and imposes limitations such as constrained memory size, which increases the development and operational burden. In contrast, AWS Nitro Enclaves are designed for constructing secure environments in the cloud. They allow Enclaves to be easily created on EC2 instances, require no modifications to the operating system kernel, and provide flexible memory allocation since the Enclave size depends on the memory of the parent instance rather than a fixed upper limit. Furthermore, Nitro Enclaves are built on Docker images, which enhances their operational

efficiency, extensibility, and reproducibility. In addition, their Docker-based design allows the Enclave environment to be discarded once the processing is complete, thereby reducing the risk of residual sensitive data. For these reasons, this study adopts Nitro Enclaves instead of Intel SGX as the implementation platform[11]. Nevertheless, the proposed method is not limited to Nitro-specific features. Its essential requirement is a TEE that can isolate sensitive OIDC processing from the host operating system and administrators. From this perspective, other TEE technologies, such as Intel SGX and AMD SEV, are also relevant alternatives. Intel SGX provides fine-grained enclave isolation at the process level, whereas AMD SEV mainly protects virtual-machine memory through encryption. Compared with SGX, Nitro Enclaves offer simpler deployment and greater operational flexibility in cloud environments. Therefore, while the detailed trust boundary and implementation method differ across platforms, the overall architecture proposed in this paper can be generalized to multiple TEE technologies. This suggests that the proposed approach is applicable beyond a specific cloud platform, although its concrete realization depends on the characteristics of the selected TEE.

3.1 TEE

A TEE is a confidential computing technology that leverages trusted hardware to create isolated regions within memory for secure data processing. It is aligned with the concept of zero trust, where no component outside the trusted boundary is inherently trusted. The memory is logically divided into two regions: the trusted region (Enclave) and the untrusted region, known as the Rich Execution Environment (REE). In this architecture, data processing is securely performed within the isolated TEE region, while the remaining memory is considered untrusted. Because a TEE processes plaintext data only within the trusted boundary, it achieves low overhead in processing speed. This makes TEE a suitable solution for authentication systems that require high performance and strong security guarantees.

In this study, we initially considered Intel SGX[4] as the TEE implementation, but ultimately adopted AWS Nitro Enclaves[1] (hereinafter referred to as Nitro) due to its operational advantages.

AWS Nitro Enclaves, provided by Amazon Web Services, are one of the implementations of TEE. It enables the creation of lightweight virtual machines (called Enclaves) that are isolated from the network and persistent storage within an AWS EC2 instance, providing a secure environment for sensitive data processing.

Within the Enclave, an isolated execution environment is established, separated from the parent instance. This isolation ensures that confidential data can be processed securely without being exposed to the operating system or administrators.

Nitro employs a dedicated communication channel (vsock) for interaction between the parent instance and the Enclave, minimizing the attack surface. In addition, Nitro can be set up easily using the AWS CLI tools. As a result, Nitro provides a secure yet flexible infrastructure for confidential computing in the cloud.

Before initiating secure communication with the Nitro Enclave, the system performs Remote Attestation (RA). RA is a process by which the Enclave proves to an external party that it is operating in a trusted, untampered state. During RA, the Enclave generates a Platform Configuration Register (PCR) and obtains a signed Attestation Document through the Nitro Hypervisor.

This Attestation Document contains information such as the hash of the Enclave's launch code and the Enclave's public key used for signing. An external attestation server verifies the authenticity of this document to ensure the integrity and trustworthiness of the Enclave. In the full design of the proposed architecture, remote attestation is used to verify the integrity of the Nitro Enclave before processing authentication requests. This verification enables the system to detect spoofing attempts in which an attacker impersonates a legitimate Enclave.

3.2 System Architecture

Fig. 2 shows the overall architecture of the prototype identity verification system developed in this study. The system was specifically built to reproduce the verification flow, and the database is implemented using MySQL.

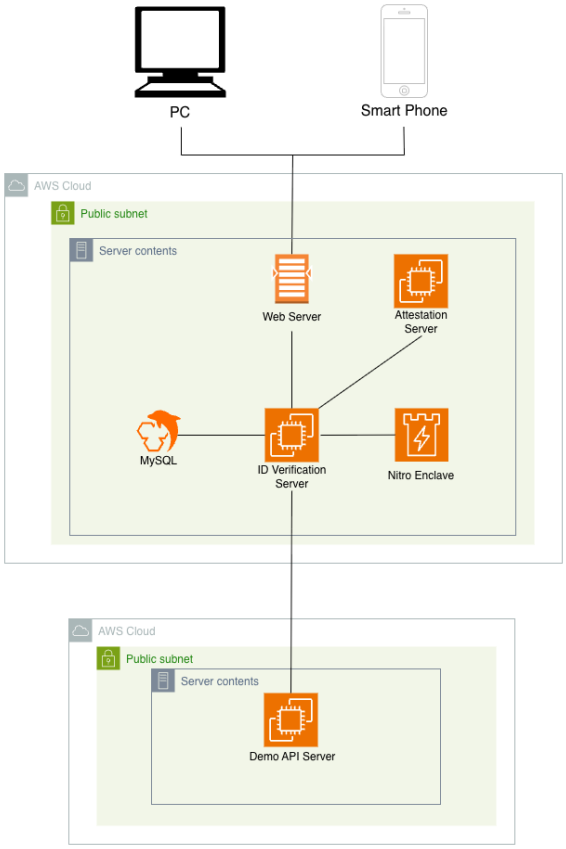

Figure 2: Overall Architecture of the Proposed System

The roles of each component are summarized as follows:

- **PC / Smartphone:** Client devices operated by users to initiate the authentication process via a web interface.
- **Web Server:** Accepts requests from users, serves as a frontend, and relays authentication requests.
- **ID Verification Server:** Backend server responsible for communication and control with Nitro Enclave.
- **Nitro Enclave:** A secure isolated environment that provides a TEE. It performs highly confidential processing such as authentication token validation.
- **Attestation Server:** A server used to verify the integrity of the TEE environment during execution.
- **Demo API Server:** A test API server that emulates the identity verification API provided by the Digital Agency. It is built based on the documented responses published by the official Digital Agency website.
- **Database:** A student information database managed by the service provider (e.g., university).

3.3 System Flow

Figure 3 shows the overall flow of the identity verification process in the proposed system. Each step is explained below.

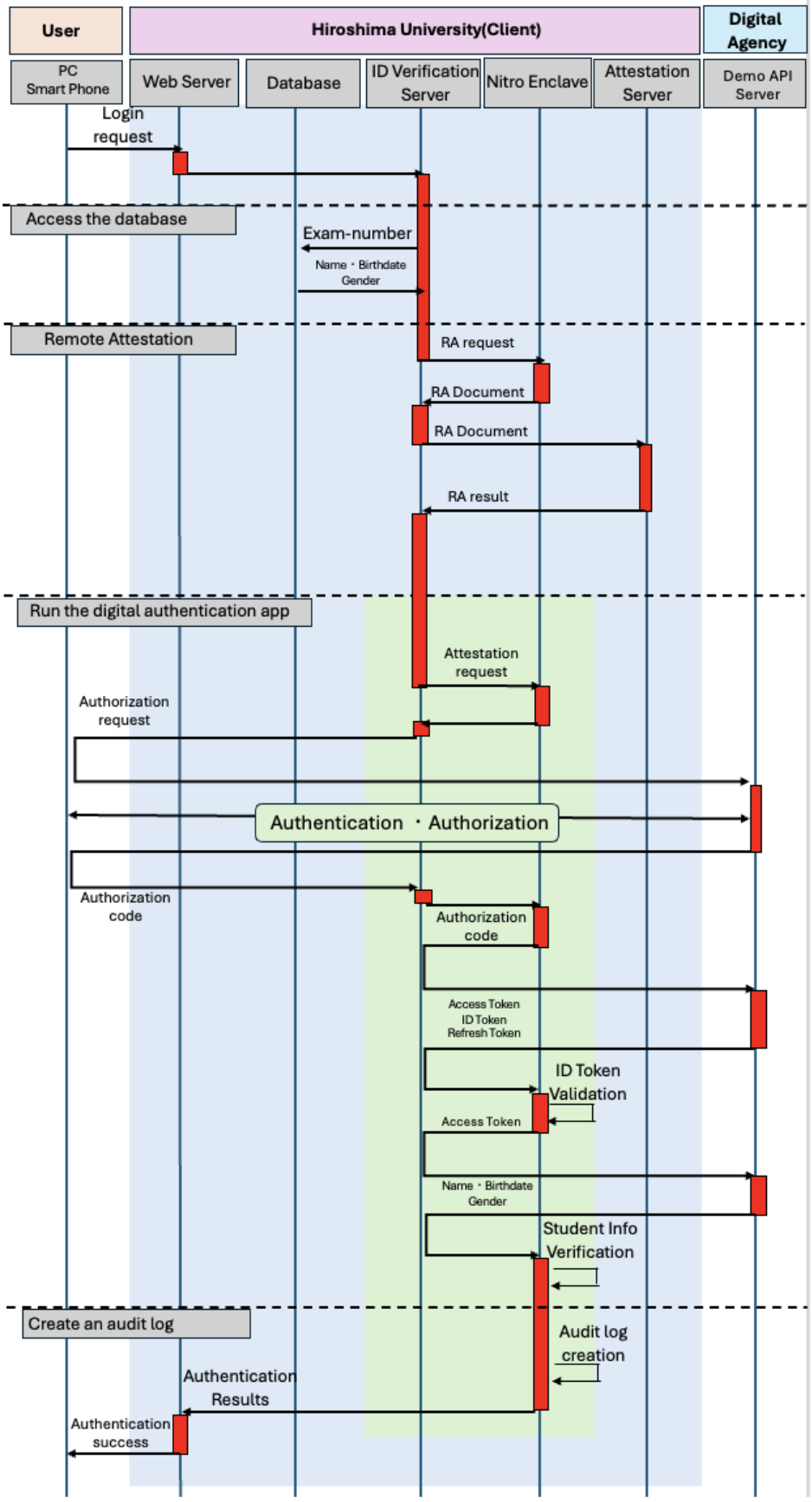

Figure 3: System Flow

1. **Login Request** The user accesses the online identity verification page from their device and sends a login request to the ID Verification Server. At this stage, the user is required to enter their examinee number and email address.

2. **Database Access** Based on the submitted examinee number, the system retrieves the corresponding student information (name, date of birth, and gender) from the database.

3. **Authorization Request** The ID Verification Server redirects the user's browser or application to the Demo API Server and sends an authorization request.

4. **Authentication and Authorization** Between the user's device and the Demo API Server, authentication and authorization are performed using the electronic certificate embedded in the My Number card. The Digital Authentication App adopts multi-factor authentication that combines a knowledge factor (a 4-digit PIN) and a possession factor (reading the My Number card via the NFC function of a smartphone).

   Furthermore, Hiroshima University specifies the `scope` (name, date of birth, gender, and openid) in the authorization request to access the user's basic information. The user is prompted to provide consent for the disclosure of this information. Through this process, it is verified that the user is the legitimate holder of a My Number card issued and guaranteed by the government.

5. **Authorization Response** The Demo API Server issues an authorization code and redirects the user's browser or application back to the ID Verification Server.

6. **Token Request and Token Response** Using the obtained authorization code, the ID Verification Server sends a request to the Demo API Server and obtains an access token, ID token, and refresh token.

7. **ID Token Verification** The ID Verification Server verifies the signature and claims of the ID token and accepts it as the authentication result of the user.

8. **UserInfo Request and Response** If the user has provided consent during the authentication and authorization process, the ID Verification Server can use the access token to retrieve the user's name, date of birth, and gender from the Demo API Server.

9. **Student Information Verification** The system compares the student information (name, date of birth, and gender) retrieved from the database with the information obtained from the Demo API Server.

10. **Authentication Result** Based on the verification result returned by the ID Verification Server, the Web server presents the result to the user's device. If authentication is successful, a success page is displayed. The success page is intended to provide initial credentials such as an initial ID and password.

3.4 Design of the Audit Log System

In the proposed system, authentication and authorization processing, as well as access to UserInfo, are confined within a TEE, preventing unauthorized access by the operating system or administrators. However, several limitations remain.

First, in a basic configuration, UserInfo can be accessed multiple times within the validity period of access and refresh tokens. Even if access is intended to occur only during identity verification, it is difficult to externally verify whether this constraint is enforced in practice. Moreover, risks arising from implementation and operation, such as side-channel attacks, cannot be completely eliminated. Second, users have no mechanism to confirm ex post facto that such access remains within the consented scope. Third, system administrators cannot demonstrate that access is strictly limited to legitimate purposes within the permitted scope of identity verification.

To reduce the risk of administrator misconduct and enable ex post verification by users, we propose a full audit-log system architecture that records system behavior as verifiable evidence. The system adopts the following three-layer architecture:

- **Log generation and signing based on TEE:** Logs are generated within the Trusted Execution Environment (TEE) and are digitally signed.
- **Immutable storage of original logs using WORM storage:** The original logs are stored in an immutable form in storage equipped with Write Once, Read Many (WORM) functionality. They are basically non-public, but can be accessed for verification purposes.
- **Recording minimal logs on a blockchain:** Only the minimum necessary information required for public verification is recorded on a blockchain as minimal logs.

By combining TEE, WORM storage, and blockchain, the proposed system achieves tamper detection, non-repudiation, and public verifiability without relying on a single component.

The current prototype implements log generation and signing within the Enclave, while the broader verification flow, including WORM storage, blockchain anchoring, and user-side verification, is not fully implemented or evaluated.

Two Types of Audit Log Structure The audit log adopts a two-layer structure consisting of original logs and minimal logs. Original logs are detailed, human-readable records stored for system administrators and used for investigating abnormal processing and identifying the causes of failures or incidents. In contrast, minimal logs contain only essential information and are recorded on the blockchain to provide tamper detection and publicly verifiable evidence of the original logs.

This separation reduces gas costs and processing overhead while preventing the exposure of sensitive and personally identifiable information on the blockchain, thereby achieving both privacy protection and evidentiary capability. By linking minimal logs to the corresponding original logs, users can verify ex post whether access to UserInfo has exceeded the permitted scope, including the allowed timing and number of accesses.

The formats of the Original Log and Minimal Log currently designed in this study are shown below. Logs are recorded in JSON format.

- **Original Log**

  Body Main information of the original log

  Timestamp Time
  LogNumber Log number
  Severity Log level (info / warning / critical)
  Operator identity ID of the entity that performed the operation
  Operator context IP address, device ID, privileges
  Invocation method Method of invocation
  Result Processing result (success / failure, error reason)
  TEE attestation RA Attestation Document
  Previous-record verification Hash of the previous log

  Signature Signature information for the original log (Body)

  Signature Signature data of the Body
  Signature algorithm Signature algorithm
  Signature public key Public key for verification

- **Minimal Log**

  Timestamp Time

  LogNumber Log number

Original verification Hash value of the original log

Original reference Reference to the storage location of the original log

Hash algorithm Hash algorithm

System Architecture and Procedures Figure 4 shows the proposed audit log system architecture. This section describes the procedures from log generation to recording, as well as the user verification flow. Note that the following procedures describe the intended verification workflow of the full audit-log architecture and are not fully implemented in the current prototype.

First, the flow from log creation to recording after the completion of OIDC processing is as follows.

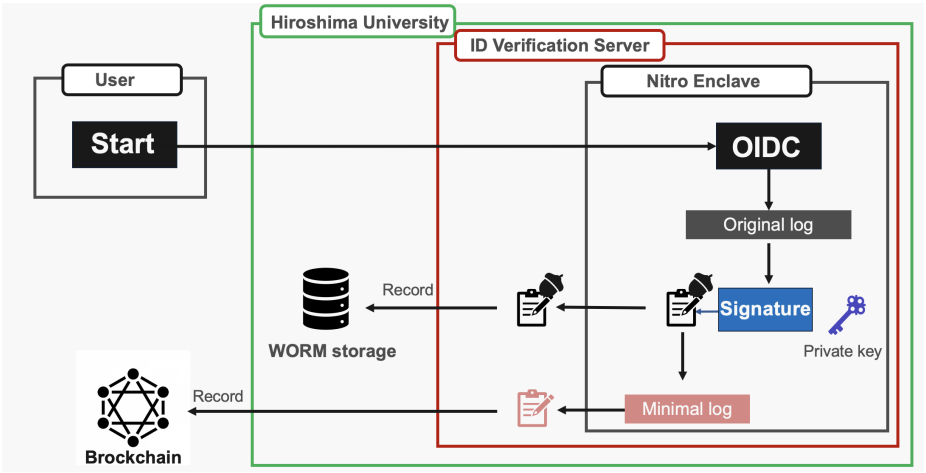

Figure 4: The proposed audit log system architecture

1. **Completion of OIDC processing** The process starts from the user's login request (Start), and finally the OIDC processing is completed within the Enclave.
2. **Creation and signing of the original log** Based on the result of the OIDC processing, the original log is created, and it is signed with the private key inside the Enclave.
3. **Creation of the minimal log** The hash value of the entire signed original log is calculated, and the minimal log is created.
4. **Recording** The ID Verification Server receives the original log and minimal log from the TEE. The original log is recorded in WORM storage, and the minimal log is recorded on the blockchain.

Next, the user verification flow is as follows. As a prerequisite for verification, the code executed within the Enclave and PCR0 are publicly disclosed, and the security of the code is assumed to be guaranteed in advance.

1. **Mapping using the minimal log** Among the minimal logs recorded on the blockchain and the original logs stored in WORM storage, the log whose LogNumber and Timestamp match is identified.
2. **Hash verification** A hash value is calculated from the identified original log, and it is verified whether this hash value matches the hash value recorded in the minimal log.
3. **Consistency verification of adjacent original logs** Using the Previous-record verification information contained in the original log, it is verified that there is no inconsistency or missing record in the chronological sequence of the preceding and succeeding original logs.

4. **Signature verification** The Signature contained in the original log is verified using the Signature public key, and it is confirmed that the Body of the original log has not been tampered with.

5. **Verification of the Attestation Document** Based on the RA Document verification procedure described in Section 2.4, the Remote Attestation Document is verified. Here, the validity of the signature generated by AWS Nitro Enclaves and the consistency of the PCR0 value with the expected value are verified, thereby confirming that the log was generated and signed within the Enclave.

Advantages and Disadvantages of the Proposed Audit Log System The proposed audit log system adopts a multi-layer architecture in which audit logs are recorded in both WORM storage and a blockchain. This design provides several advantages while introducing certain trade-offs.

A key advantage is that it enables verifiable and tamper-evident audit logging. WORM storage preserves original logs in a non-deletable and non-overwritable form, ensuring the integrity of detailed audit records. However, since WORM storage is managed by the service provider, it cannot serve as a complete trust anchor for users. To address this limitation, the blockchain records only hash values and minimal identifying information, providing tamper resistance and public verifiability without exposing sensitive data. By treating the blockchain as an external trust anchor, users can verify the consistency between the blockchain records and the original logs stored in WORM storage.

By combining these mechanisms, the system achieves integrity, non-repudiation, and user verifiability, which are difficult to realize using a single method alone. In particular, users can independently verify that their personal information and authentication tokens have not been used beyond the scope of identity verification, thereby reducing the need to rely on the operational trust of the service provider. From the administrator's perspective, the system provides objective evidence that the system operates as designed. This supports accountability by enabling clear explanations that no unauthorized use has occurred and improves operational transparency and trustworthiness.

The disadvantages of this multi-layer architecture include increased complexity in log recording and verification, as well as higher operational costs. Therefore, administrators must decide whether to adopt only WORM storage or combine it with blockchain by balancing trade-offs among trust, cost, and system complexity.

3.5 Prototype Implementation

This section describes the implementation of the prototype system constructed based on the proposed architecture. In this study, we explicitly distinguish between the complete architecture and the implemented prototype. The complete architecture includes Remote Attestation (RA) and audit-verification workflow in addition to Enclave-based OIDC processing. However, the current prototype focuses on secure OIDC token processing and signed audit-log generation within the Enclave. Since the execution environment required for RA has not yet been implemented, RA verification, the Attestation Server, and the corresponding attestation flow are excluded from the implementation scope. Therefore, the prototype should be understood as a partial implementation, and the present evaluation measures only the components implemented in this prototype.

3.5.1 Implementation Overview

In this study, in order to construct a prototype system based on the proposed method, the following programs were newly implemented. Figure 5 shows the placement and roles of each program. The ID Verification Server and the Enclave are deployed on an EC2 instance operated by Hiroshima University, while the demo API server operates on a separate EC2 instance. Table 1 shows the development environment.

- **Web application program:** Deployed on a web server, it operates as a front-end that receives identity verification requests from users. It obtains input information (examination number and email address) and forwards it to the ID Verification Server.

Table 1: Development Environment

Item	ID Verification Server	Demo API Server
Instance type	m5.large	t2.large
CPU	4 vCPU	2 vCPU
CPU (Enclave)	2 vCPU	–
RAM	16 GiB	8 GiB
RAM (Enclave)	0.5 GiB	–
OS	Amazon Linux 2023	Ubuntu
Volume size	30 GiB	8 GiB
Programming language	Go 1.24.4	Go 1.24.4
Database	MySQL 8.0.44	–

- **Outside-Enclave processing program:** Deployed on the ID Verification Server, it performs integration with the database, controls the OIDC flow, communicates with the demo API server, and communicates with the Nitro Enclave.
- **Inside-Enclave processing program:** Deployed within the TEE, it executes the OIDC process according to the number received from the Outside-Enclave processing program. After completion of the processing, confidential information (tokens and personal information) is discarded. Subsequently, audit logs are generated.
- **API processing program:** Deployed on the demo API server, it reproduces the behavior based on the API Reference published by the Digital Agency.
- **Database:** Attribute information required for identity verification (examination number, name, date of birth, and gender) is managed on MySQL.

Newly Developed detail Programs Figures 6–7 illustrate, in chronological order, the processing details of each program newly implemented in this study. In this section, the overall system processing flow is described in textual form to complement these figures.

First, the user inputs an examination number and email address through the Web application program and submits an identity verification request. The Web application program operates as a front-end on the Web server, receives user input, and forwards this information to the Outside-Enclave processing program.

Next, the Outside-Enclave processing program accesses the Database using the received examination number. In the Database, the corresponding name, date of birth, and gender are retrieved using the examination number as a key, and the results are returned to the Outside-Enclave processing program. At this stage, personal information is temporarily obtained outside the Enclave; however, in subsequent authentication processing, it is transferred into the Enclave.

After that, the Outside-Enclave processing program establishes communication with the Inside-Enclave processing program using vsock. At this time, a variable n for identifying the processing step to be executed is passed to the Inside-Enclave processing program, thereby controlling each stage of the OIDC flow. The Outside-Enclave processing program is responsible for communication mediation and Database access, and does not execute highly confidential processing itself.

Inside the Enclave, the Inside-Enclave processing program executes authentication processing based on OIDC. Specifically, this includes generation of the authorization request, creation and verification of required parameters such as state and nonce, verification of the authorization code, and acquisition and validation of tokens. Session identification and management are also performed inside the Enclave, adopting an extended design based on the state parameter. These implementation considerations are described later.

When communication with an external API is required in the OIDC flow, the communication request generated by the Inside-Enclave processing program is forwarded to the API processing program via the Outside-Enclave processing program. The API processing program operates in

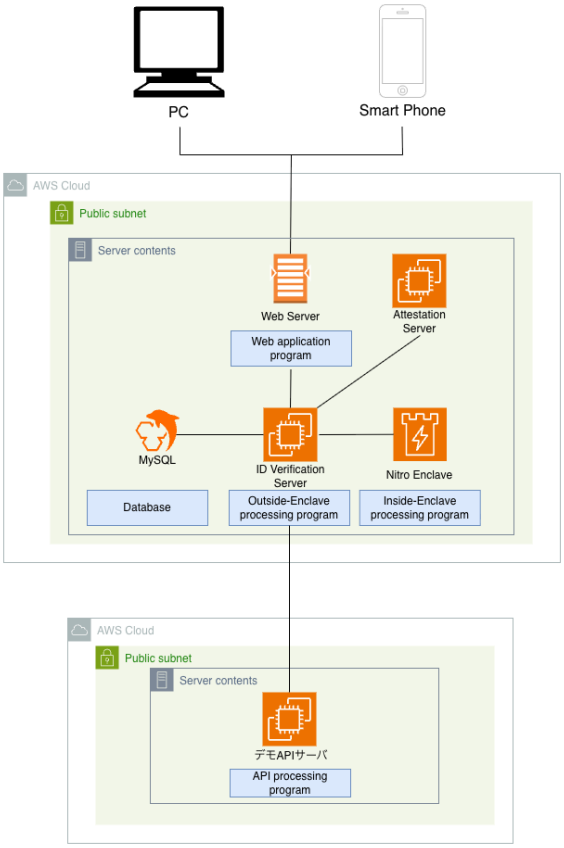

Figure 5: Program Deployment

accordance with the official specifications of the Digital Authentication API and handles processes such as authorization code issuance, token response, and UserInfo response. The response results are then returned to the Inside-Enclave processing program.

Finally, the Inside-Enclave processing program compares the acquired user information with the information retrieved from the Database to determine whether identity verification succeeds or fails. Along with the authentication result, an audit log indicating the processing trace is generated, digitally signed, and stored. The result is then returned to the Web application program via the Outside-Enclave processing program, and the authentication outcome is presented to the user.

3.5.2 Implementation Considerations

This section describes the implementation considerations and challenges encountered during development. As Nitro Enclaves are Docker-based, their development process is similar to building a standard container.

State Management and Tamper Resistance In this implementation, the OIDC state is used as a session identifier to associate authorization requests and responses. When sending an authorization request, multiple values, such as state, Code Verifier, and Nonce are generated and must be retained until the authentication process is completed.

However, communication between the ID Verification Server and the Nitro Enclave is performed on a per-request basis. Once processing is complete, values generated inside the Enclave cannot be referenced in subsequent processing. Unlike typical Web applications, the architecture in this study does not provide a persistent session management mechanism.

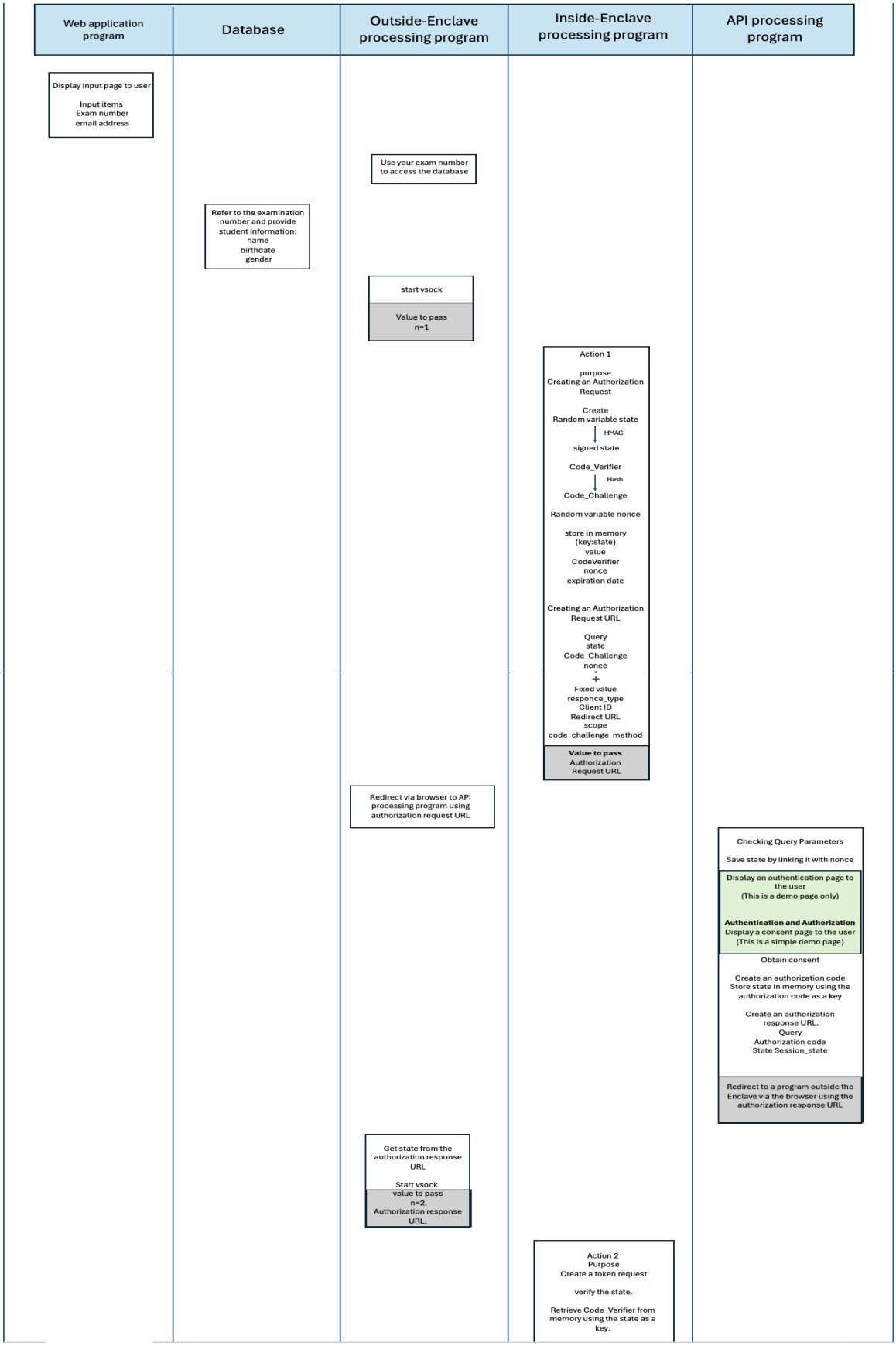

Figure 6: program detail flow1

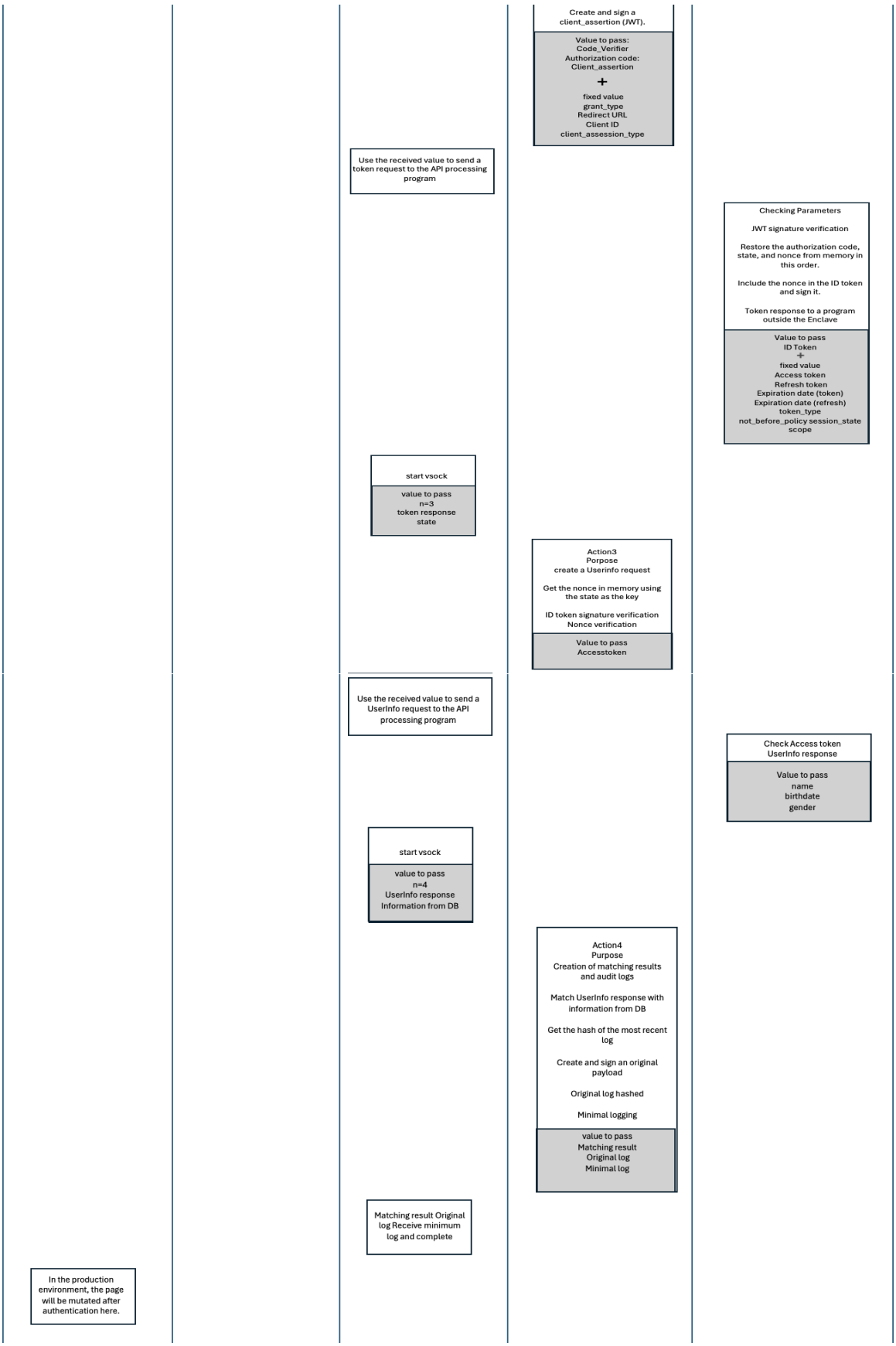

Figure 7: program detail flow2

To address this limitation, temporary information such as Code Verifier, Nonce, expiration time, and student information is stored in the memory of the ID Verification Server using state as the key, and passed to the Enclave as needed. As shown in Figure 8, a Go map structure is used to associate the state with the corresponding values. The Code Verifier is retrieved using the state contained in the authorization response, and the Nonce included in the ID token is verified.

At the time of processing the token response, the state itself is not included. However, by maintaining the state within the same processing function on the ID Verification Server, the stored information can be reused for verification inside the Enclave. This design depends on the state maintained by the ID Verification Server, and the development of a dedicated session management mechanism remains a future challenge.

In addition to these implementation constraints, the state is also a potential target for tampering. In a cloud environment, insiders with host OS or administrative privileges can observe or modify memory and logs. If the state is tampered with, the correct session information cannot be retrieved, and token requests may fail, potentially disrupting the authentication flow.

To mitigate this risk, the state is not treated as a simple random value. Instead, it is protected using an HMAC computed with a secret key stored only within the Enclave, forming a signed state verifiable inside the Enclave. Specifically, a JSON payload containing a random value, issuance time, and expiration time is generated, and an HMAC-SHA256 is computed over this payload, with the first 128 bits used as the MAC value.

Upon receiving the callback, the MAC is recalculated within the Enclave and verified together with the expiration time. If verification fails, the state is considered invalid, and the session is not restored. The expiration time also prevents replay attacks.

With this approach, even if the state is observed or tampered with on the host side, a valid HMAC tag cannot be generated outside the Enclave. Therefore, attackers cannot arbitrarily create a valid state to establish a session. Moreover, since OIDC allows the state to be treated as a random value, extending it with HMAC protection does not affect protocol compatibility as long as randomness and uniqueness are preserved.

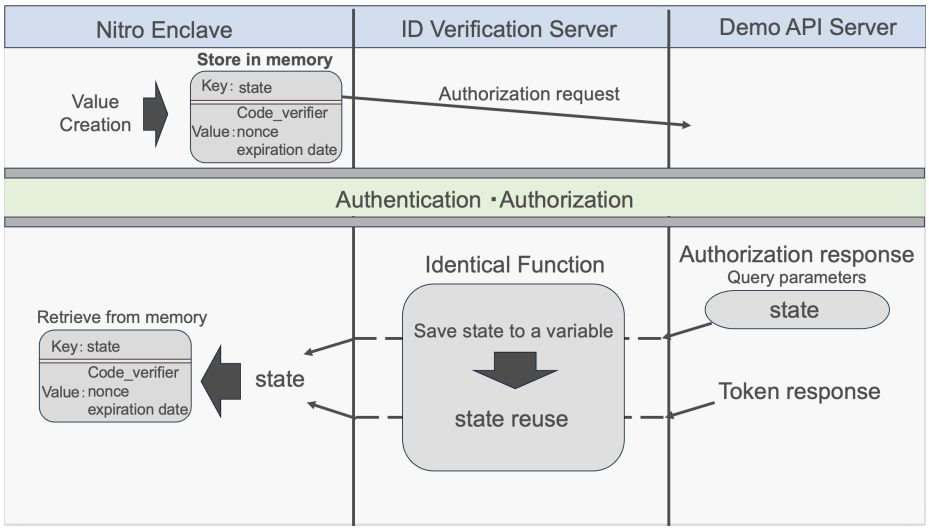

Figure 8: Proposed solution in this implementation

4 Related Work

Since the Digital Authentication App was developed in Japan, this study focuses on research and use cases related to Trusted Execution Environments (TEE) and the Digital Authentication App within Japan. Representative related studies are summarized in Table 2.

Table 2: Comparison with Related Work

Study	**Target**	**Technology**	**Logging**	**Difference from This Work**
Iwano et al.	IoT IDS	Keystone	No	Device-side IDS protection
Kitayama et al.	Password evaluation	Intel SGX	No	Limited to password verification
Itokawa et al.	Digital student ID	Verifiable Credentials	No	Application-layer focus
Shepherd et al.	Constrained devices	ARM TrustZone	Yes	Device-side logging
Yanagi et al.	Edge security	IBE + PRE	No	Encrypted data delegation
This Work	Digital Authentication App	Nitro Enclave	Yes	Server-side OIDC token protection with audit logging

TEE has been widely studied as a technology for protecting processes that require confidentiality and integrity. Iwano et al. [6] proposed a method to protect an intrusion detection system (IDS) for IoT devices within a TEE, thereby preventing attackers from disabling the IDS. While their work focuses on device-side security for IoT environments, our study differs in that it targets server-side protection of OIDC token processing.

Kitayama et al. [7] investigated a method for securely evaluating user passwords on an external server by utilizing a TEE, preventing unauthorized access to sensitive information by server administrators. In contrast to their study, which is limited to password evaluation, our research protects the entire workflow of identity verification processing.

Itokawa et al. [5] examined an identity verification method for issuing digital student IDs using Verifiable Credentials and the Digital Authentication App. While their study primarily focuses on application-level use cases, our research emphasizes enhancing the security of authentication token processing by integrating TEE.

Shepherd et al. [8] proposed EmLog, a tamper-resistant logging system for constrained devices using ARM TrustZone. While EmLog mainly targets device-side log protection, our study utilizes Nitro Enclave in a cloud environment to combine OIDC token protection with tamper-resistant logging.

Yanagi et al. [9] proposed a modular security framework for edge computing and demonstrated a method to securely delegate encrypted data between IoT devices and edge nodes by combining identity-based encryption (IBE) and proxy re-encryption (PRE). Although their work does not directly address log aggregation or token protection, the concept of securely handling encrypted data in distributed environments may be applicable to future challenges in aggregating audit logs from multiple Enclaves.

Although these studies improve confidentiality and integrity in specific contexts, to the best of our knowledge, no prior work has specifically targeted server-side OIDC token protection in the context of the Digital Authentication App. In contrast, this study integrates Nitro Enclave with the API of the Digital Authentication App. Furthermore, by combining token protection and audit logging, we propose an architecture that addresses practical risks such as token leakage and insider threats in real operational environments.

5 Evaluation Method and Results

This chapter presents quantitative and qualitative evaluations of the proposed system, and describes the evaluation methods and results.

5.1 Quantitative Evaluation

This section conducts a quantitative evaluation focusing on processing time in order to clarify the performance characteristics of the authentication process in the proposed system.

As evaluation targets, two configurations were prepared: the proposed system using Nitro Enclave (Enclave enabled, state: HMAC-based, audit log: enabled), and a baseline system without TEE (Enclave disabled, state: random, audit log: disabled). Both configurations were implemented in Go and executed on the same ID Verification Server environment.

In this evaluation, the total processing time required for the identity verification flow based on OpenID Connect (OIDC) was measured to quantify the steady-state performance overhead introduced by TEE-based processing. The experiments were conducted under controlled experimental conditions using a demo API server, with requests processed sequentially under three workloads (1, 10, and 100 consecutive requests). For each workload, measurements were repeated multiple times at fixed intervals under identical conditions, and outliers caused by external communication delays were excluded to capture the typical behavior of the authentication path. The evaluation focuses on the processing performed within the identity verification server, including OIDC processing, user data matching, and audit-log generation, and compares configurations with and without TEE. Remote Attestation (RA) and audit-log verification are not included in the quantitative measurements, as they are not implemented in the current prototype. Accordingly, the reported results should be interpreted as measurements of steady-state authentication processing within the Enclave, rather than of the full trust-establishment procedure assumed in the proposed architecture.

Measurement Method Processing time was measured using `time.Now()` from Go's `time` package. Timestamps were obtained at each processing point on the ID Verification Server, and processing time was calculated from their differences.

The measurement start point was defined as the moment when the login request reached the handler function on the ID Verification Server. The measurement end point was defined as the moment when the authentication result and audit log were returned to the ID Verification Server. In this study, Remote Attestation was excluded from performance evaluation, and this process was not included in the measurement interval.

Measurement Segments The breakdown of processing time in the Enclave-enabled configuration is shown below.

Database From receiving the login request to sending the authentication request to the Enclave.

Authorization request generation and transmission From generating the authorization request to sending it by the ID Verification Server.

Start of authorization request relay From receiving the authorization request in the Enclave to starting communication with the demo API server.

Authorization code transfer From receiving the authorization code to passing it to the Enclave.

Authorization code verification and token acquisition From processing the authorization code in the Enclave to receiving the token response.

Token response transfer From receiving the token response to transferring it to the Enclave.

Token verification and UserInfo generation From processing the token in the Enclave to generating the UserInfo request.

UserInfo communication From sending the UserInfo request to receiving the UserInfo response.

Authentication result and audit log generation From processing the UserInfo response to receiving the authentication result and audit log.

These segments include token verification, user information acquisition, and data transfer between inside and outside the Enclave, and therefore reflect the substantive processing time of the entire authentication process.

For each workload (1, 10, and 100 consecutive requests), measurements were conducted three times at 5-second intervals under identical experimental conditions. The experiments were performed for both Enclave-enabled and Enclave-disabled configurations, allowing a direct comparison of processing times to evaluate the baseline performance impact of introducing TEE.

5.1.1 Experimental Results

Figures 9 and 10 show the processing time of each measurement segment for each number of requests. Outliers are excluded.

Figures 11 and 12 summarize the results of three measurements conducted under the 100-request condition, and visualize them using box plots.

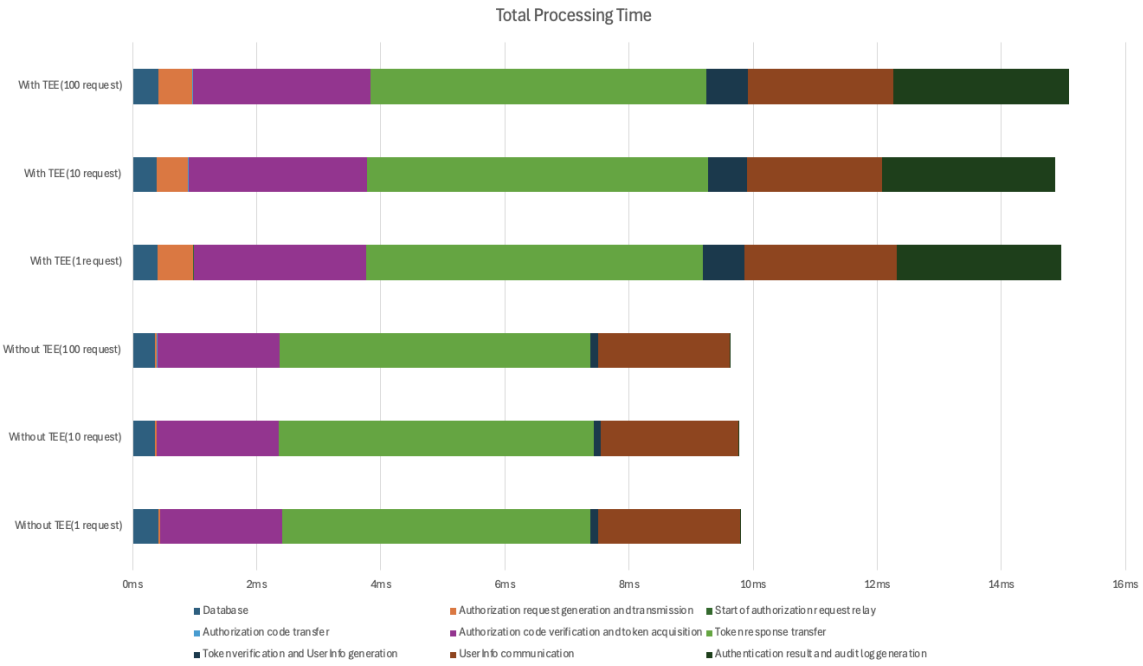

Figure 9: Comparison of OIDC processing time with and without Enclave

Figure 9 shows the difference in total processing time between the Enclave-enabled and Enclave-disabled configurations. Comparing the total time for one request, the Enclave-enabled environment shows an increase of approximately 5 ms compared to the baseline environment. Similar results, with an increase of approximately 5 ms, are observed under the 10-request and 100-request conditions. Within the same configuration, no significant increase in processing time was observed due to an increase in the number of requests. When excluding the segment "authentication result and audit log generation," which differs between the two configurations, the increase is approximately 2 ms. From this result, it can be inferred that the pure OIDC processing overhead introduced by the Enclave is approximately 2 ms, and the remaining difference is attributed to log generation processing. It is also observed that "token response transfer" and "UserInfo communication" account for approximately 60% of the total processing time, indicating that communication dominates the processing cost.

Figure 10 shows the breakdown of processing time between the two configurations. In both configurations, no segment shows a significant increase due to the number of requests. However, in the segments "authorization request generation and transmission," "authorization code verification and token request," "token verification and UserInfo generation," and "authentication result and

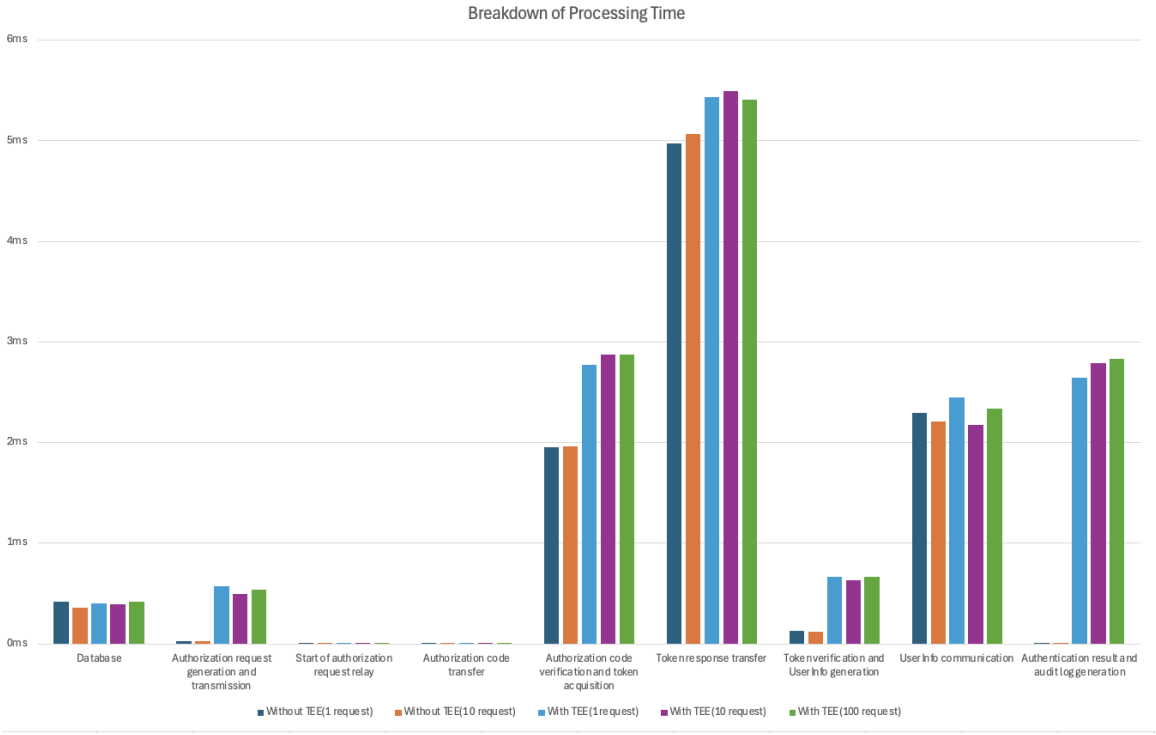

Figure 10: Comparison of OIDC processing time with and without Enclave

audit log generation," an increase of at least 0.5 ms is observed in the Enclave-enabled environment compared to the baseline. For "token response transfer," an increase of approximately 0.5 ms is observed in the Enclave-enabled configuration.

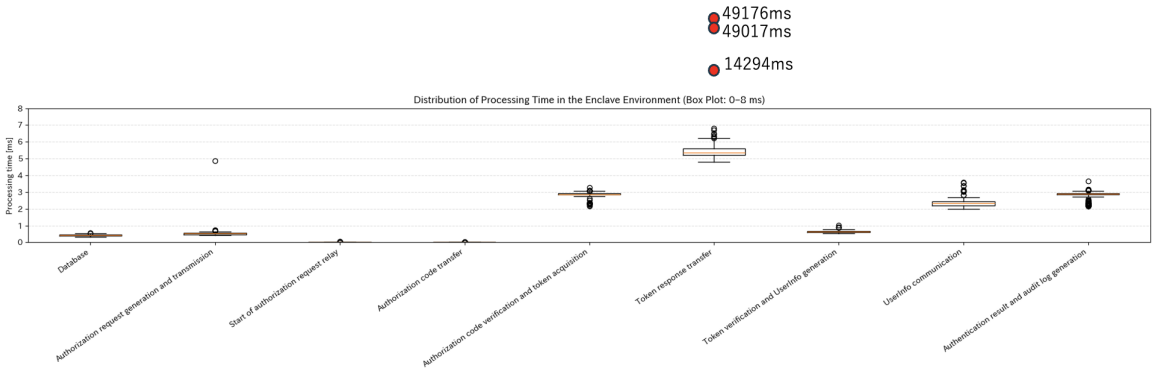

Figure 11: Box plot of the Enclave environment (300 measurements)

Figure 11 compares the variability of processing time between the two configurations. When comparing corresponding segments, processing time variability exceeding 1.0 ms is observed only in "token response transfer" and "UserInfo communication," which are related to communication or the demo API server. Furthermore, these two segments tend to exhibit outliers exceeding 1000 ms. During the experiment, such outliers appeared only when 100 consecutive requests were sent. This indicates that as the number of requests increases, the probability of outliers occurring also increases.

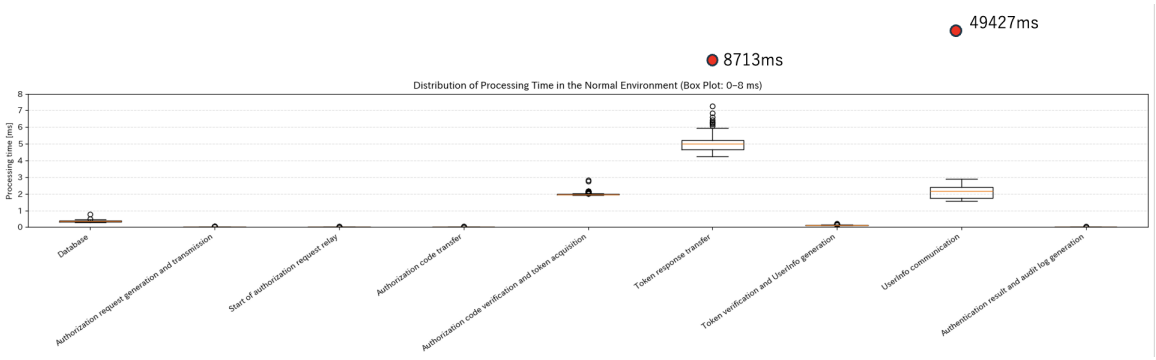

Figure 12: Box plot in normal environment (300 measurements)

5.2 Qualitative Evaluation

In this study, after clearly defining the assumptions and assets, a qualitative security evaluation of the proposed system is conducted based on the STRIDE threat model. STRIDE is a framework that evaluates threats by classifying them into six categories: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege.

5.2.1 Assumptions and Assets

Assumptions

- All communications between the Enclave and external services are protected using standard TLS with certificate validation.
- The code inside the Enclave is guaranteed to be secure, and confidential information inside the Enclave is configured to be discarded after authentication.
- The Digital Authentication App API server operates correctly and is not compromised at the time of request processing.

These assumptions do not eliminate the corresponding threats but define the scope of the proposed system. Threats outside this scope are mitigated by existing operational controls or are addressed as future work.

Assets Confidential information in OIDC (authorization code, ID token, access token, refresh token, and name, date of birth, and gender received from UserInfo), and audit logs.

5.2.2 STRIDE Risk Assessment

In this section, based on the STRIDE threat model, a qualitative risk evaluation of the proposed system is conducted.

Spoofing Spoofing is a threat in which an attacker behaves by impersonating a legitimate user, server, or system component. In the target system of this study, the following points are considered.

- **Impersonation of the ID Verification Server:** In the Digital Authentication API, since the redirect URL is fixed before the authentication request, redirection by a fake server is difficult.
- **Impersonation of the Enclave:** By verifying the PCR values of the Remote Attestation Document at Enclave startup, an illegitimate execution environment can be detected.

- **Impersonation operations by an administrator:** Even an administrator cannot directly reference the inside of the Enclave, and simultaneous modification of multiple elements would be required, therefore a realistic attack is difficult.

Tampering Tampering is a threat in which an attacker illegally rewrites internal system data or processing results. In the target system of this study, the following points are considered.

- **Tampering with authentication tokens and personal information:** Authentication tokens and personal information are all processed inside the Enclave, and are not stored in plaintext on the host OS. Therefore, even if the host OS or administrator is compromised, it is difficult to tamper with such information.
- **Tampering with audit logs:** Audit logs are generated inside the Enclave and then hash-chained and signed, and stored in WORM storage. As a result, logs cannot be deleted or rewritten after storage. Furthermore, since the hash values of logs are recorded on the blockchain as minimal logs, ex post tampering can be detected.

Repudiation Repudiation is a threat in which a user, administrator, or system component can deny the execution of its own operation or process after the fact. In the target system of this study, the following points are considered.

- **Repudiation of operations by an administrator:** Authentication processing and audit logs are generated inside the Enclave, and logs include identifiers and contextual information indicating the operating entity. Furthermore, since the entire log is signed with a secret key inside the Enclave, it is difficult for an administrator to claim afterward that the operation was not performed.
- **Repudiation through log tampering:** Audit logs are stored in WORM storage, and deletion or rewriting after storage is impossible. In addition, since the hash values of logs are recorded on the blockchain as minimal logs, repudiation through subsequent tampering can be detected.
- **Repudiation regarding the legitimacy of the execution environment:** Each audit log contains a Remote Attestation Document, and users can verify that the signing key was generated and used inside a legitimate Enclave. Therefore, it becomes difficult to claim that the processing was not executed in a legitimate execution environment.

Information Disclosure Information disclosure is a threat in which an entity that is not originally permitted access obtains confidential or personal information. In the target system of this study, the following points are considered.

- **Leakage of authentication tokens and personal information:** In the proposed method, authentication tokens and personal information used for identity verification are all processed inside the TEE, and are configured so that they cannot be directly referenced by the host OS or administrators. Therefore, the risk of information leakage through a server administrator or compromised OS can be reduced.
- **Leakage from audit logs:** Audit logs are managed separately into original logs containing detailed information and minimal logs containing only minimal information. On the blockchain, only the minimal information necessary for verification, such as hash values, is recorded, and personal information or sensitive information is not disclosed. However, if there are vulnerabilities in access control of WORM storage, original logs may be leaked.
- **Leakage via communication paths:** In this study, encryption and authentication regarding communication paths are assumed to be properly implemented, and eavesdropping or tampering of the communication path itself is outside the scope of this evaluation.

Denial of Service Denial of Service is a threat in which an attacker reduces system availability, making legitimate users unable to use the service. In the target system of this study, the following points are considered.

- **Service stoppage due to large numbers of requests:** An attacker may send a large number of requests to the ID Verification Server, causing processing delay or service stoppage. In the proposed method, it is not possible to completely eliminate the impact that processing inside the TEE becomes a bottleneck. However, since Nitro Enclave shares computational resources with the parent instance, a certain degree of mitigation is possible through control of resource allocation.
- **Concentrated load on Enclave startup and processing:** If requests are concentrated on Enclave startup or internal processing, processing performance may temporarily decrease. However, the evaluation target of this study is mainly confidentiality and integrity, and countermeasures regarding availability are left as future work.
- **Impact due to exhaustion of log storage area:** Due to large amounts of log generation, if storage resources are exhausted or blockchain gas costs become enormous, normal processing may not be able to continue. In this study, such degradation of availability is assumed to be addressed through operational control and resource management.

Elevation of Privilege Elevation of Privilege is a threat in which an attacker executes operations that were not originally permitted by obtaining higher privileges. In the target system of this study, the following points are considered.

- **Access to confidential processing through abuse of administrator privileges:** In general, a person with administrator privileges can perform extensive operations on processes and memory on the server. However, in the proposed method, authentication processing and handling of confidential information are executed inside the TEE, therefore even a person with administrator privileges cannot directly access the processing contents or data inside the Enclave.
- **Privilege escalation attack via the host OS:** Even if an attacker succeeds in privilege escalation on the host OS, since the Enclave is virtually isolated by the Nitro Hypervisor, it is difficult to gain unauthorized access to processing or key information inside the Enclave.

Summary of the Qualitative Evaluation In this section, based on the STRIDE threat model, the method in which OIDC is processed on a normal server (without TEE) and the method in which OIDC is processed inside the Enclave (with TEE) were compared, and a qualitative security evaluation of the proposed system was conducted. Table 3 shows the comparison results organizing threat resistance in both methods.

First, as a threat for which both methods have resistance, Spoofing can be mentioned. This is because, due to the constraint of pre-registered redirect destinations in the Digital Authentication App and the verification mechanisms using state and nonce provided in OIDC, impersonation of the authentication flow by an external attacker can be prevented in both methods.

Next, as threats for which only the method with TEE has resistance, Tampering, Repudiation, Information Disclosure, and Elevation of Privilege can be mentioned. In the method without TEE, when assuming administrators or a compromised host OS as attackers, it is difficult to completely prevent reference or tampering of authentication tokens and personal information, and repudiation of processing results or operation history. On the other hand, in the method with TEE, by limiting authentication processing and confidential information to inside the Enclave, verifying the execution environment by Remote Attestation, and storing signed audit logs in combination with WORM storage and blockchain, resistance to these threats is improved.

On the other hand, there are no threats for which only the method without TEE has resistance. In other words, no newly lost resistance due to the introduction of TEE was confirmed.

Table 3: Comparison of STRIDE Threat Resistance with and without TEE

Threat Category	Without TEE	With TEE (Proposed System)	Reason / Remarks
Spoofing	◯	◯	In both methods, impersonation of the ID Verification Server from external sources can be prevented due to the OIDC specification and redirect URI constraints. On the other hand, in the case with TEE, the legitimacy of the execution environment can also be verified by Remote Attestation.
Tampering	×	◯	Without TEE, tokens and processing results in memory may be tampered with by an administrator or a compromised OS. With TEE, confidential information is processed only inside the Enclave, ensuring tamper resistance.
Repudiation	×	◯	Without TEE, log tampering or repudiation of operations cannot be completely prevented. With TEE, repudiation after the fact becomes difficult due to audit logs signed inside the Enclave and stored using WORM and blockchain.
Information Disclosure	×	◯	Without TEE, there is a risk that authentication tokens and personal information may be leaked by administrators or a compromised OS. With TEE, confidential information is not exposed outside the Enclave, reducing the risk of leakage.
Denial of Service	×	×	Availability degradation due to large numbers of requests is difficult to prevent in both methods.
Elevation of Privilege	×	◯	Without TEE, acquisition of administrator privileges enables access to confidential processing. With TEE, the Enclave is isolated even from administrators by the Nitro Hypervisor.

Regarding Denial of Service, both methods do not have sufficient resistance, and availability may decrease due to large numbers of requests or concentrated load. In particular, in the method with TEE, since the Enclave becomes an additional processing point, there is a possibility that it becomes a bottleneck when parallel requests are concentrated. This point is from a different perspective than confidentiality and integrity, which are the evaluation targets of this study, and is left as future work.

From the above, it can be evaluated that the proposed method reduces risks such as tampering, repudiation, information disclosure, and elevation of privilege, including internal threats that could not be sufficiently addressed in conventional OIDC server implementations, and achieves effective security enhancement from the perspectives of confidentiality, integrity, and non-repudiation.

6 Discussion

This chapter discusses the results of the quantitative evaluation, the qualitative evaluation, and implementation issues.

6.1 Quantitative Evaluation

6.1.1 Factors Behind the Performance Difference Caused by Introducing the Enclave

From the results shown in Figures 9 and 10, it was confirmed that the total processing time in the Enclave environment increases by approximately 5 ms compared to the normal environment. On the other hand, when excluding the authentication result generation and audit log generation processes, the difference is approximately 2 ms, indicating that the pure OIDC processing overhead caused by introducing the Enclave is limited.

The main factors behind this performance difference include the memory protection mechanism associated with processing inside the Enclave, context switching due to vsock communication with the parent instance, and signature and hash computation during audit log generation. In particular, the authentication result and audit log generation process is executed only in the Enclave environment, and therefore becomes a major factor in the difference in total processing time.

6.1.2 Bottleneck Analysis Based on the Processing Breakdown

From Figure 10, it can be seen that in both the Enclave environment and the normal environment, token response transfer and UserInfo communication are dominant processes accounting for approximately 60% of the total processing time.

From these results, it is suggested that the performance bottleneck in this method is not the processing inside the Enclave itself, but strongly depends on communication with external servers and the response time of the demo API server.

6.1.3 Discussion on Variability and Outliers in Processing Time

Figures 11 and 12 visualize the variability of processing time using box plots.

In many processes, the interquartile range remains within less than 1.0 ms, and no clear tendency was confirmed that introducing the Enclave significantly increases variability.

On the other hand, outliers exceeding 1000 ms were confirmed in token response transfer and UserInfo communication. These outliers were observed only when 100 consecutive requests were sent, and are considered to be mainly caused by network delay or increased load on the demo API server. Therefore, these outliers are not an Enclave-specific problem, but a phenomenon caused by communication and dependency on external services when the number of requests increases.

6.1.4 Practical Impact and Acceptability in Real Operation

The increase in total processing time due to introducing the Enclave confirmed in this experiment is approximately 5 ms. In real operation such as on-campus procedures and online identity verification, this is considered to be a delay that users cannot perceive.

In addition, the overhead of OIDC processing alone is approximately 2 ms. Given that communication processing is dominant, the performance degradation due to introducing the TEE remains within a sufficiently acceptable range. From the above, it can be evaluated that the proposed method achieves high security while maintaining performance characteristics that do not cause problems in real operation. On the other hand, the integration and evaluation of Remote Attestation (RA) are left as future work. RA is essential for establishing trust in the Enclave before sensitive processing begins and for fully mitigating spoofing attempts in practical deployments. However, attestation is typically performed during system initialization or session establishment rather than in the per-request critical path. Therefore, although RA may introduce additional latency depending on the attestation mechanism and deployment environment, its omission in the current prototype does not invalidate the steady-state performance results reported in this paper.

6.1.5 Impact of TEE Startup and Termination on Performance

In this evaluation, processing time was measured with the Nitro Enclave started in advance, and the overhead associated with TEE startup and termination was not included in the evaluation scope.

It is expected that starting a Nitro Enclave requires several seconds, and the additional cost of Remote Attestation should be considered together with startup overhead when evaluating the initial user experience. However, because these costs are incurred outside the steady-state per-request processing path, they do not directly affect the millisecond-level overhead reported in Section 5.1.

From the above results, the additional delay due to processing inside the Enclave is at most on the order of several milliseconds. Even when considering worst-case values due to the Enclave, it is considered that this does not significantly affect the practicality of the overall authentication system. Although this evaluation is a baseline measurement under sequential processing, even at the current stage, the proposed system is highly likely to be sufficiently usable as an authentication system. In the future, to evaluate scalability, experiments using parallel requests are necessary.

6.1.6 Impact of Blockchain Logging

The present quantitative evaluation does not explicitly include the cost of blockchain operations. In the proposed architecture, only minimal logs, such as hash values and references to original logs, are recorded on the blockchain, while detailed original logs are stored in WORM storage. Since only minimal log data are recorded on-chain and blockchain operations can be performed asynchronously or in batches after authentication processing completes, their impact on the per-request authentication path is limited. Although blockchain itself may introduce higher latency than TEE processing depending on the platform and confirmation model, this overhead does not affect the per-request authentication path and can be effectively mitigated by system design. These considerations indicate that, while blockchain contributes to tamper resistance and public verifiability, its performance overhead can be effectively controlled in a way that preserves the practicality of the proposed system.

6.2 Qualitative Evaluation

This section discusses the qualitative effectiveness of the proposed method based on the results of the STRIDE-based threat analysis and comparative evaluation presented in the previous section.

6.2.1 Improvements

The most significant improvement of the proposed method is that authentication and token processing in OIDC are confined within the TEE (Nitro Enclave). As a result, even if the host OS or server administrator is compromised, the risks of direct access to or tampering with the authorization code, ID token, access token, refresh token, and personal information contained in UserInfo can be reduced.

In addition, by generating the authentication result and audit logs inside the Enclave and storing them with signatures, the integrity of processing results and the capability of ex post verification are improved. This strengthens resistance to insider threats, which was difficult to sufficiently address in conventional OIDC server implementations.

6.2.2 Relationship to STRIDE

From the perspective of the STRIDE threat model, the proposed method shows notable improvements particularly against Tampering, Information Disclosure, Repudiation, and Elevation of Privilege.

These threats are likely to become apparent when administrators or compromised host OS are assumed as attackers. In this method, by limiting confidential processing to the Enclave and enabling verification of execution environment legitimacy through Remote Attestation, higher resistance is achieved compared to the conventional method. On the other hand, for Spoofing, a certain level of resistance is provided regardless of whether TEE is used, due to mechanisms that are standard in OIDC.

6.2.3 Residual Risks

However, the proposed method cannot eliminate all threats. For example, risks due to compromise of user devices, and risks due to specification changes or specification dependency of the Digital Authentication API, are outside the scope of this method and remain as residual risks. If the API server were compromised, the proposed architecture would not be able to prevent the issuance of malicious responses. However, even in such a case, the integrity-protected audit logs generated inside the Enclave enable post-incident investigation and accountability.

In addition, for Denial of Service, it is difficult to completely prevent availability degradation due to excessive requests or concentrated load. In particular, since the use of TEE increases the number of processing points, the Enclave may become a bottleneck when parallel requests concentrate.

6.2.4 Operational Assumptions

To operate the proposed method safely, several assumptions are required. First, Remote Attestation verification must be properly performed, and it must be ensured that only legitimate Enclaves are used. Second, strict management of signing keys and cryptographic keys used inside the Enclave is required to prevent key leakage and misuse.

Furthermore, for the operation of WORM storage and the blockchain that store audit logs, establishing access control and operational policies is also essential.

6.3 Future Work

As future work, extension to a multi-Enclave environment using multiple Enclaves can be considered. To process large-scale authentication requests, scaling and load balancing of Enclaves will be necessary, and a design is required that improves performance while maintaining security.

In addition, countermeasures related to availability, and mechanisms to reduce dependency on external APIs, are also future issues. By addressing these issues, the proposed method is expected to evolve into an identity verification infrastructure more suitable for real operation.

Encryption Within the Parent Instance In this study, attacks by insiders with host OS or administrator privileges are also assumed as threats. Therefore, initially, a method was considered in which data obtained on the parent instance is encrypted and then passed to the Nitro Enclave. However, the APIs provided by the Digital Authentication App do not assume the existence of an Enclave as the entity performing authentication and authorization processing, and assume that decryption is performed on a server outside the Enclave. As a result, it became clear that data encrypted inside the Enclave cannot be appropriately decrypted and used within the existing API flow.

If encryption is performed on the parent instance, it is unavoidable that plaintext data before encryption and cryptographic keys exist, even temporarily, in the memory space of the parent instance. Therefore, there remains a possibility that confidential information is obtained through memory dumps or debugging operations by an attacker with administrator privileges.

From the above, encryption only within the parent instance is not sufficient to fully satisfy the threat model of this study, which does not trust the host OS. At present, a practical and general solution has not been obtained for securely passing data to the Enclave without exposing any plaintext data or cryptographic keys to the parent instance in communication via the parent instance, and this remains future work.

Secret Key Management Inside the Enclave Inside the Enclave, there exists a secret key for signing audit logs and JWTs. However, in the prototype implementation of this study, a method is adopted in which the secret key is directly embedded into the EIF file. In this case, in an environment where system administrators can access the code inside the server or the Enclave image, it cannot be ruled out that the secret key may be leaked.

As a countermeasure to this issue, the use of AWS Key Management Service (KMS) is considered effective. KMS is a managed service for generating and managing cryptographic keys. Even key

creators or administrators cannot directly access the key material of the generated secret key. In addition, cryptographic operations such as encryption, decryption, and signing can be performed using the secret keys managed by KMS.

The official Nitro Enclave documentation indicates that functionality is provided to integrate Enclaves with KMS. By using this functionality, and setting the PCR values contained in the Remote Attestation Document as conditions for key usage, a configuration can be considered in which signing using the secret key is performed only when a legitimate Enclave environment is verified. This may enable signing while keeping the secret key hidden from outside the Enclave and from administrators. Since implementation and evaluation of this method have not been achieved in this study, it remains future work.

Necessity of a Session ID In the prototype implementation of this study, the state parameter in OIDC is repurposed as a session ID to identify the authentication process. This is due to an implementation constraint that session information is managed on the ID Verification Server on the parent instance without encryption.

However, state is originally a parameter to prevent CSRF attacks, and using it as an identifier for session management is not necessarily appropriate from a design perspective. Therefore, when assuming real operation, it is considered necessary to introduce a session ID independent of state and uniquely manage authentication processes.

On the other hand, adding a session ID as a new parameter to the OIDC protocol may affect compatibility with existing specifications and implementations. Therefore, rather than adding a new parameter, it is considered realistic to manage sessions by embedding a session ID within existing random values such as state, nonce, and code verifier. Since detailed design and evaluation of this method have not been achieved in this study, it remains future work.

7 Conclusion

In this study, we addressed the secure handling of authentication tokens and personal information aggregated on the the service provider's server-side environment in online identity verification using the My Number Card, and proposed and implemented an identity verification processing infrastructure using a Trusted Execution Environment (TEE).

In the proposed method, AWS Nitro Enclaves are used to isolate the verification processing of authentication tokens and user information from the host OS and operational administrators, and Remote Attestation enables verification of the legitimacy of the execution environment. Furthermore, by externally anchoring logs generated and signed inside the Enclave to WORM storage and a blockchain, we designed an architecture that achieves tamper detection and non-repudiation.

In the threat analysis, we organized assumed risks based on the STRIDE model. As a result, we showed that the TEE-centered architecture is effective against threats such as spoofing, tampering, repudiation, and information disclosure, even under an attack model that does not trust administrators or the host OS.

In the performance evaluation, we compared an Enclave-enabled configuration with an Enclave-disabled configuration. Although the average processing time increases due to the introduction of the Enclave, we confirmed that no delay on the order of seconds, which would affect the user authentication experience, occurs. From this result, it is suggested that the proposed method provides baseline performance even in practical operational environments that assume continuous identity verification processing.

From the above, we conclude that the identity verification processing infrastructure using TEE proposed in this study is an effective approach to achieving highly trustworthy identity verification without trusting administrators or the host OS, while considering the trade-off between performance and security.

As future work, detailed performance measurements related to Remote Attestation, and investigation of methods to guarantee end-to-end security from processing inside the Enclave to external communication, remain to be addressed.

References

[1] Amazon Web Services, Inc. AWS Nitro Enclaves. https://aws.amazon.com/jp/ec2/nitro/nitro-enclaves/. Amazon EC2 Documentation. Accessed: Aug. 7, 2025.

[2] Digital Agency, Japan. Digital Authentication App. https://services.digital.go.jp/auth-and-sign/. Accessed: Aug. 7, 2025.

[3] Digital Agency of Japan. Digital Authentication App Implementation Guideline. https://developers.digital.go.jp/documents/auth-and-sign/implement-guideline/. Accessed: Aug. 7, 2025.

[4] Intel Corporation. Intel Software Guard Extensions (SGX): Confidential Computing for Data-in-Use. https://www.intel.com/content/www/us/en/products/docs/accelerator-engines/software-guard-extensions.html. Accessed: Aug. 7, 2025.

[5] R. Itokawa, T. Yamaguchi, and E. Ito. A study of identity verification methods for digital student id. *IPSJ SIG Technical Report (IOT)*, 2024-IOT-66(1):1–6, July 2024.

[6] S. Iwano, H. Yanagi, and K. Sakurai. Keyspector: Secure execution of intrusion detection systems for iot devices using keystone. *IPSJ SIG Technical Report (OS)*, 2025-OS-167(3):1–8, May 2025.

[7] T. Kitayama, Y. Nishihira, A. Kanaoka, Y. Kakizaki, and T. Ohigashi. Development of an outsourcing password evaluation system using intel sgx. In *Proc. Int. Symp. Inf. Theory Appl. (ISITA)*, pages 407–412, Taipei, Taiwan, November 2024.

[8] C. Shepherd, R. N. Akram, and K. Markantonakis. Emlog: Tamper-resistant system logging for constrained devices with tees. *arXiv preprint arXiv:1712.03943*, December 2017.

[9] H. Yanagi, H. Kimura, T. Kondo, H. Watanabe, and T. Ohigashi. Implementation and evaluation of security modules for the edge computing platform based on modular architecture. *IPSJ SIG Technical Report (IOT)*, 2018-IOT-040:1–8, February 2018.

[10] Tomoki Yasui and Hidenobu Watanabe. Considering the use of tee for authentication and authorization of digital authentication app service apis. *IPSJ SIG Technical Report (IOT)*, 2025-IOT-68(36):1–8, February 2025.

[11] Tomoki Yasui and Hidenobu Watanabe. Proposal of an identity verification system using trusted execution environment in the japan's digital authentication app. In *Proceedings of the Thirteenth International Symposium on Computing and Networking Workshops (CANDARW 2025)*, pages 308–313, Yamagata, Japan, November 2025.