applied sciences

MDPI

Article

A Secure CDM-Based Data Analysis Platform (SCAP) in Multi-Centered Distributed Setting

Seungho Jeon, Chobyeol Shin, Eunnarae Ko and Jongsub Moon *

Division of Information Security, Graduate School of Information Security, Korea University, Seoul 02841, Korea; ohgnu90@korea.ac.kr (S.J.); lulustar@korea.ac.kr (C.S.); eun13@korea.ac.kr (E.K.)
* Correspondence: jsmoon@korea.ac.kr; Tel.: +82-02-3290-4750

Abstract: Hospitals have their own database structures and maintain their data in a closed manner. For this reason, it is difficult for researchers outside of institutions to access multi-center data. Therefore, if the data maintained by all hospitals follow a commonly shared format, researchers can analyze multi-center data using the same method. To safely analyze data using a common data model (CDM) in a distributed multi-center network environment, the objective of this study is to propose and implement the processes for distribution, executing the analysis codes, and returning the results. A secure CDM-based data analysis platform (SCAP) consists of a certificate authority (CA), authentication server (AS), code signer (CS), ticket-granting server (TGS), relaying server (RS), and service server (SS). The AS, CS, TGS, and RS form the central server group of the platform. An SS is stored on a hospital server as an agent for communication with the server group. We designed the functionalities and communication protocols among servers. To safely conduct the intended functions, the proposed protocol was implemented based on a cryptographic algorithm. An SCAP was developed as a web application running on this protocol. Users accessed the platform through a web-based interface.

Keywords: common data model; secure protocol; web application; distributed network

Citation: Jeon, S.; Shin, C.; Ko, E.; Moon, J. A Secure CDM-Based Data Analysis Platform (SCAP) in Multi-Centered Distributed Setting. *Appl. Sci.* **2021**, *11*, 9072. https://doi.org/10.3390/app11199072

Academic Editors: Toralf Kirsten and Oya Beyan

Received: 23 August 2021
Accepted: 26 September 2021
Published: 29 September 2021

1. Introduction

The importance of medical data has been emphasized over the past several decades, and numerous medical institutions around the world have built systems to accumulate medical records. Following this trend, electronic medical record (EMR) or electronic health record (EHR) systems have been actively adopted in local hospital networks [1,2]. With the recent development of big data and artificial intelligence technologies, researchers are attempting to use large numbers of medical data for secondary purposes such as the research and development of patient-centered services. However, it is unreasonable to derive statistically meaningful results using only the accumulated data in a single hospital. To overcome this problem, studies have been proposed to analyze data collected from several medical institutions, including multi-center or multi-source datasets [3,4].

However, despite the apparent advantages of multi-center datasets, the following practical problems occur: (1) Hospitals maintain medical data inside the local network boundary of the institution for management and security reasons. Because this setting allows resources to be distributed over the network, it is necessary to consolidate medical data [3,5,6]. (2) In general, hospitals have a database structure for storing medical data. Differences in data formats can be a stumbling block when merging and further analyzing medical data collected from institutions [7–9]. (3) Transmitting data from the hospital network without proper security can violate patient privacy [10]. (4) Even if medical data are transmitted through secure network channels among institutions, patients may not want personal information to be included in the transmission [11–14]. To effectively analyze data collected from various hospitals, these issues should be resolved.

In order to resolve these problems, there has been an increasing number of attempts to build a data-driven research infrastructure based on the Common Data Model (CDM). A Medical CDM is a standardized format of clinical or observational data that differs from hospital to hospital. Therefore, based on the same CDM format, investigators do not need to write multiple analysis procedures to analyze data from multiple hospitals [15]. Another advantage of CDM is that it makes it easier for the hospitals to share the same or similar infrastructure to store and analyze their data.

Currently, many consortiums in each country develop platforms to support multi-institutional medical research based on the above characteristics [3,16,17]. Although these platforms have improved the research environment and convenience, proper security should be considered, as they process medical data including patients' personal information. General security requirements for medical information system design and operation are stated in ISO/IEC 27799:2016 [18]: "information security policy", "organization of information security", "human resource security", "assert management", "access control", "cryptography", "physical and environmental security", "operational security", "communications security", "system acquisition", "development", and "maintenance", "supplier relationships", "information security incident management", "information security aspects of business continuity management", and "compliance". From a technical point of view, these requirements ensure that only authorized users perform authorized operations, and the integrity of the results is satisfied. The most important techniques to achieve this are user authentication [19,20] and digital signatures [21,22]. These techniques are used as part of many modern security protocols. For example, transport layer security (TLS) [10] supports authentication between parties communicating with each other, data encryption, and integrity guarantee for communication security. Although this protocol protects communication in a general way, it does not provide the security specific to the business logic of the medical system, such as a home health care service, between the user and the server. This problem is exacerbated in a multi-institutional medical data research environment where medical data is distributed on a network. Kerberos [23] provides authentication services in such an environment, but only one server at a time proves a user's identity and does not provide security for data sent and received after that. Therefore, a specialized security protocol is required to design a system to support multi-institutional medical data-based research.

In this paper, we propose a secure CDM-based data analysis platform (SCAP) for securely analyzing CDM data from multiple hospitals in a distributed network environment. SCAP distributes and executes software analysis codes (AC) and returns the analysis results to external investigators to analyze the CDM data of all hospitals connected to the platform. While providing an automated CDM data analysis service, SCAP provides end-to-end authentication to users accessing the platform and the integrity of CDM data analysis codes. The biggest advantage of this platform is that the medical data kept by each hospital is not shared beyond the hospital for analysis. This feature reduces the risk of the patients' personal information being leaked, and at the same time it enables us to obtain analysis results from multiple institutions. To safely manage this entire process, SCAP has the following features: (1) a secure multi-center user authentication, (2) integrity assurance for AC delivered over the network, (3) running AC and returning analysis results, and (4) providing a platform user authentication method without an account database for each hospital. To satisfy these features, the platform includes six subcomponents: a certificate authority (CA), an authentication server (AS), a code signer (CS), a ticket-granting server (TGS), a relaying server (RS), and a service server (SS). Among these components, the AS, CS, TGS, and RS form the central server group of the SCAP and provide an interface allowing users to deliver AC to hospitals registered on the platform. The CA issues certificates for public key cryptography for all participants of the platform. Finally, the SS is installed in the internal network of each hospital and analyzes the CDM data using the AC delivered from the central server group and returns the analysis results. The contributions of this paper are as follows:

- This paper proposes a process for safely analyzing medical CDM data in a multi-center distributed network environment.
- This paper describes in detail the functions of the components constituting the A secure CDM-based data analysis platform (SCAP) and the communication between components of the SCAP.

The rest of this paper is organized as follows. Section 2 presents the existing research on addressing the multi-institutional medical data and authentication methods. Section 3 describes the proposed data analysis platform. Section 4 details the implementations for the proposed platform. Lastly, Section 5 discusses the limitations and the conclusion.

2. Related Work

This section introduces the studies and popularly used user authentication methods to deal with security concerns that may arise from using multi-institutional medical data.

2.1. Approaches on Multi-Centered Medial Data

In this section, we introduce the existing privacy-preserving methods from different perspectives.

When combining data from different sources, patient privacy should be considered. Zhang et al. [3] proposed a solution for collecting data distributed across different departments and conducting data mining on an Internet of Health Things environment. The solution they proposed used locality-sensitive hashing (LSH) to consolidate data collected from multiple locations without concerning patient privacy. Ranbaduge et al. [5] and Vatsalan et al. [6] proposed privacy-preserving record linkage methods that connect multiple databases using a hash technique. In addition, Raisaro et al. [24] developed a platform that allows hundreds of clinical sites to share data and securely deliver the data number to external investigators. The platform uses homomorphic cryptography to provide end-to-end confidentiality and differential privacy for patient identification.

Most machine learning techniques that are actively employed in data analysis concentrate the data in a central storage to train a model. Due to the nature of the medical data containing sensitive information, exporting the data to the outside the institutional network is subject to numerous restrictions by policy or by law. Joint techniques for the learning of artificial intelligence models in a distributed environment without exposing data outside the institution have been actively studied [25–27]. Existing learning algorithms for artificial intelligence models collect the data necessary for learning in the central storage. By contrast, federated learning places the data in the locations where they are created and periodically aggregates only the learning (intermediate) states. Because the data are not exposed to the outside, federated learning is relatively free from privacy issues. Li et al. [28] suggested a method for learning a robust decision-making model in a distributed environment, although they did not explicitly mention any federated learning in their study.

Another obstacle is the data format. To extract the same information from data from multiple institutions, it is necessary to understand the database schema of all hospitals. Because most organizations treat database schema as confidential, not only is such an extraction extremely tedious, it is also difficult for external investigators to determine the data format of a particular hospital. To solve this problem, efforts have been made to unify the format of medical data in recent years. The common data model (CDM) is the most successful example of standardizing a medical data representation [7–9]. The data custodian of the hospital converts the original medical data of the institution into CDM data through an extract-transform-load (ETL) [11,12]. Most CDM specifications require anonymization to remove sensitive information contained in the original data during the ETL [13,14]. By unifying the format of such complex medical data, researchers and data analysts can explore and analyze data from multiple hospitals in the same way. In general, analysts use scripts or programs written in arbitrary programming languages. Although a CDM brings about many benefits to data analysis, many medical institutions

still maintain their data on the premises owing to legal issues. This operation policy allows medical data to be logically or physically distributed over the network. To improve the research environment, research platforms for multi-center medical data have been constructed [29,30].

2.2. Authentication Methods

Most modern applications implement procedures to verify the authenticity of users. In this section, we introduce communication schemes used for authentication.

The Json web token (JWT) securely transfers information between two parties in a lightweight and self-contained way using JSON objects [19]. The terminology 'self-contained' means that it has all the information needed for a claim in it. In other words, all the information required for authentication is included in the token. JWT consists of header, payload, and signature; each part is base64 encoded and separated by a dot. The header contains the algorithm for the signature generation. In the signature part, the digital signature for the payload, calculated by the algorithm specified in the header, is located. The payload holds the information to be contained in the token. This information is called claims and includes registered claims, public claims, and private claims as specified in the standard. JWT does not require a special storage because it contains all information necessary for user authentication in the token itself. However, it is difficult to set an appropriate token life cycle, and the more information the payload contains, the larger the token size.

Open authorization (OAuth) 2.0 is an open standard for delegating access of websites or applications to their information on other websites without the users providing passwords [20]. OAuth 2.0 deals with both authentication and authorization. The users (resource owner) are authenticated by the application (client) with their credentials. The client uses these credentials to request an access token from the authorization server. Authorization server verifies the received credentials and issues an access token. The client presents the access token and obtains the resources from the resource server. OAuth 2.0 is not only adopted as a popular authentication and authorization method in modern web services, it can also be used very conveniently by a user. In addition, services share resources with each other and can be easily extended with more functionalities. However, OAuth 2.0 is used only on HTTPS, and the access tokens should be managed securely. Since this scheme supports many authentication methods, developers need to correctly understand the specification, so that setting up the operating environment correctly.

Lastly, Kerberos provides a ticket-based centralized authentication service [23]. In an environment where services are distributed on a network, users typically should prove their identity for each service. The services should maintain a storage for the user's authentication information and so, it is vulnerable to security breaches. To resolve this inconvenience, Kerberos provides a simplified authentication mechanism using an authentication server (AS), a ticket-granting server (TGS), and a service server (V). The users prove their identity to the AS and receive a token. The users present the token and the ID of V to access to TGS, and receive a ticket. This ticket is presented when the users access V and is used to authenticate the users by TGS. If the users access another V, without repeating authentication from the beginning, the users obtain a new ticket by presenting the token of the AS and the ID of the new V. Owing to this simplified authentication process, Kerberos is often employed to implement single sign-on (SSO) functionality. However, Kerberos allows access to only one service at a time, although outside the scope of the protocol design.

3. Methods

In this chapter, we describe the operating environment of the SCAP, the roles of the subcomponents constructing the platform in detail. Inspired by Kerberos' convenient authentication protocol, we designed secure communication between each component of SCAP.

3.1. Operating Environment

Figure 1 shows the operating environment of the SCAP. This platform is built on a distributed network where hospitals store CDM data on their server, and these servers are connected to the network. In this figure, the central server group consists of the AS, CS, TGS, and RS. The hospital network consists of an SS and a database of CDM data. The CA is placed on the Internet and provides trust for all SCAP components with public key certificates. The SCAP uses the observational medical outcomes partnership CDM (OMOP-CDM) [7] as a medical CDM. OMOP-CDM is a database scheme defined by observational health data sciences and informatics (OHDSI). In general, researchers using OMOP-CDM analyze data using R language and software provided by OHDSI [31,32]. All hospital networks are connected to a server group through the Internet.

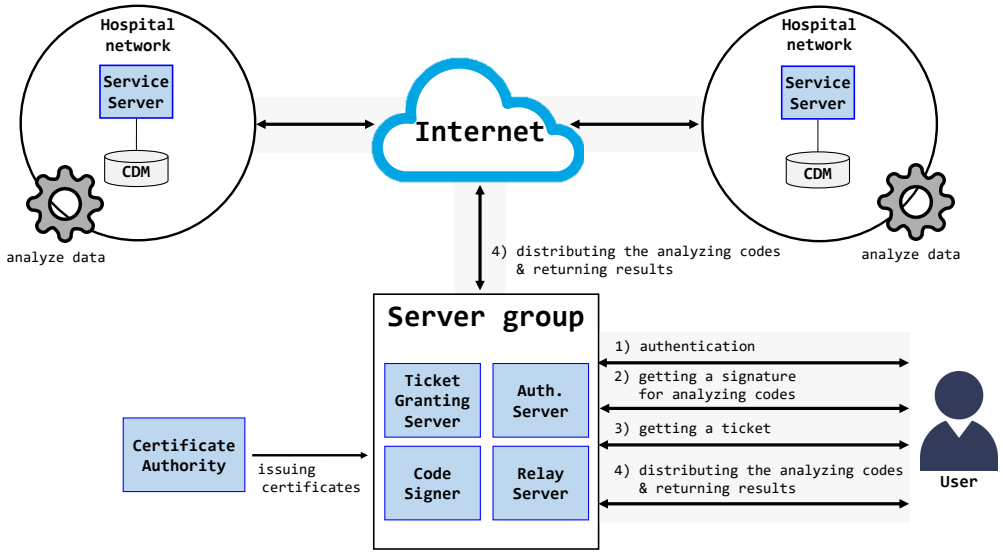

Figure 1. Operating environment of SCAP.

Each component of the SCAP and the process of analyzing the medical CDM data are as follows: (1) The users prove their identity to the platform. The AS is responsible for user authentication and issues a token to authenticated users. (2) The users obtain a digital signature for the analysis codes written to analyze medical CDM data held by hospitals through the CS. (3) The users receive a ticket from the TGS proving their identity for the SS of each hospital. (4) The users distribute the analysis codes to all hospitals connected to the platform through the RS and receive the analysis results. As a preliminary task for the safe operation these processes, all platform servers and users create their private/public keys and receive public key certificates from the CA. This certificate conforms to the X.509 version 3 [33] standard, which is currently the most widely applied. Platform participants can obtain certificates, such as generic electronic financial transactions.

In this section, we describe the network protocol and information transmitted between the components of the SCAP. In the results section, we present the detailed implementation of each component along with the algorithms used.

3.2. Communication between User and AS

Before the communication between a user and the AS, it is assumed that the user has been appropriately registered in the AS. This assumption implies that the user is also enrolled in the SCAP. Figure 2 shows the process by which the AS authenticates the user of the SCAP platform. A user who wants to use the platform sends a message MC with the user's identifier ID_C, such as an `ID/password` or biometric, and a signature sig_{M_C} of M_C to request proof of identity to the AS (flow 1 in Figure 2). The signature is transmitted along with the message to prevent tampering by attackers, whereas the message is transmitted

over the network to verify the sender. Because all components of the SCAP always transmit a message–signature pair, we omit the description of signatures for convenience in the remainder of this paper. If the requestor is a legitimate user, the AS returns a message M_{AS} including a cryptographically generated token, token, and its signature $sig_{M_{AS}}$ to the user (flow 2 in Figure 2). This token is generated by encrypting the user's identity and expiration time using a symmetric key cryptographic algorithm, such as DES [34] or AES [35]. Therefore, only participants who know the key used to create the token can verify the validity of the token. The AS does not share the key used to generate the token with anyone to prevent a third party. In addition, for users not registered on the platform or illegal requests, the AS delivers an authentication failure response to the user. The user cannot use any SCAP service.

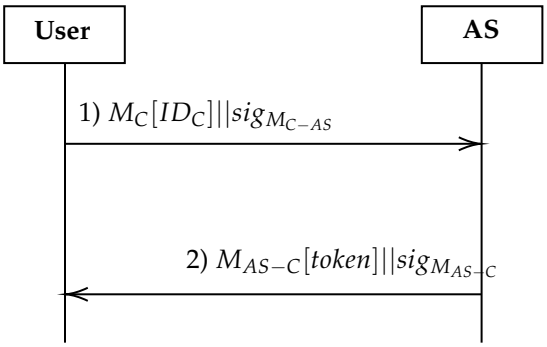

Figure 2. Sequence diagram for communication between the user and AS.

3.3. Communication between User and CS

It is assumed that the user accessing the CS has passed the authentication of the AS. Figure 3 shows how a user receives a digital signature for the AC to analyze the CDM data. Before accessing the CS, the user creates an AC to analyze the CDM data of hospitals through the SCAP. An AC can be written in a variety of ways. Because the SCAP is implemented based on the OMOP-CDM, users can use software [15,36] provided by OHDSI. The AC is generally written in R language and contains all information for analyzing the CDM data, such as cohort definitions and SQL queries. Because the AC is used to analyze medical data, it should not be tampered with until it is delivered to each hospital over the network. A digital signature is used to guarantee data integrity and non-repudiation during transmission. To satisfy this requirement, the user requests the CS to create a digital signature for the AC. To this end, the user delivers a message M_{C-CS} and its signature $sig_{M_{C-CS}}$ (flow 1 in Figure 3). The message M_{C-CS} includes the AC and the token issued from the AS in the previous section. The CS creates a signature sig^{AC} for the AC using its private key and wraps it in message M_{CS-C}. The CS then delivers M_{CS-C} and its signature $sig_{M_{CS-C}}$ in response to the user (flow 2 in Figure 3). Because this signature was created with the private key of the CS, no one except the CS can create the same signature. However, anyone can verify the digital signature to obtain the public key of the CS. If the user modifies a part of the AC or writes a completely new AC, the user requests a new signature to use the CS.

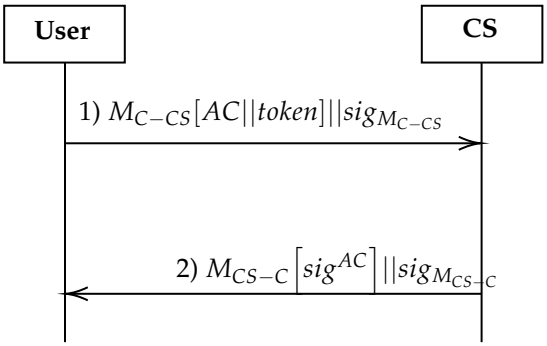

Figure 3. Sequence diagram for the communication between the user and CS.

3.4. Communication between User and TGS

Users who have passed authentication with the AS and obtained a digital signature for the AC can use the TGS to obtain a ticket to access the SSs connected to the SCAP. Figure 4 shows the process of issuing a ticket for a user to access the CDM data of all hospitals in the SCAP through the TGS. To request the issuance of a ticket to the TGS to access the SS of the hospital connected to the SCAP, a message M_{C-TGS}, which includes token and sig^{AC}, along with its signature $sig_{M_{C-TGS}}$, is transmitted (flow 1 in Figure 4). It does not matter whether M_{C-TGS} contains the AC, as long as it is proven that the generator of sig^{AC} is the CS. The TGS delivers a message M_{TGS-C}, including a cryptographically generated ticket and its signature $sig_{M_{TGS-C}}$ to the platform users (flow 2 in Figure 4). Similar to the authentication token of the AS, this ticket is generated using a symmetric key cryptographic algorithm with information such as the user's ID and expiration time. However, unlike the AS, the key used to create the ticket is encrypted with the public key of each SS receiving the ticket and then delivered to the user along with the ticket. Therefore, the user acquires the same number of tickets as the number of SSs connected to the platform. The user cannot decrypt the verification key delivered to the ticket. That is, except for the SS, a third party who has obtained a ticket, including a user, cannot create or decrypt a ticket. The TGS is an essential component in the SCAP operating on a distributed network. Without the help of the TGS, if users want to analyze CDM data from multiple medical institutions, they should prove their identity to the administrator of each institution. Even in such an environment, each institution should operate a database that stores the identities of all users who want to use the data. The TGS creates a ticket that allows any medical institution to authenticate the user's identity, in an attempt to analyze the data and alleviate these obstacles. By verifying this ticket, the SS of each hospital can authenticate the user without a user account database.

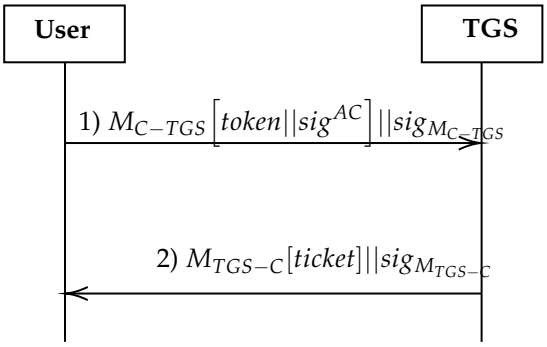

Figure 4. Sequence diagram for communication between the user and TGS.

3.5. Communication for Distributing AC

We implemented three modes for distributing the AC and analyzing the CDM data of SCAP-connected hospitals: full-automation mode, intervention mode, and hybrid mode. In this section, we describe these three modes.

Figure 5 shows the communication process between the user, RS, and SS of hospitals in full-automation mode. In this mode, as the name suggests, the entire process, from the distribution of the AC to the returning CDM data analysis results, is conducted automatically. It is assumed that the user has a token, the AC, sig^{AC}, and a ticket in advance. First, the user transmits the message M_{C-RS} with the above four pieces of information and the signature $sig_{M_{C-RS}}$ to the RS of the central server group (flow 1 in Figure 5). The RS verifies the token in M_{C-RS} to determine whether the AS has authenticated the user. If the user is verified, the RS repackages the remaining information except for the token in M_{RS-SS} and sends this message with its signature to the SS (SS_1–SS_n) (flow 2 in Figure 5). The SS checks the authenticity of the user and the AC using the ticket and sig^{AC}, respectively. If the verification of both the user and AC is successful, the SS runs the AC and analyzes the CDM data of its hospital (flow 3 in Figure 5). The analysis result (AR) is then returned to the RS (flow 4 in Figure 5). Finally, the user downloads the ARs from the hospitals (flow 5 in Figure 5). In full-automation mode, users can analyze the hospital data without any intervention.

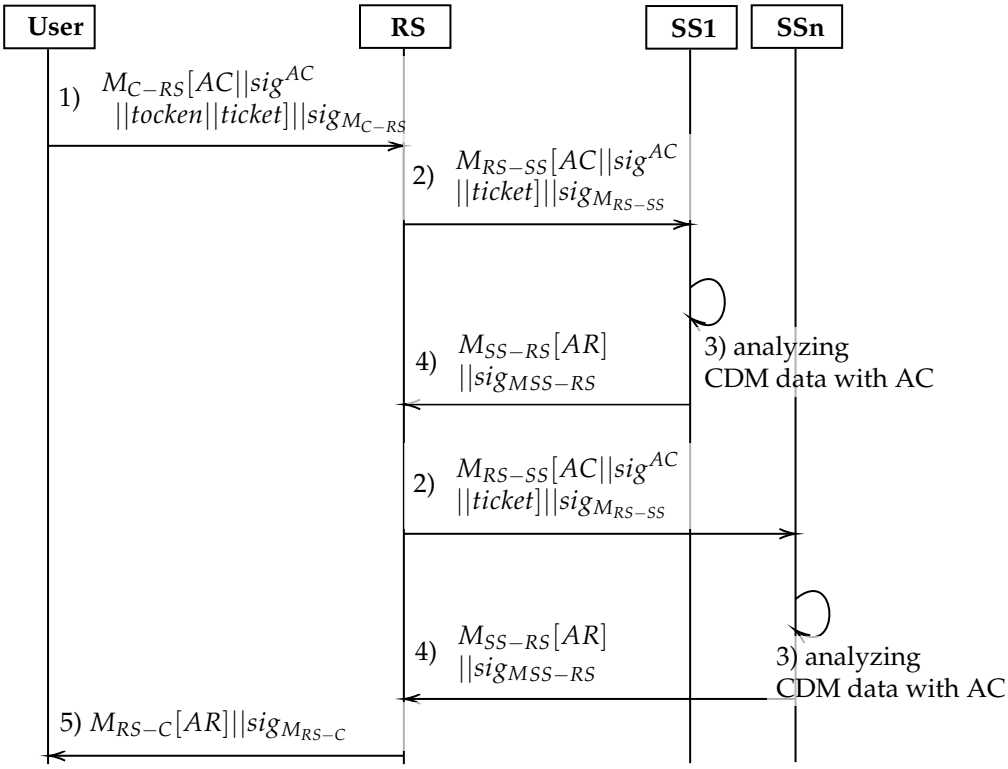

Figure 5. Sequence diagram for distributing AC in full-automation mode.

However, in full-automation mode, the hospital (or data manager) cannot control the process by analyzing the data of the institution and exporting the results. Institutions may simply want to deny the user access to the data. Allowing data to be analyzed without appropriate controls can occasionally violate the security policy of an institution. To satisfy these requirements, the intervention mode requests the hospital custodian to execute the AC and return the AR. First, as in full-automation mode, the user transmits a token, the AC, sig^{AC}, and a ticket to the RS (flow 1 in Figure 6). RS verifies the user's identity by validating the token. The RS distributes the AC, a token, and a ticket from the user to the SSs of the hospitals (flow 2 in Figure 6). The SS verifies the received ticket and digital signature to verify the user and integrity of the AC. The SS does not immediately execute the AC, but waits for approval from the data custodian of the hospital. The custodian confirms the AC delivered to the hospital and approves the execution (flow 3 in Figure 6). Upon approval, the SS runs the AC to analyze the data of the hospital CDM (flow 4 in Figure 6). If the custodian suspends or rejects the AC execution, the AC is not executed. Likewise,

the SS does not immediately return the AR for the CDM data to the user, but rather waits for the export approval of the custodian. If the custodian approves its export (flow 5 in Figure 6), the user can obtain the AR (flow 6 in Figure 6). By contrast, if the custodian suspends or rejects its export, the user cannot access the AR even after the data analysis has been completed.

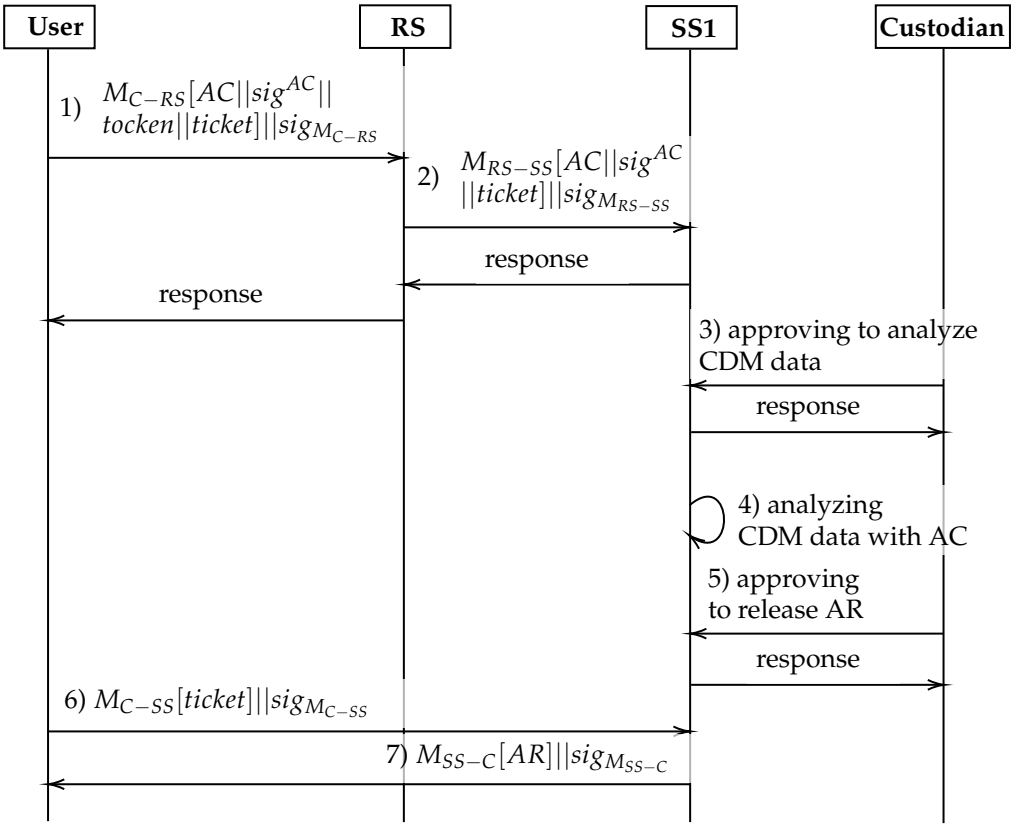

Figure 6. Sequence diagram for distributing AC in intervention mode.

Intervention mode provides hospital control over the data analysis. Hospitals can configure secure policies compared with full-automation mode. However, if there are many requests for analysis in intervention mode, some custodians might be burdened if they are responsible for all data analyses. To overcome these shortcomings, we implemented a hybrid mode, which is a combination of full automation and intervention modes. Hospitals registered on the platform each have a whitelist of users who can analyze the data of the institution without restriction. If the user is registered on a whitelist, the SS applies both full-automation and intervention modes to an unregistered user. Using hybrid mode, the hospital can flexibly and efficiently control the data analysis of the user.

4. Results

We developed the SCAP, a platform for analyzing CDM data in a distributed network environment based on the communication protocol described in the previous section. This section describes the implementation details of the SCAP. The source codes of the SCAP are available from the URLs listed in Table 1. Each of the subcomponents of the SCAP is implemented as a web application. Interfaces for communicating between subcomponents are implemented as representational state transfer (REST) APIs. By virtue of the REST API, each server can provide functionalities, including not just the SCAP, but also any applications that require a similar functionality.

Table 1. Repositories for SCAP.

Component	URL (Accessed date: 29 September 2021)
CA	https://github.com/KUSYS-LAB/Certificate-Authority-for-SCAP
RS	https://github.com/KUSYS-LAB/Web-Interface-for-SCAP
AS	https://github.com/KUSYS-LAB/Authentication-Server-for-SCAP
CS	https://github.com/KUSYS-LAB/Code-Signer-for-SCAP-master
TGS	https://github.com/KUSYS-LAB/Ticket-Granting-Server-for-SCAP
SS	https://github.com/KUSYS-LAB/Service-Server-for-SCAP

4.1. Implementation of AS

Algorithms 1 and 2 show the algorithms for issuing authentication tokens for users of the SCAP from Figure 2 (Algorithm 1) and the AS (Algorithm 2). Rather than operating individually, these algorithms fulfill their respective roles through synchronization.

Algorithm 1: The authentication algorithm on the user side

```
1  Procedure authenticate(ID_C)
2      M_{C-AS} ← make a message with ID_C
3      sig_{M_{C-AS}} ← generate a signature for M_{C-AS}
4      M_{AS-C}, sig_{M_{AS-C}} ← request to AS for authentication with M_{C-AS}||sig_{M_{C-AS}}
5
6      sig_check ← verify sig_{M_{AS-C}}
7      if sig_check failed then
8          return ERROR
9      end
10
11     token ← get a token from M_{AS-C}
12     return token
13 end
```

Algorithm 2: The authentication algorithm on the AS side

```
1  Procedure handle_authentication(M_{C-AS}, sig_{M_{C-AS}})
2      sig_check ← verify sig_{M_{C-AS}}
3      if sig_check failed then
4          return ERROR
5      end
6
7      ID_C ← get an identifier from M_{C-AS}
8      id_check ← verify an identifier
9      if id_check failed then
10         return ERROR
11     end
12
13     token ← generate token
14     M_{AS-C} ← make a message with token
15     sig_{M_{AS-C}} ← generate a signature for M_{AS-C}
16     return M_{AS-C}||sig_{M_{AS-C}}
17 end
```

A user invokes an `authenticate` procedure with identity information ID_C. This procedure creates an M_{C-AS} wrapping ID_C and its signature $sig_{M_{C-AS}}$, and requests user authentication to AS (line 4 in Algorithm 1 for flow 1 in Figure 2). When the AS receives an authentication request from the user, it calls the `handle_authentication` procedure using M_{C-AS} and $sig_{M_{C-AS}}$. This procedure verifies $sig_{M_{C-AS}}$ to determine the integrity of

M_{C-AS} and the sender (from lines 1–5 in Algorithm 2). If $sig_{M_{C-AS}}$ fails, the AS returns an appropriate ERROR to the user. Otherwise, AS uses ID_C to check whether the sender is a registered user on the SCAP platform (from lines 7–11 in Algorithm 2). If the user is not registered, the AS returns an ERROR. When a full verification is successfully completed, AS generates a token based on cryptography and returns M_{AS-C} and its signature $sig_{M_{AS-C}}$ to the user (line 16 in Algorithm 2 for flow 2 in Figure 2). When the AS returns a token or ERROR, the execution of the authenticate procedure resumes from line 4. The user verifies $sig_{M_{AS-C}}$ to verify the integrity and source of M_{AS-C} (from lines 6–9 in Algorithm 1). If M_{AS-C} is not created by the correct AS or is corrupted during network communication, the authenticate procedure returns an ERROR. Otherwise, it obtains a token from M_{AS-C} and returns it (line 12 in Algorithm 1). The user stores the acquired tokens safely during their local storage.

4.2. Implementation for CS

Algorithms 3 and 4 show algorithms for generating a digital signature for the AC written by a user between the user (Algorithm 3) and CS (Algorithm 4), as shown in Figure 3. To analyze the CDM data intended by the user, the AC should not be tampered with by attackers while being transmitted to the SS. In addition, the author of the AC should be proven in an end-to-end manner.

The user calls the `request_to_sign_AC` procedure with the AC and the token issued by the AS. This procedure requires the CS to issue a digital signature to guarantee the integrity of the AC. The procedure sends M_{C-CS} with its signature $sig_{M_{C-CS}}$ (line 4 in Algorithm 3 for flow 1 in Figure 3). The message M_{C-CS} contains both the AC and a token. Upon receiving the request, the CS invokes the `handle_signing_AC_request` procedure. This procedure uses $sig_{M_{C-CS}}$ to verify the integrity of the sender of M_{C-CS} (from lines 2 to 5 in Algorithm 4). The CS extracts the token from M_{C-CS} and checks the identity of the sender (from lines 7–11 in Algorithm 4). If any of these checks fail, the procedure returns an ERROR. Otherwise, the CS acquires AC from M_{C-CS} and generates a digital signature sig^{AC} (lines 13 and 14 in Algorithm 4). A hash algorithm, such as a secure hash algorithm (SHA) and a public key encryption algorithm (RSA [37] or ECDSA [38]) may be used to generate a digital signature. This procedure returns M_{CS-C}, which wraps sig^{AC} and its signature $sig_{M_{CS-C}}$, to the user (from lines 16 to 18 in Algorithm 4 for flow 2 in Figure 3). When a response is received from the CS, the `request_to_sign_AC` procedure is resumed from line 4. This procedure validates $sig_{M_{CS-C}}$ to verify the message and sender (lines 6 to 9 in Algorithm 3). If the integrity stands, the procedure obtains and returns sig^{AC} from M_{CS-C} (line 12 in Algorithm 3). The user safely maintains the acquired sig^{AC}.

Algorithm 3: The code signing algorithm on the user side

```
1  Procedure request_to_sign_AC(AC, token)
2      M_{C-CS} ← make a message with AC, token
3      sig_{M_{C-CS}} ← generate a signature for M_{C-CS}
4      M_{CS-C}, sig_{M_{CS-C}} ← request to CS for signing AC with M_{C-CS}||sig_{M_{C-CS}}
5
6      sig_check ← verify sig_{M_{CS-C}}
7      if sig_check failed then
8          return ERROR
9      end
10
11     sig^{AC} ← get a signature for AC from M_{CS-C}
12     return sig^{AC}
13 end
```

Algorithm 4: The code signing algorithm on the CS side

```
1  Procedure handle_signing_AC_request(M_{C-CS}, sig_{M_{C-CS}})
2      sig_check ← verify sig_{M_{C-CS}}
3      if sig_check failed then
4          return ERROR
5      end
6
7      token ← get a token from M_{C-CS}
8      token_check ← verify token
9      if token_check failed then
10         return ERROR
11     end
12
13     AC ← get analyzing codes from M_{C-CS}
14     sig^{AC} ← generate a signature for AC
15
16     M_{CS-C} ← make a message with sig^{AC}
17     sig_{M_{CS-C}} ← generate a signature for M_{CS-C}
18     return M_{CS-C}||sig_{M_{CS-C}}
19 end
```

4.3. Implementation for TGS

Algorithms 5 and 6 show the algorithms used to issue a ticket for the user to access the SS between the user (Algorithm 5) and the TGS (Algorithm 6). TGS issues tickets that allow users to authenticate multiple SSs.

The user calls the `request_to_issue_ticket` procedure with the token and sig^{AC} to receive a ticket from the TGS. The procedure parameters are included in M_{C-TGS} and sent to the TGS with the signature $sig_{M_{C-TGS}}$ (line 4 in Algorithm 5 for flow 1 in Figure 4). The TGS calls the `handle_issuing_ticket_request` procedure. As presented in the previous algorithms, this procedure verifies the message, identity, and sig^{AC} passed by the user and returns an `ERROR` if it fails (from lines 2 to 17 in Algorithm 6). If all of the information is valid, this procedure generates a ticket cryptographically. The TGS creates a message M_{TGS-C} with the ticket and returns the message to the user along with $sig_{M_{TGS-C}}$ (from lines 19 to 22 in Algorithm 6 for flow 2 in Figure 4). The `request_to_issue_ticket` procedure verifies $sig_{M_{TGS-C}}$, and if the message is not tampered with, the procedure on the user side extracts a ticket from the message and returns it. The user safely stores the acquired TGS ticket, along with the token and sig^{AC}.

4.4. Implementation of RS and SS

Algorithms 7–9 show algorithms for analyzing the CDM data among users (Algorithm 7), RS (Algorithm 8), and SS (Algorithm 9), as shown in Figure 5. In this section, only algorithms for the SS set in full-automation mode are introduced. However, there was no significant difference in the algorithm in the intervention or hybrid modes.

The user invokes the `request_to_analyze_CDM_data` procedure with a token, AC, sig^{AC}, and ticket. This procedure transmits M_{C-RS}, which includes the above parameters and signature $sig_{M_{C-RS}}$, to RS (line 4 in Algorithm 7 for flow 2 in Figure 5). RS calls the `handle_distributing_AC_request` procedure. This procedure validates the tokens in $sig_{M_{C-RS}}$ and M_{C-RS} and returns an `ERROR` if the verification fails (from lines 1 to 11 in Algorithm 8). If there are no errors, this procedure forwards M_{RS-SS}, which includes AC, sig^{AC}, and the ticket, and signature $sig_{M_{RS-SS}}$, to all SSs registered in the SCAP (line 19 in Algorithm 8 for flow 2 in Figure 5). SS calls the `handle_analyzing_CDM_data_request`. This procedure validates $sig_{M_{RS-SS}}$, ticket, and sig^{AC} before executing the AC and returns an `ERROR` if verification fails (from lines 2 to 17 in Algorithm 9). The SS does not immediately

execute the AC, even if all verifications are successful. In general, AC execution is a time-consuming task; therefore, instead of synchronizing all algorithms in Algorithms 7–9, we append the AC to the thread pool. If AC is successfully included in the thread pool, this procedure returns a `SUCCESS` message, and if it fails for any reason, it returns an `ERROR` immediately (from lines 19 to 24 in Algorithm 9). If the thread pool is full owing to many data analysis requests from other users, a scheduler suspends the execution AC until resources become available. Thus, the computational resources of the SS can be efficiently operated. The return of the `handle_analyzing_CDM_data_request` procedure is delivered to the user through the RS without hesitation.

Figure 7 shows the structure of the SS used to implement the algorithm of Algorithm 9. Unlike other components of SCAP, the SS provides various functions for analyzing medical data. (1) The SS places the AC into the queue. In general, the execution of the AC requires considerable computational resources; it is therefore impossible to execute all analysis requests from users in parallel. Thus, the SS limits the ACs that can be run at a time using the management pool for the ACs. (2) The SS can run the AC to analyze the CDM data of the institution. If all previous verifications are successful, the SS executes the AC using the R engine. Alternatively, if the AC is written in a programming language other than R, the appropriate execution engine is called. When the data analysis is complete, the SS stores the analysis results, AR, in a compressed file type, such as a zip file. (3) The SS may notify the user of the CDM data analysis status. The data analysis process is time consuming (typically several hours). In addition, there may be differences in the execution times of the AC for each institution. Because the user cannot wait for the data analysis of all hospitals to be completed, the user should periodically check the analysis status of each hospital. When the user requests the analysis status, the SS responds with a "Waiting for execution", "Running", or "Analysis complete" message. If the SS is in full-automation mode, the AC is immediately executed without going through the execution standby state. (4) The SS may notify the user of the AR export status. As in the AC execution, the SS receives instructions from the custodian regarding whether to provide the results of the CDM data analysis to the user. When the user requests the SS where AR replies are possible, the SS responds with either "available for download" or "waiting for approval". Users can only obtain the AR if they are "available for download". As with function 3, when the SS is in full-automation mode, the "waiting for export" state is not used.

Algorithm 5: The ticket issuing algorithm on the user side

1 **Procedure** `request_to_issue_ticket(`*token*, sig^{AC}`)`
2 $M_{C-TGS} \leftarrow$ make a message with `token` and sig^{AC}
3 $sig_{M_{C-TGS}} \leftarrow$ generate a signature for M_{C-TGS}
4 $M_{TGS-C}, sig_{M_{TGS-C}} \leftarrow$ request to TGS for issuing `ticket` with $M_{C-TGS}||sig_{M_{C-TGS}}$
5
6 `sig_check`$\leftarrow$ verify $sig_{M_{TGS-C}}$
7 **if** `sig_check` *failed* **then**
8 **return** `ERROR`
9 **end**
10
11 `ticket` $\leftarrow$ get a ticket from M_{TGS-C}
12 **return** `ticket`
13 **end**

Algorithm 6: The ticket issuing algorithm on the TGS side

Procedure `handle_issuing_ticket_request(`M_{C-TGS}, $sig_{M_{C-TGS}}$`)`
  `sig_check` ← verify $sig_{M_{C-TGS}}$
  if `sig_check` *failed* **then**
    return `ERROR`
  end

  `token` ← get a token from M_{C-TGS}
  `token_check` ← verify token
  if `token_check` *failed* **then**
    return `ERROR`
  end

  sig^{AC} ← get a signature from M_{C-TGS}
  `sig_AC_check` ← verify the signature for AC
  if `sig_AC_check` *failed* **then**
    return `ERROR`
  end

  `ticket` ← generate a ticket
  M_{TGS-C} ← make a message with `ticket`
  $sig_{M_{TGS-C}}$ ← generate a signature for M_{TGS-C}
  return $M_{TGS-C}||sig_{M_{TGS-C}}$
end

Algorithm 7: The AC distributing algorithm on the user side

Procedure `request_to_analyze_CDM_data(token,` AC, sig^{AC}, `ticket)`
  M_{C-RS} ← make a message with `token`, AC, sig^{AC}, and `ticket`
  $sig_{M_{C-RS}}$ ← generate a signature for M_{C-RS}
  `response` ← request to RS for distributing AC to analyze the CDM data with $M_{C-RS}||sig_{M_{C-RS}}$

  if *request success* **then**
    `// user can download AR after CDM data analysis is complete`
    return `SUCCESS`
  else
    return `ERROR`
  end
end

Algorithm 8: The AC distributing algorithm on the RS side

Procedure `handle_distributing_AC_request`(M_{C-RS}, $sig_{M_{C-RS}}$)
  `sig_check` ← verify $sig_{M_{C-RS}}$
  if `sig_check` *failed* **then**
    return `ERROR`
  end

  `token` ← get a token from M_{C-RS}
  `token_check` ← verify token
  if `token_check` *failed* **then**
    return `ERROR`
  end

  AC ← get the analyzing codes from M_{C-RS}
  sig^{AC} ← get the signature for AC from M_{C-RS}
  `ticket` ← get the ticket from M_{C-RS}

  M_{RS-SS} ← make a message with AC, sig^{AC}, and `ticket`
  $sig_{M_{RS-SS}}$ ← generate a signature for M_{RS-SS}
  `result` ← distribute the analyzing codes with $M_{RS-SS}||sig_{M_{RS-SS}}$
  if *distribution success* **then**
    return `SUCCESS`
  else
    return `ERROR`
  end
end

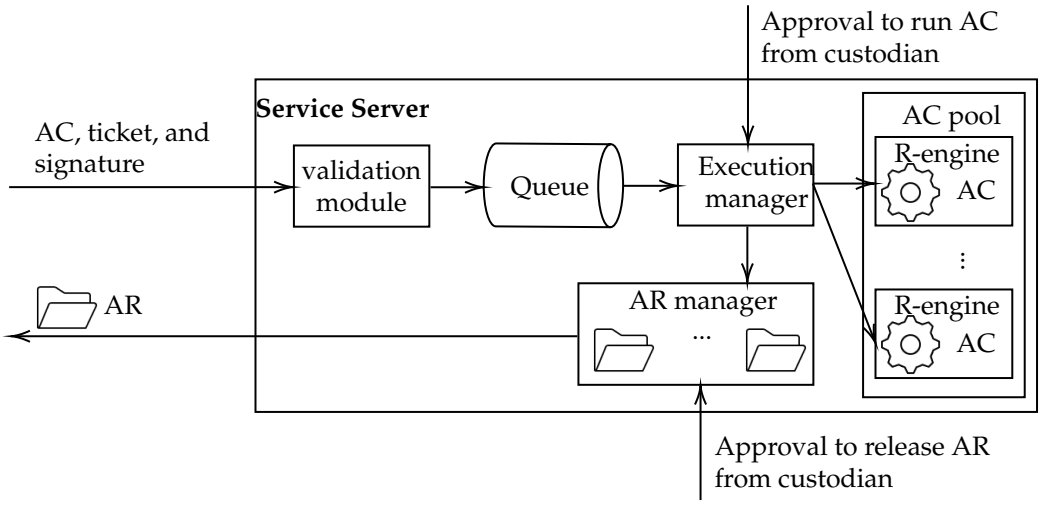

Figure 7. Architecture of SS.

Algorithm 9: The AC distributing algorithm on the SS side

```
Procedure handle_analyzing_CDM_data_request(M_RS−SS, sig_M_RS−SS)
    sig_check ← verify sig_M_RS−SS
    if sig_check failed then
        return ERROR
    end

    ticket ← get the ticket from M_RS−SS
    ticket_check ← verify ticket
    if ticket_check failed then
        return ERROR
    end

    sig^AC ← get the signature for AC
    sig_AC_check ← verify the signature for AC
    if sig_AC_check failed then
        return ERROR
    end

    result ← append AC to the thread pool
    if result success then
        // AC will be executed to analyze the CDM data by the thread
           scheduler and AR will be saved.
        return SUCCESS
    else
        return ERROR
    end
end
```

5. Discussion

5.1. Limitations

Although the SCAP has basic functions for analyzing CDM data in a distributed network environment, it has the following limitations. First, at this point, the SCAP does not have an access control mechanism for users. If each hospital operates the SS in intervention or hybrid mode, access to the data of the institution can be manually or automatically restricted to users. However, this is undesirable as an access control, and the user's access service should be controlled at the central server group level. Fortunately, the SCAP already uses the AS to authenticate users, and thus it is relatively easy to embed access control into the platform. Second, the SCAP allows CDM data from each hospital to be analyzed individually. However, because medical data show different statistical characteristics according to the geographical location, it is insufficient to analyze hospital data individually through big data analysis. Additional techniques, such as federated learning, should be considered to solve this problem.

5.2. Conclusions

This paper proposed the SCAP, a platform that facilitates an analysis in a distributed environment in which CDM data are stored in the internal network of a hospital. The SCAP consists of the CA, AS, CS, TGS, RS, and SS for safely protecting the data analysis process. We designed and implemented a communication protocol among the components. In addition, some limitations of the platform were discussed. At this point, the SCAP is at the prototype level. In the future, we plan to expand this platform with continuous research and development.

Author Contributions: Conceptualization, S.J.; methodology, S.J.; formal analysis, S.J.; writing—original draft, S.J.; visualization, S.J.; software, C.S. and E.K.; investigation, C.S. and E.K.; validation, J.M.; writing—review and editing, J.M.; supervision, J.M.; project administration, J.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by a grant from the Korea Health Technology R&D Project through the Korea Health Industry Development Institute (KHIDI) and funded by the Ministry of Health & Welfare, Republic of Korea (Grant Number: HI19C0791).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

EMR	electronic medical record
EHR	electronic health record
LSH	locality-sensitive hashing
CDM	common data model
ETL	extract-transform-load
AC	analysis codes
WI	web interface
CA	certificate authority
AS	authentication server
CS	code signer
TGS	ticket-granting server
RS	relaying server
SS	service server
OMOP-CDM	observational medical outcomes partnership CDM
OHDSI	observational health data sciences and informatics
AR	analysis result
REST	representational state transfer
PKI	public-key infrastructure
SHA	secure hash algorithm
JWT	Jwon web token
OAuth 2.0	Open authorization 2.0

References

1. Liu, J.; Li, X.; Ye, L.; Zhang, H.; Du, X.; Guizani, M. BPDS: A Blockchain Based Privacy-Preserving Data Sharing for Electronic Medical Records. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emerites, 9–13 December 2018; doi:10.1109/GLOCOM.2018.8647713. [CrossRef]
2. Kruse, C.S.; Smith, B.; Vanderlinden, H.; Nealand, A. Security Techniques for the Electronic Health Records. *J. Med. Syst.* **2017**, *41*, 1–9. [CrossRef] [PubMed]
3. Zhang, Q.; Lian, B.; Cao, P.; Sang, Y.; Huang, W.; Qi, L. Multi-source medical data integration and mining for healthcare services. *IEEE Access* **2020**, *8*, 165010–165017. [CrossRef]
4. Deng, Y.; Li, Y.; Shen, Y.; Du, N.; Fan, W.; Yang, M.; Lei, K. MedTruth: A semi-supervised approach to discovering knowledge condition information from multi-source medical data. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; doi:10.1145/3357384.3357934. [CrossRef]
5. Ranbaduge, T.; Vatsalan, D.; Christen, P.; Verykios, V. Hashing-based distributed multi-party blocking for privacy-preserving record linkage. In *Lecture Notes in Computer Science, Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Auckland, New Zealand, 19–22 April 2016*; Springer: Cham, Switzerland, 2016; Volume 9652._33. [CrossRef]
6. Vatsalan, D.; Christen, P. Scalable privacy-preserving record linkage for multiple databases. In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, Shanghai, China, 3–7 November 2014; doi:10.1145/2661829.2661875. [CrossRef]
7. OHDSI. OMOP Common Data Model. Available online: https://www.ohdsi.org/data-standardization/the-common-data-model/ (accessed on 29 September 2021).

8. Sentinel Initiative. Sentinel Common Data Model. Available online: https://www.sentinelinitiative.org/methods-data-tools/sentinel-common-data-model (accessed on 29 September 2021).
9. PCORI. PCORnet. Available online: https://pcornet.org/data/ (accessed on 29 September 2021).
10. Dierks, T.; Rescorla, E. *RFC 5246: The Transport Layer Security (TLS) Protocol—Version 1.2*; International Engineering Task Force (IETF): Wilmington, DE, USA, 2008.
11. Vassiliadis, P. A survey of extract-transform-load technology. *Int. J. Data Warehous. Min.* **2009**, *5*, 1–27. [CrossRef]
12. Denney, M.J.; Long, D.M.; Armistead, M.G.; Anderson, J.L.; Conway, B.N. Validating the extract, transform, load process used to populate a large clinical research database. *Int. J. Med. Inform.* **2016**, *94*, 271–274. [CrossRef] [PubMed]
13. Lee, H.; Kim, S.; Kim, J.W.; Chung, Y.D. Utility-preserving anonymization for health data publishing. *BMC Med. Inform. Decis. Mak.* **2017**, *17*, 1–12. [CrossRef] [PubMed]
14. Nayahi, J.J.V.; Kavitha, V. Privacy and utility preserving data clustering for data anonymization and distribution on Hadoop. *Futur. Gener. Comput. Syst.* **2017**, *74*, 393–408. [CrossRef]
15. OHDSI. ATLAS-A Unified Interface for the OHDSI Tools. Available online: https://www.ohdsi.org/atlas-a-unified-interface-for-the-ohdsi-tools/ (accessed on 29 September 2021).
16. Yao, X.; Lin, Y.; Liu, Q.; Long, S. Efficient and privacy-preserving search in multi-source personal health record clouds. In Proceedings of the IEEE Symposium on Computers and Communication (ISCC), Larnaca, Cyprus, 6–9 July 2015; doi:10.1109/ISCC.2015.7405612. [CrossRef]
17. Woong Park, R. A Clinical Real-World Evidence Sharing Platform Over the Globe. *J. Acupunct. Meridian Stud.* **2020**, *13*. [CrossRef]
18. ISO. ISO 27799: 2016-Health Informatics-Information Security Management in Health Using ISO/IEC 27002. International Organization for Standardization (ISO). 2016. Available online: https://www.iso.org/standard/62777.html (accessed on 29 September 2021).
19. Jones, M.; Bradley, J.; Sakimura, N. *RFC 7519: Json Web Token (JWT)*; Internet Engineering Task Force (IETF): Wilmington, DE, USA, 2015.
20. Hardt, D. *RFC 6749: The OAuth 2.0 Authorization Framework*; International Engineering Task Force (IETF): Wilmington, DE, USA, 2012.
21. Johnson, D.; Menezes, A.; Vanstone, S. The Elliptic Curve Digital Signature Algorithm (ECDSA). *Int. J. Inf. Secur.* **2001**, *1*, 36–63. [CrossRef]
22. Cao, Y.Y.; Fu, C. An efficient implementation of RSA digital signature algorithm. In Proceedings of the 2008 International Conference on Intelligent Computation Technology and Automation (ICICTA), Changsha, China, 20–22 October 2008; doi:10.1109/ICICTA.2008.398. [CrossRef]
23. Neuman, C.; Yu, T.; Hartman, S.; Raeburn, K. *RFC 4120: The Kerberos Network Authentication Service (V5)*; International Engineering Task Force (IETF): Wilmington, DE, USA, 2005.
24. Raisaro, J.L.; Troncoso-Pastoriza, J.R.; Misbach, M.; Sousa, J.S.; Pradervand, S.; Missiaglia, E.; Michielin, O.; Ford, B.; Hubaux, J.P. MEDCO: Enabling secure and privacy-preserving exploration of distributed clinical and genomic data. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2019**, *16*, 1328–1341. [CrossRef] [PubMed]
25. Rieke, N.; Hancox, J.; Li, W.; Milletarì, F.; Roth, H.R.; Albarqouni, S.; Bakas, S.; Galtier, M.N.; Landman, B.A.; Maier-Hein, K.; et al. The future of digital health with federated learning. *NPJ Digit. Med.* **2020**, *3*, 1–7. [CrossRef] [PubMed]
26. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.* **2019**, *10*, 1–19. [CrossRef]
27. Xu, J.; Glicksberg, B.S.; Su, C.; Walker, P.; Bian, J.; Wang, F. Federated Learning for Healthcare Informatics. *J. Healthc. Inform. Res.* **2021**, *5*, 1–19. [CrossRef] [PubMed]
28. Li, Y.; Bai, C.; Reddy, C.K. A distributed ensemble approach for mining healthcare data under privacy constraints. *Inf. Sci. (NY)* **2016**, *330*, 245–259. [CrossRef] [PubMed]
29. Gruendner, J.; Schwachhofer, T.; Sippl, P.; Wolf, N.; Erpenbeck, M.; Gulden, C.; Kapsner, L.A.; Zierk, J.; Mate, S.; Stürzl, M.; et al. Ketos: Clinical decision support and machine learning as a service—A training and deployment platform based on Docker, OMOP-CDM, and FHIR Web Services. *PLoS ONE* **2019**, *14*, e0223010. [CrossRef]
30. OHDSI. OHDSIonAWS-Automation Code and Documentation for Standing Up the OHDSI Toolstack in an AWS Environment. Available online: https://github.com/OHDSI/OHDSIonAWS (accessed on 29 September 2021).
31. Kim, G.L.; Yi, Y.H.; Hwang, H.R.; Kim, J.; Park, Y.; Kim, Y.J.; Lee, J.G.; Tak, Y.J.; Lee, S.H.; Lee, S.Y.; et al. The Risk of Osteoporosis and Osteoporotic Fracture Following the Use of Irritable Bowel Syndrome Medical Treatment: An Analysis Using the OMOP CDM Database. *J. Clin. Med.* **2021**, *10*, 2044. [CrossRef]
32. Zhang, X.; Wang, L.; Miao, S.; Xu, H.; Yin, Y.; Zhu, Y.; Dai, Z.; Shan, T.; Jing, S.; Wang, J.; et al. Analysis of treatment pathways for three chronic diseases using OMOP CDM. *J. Med. Syst.* **2018**, *42*, 1–12. [CrossRef] [PubMed]
33. Forsby, F.; Furuhed, M.; Papadimitratos, P.; Raza, S. Lightweight X.509 Digital Certificates for the Internet of Things. In *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Proceedings of the International Conference on Safety and Security in IoT, Valencia, Spain, 6–7 November 2017*; Springer: Cham, Switzerland, 2017; Volume 242._14. [CrossRef]
34. Plata, I.T.; Panganiban, E.B.; Bartolome, B.B. A security approach for file management system using data encryption standard (DES) algorithm. *Int. J. Adv. Trends Comput. Sci. Eng.* **2019**, *8*. [CrossRef]

35. Stallings, W. The advanced encryption standard. *Cryptologia* **2002**, *26*, 137–139. [CrossRef]
36. OHDSI. HADES-Health Analytics Data-to-Evidence Suite. Available online: https://ohdsi.github.io/Hades/ (accessed on 29 September 2021).
37. Fotohi, R.; Firoozi Bari, S.; Yusefi, M. Securing Wireless Sensor Networks Against Denial-of-Sleep Attacks Using RSA Cryptography Algorithm and Interlock Protocol. *Int. J. Commun. Syst.* **2020**, *33*, e4234. [CrossRef]
38. Gennaro, R.; Goldfeder, S. Fast multiparty threshold ECDSA with fast trustless setup. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; doi:10.1145/3243734.3243859. [CrossRef]