UNIVERSITY OF OULU

FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING
DEGREE PROGRAMME IN WIRELESS COMMUNICATIONS ENGINEERING

MASTER'S THESIS

NETWORK SECURITY FOR AUGMENTED REALITY APPLICATION IN HEALTH CARE SECTOR

Author	Puvaneswaran Amirthan
Supervisor	Adjunct Prof. Madhusanka Liyanage
Second Examiner	Dr. Suneth Namal
Technical Advisor	Dr.Pawani Porambage

July 2019

Amirthan P. (2019) Network Security for Augmented Reality Application in Health Care Sector. University of Oulu, Faculty of Information Technology and Electrical Engineering, Degree Programme in Wireless Communications Engineering, 61 p.

ABSTRACT

The recent advances in mobile devices and wireless communication sector transformed Mobile Augmented Reality (MAR) from science fiction to a reality. Incorporating this MAR technology in health care sector elevates the quality of diagnosis and treatment for the patients. However, due to the highly sensitive nature of the data being circulated in this process, it is also highly vulnerable to the security threats. In the thesis, an architecture is proposed for a MAR health care application based on Multi-access Edge Computing (MEC). This includes key features such as displaying augmented view of patient information on the mobile device, augmenting the X-ray or scan image on top of the patient's actual body parts to assist the doctor, and enabling the doctor to interact with an expert and get real time consultancy.

Based on the proposed architecture, all the possible network security threats are analyzed. Furthermore, a secure key management scheme is proposed for registration and authentication phases to establish a secure end-to-end communication between the participating entities in the system. The security features of the proposed scheme are formally verified by using Automated Validation of Internet Security Protocols and Applications (AIVSPA) tool, Moreover, an informal verification is provided to discuss the protection against other possible attacks. It has justified that the proposed scheme is able to provide the required level of security for the system.

Keywords: MAR, MEC, Health Care, Network Security, Privacy, Formal Verification, Authentication, Key management

TABLE OF CONTENTS

FOREWORD

This master's thesis has been carried out as a part of the curriculum in the Double Degree Masters Program between the Centre for Wireless Communications, University of Oulu and University of Peradeniya, Sri Lanka. The main objective of the thesis is to propose network security for an augmented reality application in the health care sector.

I take this opportunity to express my sincere gratitude to my supervisor Adjunct Prof.Madhusanka Liyanage, Technical Advisor Dr.Pawani Porambage for their immense support and guidance with weekly meetings. Also, I express my sincere gratitude to my second examiner from the University of Peradeniya, Dr.Suneth Namal on his guidance and feedbacks to successfully achieve the thesis objectives. I am also grateful to Professor. An Braeken for providing her expert guidance in designing the proposed security scheme.

Further, I would like to thank Professor. Nandana Rajatheva, Professor Matti Latva-aho, Professor.Kithsiri Liyanage, Dr.Himal Suraweera and other key persons behind making this Double Degree Masters program successful. My appreciation also goes Matti Isoohookana, for all the guidance and motivation provided as a coordinator of this Double Degree Programme.

I also want to thank my mother for her moral support and encouragements, which played a huge role in accomplishing success in my life.

Oulu, 30th July, 2019

Puvaneswaran Amirthan

LIST OF ABBREVIATIONS AND SYMBOLS

3D	Three Dimensional
4G	Fourth Generation Cellular Network Technology
5G	Fifth Generation Cellular Network Technology
AES	Advanced Encryption Standard
AR	Augmented Reality
AV	Augmented Virtuality
AVISPA	Automated Validation of Internet Security Protocols and Application
BAN	Burrows–Abadi–Needham
CA	Certification Authority
CL-Atse	Constraint-Logic-based Attack Searcher
CSR	Certificate Sign Request
DDoS	Distributed Denial-of-Service
DNS	Domain Name System
DoS	Denial-of-Service
DY	Dolev-Yao Model
ECQV	Elliptic Curve Qu-Vanstone
ECDH	Elliptic-Curve Diffie–Hellman
ETSI	European Telecommunications Standards Institute
EXCA	External Certificate Authority
HLPSL	High Level Protocol Specification Language
HMAC	Hash Based Message Authentication Code
ICN	Information Centric Networks
IF	Intermediate Format
LTE	Long-Term Evolution
MAR	Mobile Augmented Reality
MEC	Multi-Access Edge Computing
MR	Mixed Reality
NFV	Network Functions Virtualizations
OF	Output Format
OFMC	On the Fly Model-Checker
QoS	Quality of Service
RSA	Rivest–Shamir–Adleman
SATMC	SAT-Based Model-Checker
SDN	Software Defined Networking
SHA3	Secure Hash Algorithm 3
TA4SP	Tree Automata Based on Automatic Approximation of the Analysis of Security Protocols
UE	User Equipment
UI	User Interface
VM	Virtual Machine
VR	Virtual Reality

1 INTRODUCTION

The next generation of communication systems, 5G opens the door for countless opportunities with the integration of new technologies. Some of the key features of the 5G systems like Multi-Access Edge Computing (MEC), Network Function Virtualisation (NFV), Software-Defined Networking (SDN), helps the 5G networks to provide services with high data rates, extremely low latency with improved Quality of Service (QoS) compared to current 4G LTE networks [1, 2]. The architectural change in the 5G networks, from centralized, cloud-based processing to edge computing techniques reduces the end-to-end latency. This achievement in low latency plays a vital role in making the most demanding and latency constrained applications like Augmented Reality (AR)/Virtual Reality (VR) realizable [3]. There is often confusion among the terminologies VR and AR, the users are fully immersed in a computer-simulated virtual environment in VR applications, whereas in AR, the users will be in the real environment and the computer-generated virtual objects will be brought to the real environment through AR device. When this device is a mobile device, we name this system as a Mobile Augmented Reality (MAR) system. Since the mobile device like smartphones and tabs are widely used and easily accessible, we intended to implement an application in MAR systems. Further details on the MEC server and MAR are discussed in the literature survey.

1.1 Background and Motivation

The core objective of this MAR technology in the target applications is to provide computer-simulated assistance to the users. This assistance helps the users to understand the environment better and make the right decisions. This potential of MAR to redefine the interaction between the user and the environment enhanced the intended output of the application in many sectors such as manufacturing industries, education, and health care. Knowing the potential impact a MAR based health care application could make in human life, ignited our desire to look for a possible opportunity in the sector. Numerous researches have already been carried out to find innovative ways to incorporate MAR into many different applications in the health care sector like remote assistance in surgeries, X-ray rendering, vein image rendering, etc [4].

We have identified a real issue in the rural areas of developing countries, which has the potential for MAR incorporation. Since opportunities are concentrated in capital cities, experts and doctors tend to move there, and it leaves the rural areas with the scarcity of experts and experienced doctors. So the people in those areas are forced to go to city hospitals and wait in ques to get admissions, some times they even miss the chances to get an appointment when needed. Moreover, the inexperienced doctors in rural areas tend to misdiagnose, and it some times end up in severe consequences. These motivated us to come up with the MAR based solution discussed in the upcoming section.

1.2 Research Problem

In order to address these issues discussed in the previous section, we wanted to make the expert accessible for any doctors and patients, regardless of the location, they live in, and also wanted to give some AR assistance for the amateur doctors to improve their diagnosis. We decided to achieve this by proposing a MAR application where the doctor will be able to get the AR assistance in the diagnosis and will be able to get real-time consultations from an expert. The circulated data in this application is highly sensitive; hence, it is mandatory to ensure that security and privacy have never been compromised in our proposed solution.

1.3 Methodology of the Work

To realize the objective of proposing a secure MAR application, first and foremost we define the critical functions of the application such as tracking patient records, augmenting X-ray/scan image of a patient on top of the actual body part being diagnosed, and when needed, doctor receiving real-time consultation and guidance from expert on the diagnosis. Next, we study the typical architecture of a MAR system, their functional components, and process flow. After that, we modify the typical architecture and process flow to cater to our application functions. Depending on the process flow and architecture of our particular application and its essential functions, we analyze all the possible security threats in each entity and communication links and propose possible solutions.

With proper registration and authentication key management scheme, we will be able to establish a secure communication link between entities, and we will be able to mitigate the threats and make our application secure. So we propose such a scheme for each phase in the application. Further, we introduce two or three-way authentications consisting user name and password, facial recognition, registered device MAC addresses, and X.509 certificate. We also use some previous registration as a prerequisite in the following registration, which acts as an additional layer of security, so it is mandatory to follow the same registration order proposed in the scheme. After proposing the scheme, we verify the protocol using formal verification tool AVISPA, and check whether the scheme is safe against man in the middle attack and replay attack. Then, we use informal verification to analyze the security of the application logically.

1.4 Organization of the Thesis

The thesis has eight chapters. The rest of the thesis is organized as follows.

Chapter 2 explains the literature and related researches that have been carried out. This chapter has three sections, the first section explains the concept of MAR, typical architecture and other important details related to MAR. Then the second section MEC explains about MEC concept, reference architecture and essential functions related the elements in that architecture. Then the final section discusses the related works carried out in this domain and discusses the widely accepted formal verification tool which has been used in verifying similar key exchange schemes.

Chapter 3 describes the proposed architecture for our application and discusses the process flow for each scenario of the application that are patient identification and information retrieval, X-ray/scan rendering, and expert opinion.

Chapter 4 analyses all the possible security threats concerning the proposed architecture of the application and suggests possible solutions.

Chapter 5 proposes a registration and authentication key exchange scheme to establish a secure communication link between the entities. In this chapter, the first section elaborates the proposed scheme for the registration phase of each entity in the application, and the second section elaborates the proposed scheme for authentication phase during the interaction between registered entities.

Chapter 6 presents a detailed analysis of formal verification and results for the proposed scheme. Further, it discusses the limitations in the implementation of the actual scheme and how it has been addressed.

Chapter 7 is concerned with the informal verification for the proposed scheme and discusses the attacks prevented by the scheme.

Chapter 8 includes the evaluation of the thesis on meeting the objectives and proposes some future directions for the research.

Finally, Chapter 9 discusses the conclusion of the thesis.

2 LITERATURE REVIEW

The new technological advancements enriched our interaction and interpretation with the environment around us. Based on the interaction and interpretation, the reality-virtuality continuum can be categorized into four subcategories as Reality, Augmented Reality (AR), Augmented Virtuality (AV), and Virtual Reality (VR) [5].

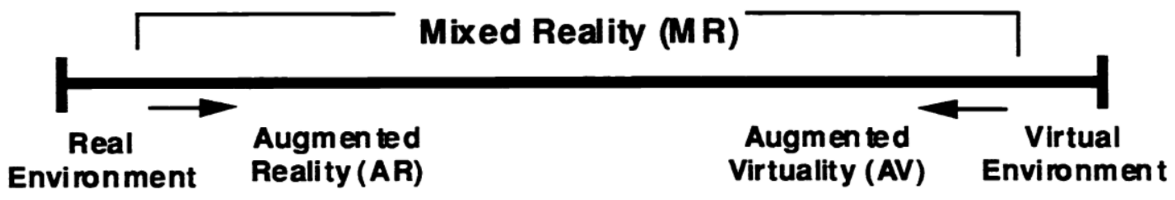


Figure 1. Reality-virtuality continuum [5].

In VR, the user is fully immersed in a computer-generated virtual environment, and he has no interaction between the real environment. However, in AR and AV, which is commonly called as mixed reality, the virtual and real environments are mixed and displayed on a chosen device. The way the environments are mixed distinguishes AV and AR. In AV, the current state of the real-world objects are blended and updated in the virtual environment the user is immersed. Whereas in AR, the generated virtual objects are blended in real-time with the real environment and displayed on the display of a chosen device. The fundamental process of any augmented reality systems can be summarized into six subsystems as image acquisition, virtual model generator, mixing realities subsystems, display, manipulator and a tracking subsystem [6]. Recent mobile technology advancements in built-in cameras, sensors, computational resources and cloud, edge-based data processing and storage facilities have made AR possible on mobile devices. AR on mobile devices such as mobile phones, tabs, google glass, other portable electronic devices with the ability to host AR applications are called as MAR [7, 8]. From [8], we can say that the characteristics of a MAR system can be defined as follows:

1. Combines real and virtual objects in a real environment.
2. Interactive in real-time.
3. Registers and align real and virtual objects with each other.
4. Runs and displays the augmented view on a mobile device.

can be categorized as MAR. According to the large number of published studies [6, 7, 8, 9, 10, 11, 12, 13, 14, 15], a typical architecture of a MAR system can be drawn as in Figure 2 and the components of typical MAR system can be shown as in Figure 3.

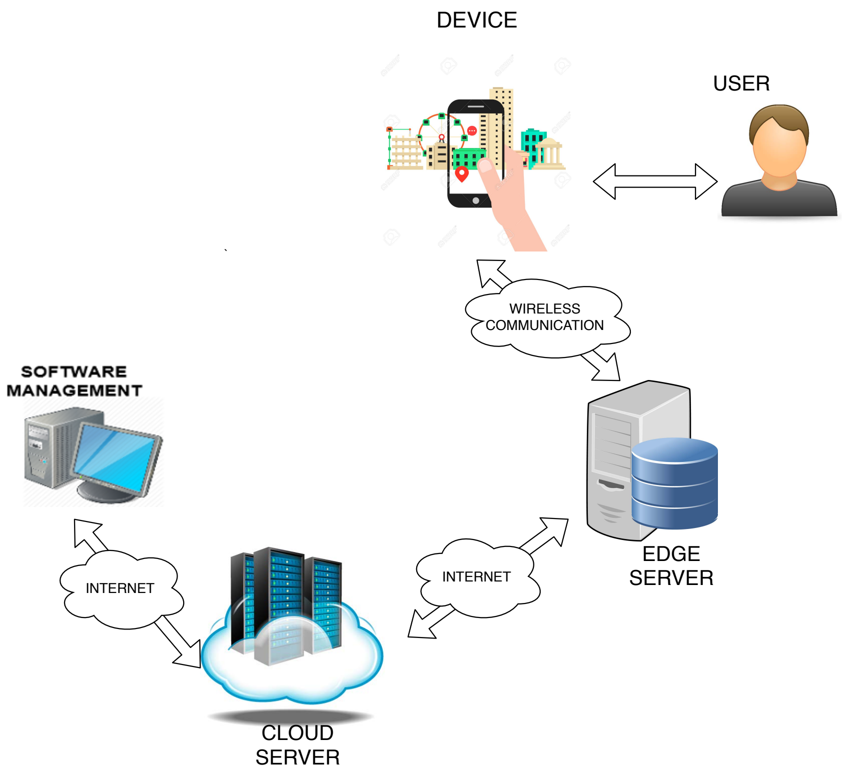


Figure 2. Typical architecture of MAR system.

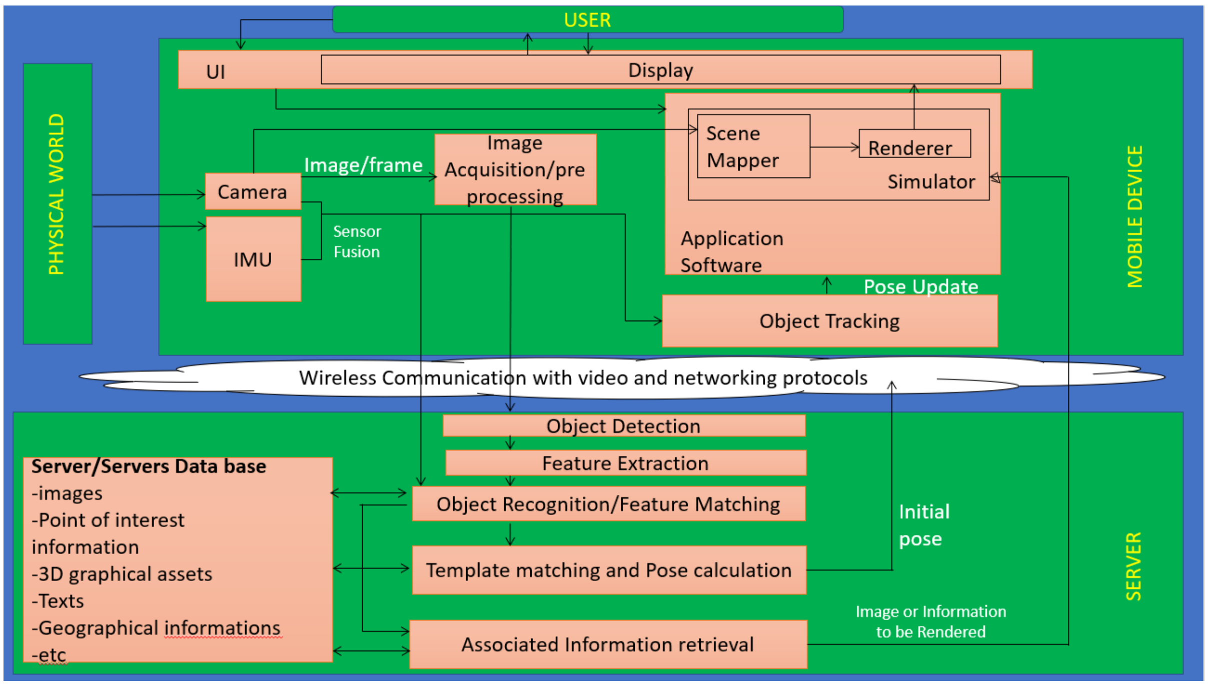

USER
UI
Display
PHYSICAL WORLD
Camera
IMU
Image/frame
Image Acquisition/pre processing
Sensor Fusion
Scene Mapper
Renderer
Simulator
Application Software
Pose Update
Object Tracking
MOBILE DEVICE
Wireless Communication with video and networking protocols
Object Detection
Feature Extraction
Object Recognition/Feature Matching
Template matching and Pose calculation
Associated Information retrieval
Server/Servers Data base
-images
-Point of interest information
-3D graphical assets
-Texts
-Geographical informations
-etc
Initial pose
Image or Information to be Rendered
SERVER

Figure 3. Components of typical MAR system.

As shown in Figure 3, the initial image acquisition from the user's environment is done through the camera of the mobile device and passed on to the server. In the server, the target object is detected, and the features are extracted from the image. Then the extracted features are matched with the databases, and the target object is identified. Then according to the software framework, the associated information or image to be rendered is retrieved and sent back to the device. Meantime, fused sensor input is passed on to the server, where the template matching and initial pose calculation is done by comparing it with the stored data. Then the calculated initial pose is passed on to the device. The device simulator renders the inputs from the server on top of the real-time camera inputs and displays it to the user. Object tracking part of the software compares the real-time fused device sensor input concerning the initial pose and update the rendered information or image, concerning the user's view of the scene. Due to the latency constraint nature [12] of the MAR, a care-full design is needed with optimally offloading the tasks to the server [15].

2.1 Multi Access Edge Computing (MEC)

As per the definition defined by ETSI, bringing the cloud computing capabilities to the edge of the mobile network closer to the users is called mobile edge computing [16]. The comparison of cloud and edge computing is illustrated in Figure 4. MEC has some advanced features such as low latency, proximity, high bandwidth, real-time insights into radio network information and location awareness which makes it a game-changer in many sectors such as consumer, enterprise and health care [17].

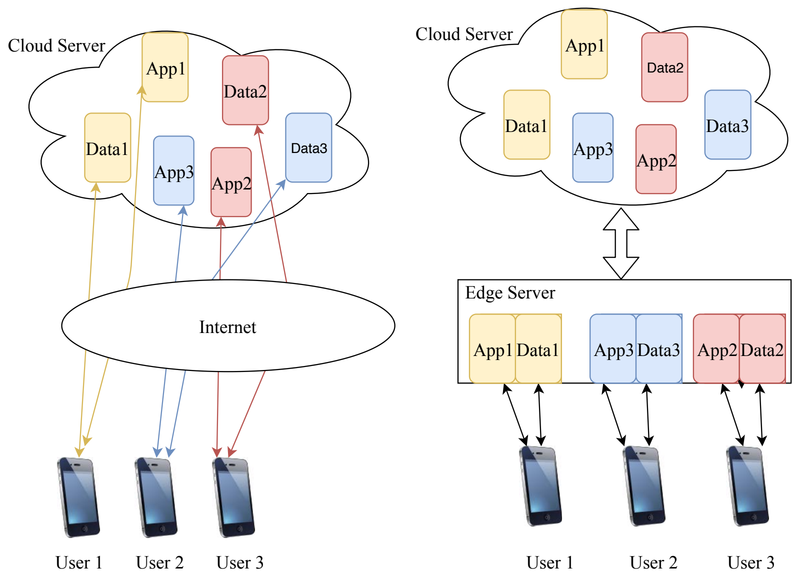


Figure 4. Comparison of cloud and edge computing.

Recent advancements in the Network Functions Virtualizations (NFV), Information-Centric Networks (ICN), and Software Defined Networking (SDN) enabled the implementation of MEC, based on virtualized platforms [18]. MEC reference architecture, as described by ETSI [16], is shown in Figure 5. This architecture allows MEC applications to run as Virtual Machine(VM) on top of virtual infrastructure. The MEC has two host level management. First one is mobile edge platform manager, which manages the life cycle of applications, application rules and requirements including service authorizations, traffic rules, DNS configurations, and resolving conflicts. The second one is the virtualization infrastructure manager, which manages the allocation management and releasing of the resources of the virtualization infrastructure. The User apps and customer-facing service portal, request the operations support systems for the service. Then the operations support system evaluates the validity of the request and forwards the valid requests to the MEC orchestrator. Orchestrator maintains an overall view of the deployed MEC hosts, available resources available MEC services and topology [17]. Depending on the service required, orchestrator will make an optimal decision on the process of the request.

The AR application needs low latency and high-speed data processing to provide the correct and excellent user experience. Hence, incorporating MEC server instead of a centralized server, for data processing and data access, reduces latency due to long propagation distances and complexity in the processing [17].

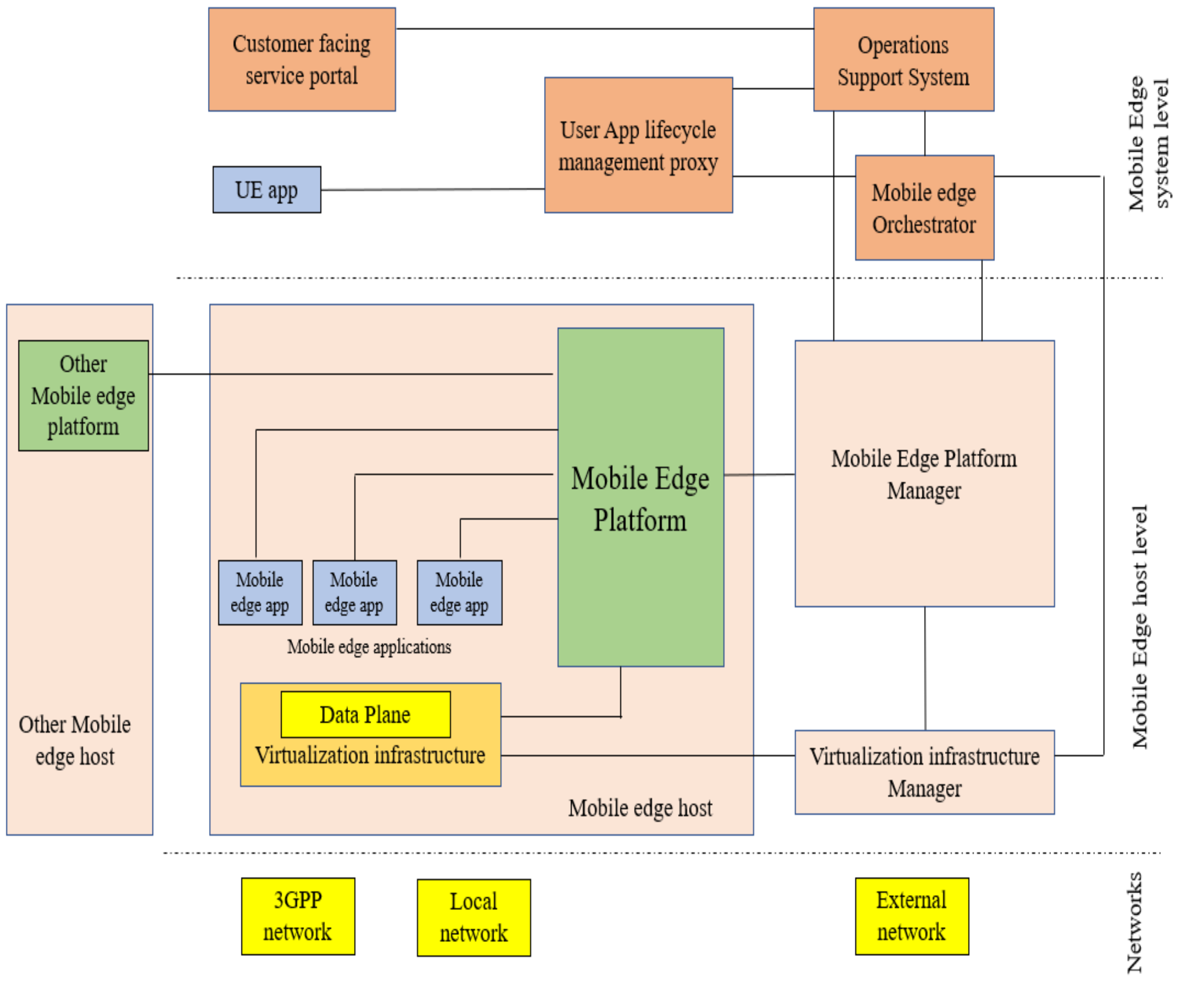


Figure 5. The Reference architecture of MEC [17].

2.2 Related Works

Due to the merits of MAR systems, the technology has its presence in many sectors such as tourism and navigation, entertainment and advertisement, training and education, geometry modeling and scene construction, assembly and maintenance, information assistant management and health care applications [7, 9]. In this research, we focus on Incorporating MAR into the health-care sector as it enhances better outcomes in patient care [9] and has a huge impact on human lives. A considerable amount of research has already been done in this area, and researchers have proposed some innovative ways to incorporate AR technology into the health care sector. The study [19] discusses on AR application in medical education in subject areas like anatomy, surgery, and forensic medicine and it also discusses the applicability of AR in assisting in actual medical procedures in laparoscopic surgery, endotracheal intubation, joint injections, and assistance in placing local anesthesia. The research [20] discusses on optical-see through augmentation for laparoscopic surgery. The studies [21, 22] discuss augmenting X-ray, MR scans imagery with the real objects. The literature [23] discusses the application of AR in surgeries. The extensive survey in [4] refers to many other AR applications and researches conducted in the health care sector. It mentions applications such as, the vein viewer which projects patient's vascular network onto the skin to help with needle insertion, assisting applications for patients in both physical and neurological rehabilitation, applications which could augment the information from medical scan images to a surgeon in a convenient manner during surgeries, applications which could be used for educational training such as visualizing human anatomy, visualizing 3D lung dynamics and training of laparoscopy skills. The work carried out in [24] discusses a virtual interactive presence and augmented reality platform that allows a remote surgeon to deliver real-time virtual assistance to a local surgeon, over a standard Internet connection.

Based on the works done in this AR applications in the health care sector, we have identified few key features to be included in our proposed MAR based health care application to enrich the health care services provided to the people regardless of the place where they are living.

2.3 Formal Verification Methods

Security protocols have to be verified before they are actually implemented. There are no such ways to prove that a protocol is perfectly safe and unbreakable. However, the strength of a protocol can be verified with the measure of relative complexity in breaching the security properties [25]. The formal verification method is used to verify the protocols by logically and mathematically model the system and security systems, and validate whether it achieves the expected level of security [26]. The extensive study carried out in [27] discusses formal verification methods. For a complex protocol, security analysis involves a large number of steps. Hence, the task needs to be automated. Depending on the automation techniques, the verification method is distinguished. Based on this, there are two notable formal verification methods. First one is logics of beliefs (BAN Logic), a modal logic system based on beliefs of the entities in the protocol, and the second one is Dolev-Yao (DY) model, a discrete state transition system which tracks the messages instead of beliefs. In this model, the behaviors of the agents participating in the protocol are defined with send, receive messages, whereas the behavior of the attacker is left undefined. The DY model could be formalized in many different ways, as discussed in the literature [27]. However, automated by state exploration is highly suitable for our application. This technique runs and looks for all possible violations of the defined security properties in the model runs, and when the desired property is violated in any of the model runs, the analysis will conclude that the protocol is unsafe. In the thesis, a widely used state exploration technique based tool named AVISPA is used for verification.

2.3.1 AVISPA

Further, the researches [28, 29, 30, 31] uses Automated Validation of Internet Security sensitive Protocols and Applications (AVISPA), a widely known formal verification tool to validate their proposed security protocols. The architecture of the AVISPA tool is shown in Figure 6.

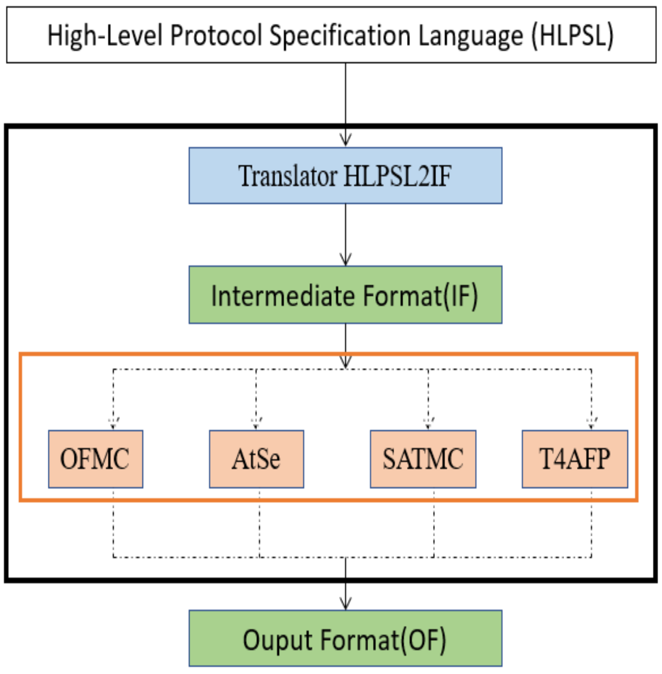


Figure 6. The architecture of AVISPA tool [29].

From the references [29, 32, 28], the following understandings were acquired. In AVISPA, the user specifies the security problem or in other words, a protocol with expected security goal in High level protocol specification language (HLPSL). The role based script written in HLPSL is then translated into Intermediate format(IF) using HLPSL2IF translator. IF specification describes an infinite state transition system which is suitable for formal analysis. This IF specification is fed to one of the four back-ends of the AVISPA tool. Each back end has different techniques to search the corresponding infinite state transition system for states that represent the attacks on the intended properties of the protocols. The current version of the tool has four back ends.

1. On the Fly Model Checker OFMC.

2. Constraint Logic based Attack Searcher CL-Atse.
3. SAT based Model Checker SATMC.
4. tree automata based on automatic approximation of the analysis of security protocols TA4SP.

These back ends, analyze protocols and check whether our security goals are met. The intruder is modeled using Dolev Yao (DY) model [33], which allows the intruder to act as an authorized entity with the knowledge we declared. After the analysis, the tool provides the result as SAFE or UNSAFE. When the output is UNSAFE, we can simulate the attack simulation and see how the attack is being carried out by the intruder. We can also simulate the protocol by using protocol simulation and check the variable values using variable monitoring tab. In this way, we can ensure that the protocol is being successfully simulated. Further explanations about the tool and declarations are discussed in the formal verification chapter.

3 PROPOSED ARCHITECTURE

From the analysis of current issues in health care systems, we have identified the following key issues.

1. Inexperienced doctors face difficulties in consulting patients by checking X-ray and scan reports.

2. Patients and doctors in rural areas require the second opinion from expert surgeons working internationally, but it is not practical in most cases.

We are proposing a MAR application to address these issues together with additional features. The key features of our MAR application are as follows.

1. Doctor can use this application and see the augmented view of the patient's X-ray/scan report on top of the patient's affected area, and he can perform precise medical check-ups.

2. Doctor/expert can access previous medical histories, reports, and other information with the patient's permission and see the augmented results on the screen.

3. Doctor can request for second opinions from a registered expert, and then the expert can join and interact with the doctor regardless of his location. The augmented view of the expert's inputs are displayed on doctor's device in real-time.

Based on the AR architectures discussed in [13, 14, 15] architecture of our MAR application is proposed. For easy understanding, the typical architecture of MAR mentioned in Figure 2 and Figure 3 is modified in accordance to each main scenarios of our application as illustrated in the following context.

Figure 7 shows the architecture of the first scenario that is patient identification and information retrieval. Registered doctor authenticates himself and logs into the application, after that he inputs patient identity details through the User Interface (UI) and requests for past medical records, X-ray, scan reports, and other information. The request is then passed on to the edge server. Edge server then verifies doctor, patient log in credentials and retrieve all the data related to the patient from cloud and stores it in the edge server. Then the edge server sends back the specific information requested by the doctor, to the doctor's mobile device. The simulator function of the application software in the mobile device grabs the user interface input, selections, the input from the edge server and renders it on top of real-time camera input on the display.

Figure 8 shows the architecture of the second scenario that is augmenting X-Ray / Scan report on top of patient's actual body part. Here the doctor places a marker on the patient's body part he wants to examine. Then he holds the device on top of the body part, and requests the application for an augmented view of X-ray or scan report. The service request and the device camera input are passed on to the edge server. The edge server then retrieves the respective X-ray or scan report from its database, and calculates the initial pose of the doctor. After that it passes both calculated initial pose, and the image to be rendered, to the requested doctor's mobile device. The simulator function in the mobile device simulates the scene by rendering the image received from the edge server on top of the device camera input. Further, the object tracking part updates the pose of the doctor based on the current camera input, and initial pose information

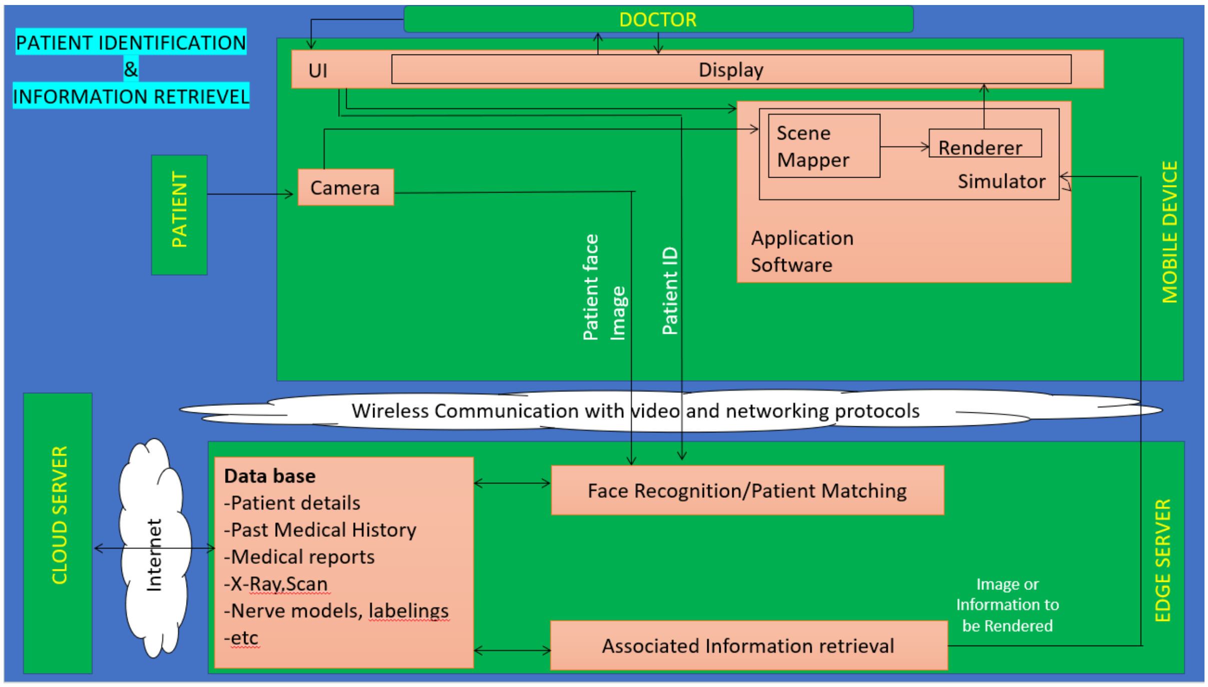


Figure 7. Patient identification and information retrieval.

it received from the edge server. Based on this pose update, the simulator adjusts the simulation on display and produces a realistic augmented view for the doctor with respect to his movements.

Figure 9 shows the architecture of the third scenario that is doctor seeking an expert opinion. While consulting the patient, the doctor may wish to have a second opinion in diagnosis or report analysis from an expert. Doctor then uses the user interface of the application and requests the expert for his consultancy. When the expert accepts the invitation, the final rendered output with a marker that has been displayed in the doctor's device is video streamed to the expert's device. The expert then interacts with the doctor regarding the issue. When needed, he requests the edge server for the patient's medical information through the user interface. Expert's device then retrieves the information from the edge server and displays the augmented output in a similar manner explained in scenario 1 Figure 7. The expert then provides his real-time interactive consultation to the doctor, his real-time markings on his display on X-rays, scan reports are streamed back to the doctor's device. In the doctor's device, the simulator function maps expert's consultation inputs with respect to the marker in the affected area of the patient and renders it in the output shown in the display.

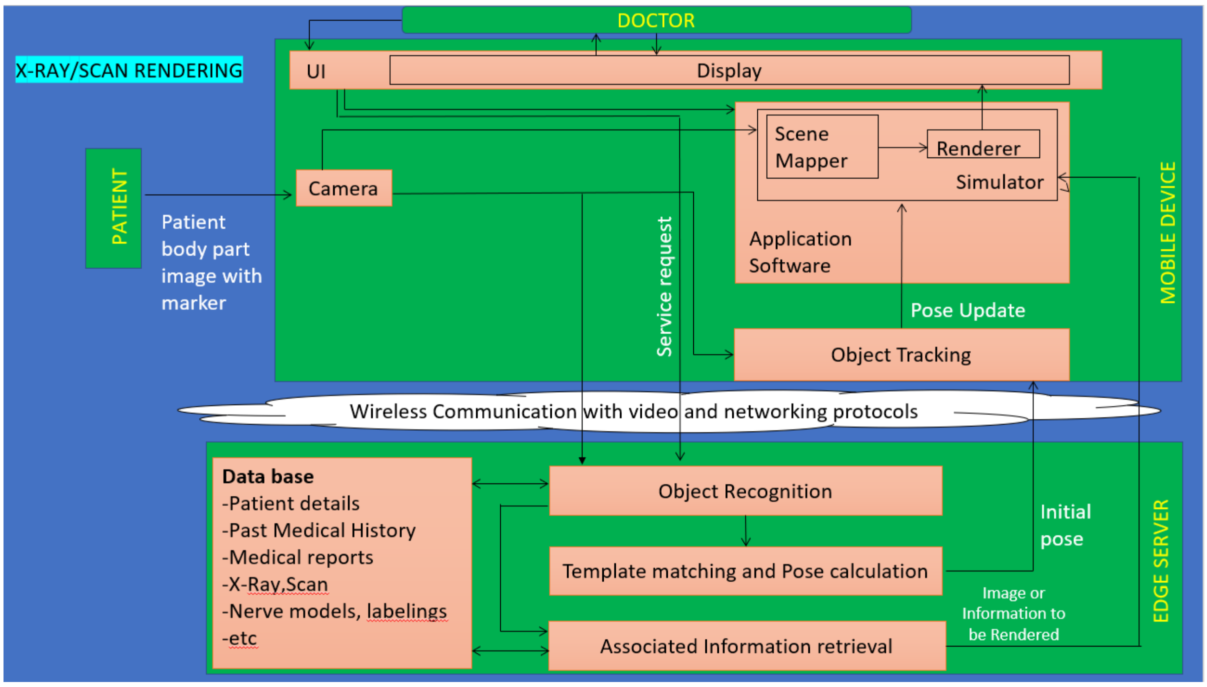


Figure 8. X-Ray/scan rendering.

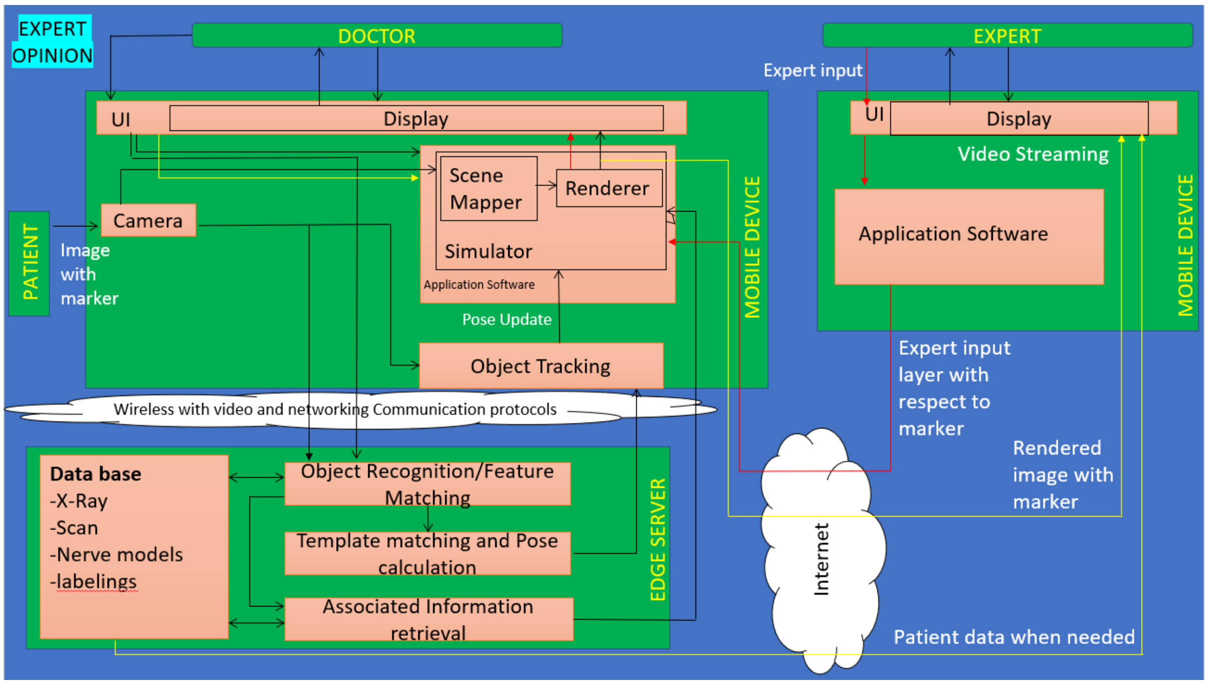


Figure 9. Expert opinion.

4 SECURITY ANALYSIS OF MEC BASED MAR ARCHITECTURE

All the possible security threats for the proposed architecture has been analyzed, and viable prevention techniques have been discussed in this section. The threat vector model for our application architecture is shown in Figure 10.

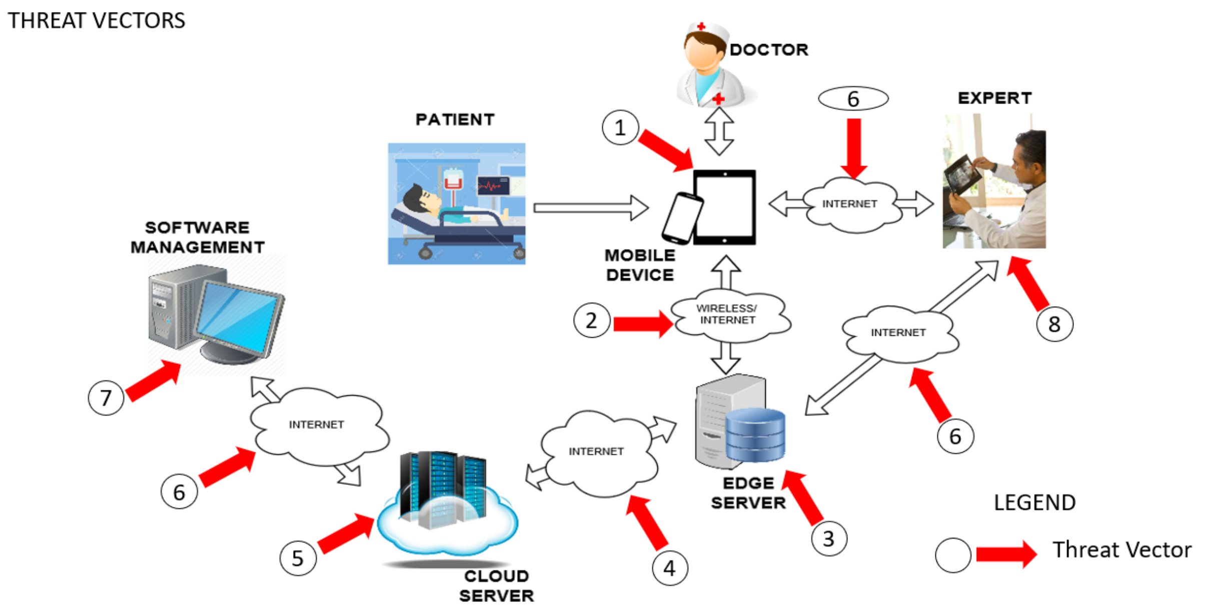


Figure 10. Security threat vectors.

Threat vector 1: Attack on mobile device

The attacker can use third-party applications to gain access to the device and its sensors and obtain information about the surroundings and track the patient's location and misuse them [34]. Similar kind of attack named as place raider attack has been discussed in [35]. An attacker can manipulate the outputs and perform clickjacking attack [36] to suddenly display sensitive request and make the user give permission which could affect the privacy of the data being handled.

possible solutions: Sensor access can be limited in a way that they start to function only when the marker on the patient's body part is scanned. Third-party access could be restricted while using this app. Output policies could be defined to detect abnormalities in the display.

Threat vector 2: Attack on the network between edge server and mobile device

An attacker can perform a man in the middle attack and access the sensitive data like X-ray, patient's medical history and exploit them. He can also generate virtual traffic and perform a DoS attack.

possible solutions: Proper authentication and encryption scheme should be followed, puzzles could be used to prevent dos attacks.

Threat vector 3: Attack on edge server

Data: The previous studies [37, 38] discuss the possible attacks that could be performed in edge server. An internal or external adversary can access edge server and view data, tamper them, redirect to another rogue storage, which will breach the data confidentiality and integrity of the data. The attacker could gain access to the system by performing kernel-level operations and escalate the privilege for the administrator. He could also perform a DoS attack and prevent the edge server from providing data services.

possible solutions: Data could be stored, processed, and transmitted in the encrypted domain, using the techniques described in [39, 40, 41]. Proper authentication scheme could be implemented, and puzzle challenges could be used to prevent DoS attack.

Services: Apart from data, the attacker can also target for other services offered by the edge server. Attackers could use third-party applications on the same platform as network functions to infiltrate the platform and affect network functions like traffic control rules, DNS handling, service registry running on the platform. Attackers can negatively use the application-aware performance optimization; for example, they can starve the competitor applications and their customers of radio resources which will force them to end up in denial of service for their customers. Attackers could also fill the local caches with useless content unusable by subscribers, and exhaust resources needed by network functions. Furthermore, they can spoof or modify the instructions transmitted by MEC orchestrator and disrupt the virtualization infrastructure manager and mobile edge platform manager.

possible solutions: Application quality assurance framework could be used only to allow trusted application, the edge computing applications and network functions could be run in robustly segregated virtual machines, a secure authentication and encryption mechanism could be implemented, and priority policies can be set to allocate a dedicated cache for AR application while using, and machine learning techniques could also be used to detect anomalies in content requests.

Threat vector 4: Attack on the network between edge server and cloud server

Adversaries can perform a man in the middle attack and access sensitive data or manipulate the medical data and mislead the doctor and the expert. Moreover, they can also overload the cloud server with multiple streams of data flow with multiple virtual machines and perform a DDoS attack.

possible solutions: Strong encryption and authentication scheme could be implemented, puzzle-based prevention could be used to prevent DDoS attack, machine learning techniques could be used to detect anomalies in the traffic patterns and divert or filter the unwanted packets.

Threat vector 5: Attack on cloud server

With attacks discussed in [42], internal or external adversaries could access cloud service and view, tamper and redirect data to another rogue storage. Adversaries could impersonate the edge server and access sensitive information from the cloud. As discussed in [43] adversaries could perform cloud Malware injection attack and side-channel attack, and also the adversaries could physically damage the cloud resources.

possible solutions: Strong encryption and authentication scheme could be implemented, backup of data with proper encryption could be kept, resources could be kept in high secured zones with limited access, a hypervisor could be deployed to validate and integrate virtual machines, a combination of virtual firewall appliances could be used to prevent side-channel attacks.

Threat vector 6: Attack on internet connection between entities

An attacker can perform a man in the middle attack and access application control related information and input malicious software updates to make the application completely insecure, and the attacker could also perform a DoS attack.

possible solutions: Proper encryption and authentication scheme could be implemented, and puzzle-based challenge could be used to prevent DoS attack.

Threat vector 7: Attack on software management

Internal employees could misuse their access to modify the software and upload malicious updates and create loopholes or access the data storage and retrieve sensitive data and exploit them.

possible solutions: Source code access could be limited to very few people in the company and could be protected with facial recognition-based authentication. All source codes and other data could be stored in an encrypted form with proper backup storage.

Threat vector 8: Attack on expert's device

Malicious third-party apps could be used to access inputs from the doctor, edge server, and outputs from the expert, and then the sensitive information about the patient and the consultation details could be exploited.

possible solutions: The access of the third-party applications could be restricted while using the app. Output policies could be defined as discussed in threat vector 1. Highly secure and efficient encryption and authentication scheme could be used.

In order to mitigate these threats, a registration and authentication scheme is proposed in the next chapter.

5 PROPOSED REGISTRATION AND AUTHENTICATION SCHEME

Our protocol has two main phases, the registration phase, and the authentication phase. In the registration phase, the details of the entities are uploaded to the certificate authority. Then the implicit certificate which has the public key of the entity is obtained from the certificate authority. The obtained certificate is used by the entity to derive its private key.

Table 1. Notations used in the scheme

Notation	Description
Q_{EXCA}	Public key of External Certificate Authority
d_{EXCA}	Private key of External Certificate Authority
N_s	Random number generated by server
N_c	Random number generated by client
k_{dos}	Puzzle difficulty level
$X509_{EXCA}$	X509 certificate of the external certificate authority
X	solution of the Puzzle
Y	solution of the hash value
gcd	Greatest common devider
Q_{Edge}	Public key of edge server
d_{Edge}	Private key of edge server
CSR	Certificate signing request
H	Oneway hash function
r_{Edge}	random integer value generated by Edge
R_{Edge}	Elliptic curve point generated by Edge
G	Generator function
r_{Admin}	random integer value generated by Admin device
R_{Admin}	Elliptic curve point generated by Admin device
R_{device}	Elliptic curve point generated by device
K	Secret key for HMAC function
$Cert_{Admin}$	Implicit certificate of Admin device
$Cert_{device}$	Implicit certificate of doctor/expert device
e	Integer used to keep hash value of implicit certificate
s	Private key reconstruction value
r_{User}	random integer value generated by registered user
R_{User}	Elliptic curve point generated by registered user
K_{AES}	Session key
$Q_{Ddevice}$	Public key of Doctor device
$d_{Ddevice}$	Private key of Doctor device
$Cert_{Ddevice}$	Implicit certificate of Doctor device
$Q_{Edevice}$	Public key of Expert device
$d_{Edevice}$	Private key of Expert device
$Cert_{Edevice}$	Implicit certificate of Expert device
Nonce	Random cryptographic nonce

Likewise, in the authentication phase, the authenticity of the entities is verified using their implicit certificates and other additional authentication mechanisms like facial recognition, and log-in credentials. Finally, a session key is established safely in this phase. In the registration process of edge server, external certificate authority acts as the certificate authority. However, in all the other cases, the edge server acts as the certificate authority to the participating entities.

5.1 Registration Phase

The main tasks in this phase are, registering the entity's details by the Certification Authority (CA), and providing a certificate for authentication purpose inside the network. For all the other cases the edge server acts as CA and saves entity's details and provides ECQV implicit certificate[44] to the requesting entities. In our scheme, we have used some registered entity's credentials as a pre-requisite for other entity's registrations, so the registration process has to follow the defined order. First and foremost, the edge server has to be registered with external CA as illustrated in Figure 11, here external CA registers edge server and provides X.509 certificate to it. Then secondly, the network administrators from hospital and software company should register themselves together with their computers/devices on their edge servers as illustrated in Figure 12. After that on the company side, their registered admin will register the company workers and their computers on their edge server, as shown in Figure 13. On the hospital side, the registered admin will register the doctor, expert devices and then they will register the doctors and experts on edge server through an already registered device as explained in Figure 14 and Figure 15. Then finally, the network admin registers the patient on the edge server through his registered device, as shown in Figure 16. The boxes with red letters in the figures indicate the stored information on that side. In the scheme, similar puzzle-based challenge described in [45] is used to prevent DoS attack where the edge server sends a puzzle and expects the client to solve it to show it is a legitimate client. From the understandings of [46, 47, 48, 49, 50, 51], we found ECDH (Elliptic-Curve Diffie–Hellman) is highly secure and faster and well suited key exchange scheme for our application.

Step 1: Edge server registration

As shown in Figure 11, edge server initiates the conversation with the external CA with a hello message and cipher suites it supports. External certificate authority chooses a random number N_s, then chooses k_{dos} depending on the severity of the DoS attack and sends back to the edge server together with a hello message, selected cipher suite, and its X.509 certificate to authenticate itself as legit. Upon completing this handshake with CA, the edge server generates a random number N_c and finds a solution X for the puzzle in a way that when N_c, N_s, X are concatenated and hashed with Q_{EXCA} as key, it gives a value with first k_{dos} bits as 0.

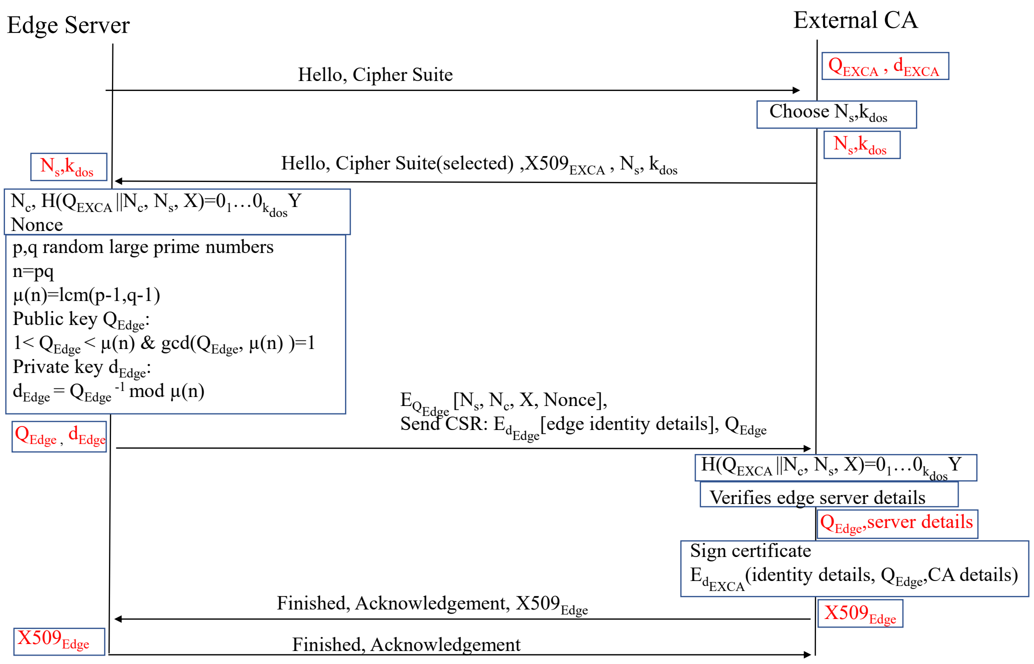

Edge Server
External CA
Q_{EXCA}, d_{EXCA}
Hello, Cipher Suite
Choose N_s,k_{dos}
N_s,k_{dos}
Hello, Cipher Suite(selected) ,$X509_{EXCA}$, N_s, k_{dos}
N_s,k_{dos}
N_c, $H(Q_{EXCA} \| N_c, N_s, X)=0_1 \ldots 0_{k_{dos}} Y$
Nonce
p,q random large prime numbers
n=pq
μ(n)=lcm(p-1,q-1)
Public key Q_{Edge}:
$1 < Q_{Edge} < \mu(n)$ & $\gcd(Q_{Edge}, \mu(n))=1$
Private key d_{Edge}:
$d_{Edge} = Q_{Edge}^{-1} \bmod \mu(n)$
$E_{Q_{Edge}}$[N_s, N_c, X, Nonce],
Send CSR: $E_{d_{Edge}}$[edge identity details], Q_{Edge}
Q_{Edge}, d_{Edge}
$H(Q_{EXCA} \| N_c, N_s, X)=0_1 \ldots 0_{k_{dos}} Y$
Verifies edge server details
Q_{Edge},server details
Sign certificate
$E_{d_{EXCA}}$(identity details, Q_{Edge},CA details)
Finished, Acknowledgement, $X509_{Edge}$
$X509_{Edge}$
$X509_{Edge}$
Finished, Acknowledgement

Figure 11. Message flow for the registration phase of edge server.

Then the edge server generates its public and private key pairs based on RSA key generation algorithm [52]. Then the edge server encrypts the N_c, N_s, X, Nonce with the public key of the external certificate authority(EXCA), to make sure it is only accessible by that certificate authority and sends it to the server together with the certificate sign request (CSR) [53]. The CSR also includes the edge identity details signed by the edge server and the derived public key of the edge server. Once the EXCA receives this message, it verifies the puzzle answer by doing the same hash function and comparing with the stored k_{dos} value. After that, EXCA sends and X.509 identity certificate digitally signed with the private key of EXCA together with the finished message. The message flow ends with the finished message from the edge server to EXCA.

Step 2: Network admin registration

As shown in Figure 12, network admin from his computer or device initiates the conversation with the edge server with a hello and cipher suites it supports. The edge server then chooses a random number r_{Edge1} and computes an elliptic curve point R_{Edge1} = G.r_{Edge1}, here the G is a known elliptic curve function for both parties. Then same as in the previous case, the edge server generates a puzzle for DoS attack prevention and sends it back to the network admin together with the newly generated elliptic curve point, and the X.509 certificate to network admin. Network admin solves the puzzle the same way the edge server did in step 1 and then generates two random numbers r_{Admin1} and r_{Admin2}. Then it computes one key for the hash function K=r_{Admin1}.R_{Edge1}, an elliptic curve point R_{Admin1} = G.r_{Admin1} to compute the key K in edge server end, an elliptic curve point to calculate the certificate R_{Admin2} = G.r_{Admin2} and a Nonce for message freshness. HMAC is produced by hashing the R_{Admin2}.Nonce with two hash functions with keys Q_{Edge} and K. After computing all these, the puzzle ans as in step 1 is encrypted with the public key of the edge server and send together with R_{Admin1}, R_{Admin2}, Nonce to indicate message freshness and HMAC value to ensure that the message is delivered without any alterations.

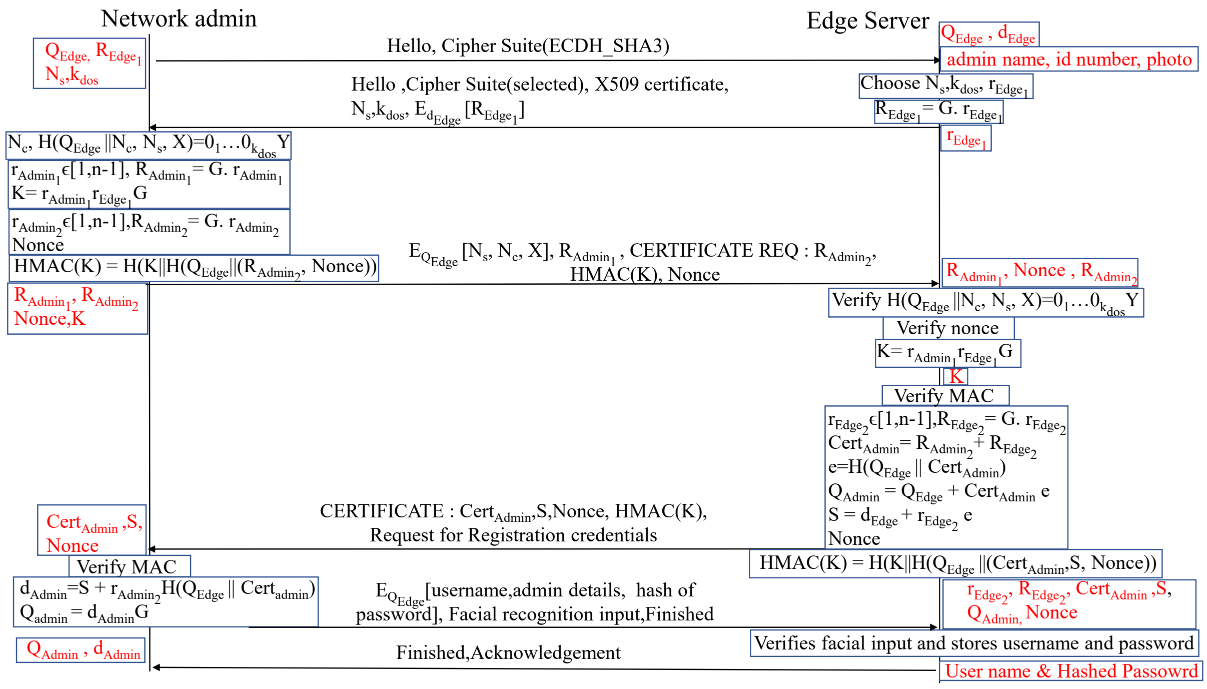

Network admin
Edge Server
Q_{Edge}, R_{Edge_1}
N_s, k_{dos}
Hello, Cipher Suite(ECDH_SHA3)
Q_{Edge}, d_{Edge}
admin name, id number, photo
Hello ,Cipher Suite(selected), X509 certificate, N_s, k_{dos}, $E_{d_{Edge}}[R_{Edge_1}]$
Choose N_s, k_{dos}, r_{Edge_1}
R_{Edge_1} = G. r_{Edge_1}
r_{Edge_1}
N_c, $H(Q_{Edge} \| N_c, N_s, X) = 0_1 \ldots 0_{k_{dos}}$ Y
$r_{Admin_1} \in [1,n-1]$, R_{Admin_1} = G. r_{Admin_1}
K= $r_{Admin_1} r_{Edge_1}$G
$r_{Admin_2} \in [1,n-1]$, R_{Admin_2} = G. r_{Admin_2}
Nonce
$HMAC(K) = H(K \| H(Q_{Edge} \| (R_{Admin_2}, Nonce))$
$E_{Q_{Edge}}[N_s, N_c, X]$, R_{Admin_1}, CERTIFICATE REQ : R_{Admin_2}, HMAC(K), Nonce
R_{Admin_1}, Nonce , R_{Admin_2}
R_{Admin_1}, R_{Admin_2}
Nonce,K
Verify $H(Q_{Edge} \| N_c, N_s, X) = 0_1 \ldots 0_{k_{dos}}$ Y
Verify nonce
K= $r_{Admin_1} r_{Edge_1}$G
K
Verify MAC
$r_{Edge_2} \in [1,n-1]$, R_{Edge_2} = G. r_{Edge_2}
$Cert_{Admin} = R_{Admin_2} + R_{Edge_2}$
$e = H(Q_{Edge} \| Cert_{Admin})$
$Q_{Admin} = Q_{Edge} + Cert_{Admin}$ e
$S = d_{Edge} + r_{Edge_2}$ e
Nonce
CERTIFICATE : $Cert_{Admin}$, S, Nonce, HMAC(K), Request for Registration credentials
$Cert_{Admin}$, S,
Nonce
Verify MAC
$HMAC(K) = H(K \| H(Q_{Edge} \| (Cert_{Admin}, S, Nonce))$
$d_{Admin} = S + r_{Admin_2} H(Q_{Edge} \| Cert_{admin})$
$Q_{admin} = d_{Admin}$G
$E_{Q_{Edge}}$[username,admin details, hash of password], Facial recognition input,Finished
r_{Edge_2}, R_{Edge_2}, $Cert_{Admin}$, S,
Q_{Admin}, Nonce
Q_{Admin}, d_{Admin}
Finished,Acknowledgement
Verifies facial input and stores username and password
User name & Hashed Passowrd

Figure 12. Message flow for the registration phase of network admin.

Upon receiving this message, the edge server verifies the puzzle answer, and then it verifies the nonce to ensure the message is fresh. After that it computes the key by K= $R_{Admin1}.r_{Edge1}$ for the second hash function used in computing the HMAC value. Once the HMAC value is verified, the edge server uses the elliptic curve point R_{Admin2} and newly generated random number r_{Edge2} to compute the certificate. The certificate is computed, as shown in Figure 12. S is the private key reconstruction value that will be used by the network admin to compute his private key.

Edge then computes the HMAC value of $Cert_{Admin}$.S.Nonce and sends it together in a message with $Cert_{Admin}$, S, Nonce and a request for network admin to provide his registration details, user name and password he wishes to use in next log-in, and a facial recognition input to verify that the message is originated from admin only. Upon receiving this message, the network admin verifies the MAC value. Then by using private key reconstruction value, $Cert_{Admin}$, and the previously generated r_{Admin2}, network admin computes the private and public keys as shown in the figure.

After obtaining the keys, he uses the public key of the edge to encrypt the admin registration details together with MAC address of the device he is using, his chosen user name and a hashed password and sends it back to the edge server with his facial recognition input. Edge server receives this message, and it checks the facial recognition input with the already stored image details, and ensures the message is from admin, and then proceed to save the registration details securely, and finally, the message flow finishes with the finished and acknowledgment message from the edge server to admin.

Step 3: Company workers and devices registration

As shown in Figure 13, the registered network admin responsible in the software company, initiates the registration with a hello message. First few steps of handshake continue as explained in previous steps, then the admin will use his log-in credentials by sending them encrypted with the public key of the edge server. Once the edge server verifies the user name and password, it asks the network admin to send the registration details together with admin's facial input as a final verification.

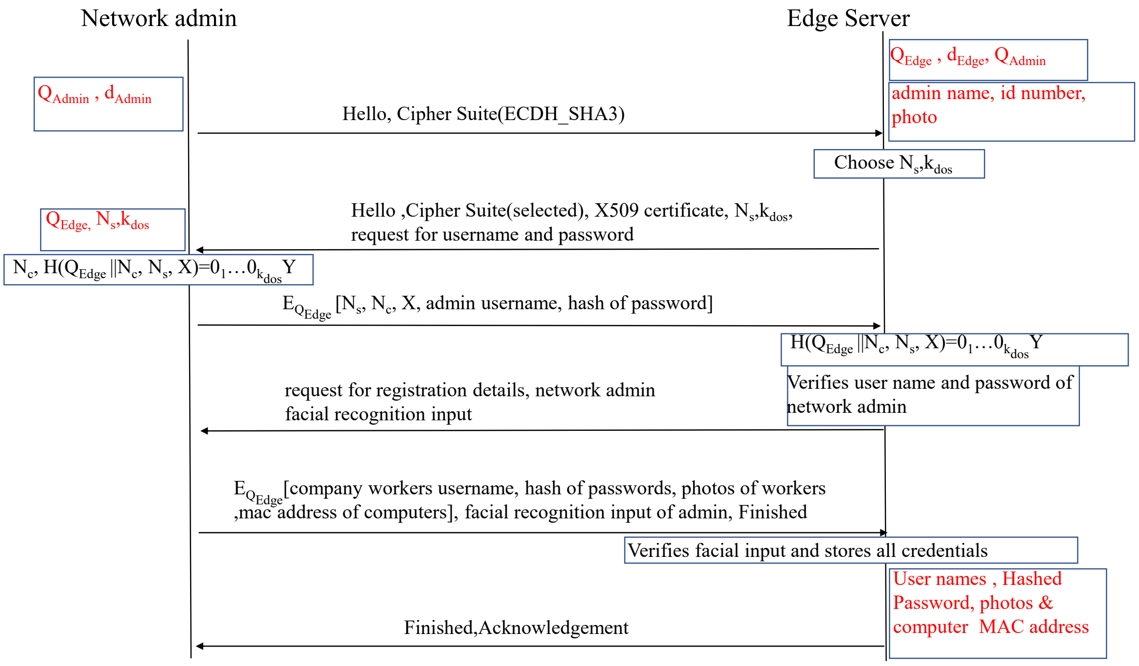

Network admin
Edge Server
Q_{Edge}, d_{Edge}, Q_{Admin}
admin name, id number, photo
Q_{Admin}, d_{Admin}
Hello, Cipher Suite(ECDH_SHA3)
Choose N_s,k_{dos}
Hello ,Cipher Suite(selected), X509 certificate, N_s,k_{dos}, request for username and password
Q_{Edge}, N_s,k_{dos}
N_c, $H(Q_{Edge} \| N_c, N_s, X)=0_1\ldots0_{k_{dos}}Y$
$E_{Q_{Edge}}$ [N_s, N_c, X, admin username, hash of password]
$H(Q_{Edge} \| N_c, N_s, X)=0_1\ldots0_{k_{dos}}Y$
Verifies user name and password of network admin
request for registration details, network admin facial recognition input
$E_{Q_{Edge}}$[company workers username, hash of passwords, photos of workers ,mac address of computers], facial recognition input of admin, Finished
Verifies facial input and stores all credentials
User names , Hashed Password, photos & computer MAC address
Finished,Acknowledgement

Figure 13. Message flow for the registration phase of company workers and devices.

The network admin uses his registered device, and then register the company workers, and their computer MAC addresses in the edge server. The network admin will create user name and passwords for the company workers and then feed those details together with workers photos to the edge server, and these photos will be used by edge server when the company worker gives his facial recognition input next time he logs in. Finally, he verifies this registration by giving his facial recognition input. The edge server verifies the facial input and then stores the registration details safely. Message flow ends with the finished and acknowledgment message from the edge server to the admin.

Step 4: Doctor and expert devices registration

The registered network admins responsible in the hospitals will use their log-in credentials from their registered devices and feed the MAC addresses of the devices about to be used by the doctor and experts. These registered devices will be provided by the network admins to the doctors and experts in the hospital, but experts might be moving from one or more hospitals, and even they may travel around the world, this will make the registration process complex, this is left for future works at the moment.

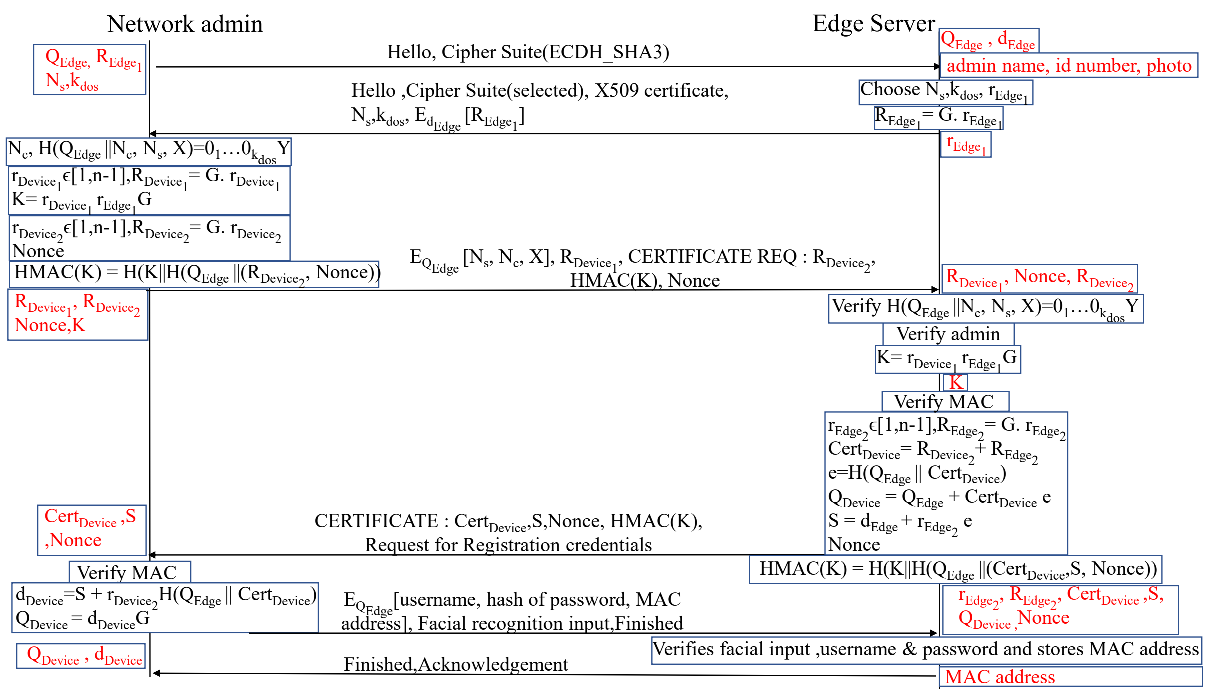

Network admin
Edge Server
Q_{Edge}, R_{Edge_1} N_s,k_{dos}
Hello, Cipher Suite(ECDH_SHA3)
Q_{Edge} , d_{Edge}
admin name, id number, photo
Hello ,Cipher Suite(selected), X509 certificate, N_s,k_{dos}, $E_{d_{Edge}}$ [R_{Edge_1}]
Choose N_s,k_{dos}, r_{Edge_1}
R_{Edge_1}= G. r_{Edge_1}
r_{Edge_1}
N_c, H(Q_{Edge} ||N_c, N_s, X)=0_1…$0_{k_{dos}}$ Y
r_{Device_1}ϵ[1,n-1],R_{Device_1}= G. r_{Device_1}
K= r_{Device_1} r_{Edge_1}G
r_{Device_2}ϵ[1,n-1],R_{Device_2}= G. r_{Device_2} Nonce
HMAC(K) = H(K||H(Q_{Edge} ||(R_{Device_2}, Nonce))
$E_{Q_{Edge}}$ [N_s, N_c, X], R_{Device_1}, CERTIFICATE REQ : R_{Device_2}, HMAC(K), Nonce
R_{Device_1}, Nonce, R_{Device_2}
R_{Device_1}, R_{Device_2} Nonce,K
Verify H(Q_{Edge} ||N_c, N_s, X)=0_1…$0_{k_{dos}}$ Y
Verify admin
K= r_{Device_1} r_{Edge_1}G
K
Verify MAC
r_{Edge_2}ϵ[1,n-1],R_{Edge_2}= G. r_{Edge_2}
$Cert_{Device}$= R_{Device_2}+ R_{Edge_2}
e=H(Q_{Edge} || $Cert_{Device}$)
Q_{Device} = Q_{Edge} + $Cert_{Device}$ e
S = d_{Edge} + r_{Edge_2} e
Nonce
$Cert_{Device}$,S ,Nonce
CERTIFICATE : $Cert_{Device}$,S,Nonce, HMAC(K), Request for Registration credentials
HMAC(K) = H(K||H(Q_{Edge} ||($Cert_{Device}$,S, Nonce))
Verify MAC
d_{Device}=S + r_{Device_2}H(Q_{Edge} || $Cert_{Device}$)
Q_{Device} = d_{Device}G
$E_{Q_{Edge}}$[username, hash of password, MAC address], Facial recognition input,Finished
r_{Edge_2}, R_{Edge_2}, $Cert_{Device}$,S, Q_{Device} ,Nonce
Q_{Device} , d_{Device}
Verifies facial input ,username & password and stores MAC address
Finished,Acknowledgement
MAC address

Figure 14. Message flow for the registration phase of expert/doctor devices.

Now it is assumed that the expert attends one particular hospital, and the network admin can do the registration of the expert and expert device, message flow in this step flows as per Figure 14. The message flow is the same as in registration of the network admin in step 2 until the device receives its public key and computes the private key.

Soon after storing the keys in the device, the network admin will use his user name, hashed password, and the device MAC address encrypted with the public key of the edge server for registration. The edge server verifies the user name and password, and then wait for the facial recognition input of the admin, once it is also verified it will store the MAC address, and consider the device as a registered device in the future log-in attempts. Finally, the message flow ends with the finished and acknowledgment from edge serve to network admin.

Step 5: Doctor and expert registration

After the registration of the devices of doctors and experts, network admin creates a user name and password for the doctor and expert and then registers that username, password, other details and photos of the doctor and expert in the edge server. Message flow for this step is as shown in Figure 15. Here the majority of the message flow is similar to step 3, but here the network admin uses his log-in credentials in the registered doctor, expert devices. Hence the MAC address of the device is verified together with network admin log-in credentials at the edge server end. Once edge server verifies all those details, it again verifies the facial input of the network admin and then stores the details of the doctor, expert, their user names, and corresponding hashed passwords. The message flow ends with the finished and acknowledgment message from the server.

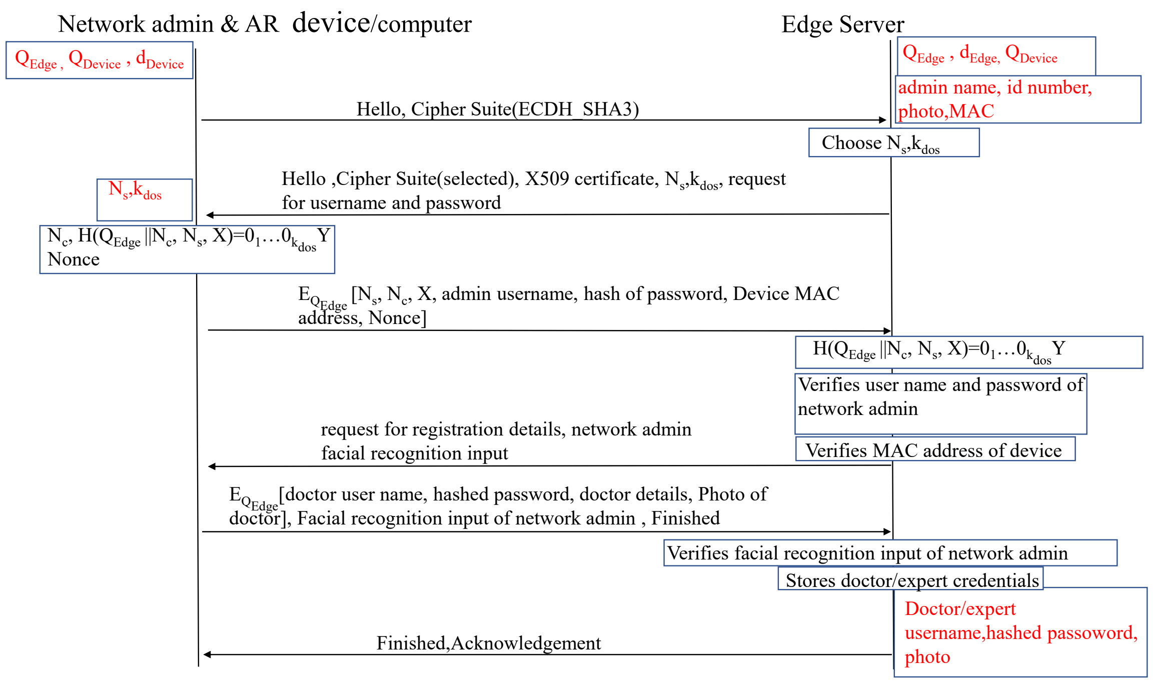

Network admin & AR device/computer
Edge Server
Q_{Edge}, Q_{Device}, d_{Device}
Q_{Edge}, d_{Edge}, Q_{Device}
admin name, id number, photo,MAC
Hello, Cipher Suite(ECDH_SHA3)
Choose N_s,k_{dos}
Hello ,Cipher Suite(selected), X509 certificate, N_s,k_{dos}, request for username and password
N_s,k_{dos}
N_c, $H(Q_{Edge} \| N_c, N_s, X)=0_1\ldots0_{k_{dos}}Y$ Nonce
$E_{Q_{Edge}}$ [N_s, N_c, X, admin username, hash of password, Device MAC address, Nonce]
$H(Q_{Edge} \| N_c, N_s, X)=0_1\ldots0_{k_{dos}}Y$
Verifies user name and password of network admin
request for registration details, network admin facial recognition input
Verifies MAC address of device
$E_{Q_{Edge}}$[doctor user name, hashed password, doctor details, Photo of doctor], Facial recognition input of network admin , Finished
Verifies facial recognition input of network admin
Stores doctor/expert credentials
Doctor/expert username,hashed passoword, photo
Finished,Acknowledgement

Figure 15. Message flow for the registration phase of expert/doctor.

Step 6: Patient registration

In the final step of the registration phase, the network admin registers the patient details when he comes to the hospital. He collects the patient details, his medical histories, Phone number, Photo, and his consent to store and use the data. After that, he creates a user name for the patient, and uses his registered device, and starts the registration process as shown in Figure 16.

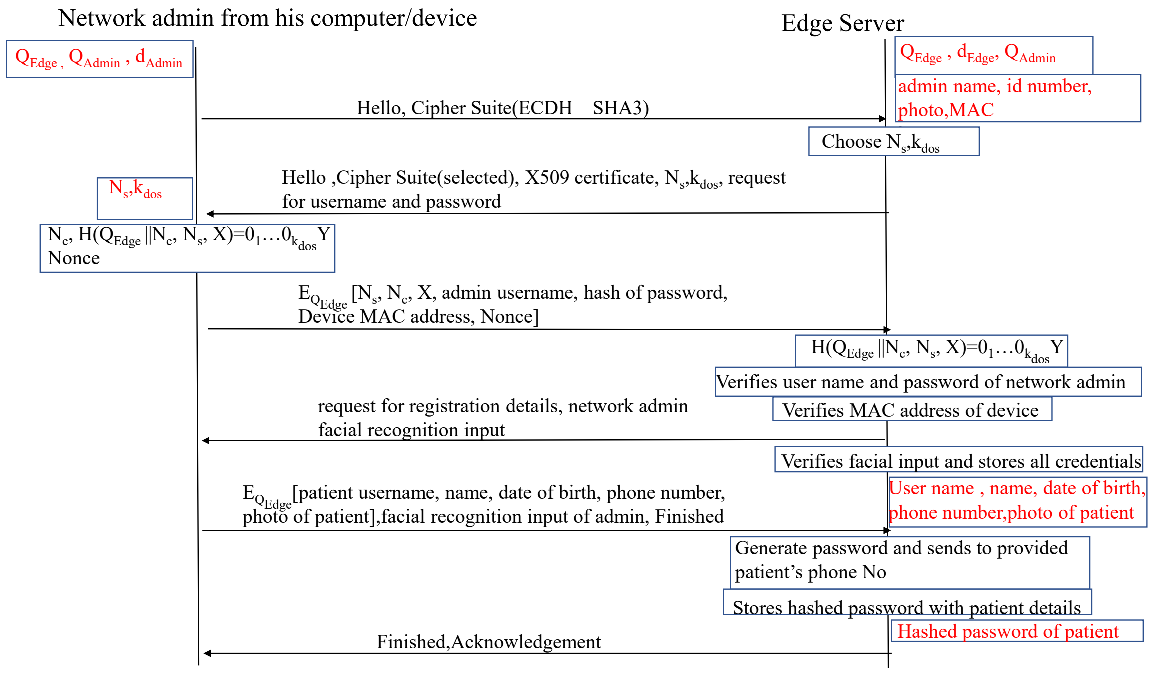

Network admin from his computer/device
Edge Server
Q_{Edge}, Q_{Admin}, d_{Admin}
Q_{Edge}, d_{Edge}, Q_{Admin}
admin name, id number, photo,MAC
Hello, Cipher Suite(ECDH SHA3)
Choose N_s,k_{dos}
Hello ,Cipher Suite(selected), X509 certificate, N_s,k_{dos}, request for username and password
N_s,k_{dos}
N_c, $H(Q_{Edge} \| N_c, N_s, X)=0_1\ldots0_{k_{dos}}Y$ Nonce
$E_{Q_{Edge}}$ [N_s, N_c, X, admin username, hash of password, Device MAC address, Nonce]
$H(Q_{Edge} \| N_c, N_s, X)=0_1\ldots0_{k_{dos}}Y$
Verifies user name and password of network admin
request for registration details, network admin facial recognition input
Verifies MAC address of device
Verifies facial input and stores all credentials
$E_{Q_{Edge}}$[patient username, name, date of birth, phone number, photo of patient],facial recognition input of admin, Finished
User name , name, date of birth, phone number,photo of patient
Generate password and sends to provided patient's phone No
Stores hashed password with patient details
Hashed password of patient
Finished,Acknowledgement

Figure 16. Message flow for the registration phase of patients.

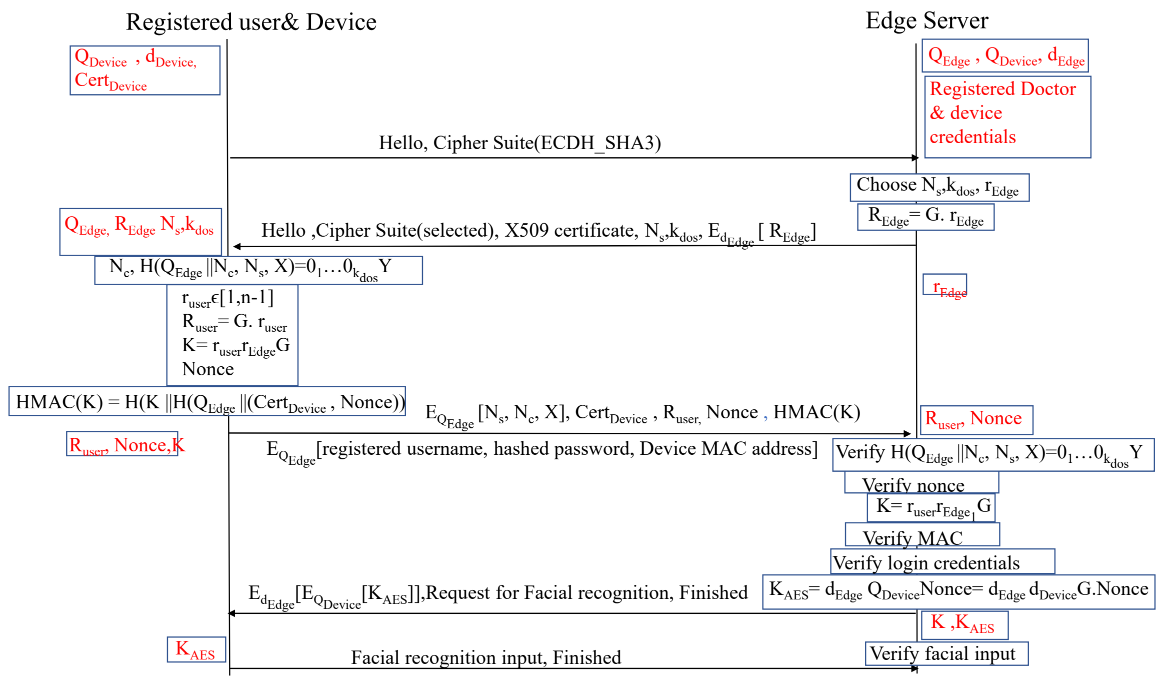
This process is also similar to step 5, but when the edge server request for the registration details, admin provides the user name of the patient and all the details he collected from the patient, together with admin's facial recognition input. Once the edge server verifies the admin facial recognition input, it stores all the patient-related information, and then sends a temporary passcode for patient's mobile number for one-time log-in. The patient can use this code to log-in the first time, and later, the password can be changed as he wishes. The message flow ends with the finished message from the edge server.

5.2 Authentication Phase

This phase explains the message flow when the registered users like doctors, experts, Patients try to communicate with the edge server and the message flow when doctor and expert initiate communication between them. The primary target of this phase is to establish a session key securely.

Phase 1: Registered user and device

In this phase, the already registered users who have their log-in credentials use their registered devices to initiate the communication with their edge server and establish a session key between the edge server and themselves. Since the message flow for the doctor, expert, network admin, patient in this phase are same, those entities have been commonly referred to as user as shown in Figure 17.



Registered user& Device
Edge Server
Q_Device , d_Device, Cert_Device
Q_Edge , Q_Device, d_Edge
Registered Doctor & device credentials
Hello, Cipher Suite(ECDH_SHA3)
Choose N_s,k_dos, r_Edge
R_Edge= G. r_Edge
Q_Edge, R_Edge N_s,k_dos
Hello ,Cipher Suite(selected), X509 certificate, N_s,k_dos, E_dEdge [R_Edge]
N_c, H(Q_Edge ||N_c, N_s, X)=0_1…0_kdos Y
r_user∈[1,n-1]
R_user= G. r_user
K= r_user r_Edge G
Nonce
r_Edge
HMAC(K) = H(K ||H(Q_Edge ||(Cert_Device , Nonce))
E_QEdge [N_s, N_c, X], Cert_Device , R_user, Nonce , HMAC(K)
R_user, Nonce
R_user, Nonce, K
E_QEdge[registered username, hashed password, Device MAC address]
Verify H(Q_Edge ||N_c, N_s, X)=0_1…0_kdos Y
Verify nonce
K= r_user r_Edge1 G
Verify MAC
Verify login credentials
E_dEdge[E_QDevice[K_AES]],Request for Facial recognition, Finished
K_AES= d_Edge Q_Device Nonce= d_Edge d_Device G.Nonce
K ,K_AES
K_AES
Facial recognition input, Finished
Verify facial input

Figure 17. Message flow for the authentication phase of user and devices with edge server.

The user initiates the communication with the edge server, by sending hello message and the cipher suite it supports. Then the edge server creates a Puzzle for DoS attack prevention just like in the registration phase, generates a random number r_{Edge1} and computes an elliptic curve point R_{Edge1} = G.r_{Edge1} to compute a key K, which is to be used for hash function in HMAC like in the registration phase. Then edge server responds with selected cipher suite, its X.509 certificate, puzzle, and R_{Edge1} signed by the edge to show that it is being originated from the edge server. User device solves the puzzle, generate a random number and an elliptic curve point r_{User1}, R_{User1}, then compute the key K as shown in the figure. With the newly generated key K, and the public key of the edge, it produces an HMAC value by hashing the $Cert_{Device}$. Then it encrypts the puzzle answer with the public key of the edge server, and then sends it together with the elliptic curve point R_{User1}, Nonce to show the freshness of the message, and HMAC to verify that the message is delivered as it is generated. Together with this message, the user sends his registered user name, hashed password, and device MAC address encrypted with the public key of the edge server. Upon receiving this message, the edge server first checks the puzzle answer, and once it is correct, it verifies the nonce and confirms the freshness of the message. After that, it computes the key with K = $R_{User1}.r_{Edge1}$, then it uses its public key and K to verify the HMAC value, and confirms the integrity of the message.

After this, it computes the symmetric AES session key using the private key of the K_{AES}=$d_{Edge}.Q_{Device}$.Nonce which is equivalent to K_{AES}=$d_{Edge}.d_{Device}$.G.Nonce here, the nonce is added to make the key only valid for the session, and it forbids the reuse of that key by any entity. After computing this key, the edge server encrypts it, using the public key of the user device, and sign it with its private key, and this ensures that only the user can access the key, and it also verifies to the user that the key is being by the edge server. Edge server also requests facial recognition input from the user, to ensure it has delivered the key correctly to the correct person. Once the user receives the key successfully, he will give his facial recognition input with a finished message. Edge server finally verifies the facial input from the user and concludes that the session key has been delivered to the user safely, and a safe session has been established with the user.

Phase 2: Registered doctor and expert

In this phase, the doctor and expert will authenticate each other and establish a session key between them. Since doctor and expert will be far away from each other, hence they will have different edge servers as their certificate authorities. In this case, they cant use their certificates provided by their edge servers for mutual authentication. Message flow for this phase is as shown in Figure 18. When the doctor wants to get the expert opinion, he initiates the communication with the expert by sending a hello message, and the cipher suite he supports, upon receiving this the expert sends back the hello message with selected cipher suite, and the certificate provided by his edge server, in the Figure 18 expert's edge server is named as edge server 2. The doctor then forwards this certificate to his edge server, which is named as edge server 1 in the Figure 18, and requests edge server 1 to provide a session key to communicate with the expert. Then edge server1 generates a random number r_{Edge1} and generates an elliptic curve point R_{Edge1} = G.r_{Edge1}. After that the edge server 1 initiates the communication with the edge server 2, by sending hello message, cipher suite it supports, X.509 certificate provided by an external certificate authority to prove his identity, the certificate of the expert device which was forwarded by doctor, R_{Edge1} signed with his private key, and a nonce to ensure the message freshness. The edge server 2 understands that a client of edge server 1 needs a session key to communicate with his client indicated in the $Cert_{Edevice}$, then it generates a random number and elliptic curve point r_{Edge2},R_{Edge2} = G.r_{Edge2}, and computes the session key using K_{AES}=$r_{Edge2}.R_{Edge1}$ which is equivalent to K_{AES}=r_{Edge2}.G.r_{Edge1}.

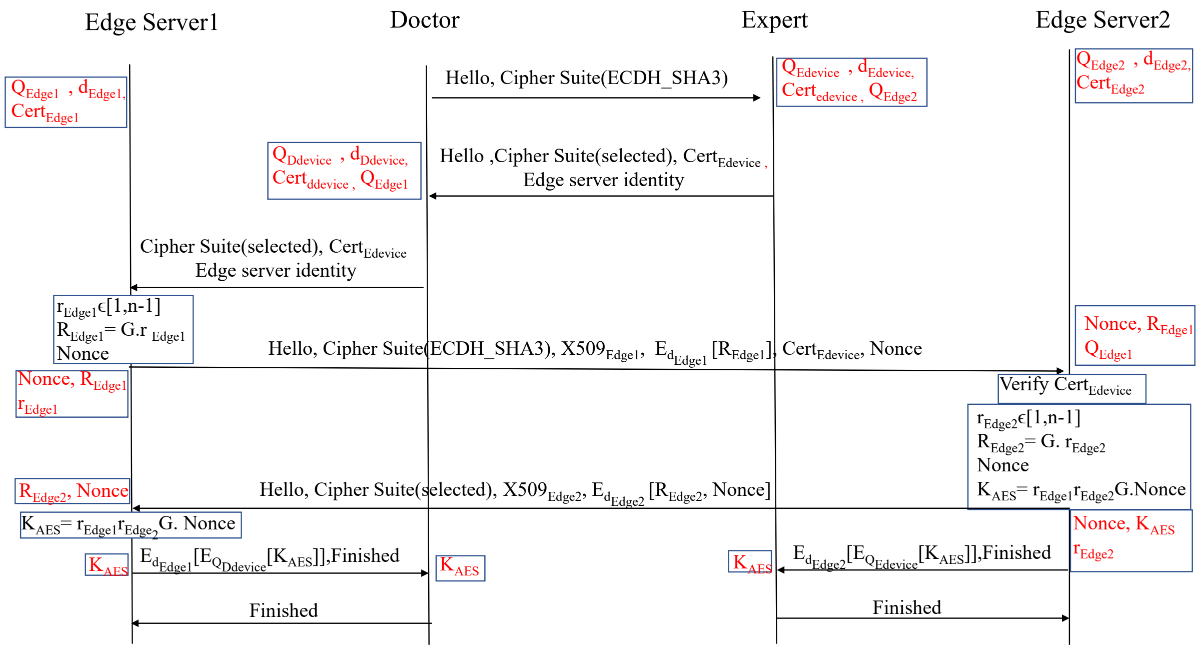

Edge Server1
Doctor
Expert
Edge Server2
QEdge1 , dEdge1, CertEdge1
QDdevice , dDdevice, Certddevice , QEdge1
QEdevice , dEdevice, Certedevice , QEdge2
QEdge2 , dEdge2, CertEdge2
Hello, Cipher Suite(ECDH_SHA3)
Hello ,Cipher Suite(selected), CertEdevice , Edge server identity
Cipher Suite(selected), CertEdevice Edge server identity
rEdge1∈[1,n-1] REdge1= G.r Edge1 Nonce
Hello, Cipher Suite(ECDH_SHA3), X509Edge1, EdEdge1 [REdge1], CertEdevice, Nonce
Nonce, REdge1 QEdge1
Nonce, REdge1 rEdge1
Verify CertEdevice
rEdge2∈[1,n-1] REdge2= G. rEdge2 Nonce KAES= rEdge1rEdge2G.Nonce
REdge2, Nonce
Hello, Cipher Suite(selected), X509Edge2, EdEdge2 [REdge2, Nonce]
KAES= rEdge1rEdge2G. Nonce
Nonce, KAES rEdge2
KAES
EdEdge1[EQDdevice[KAES]],Finished
KAES
KAES
EdEdge2[EQEdevice[KAES]],Finished
Finished
Finished

Figure 18. Message flow for the authentication phase between doctor and expert.

After computing the session key, edge server 2 signs the elliptic curve point R_{Edge2} concatenated with a Nonce, to ensure the authenticity and freshness of the elliptic curve point, and sends it back to edge server 1, together with usual handshake message as shown in Figure 18. Upon receiving this point, edge server 1 computes the required session key by K_{AES}=r_{Edge1}.R_{Edge2}, which is equivalent to K_{AES}=r_{Edge1}.G.r_{Edge2}. By this time, both servers have computed the session key required for their own clients. Then edge server1 encrypts the session key with the public key of the doctor and sign it with its private key, and sends it together with a finished message. Edge server 2, on the other hand, does the same but it encrypts the session key with the public key of the expert, and signs it with its private key and sends with a finished message. Upon receiving the keys, both doctor and expert send a finished message to their own servers and ends the communication.

6 FORMAL VERIFICATION

This chapter discusses the implementation and verification of the protocols proposed in chapter 5, using HLPSL language and AVISPA tool. Moreover, a detailed explanation of each scenario of the protocol is described in each subsection.

6.1 Simulation for Registration Phase

In this section, formal verification of the registration phase of each entities has been discussed. The registration scheme for each cases has been implemented in HLPSL language and verified by setting proper goals. Further, each subsections clearly explains the HLPSL implementation of a registration scenario in AVISPA tool, and discusses about the limitations and alternative approaches in the implementation of the actual scheme.

6.1.1 Registration of Network Admin

As we can see from Figure 12, the communication is taking place between the network admin and the edge server, therefore, there are two basic roles: network admin device and edge server, which are denoted by the *Admindevice* and *Edge*, respectively. For the *Admindevice*, the role specifications are shown in Figure 19, and role specifications for the *Edge* is shown in Figure 20, the *Admindevice* receives (RCV) a start signal and then changes its initial state from 0 to 1 and sends *(Hello')* using *SND*() operation to the *Edge*. Here the ' stands for a new value or new state.

Upon receiving this message with *RCV(Hello')*, *Edge* moves from state 0 to state 1, and then *Edge* generates new random value for *Xedge1* with the command *new()*, here *Xedge1* represents the r_{edge1} in Figure 12. *Edge* then computes the elliptic curve point *Redge1'* using *exp(G,Xedge1')* function, it also generates new values and assigns it to *Kdos'* and *Ns'*. After these computation, by using *SND* command it sends the message *(Hello'.{Kedge}_inv(Kexca).Ns'.Kdos'.{Redge1'}_inv(Kedge))* to agent *Admindevice*.

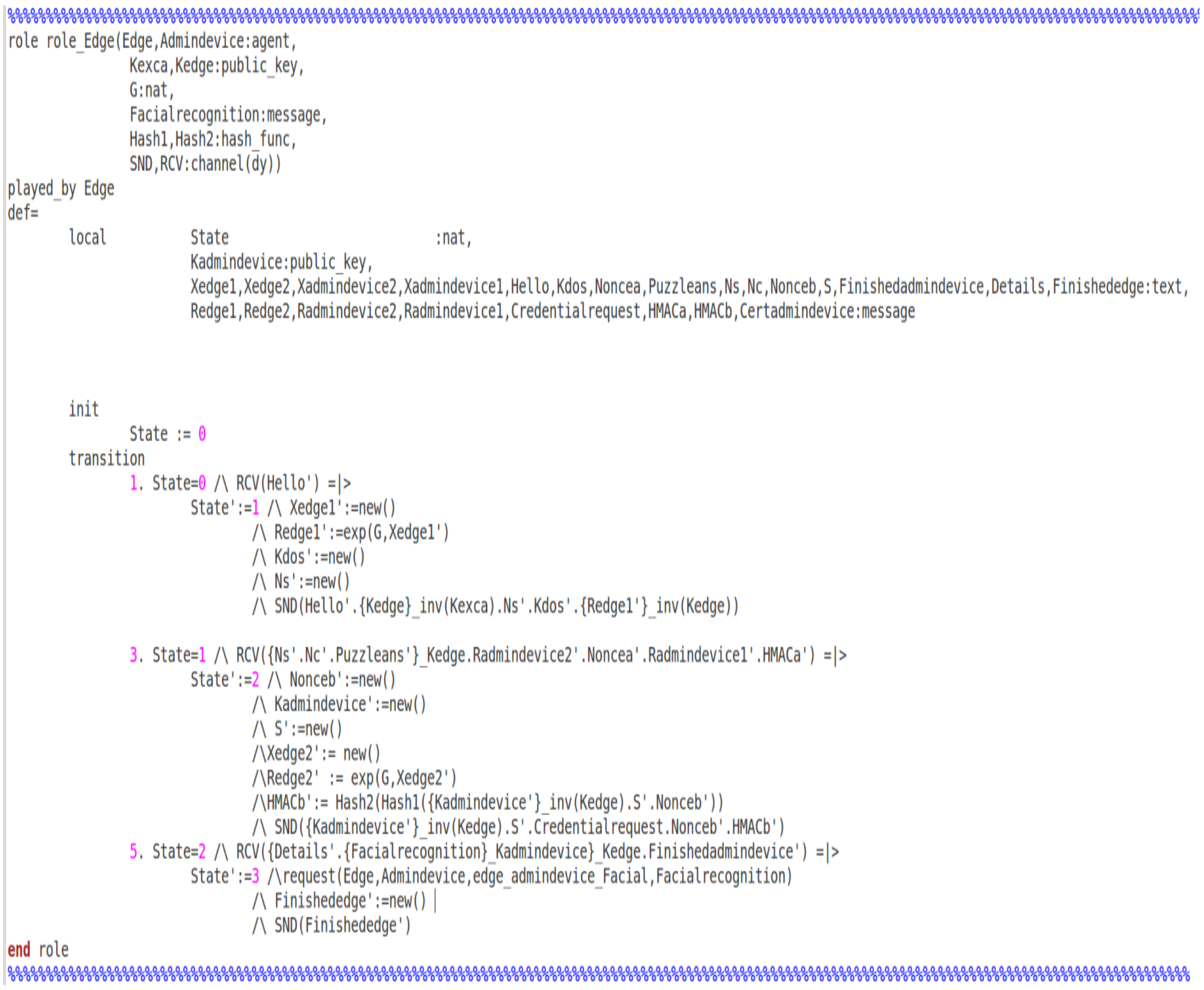
In HLPSL, *{Redge1'}_inv(Kedge)* stands for encrypting the *Redge1'* with the private key of the *Edge*. Likewise, *{Kedge}_inv(Kexca)* stands for the public key of *Edge* signed by the private key of the external certificate authority. Since there is no direct way to implement the X.509 certificate, *{Kedge}_inv(Kexca)* has been used to replicate X.509 certificate issued by external certificate authority. As a continuation of the message flow, *Admindevice* receives the message *(Hello'.{Kedge'}_inv(Kexca).Ns'.Kdos'.{Redge1'}_inv(Kedge'))* with RCV command. Here *Admindevice* receives the *Kedge* value for the first time therefore, it has been denoted as *Kedge'*. After receiving this *Admindevice* goes from state 1 to state 2, and generates values for *Xadmindevice1* and *Xadmindevice2* as shown in the Figure 19. The values *Xadmindevice1* and *Xadmindevice2* represents the r_{Admin1} and r_{Admin2} from Figure 12 respectively. AVISPA tool does not support any computations, hence we could not implement the process of *puzzleans*, instead, we have generated a fresh value from the *Admindevice*. Likewise, our actual HMAC function has two hash functions with two different keys but we could not implement that so we defined two different hash functions as *Hash1* and *Hash2*, and used it to calculate the HMAC value. After computing the values it sends them to *Edge* as shown in Figure 19. The *Edge* receives this message and moves from state 1 and state 2. Since the edge server already knows *Kedge*,

```
role role_Admindevice(Admindevice,Edge:agent,
                Kexca:public_key,
                G:nat,
                Facialrecognition:message,
                Hash1,Hash2:hash_func,
                SND,RCV:channel(dy))

played_by Admindevice
def=
        local
                State                                                                                           :nat,
                Kadmindevice,Kedge                                                                              :public_key,
                Hello,Xedge1,Xedge2,Xadmindevice2,Xadmindevice1,Kdos,Noncea,Nonceb,Puzzleans,Ns,Nc,S,Finishedadmindevice,Details,Finishededge   :text,
                Redge1,Redge2,Radmindevice2,Radmindevice1,Credentialrequest,HMACa,HMACb,Certadmindevice:message

        const   sec_details                                                                                     :protocol_id

        init    State := 0
        transition
                1. State=0 /\ RCV(start) =|>
                        State':=1 /\ Hello':=new()
                                /\ SND(Hello')
                2. State=1 /\ RCV(Hello'.{Kedge'}_inv(Kexca).Ns'.Kdos'.{Redge1'}_inv(Kedge')) =|>
                        State':=2 /\Noncea':=new()
                                /\ Xadmindevice2':=new()
                                /\ Radmindevice2' := exp(G,Xadmindevice2')
                                /\ Xadmindevice1':=new()
                                /\ Radmindevice1' := exp(G,Xadmindevice1')
                                /\ Puzzleans':=new()
                                /\ Nc':=new()
                                /\ HMACa' :=Hash2(Hash1(Radmindevice2'.Noncea'))
                                /\ SND({Ns'.Nc'.Puzzleans'}_Kedge'.Radmindevice2'.Noncea'.Radmindevice1'.HMACa')
                4. State=2 /\ RCV({Kadmindevice'}_inv(Kedge).S'.Credentialrequest.Nonceb'.HMACb') =|>
                                State':=3 /\ Finishedadmindevice':=new()
                                        /\ Details':=new()
                                        /\ SND({Details'.{Facialrecognition}_inv(Kadmindevice)}_Kedge.Finishedadmindevice')
                                        /\secret(Details',sec_details,{Edge,Admindevice})
                6. State=3 /\ RCV(Finishededge') =|> State':=4
end role
```

Figure 19. HLPSL specification for role admin.

the code changes from *Kedge'* to *Kedge*, all the other new values remained to be denoted with ' within the *RCV* command.

After receiving this, *Edge* now has to generate the certificate, but AVISPA tool does not support the arithmetic operations involved in the certificate computation. Hence, a fresh value is generated for *Kadmindevice'*, and signed it with the private key of *Edge*, and used it as a certificate issued by the edge to the *Admindevice*. The private key reconstruction value is generated just as a new value, without any actual arithmetic. The other functions are carried out as previously explained. Then, *Edge* sends the message back to *Admindevice* as shown in Figure 20. The *Admindevice* receives this message and generates a random value to represent a freshly generated *Details*, and sends it back together with facial recognition input. However, in HLPSL there are no direct ways to implement the facial recognition input, hence a facial recognition input is defined as a known secret message *Facialrecognition* between *Admindevice* and *Edge*. Upon receiving the last message, *Admindevice* signs the *Facialrecognition* message with its private key, concatenate with generated *Details'*, and encrypts it with the public key of the *Edge* and sends it to the *Edge*. This alternative approach for the facial recognition input can authenticate the legitimacy of the *Admindevice*. The Figure 21 shows the specification of the environment role which consists of global constants, composition of the sessions, and the knowledge of the intruder.

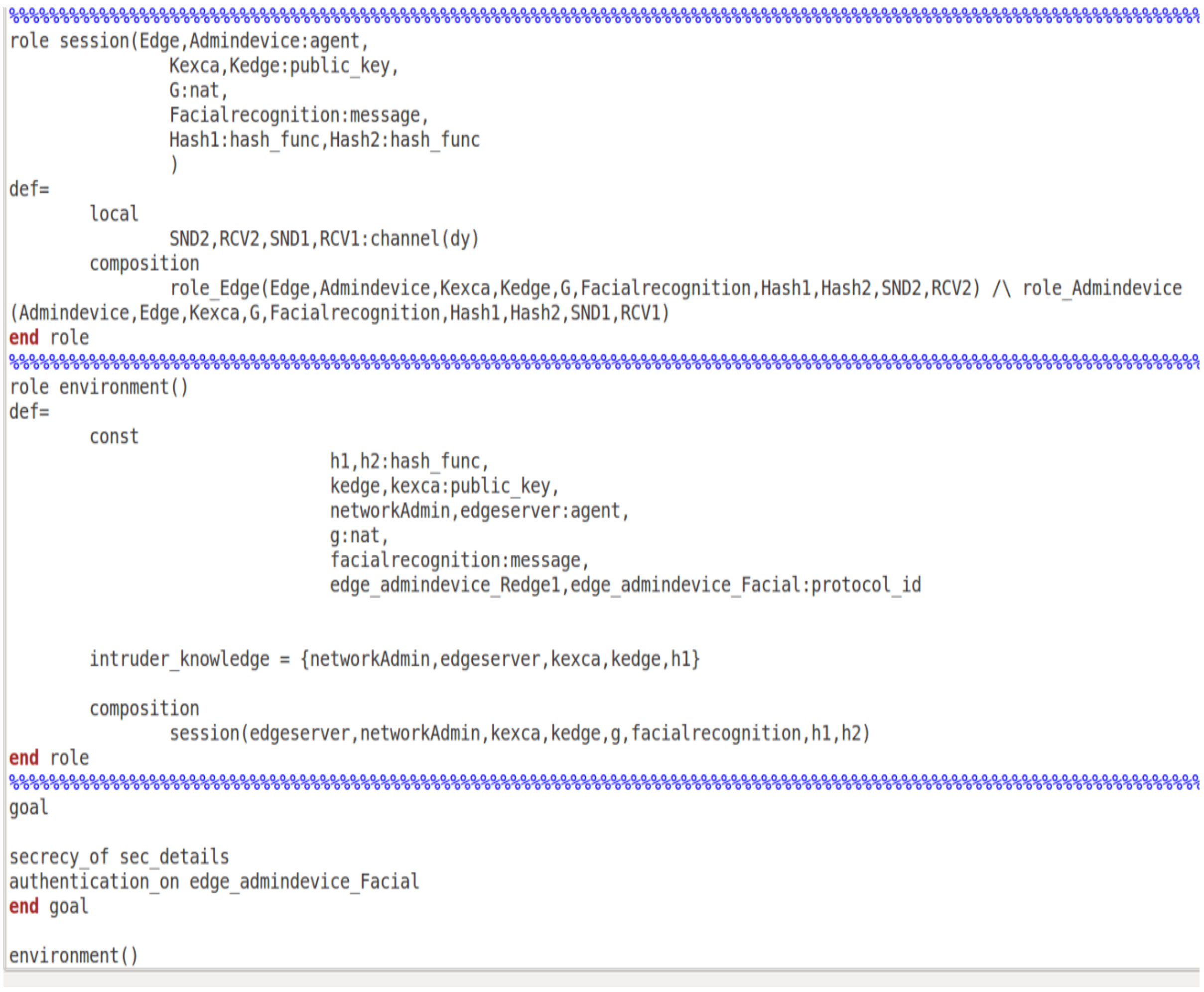


```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role role_Edge(Edge,Admindevice:agent,
                Kexca,Kedge:public_key,
                G:nat,
                Facialrecognition:message,
                Hash1,Hash2:hash_func,
                SND,RCV:channel(dy))
played_by Edge
def=
        local           State                           :nat,
                        Kadmindevice:public_key,
                        Xedge1,Xedge2,Xadmindevice2,Xadmindevice1,Hello,Kdos,Noncea,Puzzleans,Ns,Nc,Nonceb,S,Finishedadmindevice,Details,Finishededge:text,
                        Redge1,Redge2,Radmindevice2,Radmindevice1,Credentialrequest,HMACa,HMACb,Certadmindevice:message



        init
                State := 0
        transition
                1. State=0 /\ RCV(Hello') =|>
                        State':=1 /\ Xedge1':=new()
                                /\ Redge1':=exp(G,Xedge1')
                                /\ Kdos':=new()
                                /\ Ns':=new()
                                /\ SND(Hello'.{Kedge}_inv(Kexca).Ns'.Kdos'.{Redge1'}_inv(Kedge))

                3. State=1 /\ RCV({Ns'.Nc'.Puzzleans'}_Kedge.Radmindevice2'.Noncea'.Radmindevice1'.HMACa') =|>
                        State':=2 /\ Nonceb':=new()
                                /\ Kadmindevice':=new()
                                /\ S':=new()
                                /\Xedge2':= new()
                                /\Redge2' := exp(G,Xedge2')
                                /\HMACb':= Hash2(Hash1({Kadmindevice'}_inv(Kedge).S'.Nonceb'))
                                /\ SND({Kadmindevice'}_inv(Kedge).S'.Credentialrequest.Nonceb'.HMACb')
                5. State=2 /\ RCV({Details'.{Facialrecognition}_Kadmindevice}_Kedge.Finishedadmindevice') =|>
                        State':=3 /\request(Edge,Admindevice,edge_admindevice_Facial,Facialrecognition)
                                /\ Finishededge':=new() |
                                /\ SND(Finishededge')
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 20. HLPSL specification for role edge.

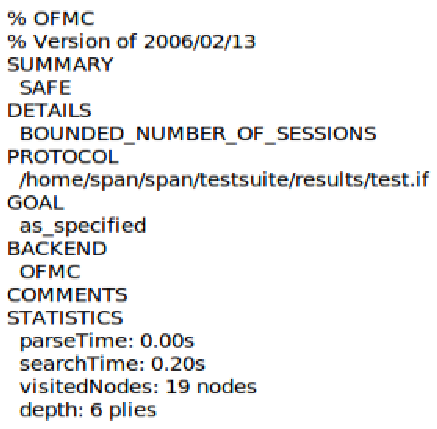
The *Hash2* is representing the hash function used in HMAC calculation with the secret key *K* as shown in Figure 12. Since the intruder is not aware of the *K*, the *Hash2* is not included in intruder knowledge. The session role shows the basic roles *Admindevice* and *Edge* are instanced with concrete arguments. Moreover, the Goal section in the same figure shows that secrecy goal and authentication goal are verified. Here the secrecy of *sec_details* represents that the *Details'* is kept as a secret between *Edge* and *Admindevice*. In the code it has been defined with the *secret(Details',sec_detail,{Edge, Admindevice})*. On the other hand, the *Authentication_on edge_admindevice_Facial* states that facial input is verified at the *Edge*. This has been defined with the code *request(Edge, Admindevice, edge_admindevice_Facial, Facialrecognition)*.

Results of the admin registration part of the proposed scheme using OFMC backend can be seen from the Figure 22. The results validates that the proposed scheme is safe.



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role session(Edge,Admindevice:agent,
                Kexca,Kedge:public_key,
                G:nat,
                Facialrecognition:message,
                Hash1:hash_func,Hash2:hash_func
                )
def=
        local
                SND2,RCV2,SND1,RCV1:channel(dy)
        composition
                role_Edge(Edge,Admindevice,Kexca,Kedge,G,Facialrecognition,Hash1,Hash2,SND2,RCV2) /\ role_Admindevice
(Admindevice,Edge,Kexca,G,Facialrecognition,Hash1,Hash2,SND1,RCV1)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role environment()
def=
        const
                                h1,h2:hash_func,
                                kedge,kexca:public_key,
                                networkAdmin,edgeserver:agent,
                                g:nat,
                                facialrecognition:message,
                                edge_admindevice_Redge1,edge_admindevice_Facial:protocol_id


        intruder_knowledge = {networkAdmin,edgeserver,kexca,kedge,h1}

        composition
                session(edgeserver,networkAdmin,kexca,kedge,g,facialrecognition,h1,h2)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
goal

secrecy_of sec_details
authentication_on edge_admindevice_Facial
end goal

environment()
```

Figure 21. HLPSL specification for role session and environment.



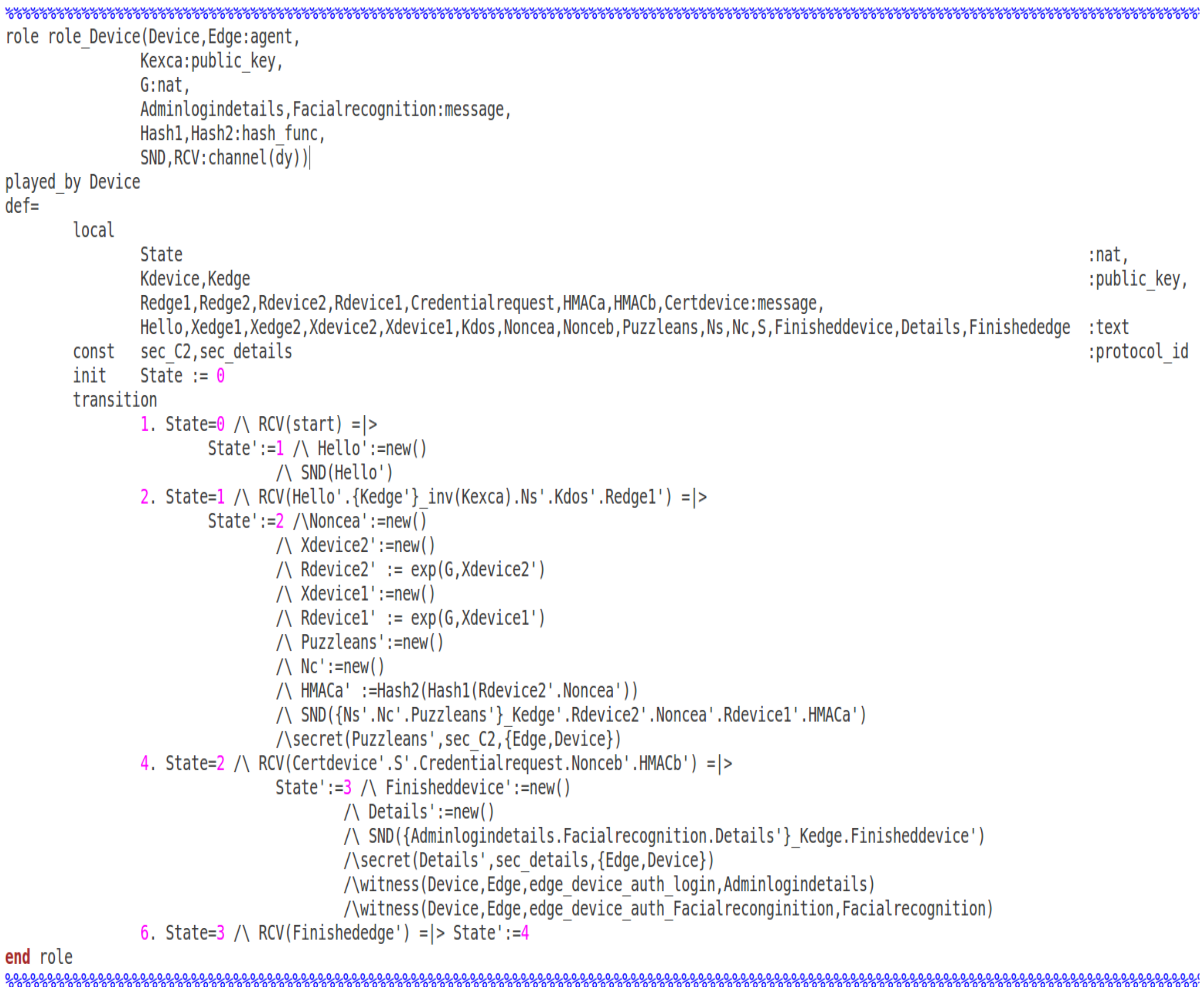
```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/test.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.20s
  visitedNodes: 19 nodes
  depth: 6 plies
```

Figure 22. Protocol verification results using OFMC backend.

6.1.2 Registration of Company Doctors/Experts Devices

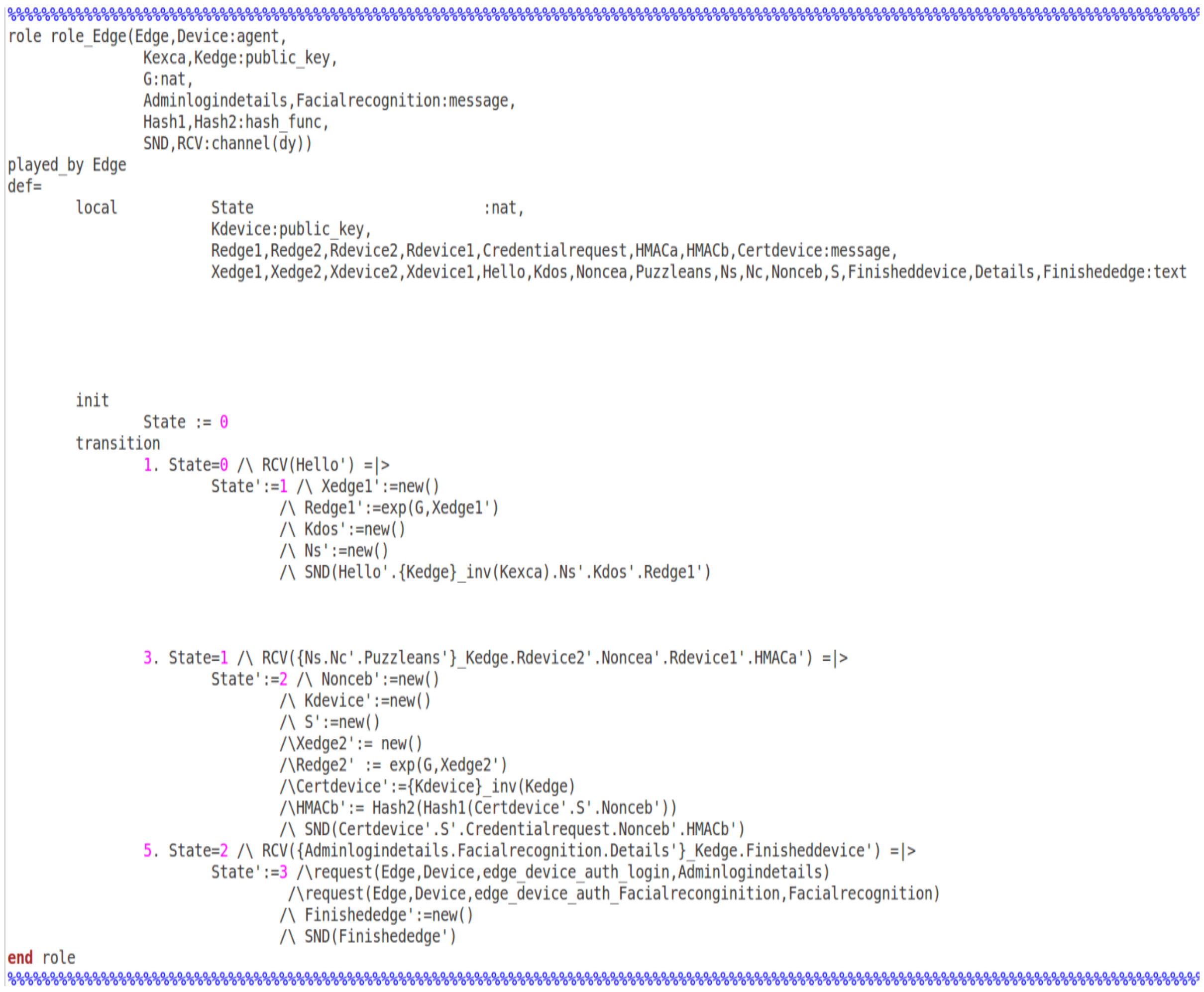
The implementation is based on the message flow shown in Figure 14. The implemented HLPSL script and specifications of the role device is shown in Figure 23, and role *Edge* is shown in Figure 24. The script for roles *session* and *environment* is shown in Figure 25. The HLPSL script works just like explained in the previous section but the notable point in this scenario is that the network admin is already registered when he is about to register the devices. Hence, *Adminlogindetails* have been declared as the known message for both agents *Device* and *Edge*, before starting the initiation of communication.

Here two secrecy goals *sec_C2*, *sec_details* and two authentication goals *edge_device_auth_login* and *edge_device_auth_Facialrecognition* have been defined. *sec_C2* represents the secrecy of *puzzleans'* between *Edge* and *Device*, *sec_details* represents the secrecy of doctor expert device details between the *Edge* and the *Device*/network admin, *edge_device_auth_login* states *Adminlogindetails* of the admin are verified at edge, *edge_device_auth_Facialrecognition* states that the *Facialrecognition* input of the admin is verified at the edge.



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role role_Device(Device,Edge:agent,
                Kexca:public_key,
                G:nat,
                Adminlogindetails,Facialrecognition:message,
                Hash1,Hash2:hash_func,
                SND,RCV:channel(dy))
played_by Device
def=
        local
                State                                                                                 :nat,
                Kdevice,Kedge                                                                         :public_key,
                Redge1,Redge2,Rdevice2,Rdevice1,Credentialrequest,HMACa,HMACb,Certdevice:message,
                Hello,Xedge1,Xedge2,Xdevice2,Xdevice1,Kdos,Noncea,Nonceb,Puzzleans,Ns,Nc,S,Finisheddevice,Details,Finishededge  :text
        const   sec_C2,sec_details                                                                    :protocol_id
        init    State := 0
        transition
                1. State=0 /\ RCV(start) =|>
                        State':=1 /\ Hello':=new()
                                /\ SND(Hello')
                2. State=1 /\ RCV(Hello'.{Kedge'}_inv(Kexca).Ns'.Kdos'.Redge1') =|>
                        State':=2 /\Noncea':=new()
                                /\ Xdevice2':=new()
                                /\ Rdevice2' := exp(G,Xdevice2')
                                /\ Xdevice1':=new()
                                /\ Rdevice1' := exp(G,Xdevice1')
                                /\ Puzzleans':=new()
                                /\ Nc':=new()
                                /\ HMACa' :=Hash2(Hash1(Rdevice2'.Noncea'))
                                /\ SND({Ns'.Nc'.Puzzleans'}_Kedge'.Rdevice2'.Noncea'.Rdevice1'.HMACa')
                                /\secret(Puzzleans',sec_C2,{Edge,Device})
                4. State=2 /\ RCV(Certdevice'.S'.Credentialrequest.Nonceb'.HMACb') =|>
                                State':=3 /\ Finisheddevice':=new()
                                        /\ Details':=new()
                                        /\ SND({Adminlogindetails.Facialrecognition.Details'}_Kedge.Finisheddevice')
                                        /\secret(Details',sec_details,{Edge,Device})
                                        /\witness(Device,Edge,edge_device_auth_login,Adminlogindetails)
                                        /\witness(Device,Edge,edge_device_auth_Facialrecongginition,Facialrecognition)
                6. State=3 /\ RCV(Finishededge') =|> State':=4
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

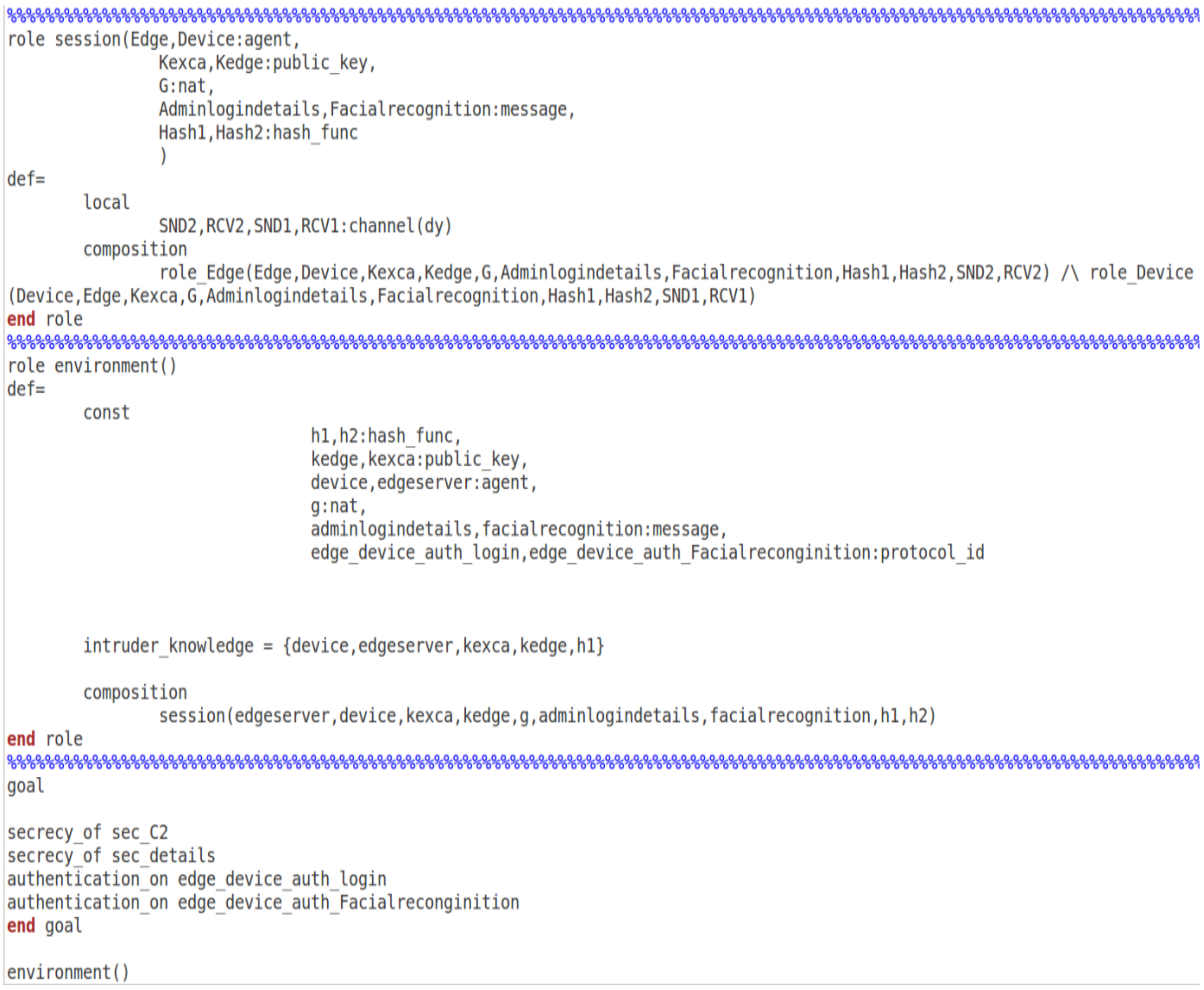
Figure 23. HLPSL specification for role device.



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role role_Edge(Edge,Device:agent,
                Kexca,Kedge:public_key,
                G:nat,
                Adminlogindetails,Facialrecognition:message,
                Hash1,Hash2:hash_func,
                SND,RCV:channel(dy))
played_by Edge
def=
        local           State                           :nat,
                        Kdevice:public_key,
                        Redge1,Redge2,Rdevice2,Rdevice1,Credentialrequest,HMACa,HMACb,Certdevice:message,
                        Xedge1,Xedge2,Xdevice2,Xdevice1,Hello,Kdos,Noncea,Puzzleans,Ns,Nc,Nonceb,S,Finisheddevice,Details,Finishededge:text




        init
                State := 0
        transition
                1. State=0 /\ RCV(Hello') =|>
                        State':=1 /\ Xedge1':=new()
                                /\ Redge1':=exp(G,Xedge1')
                                /\ Kdos':=new()
                                /\ Ns':=new()
                                /\ SND(Hello'.{Kedge}_inv(Kexca).Ns'.Kdos'.Redge1')



                3. State=1 /\ RCV({Ns.Nc'.Puzzleans'}_Kedge.Rdevice2'.Noncea'.Rdevice1'.HMACa') =|>
                        State':=2 /\ Nonceb':=new()
                                /\ Kdevice':=new()
                                /\ S':=new()
                                /\Xedge2':= new()
                                /\Redge2' := exp(G,Xedge2')
                                /\Certdevice':={Kdevice}_inv(Kedge)
                                /\HMACb':= Hash2(Hash1(Certdevice'.S'.Nonceb'))
                                /\ SND(Certdevice'.S'.Credentialrequest.Nonceb'.HMACb')
                5. State=2 /\ RCV({Adminlogindetails.Facialrecognition.Details'}_Kedge.Finisheddevice') =|>
                        State':=3 /\request(Edge,Device,edge_device_auth_login,Adminlogindetails)
                                 /\request(Edge,Device,edge_device_auth_Facialrecongininition,Facialrecognition)
                                /\ Finishededge':=new()
                                /\ SND(Finishededge')
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

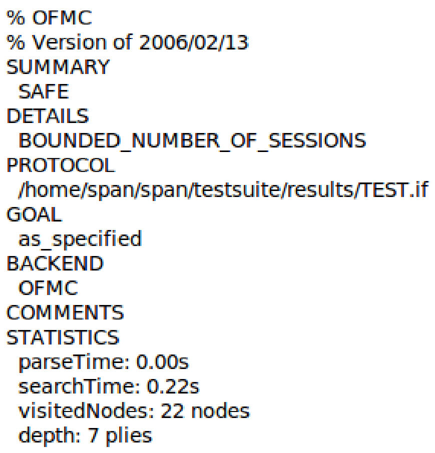
Figure 24. HLPSL specification for role edge.

Results of the registration of company doctors/experts devices part, of the proposed scheme using OFMC backend can be seen from Figure 26. The results validate that the proposed scheme is safe.



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role session(Edge,Device:agent,
                Kexca,Kedge:public_key,
                G:nat,
                Adminlogindetails,Facialrecognition:message,
                Hash1,Hash2:hash_func
                )
def=
        local
                SND2,RCV2,SND1,RCV1:channel(dy)
        composition
                role_Edge(Edge,Device,Kexca,Kedge,G,Adminlogindetails,Facialrecognition,Hash1,Hash2,SND2,RCV2) /\ role_Device
(Device,Edge,Kexca,G,Adminlogindetails,Facialrecognition,Hash1,Hash2,SND1,RCV1)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role environment()
def=
        const
                                h1,h2:hash_func,
                                kedge,kexca:public_key,
                                device,edgeserver:agent,
                                g:nat,
                                adminlogindetails,facialrecognition:message,
                                edge_device_auth_login,edge_device_auth_Facialreconginition:protocol_id



        intruder_knowledge = {device,edgeserver,kexca,kedge,h1}

        composition
                session(edgeserver,device,kexca,kedge,g,adminlogindetails,facialrecognition,h1,h2)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
goal

secrecy_of sec_C2
secrecy_of sec_details
authentication_on edge_device_auth_login
authentication_on edge_device_auth_Facialrecongintion
end goal

environment()
```

Figure 25. HLPSL specification for role session and environment.

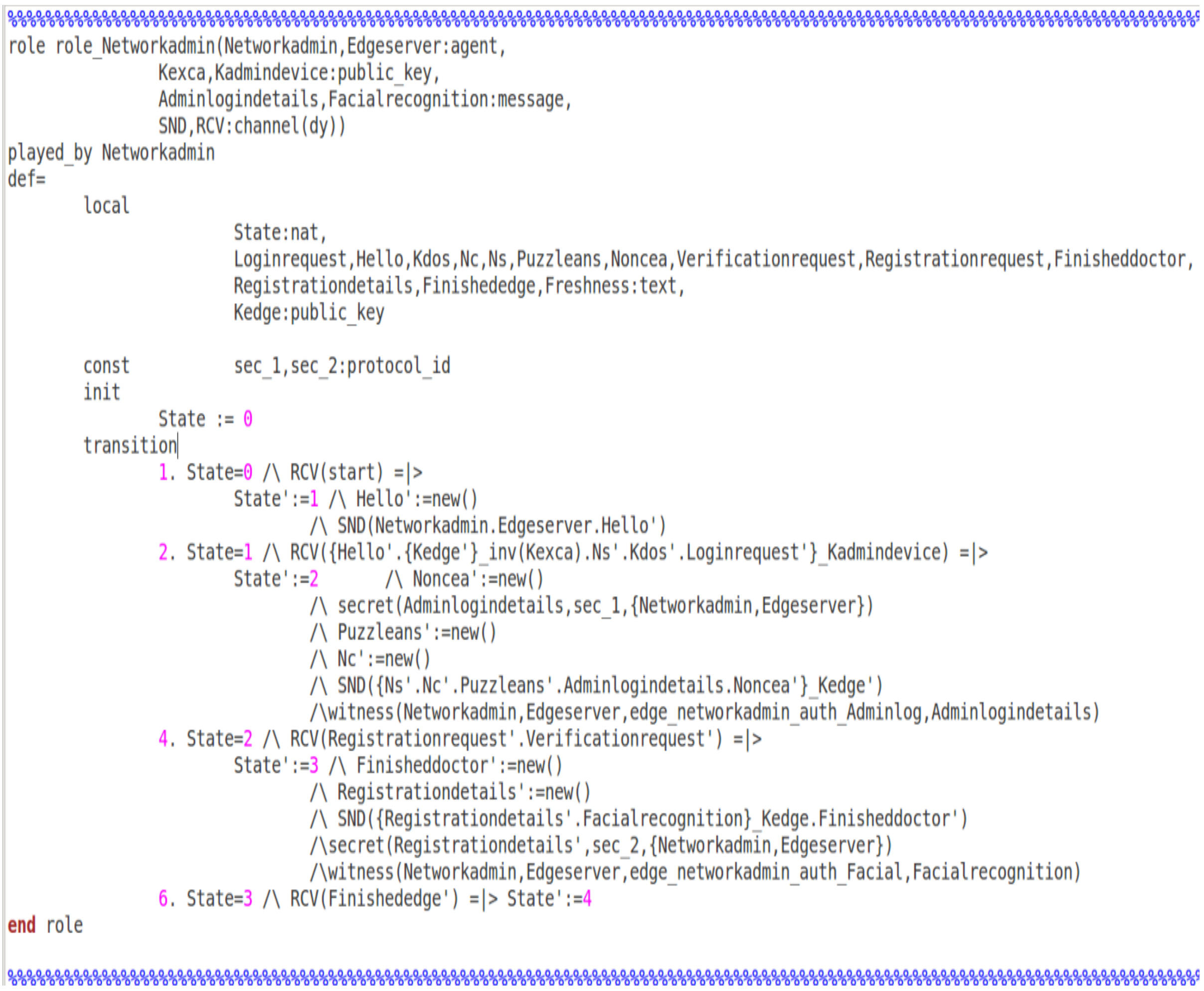


```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/TEST.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.22s
  visitedNodes: 22 nodes
  depth: 7 plies
```

Figure 26. Protocol verification results using OFMC backend.

6.1.3 Registration of Company Workers/Doctors/Experts

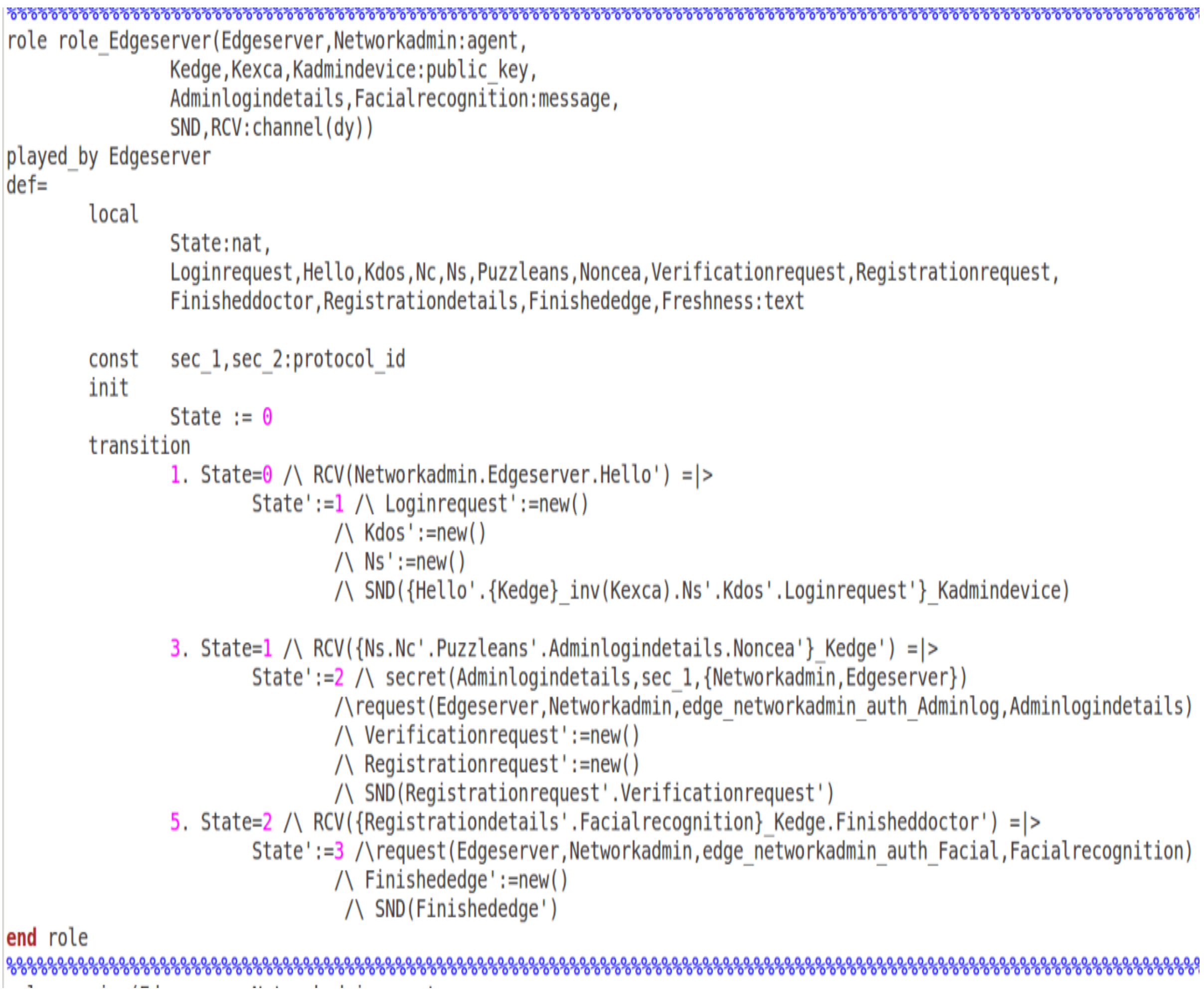
Explanation for the HLPSL implementation stays the same just like in the previous case, but the message flow in this phase follows the Figure 15, the procedure for the company workers follows the same flow as in the registration of the doctor and expert, so inclusion of the separate code for the registration of the company workers has been omitted. The implemented HLPSL script and specifications of the role *Networkadmin* is shown in Figure 27, role edge is shown in Figure 28, role of *session* and *environment* is shown in Figure 29.



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role role_Networkadmin(Networkadmin,Edgeserver:agent,
                Kexca,Kadmindevice:public_key,
                Adminlogindetails,Facialrecognition:message,
                SND,RCV:channel(dy))
played_by Networkadmin
def=
        local
                        State:nat,
                        Loginrequest,Hello,Kdos,Nc,Ns,Puzzleans,Noncea,Verificationrequest,Registrationrequest,Finisheddoctor,
                        Registrationdetails,Finishededge,Freshness:text,
                        Kedge:public_key

        const           sec_1,sec_2:protocol_id
        init
                State := 0
        transition
                1. State=0 /\ RCV(start) =|>
                        State':=1 /\ Hello':=new()
                                /\ SND(Networkadmin.Edgeserver.Hello')
                2. State=1 /\ RCV({Hello'.{Kedge'}_inv(Kexca).Ns'.Kdos'.Loginrequest'}_Kadmindevice) =|>
                        State':=2       /\ Noncea':=new()
                                /\ secret(Adminlogindetails,sec_1,{Networkadmin,Edgeserver})
                                /\ Puzzleans':=new()
                                /\ Nc':=new()
                                /\ SND({Ns'.Nc'.Puzzleans'.Adminlogindetails.Noncea'}_Kedge')
                                /\witness(Networkadmin,Edgeserver,edge_networkadmin_auth_Adminlog,Adminlogindetails)
                4. State=2 /\ RCV(Registrationrequest'.Verificationrequest') =|>
                        State':=3 /\ Finisheddoctor':=new()
                                /\ Registrationdetails':=new()
                                /\ SND({Registrationdetails'.Facialrecognition}_Kedge.Finisheddoctor')
                                /\secret(Registrationdetails',sec_2,{Networkadmin,Edgeserver})
                                /\witness(Networkadmin,Edgeserver,edge_networkadmin_auth_Facial,Facialrecognition)
                6. State=3 /\ RCV(Finishededge') =|> State':=4
end role

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 27. HLPSL specification for role network admin.

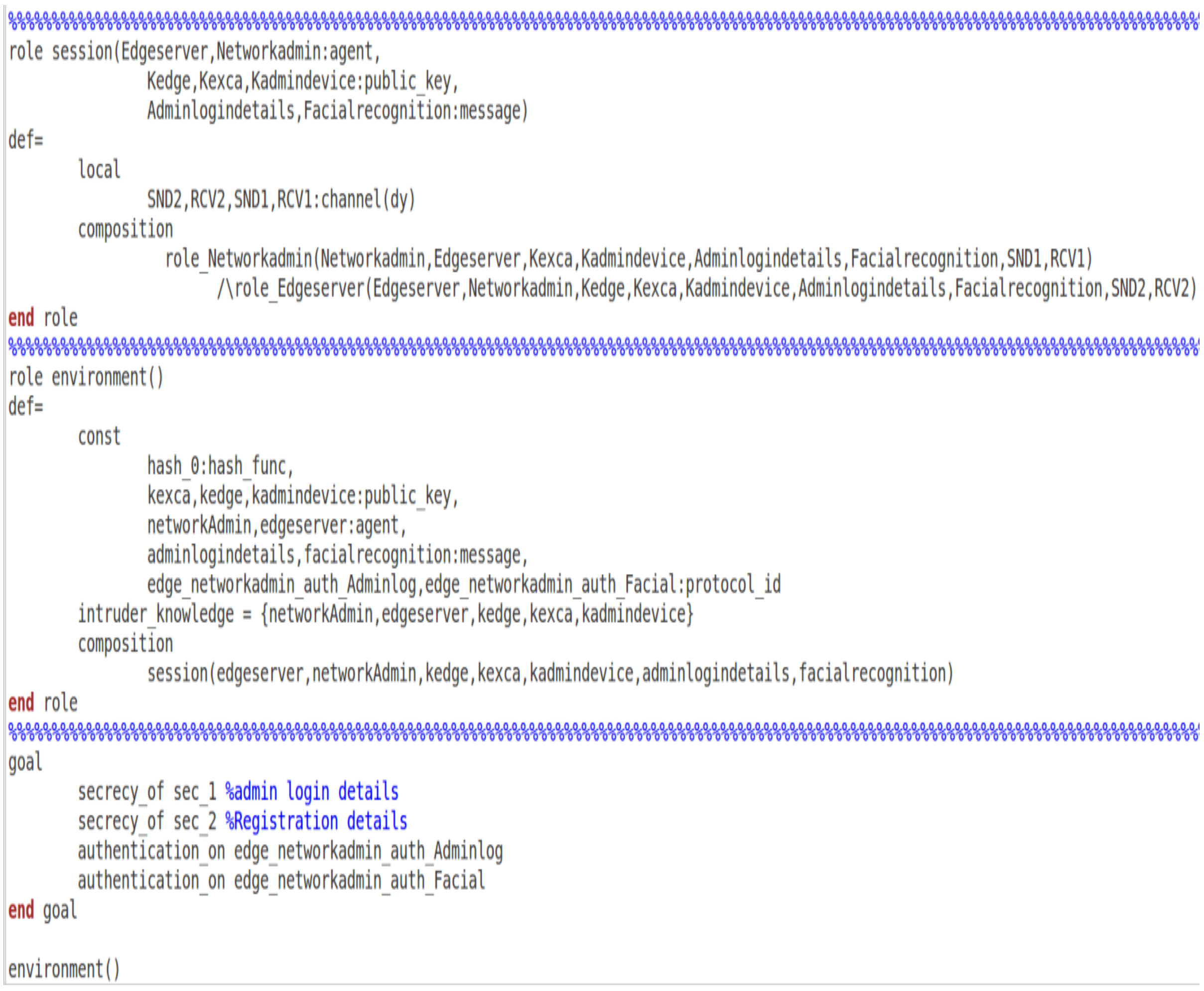
Here two secrecy goals denoted as *sec_1, sec_2* and two authentication goals denoted as *edge_networkadmin_auth_Adminlog, edge_networkadmin_auth_Facial* have been defined. *sec_1* represents the secrecy of *Adminlogindetails* between *Edge* and *Networkadmin*, likewise *sec_2* represents the secrecy of *Registrationdetails* of the doctor, expert and workers between the *Edge* and the *Networkadmin*, The *edge_networkadmin_auth_Adminlog* states log-in details of the admin are verified at *Edgeserver*, *edge_networkadmin_auth_Facial* states the facial recognition input of the admin is verified at the *Edgeserver*.



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role role_Edgeserver(Edgeserver,Networkadmin:agent,
                Kedge,Kexca,Kadmindevice:public_key,
                Adminlogindetails,Facialrecognition:message,
                SND,RCV:channel(dy))
played_by Edgeserver
def=
        local
                State:nat,
                Loginrequest,Hello,Kdos,Nc,Ns,Puzzleans,Noncea,Verificationrequest,Registrationrequest,
                Finisheddoctor,Registrationdetails,Finishededge,Freshness:text

        const   sec_1,sec_2:protocol_id
        init
                State := 0
        transition
                1. State=0 /\ RCV(Networkadmin.Edgeserver.Hello') =|>
                        State':=1 /\ Loginrequest':=new()
                                /\ Kdos':=new()
                                /\ Ns':=new()
                                /\ SND({Hello'.{Kedge}_inv(Kexca).Ns'.Kdos'.Loginrequest'}_Kadmindevice)

                3. State=1 /\ RCV({Ns.Nc'.Puzzleans'.Adminlogindetails.Noncea'}_Kedge') =|>
                        State':=2 /\ secret(Adminlogindetails,sec_1,{Networkadmin,Edgeserver})
                                /\request(Edgeserver,Networkadmin,edge_networkadmin_auth_Adminlog,Adminlogindetails)
                                /\ Verificationrequest':=new()
                                /\ Registrationrequest':=new()
                                /\ SND(Registrationrequest'.Verificationrequest')
                5. State=2 /\ RCV({Registrationdetails'.Facialrecognition}_Kedge.Finisheddoctor') =|>
                        State':=3 /\request(Edgeserver,Networkadmin,edge_networkadmin_auth_Facial,Facialrecognition)
                                /\ Finishededge':=new()
                                 /\ SND(Finishededge')
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

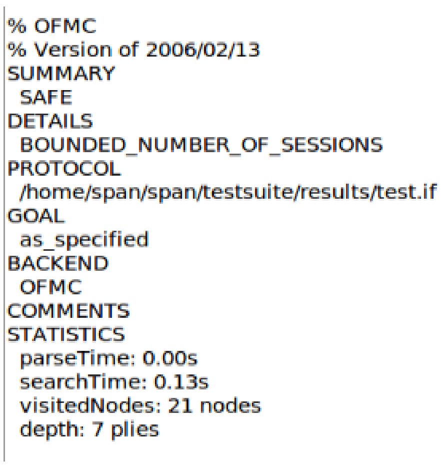
Figure 28. HLPSL specification for role edge server.

Results of the registration of company workers/doctors/experts part of the proposed scheme using OFMC backend can be seen from Figure 30. The result validates that the proposed scheme is safe.



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role session(Edgeserver,Networkadmin:agent,
             Kedge,Kexca,Kadmindevice:public_key,
             Adminlogindetails,Facialrecognition:message)
def=
       local
             SND2,RCV2,SND1,RCV1:channel(dy)
       composition
               role_Networkadmin(Networkadmin,Edgeserver,Kexca,Kadmindevice,Adminlogindetails,Facialrecognition,SND1,RCV1)
                    /\role_Edgeserver(Edgeserver,Networkadmin,Kedge,Kexca,Kadmindevice,Adminlogindetails,Facialrecognition,SND2,RCV2)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role environment()
def=
       const
             hash_0:hash_func,
             kexca,kedge,kadmindevice:public_key,
             networkAdmin,edgeserver:agent,
             adminlogindetails,facialrecognition:message,
             edge_networkadmin_auth_Adminlog,edge_networkadmin_auth_Facial:protocol_id
       intruder_knowledge = {networkAdmin,edgeserver,kedge,kexca,kadmindevice}
       composition
             session(edgeserver,networkAdmin,kedge,kexca,kadmindevice,adminlogindetails,facialrecognition)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
goal
       secrecy_of sec_1 %admin login details
       secrecy_of sec_2 %Registration details
       authentication_on edge_networkadmin_auth_Adminlog
       authentication_on edge_networkadmin_auth_Facial
end goal

environment()
```

Figure 29. HLPSL specification for role session and environment.

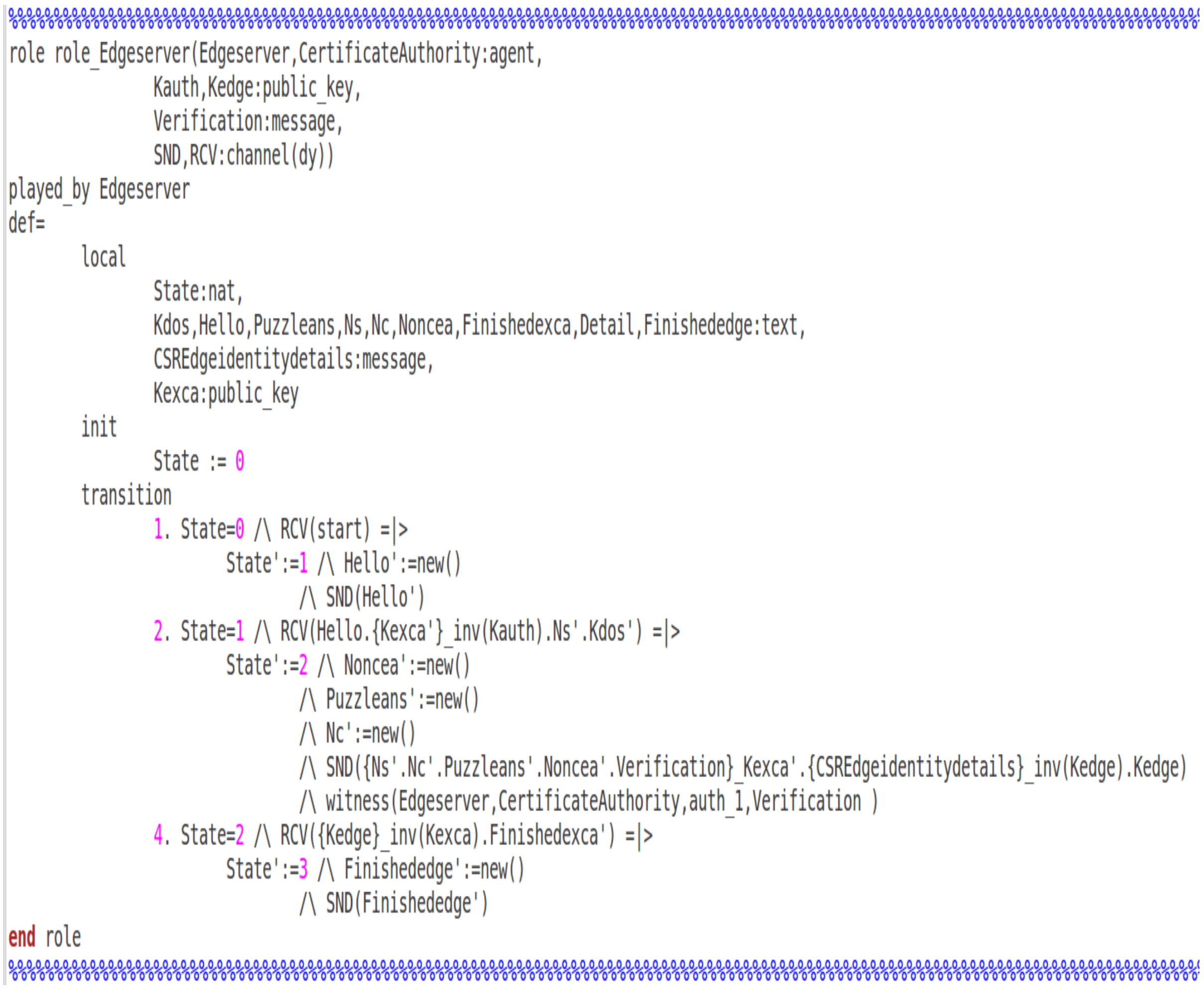


```
% OFMC
% Version of 2006/02/13
SUMMARY
 SAFE
DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
 /home/span/span/testsuite/results/test.if
GOAL
 as_specified
BACKEND
 OFMC
COMMENTS
STATISTICS
 parseTime: 0.00s
 searchTime: 0.13s
 visitedNodes: 21 nodes
 depth: 7 plies
```

Figure 30. Protocol verification results using OFMC backend.

6.1.4 Registration of Edge Server

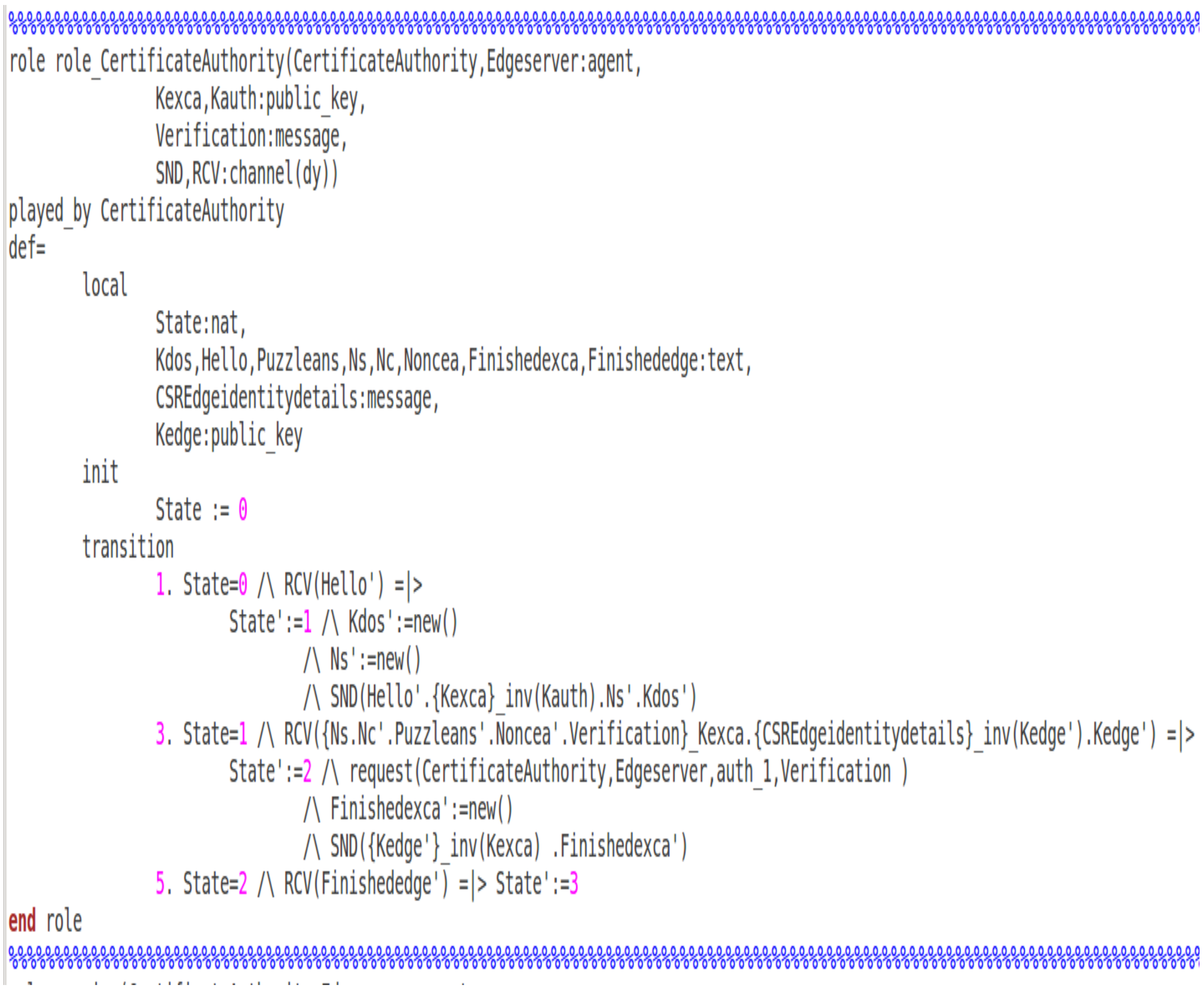
The implementation in this scenario is based on the message flow shown in Figure 11, the implemented HLPSL script and specifications of the role *Edgeserver* is shown in Figure 31, role of *CertificateAuthority* which is the external certificate authority for the edge server is shown in Figure 32, role of *session* and *environment* is shown in Figure 33.



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role role_Edgeserver(Edgeserver,CertificateAuthority:agent,
                Kauth,Kedge:public_key,
                Verification:message,
                SND,RCV:channel(dy))
played_by Edgeserver
def=
        local
                State:nat,
                Kdos,Hello,Puzzleans,Ns,Nc,Noncea,Finishedexca,Detail,Finishededge:text,
                CSREdgeidentitydetails:message,
                Kexca:public_key
        init
                State := 0
        transition
                1. State=0 /\ RCV(start) =|>
                        State':=1 /\ Hello':=new()
                                /\ SND(Hello')
                2. State=1 /\ RCV(Hello.{Kexca'}_inv(Kauth).Ns'.Kdos') =|>
                        State':=2 /\ Noncea':=new()
                                /\ Puzzleans':=new()
                                /\ Nc':=new()
                                /\ SND({Ns'.Nc'.Puzzleans'.Noncea'.Verification}_Kexca'.{CSREdgeidentitydetails}_inv(Kedge).Kedge)
                                /\ witness(Edgeserver,CertificateAuthority,auth_1,Verification )
                4. State=2 /\ RCV({Kedge}_inv(Kexca).Finishedexca') =|>
                        State':=3 /\ Finishededge':=new()
                                /\ SND(Finishededge')
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 31. HLPSL specification for role edgeserver.

In the message flow, the edge server sends a *Hello'* message to the external certificate authority and then the external certificate authority sends back puzzle for DoS attack prevention together with *{Kexca'}_inv(Kauth)*, which represents the X.509 certificate of the external certificate authority. Here the *Kauth* is the public key corresponding to the authority which gave the certificate to the external certificate authority. Upon receiving this message, the edge server generates the public key and private key using RSA key generation algorithm, but in HLPSL, it was not possible to implement that arithmetic, hence as an alternative it is declared that the edge server already knows its public and private key. Then the *Edgeserver* sends *{Ns'.Nc'.Puzzleans'.Noncea'.Verification}_Kexca'* concatenated with the *Kedge* and *{CSREdgeidentitydetails}_inv(Kedge)*. In a normal scenario, once the CSR request is received, the external certificate authority verifies the details of the edge server and authenticity of the details provided by other means, but it



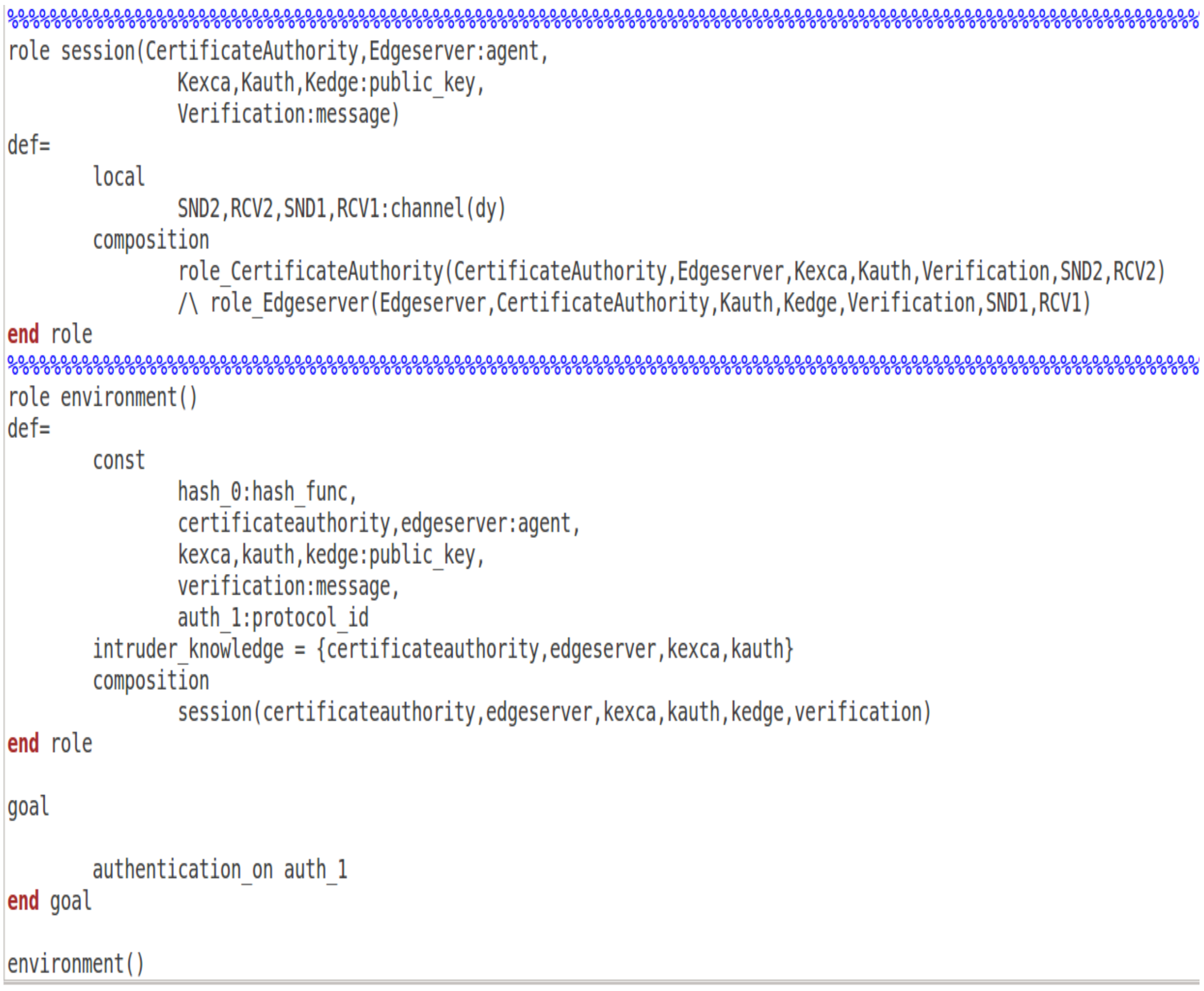
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role role_CertificateAuthority(CertificateAuthority,Edgeserver:agent,
                Kexca,Kauth:public_key,
                Verification:message,
                SND,RCV:channel(dy))
played_by CertificateAuthority
def=
        local
                State:nat,
                Kdos,Hello,Puzzleans,Ns,Nc,Noncea,Finishedexca,Finishededge:text,
                CSREdgeidentitydetails:message,
                Kedge:public_key
        init
                State := 0
        transition
                1. State=0 /\ RCV(Hello') =|>
                        State':=1 /\ Kdos':=new()
                                /\ Ns':=new()
                                /\ SND(Hello'.{Kexca}_inv(Kauth).Ns'.Kdos')
                3. State=1 /\ RCV({Ns.Nc'.Puzzleans'.Noncea'.Verification}_Kexca.{CSREdgeidentitydetails}_inv(Kedge').Kedge') =|>
                        State':=2 /\ request(CertificateAuthority,Edgeserver,auth_1,Verification )
                                /\ Finishedexca':=new()
                                /\ SND({Kedge'}_inv(Kexca) .Finishedexca')
                5. State=2 /\ RCV(Finishededge') =|> State':=3
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 32. HLPSL specification for role certificate authority.

was not possible to simulate this scenario in the HLPSL language, hence a secret message named *Verification* has been introduced between edge server and the external certificate authority and verification is simulated by exchanging *Verification* message encrypted with the public key of the external certificate authority.

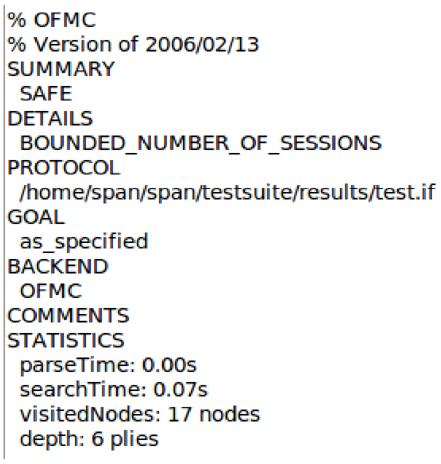
The security goal for this scenario has been defined as the *authentication_on auth_1*, which states the *Verification* message is verified at the external certificate authority.

The simulation result using OFMC backend, for this part of the proposed scheme using can be seen from Figure 34. The result validates that the proposed scheme is safe.



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role session(CertificateAuthority,Edgeserver:agent,
            Kexca,Kauth,Kedge:public_key,
            Verification:message)
def=
      local
            SND2,RCV2,SND1,RCV1:channel(dy)
      composition
            role_CertificateAuthority(CertificateAuthority,Edgeserver,Kexca,Kauth,Verification,SND2,RCV2)
            /\ role_Edgeserver(Edgeserver,CertificateAuthority,Kauth,Kedge,Verification,SND1,RCV1)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role environment()
def=
      const
            hash_0:hash_func,
            certificateauthority,edgeserver:agent,
            kexca,kauth,kedge:public_key,
            verification:message,
            auth_1:protocol_id
      intruder_knowledge = {certificateauthority,edgeserver,kexca,kauth}
      composition
            session(certificateauthority,edgeserver,kexca,kauth,kedge,verification)
end role

goal

      authentication_on auth_1
end goal

environment()
```

Figure 33. HLPSL specification for role session and environment.



```
% OFMC
% Version of 2006/02/13
SUMMARY
 SAFE
DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
 /home/span/span/testsuite/results/test.if
GOAL
 as_specified
BACKEND
 OFMC
COMMENTS
STATISTICS
 parseTime: 0.00s
 searchTime: 0.07s
 visitedNodes: 17 nodes
 depth: 6 plies
```

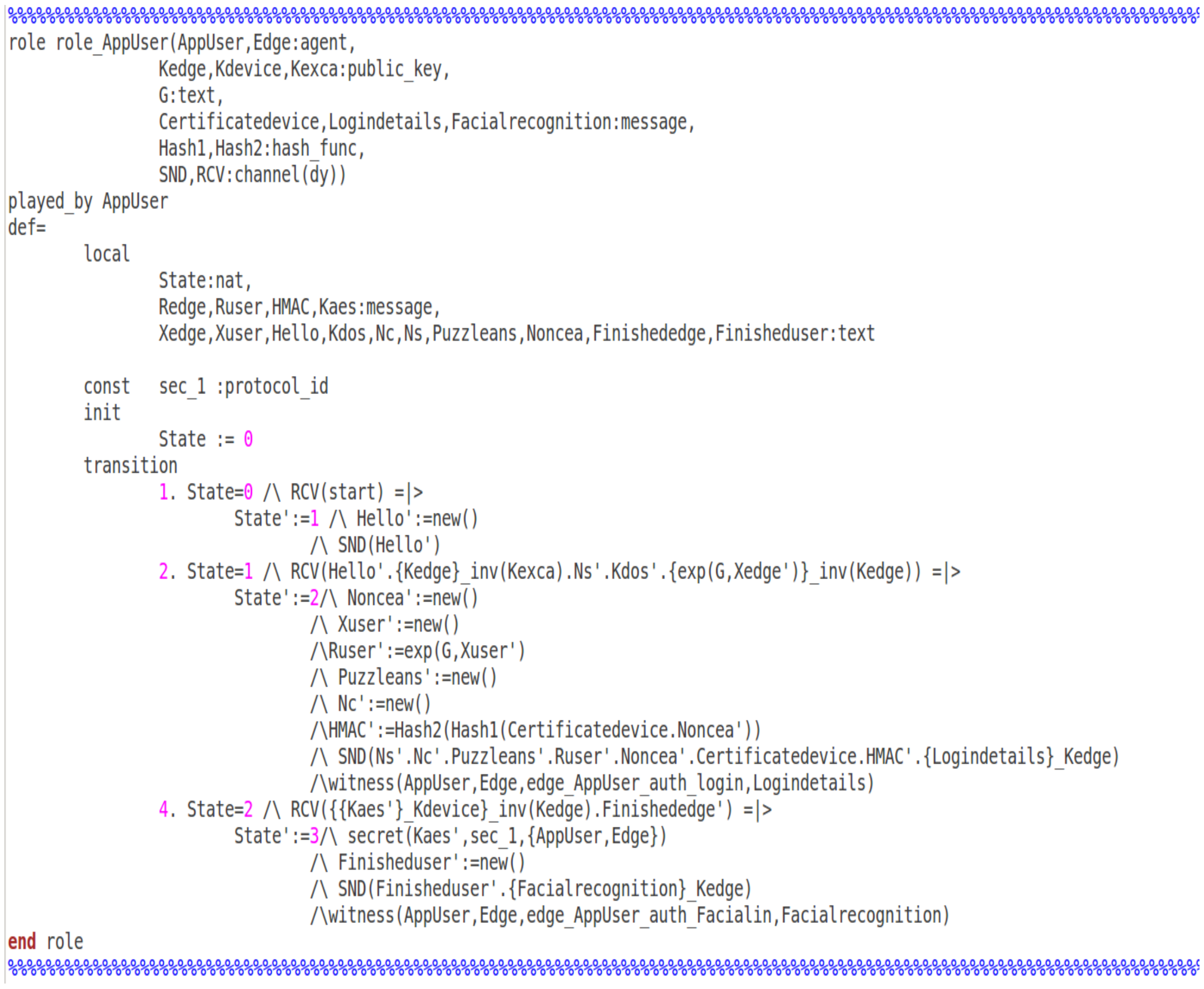
Figure 34. Protocol verification results using OFMC backend.

6.2 Authentication Phase

The authentication phase has two main scenarios, first scenario is the authentication phase between the *Edge* and *Appuser*. Since the registered users like network admin, doctor, expert, and company workers, follow the similar authentication message flow, all those entities have been commonly referred to as *Appuser* in the simulation. The second scenario of the authentication phase is the authentication between *Doctor* and *Expert*. These entities will mutually authenticate each other in the process of establishing a secure connection with secret session key this process for each scenario is explained in the upcoming subsections.

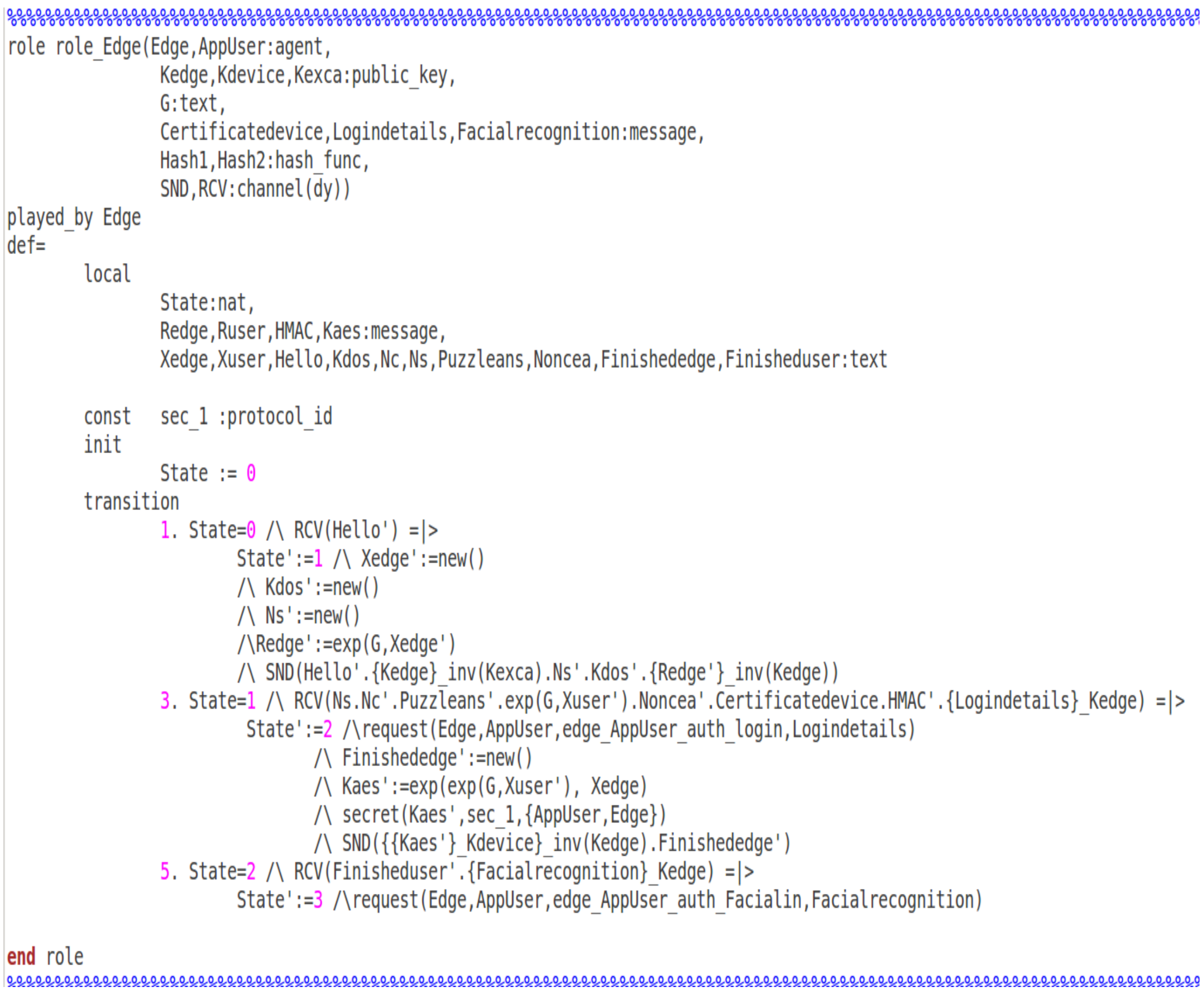
6.2.1 Authentication of Registered User

The message flow in this scenario follows Figure 17, the implemented HLPSL script and specifications of the role *Appuser* is shown in Figure 35, role *Edge* is shown in Figure 36, role of *session* and *environment* is shown in Figure 43.



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role role_AppUser(AppUser,Edge:agent,
                Kedge,Kdevice,Kexca:public_key,
                G:text,
                Certificatedevice,Logindetails,Facialrecognition:message,
                Hash1,Hash2:hash_func,
                SND,RCV:channel(dy))
played_by AppUser
def=
        local
                State:nat,
                Redge,Ruser,HMAC,Kaes:message,
                Xedge,Xuser,Hello,Kdos,Nc,Ns,Puzzleans,Noncea,Finishededge,Finisheduser:text

        const   sec_1 :protocol_id
        init
                State := 0
        transition
                1. State=0 /\ RCV(start) =|>
                        State':=1 /\ Hello':=new()
                                /\ SND(Hello')
                2. State=1 /\ RCV(Hello'.{Kedge}_inv(Kexca).Ns'.Kdos'.{exp(G,Xedge')}_inv(Kedge)) =|>
                        State':=2/\ Noncea':=new()
                                /\ Xuser':=new()
                                /\Ruser':=exp(G,Xuser')
                                /\ Puzzleans':=new()
                                /\ Nc':=new()
                                /\HMAC':=Hash2(Hash1(Certificatedevice.Noncea'))
                                /\ SND(Ns'.Nc'.Puzzleans'.Ruser'.Noncea'.Certificatedevice.HMAC'.{Logindetails}_Kedge)
                                /\witness(AppUser,Edge,edge_AppUser_auth_login,Logindetails)
                4. State=2 /\ RCV({{Kaes'}_Kdevice}_inv(Kedge).Finishededge') =|>
                        State':=3/\ secret(Kaes',sec_1,{AppUser,Edge})
                                /\ Finisheduser':=new()
                                /\ SND(Finisheduser'.{Facialrecognition}_Kedge)
                                /\witness(AppUser,Edge,edge_AppUser_auth_Facialin,Facialrecognition)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 35. HLPSL specification for role app user.



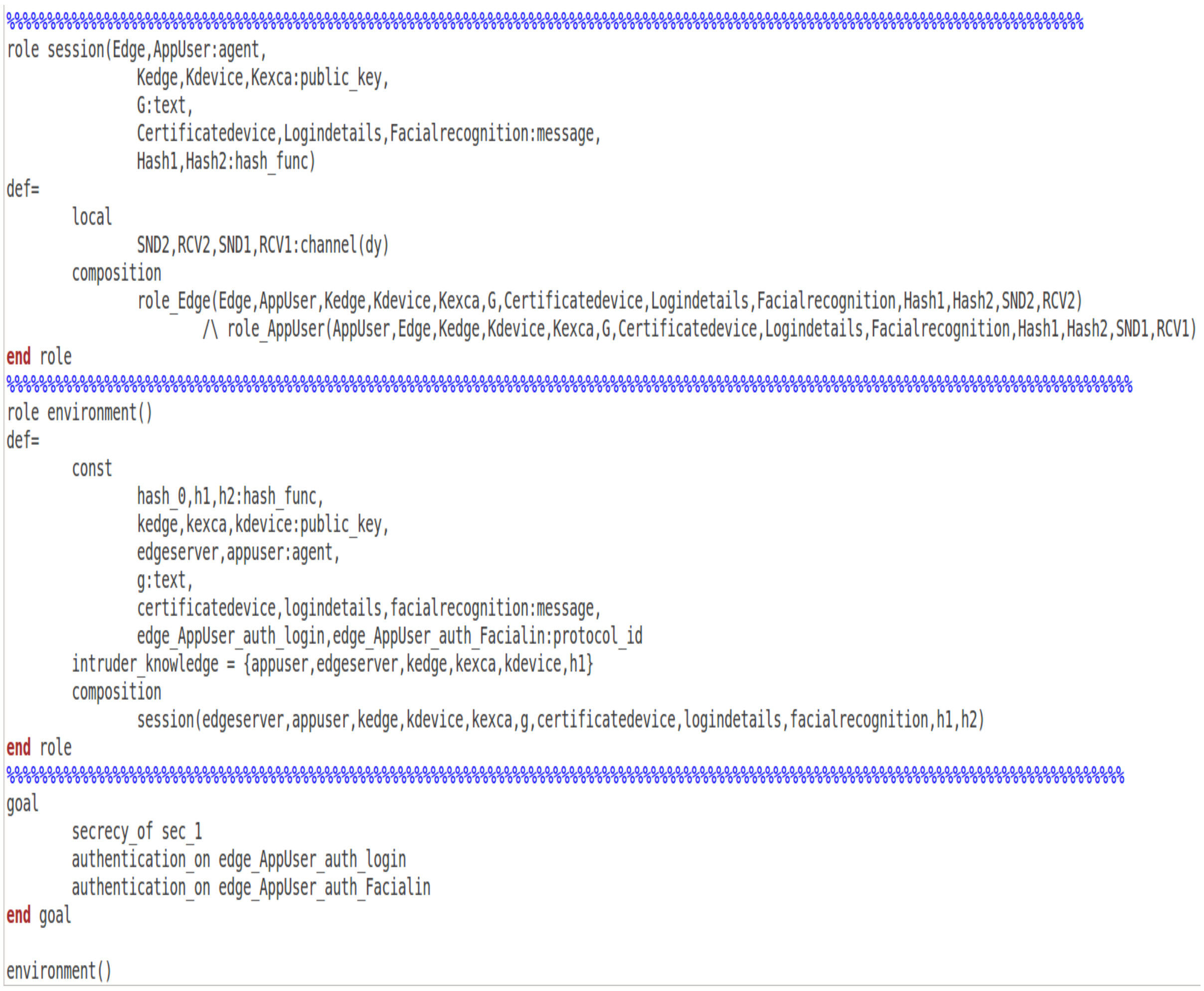
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role role_Edge(Edge,AppUser:agent,
               Kedge,Kdevice,Kexca:public_key,
               G:text,
               Certificatedevice,Logindetails,Facialrecognition:message,
               Hash1,Hash2:hash_func,
               SND,RCV:channel(dy))
played_by Edge
def=
        local
               State:nat,
               Redge,Ruser,HMAC,Kaes:message,
               Xedge,Xuser,Hello,Kdos,Nc,Ns,Puzzleans,Noncea,Finishededge,Finisheduser:text

        const  sec_1 :protocol_id
        init
               State := 0
        transition
               1. State=0 /\ RCV(Hello') =|>
                      State':=1 /\ Xedge':=new()
                      /\ Kdos':=new()
                      /\ Ns':=new()
                      /\Redge':=exp(G,Xedge')
                      /\ SND(Hello'.{Kedge}_inv(Kexca).Ns'.Kdos'.{Redge'}_inv(Kedge))
               3. State=1 /\ RCV(Ns.Nc'.Puzzleans'.exp(G,Xuser').Noncea'.Certificatedevice.HMAC'.{Logindetails}_Kedge) =|>
                       State':=2 /\request(Edge,AppUser,edge_AppUser_auth_login,Logindetails)
                             /\ Finishededge':=new()
                             /\ Kaes':=exp(exp(G,Xuser'), Xedge)
                             /\ secret(Kaes',sec_1,{AppUser,Edge})
                             /\ SND({{Kaes'}_Kdevice}_inv(Kedge).Finishededge')
               5. State=2 /\ RCV(Finisheduser'.{Facialrecognition}_Kedge) =|>
                      State':=3 /\request(Edge,AppUser,edge_AppUser_auth_Facialin,Facialrecognition)

end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 36. HLPSL specification for role edge.

The message exchanges in this scenario happens in the usual way as explained in the previous scenarios in registration phase. The AppUser sends his *Logindetails* encrypted with the public key of the edge to make sure it is only accessible by edge server. Moreover, the *Appuser* is already registered with the edge server, hence the *logindetails* are declared to be known secret message between *Appuser* and *Edge*. Once the *Edge* receives the message containing the *logindetails*, it verifies the *logindetails* with the stored details. Then with the new *Ruser'= exp(G,Xuser')* and already generated *Xedge* (in Figure17, *Xuser'* is denoted as r_{user} and *Xedge* as r_{edge}), it generates the session key from *Kaes':=exp(Ruser',Xedge)*, which is equivalent to *Kaes':= exp(exp(G,Xuser'),Xedge)*. Then the *Edge* encrypts *Kaes'* with the public key of the app user's device, and signs it with its private key. After that *Edge* sends the final message back to the app user together with a finished message. The *Appuser* receives the session key safely and finally sends back finished message.

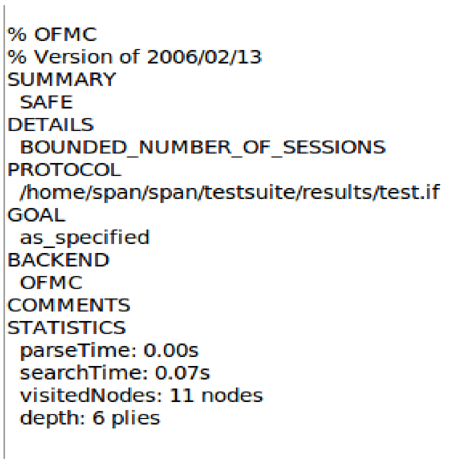
Here the secrecy goal is defined as *sec_1*, and the authentication goals are defined as *edge_Appuser_auth_login* and *edge_AppUser_auth_Facialin*. The *sec_1* represents the secrecy of session key *Kaes* between *Edge* and the *Appuser*. The *edge_Appuser_auth_login* states *logindetails* of the *Appuser* is verified at the *Edge*, and *edge_AppUser_auth_Facialin* states the facial recognition input of the user is verified at the *Edge*.

Simulated result in OFMC backend for the proposed authentication scheme is shown in Figure 38. The result validates that the proposed scheme is safe.



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role session(Edge,AppUser:agent,
                Kedge,Kdevice,Kexca:public_key,
                G:text,
                Certificatedevice,Logindetails,Facialrecognition:message,
                Hash1,Hash2:hash_func)
def=
        local
                SND2,RCV2,SND1,RCV1:channel(dy)
        composition
                role_Edge(Edge,AppUser,Kedge,Kdevice,Kexca,G,Certificatedevice,Logindetails,Facialrecognition,Hash1,Hash2,SND2,RCV2)
                        /\ role_AppUser(AppUser,Edge,Kedge,Kdevice,Kexca,G,Certificatedevice,Logindetails,Facialrecognition,Hash1,Hash2,SND1,RCV1)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role environment()
def=
        const
                hash_0,h1,h2:hash_func,
                kedge,kexca,kdevice:public_key,
                edgeserver,appuser:agent,
                g:text,
                certificatedevice,logindetails,facialrecognition:message,
                edge_AppUser_auth_login,edge_AppUser_auth_Facialin:protocol_id
        intruder_knowledge = {appuser,edgeserver,kedge,kexca,kdevice,h1}
        composition
                session(edgeserver,appuser,kedge,kdevice,kexca,g,certificatedevice,logindetails,facialrecognition,h1,h2)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
goal
        secrecy_of sec_1
        authentication_on edge_AppUser_auth_login
        authentication_on edge_AppUser_auth_Facialin
end goal

environment()
```

Figure 37. HLPSL specification for role session and environment.



```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/test.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.07s
  visitedNodes: 11 nodes
  depth: 6 plies
```

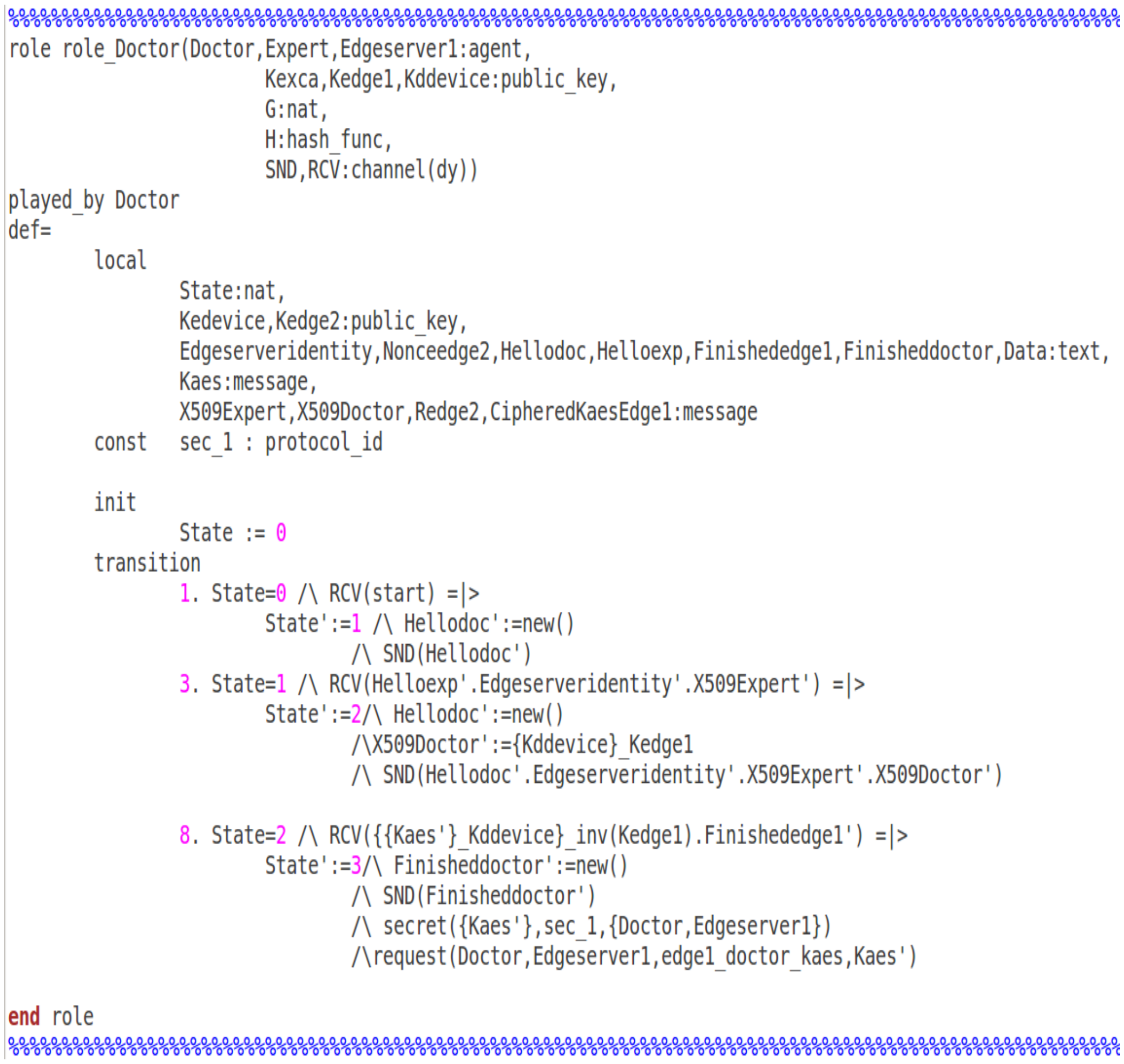
Figure 38. Protocol verification results using OFMC backend.

6.2.2 Authentication Between Doctor and Expert

The message flow in this scenario follows Figure 18, the implemented HLPSL script and specifications of the role *Doctor* is shown in Figure 39, the role *Expert* is shown in Figure 40, role *Edgeserver1* is shown in Figure 41, role *Edgeserver2* is shown in Figure 42, role *session* and *environment* is shown in Figure 43.

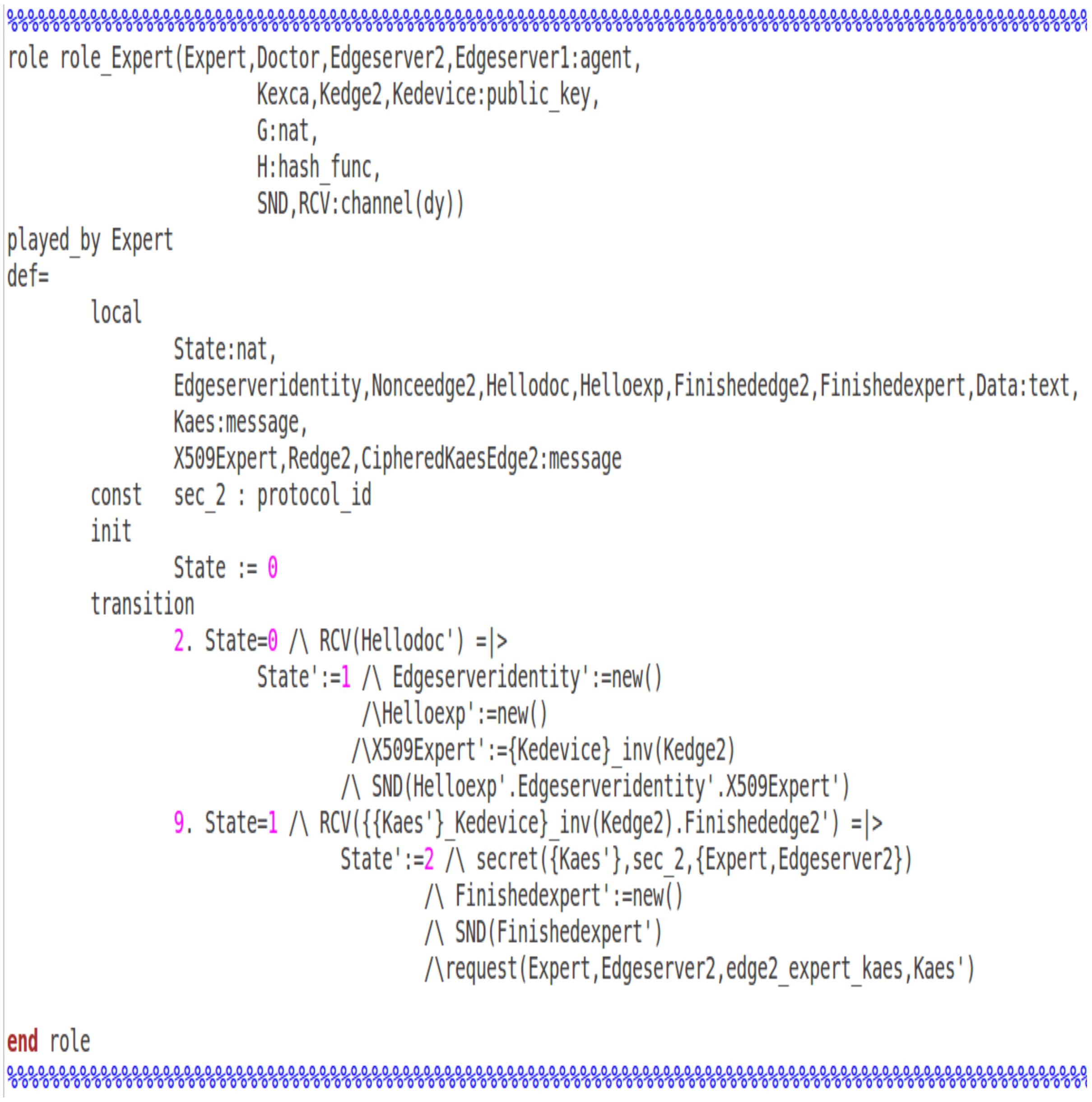
In Figure 43 two secrecy goals *sec_1,sec_2*, and two authentication goals *edge1_doctor_kaes* and *edge2_expert_kaes* have been defined. *sec_1* represents the secrecy of *Kaes* between *Edge1* and *Doctor*, *sec_2* represents the secrecy of *Kaes* between the *Edge2* and the *Expert*. On the other hand, *edge1_doctor_kaes* states *Kaes* is verified at *Doctor* that it has been generated by *Edge1*, and *edge2_expert_kaes* states *Kaes* is verified at *Expert* that it has been generated by *Edge2*.

Verification result using OFMC backend for the proposed authentication scheme between *Doctor* and *Expert* can be seen from the Figure 44. The result validates that the proposed scheme is safe.



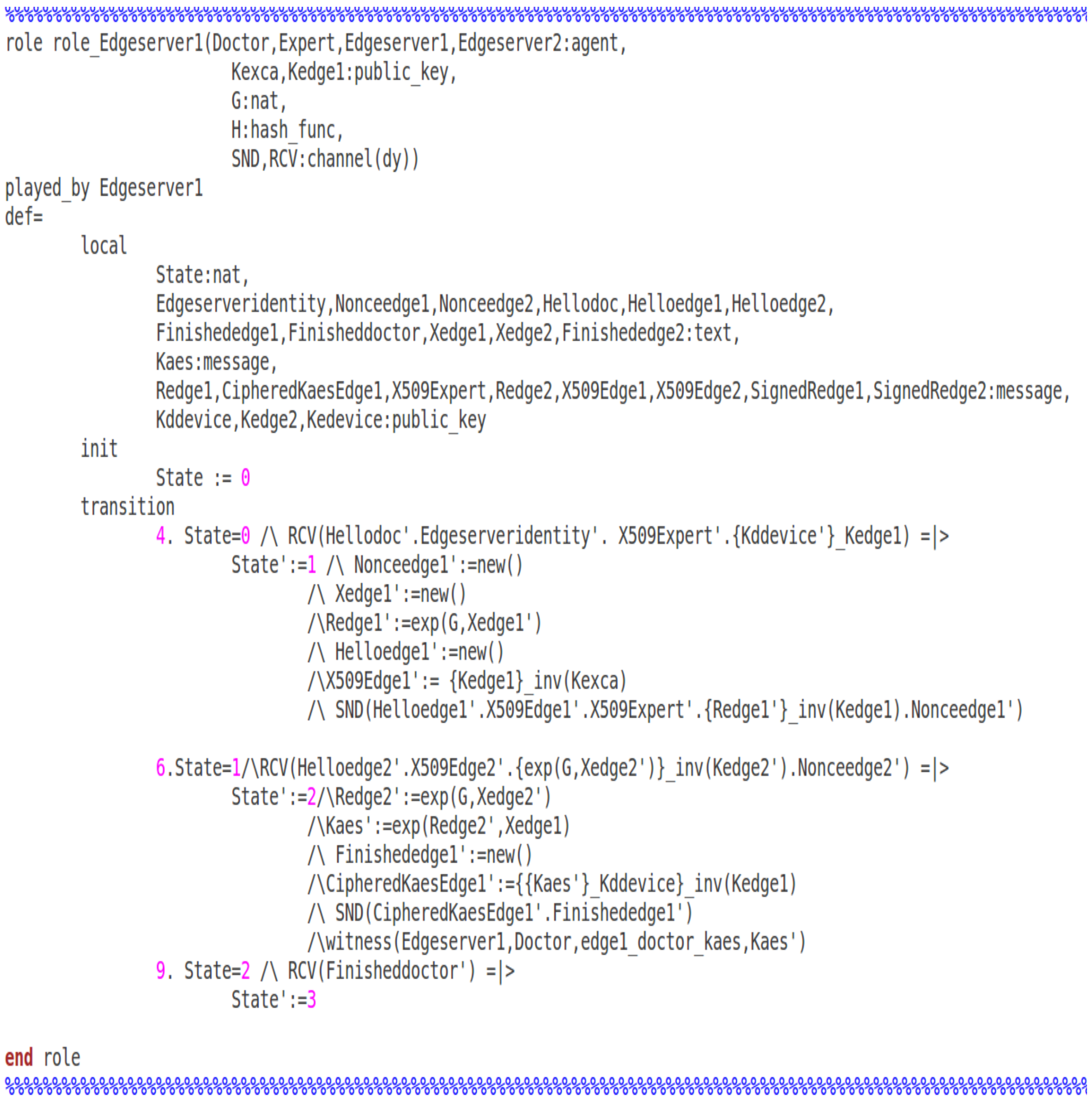
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role role_Doctor(Doctor,Expert,Edgeserver1:agent,
                        Kexca,Kedge1,Kddevice:public_key,
                        G:nat,
                        H:hash_func,
                        SND,RCV:channel(dy))
played_by Doctor
def=
        local
                State:nat,
                Kedevice,Kedge2:public_key,
                Edgeserveridentity,Nonceedge2,Hellodoc,Helloexp,Finishededge1,Finisheddoctor,Data:text,
                Kaes:message,
                X509Expert,X509Doctor,Redge2,CipheredKaesEdge1:message
        const   sec_1 : protocol_id

        init
                State := 0
        transition
                1. State=0 /\ RCV(start) =|>
                        State':=1 /\ Hellodoc':=new()
                                /\ SND(Hellodoc')
                3. State=1 /\ RCV(Helloexp'.Edgeserveridentity'.X509Expert') =|>
                        State':=2/\ Hellodoc':=new()
                                /\X509Doctor':={Kddevice}_Kedge1
                                /\ SND(Hellodoc'.Edgeserveridentity'.X509Expert'.X509Doctor')

                8. State=2 /\ RCV({{Kaes'}_Kddevice}_inv(Kedge1).Finishededge1') =|>
                        State':=3/\ Finisheddoctor':=new()
                                /\ SND(Finisheddoctor')
                                /\ secret({Kaes'},sec_1,{Doctor,Edgeserver1})
                                /\request(Doctor,Edgeserver1,edge1_doctor_kaes,Kaes')

end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 39. HLPSL specification for role doctor.



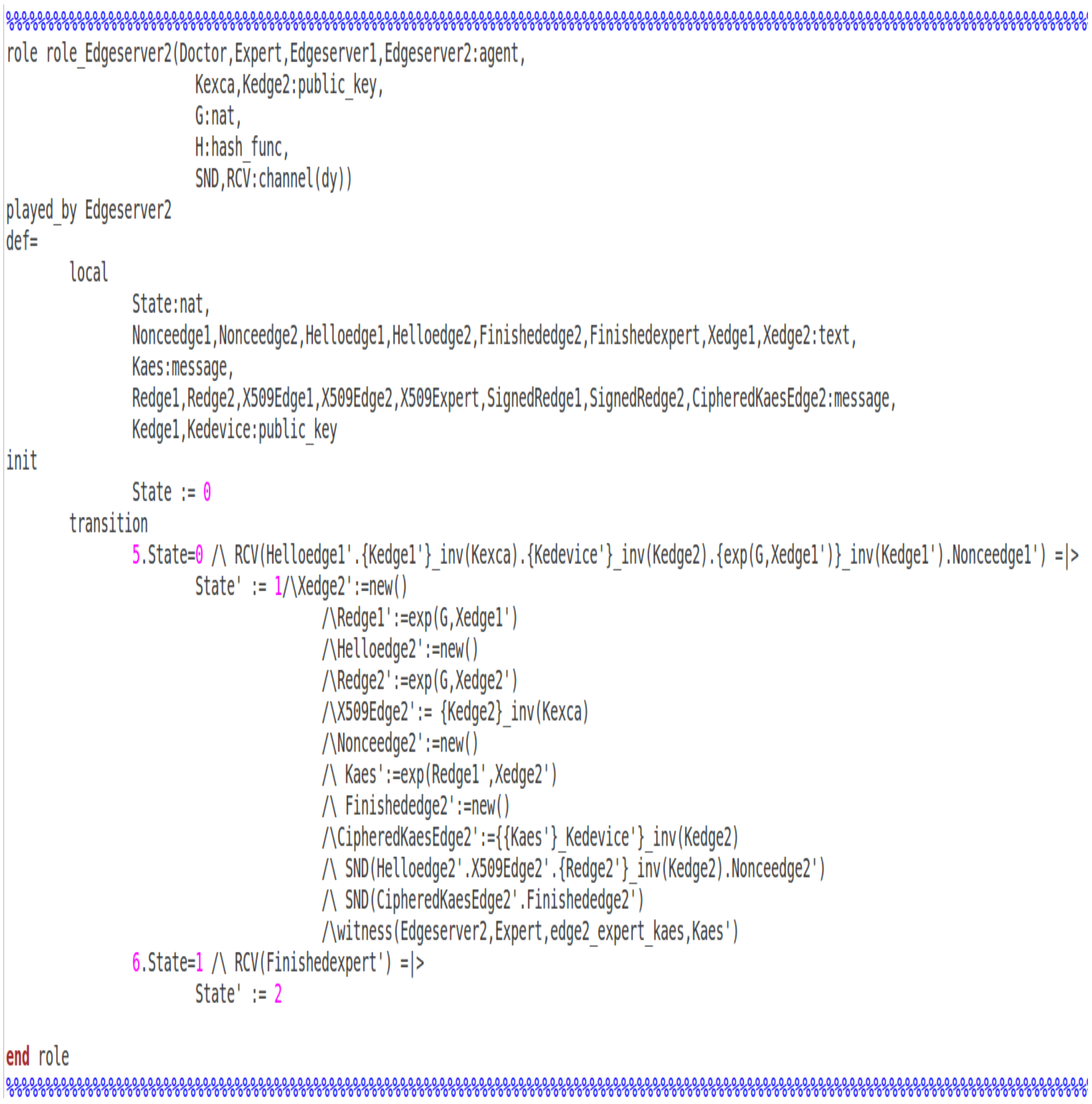
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role role_Expert(Expert,Doctor,Edgeserver2,Edgeserver1:agent,
                        Kexca,Kedge2,Kedevice:public_key,
                        G:nat,
                        H:hash_func,
                        SND,RCV:channel(dy))
played_by Expert
def=
        local
                State:nat,
                Edgeserveridentity,Nonceedge2,Hellodoc,Helloexp,Finishededge2,Finishedexpert,Data:text,
                Kaes:message,
                X509Expert,Redge2,CipheredKaesEdge2:message
        const   sec_2 : protocol_id
        init
                State := 0
        transition
                2. State=0 /\ RCV(Hellodoc') =|>
                        State':=1 /\ Edgeserveridentity':=new()
                                  /\Helloexp':=new()
                                 /\X509Expert':={Kedevice}_inv(Kedge2)
                                /\ SND(Helloexp'.Edgeserveridentity'.X509Expert')
                9. State=1 /\ RCV({{Kaes'}_Kedevice}_inv(Kedge2).Finishededge2') =|>
                                State':=2 /\ secret({Kaes'},sec_2,{Expert,Edgeserver2})
                                        /\ Finishedexpert':=new()
                                        /\ SND(Finishedexpert')
                                        /\request(Expert,Edgeserver2,edge2_expert_kaes,Kaes')

end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 40. HLPSL specification for role expert.



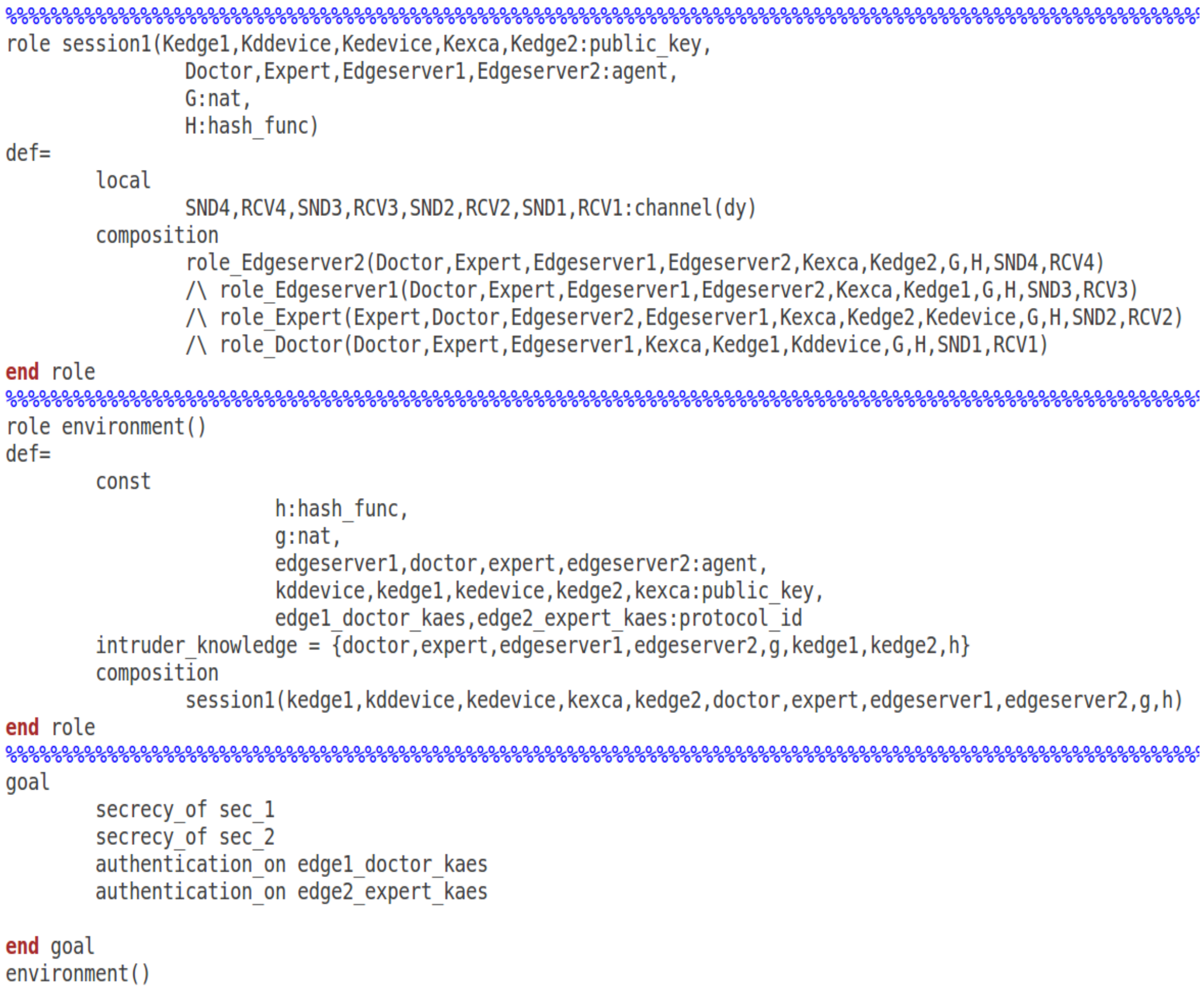
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role role_Edgeserver1(Doctor,Expert,Edgeserver1,Edgeserver2:agent,
                        Kexca,Kedge1:public_key,
                        G:nat,
                        H:hash_func,
                        SND,RCV:channel(dy))
played_by Edgeserver1
def=
        local
                State:nat,
                Edgeserveridentity,Nonceedge1,Nonceedge2,Hellodoc,Helloedge1,Helloedge2,
                Finishededge1,Finisheddoctor,Xedge1,Xedge2,Finishededge2:text,
                Kaes:message,
                Redge1,CipheredKaesEdge1,X509Expert,Redge2,X509Edge1,X509Edge2,SignedRedge1,SignedRedge2:message,
                Kddevice,Kedge2,Kedevice:public_key
        init
                State := 0
        transition
                4. State=0 /\ RCV(Hellodoc'.Edgeserveridentity'. X509Expert'.{Kddevice'}_Kedge1) =|>
                        State':=1 /\ Nonceedge1':=new()
                                /\ Xedge1':=new()
                                /\Redge1':=exp(G,Xedge1')
                                /\ Helloedge1':=new()
                                /\X509Edge1':= {Kedge1}_inv(Kexca)
                                /\ SND(Helloedge1'.X509Edge1'.X509Expert'.{Redge1'}_inv(Kedge1).Nonceedge1')

                6.State=1/\RCV(Helloedge2'.X509Edge2'.{exp(G,Xedge2')}_inv(Kedge2').Nonceedge2') =|>
                        State':=2/\Redge2':=exp(G,Xedge2')
                                /\Kaes':=exp(Redge2',Xedge1)
                                /\ Finishededge1':=new()
                                /\CipheredKaesEdge1':={{Kaes'}_Kddevice}_inv(Kedge1)
                                /\ SND(CipheredKaesEdge1'.Finishededge1')
                                /\witness(Edgeserver1,Doctor,edge1_doctor_kaes,Kaes')
                9. State=2 /\ RCV(Finisheddoctor') =|>
                        State':=3

end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 41. HLPSL specification for role edge server1.



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role role_Edgeserver2(Doctor,Expert,Edgeserver1,Edgeserver2:agent,
                      Kexca,Kedge2:public_key,
                      G:nat,
                      H:hash_func,
                      SND,RCV:channel(dy))
played_by Edgeserver2
def=
        local
                State:nat,
                Nonceedge1,Nonceedge2,Helloedge1,Helloedge2,Finishededge2,Finishedexpert,Xedge1,Xedge2:text,
                Kaes:message,
                Redge1,Redge2,X509Edge1,X509Edge2,X509Expert,SignedRedge1,SignedRedge2,CipheredKaesEdge2:message,
                Kedge1,Kedevice:public_key
init
                State := 0
        transition
                5.State=0 /\ RCV(Helloedge1'.{Kedge1'}_inv(Kexca).{Kedevice'}_inv(Kedge2).{exp(G,Xedge1')}_inv(Kedge1').Nonceedge1') =|>
                        State' := 1/\Xedge2':=new()
                                        /\Redge1':=exp(G,Xedge1')
                                        /\Helloedge2':=new()
                                        /\Redge2':=exp(G,Xedge2')
                                        /\X509Edge2':= {Kedge2}_inv(Kexca)
                                        /\Nonceedge2':=new()
                                        /\ Kaes':=exp(Redge1',Xedge2')
                                        /\ Finishededge2':=new()
                                        /\CipheredKaesEdge2':={{Kaes'}_Kedevice'}_inv(Kedge2)
                                        /\ SND(Helloedge2'.X509Edge2'.{Redge2'}_inv(Kedge2).Nonceedge2')
                                        /\ SND(CipheredKaesEdge2'.Finishededge2')
                                        /\witness(Edgeserver2,Expert,edge2_expert_kaes,Kaes')
                6.State=1 /\ RCV(Finishedexpert') =|>
                        State' := 2

end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

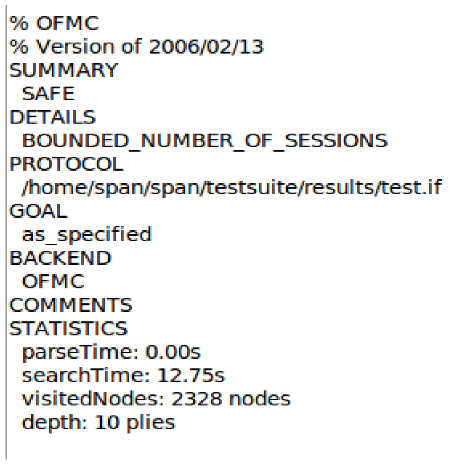
Figure 42. HLPSL specification for role edge server2.



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role session1(Kedge1,Kddevice,Kedevice,Kexca,Kedge2:public_key,
                Doctor,Expert,Edgeserver1,Edgeserver2:agent,
                G:nat,
                H:hash_func)
def=
        local
                SND4,RCV4,SND3,RCV3,SND2,RCV2,SND1,RCV1:channel(dy)
        composition
                role_Edgeserver2(Doctor,Expert,Edgeserver1,Edgeserver2,Kexca,Kedge2,G,H,SND4,RCV4)
                /\ role_Edgeserver1(Doctor,Expert,Edgeserver1,Edgeserver2,Kexca,Kedge1,G,H,SND3,RCV3)
                /\ role_Expert(Expert,Doctor,Edgeserver2,Edgeserver1,Kexca,Kedge2,Kedevice,G,H,SND2,RCV2)
                /\ role_Doctor(Doctor,Expert,Edgeserver1,Kexca,Kedge1,Kddevice,G,H,SND1,RCV1)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role environment()
def=
        const
                        h:hash_func,
                        g:nat,
                        edgeserver1,doctor,expert,edgeserver2:agent,
                        kddevice,kedge1,kedevice,kedge2,kexca:public_key,
                        edge1_doctor_kaes,edge2_expert_kaes:protocol_id
        intruder_knowledge = {doctor,expert,edgeserver1,edgeserver2,g,kedge1,kedge2,h}
        composition
                session1(kedge1,kddevice,kedevice,kexca,kedge2,doctor,expert,edgeserver1,edgeserver2,g,h)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
goal
        secrecy_of sec_1
        secrecy_of sec_2
        authentication_on edge1_doctor_kaes
        authentication_on edge2_expert_kaes

end goal
environment()
```

Figure 43. HLPSL specification for role session and environment

Result of the Authentication between doctor and expert part of the proposed scheme using OFMC backend can be seen from the Figure 44. The results validates that the proposed scheme is safe.



```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/test.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 12.75s
  visitedNodes: 2328 nodes
  depth: 10 plies
```

Figure 44. Protocol verification results using OFMC backend.

7 INFORMAL SECURITY ANALYSIS

This chapter discusses the informal verification of the prevention of possible attacks in our proposed scheme. In the formal verification, we have already proved that the scheme provides protection against man in the middle attack and replay attack. Apart from those attacks following threats are also mitigated by our scheme.

Protection against impersonation attack:

In the registration phase when the entity initiates the communication with the edge server, it expects the edge server to send back an initial message with its X.509 certificate issued by a legitimate external certificate authority. This certificate cannot be generated by the intruder, hence it prevents the adversary from impersonating the edge server. Moreover, in the registration phase of the network admin, network admin is required to provide his facial recognition input, after feeding the registration details. Then the edge server verifies it with the already stored details of the network admin and this prevents an adversary from impersonating the network admin. In all the other cases of the registration phase, network admin is required to provide his user name and password as a way to authenticate himself to the edge server and at the end of registration, he is also required to give his facial recognition input as an additional authentication to prove his identity. This two factor authentication provides strong protection against the impersonation attack.

Likewise, in the authentication phase, when the registered users like network admin, workers, doctor, patient, and expert initiates the communication with the edge server, they are required to provide their user name and password, and their facial recognition input. This two factor authentication prevents the intruder from impersonating a registered user.

In the authentication phase between doctor and expert, edge servers verifies the client requests with their implicit certificates, and they also authenticate the session keys by signing it with their private keys. It prevents from any impersonation attack in this phase. Hence, we conclude that our scheme has strong protection against the impersonation attack.

Protection against DoS attack:

In our scheme, we have implemented a challenge-based mechanism [45] to prevent the DoS attack. In both phases, the edge server provides a challenge to the entity initiating the communication. The difficulty of the challenge which depends on K_{dos} is adjusted with the severity of the DoS attack present. Normally the challenge will be in moderate difficulty but when there is a lot of requests in the ques for the edge server, it increases the difficulty. Also answer for the puzzle depends on the random number generated in the client's side, so this answer cannot be used in another instance. Incorporating this puzzle-based prevention mechanism guards our scheme against DoS attack.

Protection against password guessing attack:

In the message exchanges which consists of the user name and password, the secrecy of the password is preserved by encrypting it with the public key of the edge server, which only allows the edge server to have the access of it. Then as an extra layer of protection, only the hash of the password is exchanged instead of the direct passwords. Hence, even if the attacker gets the access for the hashed password, he will not be able to decode it. Also in the scheme it is required to provide facial recognition input as a final authentication, it prevents the attacker to fully authenticate himself, therefore he will never be able to pass these three layer protection and log into the account.

Protection against privileged insider attack:

When a privileged insider from the company or hospital gets the access of the password and use it to log into the user account, the scheme requires the attacker to use a device with registered MAC address which acts as a layer of protection against insiders. Even if the adversary acquired the access for a registered device, as a final authentication requirement of the scheme, he will have to provide facial recognition input. This three-factor authentication in our scheme prevents the privileged insider attack from happening.

In the case of patient's registration, he does not have a device to be registered. Therefore the network admin feeds all the details from his device, then the password for the patient will be sent to the patient's phone number. Each time the patient tries to log in with his user name and password, a verification code will be sent to his registered mobile number, and he will also be asked for facial recognition input similar to other scenarios. This three-factor authentication ensures that, even the highly privileged insider will not be able to have access to the patient's account and data.

Protection against device capture attack:

Assume the adversary gets the access to the MAR device, The required log-in credentials prevents the adversary from accessing the account. Even if he manages to get the user name and passwords he will be required to provide the facial recognition input, which makes it impossible for the adversary to perform the attack even after taking back the control of the MAR device.

Protection against message tampering:

In our scheme we have used HMAC function with two keys, one key is the public key of the edge server, and the other key is a secret key K between Edge server and the communicating entity. The secret key is computed for message instance and exchanged using Elliptic-Curve-Diffie-Hellman(ECDH) key exchange algorithm. ECDH makes it impossible for the intruder to access the secret key K. Normally HMAC is enough to check the integrity of the message but the strength of the algorithm depends on the complexity of the key being used for the hash functions involved. So by generating a secret key K which can never be accessible by the adversary, and using it in the computation of HMAC, we ensured the message integrity throughout the process and made our system highly protected against message tampering.

8 DISCUSSION

In this section, a critical analysis is presented considering the objectives of the thesis and how well they are being met. Finally, future research directions have been discussed.

8.1 Evaluation on Meeting the Thesis Objectives

The objective of the thesis is to propose a secure and privacy-preserving AR application to be used in the health care sector. Given the potential of AR technology, there are numerous options in incorporating technology in the health care sector. However, this thesis is directed towards addressing a current issue in the health care sector in rural areas, without compromising the security and privacy of the users.

In the thesis initially an architecture for a MAR application has been defined with key features such as displaying augmented view of patient information on the mobile device, augmenting the X-ray or scan image on top of the patient's actual body parts to assist the doctor, and enabling the doctor to interact with an expert and get real-time consultancy. Then all the possible security threats for the defined architecture have been analyzed, and possible solutions have been proposed. In order to establish a secure end to end communication link between the participating entities, it is vital to have strong authentication and key management scheme for the application. Hence, a strong authentication and key exchange scheme has been proposed for each scenario of the application. After that, the proposed scheme has been verified by using widely accepted formal and informal protocol verification methods.

8.2 Future Research Directions

The thesis defines the architecture of the AR application with basic functions, which includes a doctor, a patient, and an expert. This application considers one doctor requesting assistance from one expert, and it was assumed that the expert is always available, which is not practical, in reality, there will be multiple doctors and multiple experts, and there could be overwhelming requests for the experts, which might question the availability of the experts. We can address this issue with two options; first, we could incorporate machine learning-based model which could learn from the previous collective consultations from the all the experts using the application, and then provide automated consultations and assistance in less serious issues. Secondly, we could introduce a pool of experts and depending on the availability of the experts, the doctors should be able to initiate the communication resource management queuing, and security has to be properly addressed for this case.

Further, we have assumed that the expert and doctor are being registered in an edge server and they are always reporting to their respective edge servers only. But in reality, the expert might be traveling to many places, and in this case, his credentials, certificate and other authentication details should be safely migrated from one edge server to another, and a proper key management scheme should be proposed for this complex case as well.

Even though the proposed scheme is secure enough to make the proposed application to reality, there is always room for additional protection by incorporating all the proposed solutions in the security analysis chapter and also incorporating the other security techniques like blockchain and embedded systems level security.

The thesis is focused on the conceptual design of the AR application, and the security validation is done using formal and informal verification of the proposed protocol. But the security of the application could be further justified by implementing the full real AR application in MAR device and simulating all the communication scenarios in a real experimental environment and validating the security level of the application by performing all the possible attacks. This practical software implementation also introduces another research direction to optimize protocol, process flow, and device selection for the application in order to achieve the desired latency levels.

9 CONCLUSION

The goal of the research work is to propose a secure key exchange scheme for MAR application in the health care sector. In this thesis, we proposed an architecture for a MAR health care application based on MEC. Then we analyzed the security threats specific to the proposed architecture and proposed viable solutions to address those issues. After that, we proposed a registration and authentication key management scheme for each scenario of the application. Then we used widely accepted formal and informal verification methods to validate the security properties of our proposed scheme. The results of both methods yielded that the proposed scheme is secure enough to be implemented. As an outcome of the thesis, we have proposed a viable architecture for a revolutionary MAR health care application and produced a highly secure, registration, and authentication key management scheme for the application. With application specific modifications, our proposed key management scheme has the potential to be used in any similar, privacy concerned sensitive MAR health care applications.

10 REFERENCES

[1] Parkvall S., Dahlman E., Furuskar A. & Frenne M. (2017) Nr: The new 5g radio access technology. IEEE Communications Standards Magazine 1, pp. 24–30.

[2] Agiwal M., Roy A. & Saxena N. (2016) Next generation 5g wireless networks: A comprehensive survey. IEEE Communications Surveys & Tutorials 18, pp. 1617–1655.

[3] Erol-Kantarci M. & Sukhmani S. (2018) Caching and computing at the edge for mobile augmented reality and virtual reality (ar/vr) in 5g. In: Ad Hoc Networks, Springer, pp. 169–177.

[4] Chen L., Day T.W., Tang W. & John N.W. (2017) Recent developments and future challenges in medical mixed reality. In: 2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), IEEE, pp. 123–135.

[5] Milgram P., Takemura H., Utsumi A. & Kishino F. (1995) Augmented reality: A class of displays on the reality-virtuality continuum. In: Telemanipulator and telepresence technologies, vol. 2351, International Society for Optics and Photonics, vol. 2351, pp. 282–293.

[6] Billinghurst M., Clark A., Lee G. et al. (2015) A survey of augmented reality. Foundations and Trends® in Human–Computer Interaction 8, pp. 73–272.

[7] Huang Z., Hui P., Peylo C. & Chatzopoulos D. (2013) Mobile augmented reality survey: a bottom-up approach. arXiv preprint arXiv:1309.4413 .

[8] Chatzopoulos D., Bermejo C., Huang Z. & Hui P. (2017) Mobile augmented reality survey: From where we are to where we go. Ieee Access 5, pp. 6917–6950.

[9] Van Krevelen D. & Poelman R. (2007) Augmented reality: Technologies, applications, and limitations. Vrije Univ. Amsterdam, Dep. Comput. Sci .

[10] Zhang W., Lin S., Hassani Bijarbooneh F. & Cheng H.F. (2018) Cloudar: A cloud-based framework for mobile augmented reality. arXiv preprint arXiv:1805.03060 .

[11] Renevier P. & Nigay L. (2001) Mobile collaborative augmented reality: the augmented stroll. In: IFIP International Conference on Engineering for Human-Computer Interaction, Springer, pp. 299–316.

[12] Azuma R., Baillot Y., Behringer R., Feiner S., Julier S. & MacIntyre B. (2001) Recent advances in augmented reality. IEEE computer graphics and applications 21, pp. 34–47.

[13] Kim Y. & Kim W. (2014) Implementation of augmented reality system for smartphone advertisements. international journal of multimedia and ubiquitous engineering 9, pp. 385–392.

[14] Huang Z., Li W., Hui P. & Peylo C. (2014) Cloudridar: A cloud-based architecture for mobile augmented reality. In: Proceedings of the 2014 workshop on Mobile augmented reality and robotic technology-based systems, ACM, pp. 29–34.

[15] Zhang W., Han B. & Hui P. (2017) On the networking challenges of mobile augmented reality. In: Proceedings of the Workshop on Virtual Reality and Augmented Reality Network, ACM, pp. 24–29.

[16] (2014) Mobile-edge computing– introductory technical white paper. Tech. Rep. Issue 1, European Telecommunications Standards Institute. URL: `https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobile-edge_Computing_-_Introductory_Technical_White_Paper_V1`.

[17] Ai Y., Peng M. & Zhang K. (2018) Edge computing technologies for internet of things: a primer. Digital Communications and Networks 4, pp. 77–86.

[18] Mao Y., You C., Zhang J., Huang K. & Letaief K.B. (2017) Mobile edge computing: Survey and research outlook. arXiv preprint arXiv:1701.01090 .

[19] Herron J. (2016) Augmented reality in medical education and training. Journal of Electronic Resources in Medical Libraries 13, pp. 51–55.

[20] Fuchs H., Livingston M.A., Raskar R., Keller K., Crawford J.R., Rademacher P., Drake S.H., Meyer A.A. et al. (1998) Augmented reality visualization for laparoscopic surgery. In: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, pp. 934–943.

[21] Navab N., Bani-Kashemi A. & Mitschke M. (1999) Merging visible and invisible: Two camera-augmented mobile c-arm (camc) applications. In: Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99), IEEE, pp. 134–141.

[22] Merten M. (2007) Erweiterte realität-verschmelzung zweier welten. Deutsches Ärzteblatt 104, pp. 840–2.

[23] Shuhaiber J.H. (2004) Augmented reality in surgery. Archives of surgery 139, pp. 170–174.

[24] Shenai M.B., Dillavou M., Shum C., Ross D., Tubbs R.S., Shih A. & Guthrie B.L. (2011) Virtual interactive presence and augmented reality (vipar) for remote surgical assistance. Operative Neurosurgery 68, pp. ons200–ons207.

[25] Smyth B. (2011) Formal verification of cryptographic protocols with automated reasoning. Ph.D. thesis, University of Birmingham.

[26] Meadows C. (2003) Formal methods for cryptographic protocol analysis: Emerging issues and trends. IEEE journal on selected areas in communications 21, pp. 44–54.

[27] Pironti A., Pozza D. & Sisto R. (2012) Automated formal methods for security protocol engineering. In: Cyber Security Standards, Practices and Industrial Applications: Systems and Methodologies, IGI Global, pp. 138–166.

[28] Das A.K. (2017) A secure and effective biometric-based user authentication scheme for wireless sensor networks using smart card and fuzzy extractor. International Journal of Communication Systems 30, p. e2933.

[29] Kumar P., Braeken A., Gurtov A., Iinatti J. & Ha P.H. (2017) Anonymous secure framework in connected smart home environments. IEEE Transactions on Information Forensics and Security 12, pp. 968–979.

[30] Nicanfar H. & Leung V.C. (2013) Multilayer consensus ecc-based password authenticated key-exchange (mcepak) protocol for smart grid system. IEEE Transactions on Smart Grid 4, pp. 253–264.

[31] Islam S.H., Amin R., Biswas G., Farash M.S., Li X. & Kumari S. (2017) An improved three party authenticated key exchange protocol using hash function and elliptic curve cryptography for mobile-commerce environments. Journal of King Saud University-Computer and Information Sciences 29, pp. 311–324.

[32] Viganò L. (2006) Automated security protocol analysis with the avispa tool. Electronic Notes in Theoretical Computer Science 155, pp. 61–86.

[33] Dolev D. & Yao A. (1983) On the security of public key protocols. IEEE Transactions on information theory 29, pp. 198–208.

[34] Roesner F., Kohno T. & Molnar D. (2014) Security and privacy for augmented reality systems. Commun. ACM 57, pp. 88–96.

[35] Templeman R., Rahman Z., Crandall D. & Kapadia A. (2012) Placeraider: Virtual theft in physical spaces with smartphones. arXiv preprint arXiv:1209.5982 .

[36] McPherson R., Jana S. & Shmatikov V. (2015) No escape from reality: Security and privacy of augmented reality browsers. In: Proceedings of the 24th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, pp. 743–753.

[37] Zhang J., Chen B., Zhao Y., Cheng X. & Hu F. (2018) Data security and privacy-preserving in edge computing paradigm: Survey and open issues. IEEE Access 6, pp. 18209–18237.

[38] Roman R., Lopez J. & Mambo M. (2018) Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. Future Generation Computer Systems 78, pp. 680–698.

[39] Qin Z., Yan J., Ren K., Chen C.W. & Wang C. (2014) Towards efficient privacy-preserving image feature extraction in cloud computing. In: Proceedings of the 22nd ACM international conference on Multimedia, ACM, pp. 497–506.

[40] Hsu C.Y., Lu C.S. & Pei S.C. (2012) Image feature extraction in encrypted domain with privacy-preserving sift. IEEE transactions on image processing 21, pp. 4593–4607.

[41] Yang T., Ma J., Wang Q., Miao Y., Wang X. & Meng Q. (2018) Image feature extraction in encrypted domain with privacy-preserving hahn moments. IEEE Access 6, pp. 47521–47534.

[42] Sabahi F. (2011) Cloud computing security threats and responses. In: 2011 IEEE 3rd International Conference on Communication Software and Networks, IEEE, pp. 245–249.

[43] Chouhan P. & Singh R. (2016) Security attacks on cloud computing with possible solution. International Journal of Advanced Research in Computer Science and Software Engineering 6.

[44] Park C.S. (2016) A secure and efficient ecqv implicit certificate issuance protocol for the internet of things applications. IEEE Sensors Journal 17, pp. 2215–2223.

[45] Aura T., Nikander P. & Leiwo J. (2000) Dos-resistant authentication with client puzzles. In: International workshop on security protocols, Springer, pp. 170–177.

[46] Koblitz N., Menezes A. & Vanstone S. (2000) The state of elliptic curve cryptography. Designs, codes and cryptography 19, pp. 173–193.

[47] Durlanik A. & Sogukpinar I. (2005) Sip authentication scheme using ecdh. World Enformatika Soc Trans Eng Comput Technol 8, pp. 350–353.

[48] Mehibel N. & Hamadouche M. (2017) A new approach of elliptic curve diffie-hellman key exchange. In: 2017 5th International Conference on Electrical Engineering-Boumerdes (ICEE-B), IEEE, pp. 1–6.

[49] Miller V.S. (1985) Use of elliptic curves in cryptography. In: Conference on the theory and application of cryptographic techniques, Springer, pp. 417–426.

[50] Koblitz N. (1987) Elliptic curve cryptosystems. Mathematics of computation 48, pp. 203–209.

[51] Lauter K. (2004) The advantages of elliptic curve cryptography for wireless security. IEEE Wireless communications 11, pp. 62–67.

[52] Rivest R.L., Shamir A. & Adleman L. (1978) A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 21, pp. 120–126.

[53] Nystrom M. & Kaliski B. (2000) Pkcs# 10: Certification request syntax specification version 1.7. Tech. rep.