ESSMAR: Edge Supportive Secure Mobile Augmented Reality Architecture for Healthcare

An Braeken*, Pawani Porambage†, Amirthan Puvaneswaran‡, Madhusanka Liyanage§
*Industrial Engineering INDI, Vrije Universiteit Brussel, Belgium
†‡§Centre for Wireless Communications, University of Oulu, Finland
§School of Computer Science, University College Dublin, Ireland
an.braeken@vub.be*, pawani.porambage@oulu.fi†, amirthan.puvaneswaran@student.oulu.fi‡, madhusanka@ucd.ie§

***Abstract*—The recent advances in mobile devices and wireless communication sector transformed Mobile Augmented Reality (MAR) from science fiction to reality. Among the other MAR use cases, the incorporation of this MAR technology in the healthcare sector can elevate the quality of diagnosis and treatment for the patients. However, due to the highly sensitive nature of the data available in this process, it is also highly vulnerable to all types of security threats. In this paper, an edge-based secure architecture is presented for a MAR healthcare application. Based on the ESSMAR architecture, a secure key management scheme is proposed for both the registration and authentication phases. Then the security of the proposed scheme is validated using formal and informal verification methods.**

I. Introduction

Recent mobile technology advancements, edge-based data processing, and advanced storage facilities enable Augment Reality (AR) on the current generation of mobile devices such as mobile phones, tablets, Google glasses, etc. This is also Mobile Augmented Reality (MAR) in [1]. The main purpose of using this MAR technology in a large variety of applications is to assist the users with computer-simulated additional information. Thanks to this extra information, the users can have a better understanding of the situation and become able to make more informed decisions. MAR can be used in different areas such as industrial environments, healthcare, military, education, tourism, and entertainment [1]. The potential benefits of MAR in the healthcare sector have motivated numerous researchers to find innovative ways to incorporate MAR into many different healthcare applications, such as remote assistance in surgeries, X-ray rendering, etc [2].

Most of the existing MAR applications run as cloud-based solutions [3] [4]. However, such cloud-based solutions are expected to endure high latency, high jitter, in addition to less location awareness and localize service support. These factors are not particularly suitable for healthcare MAR applications.

In edge computing, the computing capabilities of the cloud are brought closer to the users, also called the edge of the mobile network [5]. Latency can be largely reduced by incorporating the edge server instead of a cloud based centralized server for managing both the data processing and data access tasks. This follows by the fact that propagation distances and complexity in the processing significantly decreases [6]. Hence, edge-based architecture is favoured for latency constraint applications like MAR.

In this paper, we propose an Edge Supportive Secure MAR (ESSMAR) architecture for healthcare applications to provide assistance for the doctors by improving their diagnosis using the additionally obtained information. However, due to the highly sensitive data in these processes, it is vital to address the security threats in the application. Hence, we analyze all the security threats based on the ESSMAR architecture and propose possible solutions. Further, we propose a registration and authentication key management scheme to establish secure communication between the participating entities. Then, we validate the security of the proposed scheme against the Man-in-the-Middle (MITM) attack and the replay attack, through formal verification using the tool AVISPA. In addition, an informal verification of the security strength against other threats is also presented for the proposed scheme.

The remainder of this paper has the following structure. Section II presents the proposed ESSMAR architecture. In Section , some background on related work is discussed. The proposed registration and authentication schemes are explained in Section IV. The formal and informal security analysis of the proposed architecture are respectively presented in Section V and VI. Finally, Section VII presents the conclusions and future work.

II. Proposed ESSMAR Architecture

Based on the AR architectures discussed in [3] [4], we propose an edge-based MAR architecture for healthcare applications, as shown in Figure 1. In the architecture, the cloud server keeps all the data and software updates related to the ESSMAR application. Here, the edge server acts as an intermediate entity between user and cloud server. When the user accesses the application, the edge server downloads the necessary data and services from the cloud server and keeps it in its storage. After that, the edge server manages all the data and computational offloading requests from the user. Mobile devices such as smartphones or tabs, act as MAR devices in the architecture. Based on the user requests, the MAR device augments the information from the edge server with the camera inputs and displays it to the user.

Our ESSMAR system can support the following key features.

TABLE I: Security analysis for ESSMAR architecture

Target	Attack	Solution
① Mobile device	Use third party applications to perform the place raider attack [7]. Manipulate the outputs and perform the click-jacking attack [8].	Access should be strictly controlled. Output policies should be defined to detect abnormalities in the display.
② Network between the edge and the device	Perform MITM, impersonation and DoS attacks.	Proper key agreement and authentication encryption schemes should be utilized. Puzzles can be used to prevent DoS attacks.
③ Edge server	**Data:** Access and tamper the data. Redirect the user to another rogue storage, resulting in a breach of confidentiality and integrity of the data. Perform DoS attack [9]. **Services:** Use on the same platform third party malicious applications in order to first infiltrate and then affect the network functions running on that platform. Use the application aware performance enhancers to enable DoS attacks by overloading the applications of the competitor with radio resources forcing them to end up. Fill the local caches with useless content in order to exhaust the regular network related resources. Spoof or modify the instructions transmitted by the orchestrator resulting in a disruption of the managers of both the virtualization infrastructure manager and the edge platform.	**Data:** Data should be first encrypted before being stored, processed, and transmitted. Proper authentication should be used. Puzzles allow to prevent DoS attack. **Services:** An application quality assurance framework is required to allow only trusted applications. Strong authentication and encryption mechanisms should be implemented and priority policies need to be set to allocate a dedicated cache for the AR application while being used.
④ Network between the edge and the cloud servers	Perform MITM, impersonation and DDoS attacks.	A strong encryption and authentication scheme should be implemented. DDoS can be avoided or at least be more complicated with the usage of puzzles.
⑤ Cloud server	Access the cloud service in order to view, tamper or redirect data to other rogue storage places. Perform cloud malware injection and side-channel attacks [10].	A strong encryption and authentication scheme should be implemented. The deployment of a hypervisor is needed to enable validation and integration of the virtual machines. Also, side-channel attacks can be further prevented by combining virtual firewall appliances. Intrusion detection and prevention mechanisms should be installed in order to monitor potential problems.

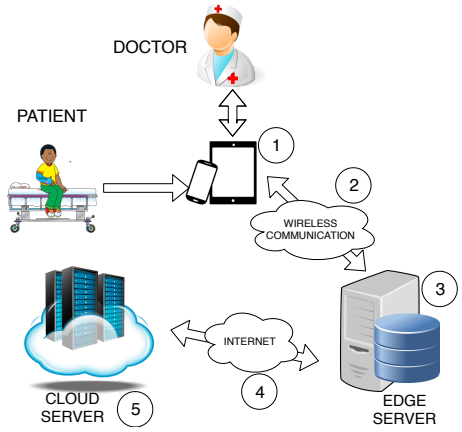

Fig. 1: Proposed ESSMAR architecture.

1) The doctor can access previous medical histories, reports of the patient, and is also able to see the obtained augmented results on the screen.
2) The doctor can see the augmented view of the data obtained from the patient's X-ray of MR scan on top of the affected area, which facilitates the medical check ups.

To support these features, there are two scenarios in the proposed ESSMAR system. The first scenario entails the patient identification and information retrieval processes. Here, the registered doctor uses the registered MAR device and logs into the system. Next, the doctor provides the identity details of the patient through the available user interface (UI) and requests the edge server for previous medical histories and reports. The edge server verifies the device and log-in credentials of the doctor and retrieves all the data related to the patient from the cloud and puts it in its storage. Then, the edge server sends the specific details requested by the doctor to the mobile device. The mobile device displays the information in the augmented view of the doctor.

The second scenario of the application consists of the situation in which the doctor gets assistance through the augmented view of the X-Ray or MR scan reports. In this setting, the doctor first places a marker on the parts of the body of the patient that need to be examined. Then, the device is hold on top of it in order to request the augmented data coming from X-ray or MR scan reports for these particular areas. After that, the edge server receives both this service request and the corresponding inputs of the device camera. Based on this request, the respective X-ray and/or MR scan reports are retrieved from the database by the edge server, which is then send back to the mobile device.

The mobile device simulates the augmented scene with X-ray/MR scan report. Because the data being handled in this application contains highly sensitive information, it is vital to prevent any security breach.

We analyzed the most common security threats to the ESSMAR architecture in Figure 1, and proposed viable prevention techniques in Table I.

III. Related Work

In literature, there are a lot of key agreement schemes proposed for different types of architectures, establishing a rich

set of security features like perfect forward secrecy, anonymity, untraceability, etc. The most common architecture is the one of a client server setting (e.g. schemes like [11]–[13]). Schemes for an architecture where three entities are involved that try to establish a common shared key are called tripartite schemes (e.g. schemes like [14]–[16]). Our scheme inherits parts of both. On the one hand, there are three entities involved, but on the other hand only common shared keys are established between client and edge. The main difference between our scheme and the ones in literature is that we consider that the involved entities are not too constrained with respect to computation and power. As a consequence, we use as much as possible standardized mechanisms like X509 certificates and TLS communication. We also utilize puzzles in order to avoid DDoS, which is certainly not done in case the client consists of a simple sensor or actuator. As we combine these mechanisms, we also provide in this paper a proof to show that the end result still offers the required security features.

IV. Registration and Authentication Phases for ESSMAR Architecture

In this section, we discuss how to organize the registration phase and the authentication phase for the ESSMAR architecture in a secure manner. The proposed mechanisms are relying on Elliptic Curve Cryptography (ECC) [17], providing the same level as protection than RSA based schemes but with smaller key sizes. Denote by E the elliptic curve over Z_p. A secret key pair consisting of private and public key is defined by (d, Q) with $Q = dG$ and G the generator of the curve of order q. The security of ECC is based on the computational hardness of both the Elliptic Curve Discrete Logarithm Problem (ECDLP) and Elliptic Curve Diffie Hellman Problem (ECDHP) [17]. Table II presents all the notations used in the schemes.

TABLE II: Notations used in the schemes

Notation	Description
N_s, N_c	Random number generated by server/client
k_{dos}	Puzzle difficulty level
X, Y	Solution of the puzzle/hash value
gcd	Greatest common divider
$Q_{Edge}, Q_{Device}, Q_{Admin}$	Public key of edge server/user device/admin device
$d_{Edge}, d_{Device}, d_{Admin}$	Private key of edge server/user device/admin device
CSR	Certificate signing request
H	Oneway hash function
$r_{Edge}, r_{Admin}, r_{User}$	Random integer value generated by edge/admin device/user device
$R_{Edge}, R_{Admin}, R_{User}$	Elliptic curve point generated by edge/admin device/user device
G	Generator point of curve of order q
K	Secret key for HMAC function
$Cert_{Admin}, Cert_{Device}$	Implicit certificate of admin device/registered device
e	Integer used to keep hash value of implicit certificate
s	Private key reconstruction value
K_{AES}	Session key
Nonce	Random cryptographic number used once
$E_Q(M)$	Public key encryption of M with public key Q

A. Registration Phase

This phase contains as most important tasks the registration of the entity's details in the Certification Authority (CA) and the reception of an implicit certificate to establish authentication in the network. This implicit certificate is based on the Elliptic Curve Qu Vanstone (ECQV) mechanism [18]. Moreover, the entities derive their private keys from these certificates. In the whole process, the edge server behaves as CA and is thus responsible for the storage of the details of all entities in the scheme and the generation of the implicit certificates of all the requesting entities. In the proposed scheme, the registration of the different types of users has to follow a predefined order as some category of users can only be registered using credentials of other types of users. The first type of users that require registration or the network administrators from the hospital or software company. They need to register in combination with the computers and devices linked to the corresponding edge server. After that, the registered admin uses his log-in credentials in one of the registered devices in order to start the registration of the employees and doctors, together with their devices. The last type of users that require registration on the edge server are the patients.

1) Network admin registration: The message flow for this scenario is illustrated in Figure 2. During the handshake, the edge server uses a puzzle-based challenge [19] to prevent denial of service (DoS) attacks. The edge server generates a puzzle for which the difficulty depends on the required level of security to protect against DoS attacks and expects the network admin to find the X value, which makes the first k_{dos} bits as 0 when the hash function is performed as shown. Both entities generate and share a secret key K using the Elliptic Curve Diffie-Hellman (ECDH) key agreement protocol. This K is used as the second key for the hash function included in the HMAC value calculation. This HMAC value ensures the integrity of the messages. Then the network admin requests the edge server to provide a certificate based on the Elliptic Curve (EC) point R_{Admin2} via the ECQV algorithm. After that, the edge server provides the certificate and requests for registration details of the network admin and his device. Upon receiving the certificate, the network admin device computes the private key and sends back the registration details. Finally, the network admin verifies his identity using facial recognition inputs. The message flow ends with the finished message.

2) Doctor/company worker device registration: The registered network admin of the hospital/company uses the log-in credentials of the doctor/company worker devices and registers their MAC (Media Access Control) addresses. The message flow of this scenario is similar to Figure 2. However, after obtaining the certificate and deriving the keys in the device, the user name, hashed password, facial recognition input is used of the network admin in order to feed the MAC address to the edge server. The edge server verifies the user name and password, facial recognition input of the admin, and stores the MAC addresses.

3) Doctor/company worker registration: Figure 3 illustrates the flow of messages for this type of doctor/company worker registration. In this phase, the network administrator generates first a user name and password for the corresponding doctor or employee. Then, the own log-in credentials of the admin are used in the registered devices of the doctors/company workers (see previous phase) in order to register the user name, password, other details, and photos in the edge server.

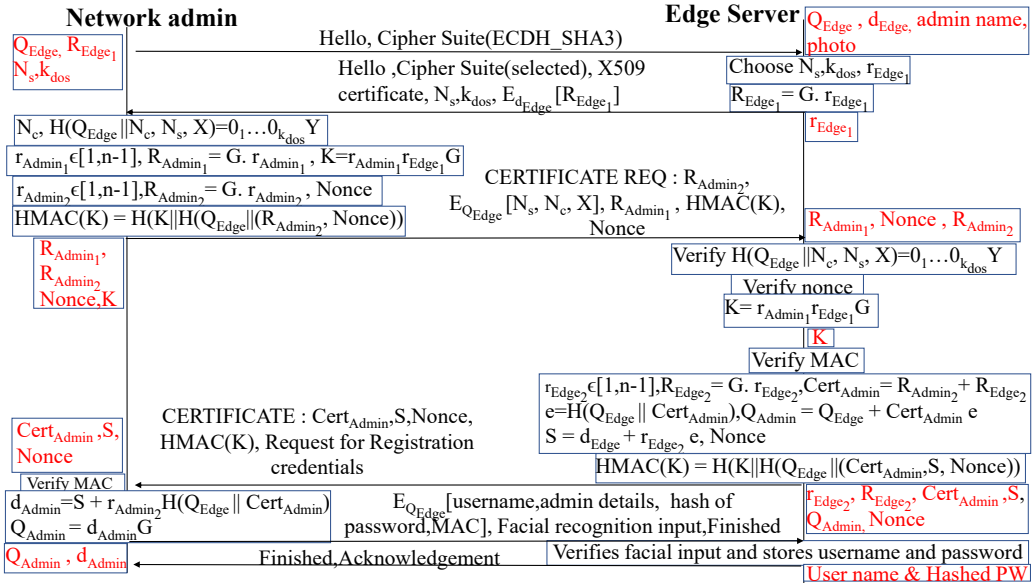

Fig. 2: Message flow for the registration phase of network admin.

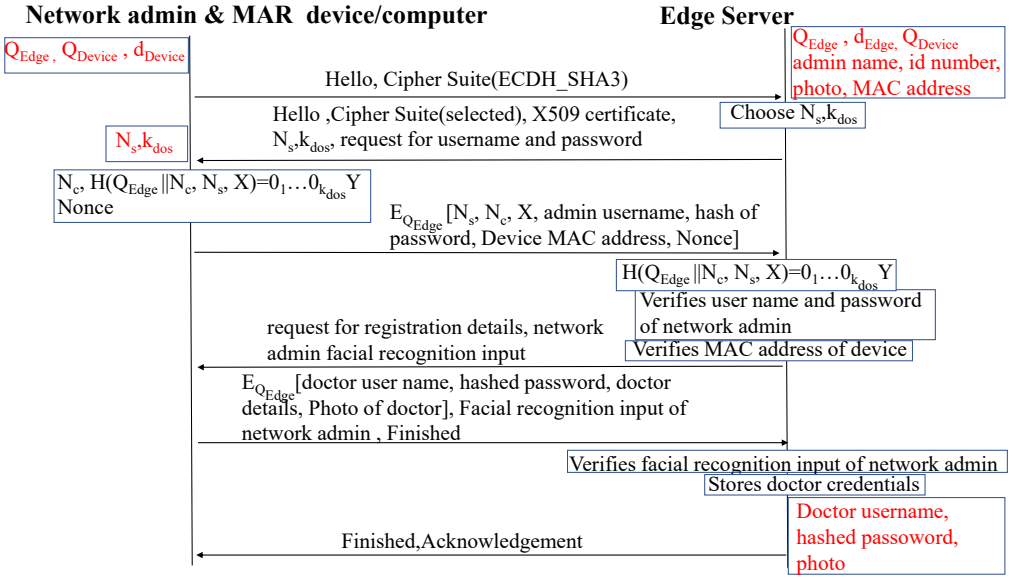

Fig. 3: Message flow for the registration phase of the doctor.

Note that at the server end, the MAC address of the registered devices and the log-in credentials of the network administrator are verified. Moreover, the edge server verifies the facial input of the network admin as a final step of verification and then securely stores the details.

4) Patient registration: In the final scenario of this phase, the patient becomes registered when coming to the hospital for examination. To this end, the network administrator requests required patient data like for instance personal data as name, phone number, photo, together with medical historical data and his/her consent to store and use the data (cf GDPR regulation). Next, the admin generates a user name for the patient and starts the registration process in a registered device. The message flow for this scenario is similar to Figure 3. As in the previous scenarios, after verifying the admin facial recognition input in the final step, the edge server stores the received information related to the patient. Next, a temporary password is sent by the edge server to the mobile device of the patient to be used at the first log-in. This code needs to be replaced for the following logins and can be chosen by the patient. Finally, the edge server sends an acknowledgement to confirm the end of the registration phase.

B. Authentication Phase

Figure 4 shows the flow of the messages for the authentication process. The primary goals of this phase are to do mutual authentication and to establish a session key securely.

Each of the registered users have a corresponding registered device, which is used during the communication with the edge server to enable multi-factor authentication. Because in the authentication phase, the role of the different entities (network admin, doctor/worker, patient) in the scheme is exactly the same, these entities are all referred to as user in Figure 4. As illustrated in the message flow, the edge server computes the symmetric Advanced Encryption Standard (AES) session key using K_{AES}=d_{Edge}.Q_{Device}.Nonce, which is equivalent to K_{AES}=d_{Edge}.d_{Device}.G.Nonce. Note that the nonce is added in the message to ensure that the key is only legitimate for that session. The edge server verifies the user's identity using their user name and password, MAC address of the registered device and facial recognition input. Finally, after successful verification, the edge server establishes a secret session key with the user.

V. Formal Security Verification

The proposed scheme for each scenario has been implemented in the formal verification tool AVISPA and validated by defining the required security goals. The implementation of the HLPSL codes follows the message flow proposed in the previous section.

A. Authentication Phase

Figure 4 illustrates the message flow in the authentication process. The registered user in Figure 4 is in the Automated Validation of Internet Security Protocols (AVISPA) simulation referred to as *Appuser*. The implemented High Level Protocol Specification Language (HLPSL) script and specifications of the role *Appuser* are shown in Figure 5, the role of *Edge* is shown in Figure 6, the roles of *session* and *environment* are shown in Figure 7.

In the implementation, *Logindetails* refers to the user name and hashed password. The *Appuser* is already registered with the edge server. As a consequence, the *logindetails* can be seen as the known secret message between *Appuser* and *Edge*. Further in the implementation, the elliptic curve point is calculated using *Ruser'= exp(G,Xuser')*, with *Xuser'* representing the r_{user} in Figure 4, and the accent ' indicates that the value is freshly generated.

The secrecy goal is defined as *sec_1*. The authentication goals are provided by *edge_Appuser_auth_login* and *edge_AppUser_auth_Facialin*. The *sec_1* defines the secrecy of the determined session key *Kaes* between the entities *Edge* and *Appuser*. The *edge_Appuser_auth_login* represents *logindetails* of the *Appuser* and is checked at the *Edge*. The *edge_AppUser_auth_Facialin* represents the facial recognition input of the user, which is also verified at the *Edge*.

The result, following from the simulation of the authentication process in the Onthefly ModelChecker (OFMC) backend is shown in Figure 8. This report demonstrates that the proposed scheme is secure.

B. Registration Phase

The HLPSL implementation for the registration phase of each scenario is carried out in the same way as in the authentication phase. The simulated result for the registration scheme for network admin is shown in Figure 9. Similar results have been achieved for all the other scenarios. From the simulation results, it is verified that the proposed scheme is secure against the MITM attack and the replay attack.

VI. Informal Security Verification

Now, we also discuss the informal security analysis of the proposed security schemes by focusing on the prevention of the most important, currently known security attacks for this type of schemes.

Protection against impersonation attack: In the scheme, the entities use X.509 certificates, log-in credentials, registered

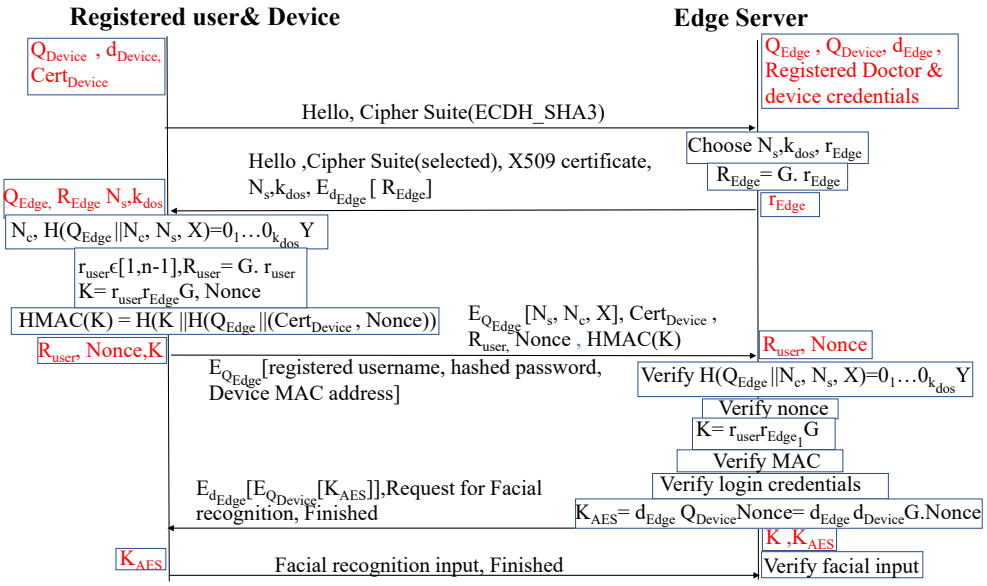

Fig. 4: Message flow for the authentication phase of registered user and device with edge server.

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

Fig. 5: HLPSL specification for role *Appuser*.

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

Fig. 6: HLPSL specification for role *Edge*.

device MAC addresses, and facial recognition inputs to verify their identity. Hence, the authentication of an entity depends on multiple factors in each scenario, which makes impersonation attacks impossible.

Protection against DoS attack: In order to avoid DoS attacks, a challenge-based mechanism [19] is used, which is the typical counter measure against DoS attacks. In both registration and authentication phases, a challenge is provided to initiate the communication. Depending on the severity of the currently present DoS, the challenge can adjust its difficulty level. Also, the answer to the puzzle is unique and can only be used in one particular instance as it is dependent of a

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
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()
```

Fig. 7: HLPSL specification for roles *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
```

Fig. 8: Protocol verification result using OFMC backend.

```
% 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
```

Fig. 9: Protocol verification result using OFMC backend.

random number, which is generated by the client. Therefore, our scheme mitigates the DoS attack.

Protection against password guessing attack: The user name and password are encrypted using the public key of the edge server. Consequently, only the edge server, who is in possession of the corresponding private key, is able to retrieve the password. During the communication, the hashed (and not the raw) passwords are exchanged. Due to the strength of the used hash functions, resisting pre-image and collision attacks, it is impossible for the attacker to retrieve the original

password. Finally, as also facial recognition is verified by the edge server, corresponding to the multi-factor authentication mechanism in the scheme, the attacker will never be successful in retrieving the data even in the worst case that the attacker would be able somehow to find the password of the user.

Protection against privileged insider attack: Here, the multi-factor authentication plays an important role. In case a privileged insider from the hospital is able to retrieve in some way the password of a user, it still needs to have at the same time access to the corresponding registered device and to provide the correct facial data of the user. This combination is a fairly impossible task for an attacker.

Protection against message tampering: The HMAC function in the scheme requires two keys. The first key is the public key of the edge server and the second key is a secret session key K generated and shared between the edge server and the corresponding communicating entity using the ECDH key exchange protocol. Due to the complexity in cracking the secret key, the usage of the HMAC function ensures the integrity of the message during the whole process. Hence, our scheme offers resistance for message tampering like changing, removing or adding data.

VII. Conclusion

The goal of this paper was to present a security architecture and protocol to enable the required security support to MAR applications in the healthcare sector. We first defined the ESSMAR architecture to be applied in a healthcare context. Then we discussed the potential security threats and attacks present in this architecture. Next, we provided viable solutions to encounter these identified problems. As a result, we defined for the different scenarios in a MAR application both a registration, authentication and key management scheme. In order to evaluate the security and safety of the proposed schemes, we provided a detailed formal and informal analysis of the schemes. This is resulted in sufficient proof of evidence on the fact that the proposed mechanisms offer the required security features to be implemented in real systems.

In future work, we plan to implement the proposed scheme and to thoroughly evaluate the performance with respect to latency and computational power. In addition, we will also compare the performance results of the proposed solution to the classical cloud based approach in order to validate the added value of our ESSMAR architecture.

Acknowledgment

This work has been performed under the framework of 6Genesis Flagship (grant 318927), 5GEAR, SecureConnect, Cost Action CA17124 and RESPONSE 5G (Grant No: 789658) projects. This research is funded by the Academy of Finland, Business Finland and the European Union.

References

[1] Z. Huang, P. Hui, C. Peylo, and D. Chatzopoulos, "Mobile Augmented Reality Survey: A Bottom-up Approach," arXiv preprint arXiv:1309.4413, 2013.

[2] L. Chen, T. W. Day, W. Tang, and N. W. John, "Recent Developments and Future Challenges in Medical Mixed Reality," in 2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR). IEEE, 2017, pp. 123–135.

[3] W. Zhang, B. Han, and P. Hui, "On The Networking Challenges of Mobile Augmented Reality," in Proceedings of the Workshop on Virtual Reality and Augmented Reality Network. ACM, 2017, pp. 24–29.

[4] D. Chatzopoulos, C. Bermejo, Z. Huang, and P. Hui, "Mobile Augmented Reality Survey: From Where We Are to Where We Go," IEEE Access, vol. 5, pp. 6917–6950, 2017.

[5] "Mobile-Edge Computing– Introductory Technical White Paper," European Telecommunications Standards Institute, Tech. Rep. Issue 1, 09 2014. [Online]. Available: https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobile-edge_Computing_-_Introductory_Technical_White_Paper_V1

[6] Y. Ai, M. Peng, and K. Zhang, "Edge Computing Technologies for Internet of Things: A Primer," Digital Communications and Networks, vol. 4, no. 2, pp. 77–86, 2018.

[7] R. Templeman, Z. Rahman, D. Crandall, and A. Kapadia, "Placeraider: Virtual Theft in Physical Spaces with Smartphones," arXiv preprint arXiv:1209.5982, 2012.

[8] R. McPherson, S. Jana, and V. Shmatikov, "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, 2015, pp. 743–753.

[9] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu, "Data Security and Privacy-Preserving in Edge Computing Paradigm: Survey and Open Issues," IEEE Access, vol. 6, pp. 18 209–18 237, 2018.

[10] P. Chouhan and R. Singh, "Security Attacks on Cloud Computing with Possible Solution," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 6, no. 1, 2016.

[11] A. M. A. Braeken, P. Kumar, "Efficient and provably secure key agreement for modern smart metering communications," Energies, vol. 11, no. 10, 2018.

[12] A. G. J. I. P. H. P. Kumar, A. Braeken, "Anonymous secure framework in connected smart home environments," IEEE Transactions on Information Forensics and Security, vol. 12, no. 4, pp. 968–979, 2017.

[13] A. T. A. Braeken, "Efficient anonymous user authentication on server without secure channel during registration," in 2nd International Conference on Cloud Computing Technologies and Applications (CloudTech). IEEE, 2016, pp. 13–20.

[14] N. K. K. C. X. Jia, D. He, "Authenticated key agreement scheme for fog-driven iot healthcare system," Wireless Networks, vol. 25, no. 8, 2018.

[15] T. C. T. L. C.L. Liu, W.J. Tsai, "Ephemeral-secret-leakage secure id-based three-party authenticated key agreement protocol for mobile distributed computing environments," Symmetry, vol. 10, no. 4, 2018.

[16] K. S. S. Patonico, A. Braeken, "Identity-based and anonymous key agreement protocol for fog computing resistant in the canetti-krawczyk security model," Wireless Networks, pp. 1–13, 2019.

[17] V. S. Hankerson D., Menezes A.J., "Guide to Elliptic Curve Cryptography," in Springer-Verlag Berlin, Heidelberg, 2003.

[18] C. Research, "SEC4: Elliptic Curve Qu-Vanstone implicit certificate scheme, Standards for Efficient Cryptography Group. Version 1.0." 2013.

[19] T. Aura, P. Nikander, and J. Leiwo, "DoS-Resistant Authentication with Client Puzzles," in International workshop on security protocols. Springer, 2000, pp. 170–177.