An Efficient Authentication Scheme for Internet of Things

Jaafer Al-Saraireh, Haya Joudeh[1]

[1]King Hussein School of Computing Sciences, Princess Sumaya University for Technology, Jordan

***Abstract*:** The Internet of Things (IoT) is increasingly affecting human lives in multiple profound ways. “Things” have the ability to communicate, generate, transmit and store data over the network connection. During each communication between “Things”, the data transmitted is potentially vulnerable to malicious attacks, loss, distortions and interruption which impair functionality, system efficiency and user satisfaction. Therefore, communications between things need to be authenticated, authorized, secured and ensured to have high privacy by applying a strong authentication protocol.

The aim of this research is to enhance the authentication protocol, starting by reducing the heavy use of storage in “Things”, and eliminating unnecessary messages during authentication steps. This research represents a security performance analysis and enhancement authentication for the IoT. The results indicate that the enhanced protocol has a positive effect on minimizing packet length and time performance compared with the other two protocols used for comparative purposes, with 33% increased the proposed protocol performance.

***Keywords*:** Internet of Things, authentication, wireless sensor, integrity, confidentiality**.** HRA.

1. Introduction

IoT is a new concept that encompasses the general future of the internet, particularly as it relates to digital technologies and e-commerce. It fundamentally links objects by assigning them a unique object-identification connected to the network [1]. The IoT is a technological revolution that is essential for the future of computing and communications, relying on dynamic technical inventions in a number of fundamental fields such as wireless sensors and nanotechnology [2]. IoT’s application areas include a wide variation of industries, including healthcare, transportation, smart home systems, public services management and oil fields [3].

IoT connects billions of devices with each other, thus its most important function is the authentication process to determine correct user identity and information integrity [4]. The authentication of smart devices consists of Medium Access Control (MAC) addresses, Quick Response (QR) codes and IPv4/IPv6 [5]. In MAC each device has a unique 48 bit number called MAC address. The access points authenticate users using these numbers by saving the approved ones in a table. The access point may keep a bad address in its table as authenticated while it may prevent good ones. Therefore, Internet Protocol (IPv4/IPv6) is the most traditional way to assign unique address for each device under the internet and use them as identifiers in data communication.

Before explaining authentication, protocols or solutions implemented nowadays, one has to consider that the IoT environment is divided into three main tiers – application, network/transport and devices/gateways – responsible for different security considerations. When implementing IoT authentication solutions in business some specifications are required, namely to: (1) Design the authentication and authorization schemes based on system-level threat models. (2) Plan for the introduction of IPv6. (3) Consider design updates to the used Public Key Infrastructure (PKI) environment to support provisioning of certificates to IoT devices in the organization. (4) Establish a plan for sharing IoT-related data with device manufacturers [2, 3 and 5].

In the IoT network, secure and solid communication should be set up between entities, objects and servers by a certain procedure that implements an authentication protocol for all communication partners to ensure solid communication in high layers, such as application layer. The most challenging in IoT are security and privacy aspects, which are fundamentally addressed in computer networks by the authentication process. Authentication must be balanced against user convenience within the context of system efficiency, current-oriented computing devices, power resources, security scalability and storage features in order to insure the best quality of services relative to safety for network users. [1, 2, and 3]

This research paper addresses the problem of authentication mechanisms during user authentication when resources or services are accessed in IoT networks. The aim of this research is to enhance the authentication protocol, starting by reducing the heavy use of storage in “Things”, and eliminating unnecessary messages during authentication steps, taking into consideration the network security analysis

2. Background and Related Work

Extensive research has been conducted on authentication in IoT. The importance of authentication process means to secure communication between objects, which has been demonstrated in major studies [6, 7, 8 and 9]. Liu, et al. [6] developed a procedure to authenticate legitimate users to access the IoT network with secure and solid communication while connecting with things in the high layer. Primarily the authors focused on the main sub-tasks including key establishments, key switching and consultation. They used a key establishment procedure based on the elliptic curve cryptography (ECC), and proposed the architecture of authentication by involving the use of home registration authority (HRA) whereby all users’ devices are registered, and a registration authority (RA) whereby all things or objects are also pre-registered.

All calculations were done on both RA and HRA, both of which are instrumental in registering users and things before network deployment, and the RA initiates a secret key before joining the IoT network [6]. A critical analysis of the protocol reveals the following problems: (1) The proposed protocol lacks mutual authentication security service requirements, such as session key establishment [6]. The RA did not send a replay message to user after checking the hash values. (2) No timestamps were used, so the proposed protocol suffers from replay attacks and device attacks. (3) Excess messages were used during login and key establishment phase.

Ndibanje et al. [7] attempted to improve on the work of Liu et al., [6] with a protocol to fulfill the missing security requirements and weaknesses, including message exchange cost and the nature of IoT devices. The security assessment is not strong enough for such a protocol and the important point is ensuring the mutual authentication during the communication. This enhancement started by formatting Liu et al. [6] protocol by separating it into the main composite steps of protocol standards: registration phase (offline or online), login and verification phase. Before execution of the scheme they noted the following assumptions were violated: (1) All clients and server devices are assumed to be trusted during registration phase. (2) Once the registration phase is done, all devices have to authenticate themselves. (3) All users will communicate only with RA to save the energy and reduce the computation cost in sensor nodes. (4) RA must assign the secret key before joining the IoT network.

The registration phase, assumed that each user must be registered into the HRA server. The goal of this phase is to negotiate and compute different secret parameters in the login and authentication phases between the user and gateway node (RA). The authentication phase has the following phases Login phase and Verification Phase: In login Phase: the user sends the login request message over a public channel to RA, after the needed calculation of identification for paired RA (Dra) and secret encrypted value for RA (Bra) parameters. This phase consumes a lot of energy to achieve a mutual authentication; that is why they limit the authentication phase to the RA. Verification phase: this phase is required to mutually authenticate the user by the RA and vice-versa, while the user needs to obtain access specific object or service in the IoT environment.

An earlier study by Tang and Wu introduced an authentication scheme tailored to low power devices, which mainly focused on generating delegation passcode for mobile station authentication using ECC-based trust delegation mechanism. Their proposed scheme only required two messages for a mobile station to authenticate itself to a visitor's location register (VLR), using trust delegation method to enable a VLR to authenticate a mobile station after home location register (HLR) registration finished. The security problems in this scheme render it vulnerable to be hacked by any attacks related to impersonation of VLR, redirection attacks and false base station attacks, because it does not require a particular VLR to forward the service request; in other words, a false VLR could perform this, and this would not affect the security requirements.

On the other hand, Lu et al. [11] enhanced the authentication scheme proposed by [10] by increasing the strength of the security of the scheme by authenticating the identity of visited location register. Consequently, any adversary cannot obtain the communication key between a mobile user and a service provider, or prevent them from establishing this key to obtain essential authentication procedures for preventing illegitimate, unauthorized or insecure devices from accessing the network.

Ye et al. [12] proposed approach focused on simple-efficient mutual authentication protocol and secure key establishment based on ECC, which has much lower storage and communication overheads. Their proposal for authentication is separated into two parts: (1) Authentication: authentication between user and terminal nodes to ensure that only legitimate users can access the network. (2) Key establishment: session keys created between the user and nodes for secure communication.

Gope and Hwang [13] proposed a new technique to increase efficiency relative to existing authentication solutions to check a variety of security requirements, such as user anonymity, ensuring forward secrecy and how to deal with the stolen smart violation. In addition, their protocol is based on using computations operation on servers.

Park and Kang [14] presented an inter-device authentication and session key establishment for the devices. In their proposed methodology, the user is involved in establishing the session key, contrary to what existed in the wireless environment. Therefore, each sensor or user device that is part of the communication parties will be involved in the generation of session keys. They proposed a methodology to enhance the performance of computing the session keys by devices, starting from this point, using multiple parameters for the authentication phase, such as authentication initiator I, responder to the message R, random numbers, shared secret key, and the shared session key known by the responder and authenticator. The proposed methodology assumes that the shared secret key and the encryption function are saved securely in the user device, based on the parameters and saved values.

Low power consumption Machine Learning (ML) techniques for detecting IoT botnet attacks using Random forest as ML-based detection method and describing IoT common attacks with its countermeasures was proposed by [15, 16 and 17].

A deep-learning based classifier that learns hardware imperfections of low-power radios that are challenging to emulate, even for high-power adversaries is presented by [18].

A new secure remote user mutual authentication protocol based on transitory identities and multi-factor authentication for IoT smart building environment was proposed by [19].

A blockchain system framework for IoT identity authentication is proposed by [20 and 21], which implements the authentication between devices and cloud servers, IoT base stations as well as devices, and then analyzes its feasibility

Finally, a comprehensive survey on IoT authentication protocols by [4, 22, 23 and 24] were presented. The authors compared the authentication protocols based on different terms, titled with security and privacy for the IoT, goals, network models, and communication cost.

As a result of the review, the main goal in this research is to enhance an authentication protocol to secure the communication between nodes and prevent unauthorized behavior using a new authentication approach while preserving the access of required services without the manifestation of any obstacles.

3. Proposed Technique

3.1 Overview of the Proposed Solution

Several major works on authentication protocol noted the problem of communication between the user and things and limitations to access of needed services in IoT networks, and the vulnerability of the authentication process, necessitating security to protect against unauthorized access. This work builds on previous protocols, particularly those of Liu et al [6] and Ndibanje et al. [7], to enhance the authentication scheme during the communication processing procedure.

The main concern of Liu et al. [6] was to enable things to communicate directly by sending request packets to initiate the authentication process between the RA, HRA and user, and finally to generate the session key by the RA using elliptic curve cryptography key establishment, then sending it to things and user, so the user can obtain access to the required things. The process of authentication is repeated for each access between users and other things and services in the same network. From this side, the proposed protocol comes to minimize the extraneous involvement of things for every connection to authenticate the same user, eliminating unnecessary messages during the authentication process, taking into consideration the security of services.

3.2 Architecture of the Proposed Technique

The proposed solution consists of two main phases, the registration and authentication phases. Before explaining each phase, the following assumptions should be noted:

1. Before registration, all users and things are supposed to be trusted to HRA and RA.
2. After finishing the registration phase all users, things, RA and the HRA server need to verify the authenticity of each other's identities to safely communicate with one another.
3. The user will only communicate with RA related to the things to achieve the main goal of minimizing the possibility of applying arithmetic operations to things, and these operations will be applied in RA.
4. Each thing will be pre-registered with its RA.
5. Each user will be pre-registered to its HRA.

Figure 1 shows an abstract architecture for our proposed scheme.

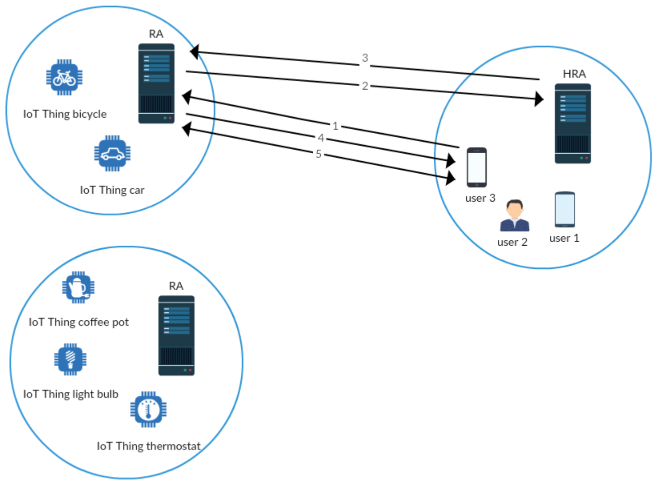


Figure 1. Abstract architecture for proposed scheme

The authentication process goes through the following steps:

1. The user sends an "authentication request" to RA.
2. Having received the "authentication request", RA sends the "authentication data request" to the user's HRA to generate the authentication vectors for the first time; otherwise, the RA selects unused AV.
3. HRA sends "authentication data response" in case the RA requests new authentication vectors.
4. RA sends "authentication response" to the user with the token parameter AUTN to calculate the response parameter RES and send it back to RA.
5. RA checks if the received response equals the stored one. If yes, then the user is authenticated; otherwise the process is rejected.

To better describe the proposed authentication scheme, some relevant terms involved in the authentication process are defined in Table 1, while the following sections explain the steps and goals of each phase, leading to establishing the session key between two entities using ECC.

Table 1. Relevant terms for proposed scheme

Symbol	Description
PW	Password of IDu
IDu	User identity
Nu	Generated Nonce by HRA to User
Nra	Generated Nonce for the gateway
IDhra	User ID of the HRA
h()	A public one-way hash function
S	The RA's private key
IDt	Thing identity
E_p	An elliptic curve group defined on p with a large order
G	Elliptic curve generator points on E
MsgReg	Message registration request
Tu	Timestamp of message submitted by user
$\oplus$	XOR operation
||	Concatenation operation

The used one-way hash function described as a mathematical method takes a different variable-length input as a string and converts it into a fixed-length output. This function is designed to find a one-way hash value whereby it is hard to find the original value again in the reverse process, which is why it is called one-way hash; its goal is to generate different hash values to a different given value, taking into consideration the difficulty of generating the same hash value for two different strings.

3.3 Registration/Initialization Phase

The goal of the registration/initialization phase is to register each user in the HRA server. During this phase some necessary information like user and thing IDs, keys such as secret key, random number (RN) and ephemeral number for each object is pre-distributed or computed to be ready for the login/authentication phase. The following steps are required from the involved users and objects to complete the registration phase, as depicted in Figure 2.

First, the necessary computations needed by user:

1. Enter the password with IDu.
2. Generate a random number Ru for each user.
3. Compute MsgReg = h(Ru $\oplus$ PW)||IDu,Tu (in addition to assign the timestamp).
4. Submit the MsgReg message to HRA.
   After HRA is received:
5. Check the timestamp Tu is valid.
6. Check New IDu (New) exists in the existing IDs table in database.
7. If No, Store the new user in database with Assign Nonce and random starting point of sequence.
8. Generate a secret number Rg.
9. Compute private key Su and public key Pu using ECC. (ECC is described in more detail in section 3.4.1).
10. Then compute:
    a. Hra = h(IDhra $\oplus$ Rg)
    b. Sra = Ekhra (Hra).

c. $Ura = Hra^{h(IDu||h(Ru\oplus PW))}$

11. Compute MsgRep = {Sra, Ura, Pu, Su, IDhra, Thra,KNu}
12. Submit the MsgRep to user.
After user received:
13. Check the timestamp Thra is valid.
14. Once the user received the MsgRep, store MsgRep in the tag or smart card used during access to the IoT network.

3.4 Login/Authentication Phase

This section describes the authentication phase (as shown in Figure 3). This phase aims to authenticate the user who wants to access object or thing in the visited location area by performing mutual authentication between the user and its RA, and vice-versa. The user does not send the authentication request to the thing directly. Once the user logs into the machine and inputs their user ID and password or by inserting the smart card or tag.

As shown in Figure 4, the user enters their credentials and the authenticity checking begins. The authentication methodology is explained in the following points:

1. After user login, in the local machine compute
$$Uhra' = Hra^{h(IDu||h(Ru\oplus PW))}$$
2. Check if the Uhra' is similar to the saved Uhra in the local machine; if yes, go to the next step, otherwise reject the login request.
3. Send to RA Login request message, which contains Shra that includes the information about its HRA server, computed Uhra', the current timestamp and the requested RA's ID.
$$LgnRegMsg = \{S_{hra},U_{hra},T_u,IDra,IDu\}$$
After RA received the login request message:
4. Check if Tu is valid.
5. Check if IDu (New) = IDs existing in the database and check if this user has unused authentication vectors; if no, then go step 6; otherwise go to step 9.
6. Send an authentication data request to user's HRA to generate the needed authentication vectors (AV). (The user's HRA identity is known by extracting it from the login request after decrypting the received Sra).
$$AuthDataReq = EK\{IDu,T_{ra}\}$$
7. Upon HRA receipt of request from the RA, the HRA checks if the user belongs to this HRA. If no, reject the login request; otherwise, generate an ordered list of n authentication vectors for the specified user for the requested RA (a procedure to distribute authentication information from HRA to RA, then send the authentication data response to RA.
$$AuthDataResp = AV\{1..n\}$$
8. After the RA received the authentication data response, store the new AVs in the database.
9. Then RA will select the next unused AV (i) from the ordered list of authentication vectors stored in the RA database. The particular using of authentication vectors per nodes is based on first-in/first-out (FIFO).

After receiving the AVs from the HRA and selecting the unused AV(i), the RA and user perform the mutual authentication operations as follows:

10. RA sends user authentication request containing RAND(i) || AUTN(i) to the user device.
$$AuthReq = RAND(i) \,||\, AUTN(i)$$
11. The user checks whether AUTN can be accepted and, if so, compute a response RES which is sent back to the RA.
$$AuthResp = RES(i)$$
12. Upon Receipt of the authentication response from the user, the RA compares the received RES(i) with XRES(i); if it matches then go to the next step, otherwise authentication failed.
13. Now the RA believes that the user is a legitimate one who can access required data; the RA sends the acknowledgment to inform the user about the creation of session key.

3.4.1 Elliptic Curve Cryptography (ECC)

Elliptical curve cryptography (ECC) is a public key encryption strategy based on elliptic curve theorem that can be used to create smaller, faster shorter keys as robust as long keys for RSA, and more efficient. Moreover, the advantages of using ECC include that it is low on CPU consumption and memory usage. The ECC creates the keys based on the elliptic curve equation instead of traditional methods of generation used by other techniques. According to Garg [25], ECC can yield a level of security with a 164-bit key for which other systems would require a 1,024-bit key.

Because ECC helps to establish equivalent security with lower computing power and battery resource usage, it is becoming widely used for mobile applications. The main steps that needed for the ECC technique depend on choosing a generator point G, $G \in E_p(\mathrm{a,b})$ to form the curve by the following equation

$$y^2 \;=\; x^3 + ax + b$$

Such that the smallest value of x such that xG = O is a large prime number. Then calculate all points/groups for this curve and make the elliptic group $E_p(\mathrm{a,b})$ and the generator G public. For each user select a private key randomly Sa < x, and then compute the public key Pa = Sa*G.

3.4.2 Distribute Authentication Information from HRA to RA

The purpose of this section is to provide the RA with a list of fresh authentication vectors (AV) from the user's HRA server to perform the n time of authentications. The RA invokes the procedures by requesting the authentication data to HRA server for the accessed user. Upon receipt the authentication data request from RA, the HRA starts computing the required parameters to produce the required number of authentication vectors. Figure 5 shows the generation of authentication vectors based on Users' key nonce and Sequence number parameters determined by the HRA server for each user.

Thus, the single authentication vector will computed as:
$$AUTN = SQN \oplus IK \,||\, AMF \,||\, MAC$$
$$AV = RAND \,||\, XRES \,||\, CK \,||\, IK \,||\, AUTN$$
Figure 5 demonstrated that each AV consists of a Random number (RAND), an expected response (XRES), Cipher key (CK), Integrity key (IK) and an authentication token (AUTN). The HRA starts computing the new sequence number (SQN) for each user, keeping track of a counter in the database and unpredictable random number. An Authentication Management Field (AMF) is included in the AUTN parameter of each AV. AMF indicates which algorithm and key are used to generate a particular AV when different algorithms and keys might be used, and it is responsible for setting threshold

values for key lifetimes. The HRA computes the following values:

1. Message Authentication Code (MAC) = fcn1(KNu||SQN||RAND||AMF), where fcn1 is a message authentication function.
2. Expected response, XRES = fcn2 (KNu ||RAND), fcn2 is a truncated message authentication function.
3. A Cipher Key, CK = fcn3 (KNu ||RAND), using a key generating function fcn3..
4. An Integrity Key, IK = fcn4 (KNu ||RAND), fcn4 is a key generating function.

The length of authentication parameters CK, IK, RAND and Knu all are 128 bit long, expandable up to 256 bits if needed in the future.

3.4.3 User Verification Function in User Device

The purpose of this step is to authenticate the user. During the authentication, the user checks the validity of the selected authentication vector used. The RA server invokes the procedure by selecting a new (unused) AV from the database, then the RA sends the RAND and AUTN of the selected authentication vector. Figure 6 shows the user authentication function in the user device upon receipt of the authentication request.

Upon receipt of the RAND and AUTN, the user computes IK = fcn4(RAND) and retrieves the SQN from the calculated IK; SQN = (SQN $\oplus$ IK) $\oplus$ IK. Next, the user computes XMAC = fcn1(SQN ||RAND||AMF) and compares it with the received MAC included in AUTN. If they match then mutual authentication succeeded and acknowledgment is sent to RA to begin establishing the session key, otherwise the user sends the failed message. In this case, RA will recognize that the user is not legitimately authorized to access the visited location.

To establishing Session Key (SEK), upon the Acknowledgment receipt by the user to the RA server for authentication succeeded, the RA will select the CK of the selected authentication vector and the user will compute the cipher key (CK) from the proceeds. CK = fcn3(RAND).

After calculation of the CK, the session key is computed to perform subsequent operations during a session as: SEK = h(CK)

The Expected Response Creation Function (Fcn2) is responsible to create the XRES parameter. It depends on the deliverable inputs such as the secret key of each user (KNu) and RAND. fcn2: (K; RAND) $\rightarrow$ RES (or XRES)

The MAC Code Creation Function (fcn1)is responsible to create the MAC parameter. It depends on the deliverable inputs such as the secret key of each user (KNu), RAND, SQN and AMF that describe the algorithm and key used to generate a particular AV. fcn1: (K; RAND, SQN, AMF) $\rightarrow$ MAC (or XMAC)

The Cipher Key Creation Function (fcn3) is responsible to create the CK parameter. It depends on the deliverable inputs such as the secret key of each user (KNu) and RAND.fcn3: (K; RAND) $\rightarrow$ CK

The Integrity Key Creation Function (fcn4) is responsible to create the IK parameter. It depends on the deliverable inputs such as the secret key of each user (KNu) and RAND.fcn4: (K; RAND) $\rightarrow$ Ik

4. Simulation Environment

To evaluate the performance of simulation protocols in the network domain, it is necessary for researchers and programmers to use a solid simulation tool. For the proposed design, Omnet++ simulator was used. The reason for choosing this tool is that it is open source and commonly used by international researchers in the network domain.

The operating system used for all experiments was Microsoft Windows 10 Enterprise, with system type x64 bit. An Intel® core i7 processor machine with eight GB RAM was used.

The topology for the proposed scheme simulation is explained in this section. All user devices, things and servers were connected through wireless connection. To run the simulation OMNeT++ offers three files to configure the scenario, described below.

(1) NED File: Firstly, the structure of simulation model is described using network description (NED) language. In NED files simple modules are declared, connected and assembled into compound modules. In addition, the necessary component type is channels. Figure 7 shows the proposed scheme's NED file. The NED file describing the proposed simulation topology is visualized in Figure 8.

(2) Programming Proposed Protocol and Data Store Diagram File: The simple modules of a created model contain algorithms using C++ classes and functions. The simulation of proposed protocol uses C++ code to define the exact scenario. Figure 9 specifies a snapshot of one of the client device modules that shows how to handle the received messages, generating new messages and exchanging it along other devices.

On the other hand, the diagram of the data store used for each server is built using XML files, aiming to store the related data such as storing the user devices' information in HRA server. Figures 10 and 11 show the HRA and RA servers' data models.

(3) Simulation Control Configuration (omnetpp.ini) File: This file defines the values of the variables and the name of the running network in case of multiple network topologies being specified in the same directory [26]. In the introduced simulation, there are multiple networks defined in the same directory; one for our proposed network and the others for the protocols proposed by Liu et al. [6] and Ndibanje et al., [7]. Figure 12 shows a snapshot of the omnetpp ini file.

The variables used in the proposed simulations consist of the following, which are configurable depending on the running tested protocol:

1. The type of mobility attached for each user device. Here "Random Waypoint mobility" model (RandomWPMobility) is used. Due to the nature of the IoT environment, where each user is randomly moved, this model sets a new random position satisfying the constraint area.
2. The number of user devices in the simulation was 10.
3. Number of things located in visited location area (VLR); 10 objects located per VLR.
4. The features of wireless protocol are specified here too, such as transmitter communication rate (500m), display Communication Range set to true, radio channel bandwidth (2MHz), the networks application used (TCPBasicApp) and some variables related to the animations display of channels.
5. The client devices' bit rate was set to 1Mbps.

5. Results Analysis and Discussion

The results of both performance and security analyses appear indicate that the proposed protocol fulfills the goals of this research regarding the performance and the demands of the network security services in the IoT environment, such as confidentiality, integrity, authenticity and obtaining better efficiency at a lower communication cost. The following sections explain in detail the performance and security analysis and compare it with two different algorithms.

5.1 Performance Analysis

The performance analysis of the proposed protocol is based on the communication time while authenticating users to access any things or services in the IoT network, and the size of each packet sent between the involved parties in comparison with existing works by Liu et al. [6] and Ndibanje, et al. [7] (hereinafter cited in the remainder of this paper as the Liu and Ndibanje protocols, respectively). The analysis is divided into two parts: performance analysis for the registration phase and login/authenticate phase.

5.1.1 Registration Phase

In this phase, the metrics used in the performance evaluation of computational and communication cost compared with the Liu and Ndibanje protocols are: (1) Random number generator function, (2) Cryptosystem used (ECC), (3) XOR operation, (4) Time to perform one-way hash function, (5) Executing multiplication on ECC cryptography, and (6) The communication time needed to register each user.

Table 2 explains the performance metrics and execution time of the registration phases by the Liu, Ndibanje and proposed protocols. It displays the performance analysis of the statistical outputs of execution the main functions and operations during the registration phase. It also shows the execution time for registering the user into the HRA server, with the proposed protocol's performance in terms of computation cost, requiring four times for RAND generator and one time for the cryptosystem, compared to two times for RAND and three times for cryptosystem for Liu and three times for RAND and one time for cryptosystem for Ndibanje in registration phase.

Regarding other parameters, three times are needed to perform a hash function in Liu and proposed protocols and two times needed in the Ndibanje protocol. For the ECC multiplication operation metric, the Ndibanje protocol does not use it, but the Liu protocol needs three times and the proposed protocol needs one time for multiplication operation. Finally, in the case of XOR operation, two times are needed for the proposed protocol and Ndibanje protocol while the Liu protocol does not use it. Furthermore, the time duration needed to register each user into HRA server differed between mentioned protocols. The Ndibanje protocol needs 6096 milliseconds to register the user into the HRA server and at least for one VLR's registration authority server, while the Liu protocol needs 4276 milliseconds for the same scenario, but the proposed protocol needs to register the user into HRA server only, and it will take 2057 milliseconds. Table 3 shows the packet length for the whole registration process while the user needs to register to one of the available HRA servers and at least one VLR registration authority server in the Liu and Ndibanje protocols.

Table 2. Performance analysis for registration phase

Algorithm used	Random number generator	One-way hash function	Cryptography	XOR operation	ECC multiplication operation	Total registration time
	T. R.	T. R.	T. R.	T. R.	T. R.	Avg. Duration
Liu protocol	2	3	3	0	3	3276 ms
Ndibanje protocol	3	2	1	2	0	4096 ms
Proposed protocol	4	3	1	2	1	2057 ms

Note: **T. R.**: times required to achieve the selected function, **Avg. Duration:** average duration of achieving the whole registration process for 10 users in millisecond.

Table 3. Packet length in registration process

Algorithm used	Packet length
Liu protocol	N/A the exact data
Ndibanje protocol	Approximate total 1472 bits
Proposed protocol	Approximate total 1144 bits

As shown in Table 3, packets' content length varies depending on the content of the request and response packets for both the proposed and Ndibanje protocols, whereby the Liu protocol lacks the exact data exchanged between entities in the registration phase. Figure 13 shows the OMNeT++ statistical tool that used to extract the communication time information between the parties.

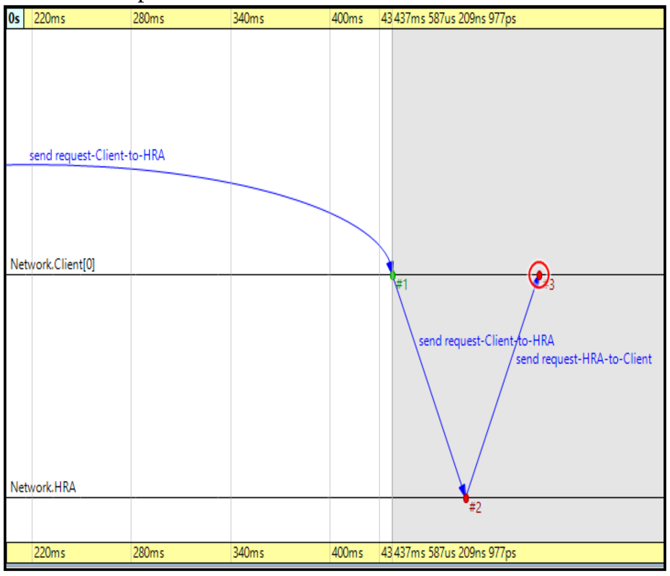

0s
220ms
280ms
340ms
400ms
43 437ms 587us 209ns 977ps
send request-Client-to-HRA
Network.Client[0]
#1
#3
send request-Client-to-HRA
send request-HRA-to-Client
Network.HRA
#2
220ms
280ms
340ms
400ms
43 437ms 587us 209ns 977ps

Figure 13. Time cost of registration phase in proposed protocol

After calculating the time taken for each implemented protocol and getting the average time to finish the registration phase for 10 users, Figure 14 illustrates the improvements in percentages achieved by the proposed protocol relative to the Liu and Ndibanje protocols. And it shows how the proposed protocol finished the registration phase ends in less time than the other protocols.

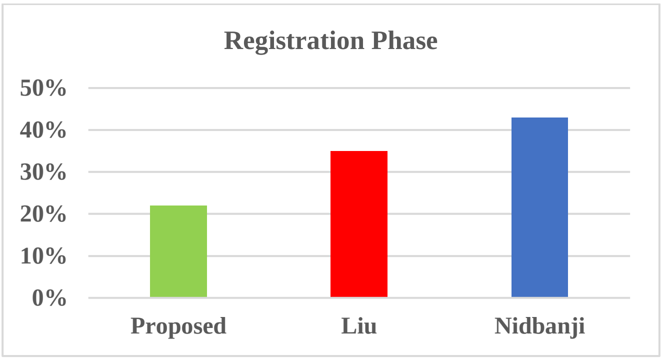


Figure 14. Communication cost of registration phase

5.1.2 Authentication Phase

In this phase, the metrics used in the performance evaluation of computational and communication cost in comparison with the Liu and Ndibanje protocols are: (1) Random number generator function, (2) Cryptosystem used (ECC), (3) XOR operation, (4) Time to perform one-way hash function, (5) Executing multiplication on ECC cryptography, and (6) The communication time needed to verify and authenticate user into VLR.

Table 4 explains the performance metrics and execution time of the authentication phase for the Liu, Ndibanje and proposed protocols. The performance analysis shows the statistical output of execution the main functions and operations during the authentication phase, where the proposed protocol in terms of computation cost does not use the random number generator and one time cryptosystem.

In Liu and Ndibanje, one time of random number generator and four times cryptosystem, one time of random number generator and also two time cryptosystem are required, respectively, in their authentication phases. Regarding other parameters, two times are needed to perform a hash function in the Liu and Ndibanje protocols whereas one time is needed in the proposed protocol.

For the ECC multiplication operation metric, the Liu protocol needs two times. However, Ndibanje and the proposed protocols do not use it for the authentication phase. In XOR operation, one time is needed for the proposed protocol and Ndibanje protocol while the Liu protocol does not use it. Finally, the proposed protocol needs to generate the authentication vectors one time for the first round of accessing the new visited location area, whereas the other protocols do not use it.

On the other hand, the time duration needed to authenticate the user into VLR differed among the protocols. For the first round, the proposed protocol takes on average 3885 milliseconds. On the other hand, after any round the Ndibanje protocol takes 453 milliseconds to check the authenticity of the user to access the VLR while the Liu protocol takes 379 milliseconds for the same scenario, but the proposed protocol takes only 311 milliseconds, as shown in Table 5.

Table 6 also shows the packet length for the whole authentication process to check the authenticity of the user to access the needed VLR in the proposed, Liu and Ndibanje protocols. It can be seen that message sizes vary depending on the content of the request or response packets for the implemented protocols. As shown in the table, the content of packets in the proposed protocol depends on the visit round of accessing the needed VLR. In the first round of authenticating a user into a new VLR, the total packets size will be approximately 8856 bits (including the first initiation for authentication vectors), but after the first round, the packets will be minimized to approximately 1136 bits.

Table 4. Performance analysis for authentication phase

Algorithm used	Random number generator	One-way hash function	Cryptography	XOR operation	ECC multiplication operation	AVs generator
	T. R.	T. R.	T. R.	T. R.	T. R.	T. R.
Liu protocol	1	2	4	0	2	0
Ndibanje protocol	1	2	1	1	0	0
Proposed protocol	0	1	1	1	0	1 for first round only
Note: **T. R.**: times required to achieve the selected function,						

Table 5. Average authentication time delay

Protocol-algorithm used	Authentication phase	Protocol-algorithm used
	Average Duration	Average Duration
Liu protocol	-	453ms
Ndibanje protocol	-	379ms
Proposed protocol	3885ms	311ms

Table 6. Packet length in authentication process

Algorithm used	Packet length
Liu protocol	Approximate total 1720 bits
Ndibanje protocol	Approximate total 1440 bits
Proposed protocol	Approximate total 8856 bit (including first initiation for authentication vector)

In the Liu protocol the packet size will be approximately 1720 bits and in the Ndibanje protocol the packet size will be approximately 1440 bits. Figure 15 shows the time delay of authentication phase while authenticating user access to a thing in a specific VLR. It reveals that the cost time of the communication to authenticate the user before accessing the thing or services over IoT network in the proposed protocol is lower than in the other protocols.

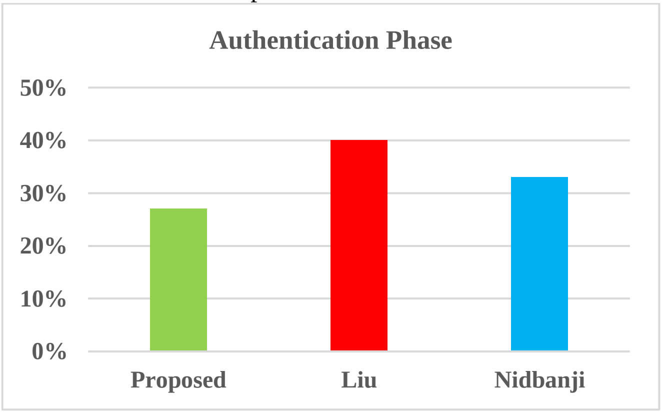


Figure 15. Time delay of authenticating user after first round

5.2 Security Analysis

This section presents a performance evaluation in terms of security analysis. The proposed protocol took the security services into consideration and to analyze the proposed protocol some assumptions were made to check if the intruder may perform a particular attack to break the proposed protocol security, assume that the intruder may: (1) Catch the messages at any time, and (2) Steal the password, user device or get the secret keys, but not simultaneously. Based on the previous assumptions, the proposed authentication protocol is secure from the following attacks:

1. User identity management: The home registration authority server saves all newly registered user IDs in the identity management table and checks the constraint of the uniqueness of user ID in each new registration process. Furthermore, the IDs are encrypted to be ready for transfer over the IoT network. In this scenario, the proposed protocol is secure against this threat.
2. Mutual authentication: The proposed protocol implements the mutual authentication, for example the messages after selecting the unused authentication vector: the user devices and RA implement the mutual authentication messages during authentication phase, from which point the two parties make sure they are legitimate.
3. Confidentiality: As in the wireless networks, the communication over the IoT network is over the open air, where an unknown number of messages are exchanged, which might be a grainy situation for intruders. From this analysis, assume that the intruder can easily catch sensitive data during the communication process. The proposed protocol provides appropriate confidentiality to the exchanged message. Therefore, the intruder cannot extract any worthwhile information from the air messages.
4. Replay attacks: The proposed protocol is resistant to replay attack by introducing a timestamp and nonce values in every exchanged message. The validity of exchanged message is achieved by checking the freshness of timestamps and nonce values.
5. Man-in-the-middle attacks: An intruder may capture the traffic ang get the login message and modify the content from LgnRegMsg = {Shra,Uhra,Tu,IDra} to LgnRegMsg = {Shra*,Uhra*,Tu,IDra}. This unacceptable attempt will not work, as the included IDu will not be legitimate to RA after checking the validity of users to HRA server. Therefore, in this scenario, the proposed protocol is secure against this attack.
6. Session key establishment: The proposed protocol implements session key establishment after ensuring the user is authenticated. A session key is created depending on the selected authentication vector between the users and RA. It will be different and cannot be replayed after the session lifetime expires.

6. Conclusions

This research work has mainly analyzed existing authentication protocols to propose a new efficient authentication protocol to enhance the security and performance aspects to authenticate the user device in IoT networks. The proposed protocol is based on guaranteed mutual authentication between entities that exist in the network. It is also built based on generating AV for each user per VLR. These AV contain different parameters that help in authenticating the user and issuing the session key. Furthermore, assessment was made of the enhanced protocol by performance and security analysis.

The results reveal that the enhanced protocol has a positive effect on the time performance of authenticating users once they have obtained access to the visited location area (after their first log in) compared with the other two protocols, with 33% improved time performance. In addition, it ensures the optimization of minimizing the use of storage in "Things" and eliminates unnecessary messages. Finally, our analysis results show that our enhanced protocol prevents several malicious possibilities, including reply and man-in-the-middle attacks and eavesdropping.

Based on the effective results gained from this research, the completed work can be to apply a new mechanism to generate a dynamic number of authentication vectors generated by the HRA servers for each user per RA.

References

[1] L. Coetzee and J. Eksteen, "The Internet of Things - promise for the future? An introduction," 2011 IST-Africa Conference Proceedings, Gaborone, Botswana, pp. 1-9, 2011.

[2] M. Wu, J. Lu, Y. Ling, J. Sun and Y. Du, "Research on the architecture of Internet of Things," 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), Chengdu, China, pp. 484–487, 2010.

[3] Z. Shi, K. Liao, S.Yin and Q. Ou, " Design and implementation of the mobile internet of things based on td-scdma network," 2010 IEEE International Conference on Information Theory and Information Security, Beijing, China, pp. 954–957, 2010.

[4] M. El-hajj, F. Ahmad, C. Maroun and S. Ahmed, "A Survey of Internet of Things (IoT) Authentication Schemes," Sensors, Vol.19, No. 5, pp. 1-43, 2019.

[5] S. Pallavi and S. Smruti, "Internet of Things: Architectures, Protocols, and Applications," Journal of Electrical and Computer Engineering, vol. 2017, pp. 1-25, 2017

[6] J. Liu, Y. Xiao and C. Chen, "Authentication and Access Control in the Internet of Things," 32nd International Conference on Distributed Computing Systems Workshops, Macau, China, pp. 588–592, 2012.

[7] B. Ndibanje, J. Lee and G. Lee, "Security Analysis and Improvements of Authentication and Access Control in the Internet of Things,". Sensors, Vol. 14, No. 8, pp. 14786–14805, 2014.

[8] J. AL-Saraireh, "Reducing Authentication Signaling Traffic for LTE Mobile Networks," International Journal of Applied Engineering Research, Vol. 12, No. 20, pp. 9306-9314, 2017

[9] J. Al-Saraireh, and S. Yousef, "Analytical Model: Authentication Transmission Overhead Between Entities in Mobile Networks," Elsevier, Computer Communications Journal, Vol. 30, No. 9. Pp. 1713-1720, 207

[10] C. Tang, and D. Wu, "An Efficient Mobile Authentication Scheme for Wireless Networks," IEEE Transactions on Wireless Communications, Vol. 7, No. 4, pp. 1408–1416, 2008.

[11] J. Lu, X. Fan, J. Zhou and H. Yang, "An Improvement on An Efficient Mobile Authentication Scheme for Wireless Networks," TELKOMNIKA, Vol. 11, No. 10, pp. 6250–2087, 2013.

[12] N. YE, Y. Zhu, R. WANG, R. Malekian and L. Qiao-min, "An Efficient Authentication and Access Control Scheme for Perception Layer of Internet of Things," Applied Mathematics & Information Sciences, Vol. 8, No. 4, pp. 1617–1624, 2014.

[13] P. Gope and T. Hwang, “A Realistic Lightweight Anonymous Authentication Protocol for Securing Real-Time Application Data Access in Wireless Sensor Networks,” IEEE Transactions on Industrial Electronics, Vol. 63, No. 11, pp. 7124-7132, 2016

[14] N. Park and N. Kang, “Mutual Authentication Scheme in Secure Internet of Things Technology for Comfortable Lifestyle,” Sensors, Vol. 16, No. 1, pp. 1-16, 2015.

[15] Z. Rachid and A. Reem, “Machine Learning based Attacks Detection and Countermeasures in IoT,” International Journal of Communication Networks and Information Security (IJCNIS) Vol. 13, No. 2, pp. 158-167, 2021

[16] A. Rasheed, and A. Izzat, “Machine learning approaches to IoT security: A systematic literature review,” Internet of Things, Vol. 14, pp. 1-14, 2021.

[17] A. Istiaque., D. Kazi, T. Mohammad, H. Mohamed, L. Sian, and A. Abdul, “Machine Learning for Authentication and Authorization in IoT: Taxonomy, Challenges and Future Research Direction,” Sensors Vol. 21, Mo. 15: pp. 1-34, 2021

[18] R. Das, A. Gadre, S. Zhang, S. Kumar and J. Moura, “A Deep Learning Approach to IoT Authentication,” *2018 IEEE International Conference on Communications (ICC)*, Kansas, USA, pp. 1-6, 2018.

[19] A. Mohammed, “Secure Multifactor Remote Access User Authentication Framework for IoT Networks,” Computers, Materials & Continua, Vol. 68, No.3, pp. 3235–3254, 2021.

[20] A, Ahmed, M. Ammar Mohamed and H. Hesham, “Proposed Authentication Protocol for IoT using Blockchain and Fog Nodes,” International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 11, No. 4, 2020.

[21] T. Zongqing, Y. Biwei, G. Qiang, H. Jianyun, D. Qingyu, “Feasibility of Identity Authentication for IoT Based on Blockchain, Procedia Computer Science,” Vol. 174, pp. 328-332. 2020.

[22] M. Ferrag, A. Maglaras, H. Janicke, J. Jiang, J. and L. Shu, “Authentication Protocols for Internet of Things: A Comprehensive Survey,” Security and Communication Networks, Vol. 10, No. 40, pp. 589–630, 2017.

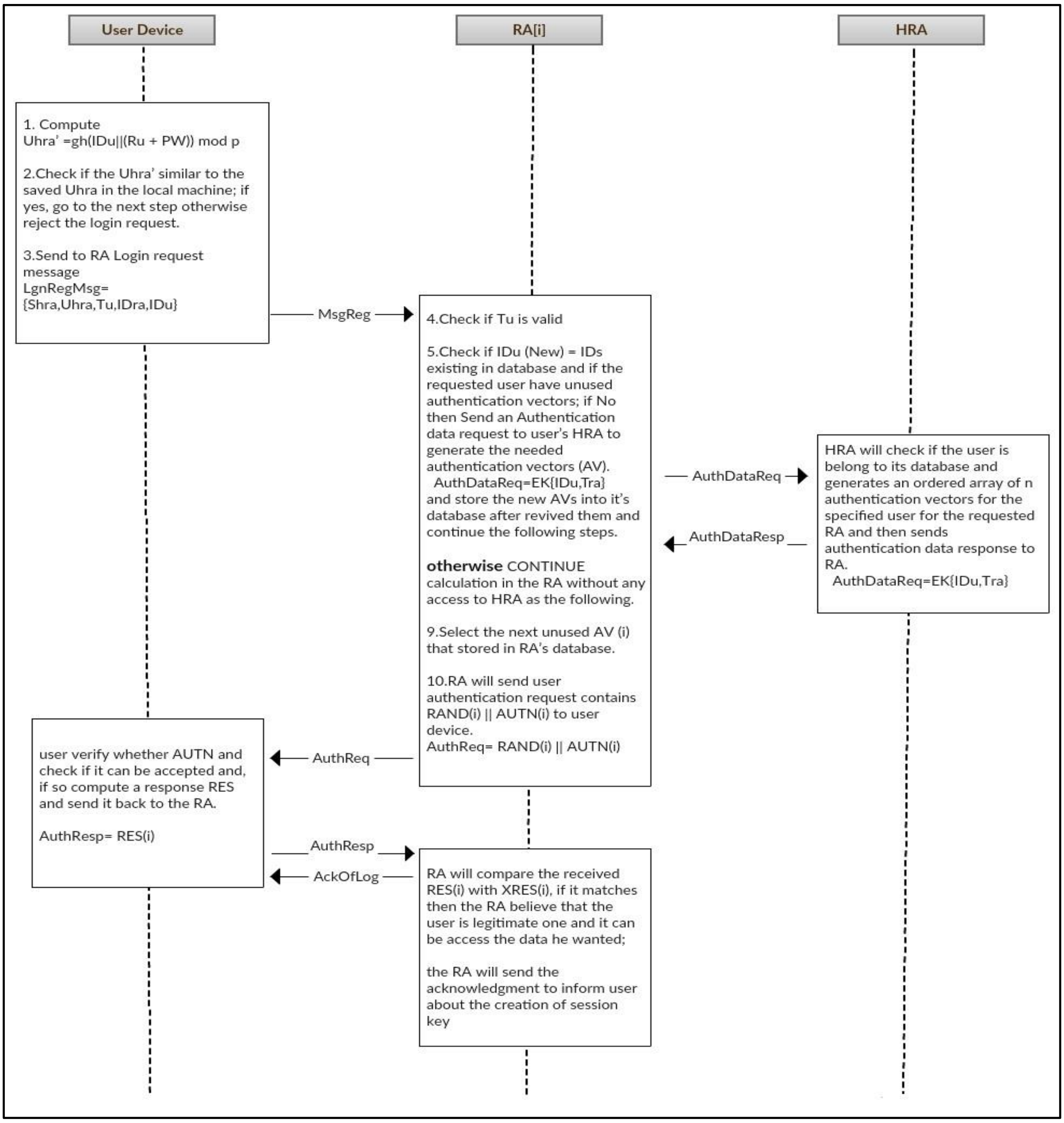
[23] G. Hemangi and C. Hiten, “Security of IoT in 5G Cellular Networks: A Review of Current Status, Challenges and Future Directions,” International Journal of Communication Networks and Information Security (IJCNIS) Vol. 13, No. 2, pp. 278-289, 2021

[24] K. Sanaz, S. Bharanidharan, A. Sami, Z. Mazdak, S, Ganthan and H. Friso, “A Systematic Literature Review of Authentication in Internet of Things for Heterogeneous Devices,” Journal of Computer Networks and Communications, Vol. 2019, pp. 1-23, 2019.

[25] V. Garg, “Wireless Communications & Networking,” Morgan Kaufmann, 2007.

[26] A. Varga and R. Hornig, “An overview of the OMNeT++ simulation environment,” 1st International Conference on Simulation tools and techniques for communications, networks and systems & workshops, Marseille, France, pp. 1–10. 2008

Figure 2. Registration/ initialization phase



1. Compute
Uhra' =gh(IDu||(Ru + PW)) mod p

2.Check if the Uhra' similar to the saved Uhra in the local machine; if yes, go to the next step otherwise reject the login request.

3.Send to RA Login request message
LgnRegMsg=
{Shra,Uhra,Tu,IDra,IDu}

4.Check if Tu is valid

5.Check if IDu (New) = IDs existing in database and if the requested user have unused authentication vectors; if No then Send an Authentication data request to user's HRA to generate the needed authentication vectors (AV).
AuthDataReq=EK{IDu,Tra}
and store the new AVs into it's database after revived them and continue the following steps.

otherwise CONTINUE calculation in the RA without any access to HRA as the following.

9.Select the next unused AV (i) that stored in RA's database.

10.RA will send user authentication request contains RAND(i) || AUTN(i) to user device.
AuthReq= RAND(i) || AUTN(i)

HRA will check if the user is belong to its database and generates an ordered array of n authentication vectors for the specified user for the requested RA and then sends authentication data response to RA.
AuthDataReq=EK{IDu,Tra}

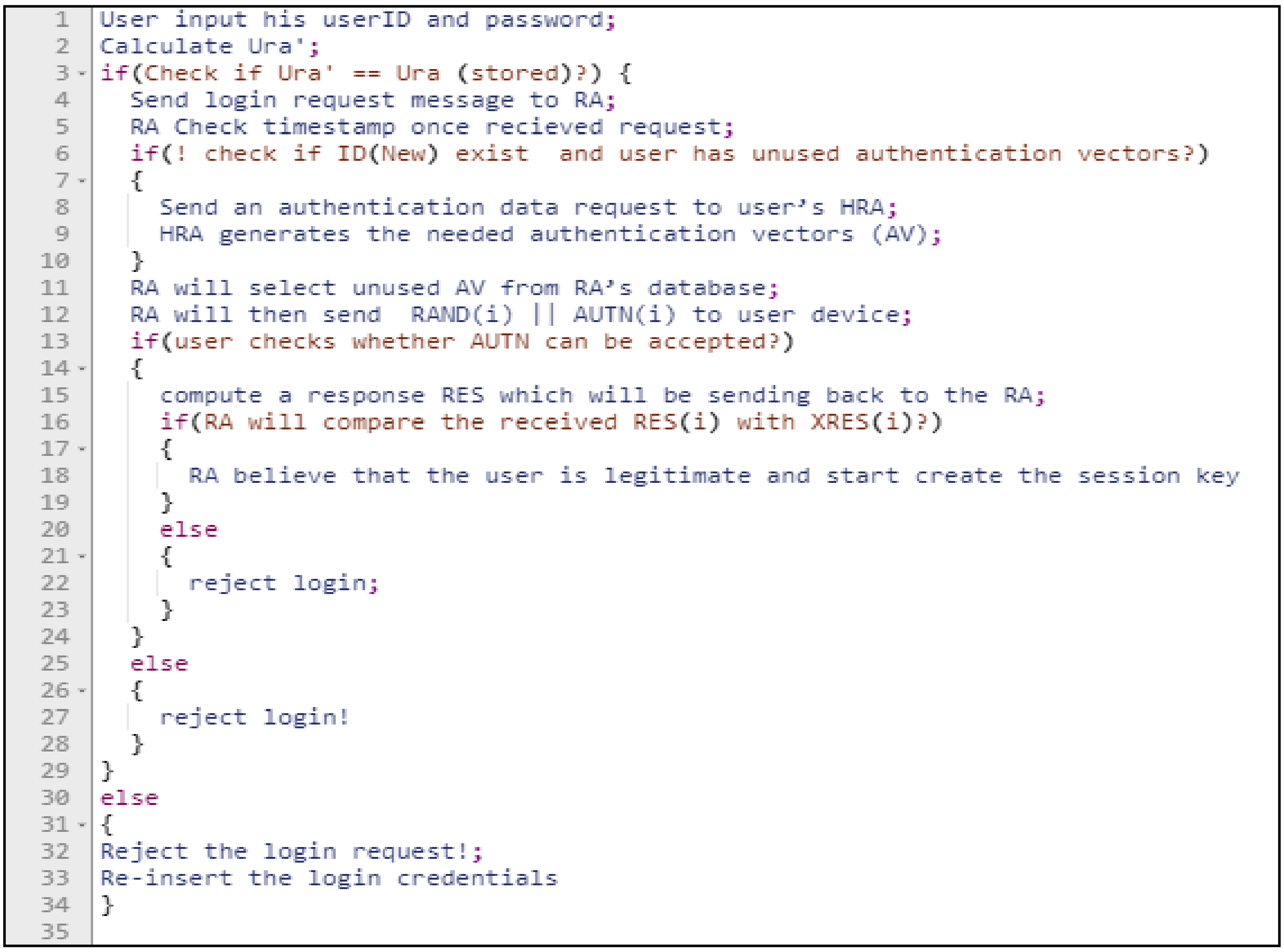
user verify whether AUTN and check if it can be accepted and, if so compute a response RES and send it back to the RA.

AuthResp= RES(i)

RA will compare the received RES(i) with XRES(i), if it matches then the RA believe that the user is legitimate one and it can be access the data he wanted;

the RA will send the acknowledgment to inform user about the creation of session key

Figure 3. Exchanged messages in login/authentication phase



```
User input his userID and password;
Calculate Ura';
if(Check if Ura' == Ura (stored)?) {
  Send login request message to RA;
  RA Check timestamp once recieved request;
  if(! check if ID(New) exist  and user has unused authentication vectors?)
  {
    Send an authentication data request to user’s HRA;
    HRA generates the needed authentication vectors (AV);
  }
  RA will select unused AV from RA’s database;
  RA will then send  RAND(i) || AUTN(i) to user device;
  if(user checks whether AUTN can be accepted?)
  {
    compute a response RES which will be sending back to the RA;
    if(RA will compare the received RES(i) with XRES(i)?)
    {
      RA believe that the user is legitimate and start create the session key
    }
    else
    {
      reject login;
    }
  }
  else
  {
    reject login!
  }
}
else
{
Reject the login request!;
Re-insert the login credentials
}
```

Figure 4. Pseudo code of proposed authentication phase

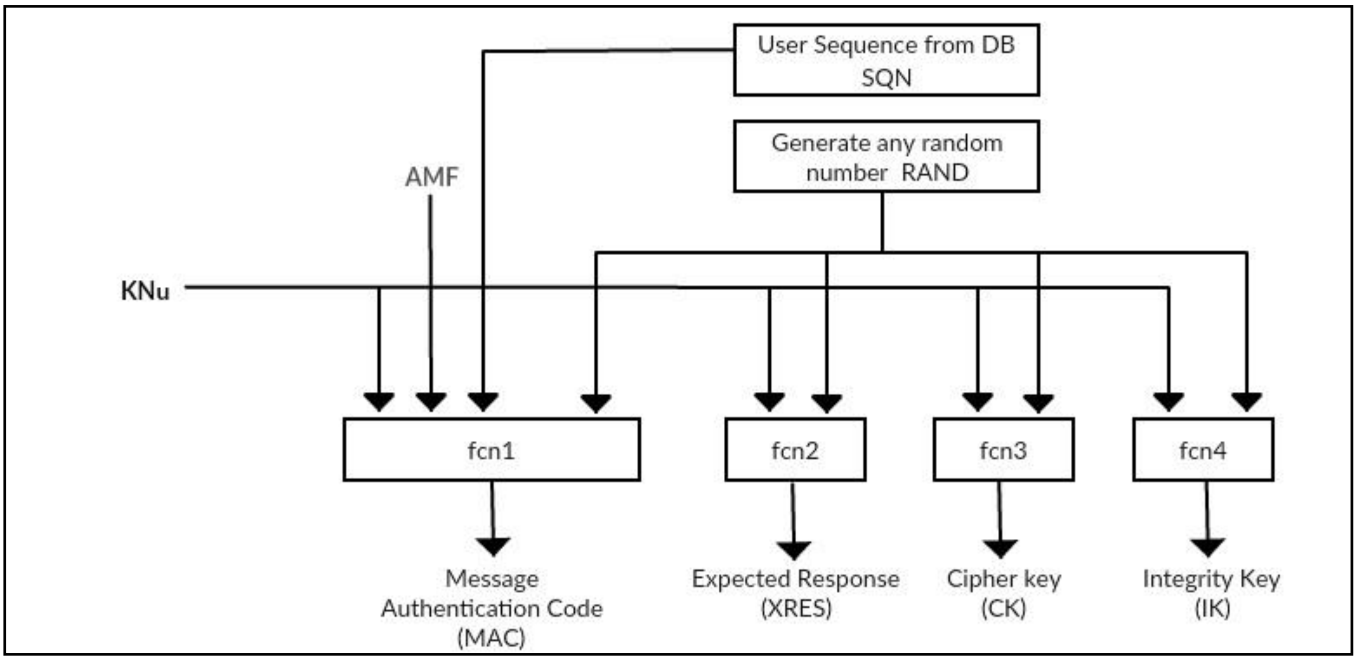

User Sequence from DB SQN
Generate any random number RAND
AMF
KNu
fcn1
fcn2
fcn3
fcn4
Message Authentication Code (MAC)
Expected Response (XRES)
Cipher key (CK)
Integrity Key (IK)

Figure 5. Generation of authentication vectors

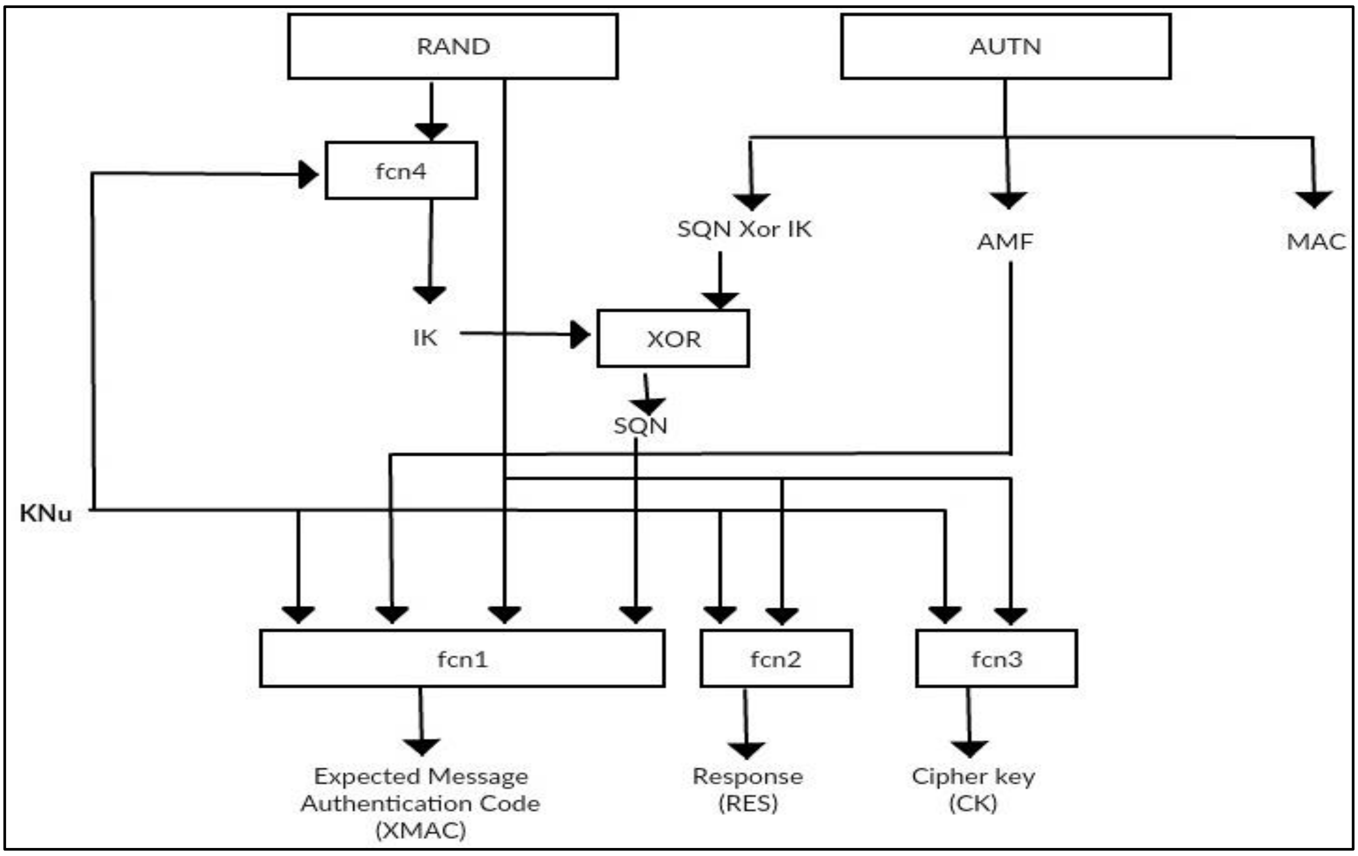
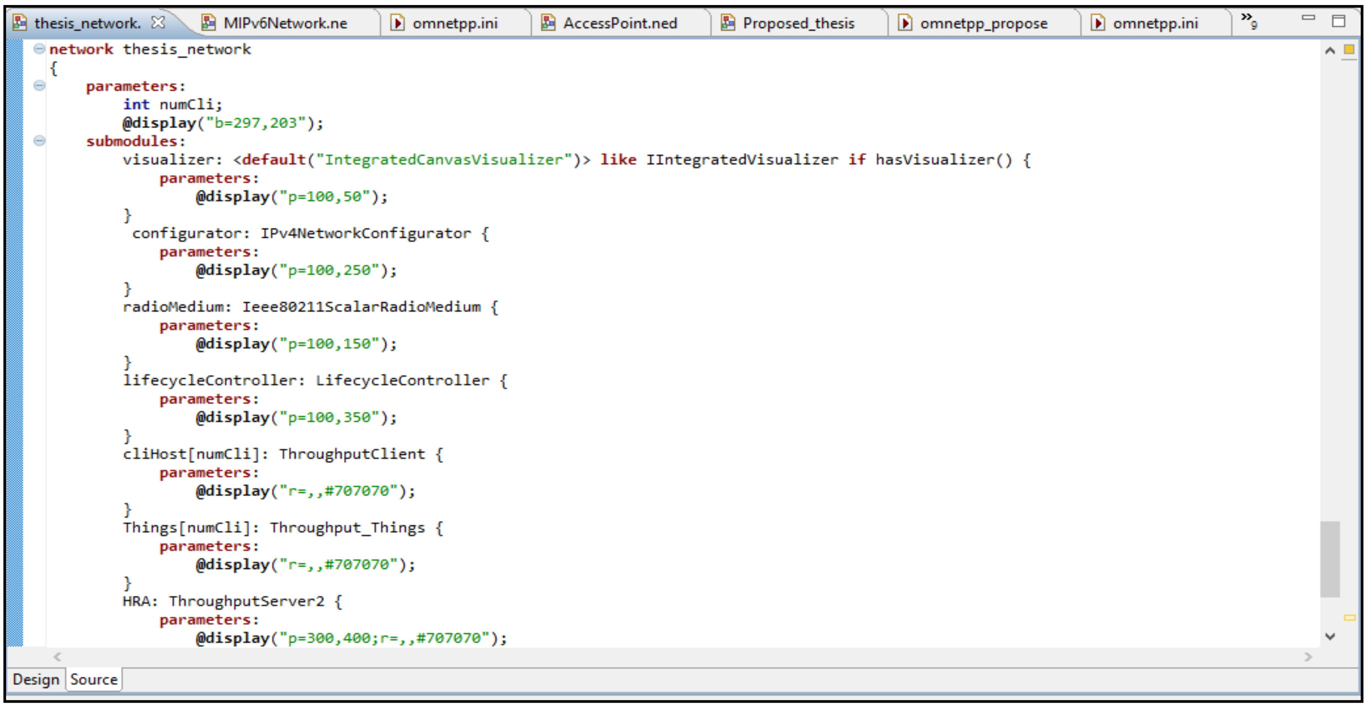


Figure 6. User authentication function in user device



```
network thesis_network
{
    parameters:
        int numCli;
        @display("b=297,203");
    submodules:
        visualizer: <default("IntegratedCanvasVisualizer")> like IIntegratedVisualizer if hasVisualizer() {
            parameters:
                @display("p=100,50");
        }
         configurator: IPv4NetworkConfigurator {
            parameters:
                @display("p=100,250");
        }
        radioMedium: Ieee80211ScalarRadioMedium {
            parameters:
                @display("p=100,150");
        }
        lifecycleController: LifecycleController {
            parameters:
                @display("p=100,350");
        }
        cliHost[numCli]: ThroughputClient {
            parameters:
                @display("r=,,#707070");
        }
        Things[numCli]: Throughput_Things {
            parameters:
                @display("r=,,#707070");
        }
        HRA: ThroughputServer2 {
            parameters:
                @display("p=300,400;r=,,#707070");
```

Figure 7. Proposed scheme's NED file

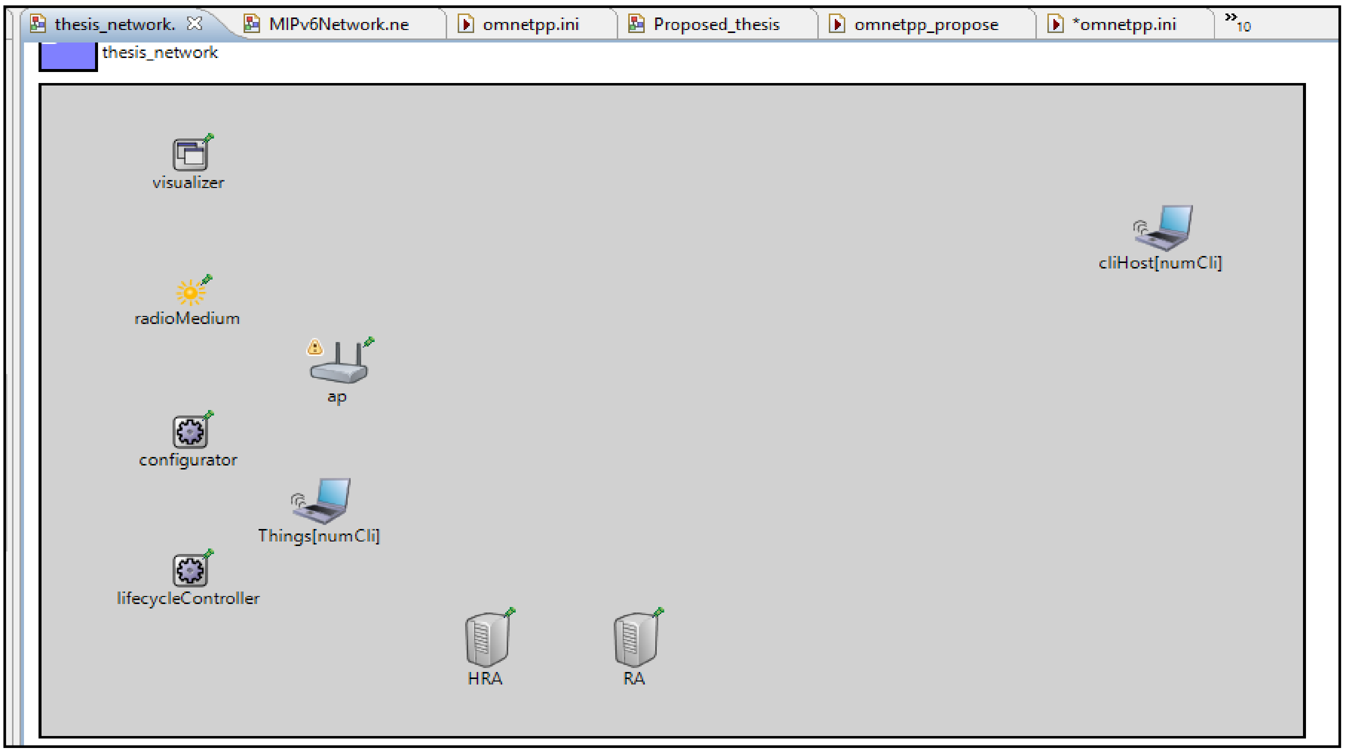
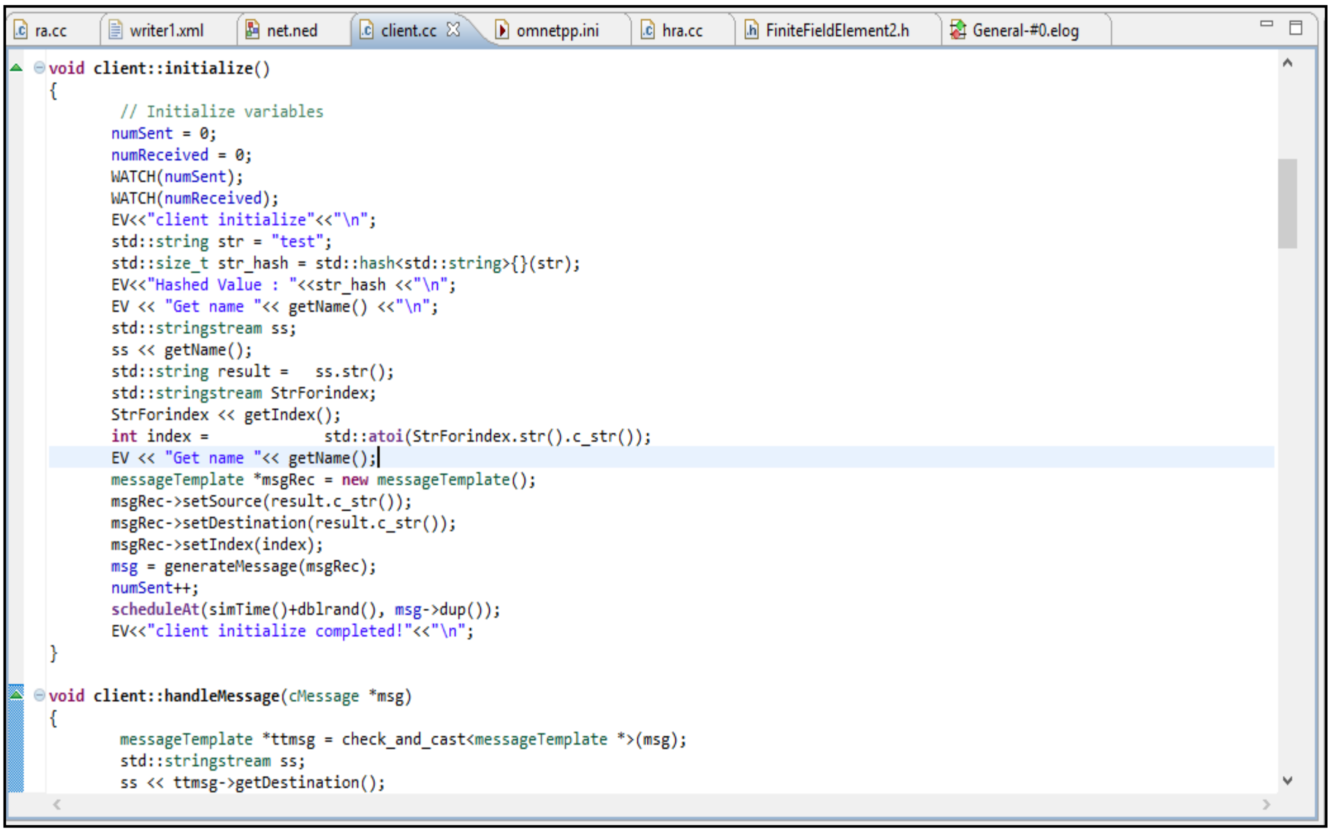


Figure 8. Proposed simulation topology



```
void client::initialize()
{
        // Initialize variables
       numSent = 0;
       numReceived = 0;
       WATCH(numSent);
       WATCH(numReceived);
       EV<<"client initialize"<<"\n";
       std::string str = "test";
       std::size_t str_hash = std::hash<std::string>{}(str);
       EV<<"Hashed Value : "<<str_hash <<"\n";
       EV << "Get name "<< getName() <<"\n";
       std::stringstream ss;
       ss << getName();
       std::string result =   ss.str();
       std::stringstream StrForindex;
       StrForindex << getIndex();
       int index =             std::atoi(StrForindex.str().c_str());
       EV << "Get name "<< getName();
       messageTemplate *msgRec = new messageTemplate();
       msgRec->setSource(result.c_str());
       msgRec->setDestination(result.c_str());
       msgRec->setIndex(index);
       msg = generateMessage(msgRec);
       numSent++;
       scheduleAt(simTime()+dblrand(), msg->dup());
       EV<<"client initialize completed!"<<"\n";
}

void client::handleMessage(cMessage *msg)
{
        messageTemplate *ttmsg = check_and_cast<messageTemplate *>(msg);
        std::stringstream ss;
        ss << ttmsg->getDestination();
```

Figure 9. Source file that handles exchanged messages

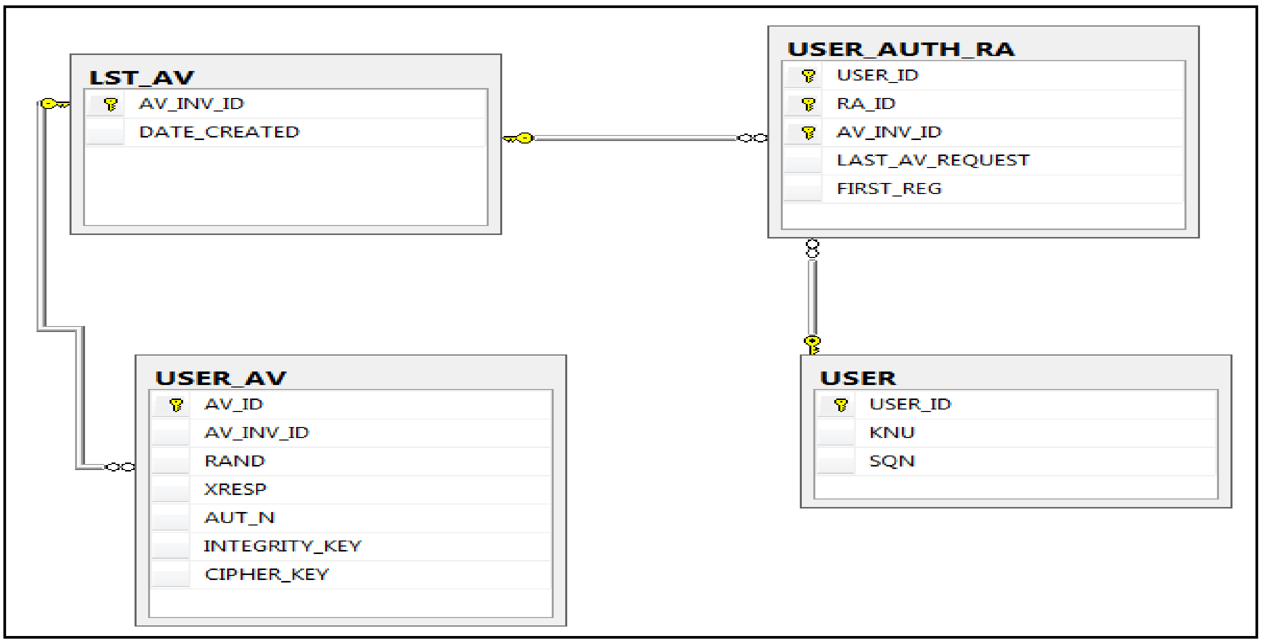


Figure 10. HRA data model

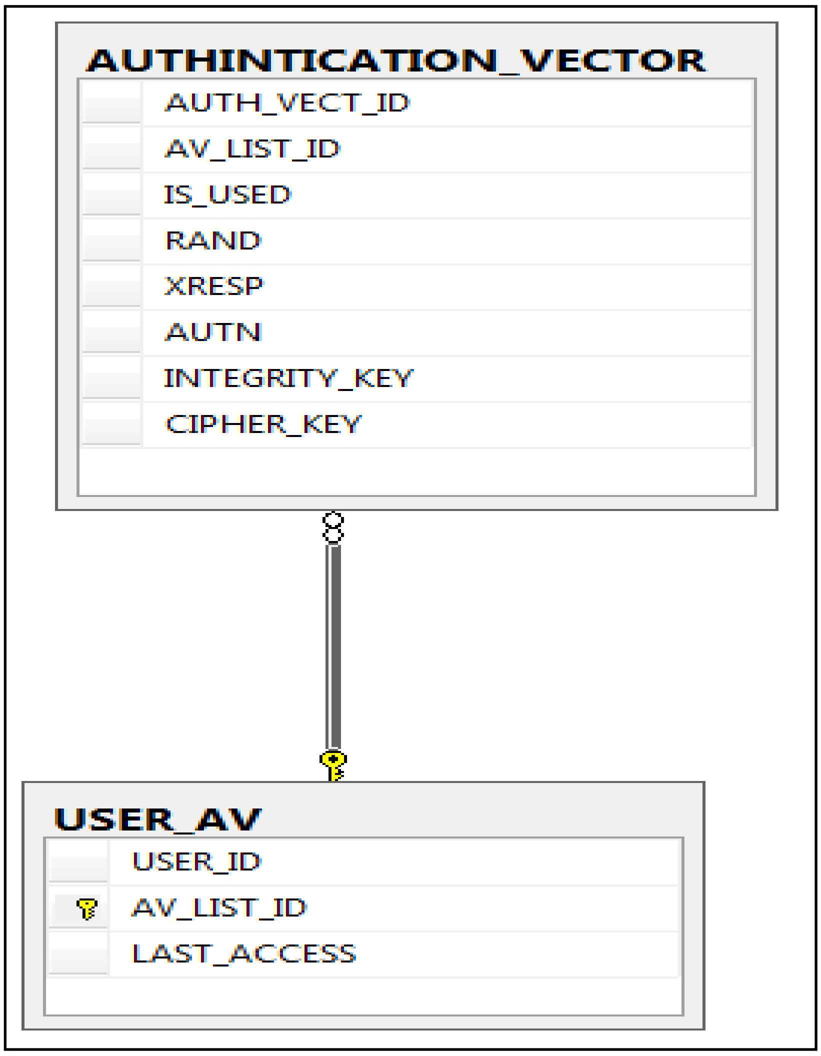
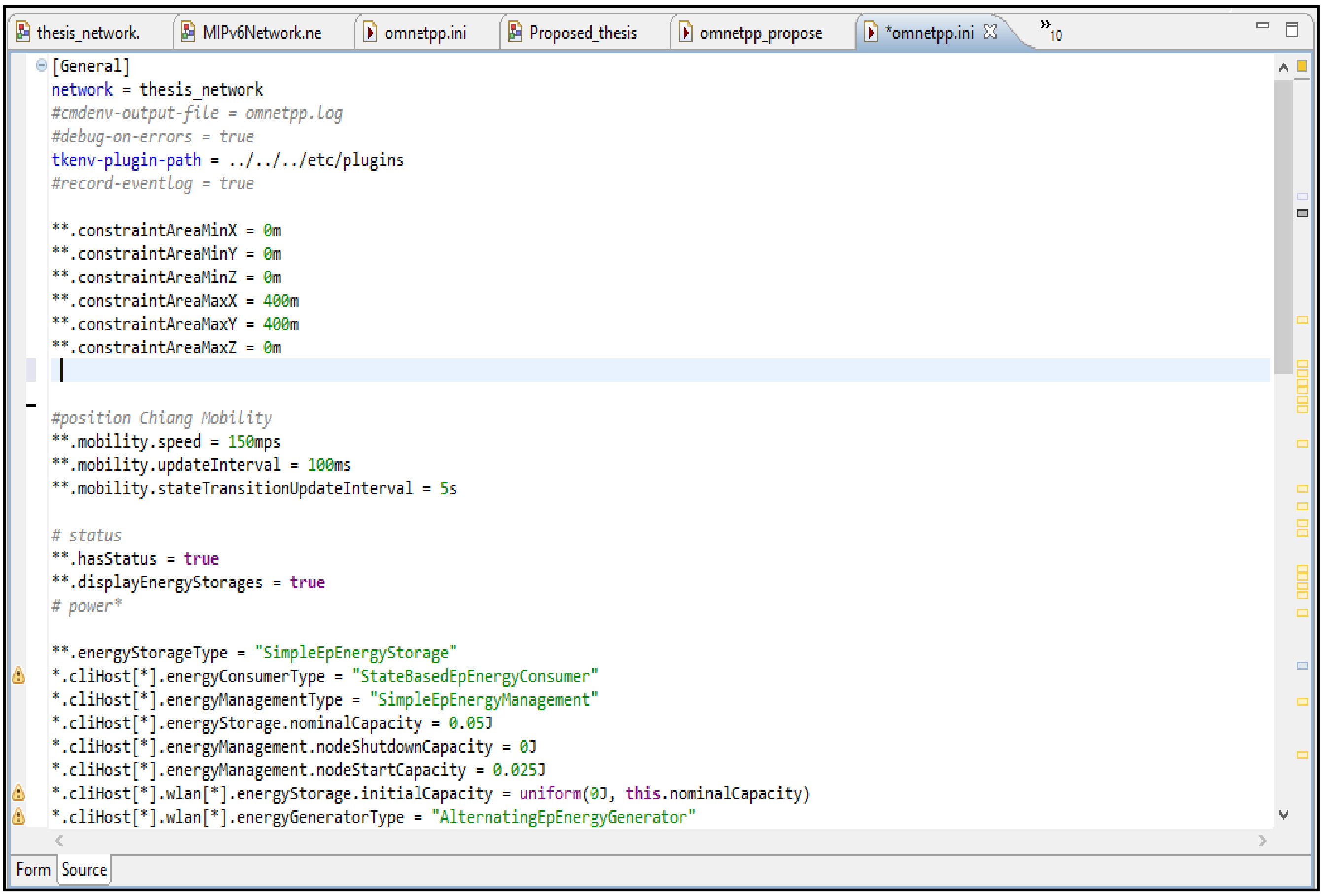


Figure 11. RA data model



```
[General]
network = thesis_network
#cmdenv-output-file = omnetpp.log
#debug-on-errors = true
tkenv-plugin-path = ../../../etc/plugins
#record-eventlog = true

**.constraintAreaMinX = 0m
**.constraintAreaMinY = 0m
**.constraintAreaMinZ = 0m
**.constraintAreaMaxX = 400m
**.constraintAreaMaxY = 400m
**.constraintAreaMaxZ = 0m


#position Chiang Mobility
**.mobility.speed = 150mps
**.mobility.updateInterval = 100ms
**.mobility.stateTransitionUpdateInterval = 5s

# status
**.hasStatus = true
**.displayEnergyStorages = true
# power*

**.energyStorageType = "SimpleEpEnergyStorage"
*.cliHost[*].energyConsumerType = "StateBasedEpEnergyConsumer"
*.cliHost[*].energyManagementType = "SimpleEpEnergyManagement"
*.cliHost[*].energyStorage.nominalCapacity = 0.05J
*.cliHost[*].energyManagement.nodeShutdownCapacity = 0J
*.cliHost[*].energyManagement.nodeStartCapacity = 0.025J
*.cliHost[*].wlan[*].energyStorage.initialCapacity = uniform(0J, this.nominalCapacity)
*.cliHost[*].wlan[*].energyGeneratorType = "AlternatingEpEnergyGenerator"
```

Figure 12. Simulation control file