Scientia Iranica D (2022) 29(6), 3273–3287

Sharif University of Technology
Scientia Iranica
Transactions D: Computer Science & Engineering and Electrical Engineering
http://scientiairanica.sharif.edu

A novel group-based secure lightweight authentication and key agreement protocol for machine-type communication

M.M. Modiri[a,*], J. Mohajeri[b], and M. Salmasizadeh[b]

a. *Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran.*
b. *Electronics Research Institute, Sharif University of Technology, Tehran, Iran.*

Received 13 November 2019; received in revised form 10 October 2020; accepted 15 February 2021

KEYWORDS
IoT;
Network security;
M2M communication;
Group-based AKA;
AVISPA.

Abstract. Nowadays, one of the most important criteria in designing different generations of cellular technology is to handle a large number of heterogeneous devices with high security guarantees. The first significant security issue considered in this field is the mutual authentication of the devices and the network as well as the authenticated key agreement between them. Hence, various Authentication and Key Agreement (AKA) protocols were proposed for Long-Term Evolution (LTE) and 5G networks. However, each of the protocols suffers various security and performance issues. This study proposes a Group-based Secure Lightweight Authentication and Key Agreement (GSL-AKA) protocol for Machine-to-Machine (M2M) communication. Security analysis and formal verification using the Automated Validation of Internet Security-sensitive Protocols and Applications (AVISPA) tool prove that the proposed protocol overcomes various known security attacks and provides all the considered security requirements. Moreover, performance analysis shows that the communication and computation overheads of the proposed protocol are the lowest of all other existing group-based AKA protocols.

1. Introduction

Wireless telecommunication technology is constantly growing. Currently, the telecommunication technology is being developed to provide wide and secure communications between things with different applications around the world. Moreover, full support for Internet of Things (IoT) is one of the most important parts in the design of the next generation of mobile communications system (5G). Machine Type Communication (MTC), also known as Machine-to-Machine (M2M) communication, is an important practical application of the IoT. In M2M communication, MTC Devices (MTCDs) simply communicate with each other through wired and wireless networks. This type of communication has a variety of applications such as smart health-care systems, smart cities, remote control systems of industrial robots, smart electricity grids, and vehicles tracing and tracking systems [1–3].

The 3GPP architecture for M2M communication in Long-Term Evolution (LTE) and 5G networks includes three communication parties: Home Subscriber Server (HSS), Mobile Management Entity (MME), and MTCDs. The role of each entity and how to communicate between them are fully explained in [4]. The group-based Authentication and Key Agreement (AKA) protocols for M2M communication mutually authenticate a group of MTCDs and network entities

*. *Corresponding author.*
E-mail address: m.modiri96@student.sharif.edu (M.M. Modiri)

doi: 10.24200/sci.2021.54832.3936

(MME and HSS) and share some secret session keys between each MTCD and the network. These protocols provide a secure communication between MTCDs and the network and prevent eavesdropping and modifying the messages transmitted through wireless channels.

Security and privacy are the main challenges for M2M communication [5]. Moreover, the MTCDs, such as mobile devices and smart cards, have limited computing sources and most of them are not capable to communicate independently with the network. In addition, if each MTCD performs the AKA protocol individually, the network will face significant congestion and huge communication and computation overheads [6]. Hence, it is necessary to propose a group-based secure and lightweight AKA protocol for M2M communication.

Various group-based AKA protocols have been proposed. In this paper, the group-based secure lightweight Group-based Secure Lightweight Authentication and Key Agreement (GSL-AKA) protocol is proposed. The proposed protocol overcomes the security and performance problems existing in the previous protocols and resists known attacks. Moreover, due to the lack of encryption operations (whether symmetric or asymmetric) and the use of only hash functions, the proposed protocol has the best performance among the existing protocols in terms of network overheads. It is worth noting that this paper is an extended version of our IST'2018 conference paper [7] to which some other contents such as formal security verification, full discussion of security properties, how to calculate the network overheads, and so on were added.

The remainder of this paper is organized as follows: Section 2 discusses the related research works of the group-based AKA protocols. Section 3 presents the proposed GSL-AKA protocol for M2M communication in LTE and 5G networks. Section 4 illustrates the formal security analysis of the protocol using Automated Validation of Internet Security-sensitive Protocols and Applications (AVISPA) tool. Section 5 analyzes the security properties of the proposed protocol and compares it with the other group-based AKA protocols. Section 6 illustrates the comparative performance analysis of the proposed protocol with existing group-based AKA protocols and shows that the network overheads of the proposed protocol are the lowest. Then, Section 7 discusses and proves why the proposed protocol, by meeting all the security requirements mentioned in Section 5, is able to achieve the lowest overheads calculated in Section 6. Finally, Section 8 presents the concluding remarks and future work.

2. Related works

The main goals of the group-based AKA protocols are mutual AKA between MTCDs and the network along with preserving MTCDs privacy and reducing network overheads. According to these goals, a variety of group-based AKA protocols have been proposed for M2M communication. In this section, an overview of available group-based AKA protocols is provided.

The first group-based AKA protocol was proposed by Chen et al. [8] with the name of G-AKA. In this protocol, the MME uses the existing information of the first authenticated device to authenticate the rest of devices. Thus, the AKA process can be simplified for all the remaining devices in the group. However, the protocol produces the signaling congestion when a mass of MTCDs simultaneously requires network access. It also suffers from various security attacks such as Denial of Service (DoS) and Man-in-the-Middle (MitM). To improve the security of G-AKA, Lai et al. [9] proposed SE-AKA for 3GPP networks and Jiang et al. [10] proposed EG-AKA for non-3GPP networks. These protocols resist mentioned attacks, but suffer from high computation overhead due to asymmetric key operations.

To reduce computation overhead, Lai et al. [11] proposed the symmetric key-based NOVEL-AKA protocol. However, it is subject to some other problems such as network signaling congestion and is also vulnerable to DoS and redirection attacks. Moreover, Choi et al. [12] proposed the GROUP-AKA protocol that successfully reduced signaling congestion when a mass of MTCDs simultaneously requires access to the network and maintains the unlinkability of the group key. However, the protocol suffers the privacy preservation problem and DoS attack. To improve the security of group-based AKA protocols, Cao et al. [13] suggested a group signature-based GBAAM-AKA protocol, but due to asymmetric key operations, the protocol generates high computation overhead and fails to preserve the privacy. For preserving the privacy, Fu et al. [14] proposed the PRIVACY-AKA protocol that used asymmetric cryptography. The protocol resists known attacks, but generates high computation overhead and fails to achieve the key forward and backward secrecy.

To reduce computation and communication overheads, Lai et al. [15] proposed lightweight GLARM-AKA protocol. The protocol is useful for resource-constrained MTCDs, but it fails to maintain the unlinkability in the group key and suffers from identity catching and impersonation attacks. To maintain security and privacy, Li et al. [16] proposed the GR-AKA protocol which preserves the identifier of MTCDs with complex and time-consuming Lagrange Component (LC) computations. Yao et al. [17] proposed the group-based secure GBS-AKA protocol to resist attacks and reduce communication overhead. However, it fails to preserve the privacy of MTCDs and suffers

Table 1. Notations and symbols.

Notation	Description	Size (bits)
$IMSI_{G1-i}$	International mobile subscriber identifier	128
$TMSI_{G1-i}$	Temporary mobile subscriber identifier	128
LAI	Location area identifier	40
ID_{G1}	Group identifier	128
TID_{G1}	Group identifier	128
AMF	Authentication management field	48
$MAC/XMAC$	Message authentication code	64
TS	Time stamp	64
GAV	Group authentication vector	560
$RAND$	Random number	128
K_{G1-i}	Pre-shared secret key	128
K_{ASME}	Access security management entity key	256
K_{G1}	Group key	128
TK_{G1}	Group temporary key	128
CK/AK	Cipher/integrity key	128
SN_{ID}	Serving network identifier	128
SQN	Sequence number	48
LMK	Local master key	256
KID_i	Key identifier	128
LC	Lagrange components	128
$ECDK$	Elliptic curve DH key	192
$ECDS$	Elliptic curve digital signature	448

the impersonation and DoS attacks. Moreover, it fails to maintain the unlinkability in the group key.

To improve security, Parne et al. [18] proposed the security-enhanced group-based SEGB-AKA protocol. The protocol preserves privacy of the MTCDs and overcomes most of the known attacks. It maintains the unlinkability in the group key and whenever one MTCD wants to join or leave the group, the group key will be changed. Moreover, it has reasonable computation and communication overheads. However, the protocol suffers from one DoS attack and contrary to its claims, it fails to overcome the single key problem in the communication networks, which are fully explained in Section 5.

In view of the above-identified security and non-security problems, GSL-AKA protocol for M2M communication is proposed. The proposed protocol has the same structure as its two previous protocols, GBS-AKA [17] and SEGBAKA [18], and is able to improve their properties. Moreover, the proposed protocol successfully overcomes known security and non-security problems and preserves the privacy of MTCDs and the group. The proposed protocol uses only hash functions to mutually authenticate the entities and for this reason, the network overheads of the protocol are the lowest of all other protocols. Finally, the proposed protocol is able to overcome the single key problem that the previous group-based AKA protocols could not solve.

3. The proposed GSL-AKA protocol

This section introduces the proposed GSL-AKA protocol for M2M communication in LTE and 5G networks. In the proposed protocol, a mass of MTCDs with the same local communication area form a group and a device with high communication capability is chosen as the group leader. The group leader transfers and receives data to/from each group members and generates and verifies the group MACs with the participation of all group members. If the group leader sabotages during the authentication process, the network and the group members will be found out and the process of authentication will abort.

The proposed protocol consists of two phases: i) Group initialization and key establishment phase, and ii) Group-based authentication and key agreement phase. The notations and symbols used in the protocol are presented in Table 1. Note that, in these symbols, the notation G_1 represents the group "G_1" and the notation G_{1-i} represents the ith member of the group "G_1".

3.1. Pre-shared parameters

Before illustrating the GSL-AKA protocol, we define some basic notions and assumptions for the protocol which are pre-shared between protocol entities. These are as follows:

- According to the 3GPP architecture, $K_{G_{1-i}}$ is the pre-shared secret key between each MTCD and HSS;
- The Key Generation Center (KGC) generates the secure group key (K_{G_1}) and unique group identifier (ID_{G_1}) for the group;
- We introduced a temporary mobile subscriber identifier ($TMSI_{G_{1-i}}$) and a group temporary identifier (TID_{G_1}), which are used to preserve $IMSI_{G_{1-i}}$ and ID_{G_1};
- The symbol $f^1(.)$ is a hash-based MAC generation function and $f^2(.)$, $f^3(.)$, and $f^4(.)$ are key generation functions. The structure of these functions is explained in [19,20]. Note that the key used in these functions is written as a subscript, e.g., $f^1_{K_{G_{1-i}}}(.)$ applies $K_{G_{1-i}}$ for generating MACs;
- Moreover, $f^s(.)$ is a supplementary cryptographic one-way function. It is used to generate new TMSI and TID;
- The channel between MME and HSS is assumed to be secured.

3.2. Group initialization and key establishment

The key and identifier of each group are generated with the participation of all group members. It is necessary to update the group key whenever a member wants to join/leave the group. To generate a new key or identifier for the group, each member chooses a random value. These values are associated with the leaf nodes of the binary Merkle Tree; then, root node value is calculated as the new key or identifier. The group key creation, distribution, and revocation in the group communication have been widely studied and these are out of scope of our work. These issues were covered in [21,22].

The main difference between the proposed protocol and the previous protocols is that in the proposed protocol, the MTCDs and the group, transmit temporary IMSI ($TMSI_{G_{1-i}}$) and group temporary ID (TID_{G_1}) in wireless channels, instead of their original values. These two temporary values are used for preserving the identifiers and can be generated only by the corresponding MTCD and the HSS. At the time of the group initialization, $TMSI_{G_{1-i}}$ and TID_{G_1} take certain initial values and after each successful protocol process, these are updated as follows:

$$TID^{new}_{G_1} = f^s_{K_{G_1}}(ID_{G_i}||RAND_{HSS}), \tag{1}$$

$$TMSI^{new}_{G_{1-i}} = f^s_{K_{G_{1-i}}}(IMSI_{G_{1-i}}||RAND_{HSS}). \tag{2}$$

3.3. Group-based AKA

At this phase, the GSL-AKA protocol is presented to mutually authenticate a group of MTCDs and the HSS/MME and share some secret session keys between them. The proposed protocol, as its two previous protocols, uses hash functions during the authentication process. The structure of the protocol is shown in Figure 1 and the details are given in the following steps:

Step 1: The MTCDs, $MTCD_{G_{1-1}}$, $MTCD_{G_{1-2}}$, and $MTCD_{G_{1-n}}$ of the group G_1, request the MME to access to the network through the group leader ($MTCD_{G_{1-leader}}$).

Step 2: The MME requests the identifier of each $MTCD_{G_{1-i}}$ from the $MTCD_{G_{1-leader}}$.

Step 3: The $MTCD_{G_{1-leader}}$ generates the identifier response message ($AUTH_{G_1}$) as follows:

- Each $MTCD_{G_{1-i}}$ calculates its own authentication code as:

$$MAC_{G_{1-i}} = f^1_{K_{G_{1-i}}}(ID_{G_1}||IMSI_{G_{1-i}}||TS_{G_1}). \tag{3}$$

- Later, the MTCDs forward their $MAC_{G_{1-i}}$ to the $MTCD_{G_{1-leader}}$ (If $MTCD_{G_{1-leader}}$ does not have $TMSI_{G_{1-i}}$ of each MTCD, these send this value to it.).
- The $MTCD_{G_{1-leader}}$ generates the aggregated authentication code (MAC_{G_1}) using all $MAC_{G_{1-i}}$ and the Location Area Identifier (LAI) of the corresponding base station (The LAI parameter is used for avoiding form redirection attacks [23].).

$$MAC_{G_1} = f^1_{K_{G_1}}(MAC_{G_{1-i}}||...|| MAC_{G_{1-i}}||LAI). \tag{4}$$

- The $MTCD_{G_{1-leader}}$ generates the identifier response message ($AUTH_{G_1}$) as follows:

$$AUTH_{G_1} = (TID_{G_1}||TMSI_{G_{1-1}}||...|| TMSI_{G_{1-n}}||MAC_{G_1}||TS_{G_1}). \tag{5}$$

- Finally, the $MTCD_{G_{1-leader}}$ transmits $AUTH_{G_1}$ to the MME.

Step 4: After receiving $AUTH_{G_1}$, MME concatenates the LAI of the corresponding base station with $AUTH_{G_1}$ ($AUTH_{G_1}||LAI$) and forwards these to the HSS.

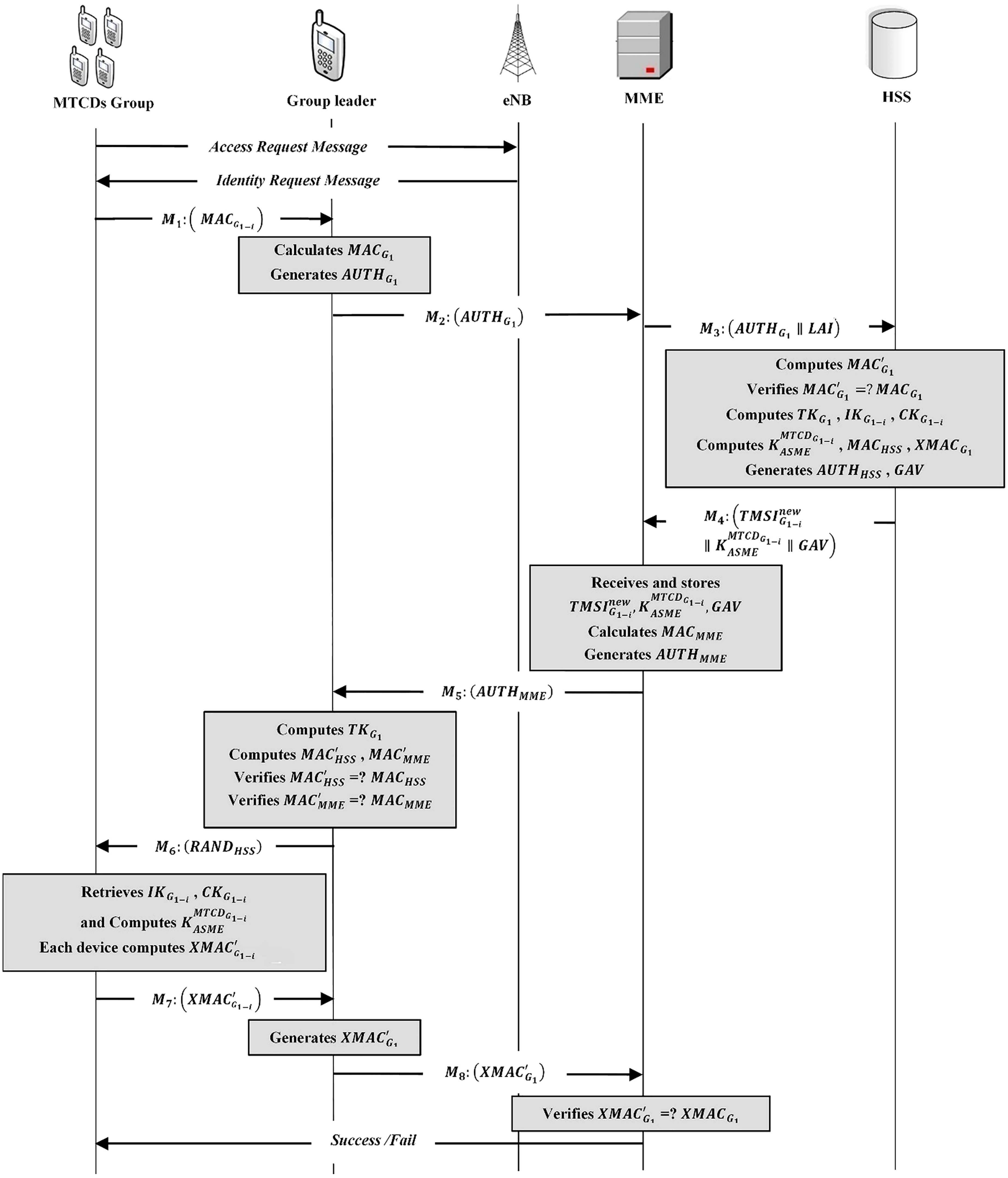


Figure 1. The GSL-AKA protocol.

Step 5: After acquiring $AUTH_{G_1}||LAI$ from MME, HSS authenticates all the MTCDs in the group as follows:

- First, it verifies whether TS_{G_1} is in the acceptable range or not. If it is not, HSS declines the authentication request;
- By using TID_{G_1} and $TMSI_{G_{1-i}}$ assigned to ID_{G_1} and $IMSI_{G_{1-i}}$, HSS retrieves the group key (K_{G_1}) and the respective MTCDs key ($K_{G_{1-i}}$);
- Finally, HSS computes MAC'_{G_1} using Eqs. (3) and (4) and verifies whether the computed MAC'_{G_1} is matched with the received MAC_{G_1}. If these are equal, HSS authenticates all the MTCDs in the group; otherwise, HSS declines the authentication request.

Step 6: After verifying MAC_{G_1}, HSS computes the secret session keys of each $MTCD_{G_{1-i}}$ as follows:

- HSS generates a random number ($RAND_{HSS}$).

- Then, HSS computes the group temporary key as:

$$TK_{G_1} = f^2_{K_{G_1}}(ID_{G_1}||RAND_{HSS}). \quad (6)$$

- According to the 3GPP standards of key derivations [23], the HSS computes the integrity and cipher keys of each $MTCD_{G_{1-i}}$ as:

$$IK_{G_{1-i}} = f^3_{K_{G_{1-i}}}(IMSI_{G_{1-1}}||RAND_{HSS}), \quad (7)$$

$$CK_{G_{1-i}} = f^4_{K_{G_{1-i}}}(IMSI_{G_{1-1}}||RAND_{HSS}). \quad (8)$$

- Then, the secret session key is computed using KDF (Key Derivation Function) for each $MTCD_{G_{1-i}}$ from those keys ($IK_{G_{1-i}}$, $CK_{G_{1-i}}$) as:

$$K_{ASME}^{MTCD_{G_{1-i}}} = KDF(TK_{G_1}||IK_{G_{1-i}} ||CK_{G_{1-i}}||ID_{G_1}||IMSI_{G_{1-1}}). \quad (9)$$

Step 7: After computing the session keys, HSS generates the authentication response message and new temporary identifiers as follows:

- HSS computes the MAC_{HSS} as:

$$MAC_{HSS} = f^1_{TK_{G_1}}(RAND_{HSS}||AMF). \quad (10)$$

- HSS generates $AUTH_{HSS}$ as:

$$AUTH_{HSS} = (MAC_{HSS}||RAND_{HSS}||AMF). \quad (11)$$

- HSS computes the respective authentication code for each $MTCD_{G_{1-i}}$ as:

$$XMAC_{G_{1-i}} = f^1_{K_{G_{1-i}}}(ID_{G_1}||IMSI_{G_{1-i}} ||RAND_{HSS}). \quad (12)$$

- HSS generates the aggregated respective authentication code for the group as follows:

$$XMAC_{G_1} = f^1_{K_{G_1}}(XMAC_{G_{1-i}}||...|| XMAC_{G_{1-i}}). \quad (13)$$

- HSS assigns a new TID_{G_1} for G_1 using Eq. (1) and a new $TMSI_{G_{1-i}}$ for each $MTCD_{G_{1-i}}$ using Eq. (2) for establishing further communications.
- Later, the HSS generates the Group Authentication Vector (GAV) from the above computed parameters.

$$GAV = (AUTH_{HSS}||XMAC_{G_1}|| TID_{G_1}||TK_{G_1}). \quad (14)$$

- Finally, HSS transmits the authentication response message ($TMSI_{G_{1-i}}||K_{ASME}^{MTCD_{G_{1-i}}}||GAV$) to MME.

Step 8: After receiving the authentication response message ($TMSI_{G_{1-i}}||K_{ASME}^{MTCD_{G_{1-i}}}||GAV$):

- MME stores the authentication response message for communicating with MTCDs and further authentication processes.
- Then, MME generates $RAND_{MME}$ and calculates the MAC_{MME} as follows:

$$MAC_{MME} = f^1_{TK_{G_1}}(MAC_{HSS}||RAND_{MME}). \quad (15)$$

- Later, MME generates authentication token as:

$$AUTH_{MME} = (MAC_{MME}||RAND_{MME} ||MAC_{HSS}||RAND_{HSS}||AMF). \quad (16)$$

- Finally, it sends $AUTH_{MME}$ to the $MTCD_{G_{1-leader}}$.

Step 9: After acquiring $AUTH_{MME}$, $MTCD_{G_{1-leader}}$ performs the following operations:

- $MTCD_{G_{1-leader}}$ computes MAC'_{HSS} using Eq. (10) and verifies whether the computed MAC'_{HSS} is matched with the received MAC_{HSS}. If these are equal, the $MTCD_{G_{1-leader}}$ authenticates the HSS; otherwise, the $MTCD_{G_{1-leader}}$ aborts the authentication process;
- The $MTCD_{G_{1-leader}}$ computes MAC'_{MME} using Eq. (15) and verifies whether the computed MAC'_{MME} is matched with the received MAC_{MME}. If these are equal, the $MTCD_{G_{1-leader}}$ authenticates the MME; otherwise, $MTCD_{G_{1-leader}}$ aborts the authentication process;
- Finally, the $MTCD_{G_{1-leader}}$ broadcasts $RAND_{HSS}$ and the successful HSS/MME authentication message to all the MTCDs in the group. If the $MTCD_{G_{1-leader}}$ sabotages during the authentication process, the network and the group members are determined by $XMAC_{G_{1-i}}$ and $XMAC_{G_1}$.

Step 10: Now, each $MTCD_{G_{1-i}}$ performs the following operations:

- Each $MTCD_{G_{1-i}}$ computes the group temporary key, integrity key, and cipher key using Eqs. (6), (7), and (8) and generates the secret session key $K_{ASME}^{MTCD_{G_{1-i}}}$ using Eq. (9) to communicate securely with the HSS/MME;
- Each $MTCD_{G_{1-i}}$ generates its respective authentication code ($XMAC'_{G_{1-i}}$) using Eq. (12) and sends it to the $MTCD_{G_{1-leader}}$.

Step 11: The $MTCD_{G_{1-leader}}$ calculates the aggregated respective authentication code ($XMAC'_{G_1}$) using Eq. (13) and sends it to the MME for mutual authentication of each $MTCD_{G_{1-i}}$ with the MME.

Step 12: Finally, MME verifies whether $XMAC'_{G_1}$ sent from the $MTCD_{G_{1-leader}}$ matches with

$XMAC_{G_1}$ sent from the HSS or not. If these are equal, MME broadcasts to each $MTCD_{G_{1-i}}$ unforgeable authentication success message (e.g., $TMSI^{new}_{G_{1-i}}$ value). Otherwise, MME broadcasts authentication failure message.

After the successful protocol process, the group calculates new TID_{G_1} and each $MTCD_{G_{1-i}}$ calculates new $TMSI_{G_{1-i}}$ for communicating with the network and future authentication purpose.

4. Formal security verification using AVISPA tool

The GSL-AKA protocol was coded in HLPSL [24] language and tested by the formal security verification, the AVISPA tool [11-25], to analyze its security properties. The main goal of the protocol is to provide mutual authentication between each MTCD and the network entities (HSS and MME). In addition, the proposed protocol should be able to provide the confidentiality of the pre-shared secret key for each MTCD ($K_{G_{1-i}}$) and the group key (K_{G_1}) during the authentication process. The goals of the protocol are given in Figure 2. The protocol has three main parties: MTCDs, MME, and HSS. The roles of these parties in HLPSL language are described in Appendix A. It is assumed that the channel between the HSS and MME is secure and an attacker only dominates the channel between the MTCDs and the MME. The outputs of security analysis and verification using OFMC and CL-AtSe backends in the AVISPA tool are shown in Figures 3 and 4, respectively. The results prove that the GSL-AKA protocol can reach the mentioned goals and also resist all the specific attacks (such as replay, MitM and redirection attacks), which prevent the protocol from achieving these goals.

```
goal
  secrecy_of sec_ki,sec_kg1
  authentication_on mtcd_mme
  authentication_on hss_mtcd
end goal
```

Figure 2. The goals of the proposed GSL-AKA protocol.

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/GSL_AKA.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.06s
  visitedNodes: 10 nodes
  depth: 9 plies
```

Figure 3. Result summarized by OFMC backend.

```
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL
PROTOCOL
  /home/span/span/testsuite/results/GSL_AKA.if
GOAL
  As Specified
BACKEND
  CL-AtSe
STATISTICS
  Analysed    : 14 states
  Reachable   : 6 states
  Translation: 0.01 seconds
  Computation: 0.00 seconds
```

Figure 4. Result summarized by CL-AtSe backend.

5. Security analysis

This section discusses the security properties of the proposed GSL-AKA protocol in terms of mutual authentication between protocol entities, key agreement between them, protection of the pre-shared secret keys, privacy preservation of the group and the MTCDs, and resistance against all the known attacks. Moreover, at the end of this section, we explain the reason why the previous SEGB-AKA protocol [18], contrary to its claims, fails to solve the single key problem and suffers from one DoS attack.

5.1. Security analysis of the proposed protocol

In this subsection, we analyze why the proposed protocol could achieve the defined security requirements and resist known attacks. In this security analysis, it is assumed that there is a secure channel between MME and HSS. Thus, an attacker can only access the channel between each MTCD to the group leader and the channel between the group leader to the MME, and he/she can eavesdrop and modify messages transmitted in these channels. Security analysis of the protocol is as follows:

- ***Basic security requirements***: In the proposed protocol, one entity authenticates another secondary entity by unique MACs sent from the secondary entity. The HSS authenticates the group and each $MTCD_{G_{1-i}}$ by verifying the MAC_{G_1} (Eq. (4)) sent from the group. In the same way, the MME authenticates them using the $XMAC_{G_1}$ (Eq. (13)). These two values are as a function of pre-shared secret keys and can be generated only by the corresponding MTCD and the group. Moreover, the group authenticates the HSS and MME by verifying MAC_{HSS} and MAC_{MEE} (Eqs. (10) and (15), respectively) sent from the network. These MACs are generated by using group temporary keys (TK_{G_1}). Thus, an adversary cannot generate them without knowing group temporary keys. Finally, for avoiding reuse of the MACs, the random numbers (TS_{G_1}, $RAND_{HSS}$ and $RAND_{MME}$) are embedded in their generation functions;

- ***Privacy preservation***: To preserve the privacy, the previous protocols encrypt identifiers with symmetric or asymmetric keys, but the proposed protocol uses temporary identifiers (TID_{G_1} and $TMSI_{G_{1-i}}$) without using any encryption operations. These two temporary identifiers get new values after each successful protocol process using one-way hash functions (Eqs. (1) and (2) show how these values are generated) and no one else can find out their original values from them. Thus, the proposed protocol preserves the privacy and due to the lack of encryption operations for preserving the privacy, it has very low computation overhead;
- ***Network signaling congestion prevention***: In the proposed protocol, each $MTCD_{G_{1-i}}$ generates its unique MAC ($MAC_{G_{1-i}}$ and $XMAC_{G_{1-i}}$, Eqs. (3) and (12), respectively) and sends them to $MTCD_{G_{1-leader}}$. For reducing signaling congestion and communication overhead, $MTCD_{G_{1-leader}}$ aggregates them into the one MAC (MAC_{G_1} and $XMAC_{G_1}$, Eqs. (4) and (13), respectively) and afterwards, sends them to the network. Moreover, the HSS and MME, for authenticating themselves to the group, use only one MAC (MAC_{HSS} and MAC_{MME}, Eqs. (10) and (15), respectively) per group instead of one MAC per each MTCD. Hence, the proposed protocol prevents the network from signaling congestion;
- ***Maintaining the unlinkability of the session keys***: After each successful running of the proposed protocol, the session keys between each $MTCD_{G_{1-i}}$ and the network ($K_{ASME}^{MTCD_{G_{1-i}}}$, $IK_{G_{1-i}}$, and $CK_{G_{1-i}}$) are updated as a function of random numbers ($RAND_{HSS}$). Then, when one of these session keys is revealed, an adversary cannot link it to its previous and next session keys;
- ***Maintaining the unlinkability of the group keys***: Whenever one MTCD wants to join or leave the group, the key of the group (K_{G_1}) is updated with the participation of all group members. The group key update process was explained in Section 3.2. Then, there is no way to link the current group key to the next or previous group keys;
- ***Solution to the single key problem***: The security of the symmetric key AKA protocols completely depends on the pre-shared secret keys and once these keys are compromised, all other secret data can be recovered and then an adversary can authenticate him/herself to the network. First, in the previous SEGB-AKA protocol paper [18], Parne et al. introduced a method for preserving pre-shared keys and solving the single key exposure problem. This method consists of two recommendations: First, the pre-shared secret keys should only be used as keys of MAC functions and key generators and never be used explicitly. Second, whenever an adversary discovers these pre-shared keys, he/she can never discover any session keys and authenticate him/herself to the network. One of the most important properties of the proposed protocol is that the proposed protocol can overcome the single key problem which all AKA protocols could not overcome. For overcoming this problem, first, in the proposed protocol, we use only hash functions ($f^s_{K_{G_{1-i}}}$, $f^1_{K_{G_{1-i}}}$, $f^3_{K_{G_{1-i}}}$, and $f^4_{K_{G_{1-i}}}$) during the protocol process in which the keys of these functions are embedded in the pre-shared secret key ($K_{G_{1-i}}$). Second, if an adversary compromises the pre-shared secret key ($K_{G_{1-i}}$, one of the inputs of the hash functions) and obtains the outputs of these functions by eavesdropping on the channel, he/she will never be able to obtain other inputs of the hash functions and compromise any other secret data and keys. In the proposed protocol, there are several secret data such as $IMSI_{G_{1-i}}$ of each $MTCD_{G_{1-i}}$ and ID_{G_1} and K_{G_1} of the group which are used only as one input of hash functions and never be revealed in any way (note that when the MTCDs and the group want to introduce themselves to the network and other entities, use temporary identifiers ($TMSI_{G_{1-i}}$ and TID_{G_1})). For this reason, the $IMSI_{G_{1-i}}$, ID_{G_1}, and K_{G_1} can be used as a key and secret data. Thus, whenever an adversary compromises the pre-shared secret key ($K_{G_{1-i}}$) without knowing other secret data (such as pre-mentioned $IMSI_{G_{1-i}}$, ID_{G_1}, and K_{G_1}), he/she can never generate message authentication codes (MAC_{G_1} and $XMAC_{G_1}$, Eqs. (4) and Eq. (13), respectively) and authenticate him/herself to the network. Thus, the proposed protocol can solve the single key exposure problem;
- ***Resistance against redirection attack***: In redirection attack, an adversary establishes a false base station to impersonate a legal MME and access users secure data. In the proposed protocol, the LAI of the connected base station is embedded into the aggregated authentication code (MAC_{G_1}, Eq. (4)) for avoiding redirection attack. Then, when the HSS computes MAC'_{G_1} using the LAI sent from MME and finds out that the MAC_{G_1} sent from MME is not equal to MAC'_{G_1}, it becomes aware of when the

attack occurs and rejects the authentication request;

- ***Resistance against MitM attack***: In the proposed protocol, the authentication codes of each MTCD ($MAC_{G_{1-i}}$ and $XMAC_{G_{1-i}}$, Eqs. (3) and (12), respectively) are generated using secret data such as pre-shared secret keys ($K_{G_{1-i}}$). Without knowing them, an adversary can never establish MitM attack and generate these authentication codes to authenticate itself instead of a legal MTCD to the network. Then, the protocol resists MitM attack;
- ***Resistance against impersonation attack***: The access security management entity key ($K_{ASME}^{MTCD_{G_{1-i}}}$, Eq. (9)) between each $MTCD_{G_{1-i}}$ and the network is generated using secret data and keys such as the pre-shared session key ($K_{G_{1-i}}$) and the group temporary key (TK_{G_1}). In this way, an adversary can never generate this key and modify and decrypt the communication messages between the $MTCD_{G_{1-i}}$ and the network. Moreover, an adversary can never generate the aggregated authentication codes (MAC_{G_1} and $XMAC_{G_1}$, Eqs. (4) and (13), respectively) and impersonate a legal group to the network;
- ***Resistance against replay attack***: In the proposed protocol, to resist replay attack, the random numbers or timestamps ($RAND_{HSS}$, $RAND_{MME}$, and TS_{G_1}) are embedded in the authentication codes of each MTCD ($MAC_{G_{1-i}}$ and $XMAC_{G_{1-i}}$, Eqs. (3) and (12), respectively) and the MACs of the HSS and MME (MAC_{HSS} and MAC_{MME}, Eqs. (10) and (15), respectively). Thus, these random numbers prevent these MACs from replaying and reusing;
- ***Resistance against DoS attack***: In the DoS attacks, an adversary, during the authentication procedure sends invalid data or prevents valid messages from reaching to the victim entity to disrupt its actions. The usual mechanism for resisting DoS attacks applied in the existing AKA protocols, such as GLARM-AKA [15], GR-AKA [16], and SEGB-AKA [18] protocols, is using Message Authentication Codes (MACs). In the proposed protocol, the group generates their MACs (MAC_{G_1} and $XMAC_{G_1}$, Eqs. (4) and (13), respectively) as a function of all the transmitted data and then, sends them to the network to authenticate their transmitted data to it. Moreover, the HSS and MME generate their MACs (MAC_{HSS} and MAC_{MME}, Eqs. (10) and (15), respectively) as a function of all the transmitted data and then send them to the group to authenticate them to the group. Hence, it is not possible to launch DoS attacks to the protocol. Moreover, in the proposed protocol, there is no trust between the MTCDs and the group leader. The pre-mentioned (MACs) are generated such that whenever one of the group members sends invalid data to the other group members, these can determine this malicious act and prevent it. For instance, as explained in Step 9 of the protocol procedure, if the group leader sends invalid data to the MTCDs, the network entities can find out this malicious act by computing $XMAC_{G_{1-i}}$ and $XMAC_{G_1}$ (Eqs. (12) and (13), respectively) and abort the authentication process. Or, whenever a malicious MTCD sends an invalid dataset such as invalid message authentication codes ($MAC_{G_{1-i}}$ and $XMAC_{G_{1-i}}$, Eqs. (3) and (12), respectively), the network entities in collaboration with the group leader can determine this malicious act (by calculating MAC_{G_1} and $XMAC_{G_1}$, Eqs. (4) and (13), respectively) and expel the malicious MTCD from the group.

5.2. Security problems of the SEGB-AKA

In this section, we briefly explain the security problems of the previous protocol, the SEGB-AKA protocol [18], and show that the protocol, contrary to its claims, could not solve the single key problem and suffer from one DoS attack.

Contrary to the claims of the SEGB-AKA protocol, it fails to overcome the single key problem. The SEGB-AKA protocol encrypts $IMSI_{G_{1-i}}$ of each $MTCD_{G_{1-i}}$ with symmetric keys ($SSDK_i$s). The symmetric keys in the SEGB-AKA protocol are updated after each successful protocol process by unique key identifiers (KID_is). The $SSDK_i$s represent a function of KID_i and pre-shared secret keys. Since the KID_is are transmitted in a non-encrypted manner through the wireless channels, if an adversary compromises the pre-shared secret key, he/she will be able to achieve the $SSDK_i$s and decrypt the identifiers and other data and compromise the session keys. Then, the SEGB-AKA protocol fails to overcome the single key problem in the communication networks.

Moreover, the SEGB-AKA protocol is also vulnerable to one DoS attack. In this protocol, the symmetric keys for encryption ($SSDK_i$s) are updated after each successful protocol process using the unique key identifiers (KID_is). In this method, HSS sends encrypted new KID_i to each MTCD and then the MTCD decrypts KID_i and generates new $SSDK_i$ as a function of it. However, in the SEGB-AKA protocol, there is no mechanism to confirm the accuracy of new KID_is. In this way, whenever an adversary changes the encrypted new KID_i, the MTCD cannot find out that this value has been changed. Then, the MTCD decrypts wrong KID_i and generates invalid new $SSDK_i$ and thus, it can never run the protocol again.

The comparative security analysis between the

Table 2. Security analysis between group-based AKA protocols.

Security Properties	Group-based authentication and key agreement protocols											
	G-AKA [8]	SE-AKA [9]	EG-AKA [10]	NOVEL-AKA [11]	GROUP-AKA [12]	GBAAM-AKA [13]	PRIVACY-AKA [14]	GLARM-AKA [15]	GR-AKA [16]	GBS-AKA [17]	SEGB-AKA [18]	GSL-AKA
SP1	Symmetric	Hybrid	Hybrid	Symmetric	Symmetric	Asymmetric	Hybrid	Symmetric	Asymmetric	-	Symmetric	-
SP2	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SP3	Yes	No	No	Yes	No	No	No	Yes	No	Yes	Yes	Yes
SP4	No	Yes	Yes	Yes	No	No	Yes	No	Yes	No	Yes	Yes
SP5	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SP6	No	Yes	No	Yes	Yes	Yes	No	Yes	Yes	No	Yes	Yes
SP7	No	No	No	No	Yes	No	No	No	Yes	No	Yes	Yes
SP8	No	No	No	No	No	No	No	No	No	No	No	Yes
SP9	No	Yes	No	No	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
SP10	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SP11	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	Yes
SP12	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SP13	No	Yes	No	No	No	Yes	Yes	Yes	Yes	No	No	Yes

Note: SP1: Type of cryptosystem; SP2: Basic security requirements; SP3: Follow the 3GPP standard; SP4: Privacy preservation; SP5: Network signaling congestion prevention; SP6: Maintenance the unlinkability of session keys; SP7: Maintenance the unlinkability of group keys; SP8: Solution to the single key problem; SP9: Resistance against redirection attack; SP10: Resistance against MitM attack; SP11: Resistance against impersonation attack; SP12: Resistance against replay attack; SP13: Resistance against DoS attack.

proposed protocol and the previous group-based AKA protocols is shown in Table 2. It is observed that the proposed protocol is able to achieve all the security goals without using any symmetric or asymmetric encryption operations.

6. Performance analysis

In this section, our proposed GSL-AKA protocol is compared with the existing group-based AKA protocols in terms of communication and computation overheads and shows that the proposed protocol has the lowest overheads. To evaluate the mentioned overheads, let there be n number of MTCDs in m groups. Note that since the overheads for creating groups, joining or leaving groups and distributing the group keys are negligible (these are shown in [21,22]), we ignored them in computing the total overheads of the group-based AKA protocols.

6.1. Communication overhead

The total bits transmitted in protocol process is the communication overhead of protocol. According to Figure 1 and Table 1, the communication overhead per each message of the proposed protocol can be calculated as follows:

$$M_1 = (MAC_{G_{1-i}}) = 64 * n,$$

$$M_2 = (AUTH_{G_1}) = 128 * n + 256 * m,$$

$$M_3 = (AUTH_{G_1}||LAI) = 128 * n + 296 * m,$$

$$M_4 = (TMSI_{G_{1-i}}||K_{ASME}^{MTCD_{G_{1-i}}}||GAV) = 384 * n,$$

$$+560 * m,$$

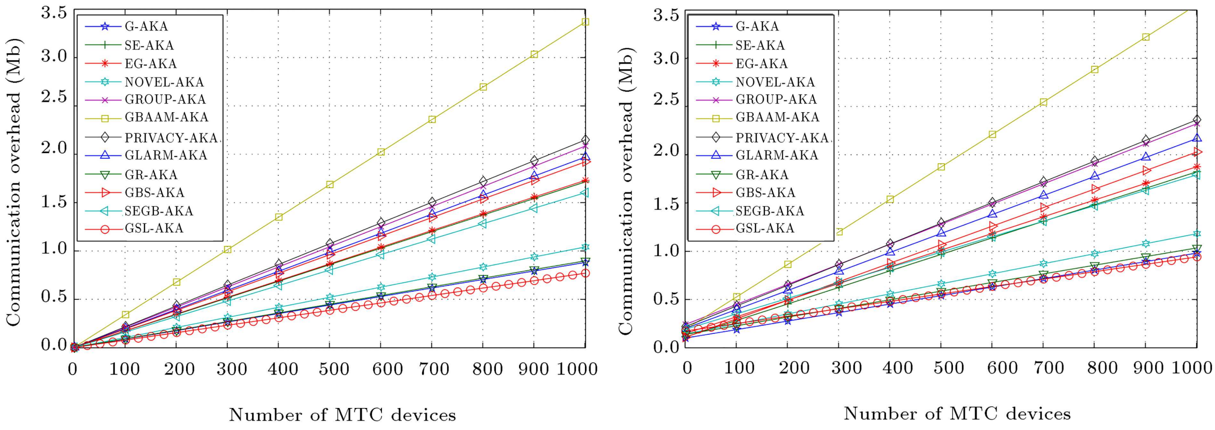
$$M_5 = (AUTH_{MME}) = 432 * m,$$

$$M_6 = (RAND_{HSS}) = 128 * m,$$

$$M_7 = (XMAC_{G_{1-i}}) = 64 * n,$$

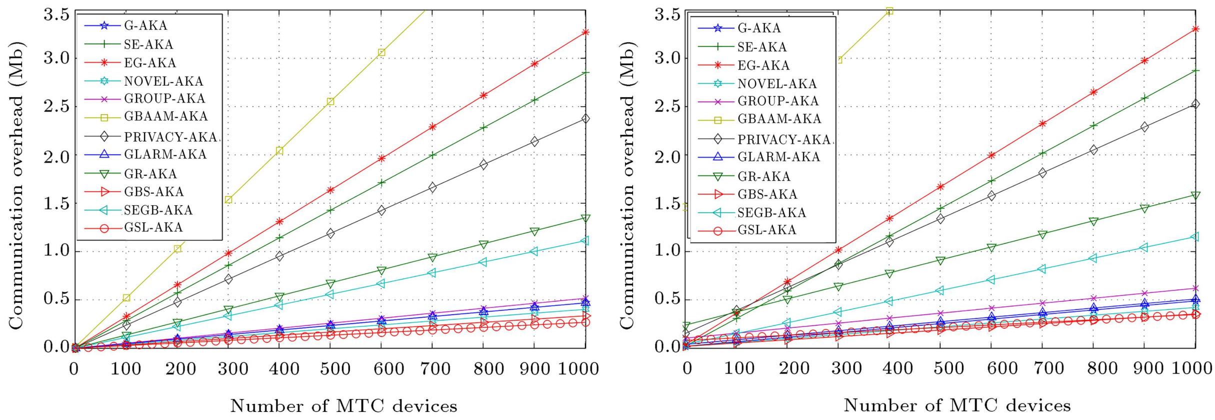
$$M_8 = (XMAC_{G_1}) = 64 * m.$$

Then, the total communication overhead of the proposed protocol is the sum of the above calculated overheads and equal to $768 * n + 1736 * m$. The communication overhead of other group-based AKA protocols were calculated in [18], like our calculation



(a) (b)

Figure 5. Comparison of the communication overheads: (a) $m = 1$ and (b) $m = 100$.



(a) (b)

Figure 6. Comparison of the computation overheads: (a) $m = 1$ and (b) $m = 100$.

method. The comparative analysis of communication overhead of existing group-based AKA protocols is illustrated in Figure 5. It is observed that the proposed protocol has the lowest communication overhead of all other group AKA protocols.

6.2. Computation overhead

The total computation overhead of the proposed protocol can be calculated by considering the execution time of the applied cryptographic functions in terms of n and m. The execution time of cryptographic functions is presented in [12,16]. The computation overhead of the proposed protocol at the MTCDs is equal to:

$$(2T_{hash}) * n + (4T_{hash}) * m,$$

and at the network is:

$$(2T_{hash}) * n + (4T_{hash}) * m.$$

Thus, the total computation overhead is equal to:

$$(4T_{hash}) * n + (8T_{hash}) * m.$$

Moreover, the computation overheads of other protocols were calculated in [18], the same as our calculation method. Figure 6 illustrates the comparative analysis of computation overhead of the existing group AKA protocols. It can be seen that the proposed GSL-AKA protocol has the lowest computation overhead as it only uses hash functions during the authentication process.

7. Proving of the relative lowest overheads

This section illustrates the reason why the communication and computation overheads of the proposed GSL-AKA protocol, in comparison with the previous group-based AKA protocols, are the lowest. This illustration proves that the proposed protocol, by meeting all the security requirements mentioned in Section 5, is able to achieve the lowest overheads calculated in Section 6. Thus, this section combines the results of the two previous sections.

7.1. Communication overhead

In this part, we discuss why the communication overhead of the proposed protocol is the lowest. As mentioned in the previous section, the communication overhead of one protocol is the total bits transmitted during its process. The total communication overhead of the proposed protocol is divided into three different categories. First, in the proposed protocol, the group members in Step 3 embed their identifiers ($TMSI_{G_{1-i}}$ and TID_{G_1}) into the $AUTH_{G_1}$ message and then, send them to the network to introduce themselves to it. Second, at the end of Step 7, the HSS sends new session keys ($K_{ASME}^{MTCD_{G_{1-i}}}$) and new identifiers ($TID_{G_1}$ and $TMSI_{G_{1-i}}$) to the MME to communicate this entity independently with the group. Moreover, the third category is that the authentication challenges messages (MAC_{G_i}, $XMAC_{G_i}$, MAC_{HSS}, and MAC_{MME}) and their random inputs (TS_{G_1}, $RAND_{HSS}$, and $RAND_{MME}$) which are transmitted in the channel for authenticating mutually the group with the HSS and MME.

The first and second categories are necessary to transmit and the communication overhead of all AKA protocols includes these two categories. In fact, how to transfer these categories between the network entities and size of them are standardized by the 3GPP committee. In this way, these must be the same as the standard protocol for 4G networks, EPS-AKA [23]. The size of these values is mentioned in Table 1 and Figure 1 shows how these are transmitted between network entities.

About the third category, each of the AKA protocols uses their own authentication method to authenticate the network entities mutually. The first security requirement considered in this type of AKA protocols is mutual authentication between the group and the network entities (MME and HSS). Thus, there are at least four challenges applied in these protocols that each entities by solving one of them and declaring the answer, authenticates itself to the others. Each of the AKA protocols use some cryptographic functions for implementing its challenges and the proposed protocol uses only hash functions to implement them. In the proposed GSL-AKA protocol, there are existing four MACs ($MAC_{G_{1-i}}$, $XMAC_{G_{1-i}}$, MAC_{HSS}, and MAC_{MEE}) and three random numbers (TS_{G_1},$RAND_{HSS}$, and $RAND_{MME}$) that each entity by generating one of them and sending it with its random number to the others, authenticates itself (The generator entity of each MACs and these transmission method are shown in Figure 1). According to the 3GPP standards of the AKA parameters size (some of them and their size are mentioned in Table 1), the hash functions have the lowest communication overhead during its process, in comparison with existing authentication methods. Moreover, to reduce communication overhead of the authentication process, $MTCD_{G_{1-leader}}$ aggregates all the authentication codes ($MAC_{G_{1-i}}$ and $XMAC_{G_{1-i}}$) into one authentication code (MAC_{G_1} and $XMAC_{G_1}$, Eqs. (4) and (13), respectively) and then sends them to the network. Thus, the communication overhead of the authentication process in the proposed protocol is the lowest.

Finally, for achieving all the defined security requirements for the group-based AKA protocols (these security requirements are mentioned in Section 5), most of the other group-based AKA protocols transmit additional data categories and so, the communication overhead of them is increased. However, the proposed protocol transmits only those three categories and for achieving the defined security requirements, there is no need to transmit additional data. In Sections 4 and 5, it was proven that the proposed protocol could achieve all the defined security requirements. In this way, the communication overhead of the proposed protocol is equal to those three categories. Then, according to the previous paragraph contents, the communication overhead of the proposed protocol is the lowest. In other words, the proposed protocol, by meeting all the security requirements mentioned in Section 5, is able to achieve the lowest communication overhead.

7.2. Computation overhead

As mentioned in the previous subsection, to satisfy the first security requirement defined in Section 5, all the group-based AKA protocols must mutually authenticate the group with the network entities (HSS and MME). Thus, in these protocols, some challenges are considered, that is, by solving one of these challenges and presenting the response, each entity authenticates itself to the others (this process is known as challenge-response process). Thus, in the group-based AKA protocols, there are at least 4 challenges for performing the authentication mechanism. In all of the group-based AKA protocols, the existence of the challenge-response process is required and each of them performs it in different ways. The proposed protocol uses hash functions to authenticate mutually each entity. Therefore, there are four message authentication codes ($MAC_{G_{1-i}}$, $XMAC_{G_{1-i}}$, MAC_{HSS}, and MAC_{MEE}) and their random numbers (TS_{G_1}, $RAND_{HSS}$, and $RAND_{MME}$) applied as authentication challenges (in Section 5.1, the authentication process for each entity is fully explained). Since hash functions are the fastest cryptographic function in comparison with other existing cryptographic functions (the execution time of cryptographic functions is presented in [12,16]), the computation overhead of the challenge-response process in the proposed protocol is the lowest.

In addition, to satisfy another security requirements mentioned in Section 5, each protocol uses other cryptographic functions such as symmetric or asymmetric encryption during its run, thereby increasing their computation overhead. However, due to the structural features of the proposed protocol, there is no need to use any other additional cryptographic functions, and this protocol without using any other cryptographic functions could achieve all the defined security requirements (this is proved in Section 4 and Section 5.1). Thus, the total computation overhead of the proposed protocol is the same computation overhead of its challenge-response process mentioned in the previous paragraph. Moreover, since the pre-mentioned challenge response overhead of the proposed protocol is the lowest of the other existing authentication methods, the total computation overhead of the proposed protocol is the lowest. That is, none of the previous group-based AKA protocols, by using existing cryptographic functions, could achieve computation overhead less than our protocol computation overhead.

8. Conclusion and future work

In this paper, the Group-based Secure Lightweight Authentication and Key Agreement (GSL-AKA) protocol for Machine-to-Machine (M2M) communication in Long-Term Evolution (LTE) and 5G networks was proposed. Compared with the prior group-based Authentication and Key Agreement (AKA), the GSL-AKA protocol could achieve all the security goals, overcome all the known attacks, preserve the privacy of the MTC Devices (MTCDs), and overcome the single key problem which the previous group-based AKA protocols could not solve. Moreover, our performance analysis showed that the proposed protocol had the best communication and computation overheads and these were the lowest.

In the machine-type communication, when a group of MTCDs move to the coverage of new eNB simultaneously, they must re-authenticate to the network. If the existing group-based AKA protocols are used to re-authenticate, the network will face long delays and huge communication and computation overheads. For this reason, it is necessary to propose a robust group-based handover authentication protocol for re-authenticating a group of MTCDs simultaneously in handover scenarios. Therefore, our future work is proposing the group-based secure lightweight handover authentication GSLHA protocol for M2M communication which is able to achieve all the security goals and has the lowest network overheads.

References

1. Paul, P.V., Victer Paul, P., and Saraswathi, R. "The internet of things - A comprehensive survey", *2017 International Conference on Computation of Power, Energy Information and Commuincation (ICCPEIC)* (2017).
2. Ghavimi, F. and Chen, H. "M2M communications in 3GPP LTE/LTE-A networks: architectures, service erquirements, challenges, and applications", *IEEE Communications Surveys Tutorials*, **17**(2), pp. 525–549 (2015).
3. Żółkiewski, S. and Galuszka, K. "Remote control of industry robots using mobile devices", *New Contributions in Information Systems and Technologies*, pp. 323–332 (2015).
4. Roychoudhury, P., Roychoudhury, B., and Saikia, D.K. "Hierarchical group based mutual authentication and key agreement for a machine type communication in LTE and future 5G networks", *Security and Communication Networks*, **2017** (2017).
5. 3GPP. Technical Specification Group Services and System Aspects; Security Aspects of Machine-Type Communications (MTC) (Release 11), document 3GPP TR 33.868 V.0.7.0, 3rd Generation Partnership Project (3GPP), Valbonne, France (2012).
6. Lai, C., Lu, R., Zheng, D., et al. "Toward secure large-scale machine-to-machine communications in 3GPP networks: challenges and solutions", *IEEE Commun. Mag.*, **53**(12), pp. 12–19 (2015).
7. Modiri, M.M., Mohajeri, J., and Salmasizadeh, M. "GSL-AKA: group-based secure lightweight authentication and key agreement protocol for M2M communication", *2018 9th International Symposium on Telecommunications (IST)*, pp. 275–280 (2018).
8. Chen, Y.-W., Wang, J.-T., Chi, K.-H., et al. "Group-based authentication and key agreement", *Wireless Personal Communications*, **62**(4), pp. 965–979 (2012).
9. Lai, C., Li, H., Lu, R., et al. "SE-AKA: A secure and efficient group authentication and key agreement protocol for LTE networks", *Computer Networks*, **57**(17), pp. 3492–3510 (2013).
10. Jiang, R., Lai, C., Luo, J., et al. "EAP-based group authentication and key agreement protocol for machine-type communications", *Int. J. Distrib. Sens. Netw.*, **9**(11), p. 304601 (2013).
11. Lai, C., Li, H., Li, X., et al. "A novel group access authentication and key agreement protocol for machine-type communication", *Transactions on Emerging Telecommunications Technologies*, **26**(3), pp. 414–431 (2015).
12. Choi, D., Choi, H.-K., and Lee, S.-Y. "A group-based security protocol for machine-type communications in LTE-advanced", *Wireless Networks*, **21**(2), pp. 405–419 (2015).
13. Cao, J., Ma, M., and Li, H. "GBAAM: group-based access authentication for MTC in LTE networks", *Secur. Commun. Netw.*, **8**(17), pp. 3282–3299 (2015).

14. Fu, A., Song, J., Li, S., et al. "A privacy-preserving group authentication protocol for machine-type communication in LTE/LTE-A networks", *Security and Communication Networks*, **9**, pp. 2002–2014 (2016).

15. Lai, C., Lu, R., Zheng, D., et al. "GLARM: Group-based lightweight authentication scheme for resource-constrained machine to machine communications", *Computer Networks*, **99**, pp. 66–81 (2016).

16. Li, J., Wen, M., and Zhang, T. "Group-based authentication and key agreement with dynamic policy updating for MTC in LTE-A networks", *IEEE Internet of Things Journal*, **3**(3), pp. 408–417 (2016).

17. Yao, J., Wang, T., Chen, M., et al. "GBS- AKA: group-based secure authentication and key agreement for M2M in 4G network", *2016 International Conference on Cloud Computing Research and Innovations (ICCCRI)*, pp. 42–48 (2016).

18. Parne, B.L., Gupta, S., and Chaudhari, N.S. "SEGB: Security enhanced group based AKA protocol for M2M communication in an IoT enabled LTE/LTE-A network", *IEEE Access*, **6**, pp. 3668–3684 (2018).

19. 3GPP. Technical Specification Group Services and System Aspects; Security architecture and procedures for 5G system. TS 33.501 V.15.0.0, 3rd Generation Partnership Project (3GPP) (2018).

20. 3GPP. Security architecture and procedures for 5G System. TS 33.501 V.0.1.0, 3rd Generation Partnership Project (3GPP) (2017).

21. Moyer, M.J., Rao, J.R., and Rohatgi, P. "A survey of security issues in multicast communications", *IEEE Netw.*, **13**(6), pp. 12–23 (1999).

22. Rafaeli, S. and Hutchison, D. "A survey of key management for secure group communication", *ACM Comput. Surv.*, **35**(3), pp. 309–329 (2003).

23. 3GPP. 3G security; Security architecture. TS 33.102 V14.1.0, 3rd Generation Partnership Project (3GPP) (2017).

24. HLPSL. The High Level Protocol Specification Language. [On-line]. Available: http://www.avispa-project.org/delivs/2.1/d2-1.pdf.

25. AVISPA. Automated Validation of Internet Security Protocols. [Online]. Available: http://www.avispa-project.org.

Appendix A

HLPSL codes of the proposed GSL-AKA protocol

The basic roles of the MTCDs, HSS, and MME of the proposed protocol in HLPSL language are illustrated in Figures A.1, A.2, and A.3 respectively. Note that the *SymmetricKey* parameter mentioned in these roles is used to provide security for channels between HSS and MME.

```
role device(
  D,M,H                                   :agent,
  SND,RCV                                 :channel(dy),
  KG1,Ki                                  :symmetric_key,
  TSg1,Access_Request_Message,
  Request_Identity_Message,
  IDG1,IDi,Rhss,Rmme,Amf                  :text,
  F1,F2,F3,F4,KDF                         :hash_func)
played_by D
def=
  local
    State                                 :nat,
    LAI,TIDG1,TIDi                        :text
  const
    sec_ki,sec_kg1,
    mtcd_mme,hss_mtcd                     :protocol_id,
    success                               :text
  init State := 0
  transition
    1. State  = 0 /\  RCV(start)=|>
       State':= 1 /\  SND(Access_Request_Message)
    2. State  = 1 /\  RCV(Request_Identity_Message
                      )=|>
       State':= 2 /\  TSg1':=new()
                  /\  SND(TIDG1.TIDi.F1(KG1.F1(Ki.
                      IDG1.IDi.TSg1').LAI).TSg1')
                  /\  secret(Ki,sec_ki,{D,H})
                  /\  secret(IDi,sec_ki,{D,H})
                  /\  secret(KG1,sec_kg1,{D,H})
                  /\  secret(IDG1,sec_kg1,{D,H})
                  /\  witness(D,H,hss_mtcd,TSg1')
    3. State  = 2 /\  RCV(F1(F2(KG1.IDG1.Rhss').
                      F1(F2(KG1.IDG1.Rhss').Rhss'.
                      Amf).Rmme').Rmme'.F1(F2(KG1.
                      IDG1.Rhss').Rhss'.Amf).Rhss'
                      .Amf)=|>
       State':= 3 /\  SND(F1(KG1.F1(Ki.IDG1.IDi.
                      Rhss')))
                  /\  witness(D,M,mtcd_mme,Rhss')
                  /\  request(D,H,hss_mtcd,Rhss')
                  /\  request(D,M,mtcd_mme,Rmme')
end role
```

Figure A.1. The role of the MTCDs.

```
role hss(
  H,M,D                                   :agent,
  SND,RCV                                 :channel(dy),
  KG1,Ki,SymmetricKey                     :symmetric_key,
  TSg1,Access_Request_Message,
  Request_Identity_Message,
  IDG1,IDi,Rhss,Rmme,Amf                  :text,
  F1,F2,F3,F4,KDF                         :hash_func)
played_by H
def=
  local
    State                                 :nat,
    LAI,TIDG1,TIDi                        :text
  const
    sec_ki,sec_kg1,
    mtcd_mme,hss_mtcd                     :protocol_id,
    success                               :text
  init State := 0
  transition
    1. State  = 0 /\  RCV({TIDG1.TIDi.F1(F1(KG1.F1(
                      Ki.IDG1.IDi.TSg1')).LAI').
                      TSg1'.LAI'}_SymmetricKey)=|>
       State':= 1 /\  Rhss':=new()
                  /\  SND({F5(Ki.IDi.Rhss').KDF(F2
                      (KG1.IDG1.Rhss').F3(Ki.IDi.
                      Rhss').F4(Ki.IDi.Rhss').IDG1
                      .IDi).F1(F2(KG1.IDG1.Rhss')
                      .Rhss'.Amf).Rhss'.Amf.F1(KG1
                      .F1(Ki.IDG1.IDi.Rhss')).F2(
                      KG1.IDG1.Rhss')}_SymmetricKey)
                  /\  witness(H,D,hss_mtcd,Rhss')
                  /\  request(H,D,hss_mtcd,TSg1')
    2. State  = 1 /\  RCV(success)=|>
       State':= 2
end role
```

Figure A.2. The role of the HSS.

```
role mme(
  M,H,D                                     :agent,
  SND,RCV                                   :channel(dy),
  KG1,Ki,SymmetricKey                       :symmetric_key,
  TSg1,Access_Request_Message,
  Request_Identity_Message,
  IDG1,IDi,Rhss,Rmme,Amf                    :text,
  F1,F2,F3,F4,KDF                           :hash_func)
played_by M
def=
  local
    State                                   :nat,
    LAI,TIDG1,TIDi                          :text
  const
    sec_ki,sec_kg1,
    mtcd_mme,hss_mtcd                       :protocol_id,
    success                                 :text
  init State := 0
  transition
    1. State   = 0 /\  RCV(Access_Request_Message
                       )=|>
       State':= 1 /\  SND(Request_Identity_Message)
    2. State   = 1 /\  RCV(TIDG1.TIDi.F1(KG1.F1(Ki.
                       IDG1.IDi.TSg1').LAI).TSg1')=|>
       State':= 2 /\  LAI':=new()
                  /\  SND({TIDG1.TIDi.F1(F1(KG1.F1(
                       Ki.IDG1.IDi.TSg1')).LAI').
                       TSg1'.LAI'}_SymmetricKey)
                  /\  secret(Ki,sec_ki,{M,H})
                  /\  secret(IDi,sec_ki,{M,H})
                  /\  secret(KG1,sec_kg1,{M,H})
                  /\  secret(IDG1,sec_kg1,{M,H})
    3. State   = 2 /\  RCV({F5(Ki.IDi.Rhss').KDF(F2(
                       KG1.IDG1.Rhss').F3(Ki.IDi.
                       Rhss').F4(Ki.IDi.Rhss').IDG1.
                       IDi).F1(F2(KG1.IDG1.Rhss').
                       Rhss'.Amf).Rhss'.Amf.F1(KG1.
                       F1(Ki.IDG1.IDi.Rhss')).F2(
                       KG1.IDG1.Rhss')}_SymmetricKey
                       )=|>
       State':= 3 /\  Rmme':=new()
                  /\  SND(F1(F2(KG1.IDG1.Rhss').F1
                       (F2(KG1.IDG1.Rhss').Rhss'.Amf
                       ).Rmme').Rmme'.F1(F2(KG1.IDG1
                       .Rhss').Rhss'.Amf).Rhss'.Amf)
                  /\  witness(M,D,mtcd_mme,Rmme')
    4. State   = 3 /\  RCV(F1(KG1.F1(Ki.IDG1.IDi.
                       Rhss')))=|>
       State':= 4 /\  request(M,D,mtcd_mme,Rhss')
                  /\  SND(success)
end role
```

Figure A.3. The role of the MME.

Biographies

Mohammad Mahdi Modiri received his BS degree in Telecommunications Engineering from University of Tehran, Tehran, Iran in 2016. In addition, he received his MS degree in Secure Communication and Cryptography Engineering from Sharif University of Technology, Tehran Iran in 2018. He is currently working toward his PhD degree at Electrical Engineering Department of Sharif University of Technology in the field of Secure Communication and Cryptography Engineering. His research interests include network security, Internet of Things (IoT), and cryptography.

Javad Mohajeri is an Assistant Professor at the Electronics Research Institute, Sharif University of Technology, Tehran, Iran, where he is an Adjunct Assistant Professor at the Electrical Engineering Department. He has authored or co-authored 3 books and 115 research articles in refereed journals/conferences. His current research interests include data security and design and analysis of cryptographic protocols and algorithms. Dr. Mohajeri is a Founding Member of the Iranian Society of Cryptology. Also, he has been the committee program chair of second International ISC Conference on Information Security and Cryptology, and committee program member of third - 17th of this conference.

Mahmoud Salmasizadeh received the BS and MS degrees in Electrical Engineering from Sharif University of Technology, Tehran, Iran in 1972 and 1989, respectively. He also received the PhD degree in Information Technology from Queensland University of Technology, Australia in 1997. Currently, he is an Associate Professor at the Electronics Research Institute and an adjunct associate professor at the Electrical Engineering Department, Sharif University of Technology. His research interests include design and cryptanalysis of cryptographic algorithms and protocols, e-commerce security, and information theoretic secrecy. He is a founding member of Iranian Society of Cryptology.