A Novel Group-based Secure Lightweight Authentication and Key Agreement Protocol for Machine-Type Communication

M.M. Modiri a,∗, J. Mohajeri b, M. Salmasizadeh b

aDepartment of Electrical Engineering, Sharif University of Technology, Tehran, Iran
bElectronics Research Institute, Sharif University of Technology, Tehran, Iran

Abstract

Nowadays, one of the most important criterions 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 mutual authentication of the devices and the network and 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 suffer from various security and performance problems. This paper 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 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 in comparison with the other existing group-based AKA protocols.

Key words: IoT, Network Security, M2M Communication, Group-based AKA, AVISPA.

1. Introduction

Wireless telecommunication technology is constantly growing. Currently, the telecommunication technology is developed to provide wide and secure communications between things with different applications around the world. Moreover, full support of the 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 by 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 LTE and 5G networks includes three communication parties: Home Subscriber Server (HSS), Mobile Management Entity (MME) and MTCDs. The roles of each entities and how to communicate between them are fully explained in [4]. The group-based authentication and key agreement (AKA) protocols for M2M communication, authenticate mutually a group of MTCDs and the network entities (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 from eavesdropping and modifying the messages transmitted in 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 to the network. In addition, if each MTCD perform the AKA protocol individually, the network will face to large congestions and huge communication and computation overheads [6]. Hence, it is necessary to propose a group-based secure and lightweight authentication and key agreement protocol for M2M communication.

Various group-based AKA protocols have been proposed. In this paper, we propose the group-based secure lightweight GSL-AKA protocol. The proposed protocol
overcomes the security and performance problems existing in the previous protocols and resists against 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 in comparison with 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] which was added to it some other contents such as formal security verification, full discussion of security properties, how to calculate the network overheads and so on.

The remainder of this paper is organized as follows: In Section 2, we discuss 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. In Section 4, we illustrate the formal security analysis of the protocol using AVISPA tool. Section 5 analyzes the security properties of the proposed protocol and compares it with the other group-based AKA protocols. In Section 6, we illustrate the comparative performance analysis of the proposed protocol with existing group-based AKA protocols and show that the network overheads of the proposed protocol is the lowest. Then, in Section 7, we discuss and prove 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, in Section 8, we draw our conclusion and future work.

2. Related Works

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

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 first authenticated device to authenticate the rest of devices. Thus, the AKA process can be simplified for all remaining devices in the group. However, the protocol produces the signaling congestion when a mass of MTCDs simultaneously require 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 EC-AKA for non-3GPP networks. These protocols resist against mentioned attacks but due to asymmetric key operations, suffer from high computation overhead.

To reduce computation overhead, Lai et al. [11] proposed the symmetric key based NOVEL-AKA protocol. However, it suffers from some other problems such as network signaling congestion and also is vulnerable to DoS and redirection attacks. Moreover, Choi et al. [12] proposed the GROUP-AKA protocol that successfully reduces signaling congestion when a mass of MTCDs simultaneously require to access to the network and maintains the unlinkability of the group key. However, the protocol suffers from 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 also fails to preserve the privacy. For preserving of the privacy, Fu et al. [14] proposed the PRIVACY-AKA protocol used asymmetric cryptography. The protocol resists against 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 is preserving 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 against attacks and reduce communication overhead. However, it fails to preserve the privacy of MTCDs and suffers from 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 SEG-B-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 to/from 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 above-identified security and non-security problems, we propose GSL-AKA protocol for M2M communication. The proposed protocol has the same structure as its two previous protocols, GBS-AKA [17] and SEG-AKA [18] and is able to improve their properties. Moreover, the proposed protocol successfully overcomes known security and non-security problems and preserves privacy of the 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 in comparison with other protocols. Finally, the proposed protocol is able to overcome the single key problem which the previous group-based AKA protocols could not solve it.
the group “G\textsubscript{1}”. Note that, in these symbols, the notation and symbols used in the protocol are presented in Table 1. Notations and Symbols. 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 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 issues are 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 (T\textsubscript{MSI}\textsubscript{G\textsubscript{1−i}}) and group temporary ID (T\textsubscript{ID}\textsubscript{G\textsubscript{i}}) in wireless channels, instead of their original values. These two temporary values are used for preserving the identifiers and can be generated only by corresponding MTCD and the HSS. The group leader transfers and receives data to/from each group member 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 find out and the process of authentication will abort.

The proposed protocol consists of two phases: i) Group initialization and key establishment phase, 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\textsubscript{i} represents the group “G\textsubscript{i}” and the notation G\textsubscript{1−i} represents the i\textsuperscript{th} member of the group “G\textsubscript{1}”.

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 member 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 find out and the process of authentication will abort.

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\textsubscript{G\textsubscript{1−i}} is the pre-shared secret key between each MTCD and HSS.
- The key generation center (KGC) generates the secure group key (K\textsubscript{G\textsubscript{i}}) and unique group identifier (ID\textsubscript{G\textsubscript{i}}) for the group.
- We introduced a temporary mobile subscriber identifier (T\textsubscript{MSI}\textsubscript{G\textsubscript{1−i}}) and a group temporary identifier (T\textsubscript{ID}\textsubscript{G\textsubscript{i}}), which are used to preserve IMSI\textsubscript{G\textsubscript{1−i}} and ID\textsubscript{G\textsubscript{i}}.
- The symbol f\textsuperscript{1}(\cdot) is a hash-based MAC generation function and f\textsuperscript{2}(\cdot), f\textsuperscript{3}(\cdot) and f\textsuperscript{4}(\cdot) are key generation functions. The structure of these functions are explained in [19]-[20]. Note that the key used in these functions is written as a subscript, e.g., f\textsubscript{K\textsubscript{G\textsubscript{1−i}}}(\cdot) applies K\textsubscript{G\textsubscript{1−i}} for generating MACs.
- Moreover, f\textsuperscript{*}(\cdot) 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.

3.1. Pre-shared Parameters

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 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 issues are out of scopes for our work. These issues are 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 (T\textsubscript{MSI}\textsubscript{G\textsubscript{1−i}}) and group temporary ID (T\textsubscript{ID}\textsubscript{G\textsubscript{i}}) in wireless channels, instead of their original values. These two temporary values are used for preserving the identifiers and can be generated only by corresponding MTCD and the HSS. At the time of the group initialization, T\textsubscript{MSI}\textsubscript{G\textsubscript{1−i}} and T\textsubscript{ID}\textsubscript{G\textsubscript{i}} get a certain initial value and after each successful protocol process, these are updated as follows:

\[ T\textsubscript{ID}\textsuperscript{new}_{G\textsubscript{i}} = f\textsuperscript{*}_{K\textsubscript{G\textsubscript{i}}}(ID\textsubscript{G\textsubscript{i}}||RAND_{HSS}) \]  

(1)
After acquiring $TMSI_{G_{1-i}}$, assigned to $ID_{G_{1-i}}$ 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 (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 ($RAND_{HSS}$):
  \[ TK_{G_{1}} = f_{K_{G_{1}}}^1(ID_{G_{1}} || RAND_{HSS}) \]  

- Then, HSS computes the group temporary key as:
  \[ IK_{G_{1-i}} = f_{K_{G_{1}}}^3(IMSI_{G_{1-i}} || RAND_{HSS}) \]  

- According to the 3GPP standards of key derivation [23], the HSS computes the integrity and cipher keys of each $MTCD_{G_{1-i}}$ as:
  \[ CK_{G_{1-i}} = f_{K_{G_{1-i}}}^1(IMSI_{G_{1-i}} || RAND_{HSS}) \]

- And then, it computes the session key by using KDF (Key Derivation Function) for each $MTCD_{G_{1-i}}$ from those keys ($IK_{G_{1-i}}, CK_{G_{1-i}}$) as:
  \[ K_{ASME}^{MTCD_{G_{1-i}}} = KDF(TK_{G_{1}} || IK_{G_{1-i}} || ID_{G_{1}} || IMSI_{G_{1-i}}) \]

**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:
  \[ MAC_{HSS} = f_{TK_{G_{1}}}^1(RAND_{HSS} || AMF) \]  

- HSS generates $AUTH_{HSS}$ as:
  \[ AUTH_{HSS} = (MAC_{HSS} || RAND_{HSS} || AMF) \]

- HSS computes the respective authentication code for each $MTCD_{G_{1-i}}$ as:
  \[ XMAC_{G_{1-i}} = f_{K_{G_{1-i}}}^1(ID_{G_{1}} || IMSI_{G_{1-i}}) \]  

- HSS generates the aggregated respective authentication code for the group as:
  \[ XMAC_{G_{1}} = f_{K_{G_{1}}}^1(XMAC_{G_{1-i}} || ... || XMAC_{G_{1-i}}) \]
- HSS assigns a new $TID_{G_1}$ for $G_1$ using (1) and a new $TMSI_{G_1-i}$ for each $MTCD_{G_1-i}$, using (2) for establishing further communications.

- Later, the HSS generates the group authentication vector (GAV) from the above computed parameters.

$$GAV = (AUTH_{HSS} || MAC_{G_1} || TID_{G_1} || TK_{G_1})$$

- Finally, HSS transmits the authentication response message $(TMSI_{G_1-i} || K_{ASME}^{MTCD_{G_1-i}} || GAV)$

Fig. 1. The GSL-AKA protocol
Step8: After receiving the authentication response message $TMSG_{G_{1-1}}||K_{ASME}^{MTCD_{G_{1-1}}}||GAV$:

- MME stores the authentication response message for communicating with MTCDs and further authentication processes.
- Then, MME generates a $RAND_{MME}$ and calculates the $MAC_{MME}$ as:
  $$MAC_{MME} = f^2_{TRG_1}(MAC_{HSS}||RAND_{MME})(15)$$
- Later, MME generates authentication token as:
  $$AUTH_{MME} = (MAC_{MME}||RAND_{MME}||MAC_{HSS}||RAND_{HSS}||AMF)$$
- Finally, it sends $AUTH_{MME}$ to the $MTCD_{G_{1-leader}}$.

Step9: After acquiring $AUTH_{MME}$, the $MTCD_{G_{1-leader}}$ performs the following operations:

- The $MTCD_{G_{1-leader}}$ computes $MAC'_{HSS}$ using (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 (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, the $MTCD_{G_{1-leader}}$ 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 will find out by $XMAC_{G_{1-1}}$ and $XMAC_{G_{1}}$.

Step10: Now, each $MTCD_{G_{1-1}}$ performs the following operations:

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

Step11: The $MTCD_{G_{1-leader}}$ calculates the aggregated respective authentication code ($XMAC'_{G_{1-leader}}$) using (13) and sends it to the MME for mutual authentication of each $MTCD_{G_{1-1}}$ with the MME.

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

| Goal | secrecy_of_sec_ki,sec_kgl | authentication_on_mtcg_mme | authentication_on_hss_mtcg |

Fig. 3. Result summarized by OFMC backend.

Step12: Finally, MME verifies whether $XMAC'_{G_{1-leader}}$ 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 $MTCD_{G_{1-1}}$ unforgeable authentication success message (e.g. $TMSI^{'new}$ value). Otherwise, MME broadcasts authentication failure message.

After the successful protocol process, the group calculates new $TID_{G_{1}}$ and each $MTCD_{G_{1-1}}$ calculates new $TMSI_{G_{1-1}}$, 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 ($K_{G_{1-1}}$) and the group key ($K_{G_{1}}$) during the authentication process. The goals of the protocol are illustrated in Fig.2. The protocol has three main parties: MTCDs, MME and HSS. The roles of these parties in HLPSL language are described in Appendix I. 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 Fig.3 and Fig.4, respectively. The results prove that the GSL-AKA protocol can reach the mentioned goals and also resist against all the specific attacks (such as replay, MitM and redirection attacks) which are preventing the protocol from achieving these goals.
5. Security Analysis

This section discusses the security properties of the proposed GSL-AKA protocol in terms of mutual authentication between each protocol entity, 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 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 against known attacks. In this security analysis, it is assumed that there is a secure channel between MME and HSS. So, an attacker can only access to the channel between each MTCDs to the group leader and 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 of the leader by verifying the MACG1 (eq.4) sent from the group. By the same way, the MME authenticates them using the XMACG1 (eq.13). These two values are as a function of pre-shared secret keys and can be generated only by corresponding MTCD and the group. Moreover, the group authenticate the HSS and MME by verifying MACHSS and MACMME (eq.10 and eq.15, respectively) sent from the network. These MACs are generated by using group temporary keys (TKG1). So, an adversary cannot generate them without knowing group temporary keys. Finally, for avoiding from reusing of the MACs, the random numbers (TSG1, RANDHSS and RANDMME) 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 (TIDG1 and TMSSG1−1) without using any encryption operations. These two temporary identifiers get new value after each successful protocol process using one-way hash functions (eq.1 and eq.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 MTCDSG1−1 generates its unique MAC (MACG1−1 and XMACG1−1, eq.3 and eq.12, respectively) and sends them to the MTCDSG1−1leader. For reducing signaling congestion and communication overhead, the MTCDSG1−1leader aggregates them into the one MAC (MACG1 and XMACG1, eq.4 and eq.13, respectively) and afterwards, sends them to the network. Moreover, the HSS and MME, for authenticating itself to the group, use only one MAC (MACHSS and MACMME, eq.10 and eq.15, respectively) per group instead of one MAC per each MTCD. Hence, the proposed protocol prevents the network from signaling congestion.

- **Maintain the unlinkability of the session keys:** After each successful running of the proposed protocol, the session keys between each MTCDSG1−1 and the network (RASME, IKG1−1, and CKG1−1) are updated as a function of random numbers (RANDHSS). Then, when one of these session keys is revealed, an adversary cannot link it with its previous and next session keys.

- **Maintain the unlinkability of the group keys:** Whenever one MTCD wants to join or leave to/from the group, the key of the group (KG1) 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 with the next or previous group keys.

- **Solve 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. At the first time, Parne et.al, in the previous SEGB-AKA protocol paper [18], introduced a method for preserving of the 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 use
them 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 it. For overcoming this problem, firstly, in the proposed protocol, we use only hash functions ($f_{\text{K}_{1i}}, f_{\text{K}_{1i}}, f_{\text{K}_{1i}}$, and $f_{\text{K}_{1i}}$) during the protocol process that the keys of these functions are embedded the pre-shared secret key ($K_{\text{G}_{i}}$). Second, if an adversary compromises the pre-shared secret key ($K_{\text{G}_{i}}$, one of the inputs of the hash functions) and by eavesdropping the channel, obtains the outputs of these functions, he/she will never able to obtain other inputs of the hash functions and compromises any other secret data and keys. In the proposed protocol, there are several secret data such as the $\text{IMSI}_{\text{G}_{i}}$, of each MTCD, and the $\text{ID}_{\text{G}_{i}}$, and the $K_{\text{G}_{i}}$ of the group which are used only as one input of hash functions and never be revealed at any way (Note that, when the MTCDs and the group want to introduce themselves to the network and other entities, use temporary identifiers ($\text{TMSI}_{\text{G}_{i}-1}$, and $\text{TID}_{\text{G}_{i}}$)). For this reason, the $\text{IMSI}_{\text{G}_{i}-1}$, $\text{ID}_{\text{G}_{i}}$, and $K_{\text{G}_{i}}$ can be used as a key and secret data. Thus, whenever an adversary compromises the pre-shared secret key ($K_{\text{G}_{i}}$), without knowing other secret data (such as pre-mentioned $\text{IMSI}_{\text{G}_{i}-1}$, $\text{ID}_{\text{G}_{i}}$, and $K_{\text{G}_{i}}$), he/she can never generate message authentication codes ($\text{MAC}_{\text{G}_{i}}$ and $\text{XMAC}_{\text{G}_{i}}$, eq.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 as a legal MME and access to users secure data. In the proposed protocol, the LAI of the connected base station is embedded into the aggregated authentication code ($\text{MAC}_{\text{G}_{i}}$, eq.4) for avoiding from redirection attack. Then, when the HSS computes $\text{MAC}_{\text{G}_{i}}'$ using the LAI sent from MME and finds out the $\text{MAC}_{\text{G}_{i}}$ sent from MME is not equal to $\text{MAC}_{\text{G}_{i}}'$, it realizes the attack occurs and rejects the authentication request.

**Resistance against MitM attack:** In the proposed protocol, the authentication codes of each MTCD ($\text{MAC}_{\text{G}_{i}-1}$, and $\text{XMAC}_{\text{G}_{i}-1}$, eq.3 and eq.12, respectively) are generated using secret data such as the pre-shared secret keys ($K_{\text{G}_{i}}$). 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 against MitM attack.

- **Resistance against impersonation attack:** The access security management entity key ($K_{\text{ASME}}^{\text{MTCD}_{G_{i}-1}}$, eq.9) between each MTCD$_{G_{i}-1}$ and the network is generated using secret data and keys such as the pre-shared session key ($K_{\text{G}_{i}-1}$) and the group temporary key ($TK_{G_{i}}$). That way, an adversary can never generate this key and modify and decrypt the communication messages between the MTCD$_{G_{i}-1}$ and the network. Moreover, an adversary can never generate the aggregated authentication codes ($\text{MAC}_{\text{G}_{i}}$ and $\text{XMAC}_{\text{G}_{i}}$, eq.4 and eq.13, respectively) and impersonate as a legal group to the network.

- **Resistance against replay attack:** In the proposed protocol, to resist against replay attack, the random numbers or timestamps ($\text{RAND}_{\text{HSS}}$, $\text{RAND}_{\text{MME}}$ and $\text{TS}_{\text{G}_{i}}$) are embedded in the authentication codes of each MTCD ($\text{MAC}_{\text{G}_{i}-1}$, and $\text{XMAC}_{\text{G}_{i}-1}$, eq.3 and eq.12, respectively) and the MACs of the HSS and MME ($\text{MAC}_{\text{HSS}}$ and $\text{MAC}_{\text{MME}}$, eq.10 and eq.15, respectively). So, 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 against 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 generate their MACs ($\text{MAC}_{\text{G}_{i}}$ and $\text{XMAC}_{\text{G}_{i}}$, eq.4 and eq.13, respectively) as a function of all the transmitted data and then send them to the network to authenticate their transmitted data to it. Moreover, the HSS and MME generate their MACs ($\text{MAC}_{\text{HSS}}$ and $\text{MAC}_{\text{MME}}$, eq.10 and eq.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 message authentication codes (MACs) are generated in such a way that whenever one of the group members sends invalid data to the other group members, these can find out 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 $\text{XMAC}_{\text{G}_{i}-1}$, and $\text{XMAC}_{\text{G}_{i}}$ (eq.12 and eq.13, respectively) and abort the authentication process. Or, whenever a malicious MTCD sends an invalid data such as invalid message authentication.
new SSDK MTCD and then the MTCD decrypts $KID_i$ to get the symmetric keys. Since the $KID_i$ of new protocol, there is no mechanism to confirm the accuracy of the key identifiers ($KID_i$), the SEGB-AKA protocol process using the unique key identifiers ($KID_i$) and pre-shared secret keys. The SSDK$_i$s are as a function of $KID_i$ and pre-shared secret keys. Since the $KID_i$s are transmitted non-encrypted in 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 SSDK-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 ($KID_i$s). 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 SSDK-AKA protocol, there is no mechanism to confirm the accuracy of new SSDK$_i$s. That way, whenever an adversary changes the encrypted new $KID_i$, the MTCD can not find out 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 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, we compare our proposed GSL-AKA protocol with the existing group-based AKA protocols in terms of communication and computation overheads and show that the proposed protocol has the lowest overheads. To evaluate the mentioned overheads, let there are $n$ number of MTCDs in $m$ groups. Note that, since the overheads for creating groups, joining or leaving to/from 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 Fig.1 and Table 1, the communication overhead per each message of the proposed protocol can be calculated as follows:

- $M_1 = (MAC_{G_{1,i}}) = 64 \times n$
- $M_2 = (AUTH_{G_{1,i}}) = 128 \times n + 256 \times m$
- $M_3 = (AUTH_{G_{1,i}} || LAI) = 128 \times n + 296 \times m$
- $M_4 = (TMSI_{G_{1,i}} || K^{MTCD}_{ASME} || GAV) = 384 \times n + 560 \times m$
- $M_5 = (AUTH_{MMME}) = 432 \times m$
- $M_6 = (RAND_{HSS}) = 128 \times m$
- $M_7 = (MAC_{G_{1,i}}) = 64 \times n$
- $M_8 = (MAC_{G_{1,i}}) = 64 \times m$

Then, the total communication overhead of the proposed protocol is sum of the above calculated overheads and equal to $768 \times n + 1736 \times m$. The communication overhead of other group-based AKA protocols are calculated in [18], like our calculation method. The comparative analysis of communication overhead of existing group-based AKA protocols is illustrated in Fig.5. It is observed that the proposed protocol has the lowest communication overhead compared to 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 are presented in [12] and [16]. The computation overhead of the proposed protocol at the MTC devices is equal to:

$$2T_{hash} \times n + (4T_{hash}) \times m$$

And at the network is:

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

Thus, the total computation overhead is equal to:

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

Moreover, the computation overhead of other protocols are calculated in [18], same as our calculation method. Fig.6 illustrates the comparative analysis of computation overhead of existing group AKA protocols. It can be seen that
### Table 2

Security Analysis Between Group-based AKA Protocols.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SP1</td>
<td>Symmetric</td>
<td>Hybrid</td>
<td>Hybrid</td>
<td>Symmetric</td>
<td>Symmetric</td>
<td>Hybrid</td>
<td>Symmetric</td>
<td>-</td>
<td>Symmetric</td>
<td>-</td>
<td>Symmetric</td>
<td>-</td>
</tr>
<tr>
<td>SP2</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SP3</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SP4</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SP5</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SP6</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SP7</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SP8</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>SP9</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>SP10</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SP11</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SP12</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SP13</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

SP1: Type of cryptosystem; SP2: Basic security requirements; SP3: Follow the 3GPP standard; SP4: Privacy preservation; SP5: Network signaling congestion prevention; SP6: Maintain the unlinkability of session keys; SP7: Maintain the unlinkability of group keys; SP8: Solve 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.

![Fig. 5. Comparison of the communication overheads. (a) m=1. (b) m=100.](image)

7. Proving of the Relative Lowest Overheads

This section illustrates 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 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 proto-
The proposed protocol is divided into three different categories. First, in the proposed protocol, the group members in Step.3 embed their identifiers ($T M S I_{G_{i-1}}$ and $T I D_{G_{i}}$) into the $A U T H_{G_{i}}$ message and then send them to the network to introduce themselves to it. Second, at the end of the Step.7, the HSS sends new session keys ($K_{M T C D M_{G_{i-1}}}$) and new identifiers ($T I D_{G_{i}}$ and $T M S I_{G_{i-1}}$) to the MME to communicate this entity independently with the group. And, the third category is the authentication challenges messages ($M A C_{G_{i}}, X M A C_{G_{i}}, M A C_{H S S}$ and $M A C_{M M E}$) and their random inputs ($T S_{G_{i}}, R A N D_{H S S}$ and $R A N D_{M M E}$) 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. That way, these are must be same as the standard protocol for 4G networks, EPS-AKA [23]. The size of these values are mentioned in Table 1 and how these are transmitted between network entities are shown in Fig.1.

About the third category, each of the AKA protocols use their own authentication method to authenticate mutually the network entities. 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 ($M A C_{G_{i-1}}, X M A C_{G_{i-1}}, M A C_{H S S}$ and $M A C_{M M E}$) and three random numbers ($T S_{G_{i}}, R A N D_{H S S}$ and $R A N D_{M M E}$) 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 Fig.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 has the lowest communication overhead during its process, in comparison with existing authentication methods. Moreover, to reduce communication overhead of the authentication process, the $M T C D M_{G_{i-1}}$ aggregates all the authentication codes ($M A C_{G_{i-1}}$ and $X M A C_{G_{i-1}}$, $M A C_{M M E}$) into one authentication code ($M A C_{G_{i}}, X M A C_{G_{i}}$, eq.4 and eq.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 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 another 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 another additional data. In Section 4 and 5, we prove that the proposed protocol can achieve all the defined security requirements.

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 be authenticate mutually the
group with the network entities (HSS and MME). Thus, in these protocols, some challenges are applied which each entity with solving one of them and publishing the response, 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 authentication mechanism. In all of the group-based AKA protocols, the existence of challenge-response process is required and each of them perform it in different ways. The proposed protocol uses hash functions to authenticate mutually each entity. So that, there are four message authentication codes (MAC_{G_1}, MAC_{HSS} and MAC_{MME}) and their random numbers (TS_{G_1} and 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 are presented in [12], [16]), then the computation overhead of the challenge-response process in the proposed protocol is the lowest.

In addition, to achieve another security requirements mentioned in Section 5, each protocol uses another cryptographic functions such as symmetric or asymmetric encryption during its run and thus it increases 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 in comparison with other existing authentication methods, then, the total computation overhead of the proposed protocol is the lowest. That means, 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 GSL-AKA protocol for M2M communication in LTE and 5G networks was proposed. Compared with the prior group-based AKA, the GSL-AKA protocol could achieve all the security goals, overcome all the known attacks, preserve the privacy of the MTCDs and overcome the single key problem which the previous group-based AKA protocols could not solve it. Moreover, our performance analysis showed that the proposed protocol has the best communication and computation overheads and these are the lowest.

In the machine-type communication, when a group of MTCDs move to the coverage of new eNB simultaneously, these have to re-authenticate to the network. If the existing group-based AKA protocols are used to re-authenticate, the network will face to 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.

Appendix A. HLPSL Codes of the Proposed GSL-AKA Protocol

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

```
role mme{
M,H,D
 SND,RCV
 KGL,KI,Key
 Request_Request_Message,
 Request_Identity_Message,
 IDG,IDI,Rhss,Rmme,Amf
 F1,F2,F3,F4,KDF
 played_by M

def=
local
State
LAI,TIDGI,TIDI
const
sec_ki,sec_kg1,
mtcd_mme,msc_mtd
success
:

init State := 0

transition
1. State = 0
   /
   SND(Request_Identity_Message)

   State' := 1
   /
   RCV((Request_Request_Message)

   State' := 2
   /
   SND(RCV(IDGI,TIDGI,F1(KGL.F1(KI.
   IDGI.TSIG1))).LAI).TSIG1)

   secret(KI,sec_ki,sec_kg1)

   secret(IDGI,sec_kg1,sec_ki)

   secret(IDGI,sec_kg1,sec_ki)

   Rmme := new()

   SND(F1(F2(KGL.IDGI.RHSS).F1(KGL.IDGI.
   RHSS).F1(F2(KGL.IDGI.RHSS).F1(F2(KGL.IDGI.
   RHSS).F1(F2(KGL.IDGI.RHSS).F1(F2(KGL.IDGI.
   RHSS).F1(F2(KGL.IDGI.RHSS).F1(F2(KGL.IDGI.
   RHSS).F1(F2(KGL.IDGI.RHSS).F1(F2(KGL.IDGI.
   RHSS).F1(F2(KGL.IDGI.RHSS).F1(F2(KGL.IDGI.
   RHSS).F1(F2(KGL.IDGI.RHSS).F1(F2(KGL.IDGI.
   RHSS).F1(F2(KGL.IDGI.RHSS).F1(F2(KGL.IDGI.
   RHSS).F1(F2(KGL.IDGI.RHSS).F1(F2(KGL.IDGI.

   witness(M,D,msc_mme,Rmme)

4. State := 3
   /
   RCV(F1(KGL.F1(KI.TIDGI.IDI)
   .Rhss)).F1(KGL.IDGI.RHSS)

   SND(success)
}
```

Fig. A.3. The role of the HSS.

```
role hss{
H,M,D
 SND,RCV
 KGL,KI,Key
 Request_Request_Message,
 Request_Identity_Message,
 IDG,IDI,Rhss,Rmme,Amf
 F1,F2,F3,F4,KDF
 played_by H

def=
local
State
LAI,TIDGI,TIDI
const
sec_ki,sec_kg1,
mtcd_mme,msc_mtd
success
:

init State := 0

transition
1. State = 0
   /
   SND((Request_Request_Message)

   State' := 1
   /
   RCV((TIDGI,TIDGI.F1(KGL.F1(KI.
   IDGI.TSIG1))).LAI).TSIG1

   secret(KI,sec_ki,sec_kg1)

   secret(IDGI,sec_kg1,sec_ki)

   secret(IDGI,sec_kg1,sec_ki)

   secret(IDGI,sec_kg1,sec_ki)

   secret(IDGI,sec_kg1,sec_ki)

   Rmme := new()

   SND(F1(F2(KGL.IDGI.RHSS).F1(KGL.IDGI.
   RHSS).F1(KGL.IDGI.RHSS).F1(KGL.IDGI.
   RHSS).F1(KGL.IDGI.RHSS).F1(KGL.IDGI.
   RHSS).F1(KGL.IDGI.RHSS).F1(KGL.IDGI.
   RHSS).F1(KGL.IDGI.RHSS).F1(KGL.IDGI.
   RHSS).F1(KGL.IDGI.RHSS).F1(KGL.IDGI.
   RHSS).F1(KGL.IDGI.RHSS).F1(KGL.IDGI.
   RHSS).F1(KGL.IDGI.RHSS).F1(KGL.IDGI.

   witness(M,D,msc_mme,Rmme)

4. State := 3
   /
   RCV(F1(KGL.F1(KI.TIDGI.IDI)
   .Rhss)).F1(KGL.IDGI.RHSS)

   SND(success)
}
```

References


[13]


[23] 3GPP. 3G security; Security architecture. TS 33.102 V.0.1.0, 3rd Generation Partnership Project (3GPP) (2017).


List of Figures

1 The GSL-AKA protocol 5
2 The goals of the proposed GSL-AKA protocol 6
3 Result summarized by OFMC backend. 6
4 Result summarized by CL-AtSe backend. 7

5 Comparison of the communication overheads. (a) m=1. (b) m=100. 10
6 Comparison of the computation overheads. (a) m=1. (b) m=100. 11
A.1 The role of the MTCDs. 12
A.2 The role of the MME. 13
A.3 The role of the HSS. 13

Authors’ Biographies

Mohammad Mahdi Modiri received his B.S. degree in Telecommunications Engineering from University of Tehran, Tehran, Iran, in 2016. In addition, he received his M.S. degree in Secure Communication and Cryptography Engineering from Sharif University of Technology, Tehran, Iran in 2018. He is currently working toward his Ph.D. 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 with the Electronics Research Institute, Sharif University of Technology, Tehran, Iran, where he is an Adjunct Assistant Professor with the Electrical Engineering Department. He has authored or co-authored of 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. Javad is a Founding Member of the Iranian Society of Cryptology. Also he has been 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 B.S. and M.S. degrees in electrical engineering from Sharif University of Technology, Tehran, Iran, in 1972 and 1989, respectively. He also received the Ph.D. degree in information technology from Queensland University of Technology, Australia, in 1997. Currently, he is an associate professor in the Electronics Research Institute and adjunct associate professor in 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.