A Secure Group-Based AKA Protocol for Machine-Type Communications

Rosario Giustolisi, Christian Gehrmann, Markus Ahlström, Simon Holmberg

Swedish Institute of Computer Science, Stockholm, Sweden

Abstract. The fifth generation wireless system (5G) is expected to handle with an unpredictable number of heterogeneous connected devices while guaranteeing a high level of security. This paper advances a group-based Authentication and Key Agreement (AKA) protocol that contributes to reduce latency and bandwidth consumption, and scales up to a very large number of devices. A central feature of the proposed protocol is that it provides a way to dynamically customize the trade-off between security and efficiency. The protocol is lightweight as it resorts on symmetric key encryption only, hence it supports low-end devices and can be already adopted in current standards with little effort. Using ProVerif, we prove that the protocol meets mutual authentication, key confidentiality, and device privacy also in presence of corrupted devices, a threat model not being addressed in the state-of-the-art group-based AKA proposals. We evaluate the protocol performances in terms of latency and bandwidth consumption, and obtain promising results.

1 Introduction

The evolution of mobile networks has made a key achievement in each of its generations: 1G established the foundation of mobile networks; 2G increased the voice connectivity capacity to support more users per radio channel; 3G introduced high-speed internet access; 4G provided more data capacity. One of the key achievement for 5G is to be the reference network for the Internet of Things (IoT) connectivity. Analysts forecast more than 25 billion of devices to be interconnected in 2020 [16]. Providing connectivity to such a large number of devices, which may require simultaneous network access, will lead to a potential signaling overload. Signaling data is growing 50% faster than data traffic in mobile networks [22] and is expected to surpass the global IP traffic growth within three years [23]. An increased level of signaling would affect speed and data capacity of 5G. Thus, to fully support IoT connectivity, the contemporary architecture of the mobile network should be revisited, including the aspects related to security.

The Authentication and Key Agreement protocol (AKA) has a central role in the security of mobile networks as it bootstraps the parameters needed to form a security context that is agreed by the parties. The protocol provides mutual authentication between device and serving network, and establishes session keys. The state-of-the-art protocol used in 4G (EPS-AKA) [3] is almost identical to its predecessor used in 3G, which was introduced in the late 90s. A limitation of EPS-AKA is that, for each device that requires network access, the protocol requires signaling among the device, the local serving network and the device's remote home network. In particular, the signaling between serving network and home network may introduce a major delay when they are distant, which is the

case when users are roaming. This represents a bottleneck for the development of 5G as a low delay and reliable network for IoT devices.

From this situation emerged the need of a *group-based* AKA, which allows the serving network to authenticate a group of devices reducing the signaling and communication latency with the home network. Groups may consist of devices sharing similar features such as functions, locations, or ownership. In the scenario of IoT, devices often operate in groups and some use cases have been recently advanced [11, 21, 13]. While the functional goals of group-based AKA are clear, new security aspects arise. The group approach introduces additional threats, which mainly originate from colluding corrupted members [18]. This results to a more powerful intruder than one historically considered in the current AKA protocol. Thus, it seems to be an open challenge to design a group-based AKA secure against the extended threats. This paper addresses this very challenge. In particular, the contributions of this paper includes:

- A novel mechanism based on the inverted hash tree that allows the network operator to balance dynamically the requirements of security and efficiency of the designed protocol.
- The formal security analysis of the protocol in ProVerif.
- A prototype implementation of the protocol in the OpenAirInterface platform.
- A performance analysis of the protocol in terms of latency and bandwidth consumption.

Outline. The paper is organized as follows. Section 2 presents a primer on AKA. Section 3 details the group-based AKA protocol. Section 4 describes the formal analysis of the protocol in ProVerif. Section 5 details the implementation of the protocol in OpenAirInterface and discusses its performances. Section 6 analyses some related work. Finally, Section 7 draws some conclusions.

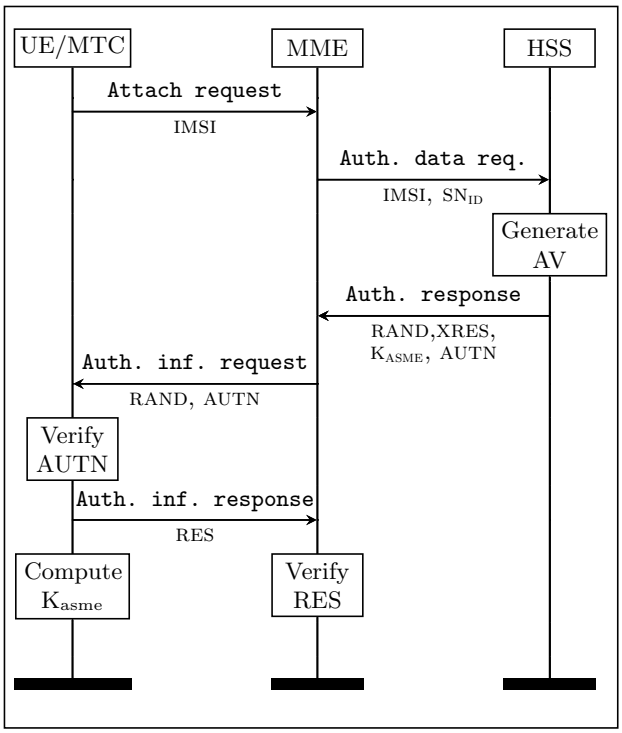
2 Background

The three main roles that concern the AKA protocol are the *User Equipment* (UE) or device, the *Mobility Management Entity* (MME) or serving network, and the *Home Subscriber Server* (HSS) or authentication server. The UE role concerns the tasks of the terminal device and USIM. A subscriber identity (IMSI) is permanently stored on the USIM so the network can identify the UE. The USIM also stores a long-term secret key K that is shared with the HSS. With the introduction of *machine-type communication* (MTC), the 3GPP consortium released a dedicated specification for MTC devices to enhance the LTE suitability for the IoT market [5]. Thus, we refer to the UE also using the term MTC.

The MME role concerns the tasks of covering the mobility of the MTC. The MME serves a number of MTCs according to its geographical area. Each MTC is connected to a *base station* (eNodeB), which in turn is directly connected to an MME. In the context of AKA, the MME authenticates the MTC and agree on a session master key K_{ASME} from which they can derive further keys to protect the signaling data.

The HSS role concerns the tasks of assisting the MME for the mutual authentication. The signaling between HSS and MME is secured with Diameter [4]. The HSS shares with the MTC IMSI, K, and a *sequence number* (SQN) to support authentication.

Fig. 1: EPS-AKA message sequence chart



2.1 EPS-AKA

The state-of-the-art AKA protocol is EPS-AKA, which is the standard for LTE. The protocol is described in Figure 1 and consists of five main messages:

- The `Attach request` message bootstraps the protocol. It normally includes the IMSI of the MTC, when the device visits the MME for the first time. Future attach requests will include the Globally Unique Temporary Identity (GUTI), which is generated by the MME and assigned to the MTC. In doing so, the MME can translate the GUTI to the corresponding IMSI, preserving the privacy of the MTC.
- The `Authentication data request` message, sent by MME with identity SN_{ID}, requires the HSS to generate an authentication vector consisting of:
  - a random value RAND that provides freshness to the session;
  - the expected response XRES, based on RAND and K, that allows the MME to authenticate the MTC;
  - the session master key K_{ASME}, to encrypt the signaling between MTC and serving network;
  - the authentication token AUTN, based on RAND, K, and SQN, that allows the MTC to authenticate the serving network.
- The `Authentication response` message contains the authentication vector and is transmitted to the MME.
- The `Authentication information request` message consists of RAND and AUTN, which the MME forwards to the MTC. The MTC checks that the SQN matches a valid one and if so, it successfully authenticates the serving

network. The MTC computes the session master key K_{ASME} and the response RES, which is based on K and on the received RAND.
– The `Authentication information response` message, which the MTC sends to the MME, contains RES. The MME successfully authenticates the MTC if RES = XRES. The MME computes K_{ASME} so the signaling between serving network and MTC can be protected with session keys derived from K_{ASME}.

The cryptographic functions for the generation of the different terms outlined above are included in *MILENAGE* [2], which is a set of algorithms currently supported by EPS-AKA. The limitation of EPS-AKA is that `Authentication response` and `Authentication data request` are required for *each* device that requires network access. The next section introduces a group-based AKA that addresses this very limitation.

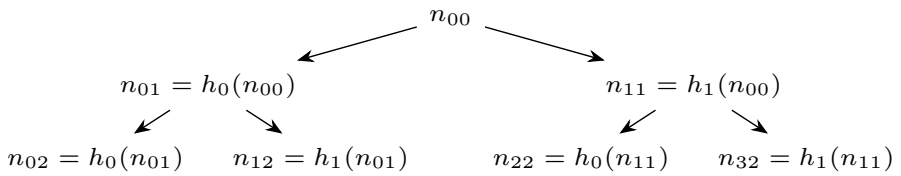
3 Group-based AKA

The design of the group-based AKA is pivoted on the *inverted hash tree*. Thus, we briefly discuss the notion of inverted hash trees prior to providing a detailed description of the protocol.

Inverted hash trees. An inverted hash tree (see Figure 2) is a data structure in which a node is linked to at most two successors (children), and the value of each node is computed using a family of hash functions h_*. The value of the root is given, while the value associated with any other node is derived from the hash value of its parent. In particular, we consider two hash functions h_0 and h_1 and recursively assign the value of each node n_{ij} located at i^{th} position and j^{th} level as follows.

$$n_{ij} = \begin{cases} h_0(n_{k(j-1)}) \text{ if } i = 2k & \text{(left children)} \\ h_1(n_{k(j-1)}) \text{ if } i = 2k+1 & \text{(right children)} \\ \text{given value if } i = j = 0 & \text{(root)} \end{cases}$$

Fig. 2: An inverted hash tree of height 2



The underlying idea of the proposed group-based AKA is to associate each MTC to a value of the leaf node, and to reveal a sub-root node to the MME so that it can authenticate the (sub)group of all MTC descendants. This allows the HSS to control the trade-off between security and efficiency dynamically. In fact, the HSS can reveal sub-roots at different levels. Revealing a sub-root at a higher level supports security at the cost of efficiency because the MME can authenticate a smaller group of MTC without involving the home network.

Conversely, revealing a sub-root at lower level supports efficiency at the cost of security because the MME can authenticate a large group of MTC without involving the home network. The proposed group-based AKA protocol supports MILENAGE. It does not introduce new primitives (e.g., secret sharing or public key encryption) to favour backward compatibility with existing mobile telephony systems and uses most of the functions already available in MILENAGE (i.e., *kdf*, *f2*, *f3*, *f4*, and *f5*).

3.1 Protocol Description

The protocol assumes two inverted hash trees of height $\mathcal{H}$, both generated by the home network. The structures of the two inverted hash trees are identical, and each MTC_i is associated with the leaf nodes with $\text{PATH}=(i, \mathcal{H})$ in both trees. The *GK tree* serves as group key tree, and the value of its root can be seen as a *master group key*. Each leaf node of the tree ($\text{GK}_{\text{i}\mathcal{H}}$) serves as master individual key and is associated to each MTC_i. Several session individual keys $\text{HGK}_{(\text{i}\mathcal{H},\ \text{N})}= \mathit{hash}(\text{GK}_\text{ij},\ \text{N})$, which are keyed with a sequence number N, can be derived from the master individual key. The generation of several session individual keys enables for several secure AKA runs using the same $\text{GK}_{\text{i}\mathcal{H}}$. The *CH tree* serves as challenge key tree. Also in this case, each leaf value of the tree ($\text{CH}_{\text{i}\mathcal{H}}$) is associated to an MTC_i and acts as individual challenge key. Several session challenge keys $\text{HCH}_{(\text{i}\mathcal{H},\ \text{N})}= \mathit{hash}(\text{CH}_\text{ij},\ \text{N})$ can be generated from $\text{CH}_{\text{i}\mathcal{H}}$.

As we shall see later, the MME will send $\text{HCH}_{(\text{i}\mathcal{H},\ \text{N})}$ to the MTC so that the device can compute $\text{HGK}_{(\text{i}\mathcal{H},\ \text{N})}$. In fact, each MTC_i knows no keys initially, but is given an obfuscated value $\text{O}_{(\text{i}\mathcal{H},\ \text{N})} = \mathit{hash}(\text{K},\ \text{HCH}_{(\text{i}\mathcal{H},\ \text{N})}) \oplus \text{HGK}_{(\text{i}\mathcal{H},\ \text{N})}$.

As soon as the MTC receives HCH and N, it can use them with O and K to retrieve HGK. The obfuscation binds both session keys to K. This choice prevents that two corrupted MTCs, say MTC_1 and MTC_2, swap their keys to break authentication.

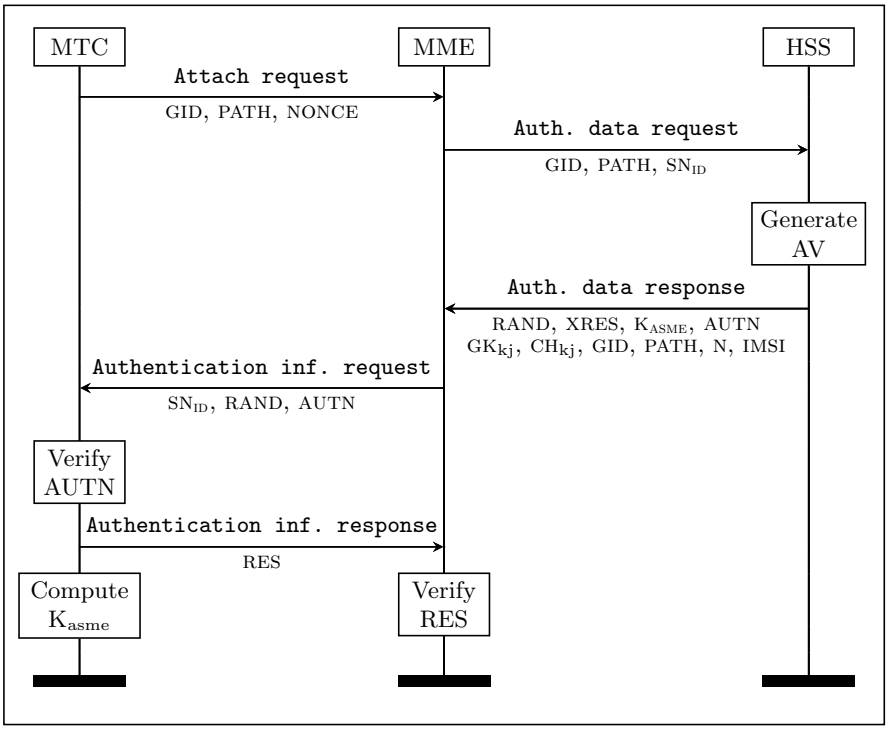
Table 1: Description of the terms introduced in the group-based AKA

Term	**Description**
GID	Group identifier
NONCE	Random number
GK_ij	The key associated with the value of the node at the i^{th} position and j^{th} level of the inverted hash tree *GK*.
CH_ij	The challenge key associated to the value of the node at the i^{th} position and j^{th} level of the inverted hash tree *CH*.
$\text{HGK}_{(\text{ij},\ \text{N})}$	The result of hashing GK_ij and N.
$\text{HCH}_{(\text{ij},\ \text{N})}$	The result of hashing CH_ij and N.
$\text{O}_{(\text{ij},\ \text{N})}$	The obfuscated value that hides the hashed keys GK_ij and CH_ij with respect to the sequence number N.
AUT_D	The authentication parameter in the group authentication
RES_D	The response parameter in the group authentication
$\text{K}_{\text{ASME}\text{D}}$	The session key generated in the group authentication

Each MTC that is member of the group shares with the home network the following terms: the group identifier GID, the assigned PATH, and a number of obfuscated values $\text{O}_{(\text{i}\mathcal{H},\ 1)}$, $\text{O}_{(\text{i}\mathcal{H},\ 2)}$,..., $\text{O}_{(\text{i}\mathcal{H},\ \text{N})}$,...,$\text{O}_{(\text{i}\mathcal{H},\ \text{M})}$. All the terms introduced by the protocol are defined in Table 1.

We distinguish *Case A* and *Case B*. In *Case A*, the MME cannot derive the needed keys to authenticate the MTC, hence the MME needs to communicate with the HSS. In *Case B*, the MME can derive the keys to authenticate the MTC without any interaction with the HSS.

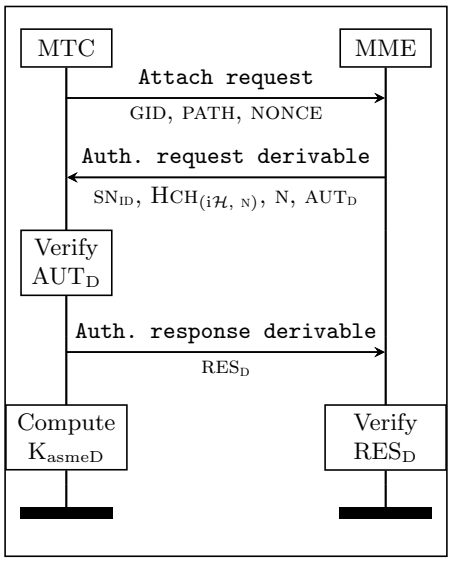
Fig. 3: Message sequence chart of Case A



The first message of the protocol is the `Attach request`, which the MTC sends to the MME, and it is exactly the same in both cases. In fact, the MTC cannot say beforehand which case applies. If this is the very first attach request that the MME receives from a member of the group or the MME cannot derive the needed keys associated to that MTC, the MME proceeds according to Case A, otherwise it follows Case B. We now describe the two cases separately. The message sequence charts for Case A and Case B are respectively depicted in Figure 3 and Figure 4.

Case A. This case requires that the MME communicates with the HSS to obtain the needed keys and then to authenticate MTC_i. Hence, the MME generates the `Authentication data request` message, which contains GID, PATH, NONCE, and SN_ID. The MME then sends the message to the HSS via Diameter. The HSS checks whether GID and PATH are valid and, according to the security policy of the group, it chooses two indexes k and j, with $j < \mathcal{H}$, such that GK_kj and CH_kj are ancestor nodes of $\text{GK}_{\text{i}\mathcal{H}}$ and $\text{CH}_{\text{i}\mathcal{H}}$ respectively. The HSS then generates an authentication vector in the same way it is generated in EPS-AKA, and sends the `Authentication data response` message to the MME. The message includes

Fig. 4: Message sequence chart of Case B



the same elements already specified in EPS-AKA plus the new elements GK_{kj}, CH_{kj}, GID, PATH, N, and IMSI. The elements GK_{kj} and CH_{kj} serve as root of two subtrees. The MME will be able to derive the values of all the leaf nodes within the subtrees without the need to communicate with the HSS. From now on, the procedure for Case A continues exactly as in EPS-AKA.

Case B. This case assumes that the MME already knows some nodes GK_{kj} and CH_{kj} that are ancestors of $GK_{i\mathcal{H}}$ and $CH_{i\mathcal{H}}$. Hence, the MME computes $GK_{i\mathcal{H}}$ and $CH_{i\mathcal{H}}$, and from those $HGK_{(i\mathcal{H},\ N)}$ and the $HCH_{(i\mathcal{H},\ N)}$. If the MME has not previously run the group-based AKA with MTC_i, then the value of the sequence number N is the one provided in Case A by the HSS. Otherwise, it sets N=N+1. The MME periodically reports the updated sequence number to the HSS to keep the synchronization of the values.

The MME computes the authentication token $AUT_D = f5(HGK_{(i\mathcal{H},\ N)}, NONCE)$, $MAC_{HGK_{(i\mathcal{H},\ N)}}(NONCE, HCH_{(i\mathcal{H},\ N)}, GID, SN_{ID}, PATH)$ and sends the `Authentication request derivable` message, which contains SN_{ID}, $HCH_{(i\mathcal{H},\ N)}$, and AUT_D. The MTC de-obfuscates the value $O_{(i\mathcal{H},\ N)}$, and retrieves the session individual key $HGK_{(i\mathcal{H},\ N)} = hash(K, HCH_{(i\mathcal{H},\ N)}) \oplus O_{(i\mathcal{H},\ N)}$. Then, it sends the `Authentication response derivable` message that contains $RES_D = f2(HGK_{(i\mathcal{H},\ N)}, HCH_{(i\mathcal{H},\ N)})$. Both MTC and MME can compute the session key $K_{ASMED} = kdf(f5(HGK_{(i\mathcal{H},\ N)}, NONCE), f3(HGK_{(i\mathcal{H},\ N)}, HCH_{(i\mathcal{H},\ N)}), f4(HGK_{(i\mathcal{H},\ N)}, HCH_{(i\mathcal{H},\ N)}), SN_{ID})$.

In the proposed group-based AKA one major modification is that the IMSI is not sent by the MTC. In Case A, the HSS sends the IMSI to the MME securely via Diameter. The attach request may still contain the temporal identity GUTI due to legacy reason. However, lawful interception is always guaranteed because the combination (GID, PATH) is unique and known to the HSS. Thus, if needed, the MME can send GID and PATH of an MTC to the HSS, and obtain the corresponding IMSI.

`Authentication request derivable` has AUT_D, which contains the data $f5(\text{HGK}_{(\text{i}\mathcal{H},\ \text{N})},\ \text{NONCE})$. This data is not strictly necessary because AUT_D already contains a MAC for integrity check. However, we prefer to maintain the data to meet the same structure of the traditional AUTN field.

We note that MME and HSS should periodically synchronize the current value of sequence number. This prevents a corrupted MTC to successfully reuse a session individual key when moving from an MME to another. However, such attack can be easily mitigated if the HSS syncronizes the sequence number with the old MME when the new MME sends to the HSS the `Authentication data request`.

4 Security Analysis

We analyze the group-based AKA protocol in ProVerif [9], a protocol analyzer that can prove reachability and equivalence-based properties automatically. The input language of ProVerif is based on the applied pi-calculus [6]. Authentication can be expressed as correspondence assertions [28] based on events, while privacy can be expressed as *observational equivalence* [24] property based on processes that differ only in the choice of terms. We consider threats originating from a Dolev-Yao intruder [14] who has full control of the network. The intruder can also inject messages of his choice into the public channels, and exploit the algebraic properties of cryptographic primitives due to an equational theory. Moreover, we extend the capabilities of the intruder with threats deriving from colluding corrupted principals. Differently from other works on formal analysis of AKA [1, 26, 10], we choose to model the communications between MME and HSS using the cryptographic primitive of probabilistic symmetric encryption rather than using ProVerif's private channels. This choice allows us to model corrupted principals by just sharing the private key with the intruder. It also increases the chance that ProVerif successfully terminates the verification, and gives the attacker more discretional power because it can observe when a communication between MME and HSS happens. As result, we achieve stronger security guarantees for the analysis of the protocol.

Table 2: Equational theory to model the proposed group-based AKA protocol

Primitive	**Equation**
Probabilistic symmetric enc.	$sdec(senc(m,k,r),k) = m$
XOR	$xor(m1, xor(m1,m2)) = m2$
Hash	$hash(m) = d$
MAC	$MAC(m,k) = d$
Inverted hash tree	$set\_node(parent, pos) = child$ $par\_path(ch\_path(par\_path, pos)) = par\_path$

The cryptographic primitives adopted in the group-based AKA protocol are illustrated in Table 2. The theory for hash, MAC, XOR, and probabilistic symmetric key encryption are well-known in ProVerif. We introduce a novel theory in ProVerif to support inverted hash trees. The function *set_node* allows us to generate a new child node which value is given by hashing the parent's value and the position of the child node (i.e. left or right). The function *ch_path* takes in

a parent's path and a position and returns the corresponding child's path. The function *par_path* takes in a child's path and returns the parent's path.

We check confidentiality of the session master keys K_{ASME} and K_{ASMED}, mutual authentication, and MTC identity privacy. The details of the formalisation in the applied pi-calculus of the requirements are in Appendix A.

Results. The results of the automatic analysis in ProVerif indicate that the protocol meets confidentiality, mutual authentication, and MTC identity privacy. Table 3 reports the execution times over an Intel Core i7 2.6 GHz machine with 12 GB RAM. Our analysis considers an unbounded number of honest MTC, HSS, and MME and an attacker in control of the network and of an unbounded number of corrupted MTCs. Note that an inverted hash tree with an unbounded number of leaves would require an unbounded number of intermediate nodes. Unfortunately, ProVerif cannot handle this scenario. We overcome this situation by fixing root and height of the tree and then generating an unbounded number of sub-trees.

Table 3: Summary of the ProVerif analysis of the group-based AKA

Requirement	**Result**	**Time**
Session master key confidentiality	✓	1.8 s
Serving network authentication	✓	4.4 s
MTC authentication	✓	4.3 s
MTC identity privacy	✓	2.8 s

5 Implementation

We choose to implement the protocol in OpenAirInterface (OAI) [7], an open-source wireless technology platform written in C. OAI is a fully-stacked EPS implementation with the goal of being used for 5G development and research. It supports MME, HSS, and a simulation of an MTC. It does not require any radio hardware since it can simulate the radio interface used in EPS via Ethernet. However, OAI supports radio hardware if needed. *OPENAIR-CN* and *Openair-interface5G* are the two main modules that constitute OAI. OPENAIR-CN is an implementation of the 3GPP specifications concerning the Evolved Packet Core Networks, in particular the MME and HSS network elements. Openair-interface5G is an implementation of a simulated MTC and provides a realistic radio stack signaling when connected to OPENAIR-CN.

5.1 Approach

Our approach to the prototype implementation is to code the group-based AKA as a patch of OAI. In doing so, we favour backward compatibility with the existing standard. It follows that, when possible, we aim to reuse the existing parameter and message structures as specified in 3GPP standards. For example, we can reuse the structure of IMSI for GID since they have a similar purpose. However, some terms have no similar counterpart in EPS so we design them from scratch. We also introduce new functions and commands that extend the

functionality currently in use in EPS with ones appropriate for group-based AKA. For example, the algorithm `traverse tree` allows both MME and HSS to find a node in the inverted hash tree. The function takes in the node's depth, the node's PATH, and an ancestor node value. Then, it traverses the subtree originating in the ancestor node according to the bit sequence in PATH: if the current bit is 0 then a byte of zeros is appended to the current node value, otherwise a byte of ones is appended to the current node value. The pseudo-code is outlined in Algorithm 1. More details regarding configuration and parameters are detailed in Appendix B.

Algorithm 1: `traverse tree`

```
input : GK_kj, PATH, z=NODE DEPTH
output: GK_iz (descendant of GK_kj)
Digest ← GK_kj;
for l ← 0 to NODE DEPTH−1 do
    current_Bit ← bit l of PATH;
    if current_Bit = 0 then
        Digest = (Digest || 00000000);
    else
        Digest = (Digest || 11111111);
    end
    Digest ← SHA256(Digest);
    Digest ← truncate_to_128_bits(Digest);
end
GK_iz ← Digest;
```

5.2 Performance analysis

We present the performance analysis of the prototype implementation of the group-based AKA in terms of latency and bandwidth consumption. The goal of the analysis is to have a quantitative evaluation of the benefit that the protocol provides with respect to the current EPS-AKA. We distinguish the analysis of the *non-access stratum* (NAS), which concerns the communication between MTC and MME, and of the *S6a* interface, which concerns the communication between MME and HSS.

Bandwidth consumption. Our analysis considers the worst case for both EPS-AKA and group-based AKA. This is because some of the existing and new parameters can have variable sizes. Thus, we select the maximum possible value for each parameter. The bandwidth consumption for EPS-AKA concerning both NAS and S6a interface is given by the sum of the size of the parameters sent within the messages, multiplied by the number of devices. The formula of the bandwidth consumption for the group-based AKA is complicated by the inverted hash tree. Given m MTCs devices, the formula is defined in Equation 1.

$$\textbf{BAND\_GB\_NAS} = m \times \left(\text{GID} + \frac{(\lceil \log_2 m \rceil \times 2 - 1)}{8} + 2 + \text{NONCE}\right) + (m-1) \times (\text{HCH} + \text{AUT}_\text{D} + \text{RES}_\text{D}) + \text{RAND} + \text{AUTN} + \text{RES}. \quad \text{(Equation 1)}$$

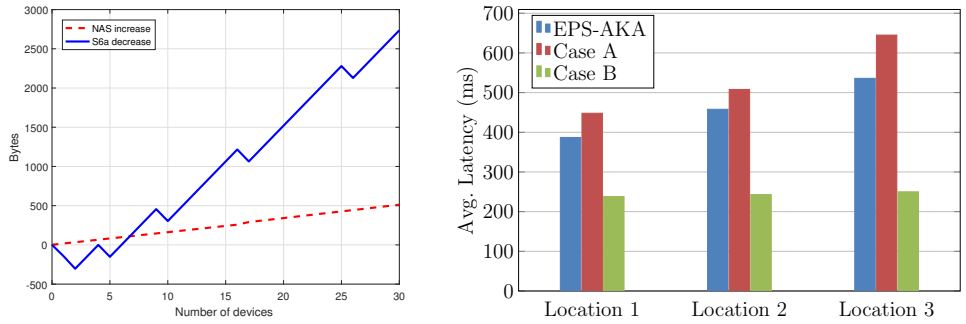
Regarding the bandwidth consumption for the S6a interface, we consider the values provided in the Authentication Information Request (AIR) and in the Authentication Information Answer (AIA) messages, which are due to the Diameter protocol. The bandwidth consumption for the group-based AKA can be computed as Equation 2.

$$\textbf{BAND\_GB\_S6a} = \text{IMSI} + 2 \times \text{GID} + \text{RAND} + \text{XRES} + \text{AUTN} + \text{K}_{\text{ASME}} + \text{GK}_{\text{ij}} + \text{CH}_{\text{ij}} + \mathcal{H} + \text{SN}_{\text{ID}} + 2 \times \left(min(\text{PATH}) + \frac{\lceil \log_2 m \rceil \times 2 - 1)}{32} \times 4 \right). \quad \text{(Equation 2)}$$

Overall, the group-based AKA consumes less bandwidth when already seven MTC devices are considered. This is described by the left picture of Figure 5.

Latency. The latency analysis consists of the evaluation of the round-trip time (RTT) between MTC, MME, and HSS. We consider fixed locations for MTC and MME, and different geographic locations for the HSS. In so doing, we simulate different scenarios of UE attaching from different countries. Since we focus on the latency between MME and HSS, we can assume that the RTT between MTC and MME is fixed. We select three different locations from the *WonderProxy* servers [27] with various distances from the MME: *Location 1* is 1 Km far; *Location 2* is 2,000 Km far; *Location 3* is 10,000 Km far. We compute the average RTT of each location by pinging 100 times the corresponding servers. Then, we run 20 instances of EPS-AKA and group-based AKA in OAI. The results are described in the right picture of Figure 5. They show that EPS-AKA and Case A for the group-based AKA have similar values, with the latter having more latency because more amount of data is communicated. As expected, there are very small variations in Case B for the group-based AKA. This confirms that when an MTC device is running within Case B there is a significant reduction in latency.

Fig. 5: On the left: The increase in NAS bandwidth consumption and the decrease in S6a bandwidth consumption when the group-based AKA is used instead of EPS AKA. On the right: latency comparison among different locations



6 Related Work

Recently, several amendments to the AKA protocol have been advanced [8, 17] and new group-based AKA protocols have been proposed. Broustis et al. [11]

designed three group-based AKA schemes with the goal to reduce the overall signaling between the parties. All the proposed schemes share the idea of using global values based on a shared group key and to introduce a gateway that mediates between MTC devices and MME. The use of global values and of a gateway is beneficial to the bandwidth consumption. However, none of the schemes meets authentication of the devices in presence of either a corrupted gateway or corrupted colluding devices [18]. Lai et al. [21] proposed *SE-AKA*, a group-based AKA protocol for LTE networks. The protocol uses public key encryption and supports key forward and backward secrecy. It reduces the communication overhead between MME and HSS to only one message exchange but increases the size of the authentication data response linearly on the size of the group, which makes the protocol not amenable for large groups. Choi et al. [13] use only symmetric cryptography for their group-based AKA protocol. The underlying idea of the protocol is to rely on a global authentication vector based on a group key shared between HSS and MTC devices. Similarly to the schemes of Broustis et al., the protocol introduces the role of a gateway, which contributes to minimizes the bandwidth consumption. However, the protocol does not guarantee any security property in presence of corrupted devices [18]. Cao et al. [12] proposed *GBAAM*, a group-based AKA that relies on the idea of using short aggregate signatures to reduce the overall signaling among the parties. The protocol benefits of pairing cryptography, which removes the need of a PKI. However, it requires each MTC device to run a classic AKA procedure to be registered with the same MME. As the devices normally require to access the network in a different geographic location than the location where they registered, this choice limits the suitability of the protocol as group-based AKA. Sun et al. [25] developed an authenticated group key agreement protocol for mobile environments. The general approach is interesting but it cannot fit the constraints of AKA in mobile telephony.

7 Conclusion

This paper demonstrates that a twenty-year-old protocol can meet modern challenges without revolutionary changes. The proposed group-based AKA is pivoted on the idea of using an inverted hash tree to manage a large number of devices efficiently. The cryptographic primitives of the protocol are based on MILENAGE so that the protocol can be adopted in the current standards. The implementation in OAI confirms that only minor modifications to EPS are needed to support the group-based AKA. The formal analysis of the protocol corroborates the security guarantees of the proposed solution, which proved to resist to threats due to colluding corrupted devices. The performance analysis yields promising results in term of latency and bandwidth consumption, with a remarkable gain when considering a large number of devices.

Future work includes the extension of the group-based AKA with support for secure handover among different MME and the resyncronization procedure of the sequence numbers. One approach is to use techniques from different areas, such as mobile cloud computing [29]. Another research direction is to support dynamic groups with key forward/backward secrecy: linkable group signature schemes [15, 20, 19] might be used on top of the protocol.

While research on areas of fundamental importance for 5G has already started (i.e, cloud security, IoT), research on 5G security is in its early stages. The results of our current implementation are promising since OAI relies on 4G network

standards. We expect even better results if the group-based AKA is implemented in the future 5G architecture.

References

1. 3GPP: Formal Analysis of the 3G Authentication Protocol. Technical Report 33.902 (2001)
2. 3GPP: Specification of the MILENAGE Algorithm Set. Technical Specification 35.205 (2001)
3. 3GPP: 3GPP System Architecture Evolution (SAE); Security architecture. Technical Specification 33.401 (2008)
4. 3GPP: MME Related Interfaces Based on Diameter Protocol. Technical Specification 29.272 (2008)
5. 3GPP: Service requirements for Machine-Type Communications (MTC); Stage 1. Technical Report 22.368 (2011)
6. Abadi, M., Fournet, C.: Mobile values, new names, and secure communication. In: POPL'01. pp. 104–115. ACM, New York (2001)
7. Alliance, O.S.: Openairinterface. http://www.openairinterface.org/ (August 2016)
8. Alt, S., Fouque, P.A., Macario-rat, G., Onete, C., Richard, B.: A Cryptographic Analysis of UMTS/LTE AKA, pp. 18–35. Springer International Publishing (2016)
9. Blanchet, B.: An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In: CSFW. pp. 82–96. IEEE Computer Society, Cape Breton, Canada (2001)
10. van den Broek, F., Verdult, R., de Ruiter, J.: Defeating imsi catchers. In: 22Nd ACM SIGSAC Conference on Computer and Communications Security. pp. 340–351. CCS '15, ACM (2015)
11. Broustis, I., Sundaram, G.S., Viswanathan, H.: Group authentication: A new paradigm for emerging applications. Bell Labs Technical Journal 17(3), 157–173 (2012)
12. Cao, J., Ma, M., Li, H.: Gbaam: group-based access authentication for mtc in LTE networks. Security and Communication Networks 8(17), 3282–3299 (2015)
13. Choi, D., Choi, H.K., Lee, S.Y.: A group-based security protocol for machine-type communications in LTE-advanced. Wireless Networks 21(2), 405–419 (2014)
14. Dolev, D., Yao, A.C.: On the security of public key protocols. Information Theory, IEEE Transactions on 29(2), 198–208 (1983)
15. Emura, K., Hayashi, T.: A Light-Weight Group Signature Scheme with Time-Token Dependent Linking, pp. 37–57. Springer International Publishing, Cham (2016)
16. Ericsson: Ericsson mobility report. Technical Report (2015)
17. Fouque, P.A., Onete, C., Richard, B.: Achieving better privacy for the 3gpp aka protocol. IACR Cryptology ePrint Archive 2016, 480 (2016)
18. Giustolisi, R., Gehrmann, C.: Threats to 5G group-based authentication. In: SECRYPT 2016 - Proceedings of the 13th International Conference on Security and Cryptography, August. SciTePress (2016)
19. Hwang, J.Y., Eom, S., Chang, K.Y., Lee, P.J., Nyang, D.: Anonymity-based authenticated key agreement with full binding property. Journal of Communications and Networks 18(2), 190–200 (2016)
20. Hwang, J.Y., Lee, S., Chung, B.H., Cho, H.S., Nyang, D.: Group signatures with controllable linkability for dynamic membership. Information Sciences 222, 761 – 778 (2013)
21. Lai, C., Li, H., Lu, R., Shen, X.S.: SE-AKA: A secure and efficient group authentication and key agreement protocol for LTE networks. Computer Networks 57,(17) (2013)
22. Nokia Siemens Networks: Signaling is growing 50% faster than data traffic. Technical Report (2012)
23. Oracle: Oracle communications lte diameter signaling index. 4th edition. White Paper (2015)

24. Ryan, M.D., Smyth, B.: Applied pi calculus. In: Formal Models and Techniques for Analyzing Security Protocols, chap. 6. IOS Press (2011)
25. Sun, H.M., He, B.Z., Chen, C.M., Wu, T.Y., Lin, C.H., Wang, H.: A provable authenticated group key agreement protocol for mobile environment. Information Sciences 321, 224 – 237 (2015)
26. Tang, C., Naumann, D.A., Wetzel, S.: Analysis of authentication and key establishment in inter-generational mobile telephony. In: IEEE 10th International Conference on Embedded and Ubiquitous Computing (HPCC_EUC). pp. 1605–1614 (2013)
27. WonderNetwork: Wonderproxy servers. https://wonderproxy.com/servers (August 2016)
28. Woo, T.Y., Lam, S.S.: A semantic model for authentication protocols. In: Research in Security and Privacy, 1993. Proceedings., 1993 IEEE Computer Society Symposium on. pp. 178–194 (1993)
29. Yang, X., Huang, X., Liu, J.K.: Efficient handover authentication with user anonymity and untraceability for mobile cloud computing. Future Generation Computer Systems 62, 190 – 195 (2016)

A Formal Specification of Security Requirements

ProVerif allows for syntactical extension of the applied pi-calculus, such as *events* and *choices*, to ease the specification of security requirements. Confidentiality can be modelled as a reachability property. The secrecy of a term m is preserved if an attacker, defined as an arbitrary process, cannot construct m from any run of the protocol. More precisely, the definition of *reachability-based secrecy* says that an attacker cannot build a process A that can output the secret term m.

Authentication can be defined using *correspondence assertions*. An event $\mathtt{e}$ is a message emitted into a special channel that is not under the control of the attacker. To model correspondence assertions, we annotate processes with events such as $\mathtt{e}\langle M_1, ...M_n\rangle$ and reason about the relationships ($\rightsquigarrow$) between events and their arguments in the form *"if an event* $\mathtt{e}\langle M_1, ...M_n\rangle$ *has been executed, then an event* $\mathtt{e}'\langle N_1, ...N_n\rangle$ *has been previously executed"*.

The applied pi-calculus supports the notion of *observation equivalence*. Informally, two processes are observational equivalent if an observer cannot distinguish the processes even if they handle different data or perform different computations. The indistinguishability characterization of the definition of observation equivalence allows us to capture privacy requirements.

Confidentiality. We check confidentiality of the session master key by proving that a fresh *secret*, which is encrypted with the key and sent in form of ciphertext on the public channel, cannot be obtained by the attacker. As soon as MTC and MME derive the session master key, each of them generates a ciphertext that encrypts the secret. They send the ciphertexts at the very end of the protocol run, accordingly the case. We specify the session master key confidentiality in ProVerif with the following query:

query attacker *(secret).*

ProVerif is suitable to prove confidentiality as it attempts to prove that a state in which the attacker knows the secret is unreachable. It follows that the secret is known only to MTC and MME.

Authentication. We specify MTC and serving network authentication requirements as correspondence assertions. Each assertion consists of a number of events. Events normally need to agree with some arguments to capture authentication. Thus, we introduce the terms that serve as arguments in our events as follows.

- IMSI refers to the permanent subscribe identity of the MTC;
- GID refers to the group identifiers of the MME;
- SN denotes the identifiers of the MME;
- K_{ASME} denotes the session master key;
- PATH_MTC denotes the path assigned to the MTC;
- HGK_MTC refers to the session individual key derived from the GK tree and associated to the MTC;
- RAND refers to the random value generated by the HSS;
- HCH_MTC refers to the session challenge key derived from the CH tree and associated to the MTC;

Having seen the arguments, we can define the list of events needed to specify mutual group authentication between MTC and MME. The events reflect the two cases defined in the group-based AKA protocol.

- `begin_mtc_A`$\langle$IMSI, GID, SN, $K_{ASME}\rangle$ means that the MME with identity SN begins the authentication of the MTC with identity IMSI and group GID, and associates it with the key K_{ASME}. The event regards the case A and is emitted by the MME after the authentication data response message.
- `begin_mtc_B`$\langle$PATH_MTC, GID, SN, HGK_MTC$\rangle$ means that the MME with identity SN begins the authentication of the MTC with path PATH_MTC and group GID, and associates it with the key HGK_MTC. The event regards the case B and is emitted by the MME after the attach request.
- `begin_mme_A`$\langle$IMSI, GID, SN, RAND, $K_{ASME}\rangle$ means that the MTC with identity IMSI and group GID begins the authentication of the MME with identity SN, and associates it with the random value RAND and key K_{ASME}. The event regards the case A and is emitted by the MTC after the authentication request.
- `begin_mme_B`$\langle$PATH_MTC, GID, SN, HCH_MTC, $K_{ASME}\rangle$ means that the MTC with path PATH_MTC and group GID begins the authentication of the MME with identity SN, and associate it with the keys HCH_MTC and K_{ASME}. The event regards the case B and is emitted by the MTC after the authentication request derivable message.
- `end_mtc_A`$\langle$IMSI, GID, SN, $K_{ASME}\rangle$ means that the MTC with identity IMSI and group GID concluded the authentication of the MME with identity SN, and computed the key K_{ASME}. The event regards the case A and is emitted by the MTC after the authentication response.
- `end_mtc_B`$\langle$PATH_MTC, GID, SN, HGK_MTC$\rangle$ means that the MTC with path PATH_MTC and group GID concluded the authentication of the MME with identity SN, and computed the key HGK_MTC. The event regards the case B and is emitted by the MTC after the authentication response derivable message.
- `end_mme_A`$\langle$IMSI, GID, SN, RAND, $K_{ASME}\rangle$ means that the MME with identity SN concluded the authentication of the MTC with identity IMSI and group GID, and associates it with the random value RAND and key K_{ASME}. The event regards the case A and is emitted by the MME after the successful verification of RES.

– $\texttt{end\_mme\_B}\langle \text{PATH\_MTC}, \text{GID}, \text{SN}, \text{HCH\_MTC}, \text{K}_{\text{ASME}} \rangle$ means that the MME with identity SN concluded the authentication of the MTC with path PATH_MTC and group GID, and associates it with keys HCH_MTC and K_{ASME}. The event regards the case B and is emitted by the MME after the successfully verification of RES_{D}.

To formalize mutual authentication we need to distinguish the authentication of the MME to MTC and the authentication of the MTC to the MME. Moreover, we need to distinguish the two cases. We formalize the authentication of the MME to MTC in Case A and Case B as follows.

Definition 1 (Serving network authentication (Case A)) *The protocol ensures serving network authentication for Case A if the correspondence assertion*

$$\begin{aligned} &\texttt{end\_mtc\_A}\langle \text{IMSI}, \text{GID}, \text{SN}, \text{K}_{\text{ASME}} \rangle \;\rightsquigarrow \\ &\texttt{begin\_mtc\_A}\langle \text{IMSI}, \text{GID}, \text{SN}, \text{K}_{\text{ASME}} \rangle \end{aligned}$$

is true on every execution trace.

Definition 2 (Serving network authentication (Case B)) *The protocol ensures serving network authentication for Case B if the correspondence assertion*

$$\begin{aligned} &\texttt{end\_mtc\_B}\langle \text{PATH\_MTC}, \text{GID}, \text{SN}, \text{HGK\_MTC} \rangle \;\rightsquigarrow \\ &\texttt{begin\_mtc\_B}\langle \text{PATH\_MTC}, \text{GID}, \text{SN}, \text{HGK\_MTC} \rangle \end{aligned}$$

is true on every execution trace.

In a similar way, we can formalize the authentication of the MTC to the MME in Case A and Case B.

Definition 3 (MTC authentication (Case A)) *The protocol ensures the authentication of MTC for Case A if the correspondence assertion*

$$\begin{aligned} &\texttt{end\_mme\_A}\langle \text{IMSI}, \text{GID}, \text{SN}, \text{RAND}, \text{K}_{\text{ASME}} \rangle \;\rightsquigarrow \\ &\texttt{begin\_mme\_A}\langle \text{IMSI}, \text{GID}, \text{SN}, \text{RAND}, \text{K}_{\text{ASME}} \rangle \end{aligned}$$

is true on every execution trace.

Definition 4 (MTC authentication (Case B)) *The protocol ensures the authentication of MTC for Case B if the correspondence assertion*

$$\begin{aligned} &\texttt{end\_mme\_B}\langle \text{PATH\_MTC}, \text{GID}, \text{SN}, \text{HCH\_MTC}, \text{K}_{\text{ASME}} \rangle \;\rightsquigarrow \\ &\texttt{begin\_mme\_B}\langle \text{PATH\_MTC}, \text{GID}, \text{SN}, \text{HCH\_MTC}, \text{K}_{\text{ASME}} \rangle \end{aligned}$$

is true on every execution trace.

Privacy. To model MTC identity privacy as equivalence property, we use the definition of labelled bisimilarity ($\approx_l$) as defined by Abadi and Fournet. We reason about the processes of MTC, MME, and HSS, which map to the corresponding roles. Each device playing the role of MTC execute the same process MTC but are instantiated with different variable values (e.g. IMSI, K). The requirement of MTC identity privacy can be conveniently specified as follows:

Definition 5 (MTC identity privacy)
$MTC\{^{\text{IMSI}_A}/_{id}\} \mid MME \mid HSS \approx_l MTC\{^{\text{IMSI}_B}/_{id}\} \mid MME \mid HSS$

The definition above states that two processes instantiated with two different IMSI values have to be observationally equivalent. Such equivalence means that an attacker cannot distinguish whether the MTC participating in the protocol run is the one associated with IMSI_A or IMSI_B, hence the privacy of the MTC identity is guaranteed. Note that the formulation of MTC identity privacy based on observational equivalence is more stringent than any formulation based on reachability. The latter formulation would need to assume that the attacker does not know any IMSI value in advance, an assumption that can be lifted up using observational equivalence.

The ProVerif code that describes the processes for MTC, MME, and HSS are respectively in Figure 6,7, and 8.

B Implementation and Analysis in OAI

The configuration used by our patched version of OAI is depicted in Figure 9. It includes three virtual machines running Linux inside a single host Intel Core i7 processor with 4GB RAM. In particular, one machine (VM1) runs the Openair-interface5G module that simulate an MTC device and the eNodeB base station. The other two machines (VM2 and VM3) run the OPENAIR-CN module. Note that OAI does not currently support multiple MTC device, namely the Openair-interface5G module include only a device. However, we can run multiple runs of Openairinterface5G module in different machines to instantiate several MTC devices at cost of instantiating the same number of base stations.

The communication between MTC device, MME, and HSS are performed through Ethernet interfaces. The communication between MTC device and eNodeB is done within VM1 and represents the *S1-U* interface in the 3GPP standard. The channel between VM1 and VM2 represent the *S1-MME* interface according the standard. VM3 is dedicated to the HSS, which uses a MySQL server for the storage of subscriber data.

B.1 Parameters

Some terms have no similar counterpart in the existing standards so we design them from scratch. This is the case of the two auxiliary parameters TREE HEIGHT and NODE DEPTH. The first gives the height $\mathcal{H}$ of the inverted hash trees. It is used as an indicator of how many bits of the path should be used. This parameter is needed because the path is communicated in full bytes even though the size of the actual path might not be divisible by eight. We thus specify that the size of TREE HEIGHT is one byte. The parameter NODE DEPTH gives the level on which the sub-root nodes GK_{ij} and CH_{ij} are placed in the inverted hash trees. The knowledge of PATH, TREE HEIGHT, and NODE DEPTH allows the MME to deduce the structure of the inverted hash tree and to assess whether next MTC devices can be served according Case A or Case B.

To compute the bandwidth consumption at NAS level, we consider the parameters and the sizes described in Table 4. We recall Equation 1 and Equation 2 concerning the bandwidth consumption for the group-based protocol for the

Fig. 6: The process of MTC in ProVerif

```
let MTC (imsi_mtc: id, key_mtc: key, gid: id, path_mtc: path,
         sqn: bitstring, o_mtc: bitstring, pos: bit) =
 new nonce_mtc: rand;
 out(ch, (gid, path_mtc, nonce_mtc, pos));
 in (ch, (case_x: int, aut_x: bitstring, sn_id: id, rand_x: rand));
 if case_x=caseA then
   (let (xored_sqn: bitstring, mac_sn: bitstring)=aut_x in
   if sqn=xor(f5((key_mtc, rand_x)),xored_sqn) then
      (if mac_sn=f1((sqn, rand_x), key_mtc) then
         let res=f2((key_mtc, rand_x)) in
         let ck=f3((key_mtc, rand_x)) in
         let ik=f4((key_mtc, rand_x)) in
         let kasme=kdf((xored_sqn, ck, ik, sn_id)) in
         event beginMMEa (imsi_mtc, gid, sn_id, rand_x, kasme);
         out(ch, res);
         let knasenc_mtc = kdf_nas_enc(kasme) in
         let knasint_mtc = kdf_nas_int(kasme) in
         out(ch, senc(secret, knasenc_mtc));
         in (ch, (nasmsgmac: bitstring , mac_nas: bitstring));
         if mac_nas=nas_mac(nasmsgmac, knasint_mtc) then
         let enc_complete_msg=senc(nas_complete_msg, knasenc_mtc) in
         out (ch , (nas_complete_msg, enc_complete_msg,
                    nas_mac(enc_complete_msg, knasint_mtc)));
         event endMTCa (imsi_mtc, gid, sn_id, kasme)
      else 0)
   else  0)
 else if case_x=caseB then
   let (f5_hgkmtc_nonce: bitstring, mac_hgkmtc: bitstring)=aut_x in
   let hgk_mtc=xor(h((key_mtc, rand_x)),o_mtc) in
   if f5((hgk_mtc, nonce_mtc))=f5_hgkmtc_nonce then
      if mac_hgkmtc=f1((nonce_mtc, rand_x, gid, sn_id, path_mtc),
                       bs_to_key(hgk_mtc)) then
         let res_b=f2((hgk_mtc, rand_x)) in
         let ck_b=f3((hgk_mtc, rand_x)) in
         let ik_b=f4((hgk_mtc, rand_x)) in
         let kasme_b=kdf((f5_hgkmtc_nonce, ck_b, ik_b, sn_id)) in
         event beginMMEb (path_mtc, gid, sn_id, rand_x, kasme_b);
         out(ch, res_b);
         let knasenc_mtc = kdf_nas_enc(kasme_b) in
         let knasint_mtc = kdf_nas_int(kasme_b) in
         out(ch, senc(secret, knasenc_mtc));
         in (ch, (nasmsgmac: bitstring , mac_nas: bitstring));
         if mac_nas=nas_mac(nasmsgmac, knasint_mtc) then
         let enc_complete_msg=senc(nas_complete_msg, knasenc_mtc) in
         out (ch , (nas_complete_msg, enc_complete_msg,
                    nas_mac(enc_complete_msg, knasint_mtc)));
         event endMTCb (path_mtc, gid, sn_id, hgk_mtc).
```

Fig. 7: The process of MME in ProVerif

```
let MME_init (sn_mme: id, hss_mme: key) =
 in(ch, (gid: id, path_mtc: path, nonce_mtc: rand, =sn_mme, pos: bit));
 if (path_mtc=get_child( get_parent(path_mtc), left) && pos=left) ||
    (path_mtc=get_child( get_parent(path_mtc), right) && pos=right)  then
         (MME_a(gid, path_mtc, sn_mme, hss_mme) |
          MME_b(gid, path_mtc, nonce_mtc, sn_mme, pos)).


let MME_a (gid: id, path_mtc: path, sn_mme: id, hss_mme: key) =
 out(ch, senc( (gid, path_mtc, sn_mme), hss_mme));
 in(ch, from_hss: bitstring);
 let (=gid, GKij: bitstring, CHij: bitstring, autn: bitstring,
      xres: bitstring, rand_hss: rand, kasme: key, imsi_mtc: id,
      n: bitstring, =path_mtc)=sdec(from_hss, hss_mme) in
 let pathx=get_parent(path_mtc) in
 insert mme_keys(GKij, CHij, gid, pathx, n);
 event beginMTCa (imsi_mtc, gid, sn_mme, kasme);
 out(ch, (caseA, autn, sn_mme, rand_hss));
 in(ch, =xres);
 let knasenc_mme = kdf_nas_enc(kasme) in
 let knasint_mme = kdf_nas_int(kasme) in
 out(ch, senc(secret, knasenc_mme));
 new nasmsgmac: bitstring;
 out(ch, (nasmsgmac, nas_mac(nasmsgmac, knasint_mme)));
 in(ch, (=nas_complete_msg, enc_msg: bitstring, mac_nas: bitstring));
 if mac_nas=nas_mac(enc_msg, knasint_mme) &&
    nas_complete_msg=sdec(enc_msg, knasenc_mme) then
  out(ch, senc(secret, knasenc_mme));
  event endMMEa (imsi_mtc, gid, sn_mme, rand_hss, kasme).


let MME_b (gid: id,path_mtc: path,nonce_mtc: rand,sn_mme: id,pos: bit)=
 get mme_keys(GKij, CHij, =gid, =get_parent(path_mtc), n) in
 let GKmtc=set_node(GKij,pos) in
 let hgkmtc=hash(GKmtc, n) in
 event beginMTCb (path_mtc, gid, sn_mme, hgkmtc);
 let CHmtc=set_node(CHij,pos) in
 let hchmtc=hash(CHmtc, n) in
 let f5_hgkmtc_nonce=f5((hgkmtc, nonce_mtc)) in
 let mac_hgkmtc=f1((nonce_mtc, hchmtc, gid, sn_mme, path_mtc),
                   bs_to_key(hgkmtc)) in
 out(ch, (caseB, (f5_hgkmtc_nonce, mac_hgkmtc), sn_mme, hchmtc));
 let ck=f3((hgkmtc, hchmtc)) in
 let ik=f4((hgkmtc, hchmtc)) in
 let kasme=kdf((f5_hgkmtc_nonce, ck, ik, sn_mme)) in
 in(ch, res_d: bitstring);
 if res_d=f2((hgkmtc, hchmtc)) then
  let knasenc_mme = kdf_nas_enc(kasme) in
  let knasint_mme = kdf_nas_int(kasme) in
  out(ch, senc(secret, knasenc_mme));
  new nasmsgmac: bitstring;
  out(ch, (nasmsgmac, nas_mac(nasmsgmac, knasint_mme)));
  in(ch, (=nas_complete_msg, enc_msg: bitstring, mac_nas: bitstring));
  if mac_nas=nas_mac(enc_msg, knasint_mme) &&
     nas_complete_msg=sdec(enc_msg, knasenc_mme) then
   event endMMEb (path_mtc, gid, sn_mme, bs_to_rand(hchmtc), kasme).
```

Fig. 8: The process of HSS in ProVerif

```
let HSS (sn_mme: id, mme_hss: key) =
 in(ch, from_mme: bitstring);
 let (gid: id, path_mtc: path, =sn_mme)=sdec(from_mme, mme_hss) in
 get hss_keys(=path_mtc, imsi, key_mtc, =gid, sqn, rootG, rootR, n) in
 new rand_hss: rand;
 let xored_sqn=xor(f5((key_mtc, rand_hss)),sqn)   in
 let mac_hss=f1((sqn, rand_hss), key_mtc) in
 let xres=f2((key_mtc, rand_hss)) in
 let ck=f3((key_mtc, rand_hss)) in
 let ik=f4((key_mtc, rand_hss)) in
 let kasme=kdf((xored_sqn, ck, ik, sn_mme)) in
 let autn=(xored_sqn, mac_hss) in
 out(ch, senc((gid, rootG, rootR, autn, xres, rand_hss, kasme, imsi, n,
               path_mtc), mme_hss)).
```

Table 4: Sizes of parameters of EPS-AKA and group-based AKA at NAS level. [1]The size of PATH is variable because it depends on the number of MTC devices considered.

Parameter	**Size (bytes)**	**EPS-AKA**	**Group-based AKA**	
			Case A	Case B
IMSI	9	✓	×	×
RAND	16	✓	✓	×
AUTN	17	✓	✓	×
RES	9	✓	✓	×
GID	9	×	✓	✓
PATH	Variable[1]	×	✓	✓
NONCE	16	×	✓	✓
N	6	×	✓	✓
HCH	16	×	×	✓
AUT_D	15	×	×	✓
RES_D	9	×	×	✓

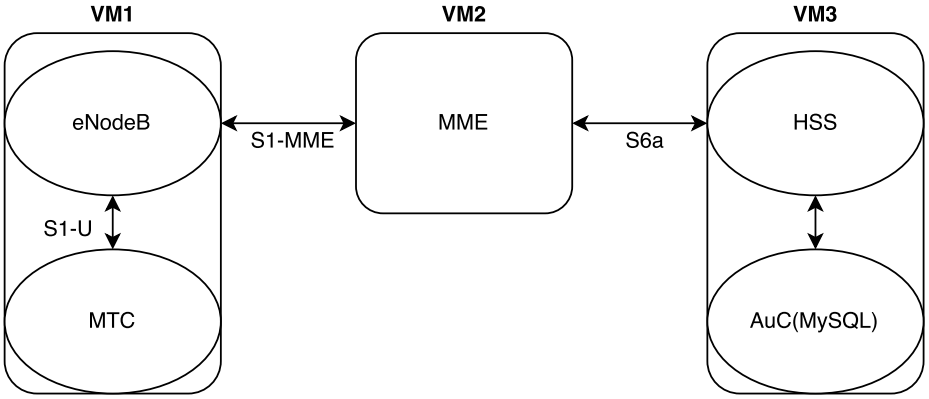


Fig. 9: Minimal network configuration needed for our patched version of OAI.

NAS and the S6a interface.

$$\textbf{BAND\_GB\_NAS} = \; m \times \left(\text{GID} + \frac{(\lceil \log_2 m \rceil \times 2 - 1)}{8} + 2 + \text{NONCE}\right) + (m-1) \times (\text{HCH} + \text{AUT}_\text{D} + \text{RES}_\text{D}) + \text{RAND} + \text{AUTN} + \text{RES}. \quad \text{(Equation 1)}$$

$$\textbf{BAND\_GB\_S6a} = \text{IMSI} + 2 \times \text{GID} + \text{RAND} + \text{XRES} + \text{AUTN} + \text{K}_\text{ASME} + \text{GK}_\text{ij} + \text{CH}_\text{ij} + \mathcal{H} + \text{SN}_\text{ID} + 2 \times \left(min(\text{PATH}) + \frac{\lceil \log_2 m \rceil \times 2 - 1)}{32} \times 4\right). \quad \text{(Equation 2)}$$

The bandwidth consumption for EPS-AKA at NAS level is

$$\textbf{Band\_EPS\_NAS} = m \times (\text{IMSI} + \text{RAND} + \text{AUTN} + \text{RES}). \quad \text{(Equation 3)}$$

Table 5: Sizes of parameters of EPS-AKA and group-based AKA in the S6A interface. [1]The size of PATH is variable because it depends on the number of MTC devices considered.

Parameter	**Size (bytes)**	**EPS-AKA**	**Group-based AKA**	
			Case A	Case B
IMSI	16	✓	✓	×
RAND	28	✓	✓	×
AUTN	28	✓	✓	×
XRES	20	✓	✓	×
K_ASME	44	✓	✓	×
SN_ID	16	✓	✓	×
N	18	×	✓	×
GID	16	×	✓	×
PATH	Variable[1]	×	✓	×
CH_ij	28	×	✓	×
GK_ij	28	×	✓	×
NODE DEPTH	16	×	✓	×
TREE HEIGHT	16	×	✓	×

Regarding the bandwidth consumption for the S6A interface, Diameter adds to each parameter 12 bytes for header and flags. Hence, the size of parameters

are bigger in S6A interface than in NAS. The values of the parameters are synthesized in Table 5. The bandwidth consumption for EPS-AKA can be computed as

$$\textbf{Band\_EPS\_s6A} = m \times (\text{IMSI} + \text{RAND} + \text{AUTN} + \text{XRES} + \text{K}_{\text{ASME}} + \text{SN}_{\text{ID}}) \quad \text{(Equation 4)}$$

Fig. 10: Bandwidth consumption comparison between EPS AKA and the group-based AKA on the NAS.

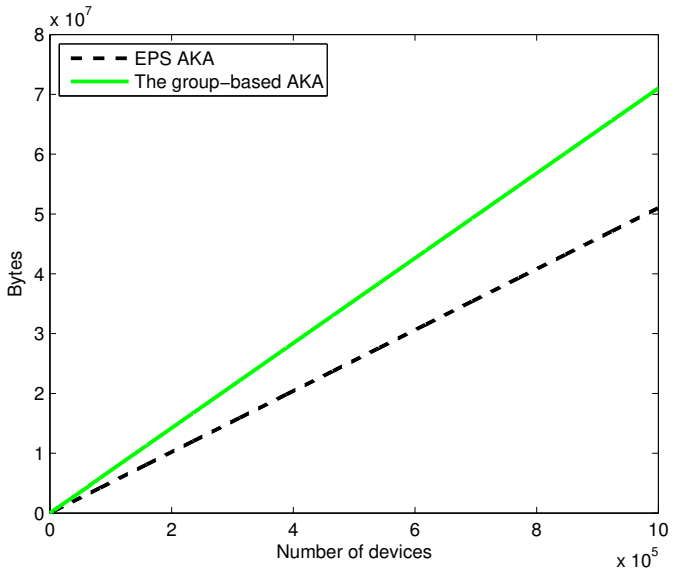


Figure 10 shows that the group-based AKA has more bandwidth consumption than the EPS-AKA at NAS level. This is because the attach request message in the group-based AKA includes the parameters PATH and NONCE in addition to the standard parameters. However, the bandwidth consumption rate is inverted in the S6a interface, as described in Figure 11. The group-based AKA consumes less bandwidth already when more than two MTC devices are considered. Notably, when the number of MTC devices to be served are more then three, the overall bandwidth consumption of group-based AKA is less than the one of EPS-AKA. This is depicted in Figure 12.

Fig. 11: Bandwidth consumption comparison between EPS AKA and group-based AKA on the S6a interface

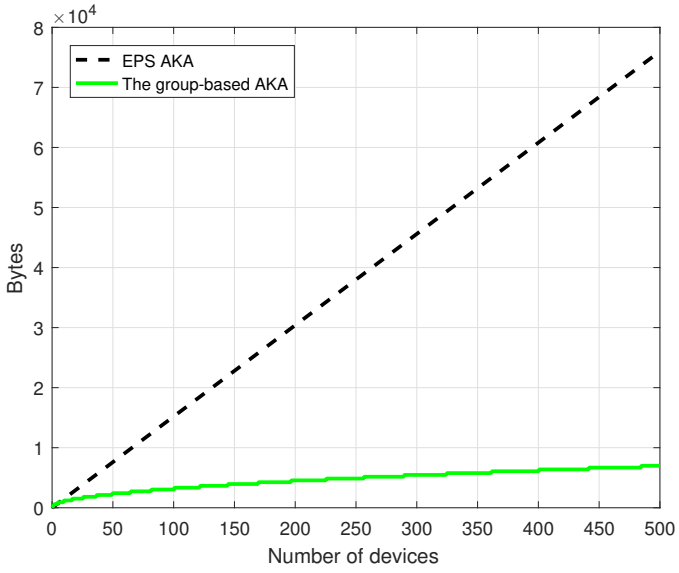


Fig. 12: Increase in NAS bandwidth consumption and decrease in S6a bandwidth consumption when the group-based AKA is used instead of EPS-AKA.