Security Extension for Relaxed Trust Requirement in Non3GPP Access to the EPS

Hiten Choudhury[1], Basav Roychoudhury[2] and Dilip Kr. Saikia[3]
(Corresponding author: Hiten Choudhury)
Department of Computer Science & IT, Cotton College State University[1]
Panbazar, Guwahati-1, Assam, India
Indian Institute of Management, Meghalaya, Shillong-14, India[2]
Department of Computer Science and Engineering, National Institute of Technology Meghalaya[3]
Laitumkhrah, Shillong-3, India
(Email: hiten.choudhury@gmail.com)
(Received Sept. 12, 2015; revised and accepted Dec. 7, 2015)

Abstract

Third Generation Partnership Project (3GPP) has standardized the Evolved Packet System (EPS) as a part of their Long Term Evolution System Architecture Evolution (LTE/SAE) initiative. In order to provide ubiquitous services to the subscribers and to facilitate interoperability, EPS supports multiple access technologies where both 3GPP and Non-3GPP defined access networks are allowed to connect to a common All-IP core network called the Evolved Packet Core (EPC). However, a factor that continues to limit this endeavor is the trust requirement with respect to the subscriber's identity privacy. There are occasions during Non-3GPP access to the EPS when intermediary network elements like the access networks that may even belong to third party operators have to be confided with the subscriber's permanent identity. In this paper, we propose a security extension that relaxes this need. Contrary to several other solutions proposed recently in this area, our solution can be adopted as an extension to the existing security mechanism.

Keywords: EPS, EPC, non 3GPP access, privacy, trust

1 Introduction

3GPP has standardized the EPS, as a part of their LTE/SAE initiative. EPS supports multiple access technologies, through a common All-IP core network called the Evolved Packet Core (EPC).

In order to expand the reach of 3GPP services, 3GPP has proposed TS 23.402 [5]. This specification specifies description for providing IP connectivity using Non-3GPP Access Networks (Non-3GPP ANs) like WiMAX, WLAN [22], etc., to the EPC.

Non-3GPP access can be split into two categories viz., trusted and untrusted. For trusted Non-3GPP access, the subscriber's User Equipment (UE) connects directly with the EPC through the Non-3GPP AN. Whereas, for untrusted Non-3GPP access, an Internet Protocol Security (IPsec) tunnel is established between the UE and the EPC [4]. The tunnel provides end to end confidentiality between the UE and the EPC, thereby relaxing the need for the subscriber and the EPC to trust the untrusted Non-3GPP AN with signalling/user data exchanged through it. Such trust relaxation facilitates interoperability, as it simplifies agreements between the 3GPP and the Non-3GPP operators. However, a factor that continues to limit interoperability is the trust requirement with respect to the subscriber's identity privacy.

Each UE is assigned a unique and a permanent identity called the International Mobile Subscriber Identity $IMSI$. This identity is assigned by the 3GPP service provider so that the UE may be uniquely identified. The $IMSI$ is a precious information that needs to be protected. Knowledge of the $IMSI$ of a subscriber may allow an adversary to track and amass comprehensive profiles about subscribers. Such profiling may expose an individual to various kind of risks, and above all may deprive an individual of his privacy. Thus, knowledge of the $IMSI$ should be restricted to the UE and its home network.

In the Authentication and Key Agreement (AKA) protocol used to provide access security in Non-3GPP access to the EPS, there are occasions when intermediary network elements like the Non-3GPP AN has to be confided with the $IMSI$ of the subscriber through the vulnerable radio link. Such trust requirement not only limits interoperability by complicating agreements/pacts between 3GPP and Non-3GPP operators, but also provides scope for eavesdroppers to compromise the $IMSI$. In today's context when multiple operators collaborate with each other to provide wider coverage, such trust requirement imposes restriction and adds overhead in providing ubiq-

uitous service to the subscribers.

In this paper, we propose a security extension for the AKA protocol used to provide access security in Non-3GPP access to EPS. The extension follows an end to end approach, where the knowledge of the $IMSI$ is restricted only to the UE and its home network, thereby relaxing the need to trust intermediary network elements like the Non-3GPP AN with the $IMSI$. Thus, the extension not only enhances identity privacy of the subscribers but also helps in setting up a conducive platform for flexible on-demand and on-the-fly agreements between the EPS and the Non-3GPP AN, instead of complicated prior agreements/pacts (with complicated trust requirements). Unlike several other solutions proposed in this area, the main strength of our proposal is that it can be adopted as an extension to the existing security mechanism. Moreover, it has to be implemented only at the operators level without tasking the intermediary network elements (that may even belong to third party operators). Hence, making it easier for an operator that already has numerous subscribers registered with it, to adopt this extension.

The rest of the paper is organized as follows: in Section 2, we present a simplified view of the security architecture for Non-3GPP access to the EPS; in Section 3, we discuss access security and the status of identity privacy in Non-3GPP access to the EPS; in Section 4, we review the literature for some of the related work done in this area; in Section 5, we put forward our security extension for the AKA protocol used during Non-3GPP access to the EPS; in Section 6, we perform a formal analysis of the proposed extension to prove that it meets its security goals; from Section 7 through Section 9, we perform computation, space and communication overhead analysis of the proposed security extension; finally in Section 10, we conclude the paper.

2 Security Architecture of Non-3GPP Access to the EPS

Figure 1, depicts a simplified view of the security architecture for Non-3GPP access to the EPS. The 3GPP Authentication Authorisation Accounting Server (3GPP AAA Server) is located at the Home Public Land Mobile Network (HPLMN). Its primary responsibility is to authenticate the subscriber, based on authentication information retrieved from the Home Subscription Server (HSS). The authentication signalling may pass via several AAA Proxies. The AAA Proxies that are used to relay AAA information may reside in any network between the Non-3GPP AN and the 3GPP AAA Server. The Packet Data Network Gateway (PDN GW) provides the UE with connectivity to the external packet data networks by being the point of exit and entry of traffic for the UE. The Serving Gateway (SGW) that is located in the Visitor Public Land Mobile Network (VPLMN), routes and forwards user data packets. Evolved Packet Data Gateway (ePDG) is a gateway with which an IPsec tunnel is established by

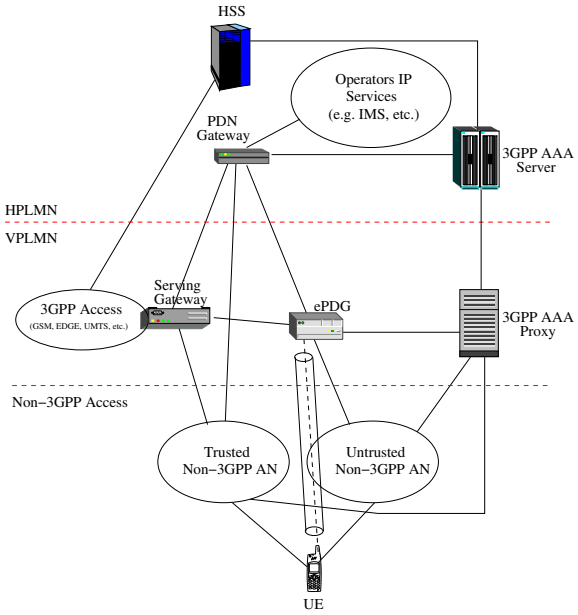

Figure 1: Security architecture for Non-3GPP access to the EPS

the UE for untrusted Non-3GPP access to EPS.

3 Access Security in Non-3GPP Access to the EPS

The AKA protocol adopted to provide access security for trusted/untrusted Non-3GPP access to EPS is Extensible Authentication Protocol for Authentication and Key Agreement (EAP-AKA) [4]. The EAP server for EAP-AKA is the 3GPP AAA Server residing in the EPC. The following subsections provides an overview of the use of EAP-AKA for trusted and untrusted Non-3GPP access.

3.1 Trusted Non-3GPP Access

In trusted Non-3GPP access, the UE connects with the EPC directly through the Non-3GPP AN. For access security, the UE and the 3GPP AAA Server executes EAP-AKA protocol between them. At the end of a successful EAP-AKA, necessary key materials for secured data communication between the UE and the Non-3GPP AN is established.

At first a connection is established between the UE and the Non-3GPP AN, using a Non-3GPP AN technology specific procedure. In order to begin the EAP-AKA procedure, the Non-3GPP AN sends an EAP Request/Identity message to the UE. In response, the UE sends an EAP Response/Identity message back to the Non-3GPP AN that contains the identity of the UE in Network Access Identifier (NAI) format [2]. The transmitted identity may either be a temporary identity allocated to the UE in the previous authentication or, in case of the first authentication, the $IMSI$. The mes-

sage is then routed towards the proper 3GPP-AAA Server through one or more AAA proxies with the help of the realm part of the NAI.

In case the NAI received from the UE contains a temporary identity, the 3GPP AAA Server extracts the corresponding $IMSI$ from it. By producing this $IMSI$, authentication data needed for mutual authentication between the UE and the 3GPP-AAA Server is acquired by the 3GPP-AAA Server from the HSS. The authentication data comprises of an Authentication Vector (AV), which is based on the authentication vectors used in UMTS [3]. It contains a random part $RAND$, an authenticator token $AUTN$ used for authenticating the network to the UE, an expected response part $XRES$, a 128-bit Integrity Key IK, and a 128-bit Cipher Key CK.

$$AV = (RAND, AUTN, XRES, IK, CK) \quad (1)$$

The $AUTN$ contains a sequence number SQN used to indicate freshness of the AV.

After an AV is acquired, the 3GPP-AAA Server derives new keying material viz. Master Session Key (MSK) and Extended Master Session Key ($EMSK$), from IK and CK. Fresh temporary identities may also be generated at this stage. The temporary identities are then encrypted and integrity protected with the keying material. The 3GPP-AAA server sends $RAND$, $AUTN$, a Message Authentication Code (MAC) (generated using the keying material) and the encrypted temporary identities to the Non-3GPP AN in an EAP Request/AKA-Challenge message. $RAND$, $AUTN$, MAC and the encrypted identities are then forwarded to the UE by the Non-3GPP AN.

The UE runs UMTS algorithm [3] on the Subscribers Identity Module (SIM). The SIM verifies that $AUTN$ is correct and hereby authenticates the network. If $AUTN$ is incorrect, the UE rejects the authentication. If $AUTN$ is correct, the UE computes RES, IK and CK. It then derives the keying material MSK and $EMSK$ from the newly computed IK and CK, and checks the received MAC with this keying material. If encrypted temporary identities were received, then the UE stores them for future authentications. The UE calculates a new MAC value covering the EAP message with the new key material. The UE then sends EAP Response/AKA-Challenge containing the calculated RES and the newly calculated MAC value to the Non-3GPP AN. The Non-3GPP AN in turn forwards the EAP Response/AKA-Challenge packet to 3GPP-AAA Server.

The 3GPP-AAA Server checks the received MAC and compares $XRES$ (received earlier from the HSS as part of AV) to the received RES. If all checks are successful, the 3GPP-AAA Server sends an EAP success message to Non-3GPP AN through a trusted link. The keying material MSK is also included in this message for Non-3GPP AN technology specific confidentiality and/or integrity protection; the Non-3GPP AN stores this keying material to be used in communication with the authenticated UE. The Non-3GPP AN informs the UE about the successful authentication with the EAP success message. This completes the EAP-AKA procedure that is required to register the UE with the Non-3GPP AN, at the end of which the UE and the authenticator in the Non-3GPP AN share keying material derived during the exchange.

3.2 Untrusted Non-3GPP Access

Unlike trusted Non-3GPP access, in untrusted Non-3GPP access, the UE connects with the EPC via the ePDG. The UE executes EAP-AKA using Internet Key Exchange version 2 (IKEv2) protocol [17] to establish an IPsec tunnel with the ePDG. The UE and the ePDG exchange a pair of messages to establish an IKEv2 channel in which the ePDG and UE negotiate cryptographic algorithms, exchange nonces and perform a Diffie Hellman exchange. In the remaining part of the authentication process, EAP-AKA (as explained in Section 3.1) is executed through this channel. After completion of the tunnel establishment and the EAP-AKA process, the UE and the ePDG share keying material that was derived during the process. The keying material is used for secured user data exchange through the tunnel during further communication between the UE and the ePDG.

3.3 Identity Privacy

In order to ensure identity privacy to the subscribers, the 3GPP-AAA Server generates and allocates temporary identities to the UE in a secured way (as discussed in Section 3.1). For identity presentation, the allocated temporary identities are transmitted by the UE instead of the permanent identity [4]. The UE does not interpret the temporary identities, it just stores them and uses them at the next authentication.

In-spite of the security measures, EAP-AKA has vulnerabilities due to which the intermediary network elements has to be entrusted with the $IMSI$ through the radio link.

3.3.1 Vulnerabilities during Trusted Non-3GPP Access

- The $IMSI$ has to be transmitted in clear text through the radio link for identity presentation during the very first authentication.
- If the 3GPP-AAA Server does not recognise a temporary identity, it will request the UE to send its permanent identity.
- A corrupt Non-3GPP AN may utilise the received $IMSI$ for various kind of malicious activities or may pass this identity to an unreliable party.
- A malicious/fake Non-3GPP AN may also take advantage of the above situation by creating a spurious EAP Request/Identity message to request the UE for its $IMSI$.

.

3.3.2 Vulnerabilities during Untrusted Non-3GPP Access

As a tunnel is established between the UE and the ePDG for secured communication during untrusted Non-3GPP access, there is no threats against identity privacy from passive attackers. However, there exist the following threats from active attackers:

- The protected channel is encrypted but not authenticated at the time of receiving the ($IMSI$). The IKEv2 messages, when using EAP, are authenticated at the end of the EAP exchange. So in case of a man-in-the middle attack, the attacker may pose as a genuine ePDG and may request the UE for the $IMSI$. Although the attack would eventually fail at the time of the authentication, the attacker would have managed to see the $IMSI$ in clear text by then.
- The $IMSI$ would be visible to the ePDG, which in roaming situations may be in the VPLMN.

4 Related Work

In mobile networks, the need to protect the identity privacy of a subscriber even from intermediary network elements like the visitor access network is well established. Herzberg *et al.* [15] pointed out that in an ideal situation no entity other than the subscriber himself and a responsible authority in the subscriber's home domain should know the real identity of the user. Even the authority in the visited network should not have any idea about the real identity.

Off late, several schemes were proposed by various researchers [7, 13, 14, 16, 19, 23]. However, none of these schemes are in line with EAP-AKA. For a mobile operator that already has a big subscriber base, changing over to a completely new authentication and key agreement protocol is a big challenge. Therefore, an ideal scheme would be the one that can be easily configured into EAP-AKA.

5 Our Proposed Security Extension

In this section, we propose a security extension where Knowledge of the $IMSI$ of a subscriber is restricted to the UE and the HSS. We propose to replace the transmission of the $IMSI$ with a Dynamic Mobile Subscriber Identity ($DMSI$). A fresh $DMSI$ is created as and when its need arises, and its value is derived from the most recent $RAND$ (Equation (1)) received during a successful EAP-AKA procedure. As a result, transmission of a $DMSI$ does not compromise the permanent identity of the user. The extension is implemented only at the Subscriber Identity Module (SIM) of the UE and the HSS. The extension can be introduced in the exiting system as an optional service, with the subscriber requiring to collect a new SIM in place of his/her existing SIM, or can be introduced on a rolling basis as new SIMs are issued. This work is based on the authors' earlier work that was proposed for UMTS [8] and LTE [10].

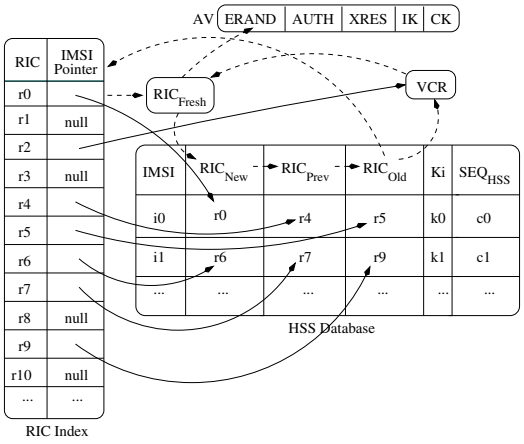

Figure 2: HSS database and the RIC index

In order to enable the UE to create a $DMSI$, a fresh random number called Random number for Identity Confidentiality (RIC) is used by the HSS. The HSS maintains a pool of $RICs$, some of which are in-use at a point of time (i.e., already assigned to different UEs). During each run of the EAP-AKA protocol, a not-in-use RIC selected from the pool is securely transferred to the UE. The selected RIC has to be sufficiently random, such that there is no correlation with a previously selected RIC. A mapping between the selected RIC and the $IMSI$ of the UE is maintained for a certain period of time (explained later in this section) at the HSS, so that the HSS can uniquely identify the UE with this RIC at a later instant. A $DMSI$ is assembled at the UE with the most recently received RIC. Thus, $DMSI$ is a function of RIC, details of which is explained later in this section (Equation (6)).

$$DMSI = f_i(RIC)$$

A RIC of size b $bits$, provide a pool of n unique RIC values. Where,

$$n = 2^b \tag{2}$$

We propose the size of RIC to be 32 bit, which is same as the size of Temporary Mobile Subscriber Identities ($TMSIs$) used in UMTS. A 32 bit RIC provides a pool of $2^{32} = 4.29$ billion (approx) unique RIC values. However, size of the RIC may even be determined by the operator depending on the anticipated subscriber base of the HSS, provided it is lesser than 128 $bits$.

In order to quickly locate a RIC in the HSS's database, a database index called *RIC-index* is maintained at the HSS (Figure 2). The *RIC-Index* contains all the $n = 2^b$ possible $RICs$ sorted according to their values. Each entry in the *RIC-Index* contains a pointer against it, which is called an *IMSI-Pointer*. This pointer either points to an $IMSI$ in the HSS's database or is *null*, depending on whether that particular RIC is allocated to an UE or is

unallocated at a particular instance of time. The collection of all the $RICs$ in the *RIC-Index* having a *null* value against it, forms a pool of not-in-use $RICs$, whereas the rest of the $RICs$ in the *RIC-Index* that points to some $IMSI$, represents the $RICs$ that are in-use. The total number of entries in the *RIC-Index* is fixed at n, irrespective of the number of $RICs$ that are currently in-use in the HSS's database. Even though such an index would require more disk space (Section 8), compared to an index whose size grows and shrinks according to the number of $RICs$ that are in-use at a particular instance in the HSS's database, it relieves the HSS of computational overhead involved during frequent insertions and deletions in the index.

In order to allocate a fresh RIC to the UE during every run of the EAP-AKA, a RIC called RIC_{Fresh} is chosen randomly from the pool of not-in-use $RICs$ at the HSS. RIC_{Fresh} is then cryptographically embedded into the $RAND$ part of the AV (Equation (1)) that reaches the UE as a challenge in due course of the AKA mechanism (as explained in Section 3). This resultant random number after embedding RIC into $RAND$ taking Ki as parameter is referred to as Embedded RAND ($ERAND$).

$$ERAND = f_{Embed_{Ki}}(RIC_{Fresh}, RAND) \quad (3)$$

The modified AV after embedding RIC into $RAND$ looks like the following:

$$AV = (ERAND, AUTH, XRES, IK, CK) \quad (4)$$

This $ERAND$ is now used by the 3GPP-AAA server, instead of the $RAND$ (as in the original EAP-AKA), to challenge the UE. Since the size of $RAND$ and $ERAND$ is same (i.e., 128 *bit*), the 3GPP-AAA server will not be able to perceive this change and will continue as before. Example algorithms to embed a 32 *bit* RIC into a 128 *bit* $RAND$ and to extract the embedded RIC form the $ERAND$ is proposed in [9]. Only the UE having knowledge of the long term shared key Ki is capable of extracting RIC from $ERAND$.

$$RIC = f_{Extract_{Ki}}(ERAND)$$

Multiple copies (m) of $RICs$ - the fresh and few previously generated $RICs$, stored in the fields: $RIC_{New}, RIC_{Prev}, RIC_{Old}$, etc. - are maintained at the HSS's database against a particular $IMSI$ along with the long term secret key Ki. This ensures robustness of the protocol even when an $ERAND$ gets lost in transition and does not reach the UE (Figure 2). Such an arrangement ensures that a mapping between the RIC that is currently stored at the UE and the corresponding $IMSI$ is always maintained at the HSS. However, like any other critical information such as the subscriber's security credentials, billing details, etc., in case of the $RICs$ maintained in the HSS's database also, it is the responsibility of the operator to have a robust backup mechanism against database crash.

We propose the value of m to be 4, i.e., $m = 4$; however, an operator may choose to have their own value for m. In UMTS, the size of the Temporary Mobile Subscriber Identity ($TMSI$) is 32 *bit*, out of which 2 *bits* are used to identify the type of identity (i.e., circuit switched, packet switched, etc.). The remaining 30 *bits* with 2^{30} possibilities are considered sufficient to allocate temporary identities to all the subscribers within a network. Therefore, in our proposal, $2^2 = 4$ is selected as the value for m. This will enable the HSS to have at the most $2^{30} = 1.073$ billion (approx) subscribers, which is 5.73 times more than the 187.302 million (approx) subscriber base of the largest cellular operator in India as of June, 2012 [20]). Thus, if s is the maximum number of subscribers that the operator wants the proposed extension to handle, then

$$s = n/m \quad (5)$$

where, n is the total number of possible $RICs$ in the entire pool and m is the number of $RICs$ maintained against each $IMSI$ in the HSS's database.

In order to verify the freshness of a received $DMSI$ and to prevent replay attacks, the HSS maintains a field called SEQ_{HSS} against every $IMSI$ in its database. SEQ_{HSS} is used to store the sequence number of the most recent $DMSI$ received from the UE.

5.1 Resolving a DMSI to an IMSI

In EAP-AKA, during situations (as explained in Sections 3.3.1 and 3.3.2) where the $IMSI$ needs to be transmitted by the UE, we propose to transmit a $DMSI$ (in NAI format) instead of the $IMSI$. The $DMSI$ is created using the RIC extracted from the most recent $ERAND$ received by the UE during an EAP-AKA as follows:

$$DMSI = MCC \| MNC \| RIC \| ERIC \quad (6)$$

where, MCC stands for the Mobile Country Code, MNC stands for the Mobile Network Code, and $ERIC$ is created by encrypting a padded RIC (say RIC_{padded}) with the Advanced Encryption Standard (AES) algorithm, taking the long term secret key Ki as parameter. Thus,

$$ERIC = f_{n_{Ki}}(RIC_{padded})$$

where,

$$RIC_{padded} = RIC \| SEQ_{UE} \| R$$

SEQ_{UE} is the value of a 32 bit counter that is maintained at the UE; whenever a new $DMSI$ is created for identity presentation, $SEQ_{UE}{}'s$ value is incremented by one. R is a $128 - (32 + b)$ bit random number. The inclusion of SEQ_{UE} ensures freshness of the $DMSIs$, whereas the inclusion of R completes the block size of 128 bits that is necessary to be fed into the AES cipher. In addition, R introduces sufficient amount of randomness to harden cryptanalysis of the ciphertext. The realm part of the $DMSI$ in NAI format helps the intermediate AAA proxy servers to guide the request to the appropriate 3GPP-AAA server. The 3GPP-AAA server treats the received

$DMSI$ as an $IMSI$ and therefore forwards the $DMSI$ along with a request for AV to the HSS. Thus, the onus of resolving the $DMSI$ is passed on to the HSS. On receipt of the request for AV, the HSS executes a sequence of instructions. First and foremost, the HSS resolves the $DMSI$, which is done by locating the RIC part of the received $DMSI$ in the *RIC-Index* and by mapping it to the corresponding $IMSI$ through the *IMSI-Pointer*. The $ERIC$ part of the $DMSI$ is then decrypted using AES and the corresponding key Ki. Thus,

$$RIC_{padded} = f_{dKi}(ERIC)$$

The RIC contained in RIC_{padded} is compared with the RIC part of the $DMSI$, the success of this comparison ensures that a malicious agent did not create the $DMSI$. The SEQ_{UE} part of RIC_{padded} is then compared with the value stored against SEQ_{HSS} field in the HSS's database. If $SEQ_{UE} > SEQ_{HSS}$, the request is proven as a fresh request. Failure of any of these two comparisons, leads to rejection of the request. If the request for AV is found to be fresh and from a genuine source (from the above comparisons), SEQ_{UE} is copied into SEQ_{HSS}

$$SEQ_{HSS} = SEQ_{UE}$$

and a fresh AV (Equation (1)) is generated using the procedure used in EAP-AKA.

5.2 Embedding a RIC into the RAND Part of AV

Whenever a RIC needs to be embedded into a $RAND$ at the HSS, a new RIC (RIC_{Fresh}) is selected from the pool of not-in-use $RICs$. In order to select RIC_{Fresh}, a b *bit* random number (say RN) is generated using a standard Pseudo Random Number Generator (PRNG). For this, we propose to use National Institute of Standards and Technology (NIST) recommended random number generator based on ANSI X9.31 Appendix A.2.4 Using AES [18], which appears in the list of approved random number generators for Federal Information Processing Standards Publication (FIPS PUB) 140-2 [11]. With a 128 *bit* key, this PRNG generates a 128 *bit* random number, the b most significant *bits* of which is selected as RN.

$$RN = f_{PRNG}(seed)$$

This RN is then searched for in the *RIC-Index*. If the *IMSI-Pointer* against RN in the *RIC-Index* is found to be *null*, RN is selected as RIC_{Fresh} and the *null* value is replaced with the address of the record in the HSS's database where the $IMSI$ is stored.

$$\begin{aligned} RIC_{Fresh} &= RN \\ RN.IMSI\text{-}Pointer &= Address\ of\ IMSI \end{aligned}$$

The oldest RIC value (i.e., RIC_{Old}) stored against the $IMSI$ is then returned to the pool of not-in-use RIC by searching for it in the *RIC-Index* and by setting the *IMSI-Pointer* against it to *null*.

$$RIC_{Old}.IMSI\text{-}Pointer = null$$

In case the *IMSI-Pointer* against RN in the *RIC-Index* is not *null*, it may be inferred that there is a collision, and RN is currently in-use. For collision resolution, a b *bit* variable called Variable for Collision Resolution (VCR) is used (Figure 2). The VCR contains a not-in-use RIC; an indication of this fact is specified in the *RIC-Index* by setting the *IMSI-Pointer* against the value in VCR to the address of VCR. At the very outset, during initialisation of the HSS's database, a b *bit* random number (say RN_0) is stored in the VCR and the *IMSI-Pointer* against it in the *RIC-Index* is set to the address of VCR.

$$\begin{aligned} RN_0 &= f_{PRNG}(seed) \\ VCR &= RN_0 \\ RN_0.IMSI\text{-}Pointer &= Address\ of VCR. \end{aligned}$$

Whenever there is a collision, the b *bit* value stored in the VCR is selected as RIC_{Fresh}. VCR is then searched for in the *RIC-Index* and the *IMSI-pointer* against it in the *RIC-Index* is made to point to the record in the HSS's database where the $IMSI$ is stored.

$$\begin{aligned} RIC_{Fresh} &= VCR \\ VCR.IMSI\text{-}Pointer &= Address\ of\ IMSI \end{aligned}$$

In order to replace the RIC stored in the VCR with a fresh RIC, the oldest RIC (i.e., RIC_{Old}) stored against the $IMSI$ is copied into VCR. RIC_{Old} is then searched for in the *RIC-Index* and the *IMSI-pointer* against it is set to the address of VCR.

$$\begin{aligned} VCR &= RIC_{Old} \\ RIC_{Old}.IMSI\text{-}Pointer &= Address\ of\ VCR \end{aligned}$$

The above procedure used to refresh the VCR introduces ample entropy to make the selection procedure of RIC even more random, because it is impossible to predict which $IMSI's\ RIC_{Old}$ value will refresh the VCR during the next EAP-AKA at the HSS. It solely depends on the call timing and usage pattern of all the active subscribers registered with the HSS. Moreover, the distribution process of the $RICs$ itself is random.

RIC_{Fresh} is then embedded into the $RAND$ part of AV (using Equation (3)). A copy of RIC_{Fresh} is also stored against the $IMSI$ in the HSS's database. To make space for RIC_{Fresh}, the value in RIC_{Prev} is copied into RIC_{Old} and the value in RIC_{New} is copied into RIC_{Prev}. Finally, the value in RIC_{Fresh} is copied into RIC_{New}.

$$\begin{aligned} RIC_{Old} &= RIC_{Prev} \\ RIC_{Prev} &= RIC_{New} \\ RIC_{New} &= RIC_{Fresh}. \end{aligned}$$

The AV is then send to the 3GPP-AAA server. The 3GPP-AAA server in turn, forwards a challenge containing $ERAND$ and $AUTH$ (extracted from AV) to the UE

Table 1: Functions used in the extension

Function	Details
f_i	Generates a $DMSI$ from a given RIC.
f_{Embed}	Embeds a 32 bit RIC into a 128 bit $RAND$.
$f_{Extract}$	Extracts the 32 bit RIC from a 128 bit $ERAND$.
f_n	Encrypts RIC_{Padded} to find $ERIC$.
f_d	Decrypts $ERIC$ to find RIC_{Padded}.
f_{PRNG}	Generates a 128 bit pseudo random number.

(through the Non-3GPP AN). The rest of the process continues in the same way as EAP-AKA. After successful authentication, the UE stores the $ERAND$ received as a challenge in its flash memory, to be used for identity presentation during subsequent authentications. The RIC embedded in $ERAND$ is extracted only when a $DMSI$ needs to be assembled.

An $ERAND$ (Say $ERAND_{First}$) that has a unique RIC called RIC_{First} embedded into it, is stored in the SIM's flash memory before a subscriber procures it from the service provider. RIC_{First} is also stored at the HSS's database and an entry in the *RIC-Index* is made accordingly. RIC_{First} is meant for one time usage for $DMSI$ creation during the very first authentication in the SIM's life time.

In some exceptional situations like failure of an ongoing EAP-AKA or due to an active attack by an adversary, the UE may not receive the next RIC (from the HSS) after it has already used the most recently received RIC to create and transmit a $DMSI$. In such a situation, if the need to transmit a $DMSI$ arises again, the UE can reuse the most recently received RIC to create the next $DMSI$. This can continue, as long as the UE does not receive a fresh RIC from the HSS (during a successful EAP-AKA). Even though such a recovery mechanism, in the worst case, may allow an adversary to link two or more failed EAP-AKA of the same UE, an adversary cannot gain anything from this, in terms of compromised identity privacy. Moreover, it is a much better option than transmitting the $IMSI$ itself.

A summary of all the functions used in the security extension is presented in Table 1.

5.3 Achievements

The key achievements of the proposed extension may be summarised as follows:

- *End to end user identity privacy*: Knowledge of $IMSI$ is confined only to the UE and the HSS.
- *Relaxed trust requirement*: Since the $IMSI$ is never revealed to the Non-3GPP AN or the ePDG, the HSS to Non-3GPP AN and HSS to ePDG trust relationship requirement with respect to $IMSI$ is relaxed. This relaxation will facilitate interoperability.
- *No overhead at the intermediary network*: The proposed security extension has to be implemented only at the UE and the HSS, intermediary elements like the Non-3GPP AN and the AAA servers can continue to maintain status quo. Thus, for an operator that adapts this extension, there is no additional cost of negotiation, implementation, computation, etc., to get the intermediary agents (that may even belong to third party operators) on board.
- *Can be adopted as an extension*: The proposed extension is in line with EAP-AKA and can be adopted as an extension.

Since, with the proposed extension, the $IMSI$ is never transmitted at any stage of the EAP-AKA protocol, all the vulnerabilities listed in Section 3.3.1 and Section 3.3.2 are eliminated; thereby relaxing the need to trust an intermediary network element with the permanent identity of a subscriber. Thus, with respect to signalling data (that does not reveal the $IMSI$ any more), the extension removes the need to establish a tunnel between the UE and the EPC during an untrusted Non-3GPP access. However, the need of the tunnel with respect to user data continues to exist.

6 Formal Analysis

We performed a formal analysis of the proposed scheme through an enhanced BAN logic [6] called AUTLOG [21]. A similar analysis is performed by 3GPP in [1]. Through this analysis, the security goals described in the following subsection are proven to be achieved by the proposed scheme.

6.1 Security Goals

$IMSI$ should be a shared secret between the UE and the HSS. The same should not be disclosed by the UE to any third party including the Non-3GPP AN.

G1: $UE \text{ believes } UE \overset{IMSI}{\longleftrightarrow} HSS$

When ever temporary identities fails to protect the permanent identity, a backup mechanism is followed according to our proposed extension, so that identity privacy may still be ensured to the subscriber. According to this mechanism (Section 5), a $DMSI$ (created with the RIC that is extracted from the most recent $RAND$ received by the UE) is transmitted in lieu of the $IMSI$. During every successful run of the EAP-AKA protocol, if the UE receives a fresh RIC, it can easily protect its permanent identity.

G2: $UE \text{ believes } UE \text{ has } RIC$

G3: $UE\ believes\ fresh(RIC)$

It should not be possible for anyone except the HSS (that has access to the RIC-Index) to map a $DMSI$ with its corresponding $IMSI$.

G4: $UE\ believes\ \neg(DMSI \equiv IMSI)$

6.2 Prerequisites

The UE recognises Ki and believes that it is a good key for communication with the HSS:

$$UE\ has\ Ki \tag{7}$$

$$UE\ recognises\ Ki \tag{8}$$

$$UE\ believes\ HSS \overset{Ki}{\longleftrightarrow} UE \tag{9}$$

Since the UE is capable of verifying freshness of SEQ contained in the $AUTN$ part of the received challenge, it believes in $SEQ's$ freshness.

$$UE\ believes\ fresh(SEQ) \tag{10}$$

The UE regards $ERAND$ as an atomic message.

$$(ERAND)_{UE} \equiv ERAND \tag{11}$$

The challenge received by the UE contains a Message Authentication Code (MAC) in the $AUTN$ part of the challenge. MAC is an encryption of SEQ and $ERAND$ with key Ki. The UE believes that it has not said MAC itself.

$$UE\ believes\ \neg(UE\ said\ enc(Ki, SEQ, ERAND)) \tag{12}$$

The UE believes that the HSS controls the freshness of RIC and that if the HSS says $ERAND$ along with an $AUTN$ with a fresh SEQ in it, the RIC contained in the $ERAND$ is also fresh.

$$UE\ believes\ HSS\ controls\ fresh(RIC) \tag{13}$$

$$UE\ believes\ (HSS\ says\ (SEQ, ERAND) \longrightarrow HSS\ believes\ fresh(RIC)) \tag{14}$$

$ERAND$ is an encrypted form of RIC. With knowledge of Ki, the UE can easily extract RIC from $ERAND$. Thus, UE is able to identify $ERAND$ with $enc(Ki, RIC)$.

$$(ERAND)_{UE} \equiv enc(Ki, RIC) \tag{15}$$

UE believes that HSS has jurisdiction and belief concerning the $IMSI$ as a shared secret between the UE and the HSS.

$$UE\ believes\ HSS\ controls\ HSS \overset{IMSI}{\longleftrightarrow} UE \tag{16}$$

$$UE\ believes\ HSS\ believes\ HSS \overset{IMSI}{\longleftrightarrow} UE \tag{17}$$

UE believes that HSS has jurisdiction on the fact that without access to the *RIC-Index*, RIC cannot be linked in any way with the corresponding $IMSI/MSIN$:

$$UE\ believes\ HSS\ controls\ \neg(RIC \equiv IMSI) \tag{18}$$

$$UE\ believes\ (HSS\ says\ ERAND \longrightarrow HSS\ believes\ \neg(f_x(Ki, ERAND) \equiv MSIN)) \tag{19}$$

$$UE\ believes\ (HSS\ believes\ \neg(f_x(Ki, ERAND) \equiv MSIN) \longrightarrow HSS\ believes\ \neg(RIC \equiv MSIN)) \tag{20}$$

UE sees the following:

$$UE\ sees\ ERAND, \{SEQ\}_{enc(Ki,ERAND)}, enc(Ki, SEQ, ERAND) \tag{21}$$

6.3 Proving the Security Goals

Figure 3 depicts step by step deduction of the security goals. Each circle in the figure represents an equation in this section. The label against each circle represents the inference rule/rules of AUTLOG that is/are used to derive the particular equation. The following $E(\#)$ denotes the Equation $(\#)$.

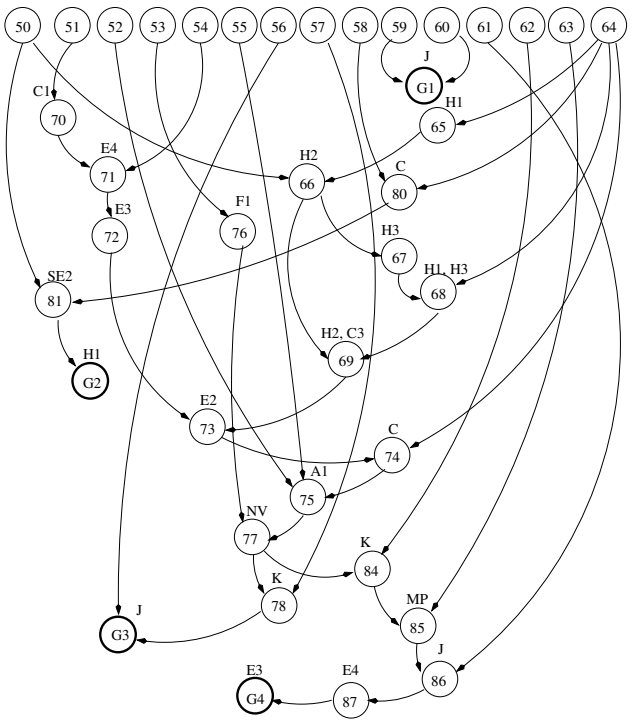

Figure 3: Deduction of security goals

$$E(21) \overset{H1}{\longrightarrow} UE\ has\ ERAND \tag{22}$$

$$E(7), E(22) \overset{H2}{\longrightarrow} UE\ has\ (Ki, ERAND) \tag{23}$$

$$E(23) \xrightarrow{H3} UE\ has\ enc(Ki, ERAND) \quad (24)$$

$$E(21), E(24) \xrightarrow{H1,H3} UE\ has\ SEQ \quad (25)$$

$$E(23), E(25) \xrightarrow{H2,C3} (enc(Ki, SEQ, ERAND))_{UE} \equiv enc((Ki, SEQ, ERAND)_{UE}) \quad (26)$$

$$E(8) \xrightarrow{C1} (Ki, SEQ, ERAND)_{UE} \quad (27)$$
$$\equiv ((Ki)_{UE}, (SEQ)_{UE}, (ERAND)_{UE})$$

$$E(11), E(27) \xrightarrow{E4} (Ki, SEQ, ERAND)_{UE} \quad (28)$$
$$\equiv (Ki, SEQ, ERAND)$$

$$E(28) \xrightarrow{E3} enc((Ki, SEQ, ERAND)_{UE}) \quad (29)$$
$$\equiv enc(Ki, SEQ, ERAND)$$

$$E(26), E(29) \xrightarrow{E2} (enc(Ki, SEQ, ERAND))_{UE}$$
$$\equiv enc(Ki, SEQ, ERAND) \quad (30)$$

$$E(21), E(30) \xrightarrow{C} UE\ believes\ UE\ sees\ enc(Ki, SEQ, ERAND) \quad (31)$$

$$E(31), E(9), E(12) \xrightarrow{A1} UE\ believes\ HSS\ said\ (SEQ, ERAND) \quad (32)$$

$$E(10) \xrightarrow{F1} UE\ believes\ fresh(SEQ, ERAND) \quad (33)$$

$$E(32), E(33) \xrightarrow{NV} UE\ believes\ HSS\ says\ (SEQ, ERAND) \quad (34)$$

$$E(34), E(14) \xrightarrow{K} UE\ believes\ HSS\ believes\ fresh(RIC) \quad (35)$$

$$E(13), E(35) \xrightarrow{J} \boxed{UE\ believes\ fresh(RIC)}\ (\mathbf{G3}) \quad (36)$$

$$E(21), E(15) \xrightarrow{C} UE\ believes\ UE\ sees\ enc(Ki, RIC) \quad (37)$$

$$E(37), E(7) \xrightarrow{SE2} UE\ believes\ UE\ sees\ RIC \quad (38)$$

$$E(38) \xrightarrow{H1} \boxed{UE\ believes\ UE\ has\ RIC}\ (\mathbf{G2}) \quad (39)$$

$$E(16), E(17) \xrightarrow{J} \boxed{UE\ believes\ (HSS \xleftrightarrow{IMSI} UE)}\ (\mathbf{G1}) \quad (40)$$

$$E(34), E(19) \xrightarrow{K} UE\ believes\ HSS\ believes\ \neg(f_x(Ki, ERAND) \equiv MSIN) \quad (41)$$

$$E(41), E(20) \xrightarrow{MP} UE\ believes\ HSS\ believes\ \neg(RIC \equiv MSIN) \quad (42)$$

$$E(18), E(42) \xrightarrow{J} UE\ believes\ \neg(RIC \equiv MSIN) \quad (43)$$

$$E(43) \xrightarrow{E4} UE\ believes\ \neg(MCC, MNC, RIC \equiv MCC, MNC, MSIN) \quad (44)$$

$$E(44) \xrightarrow{E3} \boxed{UE\ believes\ \neg(DMSI \equiv IMSI)}\ (\mathbf{G4}) \quad (45)$$

Hence, it is proven that the proposed extension meets its security goals.

7 Computational Overhead

In this section, we analyse the computational overhead of the proposed extension, using a methodology proposed in [12]. The core idea of this methodology is to determine the amount of basic operations required for implementation of an algorithm, reducing all other operations in terms of these basic operations. For overhead analysis of the proposed extension, all the other operations used in the extension are reduced to the following basic operations: Byte-wise AND, byte-wise OR, shift (bytes) and logical comparison operation. For XOR operations, we exploit the rule that a simple bit-wise XOR of x and y is equal to $x'y + y'x$. Since negations are negligible compared to AND/OR logical operations, a bit-wise XOR is considered as the sum of two bit-wise ANDs and one bit-wise OR. The methodology can be used to calculate the computational overhead of some of the key computations involved in the extension as follows:

1) *Encryption/Decryption with AES*: Let $N_{EncryptAES}$ and $N_{DecryptAES}$ be the number of basic operations needed by AES to encrypt a 128 *bit* plain-text and to decrypt a 128 *bit* cipher-text, respectively, using a 128 *bit* key. Granelli et. al. [12] found that 1720 byte-wise AND, 1268 byte-wise OR and 408 shift (byes) are involved in a 128/128 AES encryption, whereas, 5176 byte-wise AND, 3860 byte-wise OR and 1272 shift (byes) are involved in a 128/128 AES decryption. Thus,

$$N_{EncryptAES} = 3396$$
$$N_{DecryptAES} = 10308$$

2) *Encrypt RIC*: Let $N_{EncryptRIC}$ be the number of basic operations needed to encrypt a RIC to form an $ERIC$. 128/128 AES algorithm is used to carry out this encryption. Therefore,

$$N_{EncryptRIC} = N_{EncryptAES}$$

3) *Decrypt ERIC*: Let $N_{DecryptERIC}$ be the number of basic operations needed to decrypt an $ERIC$ to find a RIC. 128/128 AES algorithm is used to carry out this decryption. Therefore,

$$N_{DecryptERIC} = N_{DecryptAES}$$

4) *Search RIC*: Let $N_{SearchRIC}$ be the number of basic operations needed to search a *RIC* in the *RIC-Index*. The *RIC-Index* contains $n = 2^b$ number of entries arranged in sequential order, where b is the size of the *RIC* in bits. A *RIC* can be searched using binary search in $O(log_2 n)$ logical comparison operations. Thus, from Equation (5)

$$\begin{aligned} N_{SearchRIC} &= O(log_2 n) \\ &= O(log_2(m \times s)) \end{aligned}$$

5) *Select* RIC_{Fresh}: Let $N_{SelectRIC_{Fresh}}$ be the number of basic operations needed to select a not-in-use *RIC* as RIC_{Fresh}. At first a b *bit* random number (RN) is generated, using a standard PRNG (in say N_{Rn} number of operations). For this purpose, the PRNG based on ANSI X9.31 Using AES can be used. In this PRNG, 256 bit-wise XOR operations (which amounts to 64 byte-wise AND and 32 byte-wise OR operations) and 3 rounds of the AES encryption algorithm are performed to generate a pseudo random number [18]. Thus,

$$\begin{aligned} N_{Rn} &= 64 + 32 + 3 \times N_{EncryptAES} \\ &= 10284 \end{aligned}$$

RN is then searched in *RIC-Index* in $N_{SearchRIC}$ number of operations. If the pointer against RN is *null* (with this comparison requiring 1 comparison operation), then RN is selected as RIC_{Fresh} by setting the *IMSI-Pointer* against it to the address of the concerned $IMSI$. Otherwise, the value in VCR is selected as RIC_{Fresh}. VCR is then searched in the *RIC-index* in $N_{SearchRIC}$ number of operations and the *IMSI-Pointer* against it is set to the address of the concerned $IMSI$. Thus,

$$\begin{aligned} N_{SelectRIC_{Fresh}} &= N_{Rn} + 2 \times N_{SearchRIC} + 1 \\ &= 10284 + 2 \times O(log_2(m \times s)) + 1 \\ &= 10285 + 2 \times O(log_2(m \times s)) \end{aligned}$$

6) *Embed RIC into RAND*: Let $N_{EmbedRIC}$ be the number of basic operations needed to embed a *RIC* into a $RAND$. Considering the example algorithm proposed in [9], we found that a total of 32 bit-wise XOR operations (which amounts to 8 byte-wise AND and 4 byte-wise OR operations) and 1 round of the AES encryption algorithm are performed to embed a 32 *bit RIC* into a 128 *bit* $RAND$. Thus,

$$\begin{aligned} N_{EmbedRIC} &= 12 + N_{EncryptAES} \\ &= 3408 \end{aligned}$$

7) *Extract RIC from ERAND*: Let $N_{ExtractRIC}$ be the number of basic operations needed to extract the embedded *RIC* from an $ERAND$. Considering the algorithm proposed in [9], we found that a total of 32 bit-wise XOR operations (which amounts to 8 byte-wise AND and 4 byte-wise OR operations) and 1 round of the AES decryption algorithm are performed to extract the 32 *bit RIC* from a 128 *bit* $ERAND$. Thus,

$$\begin{aligned} N_{ExtractRIC} &= 8 + 4 + N_{DecryptAES} \\ &= 10320 \end{aligned}$$

8) *Return* RIC_{Old}: RIC_{Old} is searched in the *RIC-Index* in $N_{SearchRIC}$ operations. RIC_{Old} is then returned to the pool of not-in-use $RICs$ by setting the *IMSI-Pointer* against RIC_{Old} in the *RIC-Index* to either *null* or the address of VCR, depending on whether RN was selected as RIC_{Fresh} or VCR was selected as RIC_{Fresh}. Thus, if ($N_{ReturnRIC_{Old}}$) is the total time taken for this purpose,

$$\begin{aligned} N_{ReturnRIC_{Old}} &= N_{SearchRIC} \\ &= O(log_2(m \times s)) \end{aligned}$$

7.1 Computational Overhead at the UE

The proposed extension provides a backup mechanism that is used to identify the subscriber in situations where the temporary identities fail to identify the subscriber. According to this mechanism, a $DMSI$ is transmitted to the 3GPP-AAA Server in lieu of the permanent identity (i.e., the $IMSI$). The following computations are introduced at the UE when the UE identifies itself with a $DMSI$:

1) Extract *RIC* from the most recently received $ERAND$ in time say $T_{ExtractRIC}$.
2) Create $ERIC$ from the extracted *RIC* in time say $T_{EncryptRIC}$.

Thus, the computational overhead (say N_{UE}) introduced at the UE when a $DMSI$ is transmitted, can be calculated as follows:

$$\begin{aligned} N_{UE} &= N_{ExtractRIC} + N_{EncryptRIC} \\ &= 12 + N_{DecryptAES} + N_{EncryptAES} \\ &= 13716 \end{aligned}$$

Therefore, the overall computational overhead introduced at the UE by the extension is as follows:

$$N_{UE} = \begin{cases} 0 & \text{-when a temporary identity} \\ & \text{is transmitted,} \\ 13716 & \text{-when a } DMSI \text{ is transmitted.} \end{cases} \quad (46)$$

7.2 Computational Overhead at the HSS

During every EAP-AKA, the extension requires the HSS to embed a fresh *RIC* into the $RAND$ part of the AV. In addition, when a $DMSI$ is used for identity presentation, the HSS needs to resolve the $DMSI$.

The following computations (executed in say $N_{Resolve}$ number of operations) are introduced at the HSS when a $DMSI$ has to be resolved:

1) Search RIC in the *RIC-Index* in $N_{SearchRIC}$ operations.

2) Decrypt $ERIC$ to find RIC in $N_{DecryptRIC}$ operations.

3) Compare the decrypted RIC with the RIC contained in the $DMSI$ in 1 comparison operation.

4) Compare SEQ_{UE} and SEQ_{HSS} in 1 comparison operation.

$$\begin{aligned} N_{Resolve} &= N_{SearchRIC} + N_{DecryptRIC} + 1 + 1 \\ &= O(log_2(m \times s)) + N_{DecryptAES} + 2 \\ &= O(log_2(m \times s)) + 10310 \end{aligned} \quad (47)$$

The following computations (executed in say N_{Embed} number of operations) are introduced at the HSS, when RIC_{Fresh} has to be embedded into the $RAND$ part of AV.

1) Select RIC_{Fresh} in $N_{SelectRIC_{Fresh}}$ operations.

2) Embed RIC_{Fresh} into $RAND$ in $N_{EmbedRIC}$ operations.

3) Return the RIC_{Old} to the pool of not-in-used $RICs$ and store RIC_{Fresh} in the HSS's database in $N_{ReturnRIC_{Old}}$ operations.

Thus,

$$\begin{aligned} N_{Embed} &= N_{SelectRIC_{Fresh}} + N_{EmbedRIC} \\ &\quad + N_{ReturnRIC_{Old}} \\ &= 10285 + 2 \times O(log_2(m \times s)) + \\ &\quad 3408 + O(log_2(m \times s)) \\ &= 13693 + 3 \times O(log_2(m \times s)) \end{aligned} \quad (48)$$

The total computational overhead introduced at the HSS by the extension is equal to N_{Embed}, when a temporary identity is received at the HSS; whereas, it is equal to $N_{Resolve} + N_{EmbedRIC}$, when a $DMSI$ is received at the HSS.

Since we proposed the value of b to be 32 *bit* and the value of m to be 4 (Section 3), the value of $O(log_2(m \times s))$ can be derived using Equation (5) and Equation (2) as follows:

$$\begin{aligned} O(log_2(m \times s)) &= O(log_2(n)) \\ &= O(b) \\ &= 32 \end{aligned}$$

Thus, the overall computational overhead (say N_{HSS}) introduced at the HSS is as follows:

$$N_{HSS} = \begin{cases} 13789 & \text{-when a temporary identity} \\ & \text{is received,} \\ 24131 & \text{-when a } DMSI \text{ is received.} \end{cases}$$

Table 2: Summary of overhead analysis

Overhead	UE	HSS
Computational overhead when temporary identity is transmitted/received	0	13789
Computational overhead when $DMSI$ is transmitted/received	13716	24131
Time complexity	$O(1)$	$O(log_2(s))$
Space overhead	160 *bit*	$416 \times s$ *bit*
Communication overhead	0	0

7.3 Time Complexity

In this section, we derive the time complexity (in terms of the growth in subscriber base S) that an operator can expect at the UE and the HSS when the proposed security extension is adopted. From Equation (46), we can infer that the following time complexity is introduced at the UE.

$$T_{UE} = O(1)$$

From Equation (47) and Equation (48), the time complexity introduced at the HSS can be derived as follows:

$$\begin{aligned} T_{HSS} &= O(log_2(m \times s)) \\ &= O(log_2(s)), \quad m \text{ being a constant.} \end{aligned}$$

where, s is the maximum number of subscribers that the extension is expected to handle.

8 Space Overhead

8.1 Space Overhead at the UE

In order to store the most recently received $ERAND$, the UE needs 128 *bit* of space in the UE's flash memory. And, in order to maintain a 32 *bit* $DMSI$ counter, an additional 32 *bit* are required. Thus, the amount of space (say S_{UE}) required by the extension at the UE is:

$$S_{UE} = 160 \text{ } bit$$

8.2 Space Overhead at the HSS

Against every $IMSI$ in the HSS's database, $m = 4$ $RICs$ and a 32 *bit* SEQ_{HSS} value are stored. Thus, every record in the HSS's database will need an additional 160 *bit* space. In order to have provision for the maximum number of subscribers, i.e., s, the total amount of additional space (say S_{RIC}) needed at the HSS's database is:

$$S_{RIC} = 160 \times s \text{ } bit$$

In the *RIC-Index* there are n entries. Each of these entries has a 32 *bit RIC* and a 32 *bit IMSI-pointer*, with a total of 64 *bit*. Thus, the total amount of space (say $S_{RIC\text{-}Index}$) needed at the HSS's memory to accommodate the *RIC-Index* is:

$$\begin{aligned} S_{RIC\text{-}Index} &= 64 \times m \times s \ bit \\ &= 256 \times s \ bit, \quad \text{considering } m \text{ as } 4. \end{aligned}$$

Thus, the total amount of space (say S_{Total}) required by the extension at the the HSS is:

$$\begin{aligned} S_{Total} &= S_{RIC} + S_{RIC\text{-}Index} \\ &= 416 \times s \ bit \end{aligned}$$

where, s is the maximum number of subscribers that the extension is expected to handle.

9 Communication Overhead

In order to exchange information among the agents for smooth functioning of the extension, no additional messages are introduced to the original EAP-AKA protocol. Instead, the information are embedded into the existing messages of the EAP-AKA protocol. The procedure used for embedding the information is such that the format and size of the original messages remains the same. Thus, no communication overhead is imposed by the proposed extension at any of the agents involved in the EAP-AKA protocol.

Various overheads calculated in Section 7 through Section 9 are summarised in Table 2.

10 Conclusion

A factor that complicates Non-3GPP access to the EPS is the trust requirement on intermediary network elements like Non-3GPP AN and ePDG with respect to the subscriber's identity privacy. In this paper, we have put forward a security extension that improves the situation by taking an end to end approach, where the Non-3GPP AN or for that matter any intermediary element, need not be trusted with the permanent identity of the subscriber. This trust relaxation will not only facilitate interoperability, but also would enhance identity privacy of the subscriber. The strength of the extension lies in the fact that it can be adopted as an extension to the existing security mechanism. Moreover, it has to be implemented only at an operators level without tasking the intermediary network elements. Results of formal analysis and computational cost analysis show that the extension meets its security goals and is feasible with the existing infrastructure.

References

[1] 3GPP, "Formal Analysis of the 3G Authentication Protocol," TR 33.902, 3rd Generation Partnership Project (3GPP), 2001.

[2] 3GPP, "Numbering, addressing and identification," TS 23.003, 3rd Generation Partnership Project (3GPP), 2011.

[3] 3GPP, "3G Security; Security architecture," TS 33.102, 3rd Generation Partnership Project (3GPP), 2012.

[4] 3GPP, "3GPP System Architecture Evolution (SAE); Security aspects of non-3GPP accesses," TS 33.402, 3rd Generation Partnership Project (3GPP), 2012.

[5] 3GPP, "Architecture enhancements for non-3GPP accesses," TS 23.402, 3rd Generation Partnership Project (3GPP), 2012.

[6] M. Burrows, M. Abadi and R. M. Needham, "A logic of authentication," in *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 426, no. 1871, pp. 233–271, 1989.

[7] C. Chen, D. He, S. Chan, J. Bu, Y. Gao and R. Fan, "Lightweight and provably secure user authentication with anonymity for the global mobility network," *International Journal of Communication Systems*, vol. 24, no. 3, pp. 347–362, 2011.

[8] H. Choudhury, B. Roychoudhury and D. K. Saikia, "End-to-end user identity confidentiality for umts networks," in *2010 3rd IEEE International Conference on Computer Science and Information Technology*, vol. 2, pp. 46–50, 2010.

[9] H. Choudhury, B. Roychoudhury and D. K. Saikia, "Umts user identity confidentiality: An end-to-end solution," in *2011 Eighth International Conference on Wireless and Optical Communications Networks*, pp. 1–6, 2011.

[10] H. Choudhury, B. Roychoudhury and D. K. Saikia, "Enhancing user identity privacy in LTE," in *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 949–957, 2012.

[11] R. J. Easter and F. Carolyn, "Annex C: Approved Random Number Generators for FIPS PUB 140-2, Security Requirements for Cryptographic Modules," *NIST Federal Information Processing Standards (FIPS) Publications*, 2012.

[12] F. Granelli and G. Boato, "A Novel Methodology For Analysis of The Computational Complexity of Block Ciphers: Rijndael, Camellia And Shacal-2 Compared," TS DIT-04-004, Department of Information And Communication Technology, University of Trento, 2004.

[13] D. He, C. Chen, S. Chan and J. Bu, "Analysis and improvement of a secure and efficient handover authentication for wireless networks," *Communications Letters, IEEE*, vol. 16, no. 8, pp. 1270–1273, 2012.

[14] D. He, C. Chen, S. Chan and J. Bu, "Secure and efficient handover authentication based on bilinear pairing functions," *IEEE Transactions on Wireless Communications*, vol. 11, no. 1, pp. 48–53, 2012.

[15] A. Herzberg, H. Krawczyk and G. Tsudik, "On travelling incognito," in *1994 First IEEE Workshop on Mobile Computing Systems and Applications*, pp. 205–211, 1994.

[16] Q. Jiang, J. Ma, G. Li and L. Yang, "An enhanced authentication scheme with privacy preservation for roaming service in global mobility networks," *Wireless Personal Communications*, vol. 68, no. 4, pp. 1477–1491, 2013.

[17] C. Kaufman, P. Hoffman, Y. Nir and P. Eronen, "Internet key exchange protocol version 2 (IKEv2)," *The Internet Engineering Task Force Request for Comments (IETF RFC)*, vol. 5996, 2010.

[18] S. S. Keller, "NIST-Recommended Random Number Generator Based on ANSI X9. 31 Appendix A. 2.4 Using the 3-key Triple DES and AES Algorithms," *National Institute of Standards an Technology (NIST)*, 2005.

[19] H. Liu and M. Liang, "Privacy-preserving registration protocol for mobile network," *International Journal of Communication Systems*, vol. 27, no. 10, pp. 1653–1671, 2014.

[20] P. Raj, "Highlights on telecom subscription data as on 30th june 2012," Press Release 171/2012, Telecom Regulatory Authority of India, 2012.

[21] G. Wedel and V. Kessler, "Formal semantics for authentication logics," in *Computer Security (Esorics'96)*, pp. 219–241, Springer, 1996.

[22] C. C. Yang, K. H. Chu and Y. W. Yang, "3G and WLAN interworking security: Current status and key issues," *International Journal of Network Security*, vol. 2, no. 1, pp. 1–13, 2006.

[23] T. Zhou and J. Xu, "Provable secure authentication protocol with anonymity for roaming service in global mobility networks," *Computer Networks*, vol. 55, no. 1, pp. 205–213, 2011.

Hiten Choudhury is an Assistant Professor in the Dept. of Computer Science and Information Technology at Cotton College State University, Assam, India. He has a Ph.D in Computer Science & Engineering from Tezpur Central University, Assam, India. His areas of research interest include wireless network security and authentication protocols.

Basav Roychoudhury is an Associate Professor of Information Systems at Indian Institute of Management, Shillong. He has a Ph.D in Computer Science and Engineering from Tezpur University, Assam, India. He has worked in the area of computer network protocols and security, enterprise systems, etc.

Dilip Kr. Saikia received his BE, M. Tech and Ph. D degrees from Madras University, IIT Madras and IIT Kharagpur in 1981, 1983 and 1996 respectively. He is currently a professor at the National Institute of Technology Meghalaya, India and the director of the institute. His current areas of research interest include network security, sensor networks and malware detection.