Ravishankar Borgaonkar, Lucca Hirschi*, Shinjo Park, and Altaf Shaik

New Privacy Threat on 3G, 4G, and Upcoming 5G AKA Protocols

Abstract: Mobile communications are used by more than two thirds of the world population who expect security and privacy guarantees. The *3rd Generation Partnership Project* (3GPP) responsible for the worldwide standardization of mobile communication has designed and mandated the use of the *AKA protocol* to protect the subscribers' mobile services. Even though privacy was a requirement, numerous subscriber location attacks have been demonstrated against AKA, some of which have been fixed or mitigated in the enhanced AKA protocol designed for 5G.

In this paper, we reveal a new privacy attack against all variants of the AKA protocol, including 5G AKA, that breaches subscriber privacy more severely than known location privacy attacks do. Our attack exploits a new logical vulnerability we uncovered that would require dedicated fixes. We demonstrate the practical feasibility of our attack using low cost and widely available setups. Finally we conduct a security analysis of the vulnerability and discuss countermeasures to remedy our attack.

1 Introduction

As of 2018, around 5 billion mobile subscribers equipped with *Universal Subscriber Identity Module* cards (USIM) are accessing cellular network services (*e.g.,* Internet, calls), mostly relying on 3G or 4G technologies [1] (*e.g.,* ca. 75% of connections in Europe and North America). With growing importance of cellular network services in our daily activities, there is a crucial need to provide security and privacy protection to mobile subscribers.

Ravishankar Borgaonkar: SINTEF Digital, Norway. E-mail: ravi.borgaonkar@sintef.no
Lucca Hirschi*: ETH Zurich, Switzerland. E-mail: lucca.hirschi@inf.ethz.ch
Shinjo Park: Technische Universität Berlin, Germany. E-mail: pshinjo@sect.tu-berlin.de
Altaf Shaik: Technische Universität, Berlin, Germany. E-mail: altaf329@sect.tu-berlin.de

The *3rd Generation Partnership Project* (3GPP) group, responsible for the standardization of 3G, 4G, and 5G technologies, designed the *Authentication and Key Agreement* (AKA) protocol that aims at mutually authenticating a phone equipped with a USIM card with networks, and establishing keys to protect subsequent communications. This protocol is notably implemented in all 3G and 4G USIM cards and cellular networks worldwide. For 5G, the 3GPP has standardized *5G AKA*, an enhanced version of AKA [2]. In addition, AKA is also used in Extensible Authentication Protocol (EAP) mechanisms (*e.g.,* EAP-AKA, EPS-AKA, EPS-AKA') to secure point-to-point protocol authentication methods, wireless LAN internetworking, and generic authentication architectures including generic solutions for securing HTTP based services [3, 4]. In a nutshell, AKA is a challenge-response protocol mainly based on symmetric cryptography and a *sequence number* (SQN) to verify freshness of challenges, preventing replay attacks.

While privacy was an explicit requirement for 3G and 4G [5, 6], numerous *fake base station attacks* have been shown to compromise subscriber privacy in these networks [7–15]. The fake base station attacks typically exploit weaknesses in the AKA protocol such as the non-protected identity request mechanism (*e.g.,* with IMSI-catchers [9–15]) and the privacy-leak resulting from authentication failure messages [7, 8]. In practice, those attacks break *subscriber location privacy*: an attacker can identify if certain subscribers are in the range of attacker's fake base stations. For 5G, the 3GPP has improved AKA in order to mitigate these well-known privacy issues [2]. The 5G AKA protocol notably introduces randomized asymmetric encryption for protecting identifiers sent prior to authentication.

Contributions. In this paper, we reveal a new privacy attack against all variants of the AKA protocol (including 5G AKA and EAP variants) that breaches subscribers' privacy more severely than known location privacy attacks do. Our attack exploits a new logical vulnerability we uncovered in the protocol specification that would require dedicated fixes. More precisely, we make the following contributions:

1. New logical vulnerability and privacy attacks on AKA. We found a new logical vulnerability in the specifications of all aforementioned variants of

AKA: the protection mechanism of the SQN can be defeated under specific replay attacks due to its use of Exclusive-OR (XOR) and a lack of randomness. We show how to leverage this vulnerability to break the confidentiality of SQN, thus defeating the purpose of a dedicated protection mechanism and breaking an explicit privacy requirement [6]. We show that partly learning SQN leads to a new class of privacy attacks (*i.e., activity monitoring attacks*): an active attacker can leverage fake base stations and our attack to learn information about targeted subscribers' mobile service consumption, even when subscribers move away from the attack area (*e.g.,* range of a fake base station). This is in stark contrast to location attacks that do not reveal service consumption and requires the targeted subscribers to stay in attack areas. Less importantly, we show that our logical vulnerability also yields a new location attack.

2. Low-cost proof of concept. We demonstrate the feasibility of our attack using widely available and low-cost setup on commercial 4G networks in several European countries. Our attack affects all 3G and 4G devices currently deployed all over the world and future 5G devices (according to the specification [2]).

3. Security analysis and countermeasures. We discuss the weaknesses of the AKA protocol, its deployment, design trade-offs, and the overall cellular architecture that have made possible our attack in order to draw lessons for the future generation networks. We propose countermeasures to our attack and establish formal security guarantees using the state-of-the-art tool Tamarin [16].

Impact. Unlike the previously known location privacy attacks [7–15], we disclose a new type of privacy attack enabling *subscriber activity monitoring*. We now discuss subsequent impacts of our attack. First, an attacker can learn 3G, 4G, and 5G subscribers' typical activity patterns (*e.g.,* number of calls, SMSs sent in a given time). We stress that those activity patterns can be monitored remotely for a long time even if, most of the time, subscribers move away from the attack areas. We followed the responsible disclosure procedure and reported our findings to the 3GPP, *GSM Association* (GSMA), several manufacturers (Ericsson, Nokia, and Huawei), and carriers (Deutsche Telekom and Vodafone UK). Our findings were acknowledged by the 3GPP and GSMA and remedial actions are underway to improve the protocol for next generations. Finally, while 5G AKA will suffer from our attack in the first deployment of 5G (*i.e.,* Release 15 [2], phase 1), we are still hopeful that 5G AKA could be fixed before the deployment of the second phase (Release 16, to be completed by the end of 2019).

Outline. In section 2, we explain the general cellular security architecture and the core protocol underlying all AKA protocol variants. Section 3 presents our new logical vulnerability breaking the confidentiality of SQN. In Section 4, we show how the latter can be exploited to mount activity monitoring and location attacks and discuss practical impact. In Section 5, we show feasibility of our attacks in real 4G networks. We conduct a security analysis in Section 6 and provide potential countermeasures in Section 7. We conclude in Section 8.

2 Background

We first give an high-level overview of the security architecture used in 3G, 4G and 5G networks and explain how mobile subscribers are authenticated to the network using the AKA protocol. We then describe the different privacy requirements of 3G, 4G, and 5G networks outlined in the 3GPP specification. We conclude with a discussion on threat models, previously known attacks against the AKA protocol, formerly proposed fixes, and a comparison with our attack.

2.1 Security Architecture and the AKA Protocol

We describe a simplified view of the security architecture deployed in cellular networks and focus on the parts that are necessary to understand our attack. We also slightly simplify our description of the AKA protocol for the sake of clarity and only focus on the core protocol which is common to all the AKA protocol variants. Further, we adopt a simplified terminology since the official terminology heavily depends on the generation. We refer the knowledgeable reader to Appendix A where an informal correspondence with standardized terminologies for the different generations is given.

2.1.1 Architecture

The cellular network architecture mainly consist of three components. First, *User Equipments* (*UEs*) are carried by *subscribers* (we shall use both terms alternatively) and are typically smartphones or IoT devices containing a USIM card. Second, *Home Networks* (*HNs*) contain a

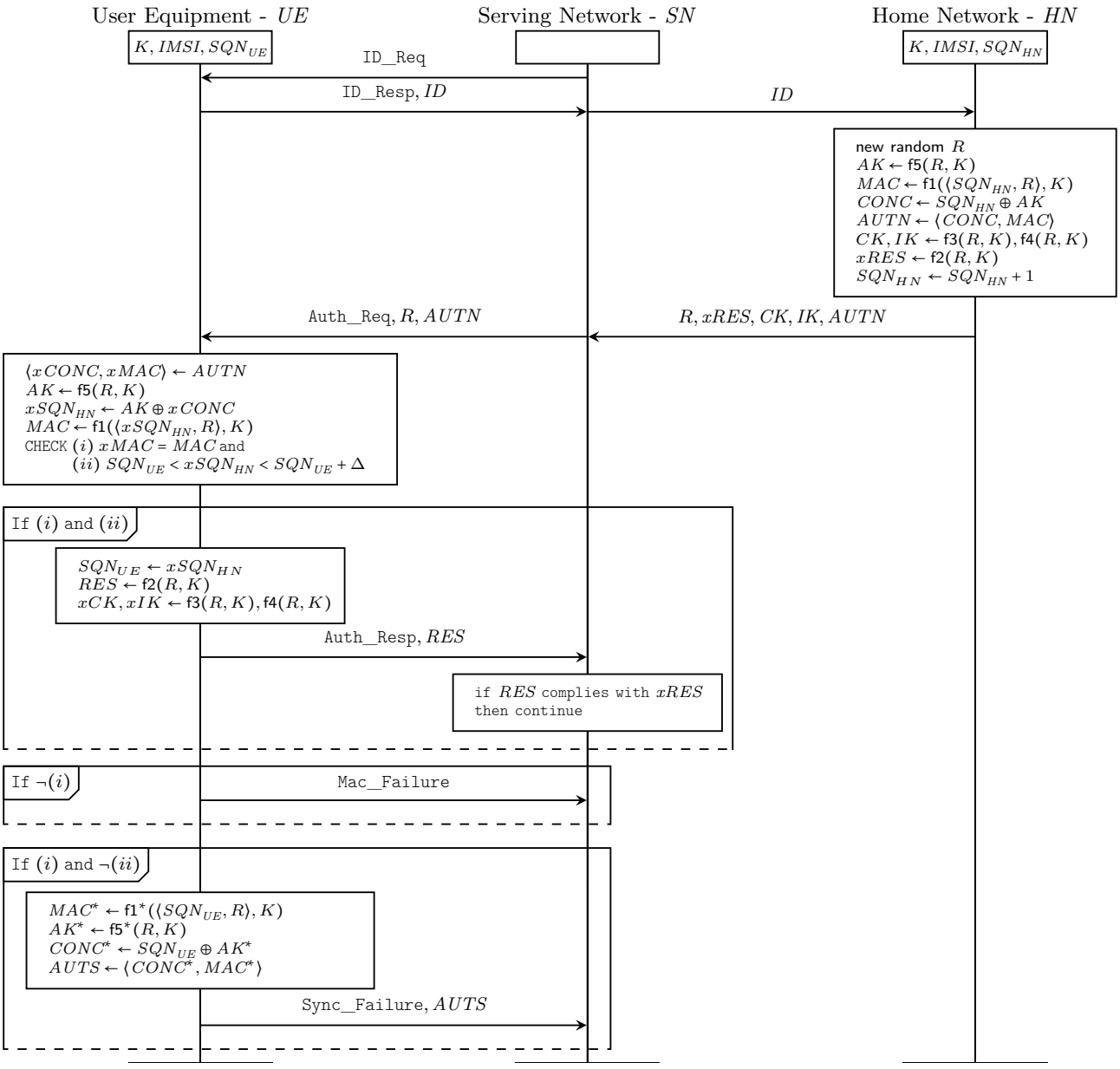


Fig. 1. The AKA protocol. K denotes K_{IMSI}.

database of their subscribers' and their corresponding USIM cards and are in charge of their authentication. However, it is often the case that *UEs* are in locations where their corresponding *HN* has no *base station* (*i.e.,* antennas that may connect *UEs* to the network). Therefore, the architecture also considers a third entity: the *Serving Networks* (*SN*) to which *UEs* may attach to and that play the role of relays to *HNs*.

Each *UE* contains a USIM having cryptographic capabilities (*e.g.,* symmetric encryption, MAC) which notably stores:

- an unique and permanent subscriber *identity*, called *International Mobile Subscriber Identity* (IMSI),
- a unique, permanent secret *symmetric key* that we indicate as K_{IMSI} (used as a share secret between a *UE* and its corresponding *HN*),
- and a 48-bits counter, called *Sequence Number* that we denote as *SQN* (used as a replay protection, as explained later in this section).

The *HN* associated to some USIM card stores the same information in its database. When the context is clear, we use K to refer to K_{IMSI}.

2.1.2 The AKA Protocol

When a *UE* attaches to a *SN* (or when *UE* requests access to a service such as sending a SMS or making a call), it needs to establish a secure channel with the *SN* (ensuring confidentiality of the user data) after authenticating itself to its corresponding *HN* (mainly for billing purpose) and authenticating its *HN* (so that fake *SN* cannot establish such a channel and break confidentiality). To do so, 3GPP has standardized the AKA protocol (the only authentication method allowed for 3G, 4G and 5G for 3GPP access). While this protocol has evolved with each generation [2, 6, 17], its core specification remained the same. We shall only focus on this core protocol that already suffers from our attack.

The AKA protocol achieves mutual authentication and key exchange between an *UE* and its corresponding *HN*, relying on some *SN* that is known by the *HN*. It allows some *UE* and *SN* to establish session keys to be used to secure subsequent communications (*e.g.,* integrity and confidentiality of calls or SMSs). As mentioned before, the key K_{IMSI} is used as a long-term shared secret, and *SQN* is used as a replay protection for the *UE*. While *SQN* is expected to be synchronized between the *UE* and *HN*, it may become out-of-sync. We thus use SQN_{UE} (resp. SQN_{HN}) to refer to the SQN value stored in the *UE* (resp. *HN*). The AKA protocol is made up of 3 main phases: *identification*, *challenge-response*, and *re-synchronization procedure* (that is optional and aims at updating SQN on the *HN* side in case SQN is out of-sync). The whole protocol flow is depicted in Figure 1.

Identification. First, the *SN* identifies the *UE*. If the current *UE*'s identity is unknown to the *SN*, it may ask for the permanent identity *IMSI* (or an encryption thereof in 5G) by sending an *Identity Request* message. The *UE* then gives its identity in an *Identity Response* message. This identity enables the *SN* to request authentication material to the appropriate *HN* in the next phase. In 5G, *UE* never reveals its permanent identity in plaintext. It rather sends a randomized encryption of it, protected with the *HN*'s public key, along with the *HN*'s identity (forming the so-called *SUCI* [2]).

Challenge-response. Upon reception of a request for authentication material from a *SN*, the *HN* computes an *authentication challenge* made of a random nonce R and some message $AUTN$. In addition, the expected authentication response $xRES = \mathsf{f2}(R,\ K_{IMSI})$, the encryption key CK, and the integrity key IK are also computed by *HN* (but not sent by *SN* to *UE*). Note that, in 5G, the message $xRES$ has a slightly different form; this has no impact on our attack. The functions $\mathsf{f1} - \mathsf{f5}$, used to compute the authentication parameters, are one-way keyed cryptographic functions completely unrelated, and $\oplus$ denotes the eXclusive-OR (XOR) operator.

$AUTN$ contains a MAC (Message Authentication Code) of the concatenation of R with the corresponding sequence number SQN_{HN} stored for this subscriber. A new sequence number is generated by increment of the counter. The sequence number SQN_{HN} allows the *UE* to verify the freshness of the authentication request to defend against replay attacks and the MAC proves authenticity of the challenge.

The *UE* replies with an *Authentication Response* message when the authentication is successful, or *Authentication Failure* message with the cause of failure otherwise. To check whether authentication is successful or not, the *UE* extracts SQN_{HN} from $AUTN$ and checks that: (i) MAC is a correct MAC value w.r.t. K_{IMSI}, replies `Mac_failure` if it is not the case; (ii) the authentication request is fresh (i.e. $xSQN_{HN} > SQN_{UE}$ and $xSQN_{HN} < SQN_{UE} + \Delta$), replies `Sync_failure`, $AUTS$ otherwise ($AUTS$ is explained next). The quantity Δ is a threshold that is fixed according to an availability vs. security trade-off. If all checks hold then the *UE* computes the ciphering key CK and the integrity key IK and stores them to secure subsequent messages. It also computes the authentication response RES and sends it to the *SN* using *Authentication Response* message. Only RES is included in the message, other computed values like CK and IK are not transmitted. The *SN* authenticates the *UE* by verifying whether the received response matches with $xRES$. If so, the AKA protocol is successfully completed and subsequent communications can be secured using the secret keys IK and CK.

Re-synchronization procedure. In case of a synchronization failure (case `(i) and` $\neg$`(ii)`), the *UE* replies with `Sync_failure`, $AUTS$. The $AUTS$ message's purpose is to allow the *HN* to re-synchronize with the *UE* by replacing its own SQN_{HN} by the sequence number of the *UE* (*i.e.,* $SQN_{UE} + 1$). However, SQN_{UE} is not transmitted in clear text to avoid being eavesdropped on. Thus, the specification requires *SQN* to be *concealed*; *i.e.,* XORed with a value, called *Anonymity Key*, that should remain private: $AK^* = \mathsf{f5}^*(RAND,\ K_{IMSI})$. Formally, the concealed value is as follows: $CONC^* = SQN_{UE} \oplus AK^*$ and allows the *HN* to extract SQN_{UE} by computing AK^*. Note that $\mathsf{f5}^*$ and $\mathsf{f1}^*$ are independent one-way keyed crypto-

graphic functions completely unrelated to functions $f1 - f5$. Finally, $AUTS = CONC^*, MAC^*$ where $MAC^* = f1^*(K_{IMSI}, (xSQN_{HN}, xRAND, AMF))$ allowing the *HN* to authenticate this message as coming from the intended *UE*.

2.2 Privacy Requirements

The 3G and 4G specifications consider user identity and location confidentiality, and user untraceability as explicit privacy requirements [6, 17]. Privacy was even more of a critical security goal in the design of 5G, as acknowledged by the standard:

[18]: *"Subscription privacy deals with various aspects related to the protection of subscribers' personal information, e.g., identifiers, location, data, etc. [...] The subscription privacy is very important area for Next Generation system as can be seen by the growing attention towards it, both inside and outside [3GPP]."*

We also emphasize that, *SQN* is considered to be privacy-sensitive, and must be protected (*i.e.,* remain confidential) by the AKA protocol:

[6]: *"[AK^*] is an anonymity key used to conceal the sequence number as the latter may expose the identity and location of the user."*

More recently, a 3GPP study on privacy explains that:

[19]: *"[AKA] is an example of how to fulfill anonymity: [...] Anonymizing technique used: use Anonymity Key in the Authentication Token to conceal (blind) the sequence number."*

As we shall see, our attack defeats the purpose of the anonymity AK^*.

2.2.1 Threat Model

While designing the AKA protocol in the year 2000, fake base stations were considered expensive in terms of required financial resources and attacker's capabilities. However, such fake base stations can now be easily built using *e.g.,* widely available hardware [20] or even WiFi technology [21]. Therefore, in our security analysis, we consider both passive and active attacker models for 3G, 4G, and 5G networks.

Passive. The passive adversary can sniff over-the-air radio broadcast channels using a dedicated hardware and software (as described in Section 5). Note that, he does not need to know any key material used in the authentication procedure.

Active. In addition, the active adversary has the capability to setup and operate a rogue base station to inject malicious traffic towards *UEs*. To achieve this, we assume he knows the protocol specification. However, we do not assume he knows any cryptographic keys.

2.3 Related Work on AKA and Known Flaws

There are numerous known attacks against the AKA protocol that were often described for older generation cellular networks but are still inherited in 4G networks due to the support of legacy system and re-use of the same core protocol from AKA.

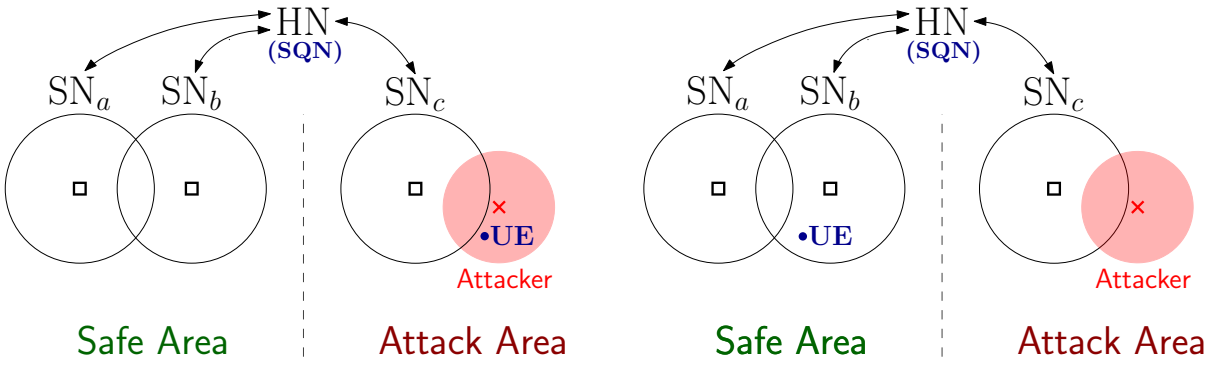
Identity Requests and IMSI-catchers. The first kind of attacks relies on the unprotected identity request mechanism that is broadcast over-the-air and is often termed as *IMSI-catchers*.

In a nutshell, an active attacker can easily broadcast an identity request to all the *UEs* in the area. Consequently, the *UEs* will reply with their permanent identities - this flaw is commonly exploited in IMSI catcher attacks [9–12] to track subscribers in certain geographical areas. Even though *UEs* may use temporary identities, a passive attacker can find co-relation between them and social identities [9] (including Facebook, Twitter, or phone number).

Several studies cover research on how to secure unprotected messages carrying identity requests by additional cryptographic mechanisms [7, 22, 23]. More importantly, in order to comply with the new stronger privacy protection requirement in 5G, 3GPP has modified the identity request phase of the AKA protocol. As mentioned earlier, the *UE* sends its permanent identity protected by a randomized, asymmetric encryption using the *HN*'s public key, in such a way that the *SNs* or fake base stations only learn the underlying *HN*. Therefore, the aforementioned IMSI catchers attacks will be defeated in 5G.

However, note that, even after fixing these vulnerable messages (including the fix in 5G phase 1 [2]), our attack re-introduces new subscriber privacy risks since it does not rely on this identification phase.

Linkability of failure messages. A few other attacks [7, 8] exploit the fact that the AKA protocol exposes to the attacker the reason of the failure when an authentication is rejected by a *UE*: either `Mac_Failure` or `Sync_Failure`. This allows to track a targeted *UE*: it suffices to replay an old authentication



(a) When the *UE* is in the attack area, it was known that it may be subject to tracking, location attack, and monitoring attacks.

(b) A *UE* was supposed to be safe when outside of attack areas. This is no longer the case due to our attack.

Fig. 2. Privacy threats depend on the *UE*'s location depicted with a dot. Attacker's fake base station (resp. genuine base station) is depicted with a cross (resp. box). Independenly of the *UE*'s location, the *UE*'s activities have an effect on *SQN* stored at the *HN*.

challenge that *UE* has already received and then observe whether the reply is `Mac_Failure` (not the targeted *UE*) or `Sync_Failure` (only replied by the targeted *UE*). Again, our attack does not rely on distinguishing between the two sources of failure and cannot be prevented by fixing this issue alone (*e.g.,* by merging the two sources of failure into a single message).

2.4 Comparisons Between Existing Attacks and our Attack

First, as explained in Section 2.3, our attack relies on a different logical vulnerability that is completely orthogonal to the attack vectors of prior, known attacks (*i.e.,* unprotected identities and linkability of failure messages). Therefore, our attack would require different and dedicated fixes.

Second, our attack poses a new kind of threat on privacy as it allows an attacker to learn subscribers' mobile services consumption patterns. In contrast, prior privacy attacks only leak the presence of targeted *UEs* in attack areas.

Third, our attack can break subscribers' privacy even when they escape attack areas. Indeed, remind that we consider a threat model for which attackers may exploit a limited number of fake base stations deployed at specific locations; typically busy crossing points (*e.g.,* subway or train stations, airports), targeted offices (*e.g.,* nearby embassies), or, places visited on a regular basis by targeted *UEs* (*e.g.,* shops). With prior attacks, there were two very different situations (depicted in Figure 2):

- if the targeted *UE* is outside the range of attacker's base stations (*e.g.,* at home), then the *UE* is completely safe: no fake base station-based attacks could break the subscriber's privacy (situation shown in Figure 2b);
- otherwise, the subscriber may be subject to location attacks or monitoring attacks (*i.e.,* attacker may eavesdrop on communication between *UE* and SN_c to learn when *UE* consumes services) (situation depicted in Figure 2a).

Therefore, even though the *UEs* may be attacked when in the range of attacker's fake base stations, as soon as they escape such (*a priori* narrow) areas, they were safe.

This is unfortunately no longer the case as our attack introduces a totally new threat on privacy. Indeed, even when *UEs* are using mobile services outside the attack area, part of this activity may be leaked to some adversary using our attack the next time the *UE* enters again the attack area. Intuitively, this is because, independently of its location, the *UE*'s activity has an effect on the counter SQN stored in the *HN* that will be leaked when the *UE* is (actively) under attack (we extensively discuss impacts on privacy in Section 4). We call this new kind of threat *activity monitoring attacks*. Hence, areas outside fake base stations range are no longer safe.

3 Logical Attack

In this section, we reveal the main attack vector based on a new logical vulnerability (Section 3.1) that can be exploited to mount an attack breaking the confidentiality of SQN (Section 3.2).

3.1 Logical Vulnerability

Our attack vector exploits a lack of randomness and the use of XOR in $AUTS$, more precisely in the concealed sequence number $CONC^* = SQN_{UE} \oplus AK^*$ where $AK^* = \mathsf{f5}^*(R,\ K_{IMSI})$. The value R is extracted from the challenge $R, AUTN$ received by the UE. Therefore, if the UE receives two times the same challenge $R, AUTN$ and yield two synchronization failures, then the two concealed $SQNs$ will be of the form: $CONC_1^* = SQN_{UE}^1 \oplus AK_1^*$ and $CONC_2^* = SQN_{UE}^2 \oplus AK_2^*$ such that

$$AK_1^* = \mathsf{f5}^*(R,\ K_{IMSI}) = AK_2^*.$$

Therefore, an attacker having a genuine challenge $R, AUTN$ for some UE can transmit it to the UE at two different times t_1 and t_2, retrieve values $CONC_1^*$ and $CONC_2^*$, and compute:

$$\begin{aligned} CONC_1^* \oplus CONC_2^* &= (SQN_{UE}^1 \oplus AK_1^*) \oplus \\ &\quad (SQN_{UE}^2 \oplus AK_2^*) \\ &= SQN_{UE}^1 \oplus SQN_{UE}^2 \end{aligned}$$

where SQN_{UE}^i is the value SQN_{UE} at time t_i. We show in the next section that by cleverly choosing several timestamps t_i's, the attacker is able to exploit values such as $SQN_{UE}^i \oplus SQN_{UE}^j$ to break the confidentiality of SQN.

3.2 Breaking the Confidentiality of SQN

We show how an active attacker who knows any UE's identity (temporary, permanent, or encrypted) is then able to learn the n least significant bits of SQN_{HN}, stored in the HN. The attacker first fetches $2^n + 2$ successive, fresh, authentication challenges intended for the targeted UE and replays a total of $2(n+2)$ of them to the UE. The interaction is depicted in Figure 3. The attack ends with an offline computation using $\texttt{algo}(\cdot)$ which takes fetched $AUTS$ messages as inputs and returns the n least significant bits of the sequence number SQN_{HN}.

In a nutshell, the attack consists in choosing appropriate injections and timestamps t_i such that the attacker can retrieve values $\delta_i = SQN_{HN} \oplus (SQN_{HN} + 2^i)$ for $1 \le i \le n$ (see Section 3.2.1). We then explain (Section 3.2.2) how one can infer from the δ_i's the n least significant bits of SQN_{HN}. Finally, we also show that, under certain circumstances (*i.e.,* when the UE is performing a lot of authentication sessions when in the attack area), a far less costly variant of the attack (only $n + 2$ injections) achieves the same goal (Section 3.2.3).

We describe the attack and our inference algorithm when the HN increments SQN_{HN} by 1 after each successful authentications as described in Section 2. Our attack works for any such increment; the interaction is always the same and we designed a generic $\texttt{algo}(\cdot)$ parametrized by the increment used by the operator. However, for the sake of clarity, we only describe here our attack and our generic algorithm for an increment equal to 1. We describe the full algorithm in Appendix B. Note that the full algorithm might actually infer more than n bits for some inputs; we report on practical results of this algorithm in Section 5.

3.2.1 Fetching Data

In a first phase (loop for $i = 0$ to 2^n from Figure 3), the attacker needs to fetch consecutive challenges $R_i, AUTN_i$ intended for the targeted UE. This is made possible by the fact that, in the AKA protocol, UE receives such challenges prior to authentication but after identification. Therefore, an attacker only needs to know one valid identity of the targeted UE (*e.g.,* IMSI, temporary identifiers such as TMSI, or encrypted permanent identities such as SUCI) in order to (partly) impersonate the UE to the SN (and the corresponding HN) and get those challenges. We will explain in Section 5 how this can be easily done in practice. Note that because SQN_{HN} is incremented by 1 after the computation of every challenge, $R_i, AUTN_i$ is computed based on some SQN value (that we denote by $SQN_{HN}(AUTN_i)$) equals to $SQN_{HN}^0 + i$.

Immediately after the first phase, the attacker injects the first challenge he obtained: $R_0, AUTN_0$. From the UE's perspective, this is a genuine challenge (the MAC verification `(i)` succeeds) that has never be received before and that is based on a recent enough $SQN_{HN}(AUTN_0) = SQN_{HN}^0$ (the freshness verification `(ii)` succeeds). At this time (before the second loop), SQN_{UE} equals $SQN_{HN}^0 + 1$. Then, the attacker injects again the challenge $R_0, AUTN_0$ yielding a synchronization failure containing some $AUTS' = \langle c', MAC^* \rangle$ message where the conceal SQN equals:

$$c' = (SQN_{HN}^0 + 1) \oplus \mathsf{f5}^*(R_0, K_{IMSI}).$$

In the last phase (loop for $j = 0$ to n from Figure 3), the attacker injects $R_{2^j}, AUTN_{2^j}$ that is accepted by the UE, in order to make the UE updates its SQN_{UE} to the value

$$SQN_{UE} := SQN_{HN}(AUTN_{2^j}) + 1 = SQN_{HN}^0 + 2^j + 1.$$

Data: $\delta_i = (2^i + X) \oplus X$ for $0 \le i \le n$ (in little-endian), $n < 48$
Result: Res: n least significant bits of X (in little-endian)
Res $\leftarrow [0,\ 0,\ \ldots,\ 0]$ `//size n`
for i **from** 0 **to** $n-1$ **do**
  `//Let's analyze` δ_i `at bit positions` $i,\ i+1$
  $(b_1,\ b_2) \leftarrow (\delta_i[i],\ \delta_i[i+1])$
  if $(b_1,\ b_2) == (1,\ 0)$ **then**
    `//no remainder propagate when` $+2^i$ `to` X
    Res$[i] \leftarrow 0$
  elif $(b_1,\ b_2) == (1,\ 1)$ **then**
    `//a remainder propagates when` $+2^i$ `to` X
    Res$[i] \leftarrow 1$
  else `//cannot happen`
    Error
end
return (Res)

Algorithm 1: *SQN* Inference Algorithm

After each such injection, the attacker then injects again the challenge $R_0, AUTN_0$ provoking a synchronization failure containing some $AUTS_j = \langle c_j^*, MAC_j^* \rangle$ where:

$$c_j = (SQN_{HN}^0 + 2^j + 1) \oplus \mathsf{f5}^*(R_0, K_{IMSI}).$$

3.2.2 Inference Algorithm

We now describe `algo`$(\cdot)$ that takes the $n+2$ fetched $AUTS$'s messages (*i.e.,* c', c_j for $0 \le j \le n$) as inputs and outputs the n least significant bits of $1 + SQN_{HN}^0$. Recall that $c' = (1 + SQN_{HN}^0) \oplus \mathsf{f5}^*(R_0, K_{IMSI})$ and $c_j = (1 + SQN_{HN}^0 + 2^j) \oplus \mathsf{f5}^*(R_0, K_{IMSI})$. Therefore, for any $0 \le j \le n$, it holds that:

$$c' \oplus c_j = \quad (1 + SQN_{HN}^0) \oplus (2^j + 1 + SQN_{HN}^0).$$

We note δ_i the quantity $c' \oplus c_i$. One has that $\delta_i = (2^i + X) \oplus X$ for all $0 \le i \le n$ where $X = 1 + SQN_{HN}^0$ is the quantity we seek to infer the n least significant bits of. In a nutshell, the idea of the algorithm consists in analyzing how remainders propagate in $(2^i + X)$ at bit position i and $i+1$ (in little-endian notation) by looking at δ_i. Considering X and δ_i as arrays of 48 bits in little-endian, we describe the algorithm in Algorithm 1 that, given the δ_i's, infers n bits of X. Note that this algorithm can be executed completely offline on the collected data.

3.2.3 Improving the Attack Under Stronger Threat Model

When the targeted *UE* stays a long time in the attack area or intensely consumes mobile services (triggering a lot of AKA authentication sessions), the attacker has a simpler way to break the confidentiality of SQN. This kind of scenarios are realistic when the attack areas are *e.g.,* offices where targeted *UEs* stay most of the day but expect to be safe when being outside attack areas (*e.g.,* at home).

Essentially, instead of fetching the challenges $R_i, AUTN_i$ and injecting the challenges that are accepted by the *UE* (*i.e.,* $R_0, AUTN_0$ and then $R_{2^j}, AUTN_{2^j}$ for $0 \le j \le n$), the attacker can let the *UE* attaches to any genuine *SN* and let it receives challenges and completes the AKA sessions. The attacker just passively eavesdrop on the exchanged messages, notably the challenges, and counts the number of successful authentications. However, the attacker still needs to (actively) replay the challenge $R_0, AUTN_0$ at appropriate times; more precisely, after the *UE* received the genuine challenge $(R_0, AUTN_0)$ and then challenges $(R_{2^j}, AUTN_{2^j})$ for $0 \le j \le n$. This variant is far less costly: it only requires passive attacking capabilities and $n+2$ additional (active) injections.

3.2.4 Variants for Other *SQN* Policies

According to non-normative parts of the specification [6]), *SQN* and its update policy can take different forms. We briefly explain how our attack can be easily adapted for those variants. We refer to Appendix C for more details.

SQN can be composed of two components $SQN = SEQ \| IND$ where *SEQ* is a 43 bits long integer that counts all past AKA sessions and *IND* is a 5 bits long index that describes the *SN* for which the given *SEQ* is valid. When such a policy is in use, one can use a slightly different variant of our attack: (i) injections of authentication challenges should be done while using the same *SN* identifier towards the *UE* and the same *SN* while fetching authentication tokens, and (ii) the algorithm used to infer bits should drop the 5 bits of *SQN* corresponding to *IND*. This allows the attacker to break the counter part of *SQN*, namely *SEQ*; leading to the same privacy attacks explained in the next section.

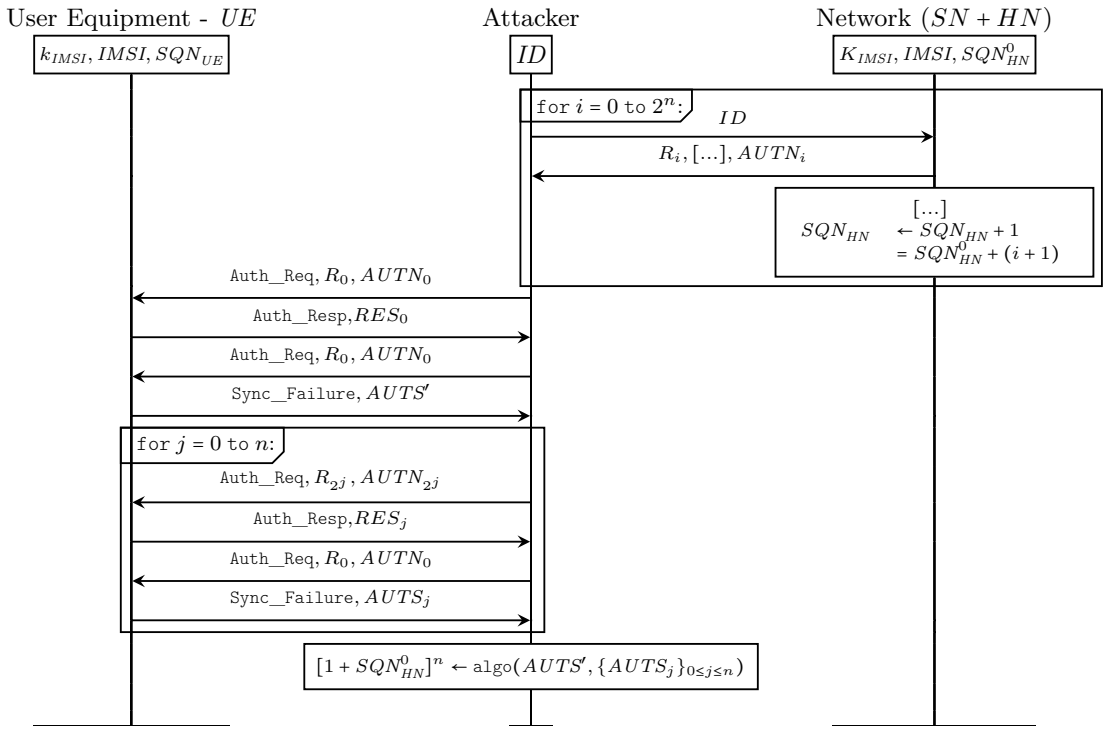


Fig. 3. Sequence Number Inference Attack (where SQN^0_{HN} is the initial *SQN* for *IMSI* stored in the *HN* and $[X]^n$ denotes the n least significant bits of X).

4 Attacks on Privacy

We now explain how one can leverage the attack presented in Section 3 to break subscribers' privacy by conducting an *activity monitoring attack*. As explained in Section 2.4, this new class of attacks allows an attacker to monitor targeted subscribers' activity, even for periods where subscribers *escape the attack area* (see Figure 2b).

In a nutshell, the attacker needs to conduct the previously described attack when targeted subscribers are in the attack area, thereby learning n significant bits of SQN at different times $t_1, t_2, \ldots$ We explain in Section 4.1 how the attacker can then relate this information to the number of AKA sessions subscribers have made between times $t_1, t_2, \ldots$. Next, we show how the attacker can relate the number of AKA sessions some *UE* has performed in a given period of time to its typical service consumption during that period.

Therefore, the attacker learns the typical service consumption of targeted subscribers between times $t_1, t_2, \ldots$ even if such subscribers escape the attack area most of the time (*i.e.,* in between times t_i). Please refer to Section 5 for practical aspects of the attack (*e.g.,* number of bits of SQN that can be inferred). We illustrate this new threat by giving some illustrations of practical attack scenarios (Section 4.2). We conclude in Section 4.3 with a variant of our attack that could yield location attacks in a variant of AKA that fix previously known privacy attacks which is notably relevant in the context of the in-progress standardization of 5G AKA, phase 2.

4.1 Relating SQN Increases to Activity Patterns

We first need to learn the value that is added to SQN after each successful authentication. The conclusion of our practical investigations (see Section 5) is that this value is 1 for all tested operators. This value is needed because equal differences of SQN could be resulted by different operations: if the victim SQN had been increased by 20, it could be the result of either 4 increases of 5 (4 authentications) or 2 increases of 10 (2 authentications). We found how much SQN is increasing upon authentication for several operators by running the algorithm algo() for several values of the increment and keeping the value yielding no Error (see Algorithm 1). We stress that this has to be done just once for or a given operator.

Next, in order to relate information about the number of AKA sessions of a victim with the victim's activity, we have to exploit the fixed authentication policies discussed in Section 5.2 (*i.e.,* which user's activities trigger an authentication and thus an AKA session). Because of the different operator configurations, authentication may or may not happen on each *SN* network attach, call or reception/sending of SMSs. As a result, we also analyzed how frequently authentication is performed by analyzing signaling messages during repeated attach procedure (by calling or sending SMSs). We found that there are little variations in authentication frequency among operators but for most of them, an authentication was required for each outgoing call and sent SMS). Despite those variations, one can easily infer the fixed policy for some operator, once for all, by inspecting signaling messages *e.g.,* on her own phone.

We leave as future work the task of doing a comprehensive review of existing policies.

4.2 Examples of Practical Scenarios

We now illustrate the potential real-life impacts of our activity monitoring attack with two practical scenarios.
Spying on embassy officials or journalists. Assuming an adversary having a fake base station nearby an embassy, he not only can learn the officials' activity when they are at the office during working hours, but also when they are not, including during evening and nights (*e.g.,* at home) or during business trips. Therefore, such an attacker may learn if targets use different SIMs cards for private use (no activity at home). It may also infer if some specific time periods (*e.g.,* one evening and night) were specifically busy (a lot of calls or SMSs were made yielding a big rise of SQN).
Better ads targeting. Consider for instance a shop that is willing to know more about its customers (*e.g.,* for improving ads targeting) using fake base stations. This kind of scenario has already been reported [24] (using Wi-Fi capabilities of smartphones) and exploited [25] in real shops. Our attack causes a new threat in that context since it leaks to the shop typical customers' mobile consumption during time periods *between* customers' visit (while they escape the attack area).

4.3 Deriving Location Attacks

Using variants of our attack, one could mount location attacks (*i.e.,* inferring if some targeted *UE* is in some physical area) even if the *leak of identity* (currently enabling IMSI-catchers attacks) and the *traceability based of failure messages* were fixed.

More precisely, we first assume that the identity request phase would be well-protected using *e.g.,* encryption (as done in 5G, phase 1 [2]). Second, we assume that the two failure cases (MAC or freshness failure) would be merged (AUTS message is also sent out in case of MAC failure, the network being able to infer the reason of the failure) to address the latter known flaw. Under those assumptions, to the best of our knowledge, there is no known attack that could break subscribers' privacy. However, either of the two following variants of our attack still allows an active adversary to perform location attacks[1].

First, if an attacker knows a value $CONC_0$ of some targeted UE_0 and obtains a value $CONC$ from some unknown UE (this can be easily obtained by replaying a genuine challenge), then he can infer if the unknown UE is UE_0 with very high probability by inspecting how large is $CONC_0 \oplus CONC$, interpreted as an integer. Indeed, when both *UEs* do not match then $CONC_0 \oplus CONC = (SQN_{UE_0} \oplus AK_0^*) \oplus (SQN'_{UE_?} \oplus AK_?^*)$ (where $AK_?^* \neq AK_0^*$; see Section 3.1) which is a 48-bits random-looking value. By contrast, when they do match, then $CONC_0 \oplus CONC = SQN_{UE_0} \oplus SQN'_{UE_0}$ which is very likely a small value (we never observed more than 10 bits-values).

Second, by learning sufficiently many bits of some targeted *UE*, an active attacker will be able to track this *UE* with reasonable probability by keeping track of the SQN values he may repeatedly learn (recall that SQNs are 48-bits long so they almost injectively identify *UEs* even taking into account the fact that they evolve). Obviously, the practicality of such an attack heavily depends on the number of bits one can infer, the frequency at which the target visits the attack area and the speed at which target's SQN rises.

We consider those location attacks as potential threats for the upcoming 5G, phase 2 that may address previous flaws but not necessarily this new attack.

1 Note that our activity monitoring attack can also be exploited under those circumstances.

5 Proof of Concept of the Attack and Practical Considerations

In this section, we show how to conduct our attacks in practice on 4G networks using a low-cost and easily available setup. We then explain practical aspects which make our attack easily feasible (*e.g.,* issues in different operator's network and security configurations). We finally discuss our PoC and our experimental results.

Feasibility in 5G. Due to unavailability of 5G devices and networks, we only demonstrate our attack in a 4G environment. As already mentioned, we know that the 5G, phase 1 specification already suffers from our attacks. Moreover, we believe that it will be feasible to demonstrate our attack against real 5G networks soon due to the fast open source developments for 5G [26]. For example, 2G has been launched in 1991 but the first open source software were only made available in 2010 (with OpenBTS [27]). In contrast, 4G has been launched in 2009 [28] and open source 4G software was already supported the same year by OpenAirInterface [29].

5.1 Experimental Setup

The first experimental setup aims at building a platform to collect victim's authentication challenges. Further, we modify and build software tools to make victim's UE to attach to a rogue 4G base station so that one can inject legitimate radio layer signaling messages.

Our hardware setup is depicted in Figure 4 which consists of a laptop, a Universal Software Radio Peripheral (USRP) B210 [20], and a PC/SC [30] capable smartcard reader (we used ACS ACR38 [31]) with commercial USIM cards.

Our complete experimental setup costs about 1140€ excluding a laptop price (that could be replaced by a cheaper Raspberry pi) - 1120€ for USRP, 20€ for commercial operator's prepaid USIM cards.

5.1.1 Attacker's Setups

We now explain how hardware and software components can be combined into different setups to demonstrate our attack.

Obtaining authentication challenges. We used the software *srsUE* from the *srsLTE* suite [32] configured with the target's IMSI with the USRP B210 for obtaining authentication challenges. Essentially, the USRP B210 tries to impersonate the target's USIM. When doing so, each session fails because *srsUE* does not know the target's secret key K (so it cannot compute the appropriate RES) but, before the failure, we obtain a new, genuine authentication challenge that is intended for the target's USIM. We were able to fetch authentication tokens using the USRP at a surprising high speed (see discussion later) but, if for some reason, a network recognizes the USRP as a fake smartphone, we can still use genuine phones with programmable [33] USIM cards (ca. 80€). We describe this alternative setup in Appendix D.

Fig. 4. Our experimental setup, showing a smartcard reader, USRP (left), set of commercial USIM cards, and a test phone.

Fetching AUTS messages using rogue base station. Utilizing OpenLTE [34] based 4G network running on an USRP B210, we sniff over-the-air signaling messages and masquerade as a real base station. We configure a rogue base station to mimic a real operator to lure victim's UE to attach by 4G reselection procedure [9]. The base station is then able to inject messages and eavesdrop on replied messages. We use this method to fetch AUTS messages that a USIM sends as part of the AKA protocol.

5.1.2 Ethical Considerations

Our research reveals weaknesses in the AKA protocol specification which is implemented in every USIM installed in 3G and 4G devices worldwide. We reported our findings to relevant standardization bodies 3GPP [35] and GSMA, and affected network operators. Our results were acknowledged by the involved parties.

We conducted passive attacks only against test USIM cards and smartphones. We operated our rogue 4G base station inside a Faraday cage [36], to comply with legal requirements.

5.2 Attack Background

Before explaining our PoC, we report on our investigation on AKA related security configurations of 4G networks which make our attack easier to perform. We selected several major European 4G operators including three German, three Austrian, two French, and one Swiss operators.

We were successfully able to collect authentication challenges intended for the targeted USIM at any moment for any subscriber in the world. Note that to achieve this step, attacker only needs to know the IMSI (or any temporary or encrypted identity) of that particular victim's USIM. If the attacker knows the subscriber's mobile phone number, he can perform HLR Lookup attacks [37] to learn victim's IMSI. Previous work [9] also demonstrates how to find IMSI and GUTI of the targeted victim by knowing mobile phone number or social identities such as email, Facebook and Twitter. Based on the data collected from our experiments, we studied the following parameters of the operator's 4G networks. We stress that there is no need to learn more information (*e.g.,* private key K_{IMSI}) about the targeted USIM).

We found that most operators allowed to fetch authentication challenges without a rate limit. Using our first setup using *srsUE*, we were able to fetch fresh, unused authentication challenges consecutively at the speed of 1 per second. Using our second setup involving a smartphone, we were able to fetch more than 30 challenges in less than 10 minutes. We expect a setup based on multiple rogue base stations to achieve much better performance.

Background in 5G. Since no 5G deployment has been completed yet, we could not conduct a full PoC in 5G. We emphasize however that, if no dedicated mitigation is implemented, the different steps of our attack could be performed in 5G as well. The only major difference concerns the way we would fetch authentication challenges. To do so, we do not have to learn *SUPI* (the subscribers' identifier in 5G that is supposed to remain secret) but only a valid *SUCI*. Since *SUCI* is sent in the clear by subscribers, the attacker just has to eavesdrop on one *SUCI* and can from then on, fetch as many authentication tokens for that subscriber he wants (using *SUCI* instead of *IMSI*). The rest of the attack remains the same for 5G infrastructure.

5.3 Proof of Concept

Upon knowing the victim's identity and location [9], an attacker can perform our subscriber activity monitoring attack. This attack requires obtaining a larger number of authentication challenges of victim's USIM. But as discussed in Section 5.2, we did not observe any countermeasure preventing us to fetch a large amount of them.

Exploiting our attack, the more consecutive challenges one fetches, the more is the number of bits he can infer from the SQN of the victim. Then, using a rogue base station, the attacker is then able to inject parts of those challenges and store replied $AUTS$ as explained in Section 5.2.

We build a modest setup as a proof of concept of our attack. We were able to request 1025 authentication challenges and collected 12 $AUTS$ from 24 injections of AKA messages. Using our generic SQN inference algorithm, those 12 $AUTS$ messages were enough to infer at least 10 bits of SQN (the least significant ones), sometimes more[2]. Obviously, an attacker with greater capabilities and more elaborate setups (notably multiple rogue base stations for fetching challenges; which turns out to be the bottleneck) could infer more bits.

5.4 Attacks Feasibility and Amplifications

We now describe the feasibility of our subscriber activity monitoring attacks against commercially deployed 4G devices. Further, we discuss possibilities of extending coverage range of the USRP device.

Impacted devices. The AKA protocol vulnerability we found is part of the 3GPP specifications and does not rely on implementation issues in 4G/3G devices. In fact, the affected AKA protocol is implemented in the USIM and not in the baseband OS of devices. Thus, any 3G/4G device deployed worldwide having active USIM card is affected by our attacks. For our investigations, we selected prepaid USIM cards of few leading cellular operators. We collected and stored unused authentication challenges of related USIM cards as described before. Then we successfully verified that these USIM cards were vulnerable to our attack. As mentioned earlier about feasibility in 5G networks, if no dedicated mechanism for mitigating our attack is implemented, 5G devices will also suffer from our attack.

2 Our generic algorithm (see Appendix B) can be more efficient.

Range of the attack. Previous research suggested that the coverage radius of a rogue base station using USRP B210 and OpenLTE ranges between 50 and 100 meters [9], without an external hardware to boost the signal. The coverage range of our attacks could be increased to locate and inject radio layer messages to a 4G device equipped with USIM within a 2 km^2 area as discussed in [9].

Detection. We also investigated possible methods for the end subscribers to detect our attack. Unfortunately our attacks cannot be detected by the mobile OS (*e.g.,* Android or iOS). The reason is that the AKA protocol is executed in USIM and the baseband chip that communicates limited information to the mobile OS.

6 Security Analysis and Lessons Learned

In this section, we discuss the AKA specification issues and their impact on 3G/4G/5G security principles. First we describe AKA protocol design choices and discuss the trade-off considerations related to security, availability, and cost which partly are responsible for our attacks. The following analysis is based on the data and practical experiments in 3G/4G networks. Finally, we draw conclusions and summarize lessons learned that may be relevant for the in-progress standardization of 5G, phase 2.

Choice of symmetric key encryption. We demonstrated (Section 5) how a low cost setup allows an attacker to fetch unused challenges (*RAND*, *AUTN*) of any active 4G subscriber in the world from any network. We now explain the AKA protocol design choice of authentication method and trade-offs responsible for enabling access to *RAND* and *AUTN*). The AKA is a challenge-response type of protocol and utilizes a symmetric encryption based authentication mechanism. We believe that the reason of choosing a symmetric encryption stems from three trade-offs.

The first is a **trade-off between security and cost**, *i.e.,* High cost of introducing a Public Key Infrastructure into the 3G/4G systems and an asymmetric encryption mechanism in USIM, paves the way for choosing a symmetric encryption based authentication technique. Due to this high cost, the 3GPP designers were limited in previous 3G and 4G networks, however, PKI is introduced in 5G, only for protecting identities [2]. Note that, authentication in 5G, excluding identification, is still based on a symmetric cryptography.

The second is a **trade-off between security and network availability -** *i.e.,* Use of symmetric key avoids a risk of shutting down legitimate subscribers during a case of network fail or crash [38]. For example, if the *SN* (in particular MME) software crashes, temporary identity of a subscriber can not be recognized. In such a case, the network needs to request the permanent identity from the subscriber.

The third is a **trade-off between privacy and network efficiency -** The AKA is a *one round-trip* authentication protocol; *i.e.,* only two exchanged messages are needed to establish mutual authentication, after identification. The chosen mechanism to achieve mutual authentication with only two exchanged messages is the synchronized *SQN*. Allowing the *UEs* to generate a random number could have enabled different authentication methods. However in year 2000 (when 3G AKA was designed), *UE*'s computational resources were limited. With three exchanged messages, the protocol would not need this synchronized state and this additional message exchange could have enhanced privacy. However, this additional message exchange would also negatively impact the network efficiency notably because it would always require a message exchange between the *HN* and the *SN* as well [38].

The above trade-offs force the network to send *RAND* and *AUTN* to perform a round of challenge-response for the authentication of subscriber's temporary or permanent identities. This allows an attacker to impersonate subscriber's identity to fetch unlimited *RAND* and *AUTN* challenges from any network. One reason why an attacker can fetch those challenges of any subscriber from any network is the trust between the *SNs* and the *HNs*. Indeed, in 3G and 4G architectures, the *HN* and *SN* trust each other due to roaming agreements. Further, such illegitimate requests are difficult to filter out from the legitimate ones due to the risk of shutting down real subscribers from accessing the network. One potential solution is to rate limit (based on time or numbers) authentication requests per subscriber, however attacker could learn such kind of rate limit by simply testing the network. We found that one of the operator is implementing the rate limit of 3 consecutive failures. Moreover, an attacker could bypass this countermeasure by requesting authentications challenges from different *SNs*.

Replay protection measures. The AKA protocol uses *SQN* as a challenge to prevent replay attacks and synchronized state between the *UE* and the *HN*. This *SQN* challenge and the synchronization method pre-

vent from replay attacks. The *UE* verifies if a challenge ($AUTN$) is fresh or replayed. In case of network failure, legitimate authentication challenges may get lost resulting to the *HN* and subscriber being de-synchronized. In order to re-synchronize, the protocol uses the $AUTS$ message which contains the current SQN_{UE} of USIM XORed with the anonymity key (AK^*). However, this message $SQN_{UE} \oplus AK^*$ contained in $AUTS$ lacks randomness due to the fact that the key AK^* derives from the same $RAND$ value as discussed in Section 3.1. Our attack from Section 3.1 indicates a lack of replay protection for $AUTS$, like the one existing for $AUTN$.

Lessons for 5G. 5G phase 1 security has been released by the 3GPP including an enhanced AKA protocol featuring *HN*'s public keys to provide subscriber identifiers privacy [39]. However, our attacks reveal another threat to subscriber's privacy. In the future, clever and sophisticated attackers may find new ways to use every obtainable information to carry out further AKA protocol related attacks in 5G networks. Hence, it is important to protect sequence numbers used in authentication procedure messages.

Though first phase of 5G security is completed, we suggest that all security protocols in 5G shall go through formal verification before releasing phase 2 (some first steps have been taken in [40] and in this paper). More generally, authors of [41] provide some industrial case studies (WiMAX, EAP, and ISO/IEC 9798) and discuss how formal methods and associated security tools could be integrated into the standardization process.

7 Countermeasures

As already explained, the main attack vector we exploit in our attacks is the use of XOR and the lack of randomness in $AUTS$ making the concealing of SQN by AK^* inefficient. We propose three main countermeasures F1, F2, and F3 to solve this problem. We also discuss how our fixes also provide an opportunity to fix the known linkability attack based on failure messages discussed in Section 2.3. We conclude by a formal analysis of our fixes using the state-of-the-art Tamarin automated prover.

Note that, when discussing our countermeasures, we also consider practical aspects related to the existing and next generation cellular networks. Thus, the following fix we propose is easy to deploy in the current cellular system and only requires changes in baseband and authentication server software in the HSS. Other two fixes F2 and F3 in appendices E.1 and E.2 are suitable for next generation networks since they require additional modifications in deployed hardware (*e.g.,* USIM).

7.1 Symmetrically Encrypt SQN_{UE}

Our simplest fix consists in modifying the concealing mechanism: instead of using XOR (having algebraic relations enabling to cancel out AK^*), USIM may use symmetric encryption. Note that current USIMs and the HSS (in particular AuC) are already capable of symmetric encryption. The symmetric key to encrypt SQN_{UE} could be derived from the key K_{IMSI} and $RAND$ in the received authentication challenge. The resulting fix is depicted in Figure 5. This can be very easily adapted to fix the linkability of failure messages (see Section 2.3) as well. It suffices to hide the failure reason inside the ciphertext $CONC^*$ as follows: $CONC^* \leftarrow \mathsf{enc}(\langle \texttt{Reason\_Failure},\ SQN_{UE} \rangle,\ CK^*)$.

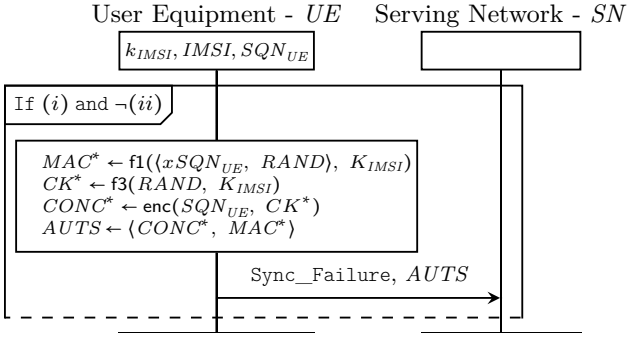


Fig. 5. Fix F1: Fix by symmetrically encrypting SQN_{UE}

The *HN* is required to decipher the $CONC^*$ in order to learn the reason of the *UE*'s authentication failure. However, this mechanism could add extra processing load on the *HN* due to the decryption requirement. Alternatively such processing load of the *HN* could be offloaded to the *SN* by transmitting decryption key in a set of authentication vectors. Finally, note that this solution suffers from a minor flaw: when the attacker triggers two times a synchronization failure by injecting the same authentication challenge while SQN_{UE} has not changed, then the two replied $CONC^*$ will be equal, leaking to the attacker the information that SQN_{UE} is still the same (we consider such an attack impractical though; more details in Appendix E.1).

7.2 Formal Verification

We have presented a new attack breaking confidentiality of SQN, its impact on subscribers' privacy and possible fixes to address this issue. A natural goal is then to evaluate those fixes. While all our fixes intuitively remove the attack vector on which our attack is based, we believe that an informal argument or even a pen and paper proof would not provide enough confidence considering the complexity and size of the AKA protocol. We actually advocate for the use of *formal methods* providing rigorous, mathematical frameworks and techniques to analyze security protocols.

Related Work. Such techniques have already been leveraged in the past (notably by 3GPP) to formally verify the AKA protocol to some extent (*e.g.,* formal analysis in enhanced BAN Logic and TLA [42], in the tool ProVerif [7, 21]). More recently, an in-depth formal analysis of 5G AKA [40] has been conducted using the tool Tamarin [16]. However, all those analyses failed to capture our attack[3] because those prior modelings abstracted the protocol too much. For instance, except for [40], they do not model the re-synchronization procedure at all, which is at the core of our attack; while [40] focuses on authentication properties and only models a specific scenario for privacy in order to capture a location privacy attack.

Challenges. We now focus on formal verification in the *symbolic model* which provides a high-level of automation. The limitations of prior analyses can be explained by the fact that the AKA protocol and its features cause several difficulties to the state-of-the-art tools and methods (such as ProVerif [43] and Tamarin [16]): (i) the modeling of the $\oplus$ operator that most tools cannot handle at all, (ii) the presence of a (non-monotonous state) state (*i.e., SQN* whose value must be stored from one session to the other), and (iii) basic arithmetic (*i.e., SQN* is basically an integer and integer additions and comparisons are carried out by the USIM). Each one of those features constitutes a major challenge to existing techniques. Finally, location privacy or untraceability, as defined *e.g.,* in [44], is not a reachability property but an observational equivalence-based property that is notoriously more difficult to verify (see the survey [45]). For an unbounded number of sessions, the only tools that can verify some sort of observational equivalence are ProVerif and Tamarin. The approximations those tools adopt (due to the fact that the underlying problem is undecidable) make the verification less precise and often lead to false attacks when verifying untraceability; despite recent research efforts [46, 47].

Our Formal Analysis. We took a first step towards a precise formal analysis of privacy for the AKA protocol in the symbolic model. We leveraged the state-of-the-art tool Tamarin [16] and built upon [40] in order to provide a symbolic model of the AKA protocol that is precise enough to capture our attack. Our models are available at [48]. We took the re-synchronization procedure into account and used the new feature [49] to precisely model the $\oplus$ operator. Note also that Tamarin is capable of modeling stateful protocols, so we were able to precisely model *SQN*. We modeled the AKA protocol without the fix as well as with the three fixes. We were not able to faithfully analyze the confidentiality of the *SQN* value as this would require to model algebraic relations of $\oplus$ on integers; this is one of the reason explaining that [40] missed our attack.

However, for two sessions, we were able to analyze the confidentiality of $\delta_0 = SQN \oplus (SQN + 1)$ which is one of the values needed to bootstrap our attack. In our model, the confidentiality of δ_0 was automatically proven for all our fixes and our attack breaking this property was automatically found without our fix. We consider this analysis as a first step towards a more precise model amenable to automatic verification of privacy-related properties which has proven itself extremely complex to produce. We leave that task as future work.

In summary, the state of the art of mechanized formal verification does not provide off-the-shelf technique to verify privacy-related properties on the AKA protocol with sufficient precision. Considering the importance and ubiquity of the AKA protocol, we think that formally verifying it and solve the aforementioned challenges in order to provide a precise symbolic model of the AKA protocol and its variants is a substantial but major goal.

8 Conclusion

We disclose a subtle vulnerability in the AKA protocol affecting the 3G, 4G, and upcoming 5G technologies. We demonstrate how this vulnerability can be exploited to mount activity monitoring attacks, allowing the attackers to learn a new type of privacy-sensitive information

3 While they did not focus on the confidentiality of *SQN*, they still could have captured the traceability variants explained in Section 4.3.

about the subscribers; *i.e.,* consumption patterns. As a proof-of-concept, we show how an active attacker equipped with a low cost and widely available setup can perform our attack in several European 4G networks, learning SQN with granularity 2^{10}. We leave a comprehensive evaluation of the privacy impact of our attack as future work. We then analyze root causes of the vulnerability and their impact on 3G/4G/5G security principles to derive lessons for future 3GPP standardization; notably 5G, phase 2. Finally, we provide countermeasures and formal guarantees that also motivate further research into improving and formally analyzing the AKA protocol.

References

[1] GSMA, "Definitive data and analysis for the mobile industry," https://www.gsmaintelligence.com/.

[2] 3GPP, "Security architecture and procedures for 5G System," (3GPP), TS 33.501. [Online]. Available: http://www.3gpp.org/DynaReport/33501.htm

[3] J. Arkko, "Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)," RFC 4187. [Online]. Available: https://rfc-editor.org/rfc/rfc4187.txt

[4] Peter Howard, "AKA usage in 3GPP." [Online]. Available: http://www.3gpp.org/ftp/tsg_sa/wg3_security/tsgs3_34_acapulco/docs/PDF/S3-040645.pdf

[5] 3GPP, "Service requirements for the Evolved Packet System (EPS)," (3GPP), TS 122.278. [Online]. Available: http://www.3gpp.org/DynaReport/22278-CRs.htm

[6] ——, "3G security; Security architecture," (3GPP), TS 33.102. [Online]. Available: http://www.3gpp.org/DynaReport/33102.htm

[7] M. Arapinis, L. Mancini, E. Ritter, M. Ryan, N. Golde, K. Redon, and R. Borgaonkar, "New privacy issues in mobile telephony: fix and verification," in Proceedings of the 2012 ACM conference on Computer and communications security. ACM, 2012, pp. 205–216.

[8] C. Hahn, H. Kwon, D. Kim, K. Kang, and J. Hur, "A Privacy Threat in 4th Generation Mobile Telephony and Its Countermeasure," The 9th International Conference on Wireless Algorithms, Systems, and Applications, pp. 624–635, 2014.

[9] A. Shaik, J. Seifert, R. Borgaonkar, N. Asokan, and V. Niemi, "Practical Attacks Against Privacy and Availability in 4G/LTE Mobile Communication Systems," in 23nd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016.

[10] S. Udar and R. Borgaonkar, "Understanding IMSI Privacy," https://www.isti.tu-berlin.de/fileadmin/fg214/ravi/Darshak-bh14.pdf.

[11] F. van den Broek, R. Verdult, and J. de Ruiter, "Defeating IMSI Catchers," Proceedings of the 2015 ACM Conference on Computer and Communications Security - CCS '15, 2015.

[12] M. S. A. Khan and C. J. Mitchell, "Another look at privacy threats in 3G mobile telephony," in Australasian Conference on Information Security and Privacy. Springer, 2014, pp. 386–396.

[13] M. Zhang and Y. Fang, "Security analysis and enhancements of 3GPP authentication and key agreement protocol," IEEE Transactions on Wireless Communications, vol. 4, no. 2, pp. 734–742, March 2005.

[14] M. Khan, A. Ahmed, and A. R. Cheema, "Vulnerabilities of UMTS Access Domain Security Architecture," in Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008. SNPD '08. Ninth ACIS International Conference on, Aug 2008, pp. 350–355.

[15] A. N. Bikos and N. Sklavos, "LTE/SAE Security Issues on 4G Wireless Networks," IEEE Security Privacy, vol. 11, no. 2, pp. 55–62, March 2013.

[16] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The Tamarin Prover for the Symbolic Analysis of Security Protocols," in Proc. 25th International Conference on Computer Aided Verification (CAV'13), ser. LNCS, vol. 8044. Springer, 2013, pp. 696–701.

[17] 3GPP, "3GPP System Architecture Evolution (SAE); Security architecture," (3GPP), TR 33.401. [Online]. Available: http://www.3gpp.org/DynaReport/33401.htm

[18] 3GPP, "Study on the security aspects of the next generation system," (3GPP), TR 33.899. [Online]. Available: http://www.3gpp.org/DynaReport/33899.htm

[19] ——, "Study on subscriber privacy impact in 3GPP," (3GPP), TR 33.849.

[20] Ettus Research, "USRP B210." [Online]. Available: https://www.ettus.com/product/details/UB210-KIT

[21] P. O'Hanlon, R. Borgaonkar, and L. Hirschi, "Mobile subscriber WiFi privacy," in Proceedings of Mobile Security Technologies (MoST'17), held as part of the IEEE Computer Society Security and Privacy Workshops (SPW'17), 2017, to appear.

[22] F. Y. Leu, I. You, Y. L. Huang, K. Yim, and C. R. Dai, "Improving security level of LTE authentication and key agreement procedure," in 2012 IEEE Globecom Workshops, Dec 2012, pp. 1032–1036.

[23] X. Li and Y. Wang, "Security Enhanced Authentication and Key Agreement Protocol for LTE/SAE Network," in Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on, Sept 2011, pp. 1–4.

[24] A. Musa and J. Eriksson, "Tracking unmodified smartphones using Wi-Fi monitors," in Proceedings of the 10th ACM conference on embedded network sensor systems. ACM, 2012, pp. 281–294.

[25] S. Clifford and Q. Hardy, "Attention Shoppers: Store is Tracking Your Cell," New York Times, vol. 14, 2013.

[26] Open5GCore, "Open5GCore – The Next Mobile Core Network Testbed Platform." [Online]. Available: https://www.open5gcore.org/

[27] Venturebeat, "DEMO: Range Networks rings in cell-phone service for USD 2 a month." [Online]. Available: https://venturebeat.com/2010/09/14/demo-range-networks-cheap-cell-phone-service/

[28] Telegraphy, "TeliaSonera launches world's first commercial LTE networks in Sweden and Norway." [On-

line]. Available: https://www.telegeography.com/products/commsupdate/articles/2009/12/14/teliasonera-launches-worlds-first-commercial-lte-networks-in-sweden-and-norway/
[29] OpenAirInterface, "History." [Online]. Available: http://www.openairinterface.org/?page_id=753
[30] PC/SC Workgroup, "PC/SC Workgroup Specifications." [Online]. Available: http://www.pcscworkgroup.com/specifications/specdownload.php
[31] Advanced Card Systems Holdings Limited, "ACR38 Smart Card Reader." [Online]. Available: http://www.acs.com.hk/en/products/4/acr38-smart-card-reader/
[32] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, "srsLTE: An Open-Source Platform for LTE Evolution and Experimentation," CoRR, vol. abs/1602.04629, 2016. [Online]. Available: http://arxiv.org/abs/1602.04629
[33] Sysmocom, "sysmoUSIM-SJS1." [Online]. Available: https://www.sysmocom.de/products/sysmousim-sjs1-sim-usim/index.html
[34] B. Wojtowicz, "OpenLTE," https://sourceforge.net/projects/openlte/.
[35] 3GPP, "SA3 - Security." [Online]. Available: http://www.3gpp.org/specifications-groups/sa-plenary/sa3-security
[36] Gamry Instrumens, "Faraday Cage: What Is It? How Does It Work?" [Online]. Available: http://www.gamry.com/application-notes/instrumentation/faraday-cage/
[37] T. Engel, "Locating Mobile Phones using Signalling System 7," https://berlin.ccc.de/~tobias/25c3-locating-mobile-phones.pdf.
[38] D. Forsberg, G. Horn, W.-D. Moeller, and V. Niemi, LTE Security, 2nd ed. Wiley Publishing, 2012.
[39] Anand R. Prasad, Alf Zugenmaier, Adrian Escott and Mirko Cano Soveri, "3GPP 5G Security." [Online]. Available: http://www.3gpp.org/news-events/3gpp-news/1975-sec_5g
[40] D. Basin, J. Dreier, L. Hirschi, S. Radomirovic, R. Sasse, and V. Stettler, "A Formal Analysis of 5G Authentication," in Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2018, pp. 1383–1396.
[41] D. Basin, C. Cremers, K. Miyazaki, S. Radomirovic, and D. Watanabe, "Improving the Security of Cryptographic Protocol Standards," IEEE Security Privacy, vol. 13, no. 3, pp. 24–31, May 2015.
[42] 3GPP, "3G Security; Formal Analysis of the 3G Authentication Protocol," (3GPP), TS 23.272.
[43] B. Blanchet, "An Efficient Cryptographic Protocol Verifier Based on Prolog Rules," in Proceedings of CSFW'01. IEEE Comp. Soc. Press, 2001, pp. 82–96.
[44] M. Arapinis, T. Chothia, E. Ritter, and M. Ryan, "Analysing unlinkability and anonymity using the applied pi calculus," in 2010 23rd IEEE Computer Security Foundations Symposium. IEEE, 2010, pp. 107–121.
[45] S. Delaune and L. Hirschi, "A survey of symbolic methods for establishing equivalence-based properties in cryptographic protocols," ArXiv e-prints, Oct. 2016.
[46] V. Cheval and B. Blanchet, "Proving more observational equivalences with ProVerif," in Principles of Security and Trust. Springer, 2013, pp. 226–246.
[47] L. Hirschi, D. Baelde, and S. Delaune, "A method for verifying privacy-type properties: the unbounded case," in Proceedings of the 37th IEEE Symposium on Security and Privacy (S&P'16), M. Locasto, V. Shmatikov, and U. Erlingsson, Eds.
[48] "Tamarin tool code." [Online]. Available: https://goo.gl/WQ6bzH
[49] J. Dreier, L. Hirschi, S. Radomirovic, and R. Sasse, "Automated Unbounded Verification of Stateful Cryptographic Protocols with Exclusive OR," in 31st IEEE Computer Security Foundations Symposium (CSF'2018), 2018.
[50] Osmocom Project, "pySIM: A python tool to program magic SIMs," http://cgit.osmocom.org/pysim/.
[51] SecT- TU Berlin, "SCAT: Signaling Collection and Analysis Tool." [Online]. Available: https://github.com/fgsect/scat
[52] 3GPP, "AT command set for User Equipment (UE)," (3GPP), TS 27.007. [Online]. Available: http://www.3gpp.org/DynaReport/27007.htm

A Notatations & Acronyms

We show in Table 1 an informal correspondence between our informal security artchitecture terminology and proper terninologies in 3G, 4G, and 5G networks.

B Generic *SQN* Inference Algorithm

This Section is dedicatd to the description of our generic SQN inference algorithm parametrized by the increment γ that is used for updating SQN_{HN} after each successful AKA session. In a nuthsell, it is an extension of the algorithm described in Section 3.2.2 that works even when $\gamma > 1$. In particular, it is based on the same idea that consists in learning whether a remainder propagated or not at specific position.

Informal description of the algorithm. We now give an informal presentation of this algorithm. While we have not reached a fixed points, we do the following.

- For all δ_i from the sequence of $\{\delta_j\}_j$ given as input (let us call i its position in the sequence), we do the following.
  - For all bit position p from 0 (least significant bit position) to 47 (the most significant bit position), we do the following.
    * We compute $\Gamma = \gamma * 2^i$ that corresponds to the value that has been added to SQN^0_{UE} resulting to the sequence number in $AUTS^i$ where $\delta_i = AUTS_0 \oplus AUTS^i$. Using all already inferred bits of SQN^0_{UE} at positions

Our acronym	In 3G	In 4G	In 5G	Description (if needed)
IMSI	IMSI	IMSI	SUPI	International Mobile Subscriber Identity
UE/Subscriber	MS	UE	UE	User Equipment (contains a USIM)
SN	VLR+SGSN+...	eNodeB+MME+...	SEAF+AMF+gNB+...	Serving Network
HN	HE+HLR+...	HSS+HLR+AuC+...	AUSF+ARPF+UDM+SIDF+...	Home Network

Table 1. Informal correspondence between terminologies

from 0 to $p-1$, we infer if $\Gamma + SQN^0_{UE}$ yields a remainder that propagates from bit position p to $p+1$. If the latter could not be inferred, continue to the next iteration. If the latter could be inferred we let $\Gamma'[p+1]$ and $\Gamma'[p]$ be the bits positions of Γ plus the (possible) remainder.

Now, using those information and the Table 6, we may be able to infer the bit of SQN^0_{UE} at position p. The symbol † means that the given δ_i is incoherent with previously analysed δ_j. This may happens if some δ_i were not fetched correctly for instance.

Increment	Bits $\delta_i[p+1]$, $\delta_i[p]$			
$\Gamma'[p+1]$, $\Gamma'[p]$	0, 0	0, 1	1, 0	1, 1
0, 1	†	0	†	1
1, 1	†	1	†	0
...	Impossible to infer, just continue			

Fig. 6. Given the bits of the increment Γ at position $p+1$ and p, and the bits of δ_i at position p and $p+1$, this table gives the inferred bit of SQN^0_{UE} at position p. † indicates an inconsistency.

When a fixed point has been reached (*i.e.,* the two loops have been completed without being able to infer more bits of SQN^0_{UE}), stop and return the inferred bits.

Discussion. Note that for increment value we sometimes observed (*e.g.,* $\gamma = 33$), this algorithm allows to infer more bits than the algorithm described in Section 3.2.2 (*i.e.,* twice when $n = 6$). Indeed, the binary representation of 33 is 100001 and thus, each time we analyse a certain δ_i with its corresponding increment $\Gamma = \gamma * 2^i$, we are able to exploit the bit position i (because of the addition of $1_2 * 2^i$) and the bit position $i+6$ (because of the addition of $100000_2 * 2^i$).

Finally, note that this algorithm can be made even more generic by dealing with $AUTS$ fetched in different ways (*e.g.,* yielding from the injection of consecutive challenges instead of power of 2): it suffices to adapt the way we compute Γ.

C Other Variants of *SQN* Policies

According to non-normative specifications (Sections C.2 and C.3 from TS 33.102 [6]), *SQN* and its update policy can take different forms. We briefly explain how our attack can be easily adapted for those variants.

SQN can be made of two components $SQN = SEQ||IND$ where *SEQ* is a 43 bits long integer that counts all past AKA sessions and *IND* is a 5 bits long index. Essentially, *IND* indicates the *SN* for which *SEQ* should be used and the USIM stores possibly other *SEQ* corresponding to other *SNs* (this is called *array mechanism* in the specification [6]). When such a policy is in use, one can use a slightly different variant of our attack: (i) injections of authentication challenges should be done while using the same *SN* identifier towards the *UE* and the same *SN* while fetching authentication tokens, and, (ii) the algorithm used to infer bits should drop the first 5 bits of *SQN*. This allows the attacker to break the counter part of *SQN*, namely *SEQ*; leading to the same privacy attacks explained in Section 4. Indeed, *SEQ* is not useful for the practical attacks we describe in Section 4, only the integer counting the number of successful AKA sessions is relevant; here *SEQ*. Note however that it seems that *IND* is actually highly predictable. For example, two *SQN* in two AUTN vectors that were requested by the *SN* are expected to use the same *IND* value.

Note that there are other policies, for example *SQN* can also be time-based. For such policies for which *SQN* does not contain ant information about the number of successful AKA session made by the *UE*, our attack no longer works. However, we have not observed any operator using such policies.

D Attacker's Setup: Using Reprogrammable USIM

In addition to the setup described in Section 5.1.1, we have shown that one case use genuine smartphones with reprogrammable USIM for fetching a large number of authentication tokens intended for the target's USIM.

We used a tool pySIM [50] to program USIM cards using an external smartcard reader. While programming commercial USIM cards is not possible, SysmoUSIM [33] cards can be reprogrammed to store any given IMSI.

We inserted the programmed USIM in a smartphone to read all the traffic (*i.e.,* signaling messages) as explained next. Having access to signaling messages, we stored the received authentication challenges ($AUTN$ and $RAND$) sent by the networks intended for the targeted USIM. We used the SCat tool [51] to gain access to signaling messages We selected Android based Huawei and Asus smartphone models due to the availability of direct access to baseband using AT commands [52] to store signaling messages automatically.

E Countermeasures

E.1 Correctly Randomise *AUTS* (F2)

One way to fix the attack vector we have discovered consists in using a fresh random generated by UE to conceal SQN_{UE} instead of reusing the one contained in the received authentication challenge (generated by the HN). This random has to be sent in the clear along with $AUTS$ in order to let the HN computes AK^* and recovers SQN_{UE}. We depict this solution in Figure 7. Note that the value MAC^* must use $RAND$ instead of $RAND^*$ so that it really plays the role of a response to the fresh challenge corresponding the received authentication challenge. Otherwise, a rogue UE could impersonate a UE by replying one of its old $AUTS$ vector forcing the HN to synchronise SQN to an older value.

With this fix, replaying two times the same authentication challenge leads to two $AUTS$ vectors whose concealed values are $c_1 = (SQN_{UE})_{t_1} \oplus \mathsf{f5}^*(RAND_1^*,\ K_{IMSI})$ and $c_2 = (SQN_{UE})_{t_2} \oplus \mathsf{f5}^*(RAND_2^*,\ K_{IMSI})$. More importantly, when xoring c_1 and c_2, the two AK^* values do not cancel out breaking the relations exploited by our attacks. Note that this fix can be implemented in

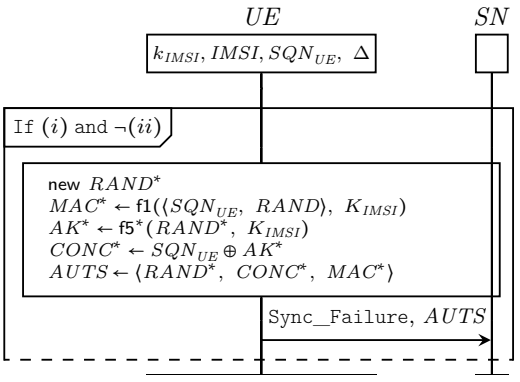


Fig. 7. Fix F2: correctly randomizing *AUTS*

combination with the fix for the linkability of failure messages described in Section 7.1.

However, this solution still suffers from the following minor flaw. When replaying two times the same authentication challenge, even though the two concealed values in the two replies are different as argued above, the two MAC^* from the two replies may be equal (*i.e.,* when SQN_{UE} has not been modified between the two replays).

The attacker may exploit such equality to link a subscriber although we believe that the underlying attack is much less severe than our attacks. In order to solve this issue, one may mix $RAND$ and $RAND^*$ in the replied MAC^* message as follows:

$$MAC^* \leftarrow \mathsf{f1}(\langle SQN_{UE},\ RAND,\ RAND*\rangle,\ K_{IMSI}).$$

E.2 Asymmetrically Encrypt SQN_{UE} (F3)

Although the use of asymmetric encryption methods was considered too costly and impractical during the 4G system design, it is now known that 5G will rely on it. A fix based on asymmetric encryption, similar to the one presented in [7], is depicted in Figure 8. Similarly to the fix F1, it can be adapted to hide the reason of the failure and can be improved using a random generated by the UE.

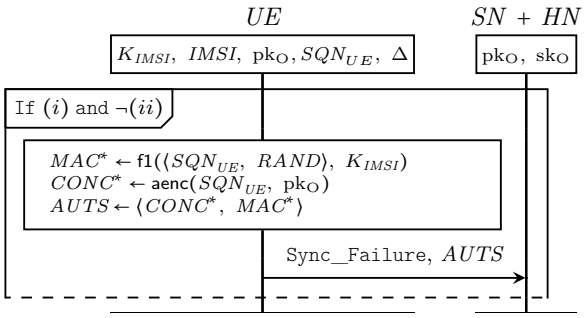


Fig. 8. Fix F3: asymmetrically encrypting SQN_{UE}