Pierre-Alain Fouque, Cristina Onete, and Benjamin Richard

Achieving Better Privacy for the 3GPP AKA Protocol

Abstract: Proposed by the 3rd Generation Partnership Project (3GPP) as a standard for 3G and 4G mobile-network communications, the AKA protocol is meant to provide a mutually-authenticated key-exchange between *clients* and associated network *servers*. As a result AKA must guarantee the indistinguishability from random of the session keys (key-indistinguishability), as well as client- and server-impersonation resistance. A paramount requirement is also that of client privacy, which 3GPP defines in terms of: *user identity confidentiality*, *service untraceability*, and *location untraceability*. Moreover, since servers are sometimes untrusted (in the case of roaming), the AKA protocol must also protect clients with respect to these third parties. Following the description of client-tracking attacks e.g. by using error messages or IMSI catchers, van den Broek *et al.* and respectively Arapinis *et al.* each proposed a new variant of AKA, addressing such problems. In this paper we use the approach of provable security to show that these variants still fail to guarantee the privacy of mobile clients. We propose an improvement of AKA, which retains most of its structure and respects practical necessities such as key-management, but which provably attains security with respect to servers and Man-in-the-Middle (MiM) adversaries. Moreover, it is impossible to link client sessions in the absence of client-corruptions. Finally, we prove that any variant of AKA retaining its mutual authentication specificities *cannot* achieve client-unlinkability in the presence of corruptions. In this sense, our proposed variant is optimal.

Keywords: privacy, security proof, AKA protocol

DOI 10.1515/popets-2016-0039
Received 2016-02-29; revised 2016-06-02; accepted 2016-06-02.

1 Introduction

Authenticated key-exchange (AKE) protocols address a fundamental goal in cryptography, namely that of allowing two parties to communicate securely over an insecure channel (like the Internet, a radio-frequency channel, or a mobile telecommunications network). Such protocols are constructed in two steps. First, the two parties exchange information that allows them to establish several *short-term* session keys. These are later used to secure and authenticate data exchanged between the parties. AKE schemes ensure that sensitive data or services are securely provided from a server to a legitimate client.

In the context of mobile networks, mobile services, such as calls, SMS privileges, or Internet use, are granted to clients over a secure channel, following an authenticated key-exchange protocol called AKA[1]. This protocol was introduced by the Third Generation Partnership Project (3GPP), which wrote and has been maintaining the specifications of 3G telecommunication systems. Note that mobile services must be provided both in domestic, and in roaming scenarios, and only to clients who are entitled to them. Moreover, the nature and destination of such services should remain private.

The AKA protocol. The AKA scheme was developed at the end of the previous century, and is symmetric-key only. The mobile-network context imposes a peculiar architecture to the AKA design. Thus, typical 3G and 4G networks involve three types of participants. Mobile *clients* may subscribe to a single *operator*, thus becoming entitled to mobile services. The latter are provided across a secure channel, not by the operator, but by an intermediate *local* network operator that we call a *server*. In domestic scenarios, the server and operator are affiliated together, and can thus both be trusted. In the case of roaming, the server is associated to a different operator and only trusted to provided services, not to know the client's long-term secret values (known only to the client and the operator). Servers do know the short-term secret values necessary for the secure-channel establishment, in order to then provide the service. The AKA setting requires three participants, unlike the classical two-party AKE setting.

The AKA scheme relies on two long-term symmetric keys. As a peculiarity of the subscriber-operator architecture, clients are associated both with a unique client-key and with their operator's key. Since this key is shared between a potentially very large number of users, the clients store a value derived from both keys denoted Top_C, and the subscriber key. only a (one-way) function of this, and the client-key.

Three crucial constraints influenced the design of AKA in the 1990s. At that time (and even to the present day)

Pierre-Alain Fouque: Université de Rennes 1/ IRISA, pa.fouque@gmail.com
Cristina Onete: INSA Rennes/IRISA, cristina.onete@gmail.com
Benjamin Richard: Orange Labs, benjaming.richard@orange.com

1 Despite the similarity of the acronyms, the AKA protocol is only an example of an AKE scheme. Other examples are for instance TLS/SSL or PACE/EAC.

SIM cards could not generate (pseudo)random numbers[2]. As a result of this limitation, the freshness of each session had to be guaranteed without user-generated nonces. As a second constraint, note that the communication between servers and operators is financially expensive, thus to be minimized. We note that some communication between these two parties is inevitable, since the server is untrusted, but must still be able to authenticate to the client as legitimate. In the case of AKA, operators minimize costs by generating *batches* of *authentication vectors*, containing short-term authentication and key-establishment values, but hiding the client's long-term secrets. Finally, the protocol is run over notoriously-noisy communication-channels, and it must thus be robust with respect to noise. As a result, AKA was designed to be *stateful*, allowing the two parties to authenticate to each other by means of a *sequence number*, which can be updated at each execution. Resistance to noise translates to a tolerance interval for the authenticated sequence number, which allows some leeway for each executions. Finally, in case the two parties become desynchronized, a procedure is provided to re-synchronize them.

In order to preserve client privacy, client are associated with several identifiers. Apart from the permanent, unique identifier called an IMSI, the client is also associated (by the servers) with temporary identifiers TMSI, which are unique per server. In an initial identification phase, the server must ascertain the IMSI of the client, which the operator must know to generate the authenticated key-exchange data. However, since IMSI values are permanent and recognizable, clients can use their TMSI values instead. A new TMSI value is sent encrypted, but not authenticated, as part of the secure-channel data. Servers must store the TMSI values for each client that authenticates to them; moreover, the TMSI value that was last associated with a given IMSI is kept for a longer time, since it may be demanded by another server. Finally, as a backup, in case a TMSI value is not found, or not recognized, the server will also accept an identification by IMSI.

AKA **Requirements.** Apart from several *security requirements*, such as guaranteeing the secure delivery of services, as well as client-to-server and server-to-client authentication, an important additional concern for 3G/4G communication is that of client *privacy*. The 3GPP specifications [1] specifically demand: "user identity confidentiality", "user untraceability", and "user location confidentiality", which are all formulated with respect to passive third-party eavesdroppers. The first notion requires that such an adversary may not learn the *permanent* user identifier IMSI of a client; the second refers to an adversary's ability of learning whether a given client uses the same, or different services[3]; and finally the third requires that adversaries cannot learn a client's location (since the servers running the protocol are area-specific).

These explicit requirements ensure a minimal amount of client privacy. The AKA protocol was moreover designed to guarantee a measure of security against corrupted, or malicious servers. This requirement is particularly relevant in the case of roaming, in which case the server may be untrusted or more vulnerable. We consider the three-party architecture with possible server corruptions and formulate security with respect to servers in two properties, namely: (i) *state-confidentiality*: the property that servers cannot learn the client's secret key, the operator's secret key, nor the client's state; (ii) *soundness*: that servers cannot in fact successfully run a key-exchange protocol with the client unless aided by the operator.

Protocol vulnerabilities. In this paper we focus on the (provable) privacy of AKA, but also consider its security. Three attacks in the literature, namely IMSI catcher attacks [8], IMSI paging attacks [13, 27], and impersonation by server-corruption [20], already prove that AKA does not offer the desired degree of client privacy. IMSI catchers allow passive and active adversaries to track clients by exploiting the fact that during the protocol run, the server will require clients to send their permanent identifier IMSI if the TMSI value cannot be traced back to an IMSI. IMSI paging attacks exploit the lack of authentication in the TMSI-reallocation message.

Zhang et Fang [20] note than in the case of roaming, the local server providing mobile services may be poorly secured and corruptible. An immediate consequence is that the data transmitted over the secure channel by this server can be compromised; however, Zhang et Fang also showed that the danger persists even after the client has left the area serviced by the corrupted network, since the latter can impersonate the legitimate server in a new, uncorrupted, strongly-secured network.

An older work by Ateniese et al. [4] examines the problem of untraceable mobility, in particular noting an informal paradigm: nobody but the client should know both the client's identity and location at the same time. In this context, they provide solutions to achieving better privacy in 3-party settings; however, these solutions are extremely generic, which makes them hard to immediately to AKA. Moreover, note that server-operator communication takes place across a channel that is implemented differently in practice by different operators. The

2 However, the next-generation SIM cards are able – and in fact expected – to be able to perform such computations [5].

3 The exact 3GPP wording allows two interpretations. One is that an adversary must not distinguish between two possible services used by the same user; the other interpretation is that adversaries must not know whether a single, known service is provided to two different clients or to the same client. We formalized this property following the first meaning.

protocols proposed by [4] require this channel to be implemented in a specific way. Finally, we note that, while highly worthwhile, the goal of preventing operators from learning a client's operator is incompatible with the way authentication is currently done in the AKA protocol (operators need to prepare session information for servers, across a secure, and mutually authenticated channel which forcibly reveals the identity – and implicitly the location of the server).

Although the AKA protocol was designed to protect client privacy, several attacks can be run on the physical layer or by using side-channel information. Since we focus only on the privacy of AKA at the protocol layer, this type of attacks are out of scope. Another class of attacks that are out of scope are those exploiting faulty TMSI reallocation choices (e.g., TMSIs not updated sufficiently often, or according to protocol), denial-of-service by jamming, or attacks enabled by version-shifting (forcing the user to employ a weaker, more vulnerable version of the protocol on 2G and 3G, rather than 4G) [27]. Such attacks, however, should encourage a closer examination of backward compatibility between AKA versions.

We outline further vulnerabilities, e.g. [20, 28, 29]. in Appendix A, and show that as a consequence, the AKA protocol does *not* attain even the basic-most privacy requirements outlined by 3GPP. This suggests the need for stronger security.

Client indistinguishability. Motivated by a hightened concern for client privacy, Arapinis et al. [22] were the first to consider the notion of *untraceability* of clients with respect to external adversaries. They exploited the fact that adversaries can induce faulty behaviour (resynchronization) and then distinguish a client that is forced to perform this operation from a fully synchronized one. They modified the protocol to address this, and gave a formal-security proof using ProVerif. However, this proof idealizes the long-term state used in the protocol runs, making it unclear how far the guarantee holds for the true scheme. As we show in this work, their variant does not in fact guarantee user untraceability. The attack we present breaks client-indistinguishability by exploiting the fact that an adversary can forge the encrypted IMSI message proposed by Arapinis et al. In work orthogonal to ours, Khan et al. [13] also critically examined the variant of [22], pointing out impracticalities and security/privacy failures. An important criticism addresses the PKI for clients and servers; as we explain below, in our own variant, we minimize the modifications both in terms of computation and administration costs.

Lee et al. [18] consider the untraceability of the 4G LTE (Long-term Evolution) protocol (similar to AKA, but with a different identifier- and key-management), but do not focus on the handshake itself. Their main result is that in the absence of server corruptions, LTE is (weakly) untraceable against an active MiM adversary. They focus on the security of TMSI values, and their retransmission, but surprisingly their model cannot capture IMSI-catcher attacks (which directly impact privacy *without* server corruptions). The reason is that Lee et al. assume that the TMSI reallocation and usage is perfect in the sense that a TMSI value will always lead to the legitimate IMSI value, and that once the TMSI allocation process starts, the IMSI will never again be demanded. This is not true for the AKA procedure, for which, if the active attacker replaces the TMSI with a random message, the server will demand the client's IMSI in clear. Lee et al. do not capture the attack by Arapinis, since they do not model desynchronizations, and they reduce the three-party setting to just two parties, by assuming that the server and the operator are the same entity[4]. Their security proofs rely on an assumption on the underlying cryptographic functions used in the protocol, but they did not study whether TUAK and MILENAGE (the two current instantiations of the cryptographic cipher suites) actually guarantee the required properties.

Finally, the AKA scenario is one use-case of the more generic constructions of anonymous, secure and authenticated channels presented by Alwen et al. [3]. However, their security model was conceived for, essentially, a two-party scenario (the ideal resource has interfaces to the eavesdropper, clients, and servers, but not operators). It is not clear, moreover, that Constructive cryptography (CC) supports adaptive corruptions. Finally, we note that the schemes of [3] rely on a primitive called key-indistinguishable MACs (KI-MACs), esssentially MACs that leak no information about the used keys. In our work, we tried to keep as close as possible to the presently-used algorithm suites used in AKA, namely TUAK and MILENAGE. This precludes the use of KI-MACs. However, a main subresult of this paper is a *sufficient* security requirement for the cryptographic algorithms. It would be interesting to explore whether KI-MACs could be used to modify existing suites or provide new ones.

Our contributions. We have two main contributions. The first is to show that the AKA protocol and two more promising improvements in the literature do not guarantee client-privacy, nor security with respect to the server. As a consequence, our second main contribution is to present a variant of the AKA which provably guarantees the following five properties:

– Wide-weak client-unlinkability: client sessions must be unlinkable to a MiM attacker (no corruptions allowed);
– Key-indistinguishability: the derived session keys must be indistinguishable from random to MiM adversaries;

4 In the absence of corruption, this treatment is justified, but it is incomplete in terms of real-world implementations.

– Client/server-impersonation security: no MiM adversary can impersonate either a client or a server;
– State-confidentiality: a malicious server cannot learn the client's long-term secret values;
– Soundness: without the operator's help, no malicious server can authenticate to the client.

As an additional main result, we also prove that achieving a stronger degree of client-unlinkability (see below) is impossible while the AKA protocol retains its current structure. Consequently, the degree of unlinkability we achieve is in that sense, optimal.

A precise formalization. As an implicit, but significant contribution we formalize the five security and privacy properties outlined above. Consequently, we are able to describe two attacks against AKA, one against client-unlinkability, the other against soundness; then we also proceed to show that the proposed AKA improvements of Arapinis et al. [21] and van den Broek et al. [9] are *not* client-unlinkable.

This precise formalization also allows us to find a gap in an impossibility result regarding client-unlinkability in the presence of corruptions for mutually authenticated protocols [24]. We formulate a different impossibility result, which is more precise, and also more generic in the case of the AKA protocol. Another main contribution of this paper is to extend the narrow-forward-privacy impossibility result for a broader class of protocols including AKA and its variants.

Our improvement. Our improved variant of the protocol mostly retains the symmetric character of the current version. We allow TMSI values to be backed up by IMSIs; however, we bypass IMSI catcher attacks by sending IMSIs encrypted with a public-key IND-CCA-secure encryption scheme. We assume each operator has a PKE key-tuple, and each client stores the public key of the operator; this minimizes key-management problems. We note that, just as for the AKA protocol, we only use the (encrypted) IMSI value as an alternative for the randomly-chosen temporary identifier values TMSI. However, we choose to update the current TMSI value by using authenticated encryption (AES-GCM) as part of the server's authentication message. The PKE scheme is also used when moving from area to area: thus, if the client switches from one server (in a given location) to another (in another location), the TMSI is *not* used. This allows us to reveal only the current area that the server is in, rather than the client's past location (as is the case for the current version of AKA); furthermore, we minimize the duration for which TMSI values must be stored by servers. On the other hand, in order to preserve client-unlinkability, we require that TMSI values are at least as long (large) as the output of the PKE encryption.

We retain the structure of the authenticated key-exchange part of AKA, using the client- and operator-state to authenticate the two parties and the derived session keys. We show that while this feature of the AKA protocol reamins in use, no client-untraceability can be achieved in the presence of corruptions. By removing the need for re-synchronization, we also implicitly prevent attacks which link client sessions depending on whether or not the re-synchronization procedure is used.

We note that our improvement follows guidelines by one of the leading mobile service providers in Europe [23]. Table 1 compares our proposal to the AKA protocol and to the two more promising variants we also analyze in this paper.

2 Privacy model

2.1 3GPP Privacy requirements

AKA Infrastructure. The mobile context for which the AKA protocol was designed contains three entities: (1) clients, which register with operators and are allowed to access a subset of services; (2) operators, which know the secret parameters of all their registered clients; and (3) local servers, which are tasked with providing services to mobile clients, but are not trusted to know the clients' personal information. In the AKA literature, opeators are usually called home local registers (HLR), while servers are known as VLR.

The security demands of 3G/4G networks are client-centric, revolving around the following parameters related to mobile clients (users) C:

IMSI : a permanent identifier, unique per customer and highly trackable;

TMSI : a temporary identifier, unique per server, and updated after each successful protocol run;

LAI : a unique local-area identifier per server; client store (TMSI, LAI) tuples whenever a server issues a new TMSI;

sk_{C} : the client's unique client key;

$\mathsf{sk}_{\mathsf{op}}$: the key of the operator C subscribes to;

$\mathsf{Sqn}_{\mathsf{C}}, \mathsf{Sqn}_{\mathsf{Op,C}}$: the client's state $\mathsf{Sqn}_{\mathsf{C}}$ has an equivalent operator state $\mathsf{Sqn}_{\mathsf{Op,C}}$, which should not be "too far" from the client's state. The state is updated by the client upon authenticating the server (correct verification of the authentication challenge); the server updates state upon authenticating the challenge (verification of the authentication response).

We refer the reader to Section 3.1 for more details about the protocol description.

The identifier and key-management schemes are as follows. Clients may know the permanent value IMSI, the

	Defeating:			Security:				
	Attack n°1	Attack n°2	Attack n°3	Prop. n°1	Prop. n°2	Prop. n°3	Prop. n°4	Prop. n°5
3G AKA	x [9]	x [22]	x	x	x	x	?	?
Arapinis	✓ [22]	✓ [22]	x	✓	x	x	?	✓* [22] [5]
Van Den Broek	✓ [9]	x	✓	✓	x	x	?	?
Our variant	✓	✓	✓	✓	✓	x	✓	✓

Attack n°1: IMSI Catcher § 3.2.	Prop. n°1: Confidentiality of the previous location §2.
Attack n°2: Linkability of failure messages § 3.2.	Prop. n°2: ww-unlink §2.2.
Attack n°3 : Our traceability attack § **??**.	Prop. n°3: nf-unlink §2.2.
Prop. n°4: State-confidentiality & soundness §2.2.	Prop. n°5: Key-indistinguishability & Client- and Server-impersonation §2.2.

Fig. 1. Comparison between several AKA variants. For attacks, a ✓ denotes the protocol resists the attack, while x denotes a vulnerability; for properties, a x denotes the property is *not* achieved, while a ✓ indicates security with respect to that property.

temporary identifiers TMSI and LAI, the keys $\mathsf{sk}_C, \mathsf{sk}_{op}$ and a function of sk_C and sk_{op} (they do not store sk_{op} in clear, as discussed in section 3.1). Operators know the tuple: $(\mathsf{IMSI}, \mathsf{sk}_C, \mathsf{sk}_{op})$. Servers keep track of tuples $(\mathsf{TMSI}, \mathsf{LAI}, \mathsf{IMSI})$. Furthermore, both servers and operators know the sequence number that the operator associated to each client. The clients update their own sequence numbers, which are highly related to the operator's sequence number.

Very notably, servers must both authenticate and exchange session keys with mobile clients, despite not knowing their secret material.

Client Privacy. The Third Generation Partnership Project (3GPP), which designed the AKA protocol in the TS.33.102 specification [1], lists the following privacy requirements:

- **user identity confidentiality:** specifically, *"the property that the permanent user identity (*IMSI*) of a user [...] cannot be eavesdropped on the radio access link."*
- **user untraceability:** namely, *"the property that an intruder cannot deduce whether different services are delivered to the same user by eavesdropping on the radio access link."*
- **user location confidentiality:** in particular, *"the property that the presence or the arrival of a user in a certain area cannot be determined by eavesdropping on the radio access link."*

The requirements quoted above are quite informal; moreover, the nomenclature is confusing, since in the *provable-security* literature, *untraceability* refers to adversaries tracing clients in distinct protocol runs (rather than it being service-related). We discuss the three requirements below, then formalize them into cryptographic requirements.

User identity confidentiality concerns only the client's *permanent* IMSI value (not, e.g. the client's sequence number) with respect to *passive* attackers (rather than active ones). However, mobile networks are notoriously prone to Man-in-the-Middle (MiM) active attacks like the IMSI catcher [8], which allows a third party (the MiM) to recover a client's IMSI. Another highly-trackable client-specific parameter is the sequence number Sqn, whose updating procedure is very simplistic and its output, predictable even without a secrey key. As a consequence we require the stronger property of *provable unlinkability*, which ensures that even an *active* MiM cannot *link* two AKA protocol runs to the same client.

For user untraceability, no attacker must know whether the same service (i.e. *any* message-exchange over the secure channel) is provided to a client multiple times. From the point of view of provable security, this is equivalent to *key-indistinguishability* if the authenticated-encryption algorithms are assumed to be secure.

User location confidentaility demands that eavesdroppers $\mathcal{A}$ cannot detect the presence of a client in a given area; however, the definition does not specify what information $\mathcal{A}$ links to each client (e.g. the IMSI, the sequence number, etc.). Attackers are aware of the current LAI; the difficulty lies in learning which clients *enter* the area. Unfortunately the AKA protocol always reveals the *past* location of any arriving client, making unique (or rare) itineraries stand out. We formalize a strong degree of location privacy as a part of client-unlinkability.

Our formalizations of *client unlinkability* and *key-indistinguishability* consequently guarantee 3GPP's three privacy requirements.

Implicit requirements. As discussed in Section 1, the AKA protocol implicitly addresses security with respect to malicious servers, which are restricted as follows: (1) the servers have no access to the tuple $(\mathsf{sk}_C, \mathsf{sk}_{op})$; (2) the (hence necessary) operator-server communication must be minimized in order to minimize costs.

We formulate the following two *implicit* requirements:

- **State-Confidentiality:** Servers must not learn any client-related long-term state.

– **Soundness:** Clients must reject authentication-challenges not explicitly provided by the operator to the server.

2.2 Security models

Due to their orthogonality, it is hard to formalize *key-indistinguishability* and *client-unlinkability* in the same generic framework. Indeed, the unlinkability notion requires the adversary to have access to clients without knowing their identities. We follow established approaches [11, 26] and associate clients in the unlinkability model with identifiers (handles) denoted VC (Virtual Client); this changes the oracle syntax. Thus, we differentiate between the model for security (including key-indistinguishability, client- and server-impersonation resistance against MiM attackers, and state-confidentiality and soundness with respect to malicious servers), and that of client-unlinkability, using similar oracles with a different syntax. Due to space restrictions, we include only intuitive descriptions of the oracles, leaving the full formalization to the full version.

Setup and participants. We consider a set $\mathcal{P}$ of participants, which are either a server S_i or a mobile client C_i. Operators Op are not modeled as active parties: for all the games except state-confidentiality and soundness, the operators Op are black-box algorithms within the server S; in those two games, Op are oracles, which the adversary (i.e. the server) may query. We assume the existence of n_{C} clients, n_{S} servers, and n_{Op} operators. We assume that all "copies" of the operator are synchronized at all times, though their outputs might depend on which server queries them. We associate each client with: long-term keys $(\mathsf{sk}_{\mathsf{C}}, \mathsf{sk}_{\mathsf{op}})$; an ephemeral state st_{C} equal to the sequence number $\mathsf{Sqn}_{\mathsf{C}}$; the identifiers IMSI (permanent) and TMSI (temporary); and a tuple of a current-, and a past local area identifier, denoted $\mathsf{past.LAI}_{\mathsf{P}}$ and $\mathsf{curr.LAI}_{\mathsf{P}}$. Servers are associated with a permanent local area identifier LAI and a unique network identifier $\mathsf{ID}_{\mathsf{S}_i}$; they also keep track of a list of tuples $(\mathsf{TMSI}, \mathsf{IMSI})$ associated with clients. Each of the at most n_{S} servers has black-box access to algorithms (or oracles for state-confidentiality and soundness) $\mathsf{Op}_1, \ldots, \mathsf{Op}_{n_{\mathsf{Op}}}$, initialized with long-term keys $(\mathsf{sk}_{\mathsf{Op}_i})$ and tuples $(\mathsf{IMSI}, \mathsf{sk}_{\mathsf{C}}, \mathsf{Sqn}_{\mathsf{Op},\mathsf{C}})$. We also assume that the key space of all operators is identical (otherwise it becomes easier to distinguish between clients of different operators).

Client-Unlinkability. Informally, a protocol Π is *client-unlinkable* if no adversary can know whether two runs of Π involve the same, or by two different clients. Sessions associated with the same client are called *linked*. Following previous work by Vaudenay [26] and Hermans et al. [11], we give the adversary access to a left-or-right oracle associating an anonymized handle VC to one of two possible clients (input by the adversary). We moreover account for client mobility, giving the adversary access to a *relocation* oracle. If an attacker can distinguish between clients based on their location, she wins the unlinkability game detailed below.

At the onset of this game, the set of clients is empty and we choose the operators' secret keys. The adversary can initialise servers by choosing their locations, and it can create clients to populate the system it attacks. Each newly-created client has a past location $\mathsf{past.LAI}_{\mathsf{C}}$ set to $\perp$, and an adversarially-chosen current location. The adversary interacts with clients by using the oracles below. Clients can be "drawn" or "free"; at creation, all clients are "free", and they may become "drawn" if used as input to a left-or-right Client-Drawing oracle[6]. Only free clients can be drawn. The left-or-right oracle associates a handle VC to either the left or the right input client, depending on a secret bit b. The adversary's goal is to guess this bit. The client-unlinkability property is more formally defined below.

– The challenger randomly chooses $b \in \{0, 1\}$ and instantiates all the operators.
– The adversary uses the oracles below (depending on adversarial class); the challenger answers the queries.
– The adversary outputs a guess d of the bit b.

The adversary *wins* if and only if $d = b$, and her winning advantage against a protocol Π is:

$$\mathcal{A}_{\Pi}^{\mathsf{c.unlink}}(\mathcal{A}) := |\mathsf{Pr}[\mathcal{A} \text{ wins } \mathsf{Exp}_{\mathcal{A}}^{\mathsf{c.unlink}}(1^{\lambda})] - \frac{1}{2}|.$$

Following [11, 26], we call an adversary *narrow* if she cannot learn whether a protocol run was successful (she cannot query Result). The opposite of *narrow* are *wide* adversaries. Adversaries are also classified in terms of their use of the Corrupt oracle: attackers that cannot corrupt clients are called *weak*; they are *forward* if any corruption query may only be followed by more corruption queries (the adversary cannot free or interact with clients or servers); or *strong* if their access to oracles is unrestricted.

3GPP only formulates queries with respect to "*eavesdroppers on the radio link*", i.e. *narrow/wide-weak* adversaries that are passive. In this paper, we also consider active weak attackers and obtain a better privacy guarantee.

Formalization. We quantify adversaries in terms of: adversarial *class*, which we abbreviate to α-c.unlink, with $\alpha \in \{\mathsf{nw}, \mathsf{ww}, \mathsf{nf}, \mathsf{wf}\}$ (for narrow/wide-weak, and narrow/wide-forward adversaries); execution time is t; maximum number

6 We use this state to rule out trivial attacks in which an adversary can distinguish a client simply because it is not in its original state (having already started the protocol run beforehand).

q_{exec} of sessions instantiated per client C; maximum number q_{id} of user identification per session; and the maximum number q_G of queries to the function G. We formalize the following definitions.

Definition 1. [**Weak Unlinkability**] *A protocol* Π *is* $(t, q_{\mathsf{exec}}, q_{\mathsf{id}}, q_G, \epsilon)$-nw/ww-*client-unlinkable if no narrow/wide-weak-adversary running in time* t*, creating at most* q_{exec} *sessions with* q_{id} *user-identifications per session, and making at most* q_G *queries to* G*, has an advantage* $\mathsf{Adv}_{\Pi}^{\mathsf{w.c.unlink}}(\mathcal{A}) \geq \epsilon$.

We define forward adversaries similarly.

Oracles. The adversary may use the following oracles, in addition to a function G modeled as a PRF, which "encompasses" all the AKA cryptographic functions:

- CreateCl(Op, LAI): creates a new, legitimate, free client labeled C_i at a location LAI for which a server is already defined (else the oracle outputs $\perp$). The adversary is given IMSI, $\mathsf{st}_{\mathsf{C}_i}$, and the label C_i (used later to select clients).
- CreateS(LAI): creates a server S_i at a new location LAI.
- Launch(VC, S_j): instantiates a new session (labeled by a unique identifier s) between the client associated with VC and the server S_j. It outputs s and an initial protocol message m from S_j to VC.
- DrawCl($\mathsf{C}_i, \mathsf{C}_j$): on input free clients $\mathsf{C}_i, \mathsf{C}_j$ with a same operator and LAI, generate a handle VC to be associated with either C_i or with C_j depending on a secret bit b. The value VC is output.
- FreeVC(VC): frees the two clients queried to get VC, aborting any outstanding sessions.
- Relocate(VC, LAI*): modifies the *current* location of the two clients associated with VC to LAI*, and aborts running executions of VC.
- Send(P, s, m): sends message m in session s either to a virtual client VC or to a terminal S, outputting a response message m'.
- Execute(VC, S, s): simulates a protocol run between the client associated with VC and the server S, outputting transcript τ.
- Result(P, s): returns a bit indicating the acceptance/rejectance by party P of its partner in session s.
- Corrupt(C): returns the full state, identifiers, and location information of client C.

MiM Security. The notion of *key-indistinguishability* requires that the session keys computed in the handshake be indistinguishable from random bitstrings of equal length. We use an (unanonymized) subset of the oracles above and give the adversary the ephemeral state (the sequence number) of both clients and servers. Corrupted parties are *adversarially controlled*.

We consider multiple protocol executions, and talk about party *instances* P_i, each corresponding to a single session of party P. For *key-indistinguishability*, clients are associated with unique identifiers UID and we abstract away location data. Sessions have identifiers sid consisting of: the client's UID (and implicity $\mathsf{sk}_{\mathsf{UID}}$), the server identifier $\mathsf{ID}_{\mathsf{S}_i}$, the randomness generated by the server, and the sequence number used for the authentication. Each party instance also keeps track of a partner ID pid; partners are defined as parties that share the same session ID. Finally each party instance has an accept/reject bit denoting whether they terminated in an accepting state (authenticating their partner as legitimate and computing the session keys).

The key-indistinguishability game begins by generating the keys for all the operators and servers. Each server is assumed to internally keep track of n_{Op} operator algorithms, which it queries in a black-box way. The algorithms may return different output depending on the server in which they run, but they are synchronized with respect to client states. The adversary may create clients, registering them with operators, she may create party instances, may run concurrent sessions with instances of servers and clients, may corrupt clients or servers, and will finally issue a single Test query, which returns either the real or the random keys of a *fresh* session. Namely, neither partner must be corrupted or adversarially controlled, and we also rule out trivial wins due to key-reveal queries. If we consider server corruptions, the adversary may query operator algorithms initialized for that server.

The full formalization can be found in [10]. For key-indistinguishability, the adversary's advantage is:

$$\mathsf{Adv}_{\Pi}^{\mathsf{K.Ind}}(\mathcal{A}) := |\Pr[\mathcal{A}\ \mathsf{wins}] - 1/2|.$$

Definition 2. [**Key-indistinguishability**.] *A key-agreement protocol* Π *is* $(t, q_{\mathsf{exec}}, q_{\mathsf{id}}, q_{\mathsf{serv}}, q_{\mathsf{Op}}, q_{\mathsf{G}}, \epsilon)$-*key-indistinguishable if no adversary running in time* t*, creating at most* q_{exec} *party instances with* q_{id} *user identification per instance, corrupting at most* q_{serv} *servers, making at most* q_{Op} *OpAccess queries per operator per corrupted server, and at most* q_{G} *queries to* G*, has an advantage* $\mathsf{Adv}_{\Pi}^{\mathsf{K.Ind}}(\mathcal{A}) \geq \epsilon$.

We define *impersonation security* in an analogous way for a different notion of freshness (we rule out relay attacks). We defer the precise definition of this property to the full paper [10].

Security w.r.t. servers. For the notions of *soundness* and *state-confidentiality*, the adversary is a malicious, but legitimate, unique server S. Though 3GPP specifications allow servers to communicate with each other, *how* they do this is

not apparent. We use the OpAccess oracle to give the server access to operators on specific clients, but change the syntax so that the oracle takes as input only a client identifier C, and outputs material only for a single protocol session sid. The adversary uses the Send, CreateCl, NewInstance, Execute, and StReveal oracles as described in the key-secrecy model. We additionally modify the Corruption oracle as follows:

– $\mathsf{Corrupt}(\mathsf{P}) \rightarrow \mathsf{sk}_\mathsf{P}$: if P is a client, behave as before. If P is an operator, return sk_op and the list of tuples $(\mathsf{UID}, \mathsf{sk}_\mathsf{UID}, \mathsf{st}_{\mathsf{Op},\mathsf{C}})$ for all clients C subscribing with that operator.

Key-indistinguishability addresses the security of *session* keys with respect to MiM adversaries; by contrast, *state-confidentiality* demands that *long-term* client state remains confidential with respect to *malicious servers*. The game begins with instantiating operators and clients. Then the adversary interacts arbitrarily with these entities, finally outputting a tuple of four values: a client identifier, a client key, an operator key, and a client or operator state, winning if at least one of the latter four values is correct. Neither the target client, nor its operator, must not be corrupted.

Definition 3. [**State-Confidentiality**.] *A key-agreement protocol* Π *is* $(t, q_{\mathsf{exec}}, q_{\mathsf{id}}, q_{\mathsf{Op}}, q_{\mathsf{G}}, \epsilon)$*-state-confidential if no adversary running in time* t*, creating at most* q_{exec} *party instances, with at most* q_{id} *user identification per instance, making at most* q_{Op} *queries to any operator* Op*, and making at most* q_{G} *queries to the function* G*, has an advantage* $\mathsf{Adv}^{\mathsf{St.conf}}_{\Pi}(\mathcal{A}) \geq \epsilon$.

In the *Soundness* game, no server must be able to make a fresh client instance terminate in an accepting state without help from the operator. This game resembles impersonation-security, but with respect to a legitimate server with access to operators. The adversary may interact with oracles in the soundness game arbitrarily, but we only allow a maximum number of q_{Op} queries to the OpAccess oracle per client. The adversary wins if there exist $(q_{\mathsf{Op}}+1)$ fresh client instances of a given client which terminated in an accepting state. Freshness is defined as in the impersonation game. The advantage of the adversary is defined as:

$$\mathsf{Adv}^{\mathsf{S.sound}}_{\Pi}(\mathcal{A}) := \Pr[\mathcal{A}\ \mathsf{wins}].$$

Definition 4. [**Soundness**.] *A key-agreement protocol* Π *is* $(t, q_{\mathsf{exec}}, q_{\mathsf{id}}, q_{\mathsf{Op}}, q_{\mathsf{G}}, \epsilon)$*-server-sound if no adversary running in time* t*, creating at most* q_{exec} *protocol instances, with at most* q_{id} *user identification per instance, making at most* q_{Op} *queries to any operator* Op*, and making* q_{G} *queries to the function* G*, has an advantage* $\mathsf{Adv}^{\mathsf{S.sound}}_{\Pi}(\mathcal{A}) \geq \epsilon$.

3 The AKA protocol

3.1 Description of the AKA protocol

The AKA protocol used by 3G (and 4G) networks, which is fully depicted in the full version, is used to establish secure channels between mobile clients and servers. Ultimately, the server must transmit mobile services to the client over the secure channel.

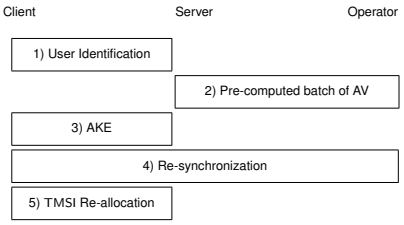


Fig. 2. The five phases of the AKA Procedure.

This protocol is actively run by clients and servers in the (selectively-active) presence of an operator. Servers and operators communicate over a secure and private channel; however, the server is considered only partially trusted. Section 2 describes in detail the setup of the three parties.

The AKA protocol consists of five phases. The first phase, *user identification*, is run by a client C and a server S on the *insecure channel* and it allows S to associate C to an IMSI value. A user ID request is first sent from the server to the client. The client's response is a tuple $(\mathsf{TMSI}, \mathsf{LAI})$, consisting of a temporary identifier and the local area identifier in which C received TMSI. If the LAI value corresponds to the LAI of S, then the latter searches for a tuple (TMSI, IMSI) in its own database; else S requests this tuple from the server S^* associated with LAI, over an unspecified channel. If no IMSI can be found, then the server demands the IMSI in clear. This procedure is the essential vulnerability leading to IMSI catcher attacks 3.2.

The second phase is run only optionally by the server S and the client's operator Op over a *secure channel*; its purpose is to enable S to then run a batch of AKE sessions with C. The server sends the client's IMSI to Op, which generates a batch of vectors AV each providing material for one out of a maximum of n sessions. For each vector, the operator's state with respect to the client $\mathsf{Sqn}_{\mathsf{S},\mathsf{C}}$ is augmented, and then the following values are generated: a fresh random value R; an server-authentication value Mac_S (for the values $\mathsf{Sqn}_{\mathsf{S},\mathsf{C}}$ and R); a client-authentication value Mac_C (for R only); the session keys CK and IK; and an anonymity key AK. Of these six values, the last five are computed by us-

ing each time a different cryptographic algorithm, denoted $\mathcal{F}_1, \ldots, \mathcal{F}_5$. In fact, the AKA protocol uses seven such algorithms, but two of them, denoted $\mathcal{F}_1^*, \mathcal{F}_5^*$, are only used in the *re-synchronization procedure*. The seven algorithms are generic, and can currently be instantiated in one of two ways, one using AES (called MILENAGE), the other using Keccak (called TUAK). Both $\mathcal{F}_1$ and $\mathcal{F}_1^*$ take as input the keys sk_C and sk_op, the random value R, and a sequence number $\mathsf{Sqn}_{\mathsf{S},\mathsf{C}}$. The other algorithms use the secret keys and the random value, but not the sequence number. At the end of this phase, the following values are sent to the server for each of the n sessions: $\mathsf{AV} = (\mathsf{R}, \mathsf{CK}, \mathsf{IK}, \mathsf{Mac}_\mathsf{S}, \mathsf{Mac}_\mathsf{C}, \mathsf{AMF}, \mathsf{AK} \oplus \mathsf{Sqn}_{\mathsf{S},\mathsf{C}})$, in which AMF is the Authentication Management Field is a 16-bit value used only in radio access specifications (for example E-UTRAN or non-3GPP access to EPS).

The sequence number $\mathsf{Sqn}_{\mathsf{S},\mathsf{C}}$ is notably *not* sent in clear to the server, but rather blinded by the value AK.

The third phase of the protocol, *authenticated-key-exchange*, is a mutual authentication and key agreement between the server and the client over the insecure channel. The server chooses the next vector available (if phase 2 was run, then this is the first tuple; else, for returning clients phase 3 is run directly, with the next authentication vector), and sends an *authentication challenge* consisting of the random value R and an authentication string $\mathsf{Autn} = (\mathsf{Sqn}_{\mathsf{S},\mathsf{C}} \oplus \mathsf{AK}) \| \mathsf{AMF} \| \mathsf{Mac}_\mathsf{S}$. The client uses R to compute AK, then it recovers $\mathsf{Sqn}_{\mathsf{S},\mathsf{C}}$ and verifies Mac_S. If the verification succeeds, and if the recovered Sqn is within a distance of Δ (a pre-defined constant) of the client's own state Sqn_C, then the client computes CK, IK, and the response Mac_C, sending this latter value to S; else, if the two sequence numbers are too far apart, then the client forces a *re-synchronization procedure*, which is the fourth phase of the protocol. If no re-synchronization is needed, then the client updates $\mathsf{Sqn}_\mathsf{C} := \mathsf{Sqn}_{\mathsf{Op},\mathsf{C}}$, and S verifies the received authentication value with respect to the Mac_C sent by Op. If the verification succeeds, then the server sends an acknowledgement to Op and goes directly to phase five. If the verification fails, then the protocol is aborted.

The fourth phase of the protocol, *resynchronization*, is run by all three parties. The client essentially retraces the operator's steps, using its own sequence number Sqn_C and computing the values $\mathsf{Mac}_\mathsf{S}^*$ and $\mathsf{AK}^* \oplus \mathsf{Sqn}_\mathsf{C}$ by using algorithms $\mathcal{F}_1^*$ and $\mathcal{F}_5^*$ (rather than $\mathcal{F}_1$ and $\mathcal{F}_5$), but keeping the same random value R. If the authentication string $\mathsf{Mac}_\mathsf{S}^*$ verifies for the Sqn value Op recovers, then Op resets its sequence number to Sqn_C and sends to the server another batch of authentication sessions. The protocol restarts. We note that this phase is executed only optionally.

Finally, the fifth phase of AKA, TMSI *re-allocation*, is run by the server and client. As the first message of the record layer, the server sends an (unauthenticated) encryption of a new TMSI value to the client C, using the session key CK computed in phases 3 or 4. The encryption is done by means of the A5/3 algorithm detailed in TS 43.020 [2], run in cipher mode. The new TMSI value, called $\mathsf{TMSI}_\mathsf{new}$, is only permanently saved by the server if the client acknowledges the receipt; else, both values $\mathsf{TMSI}_\mathsf{new}$ and the old $\mathsf{TMSI}_\mathsf{old}$ are retained and can be used in the next authentication procedure.

3.2 Some Privacy breaches in AKA

In this section, we present some vulnerabilities, both of the original AKA protocol, and of two promising improved variants of it. We begin by analyzing two client-unlinkability flaws in AKA: the first one, based on generating, then interpreting failure messages; the second, based on capturing, then tracking permanent identifiers (IMSI). These weaknesses have been partially described by Arapinis et al. [21, 22]. Then, we discuss concrete attacks against two improved variants of AKA.

Flaws in the AKA protocol

Linkability of failure messages.

Arapinis et al. [22] showed a practical attack against the AKA protocol, relying on the study of failure messages. The target client is here denoted C. The adversary wants to test whether a specific client is indeed C or another potential client C^*. By replaying an old authentication vector (in phase 3 of the protocol) which has already been used with C, the adversary forces the target client to send a desynchronization message, initiating phase 4 of the protocol. By contrast, if the adversary sends the message to C^*, the expected error message will be an error in the MAC verification. By analyzing the error, the adversary will distinguish between clients, thus breaking client-unlinkability.

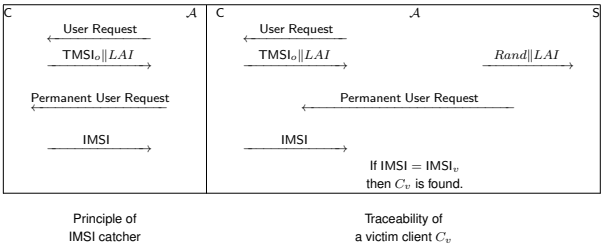


Fig. 3. Attacks based on TMSI.

IMSI *catchers and* TMSI *problems.* In the identification phase of AKA, the temporary TMSI values are meant to hide permanent IMSIs. However, even a weak active adversary can introduce noise in the communication, changing the

sent TMSI to something the server will not recognize. The server's backup procedure is then to ask for the IMSI value in clear. This attack, depicted in Figure 3 and called an "IMSI catcher" [8], is the best known threat to mobile users' privacy. This allows mass-surveillance attackers to track multiple users at a time.

Another problem stems from the TMSI re-allocation procedure in phase 5. Upon authentication, the server sends a new TMSI value $\mathsf{TMSI}_{\mathsf{new}}$ in the record layer; however, the update is only done if the client acknowledges it. If this message is dropped, the server will accept both the old TMSI value $\mathsf{TMSI}_{\mathsf{old}}$ and the new $\mathsf{TMSI}_{\mathsf{new}}$ for the next protocol run. Thus, the same TMSI will be replayed across sessions.

An adversary can also force a Denial-of-Service by sending a random value instead of the server's encryption of the new TMSI; since the encryption is unauthenticated, the client interprets this as a new (and bogus) TMSI.

Finally, note that, since the client sends in clear the LAI of its last-visited location, an adversary can also differentiate distinct itineraries.

Improved AKA variants

We proceed to show that two more promising improvements of AKA are still vulnerable to client-unlinkability attacks.

The Arapinis variant. Arapinis et al. [22] propose an AKA variant which is supposed to ensure client unlinkability by avoiding failure-message-based linking, as described in the previous section. They propose to make error messages indistinguishable by encrypting with the operator's public keya message including the IMSI, a constant "Fail" string, a random value R and the current sequence number $\mathsf{Sqn}_{\mathsf{C}}$. The latter is encrypted with an unlinkability key $\mathsf{UK} = f_{\mathsf{sk_C}}(\mathsf{R})$ in order to authenticate the error message. The operator deduces the type of failure by decrypting the error message, and deriving the IMSI and the client's sequence number.

In this variant, IMSI catcher attacks are addressed by encrypting the IMSI with the same probabilistic PKE scheme used on error messages, thus removing the need for temporary identifiers TMSI. For the analysis, Arapinis et al. model servers and operators as a single party, thus failing to take into account the costs of server-operator communication and security with respect to servers.

Despite this security proof, the variant Arapinis et al. propose is unfortunately not client unlinkable, since clients can be traced as long as the adversary knows all IMSI values. Note that IMSI values were designed not to be private (and are always known by servers).

We exploit the fact that *any party* that knows the IMSI value of a client can construct a valid encryption of the IMSI and test whether it was the appropriate value for a given client. This attack proceeds as follows in our model (we give details in the full version of our paper): (1) the adversary creates two clients (receiving their IMSI values); (2) it uses the Client-Drawing oracle to draw one of them, depending on the bit b; (3) $\mathcal{A}$ then forwards the user ID request from the server to the client, but blocks the client's reply; (4) instead, it encrypts the IMSI of the first client (the victim client), and sends it to server; (5) finally the protocol is allowed to proceed. If the client accepts the server's challenge message, consisting of R and Autn, then $\mathcal{A}$ guesses that this is the victim client; else, it guesses that the Client Drawing oracle chose the other client. In particular, letting Π be the protocol proposed by Arapinis et al., it holds that:

Lemma 1. *There exists a $(t, 1, 0, 0)$-narrow-weak client-unlinkability adversary $\mathcal{A}$ against Π running in time t, creating one session, corrupting no servers, and making no additional query to the related internal cryptographic functions, which wins with advantage $\frac{1}{2}$ (and probability 1) the client-unlinkability game.*

The van den Broek variant. Van den Broek et al. [9] recently proposed an IMSI catcher countermeasure; in this improved variant, avoid sending the IMSI in clear by replacing (IMSI, TMSI) tuples by an upgradeable pseudonym denoted PMSI. Their modified identification phase is exclusively done by means of these pseudonyms. The PMSI is chosen by the operator and sent with the authentication challenge in the preparation phase, encrypted together with the sequence number with a new secret key that is assumed to be shared by clients and their operators. The ciphertext is used as the random value R in the authentication challenge. Indeed, a successful session of the AKA protocol, ending in the establishment of new session keys, can only be attained if the PMSI is correctly updated. This variant is described in detail in [9].

From a practical point of view, using the operator at each key-exchange session is costly, and something that the original AKA design tries to avoid. Furthermore, though this variant successfully prevents IMSI catchers, it does not address client unlinkability. The pseudonym PMSI can be intercepted in one session; if this session is then aborted, the PMSI can be replayed in a second session, thus leading to user linkability. Furthermore, the protocol is vulnerable to the attack based on linking failure messages, as presented by Arapinis et al. Thus, if Π denotes the protocol proposed by van den Broek et al., it holds that:

Lemma 2. *There exists a $(t, 2, 1, 0)$-adversary $\mathcal{A}$ against the narrow-weak-client-unlinkability of Π running in time t, initiating two protocol sessions, and making no query to the internal cryptographic function G, which has an advantage $\mathsf{Adv}_{\Pi}^{\text{ww-unlink}}(\mathcal{A}) = \frac{1}{2}$ (and a probability of 1) to win the game.*

4 Our proposal: PrivAKA

The attacks described in the previous section emphasize the existing gap between what mobile AKE protocols *can* offer, and the security and privacy guarantees that they *should* offer. We consequently describe a new AKA variant which provides the security and privacy (in particular, wide-weak unlinkability) guarantees formalized in our security model.

We use the original AKA three-party infrastructure and take into account practical concerns, in particular minimizing the contact between servers and operators during the protocol executions.

4.1 Description of our variant

Instead of five phases, our variant only consists of three. Our protocol is designed to require no re-synchronization (phase 4 in AKA), and we include the TMSI reallocation (which was phase 5 in AKA) as part of the key-exchange phase (phase 3). In our construction, we use a public-key encryption scheme PKE= (PKE.KGen,PKE.Enc,PKE.Dec), such that each operator has a certified public and secret keypair denoted as $(\mathsf{pke}_{\mathsf{Op}}, \mathsf{ske}_{\mathsf{Op}})$. We assume that the client stores only its own operator's public key (and its certificate) internally. In particular, we do not give encryption keys to the servers in order to minimize key-management issues. We also use a secure authenticated encryption scheme $\mathsf{AE} = (\mathsf{AE.KGen}, \mathsf{AE.Enc}, \mathsf{AE.Dec})$. Though these can be instantiated generically, we use AES-GCM [7] for the AE scheme and e.g. Cramer-Shoup [6] for the PKE scheme. We depict our variant in Figure 6, and indicate in the grey boxes the differences to the classical AKA procedure.

We also refer the reader to Appendix C, which contains an evaluation of our modifications to the original AKA procedures; these are also summarily depicted in Figure 7.

Identification. Just as for AKA, we begin with an *identification* phase run by the client and the server over the insecure channel. The server sends an identification request which includes a random value that we denote R_{id}. The client computes a response depending on a flag $\mathsf{flag}_{\mathsf{TMSI}}$ managed by the client. If the user stays in the same LAI, $\mathsf{flag}_{\mathsf{TMSI}} := 0$ and the response is a fresh TMSI value generated in that area; else, if the user changes area, or if $\mathsf{flag}_{\mathsf{TMSI}} = 1$ (which happens e.g. upon aborts), it encrypts with the operator's public key an evaluation of $\mathcal{F}_5$ with the client key and the operator key, on input: the random value R_{id}, the client's IMSI, and an index $\mathsf{idx}_{\mathsf{C}}$, which is explained for the *Preparation* phase below. The client also appends the IMSI, R_{id}, and $\mathsf{idx}_{\mathsf{C}}$ inside the encryption. For both types of responses, the client also sends the identity of its operator Op.

Upon receiving a string of the form (m, Op), the server first checks whether the message m is a TMSI present in its database; if so, it retrieves the IMSI to which this value corresponds; else, it assumes that m is a ciphertext, and it sends it to the operator Op for decryption.

Intuitively, encrypting the IMSI prevents IMSI catcher attacks; moreover, even if all IMSI values are known, no MiM adversary can encrypt a valid identification response for a new R_{id}, since $\mathcal{F}_5$ is a PRF whose key is unknown to the adversary. In order to guarantee client unlinkability, however, we also require that TMSI values have the same length as the output of the PKE scheme. If this requirement is not fulfilled, then a distinguisher can use the length of the messages to single out one specific client e.g. one that moves into a new area. Whereas this type of attack seems less likely, it is easily feasible and highly usable in contexts of mass surveillance.

We design the TMSI reallocation in such a way as to ensure that if the client remains in the same LAI ($\mathsf{flag}_{\mathsf{TMSI}} = 0$), the server of that area can find the tuple $(\mathsf{TMSI}, \mathsf{IMSI})$ in its database. The fresh randomness R_{id} prevents replays, whereas the IND-CCA security of the PKE scheme makes encryption probabilistic and indistinguishable from random.

Preparation. This second phase is run over a secure channel, between the server and the operator. If the server received a valid TMSI in the previous phase, phase 2 begins with the server sending the corresponding IMSI to the operator; else, the server forwards the received ciphertext and the associated R_{id}. If the encryption verifies for the given IMSI and the fresh randomness, the operator prepares batches of authentication vectors which differ from the standard AKA output as follows:

- We associate each server with a unique identifier $\mathsf{Res}_{\mathsf{S}}$, which is used as input to each cryptographic function. This limits the use of server corruption attacks [19] only to the corrupted area. We also use the value AMF which is sent in clear, as an additional input.
- We use the sequence number $\mathsf{Sqn}_{\mathsf{Op,C}}$ as input to all functions except $\mathcal{F}_5$. This guarantees freshness for all the values even if the randomness R is repeated.
- We introduce an operator index $\mathsf{idx}_{\mathsf{Op,C}}$ to prevent replays of a challenge using an old sequence number. As our proofs indicate, this value is essential in preventing desynchronizations. The client also keeps track of a matching index, denoted $\mathsf{idx}_{\mathsf{C}}$, which is updated in terms of $\mathsf{idx}_{\mathsf{Op,C}}$. The client also updates this index whenever it aborts. We note that operators always begin by updating $\mathsf{idx}_{\mathsf{Op,C}}$ to the value $\mathsf{idx}_{\mathsf{C}}$ received in the identification string.

Authenticated key-exchange. The final, third phase of our protocol is run between the client and the server. We note beforehand that if any verification fails on the client side, the client aborts, updating its index $\mathsf{idx_C}$. The server sends a random value R, the authentication string Autn, and an authenticated encryption of the new TMSI *and* the index $\mathsf{idx_{Op,C}}$, using the session keys (CK, IK). The client proceeds as in the original AKA procedure, recovering AK, then checking $\mathsf{Mac_S}$. If the authentication cannot be verified, the procedure is aborted, the new TMSI is disregarded. Else, the user computes (CK, IK) and decrypts the authenticated encryption string to find the TMSI value and the operator's index $\mathsf{idx_{Op,C}}$. Then, C checks the freshness of the sequence number, i.e. it verifies if one of the following two conditions is correct:

– $\mathsf{Sqn_C} = \mathsf{Sqn}^{\{i\}}$.
– $\mathsf{Sqn_C} = \mathsf{inc}(\mathsf{Sqn}^{\{i\}})$ and $\mathsf{idx_{Op,C}} = \mathsf{idx_C} + 1$,

If the protocol is run normally, the first of these conditions is the one that will hold. However, if the previous session is aborted after receiving the server's authentication challenge, then the two sequence numbers may become desynchronized by one step (the second condition). At the next session, the server will re-use the authentication string, but with an updated index. Further desynchronization is prevented since sending the new TMSI requires fresh authentication (updated index). If the first condition holds, then the client's internal index is reset; else, the index is incremented by 1. The client updates the sequence number only upon successful authentication and if the first condition holds. If neither condition verifies, the procedure is aborted.

If the verification succeeds, the user computes a response $\mathsf{Res} := \mathcal{F}_2(\mathsf{keys}, \mathsf{R}^{\{i\}}, \mathsf{Sqn}^{\{i\}}, \mathsf{Res_S}, \mathsf{AMF})$, sends Res, then stores the TMSI and the new index value. The server checks Res against the prepared value $\mathsf{Mac_C}$ (else, if no response is received, the procedure is aborted).

One notable exception to the original AKA protocol is that whenever an abort occurs on the server's side, the second phase – *preparation* – is used instead of simply querying the next vector in the prepared batch. Though this might seem more inefficient, we note that an abort only occurs in the presence of an adversary, which is considered to be a rare event. We detail the procedure upon aborts in Figure 4.

Cryptographic algorithms. In our variant, we modified the underlying cryptographic functions to also take as input the sequence number and the new value $\mathsf{Res_S}$. In the full paper [10], we show how to modify the two algorithm suites, TUAK and MILENAGE, to take this into account. Our variant moreover no longer needs functions $\mathcal{F}_1^*, \mathcal{F}_5^*$, which were used for resynchronizations.

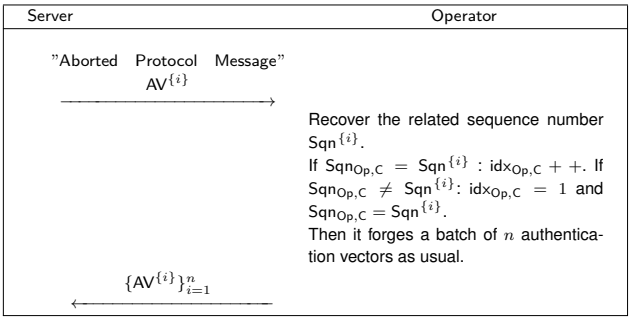


Fig. 4. Procedure after an abort.

4.2 Privacy and Security Analysis

In this section, we outline the security properties of the AKA variant. Due to space restrictions, we only include the theorems, deferring the proofs to the full paper [10]. We replace all calls to any of the cryptographic algorithms $\mathcal{F}_1, \ldots, \mathcal{F}_5$ by calls to a pseudorandom function G, which is keyed with $\mathsf{sk_C}$ (except for the state-confidentiality property, see below). This is intuitively possible for both known instantiations of the algorithms suites (MILENAGE and TUAK), since these functions are meant to produce independent output.

The following five statements are proved under some properties (indistinguishability, pseudorandomness and unforgeability) of the internal cryptographic functions. Notably, for the key-confidentiality property, we require the pseudorandomness of the function G when keyed with $\mathsf{sk_C}$ **and** when keyed with the operator key $\mathsf{sk_{op}}$. The former property can be proved for both MILENAGE and TUAK (see [10]), but the latter does not hold for MILENAGE.

We formulate security with respect to an adversary $\mathcal{A}$ which runs in time t, creates at most q_{exec} party instance, with at most q_{id} user identification per instance, corrupts at most q_{serv} servers, makes at most q_{Op} OpAccess queries per operator per corrupted server, and at most q_G, q_{AE}, q_{PKE} queries to respectively the functions G, AE and PKE. For the *legitimate-and-malicious* adversary, we quantify $\mathcal{A}$ in terms of the maximal number q_{Op} of queries to the oracle OpAccess, and the similar q_{exec}, q_{id}, q_G, q_{AE}, q_{PKE} queries. The function G is defined as above.

Theorem 1 (ww-unlink − K.Ind − C.Imp − S.Imp.)**.** *For the protocol* Π *using the unitary function* G *described above, the following properties hold:*

ww-unlink*: For any* $(t, q_{\mathsf{exec}}, q_{\mathsf{id}}, q_G, q_{\mathsf{AE}}, q_{\mathsf{PKE}})$*-adversary* $\mathcal{A}$ *against the weak privacy* ww-unlink*-security of the protocol* Π *winning with advantage* $\mathsf{Adv}_{\Pi}^{\text{ww-unlink}}(\mathcal{A})$*, there exist* $(t' \sim O(t), q' = 2 \cdot q_{\mathsf{exec}} + q_G)$*-*$\mathcal{A}_1$ *against the pseudorandomness of the function* G*, an* $(t'' \sim O(t), q'' = q_{\mathsf{exec}} + q_{\mathsf{AE}})$*-adversary* $\mathcal{A}_2$ *against the ae-security of the*

function AE, $(t''' \sim O(t), q''' = q_{\mathsf{exec}} \cdot q_{\mathsf{id}} + q_{\mathsf{PKE}})$-*adversaries* $\mathcal{A}_3$ *against the indistinguishability of the function* PKE, $(t, q_{\mathsf{exec}}, q_{\mathsf{id}}, 0, 0, q_G, q_{\mathsf{AE}}, q_{\mathsf{PKE}})$-*adversary* $\mathcal{A}_4$ *against the key-indistinguishability of the protocol* Π *such that:*

$$\mathsf{Adv}^{\mathsf{ww\text{-}unlink}}_{\Pi}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{K.Ind}}_{\Pi}(\mathcal{A}_4) + 2 \cdot \frac{1 + (q_{\mathsf{exec}} \cdot q_{\mathsf{id}})^2}{2^{|\mathsf{TMSI}|}} + \frac{q_{\mathsf{exec}}^2}{2^{|\mathsf{R}|}} + \frac{(q_{\mathsf{exec}} \cdot q_{\mathsf{id}})^2}{2^{|\mathsf{R_{id}}|}} + n_{\mathsf{C}} \cdot \big(10 \cdot \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_1) + 5 \cdot \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}_3)\big).$$

K.Ind*:* *For any* $(t, q_{\mathsf{exec}}, q_{\mathsf{id}}, q_{\mathsf{serv}}, q_{\mathsf{Op}}, q_G, q_{\mathsf{AE}}, q_{\mathsf{PKE}})$-*adversary* $\mathcal{A}$ *against the key-indistinguishability of* Π *winning with advantage* $\mathsf{Adv}^{\mathsf{K.Ind}}_{\Pi}(\mathcal{A})$*, there exist a* $(t' = O(t), q' = q_G + 2 \cdot q_{\mathsf{exec}} + 5 \cdot q_{\mathsf{serv}} \cdot q_{\mathsf{Op}})$-*prf-adversary* $\mathcal{A}_1$ *on* G*, a* $(t' = O(t), q' = q_{\mathsf{exec}} + q_{\mathsf{AE}})$-*ae-adversary* $\mathcal{A}_2$ *on* AE*, and a* $(t' = O(t), q' = q_{\mathsf{exec}} \cdot q_{\mathsf{id}} + q_{\mathsf{PKE}})$-*ind-cca2-adversary* $\mathcal{A}_3$ *on* PKE *such that:*

$$\mathsf{Adv}^{\mathsf{K.Ind}}_{\Pi}(\mathcal{A}_{\mathbb{G}_0}) \leq n_{\mathsf{C}} \cdot \Big(\frac{(q_{\mathsf{exec}} \cdot q_{\mathsf{id}})^2}{2^{|\mathsf{R_{id}}|}} + \frac{(q_{\mathsf{exec}} + q_{\mathsf{serv}} \cdot q_{\mathsf{Op}})^2}{2^{|\mathsf{R}|}} + 2\mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}_3)\Big).$$

C.Imp*:* *For any* $(t, q_{\mathsf{exec}}, q_{\mathsf{id}}, q_{\mathsf{serv}}, q_{\mathsf{Op}}, q_G, q_{\mathsf{AE}}, q_{\mathsf{PKE}})$-*adversary* $\mathcal{A}$ *against the* C.Imp*-security of* Π *winning with advantage* $\mathsf{Adv}^{\mathsf{C.Imp}}_{\Pi}(\mathcal{A})$*, there exist a* $(t' \approx O(t), q' = q_G + 2 \cdot q_{\mathsf{exec}} + 5 \cdot q_{\mathsf{serv}} \cdot q_{\mathsf{Op}})$-*prf-adversary* $\mathcal{A}_1$*, a* $(t' = O(t), q' = q_{\mathsf{exec}} + q_{\mathsf{AE}})$-*ae-adversary* $\mathcal{A}_2$ *on* AE *and a* $(t' = O(t), q' = q_{\mathsf{exec}} \cdot q_{\mathsf{id}} + q_{\mathsf{PKE}})$-*ind-cca2-adversary* $\mathcal{A}_3$ *on* PKE *such that:*

$$\mathsf{Adv}^{\mathsf{C.Imp}}_{\Pi}(\mathcal{A}_{\mathbb{G}_0}) \leq n_{\mathsf{C}} \cdot \Big(\mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}_3) + \frac{(q_{\mathsf{exec}} \cdot q_{\mathsf{id}})^2}{2^{|\mathsf{R_{id}}|}} + \frac{(q_{\mathsf{exec}})^2}{2^{|\mathsf{R}|}} + \frac{q_{\mathsf{exec}}}{2^{|\mathsf{Res}|}} + \frac{1}{2^{\kappa}} + \frac{q_{\mathsf{exec}} \cdot q_{\mathsf{id}}}{2^{|\mathsf{ID}|}} \Big).$$

S.Imp*:* *For any* $(t, q_{\mathsf{exec}}, q_{\mathsf{id}}, q_{\mathsf{serv}}, q_{\mathsf{Op}}, q_G, q_{\mathsf{AE}}, q_{\mathsf{PKE}})$-*adversary* $\mathcal{A}$ *against the* S.Imp*-security of* Π*winning with advantage* $\mathsf{Adv}^{\mathsf{S.Imp}}_{\Pi}(\mathcal{A})$*, there exist a* $(t' \approx O(t), q' = q_G + 2 \cdot q_{\mathsf{exec}} + 5 \cdot q_{\mathsf{serv}} \cdot q_{\mathsf{Op}})$-*prf-adversary* $\mathcal{A}_1$*, a* $(t' = O(t), q' = q_{\mathsf{exec}} + q_{\mathsf{AE}})$-*ae-adversary* $\mathcal{A}_2$ *on* AE *and a* $(t' = O(t), q' = q_{\mathsf{exec}} \cdot q_{\mathsf{id}} + q_{\mathsf{PKE}})$-*ind-cca2-adversary* $\mathcal{A}_3$ *on* PKE *such that:*

$$\mathsf{Adv}^{\mathsf{S.Imp}}_{\Pi}(\mathcal{A}_{\mathbb{G}_0}) \leq n_{\mathsf{C}} \cdot \big(\frac{q_{\mathsf{exec}}}{2^{|\mathsf{Mac_S}|}} + \frac{1}{2^{\kappa}} + \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}_3)\big).$$

S.sound*:* *For any* $(t, q_{\mathsf{exec}}, q_{\mathsf{Op}}, q_G, q_{\mathsf{AE}}, q_{\mathsf{PKE}})$-*adversary* $\mathcal{A}$ *against the* S.sound*-security of the protocol* Π*,winning with advantage* $\mathsf{Adv}^{\mathsf{S.sound}}_{\Pi}(\mathcal{A})$*, there exist a* $(t' \approx O(t), q' = q_G + 2 \cdot q_{\mathsf{exec}} + 5 \cdot q_{\mathsf{Op}})$-*prf-adversary* $\mathcal{A}_1$ *on* G*, a* $(t' = O(t), q' = q_{\mathsf{exec}} + q_{\mathsf{AE}})$-*ae-adversary* adv_2 *on* AE *and a* $(t' = O(t), q' = q_{\mathsf{exec}} + q_{\mathsf{PKE}})$-*ind-cca2-adversary* $\mathcal{A}_3$ *on* PKE *such that:*

$$\mathsf{Adv}^{\mathsf{S.sound}}_{\Pi}(\mathcal{A}) \leq n_{\mathsf{C}} \cdot \big(\frac{q_{\mathsf{exec}}}{2^{|\mathsf{Mac_S}|}} + \frac{1}{2^{\kappa}} + \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}_3)\big).$$

Theorem 2 (St.conf − **resistance.**)**.** *For any* $(t, q_{\mathsf{exec}}, q_{\mathsf{id}}, q_{\mathsf{Op}}, q_G, q_{G^*}, q_{\mathsf{AE}}, q_{\mathsf{PKE}})$-*adversary* $\mathcal{A}$ *against the* St.conf*-security of the protocol* Π*, winning with advantage* $\mathsf{Adv}^{\mathsf{St.conf}}_{\Pi}(\mathcal{A})$*, there exist a* $(t' \approx O(t), q' = q_G + 5 \cdot q_{\mathsf{Op}} + 2 \cdot q_{\mathsf{exec}})$-*prf-adversary* $\mathcal{A}_1$ *on* G*, a* $(t' = O(t), q' = q_{\mathsf{exec}} + q_{\mathsf{AE}})$-*ae-adversary* $\mathcal{A}_2$ *on* AE*, a* $(t' = O(t), q' = q_{\mathsf{exec}} \cdot q_{\mathsf{id}} + q_{\mathsf{PKE}})$-*ind-cca2-adversary* $\mathcal{A}_3$ *on* PKE*, and* $(t' \approx O(t), q' = q_{G^*})$-*prf-adversary* $\mathcal{A}_4$ *on* G^* *such that:*

$$\mathsf{Adv}^{\mathsf{St.conf}}_{\Pi}(\mathcal{A}) \leq n_{\mathsf{C}} \cdot \Big(2^{-|\mathsf{sk_C}|} + 2^{-|\mathsf{sk_{op}}|} + 2 \cdot 2^{-|\mathsf{Sqn_C}|} + \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}_3) + \mathsf{Adv}^{\mathsf{prf}}_{G^*}(\mathcal{A}_4)\Big).$$

MILENAGE **and** TUAK **as** G**.** In our full paper [10], we prove that the calls to both our updated MILENAGE and TUAK algorithms can be modeled as the unitary function G that we use for our proofs. The last step in our proof is to show that both algorithm suites exhibit the PRF property we require for G, when instantiated with the key $\mathsf{sk_C}$. However, as opposed to TUAK (whose symmetric design allows a lot more leeway), the MILENAGE algorithms do not have the PRF property when G is used with key $\mathsf{sk_{op}}$.

4.3 Narrow-Forward Privacy is Impossible

Our variant of AKA preserves the structure of the original protocol, but also provably attains wide-weak client unlinkability. In this section we show that this degree of client-unlinkability is optimal with respect to the structure of AKA. In particular, narrow-forward privacy is impossible.

Our result covers similar ground as that by Paise and Vaudenay [24], as we address protocols with mutual authentication. We extend the impossibility result to symmetric-key AKE protocols which also use public-key primitives. We also explain why the original impossibility result in [24] is imprecise, and presents some problems.

The result of [24]. Paise and Vaudenay showed an impossibility result for *authentication* (rather than AKE) protocols, but the extension to AKE is easy. In the terminology of our paper, [24] proved that server-authentication essentially precludes narrow-forward client-unlinkability. Their attack follows these steps: (1) the adversary $\mathcal{A}$ creates two clients; (2) $\mathcal{A}$ runs an honest protocol session between one of them (chosen uniformly at random depending on a secret bit b) and the server, but stops the last message from the server to the client; (3) $\mathcal{A}$ corrupts both clients, learning their long-term state; (4)

# bits of sec.	Nonces (R_{ID}, R)	Keys (sk, sk_{op})	IMSI	$\mathcal{F}_i$ output	Index idx	TMSI	Sqn
64	217 / 345	128 / 128	50 / 50	128 / 153	32 / 64	427 / 587	89 / 89
128	281 / 409	153 / 153	50 / 50	185 / 217	32 / 64	520 / 680	153 / 153
256	409 / 537	281 / 281	50 / 50	313 / 345	32 / 64	776 / 936	281 / 281

Fig. 5. Parameter sizes (in bits) for our proposal with at most 20 million clients per operator; for each entry, the left value indicates the size for a maximum of 2^{32} adversarial queries of each type, whilst the right figure is calculated for 2^{64} queries.

$\mathcal{A}$ distinguishes between the clients by simulating the protocol with the intercepted message.

However, this attack makes a tacit assumption on the client's behaviour, namely that if a session is aborted, the state is either not updated or it is updated in a consistent way depending on the client's internal state. Say that upon an abort, the client reverts to a random state; assumming the adversary cannot use the very short time-frame before reverting to random, $\mathcal{A}$ only gets the random state in response. Simulating the protocol with the received message will not match that state, thus reducing the adversary's success probability to $\frac{1}{2}$.

Another way to bypass this result is to update the client state before the "last message" is sent to the client; if such an update is done at every execution, the attack presented in [24] fails. This is, however, a rather artificial twist: indeed, mutual authentication implies that the prover *must* somehow identify the server's state as "valid" *before* it reaches a state which precludes it from verifying the server's authentication.

Two new attacks. In the AKA protocol, it is the server which first authenticates to the client. The values used in authentication are the sequence number $\mathsf{Sqn}_{\mathsf{Op,C}}$ and the nonce R. The value $\mathsf{Sqn}_{\mathsf{Op,C}}$ is ephemeral, being updated at every session; however, it is a long-term state, and compatible with the client's own state $\mathsf{Sqn}_{\mathsf{C}}$. In particular, corrupting a client yields $\mathsf{Sqn}_{\mathsf{C}}$ allowing the verifier to link the client with the corresponding $\mathsf{Sqn}_{\mathsf{Op,C}}$ value.

For a better comprehension of our attacks and their impact, we define the following notations: we divide a party's state (for both clients and operators) into a *static* state $\mathsf{stat.st}_{\mathsf{P}}$ and an ephemeral state $\mathsf{eph.st}_{\mathsf{P}}$. Thus, an operator's static state may contain operator-specific information, such as the secret key for a PKE scheme, but it will also include state shared with clients, i.e. $\mathsf{stat.st}_{\mathsf{Op,C}}$ for a client C. The same for the ephemeral state $\mathsf{eph.st}_{\mathsf{Op,C}}$ which for the AKA protocol consists of the sequence number $\mathsf{Sqn}_{\mathsf{Op,C}}$.

We propose the following attack:

- The adversary $\mathcal{A}$ creates two clients C and C′ with the same operator Op and the same location LAI.
- $\mathcal{A}$ uses DrawCl on C, C′ and receives the handle VC, corresponding to either C (if the hidden bit $b = 0$), or C′ (otherwise).
- $\mathcal{A}$ runs an honest execution between the server S at LAI and the client VC until $\mathcal{A}$ receives the message $\mathsf{R}, \mathsf{Autn} = (\mathsf{Sqn}_{\mathsf{Op,C}} \oplus \mathsf{AK}) \| \mathsf{AMF} \| \mathsf{Mac}_{\mathsf{S}}$ from the server. Denote $\mathsf{Autn}[1] := \mathsf{Sqn}_{\mathsf{Op,C}} \oplus \mathsf{AK}$, $\mathsf{Autn}[2] = \mathsf{AMF}$, and $\mathsf{Autn}[3] := \mathsf{Mac}_{\mathsf{S}}$.
- $\mathcal{A}$ corrupts C and learns $\mathsf{Sqn}_{\mathsf{C}}$, sk_{C}, and $\mathsf{sk}_{\mathsf{op}}$.
- By using the values $\mathsf{R}, \mathsf{sk}_{\mathsf{C}}$, and $\mathsf{sk}_{\mathsf{op}}$, the adversary computes a value AK_{C} and retrieves $\mathsf{Sqn}^* := \mathsf{Autn}[1] \oplus \mathsf{AK}_{\mathsf{C}}$. Note that if $b = 0$, then $\mathsf{AK}_{\mathsf{C}} = \mathsf{AK}$ and $\mathsf{Sqn}^* = \mathsf{Sqn}_{\mathsf{Op,C}}$, while if $b = 1$, then $\mathsf{Sqn}^* \neq \mathsf{Sqn}_{\mathsf{Op,C'}}$ with overwhelming probability.
- The adversary verifies $\mathsf{Mac}_{\mathsf{S}}$, on input $(\mathsf{sk}_{\mathsf{C}}, \mathsf{sk}_{\mathsf{op}}, \mathsf{Sqn}^*)$. If this verification succeeds, the adversary outputs a guess $d = 0$ for the bit b; else, it outputs 1.

For the analysis note that with overwhelming probability $\mathsf{Mac}_{\mathsf{S}}$ will not verify if it was computed for C′, i.e. $f_1(\mathsf{sk}_{\mathsf{C'}}, \mathsf{sk}_{\mathsf{op}}, \mathsf{R}, \mathsf{AMF}, \mathsf{Sqn}_{\mathsf{Op,C'}}) \neq f_1(\mathsf{sk}_{\mathsf{C}}, \mathsf{sk}_{\mathsf{op}}, \mathsf{R}, \mathsf{AMF}, \mathsf{Sqn}^*)$. The key vulnerability here is that, while $\mathsf{Sqn}_{\mathsf{Op,C}}$ is never sent in clear, the masking authentication key AK only depends on the client's static state. We also use the fact that the validity of the sequence number is confirmed by the value $\mathsf{Mac}_{\mathsf{S}}$.

While using the $\mathsf{Mac}_{\mathsf{S}}$ verification certainly helps an attacker, our second attack (a variation of the first one) does not use the MAC value at all. The attack is run exactly in the same way, until we reach the final step. At that point:

- $\mathcal{A}$ compares the obtained value Sqn^* with the recovered sequence number of C, namely $\mathsf{Sqn}_{\mathsf{C}}$, verifying if $|\mathsf{Sqn}^* - \mathsf{Sqn}_{\mathsf{Op,C}}| \leq \Delta$. Note that in the actual attack presented above, the client's state $\mathsf{Sqn}_{\mathsf{C}}$ should be exactly equal to the operator's state with respect to that client; however, our attack is even stronger in the sense that we do not need to control the executions of the protocol in order to obtain exact equality.

Analysis and Impact. Since the original AKA protocol is not even weak-client-unlinkable, it is not surprising that this protocol is not narrow-forward unlinkable either. However, the same attack works on our variant of the protocol and indeed, on any other extension or improvement of the original procedure which retains the characteristic of exchanging a message

of the type $f(\mathsf{eph.st}_{\mathsf{Op,C}}, \mathsf{stat.st}_{\mathsf{Op,C}}, X)$ in the presence of a function $Match$; or exchanging that same message together with a message $g(\mathsf{eph.st}_{\mathsf{Op,C}}, \mathsf{stat.st}_{\mathsf{Op,C}}, Y)$, such that:

- f is reversible and takes as input $\mathsf{eph.st}_{\mathsf{Op,C}}, \mathsf{stat.st}_{\mathsf{Op,C}} = \mathsf{stat.st}_{\mathsf{C}}$, and a set X of publicly-known variables, giving arbitrary values in the set $\{0,1\}^*$[7];
- $Match$ takes as input two ephemeral state values $\mathsf{eph.st}_{\mathsf{C}'}$ and $\mathsf{eph.st}_{\mathsf{Op,C}}$ and it outputs a boolean value: 1 if $\mathsf{C} = \mathsf{C}'$ and 0 otherwise[8];
- g takes as input the state values $\mathsf{eph.st}_{\mathsf{Op,C}}, \mathsf{stat.st}_{\mathsf{C}}$ and a set Y of public values, and which has the property that, for randomly chosen x and $\mathsf{stat.st}_{\mathsf{C}'}$ it holds that $g(x, \mathsf{stat.st}_{\mathsf{Op,C}'}, Y) \neq g(\mathsf{eph.st}_{\mathsf{Op,C}}, \mathsf{stat.st}_{\mathsf{Op,C}}, Y)$[9].

5 Practical considerations

In this section, we discuss some of our design choices for the improvement we propose of the AKA protocol. We also provide a detailed analysis of our countermeasures and their intuitive effects in Appendix C.

As opposed to the proposal of van den Broek et al. [9], we opted to continue using (TMSI, LAI) tuples in the identification phase. This infrastructure is maintained strictly by servers, with no operator contribution; thus it is efficient and inexpensive. Moreover, TMSI values and their correspondence to the client's IMSI is easy to find. In our proposal, we bypass IMSI catcher attacks by never sending IMSIs in clear, and we add a symmetric authentication step in the encryption, thus precluding the client-unlinkability attack we found against [22]. We use an IND-CCA public-key encryption scheme with a minimal, operator-only PKI. A client only stores the public key (and certificate) of its own operator, thus minimizing key-management problems. For the TMSI reallocation, we add an implicit authentication, preventing IMSI paging and DoS attacks. We also add a freshness index, which prevents replays of challenges based on old sequence numbers.

Our variant, however, can *only* guarantee client-unlinkability if the *size* of the TMSI is equal to that of the output of the PKE scheme in the first identification message. This is a non-trivial requirement as servers must keep track of all the TMSIs they issue; while using a shorter TMSI does not leak anything about the IMSI value, it does allow mass-surveillance organisms to track users down by distinguishing between the length of the encrypted IMSI as opposed to the TMSI length. On the positive side, servers may store TMSI values for a shorter while, since as soon as the user leaves the area, the TMSI becomes obsolete.

In Figure 5 we indicate minimal parameter sizes for our variant, for at most 20 million clients (per operator). For the TMSI, we used the best-case scenario, in which the PKE scheme output equals the plaintext length[10]. It is clear in Figure 5 that some parameters, such as the size of the IMSI and the sequence number, do not depend on the number of queries, as also indicated in our security theorem. Some values are indirectly affected by the number of queries: for instance, the output of the underlying functions is lower-bounded by the requirement of the size of Res, i.e., the client's response, output by $\mathcal{F}_2$, which in turn depends on the number of queries.

We assume that clients are aware of their current LAI, thus avoiding being traced by their location. This is not a very strong assumption, since mobile devices are often equipped to detect the LAI. Finally, we bypass distinguishing attacks that exploit the re-synchronization phase by ensuring that sequence numbers cannot be desynchronized. To minimize server-operator communication, our variant ensures that operators are contacted only if the protocol is abnormally run or an adversary is detected. We also simplify the AKA structure, including only three communication phases rather than five.

References

[1] 3GPP. 3G Security; Technical Specification Group (TSG) SA; 3G Security; Security Architecture. TS 33.102, 3rd Generation Partnership Project (3GPP), June 2013.

[2] 3GPP. 3rd Generation Partnership Project; Technical Specification Group Services ans System Aspects; Security related network functions (Release 12). TS 43.020, 3rd Generation Partnership Project (3GPP), June 2014.

[3] J. Alwen, M. Hirt, U. Maurer, A. Patra, and P. Raykov. Anonymous authentication with shared secrets. In *Proceedings of LatinCrypt*, volume 8895 of *LNCS*, pages 219–236. Springer-Verlag, 1999.

[4] G. Ateniese, A. Herzberg, H. Krawczyk, and G. Tsudik. Untraceable mobility or how to travel *incognito*. In *Elsevier Computer Networks*, volume 31, pages 871–884. Elsevier, 1999.

[5] BSI. A Proposal for: Functionality classes for random number generators. AIS 20 / AIS 31. Version 2.0 , Bundesamt fur

7 In our previous example, this is the string Autn1, which depends on $\mathsf{eph.st}_{\mathsf{Op,C}} = \mathsf{Sqn}_{\mathsf{Op,C}}$, on $\mathsf{stat.st}_{\mathsf{Op,C}} = (\mathsf{sk}_{\mathsf{C}}, \mathsf{sk}_{\mathsf{op}})$, and on the random value R which is public.

8 In our case, the $Match$ function returns 1 if and only if $|\mathsf{Sqn}_{\mathsf{Op,C}} - \mathsf{Sqn}_{\mathsf{C}'}| \leq \Delta$.

9 In our example, this function is f_1, and the output value is $\mathsf{Mac}_{\mathsf{S}}$.

10 This is an optimistic estimate, as the ciphertext-to-plaintext ratio is mostly higher than 1 (for ElGamal, it is 2). However, once the PKE scheme is chosen, the TMSI size is computable as the value we give times the ciphertext-to-plaintext length ratio.

Sichercheit in der Informationstechnik (BSI), 2011.
[6] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attacks. In *Advances in Cryptology – CRYPTO 1998*, volume 1462 of *LNCS*, pages 13–25. Springer, 1998.
[7] David A. McGrew and John Viega. The Security and Performance of the Galois/Counter Mode of Operation (Full Version). *IACR Cryptology ePrint Archive*, 2004:193, 2004.
[8] D.Strobel. IMSI Catcher. In *2007, Seminar Work, Ruhr-Universitat Bochum*, 2007.
[9] Fabian van den Broek and Roel Verdult and Joeri de Ruiter. Defeating IMSI Catchers. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, USA, October 12-6, 2015*, pages 340–351, 2015.
[10] P. A. Fouque, C. Onete, and B. Richard. Achieving Better Privacy for the 3GPP AKA Protocol. Cryptology ePrint Archive, Report 2001/112, 2016.
[11] Jens Hermans and Andreas Pashalidis and Frederik Vercauteren and Bart Preneel. A New RFID Privacy Model. In *Computer Security - ESORICS 2011 - 16th European Symposium on Research in Computer Security, Leuven, Belgium, September 12-14, 2011. Proceedings*, pages 568–587, 2011.
[12] Jens Hermans and Andreas Pashalidis and Frederik Vercauteren and Bart Preneel. A New RFID Privacy Model. In V. Atluri and C. Diaz, editors, *Esorics*, volume 6879, pages 568–587, 2011.
[13] M. S. A. Khan and C. J. Mitchell. Another look at privacy threats in 3G mobile telephony. In *Proceedings of ACISP*, volume 8544 of *Lecture Notes in Computer Science*, pages 386–396. Springer, 2014.
[14] Michael Burrows and Martín Abadi and Roger M. Needham. A Logic of Authentication. *ACM Trans. Comput. Syst.*, 8(1):18–36, 1990.
[15] Mihir Bellare and David Pointcheval and Phillip Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. In *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques*, pages 139–155, 2000.
[16] Mihir Bellare and Phillip Rogaway. Entity Authentication and Key Distribution. In D. R. Stinson, editor, *Advances in Cryptology - CRYPTO '93*, volume 773 of *LNCS*, pages 232–249. Springer, 1993.
[17] Mihir Bellare and Ran Canetti and Hugo Krawczyk. A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols. In *Proceedings of the ACM Symposium on the Theory of Computing*, pages 419–428, 1998.
[18] Ming-Feng Lee and Nigel P. Smart and Bogdan Warinschi and Gaven J. Watson. Anonymity guarantees of the UMTS/LTE authentication and connection protocol. *Int. J. Inf. Sec.*, 13(6):513–527, 2014.
[19] Muxiang Zhang. Provably-Secure Enhancement on 3GPP Authentication and Key Agreement Protocol. *IACR Cryptology ePrint Archive*, 2003:92, 2003.
[20] Muxiang Zhang and Yuguang Fang. Security analysis and enhancements of 3gpp authentication and key agreement protocol. *IEEE Transactions on Wireless Communications*, 4(2):734–742, 2005.
[21] Myrto Arapinis and Loretta Ilaria Mancini and Eike Ritter and Mark Ryan. Privacy through Pseudonymity in Mobile Telephony Systems. In *21st Annual Network and Distributed System Security Symposium, NDSS*, 2014.
[22] Myrto Arapinis and Loretta Ilaria Mancini and Eike Ritter and Mark Ryan and Nico Golde and Kevin Redon and Ravishankar Borgaonkar. New privacy issues in mobile telephony: fix and verification. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 205–216, 2012.
[23] S. provider. Personal communication with one of europe's largest service providers, 2015.
[24] Radu-Ioan Paise and Serge Vaudenay. Mutual Authentication in RFID: Security and Privacy. In *Proc. on the 3^{rd} ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, pages 292–299. ACM, 2008.
[25] Ran Canetti and Hugo Krawczyk. Universally Composable Notions of Key Exchange and Secure Channels. In *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 337–351, 2002.
[26] Serge Vaudenay. On Privacy Models for RFID. In *ASIACRYPT '07*, volume 4883, pages 68–87, 2007.
[27] A. Shaik, R. Borgaonkar, N. Asokan, V. Niemi, and J.-P. Seifert. Practical attacks against privacy and availability in 4g/lte mobile communication systems. In *Proceedings of NDSS*. Internet Society, 2016.
[28] Ulrike Meyer and Susanne Wetzel. A man-in-the-middle attack on UMTS. In *Proceedings of the 2004 ACM Workshop on Wireless Security, Philadelphia, PA, USA, October 1, 2004*, pages 90–97, 2004.
[29] Zahra Ahmadian and Somayeh Salimi and Ahmad Salahi. New attacks on UMTS network access. In *2009 Wireless Telecommunications Symposium, WTS 2009, Prague, Czech Republic, April 22-24, 2009*, pages 1–6, 2009.

A A full review of related work

Though initially the designers of AKA provided a security proof using BAN logic [14], subsequent vulnerabilities belied the purported guarantees. The attacks of [28, 29] indicate that servers can be impersonated within the protocol run; the second paper also indicates that well-known weaknesses of the GSM protocol, e.g., weak encryption and the lack of mutual authentication, can pose security problems for AKA. Zhang and Fang [20] pointed out that the use of sequence numbers, together with potential corrupted server redirection, allow attackers to illicitly trace and impersonate clients.

A known *privacy* problem of the AKA protocol is the "IMSI catcher" attack [8], which exploits the fact that the permanent identifier IMSI is sent as a back-up for a faulty tuple $(\mathsf{TMSI}, \mathsf{LAI})$. By either observing such faulty behaviour (due to transmission errors or to server database problems), or by causing it, MiM attackers can easily track clients, in violation

of the user-identity confidentiality requirement. This is a problem in all generations of mobile communication networks. Van den Broek et al. [9] addressed IMSI catchers by replacing IMSI values by unlinkable pseudonyms PMSI. We discuss this variant in detail in Section 3.2.

Arapinis et al. [22] showed that failure messages (in authentication/resynchronization) can be used to trace users. A follow-up paper [21] identifies ways in which a specific implementation deviates from 3GPP recommendations, enabling the linkability of client sessions.

Though several improvements of AKA have been proposed, only two analyse the resulting security. Thus, [19] describes AP-AKA, a stateless protocol which can thwart replay-based impersonation and lowers the impact of server corruptions, but at the cost of no privacy at all. A second variant [22] retains the stateful nature of AKA, but model (in the proof) the state as random. We show some weaknesses of [22] in Section 3.2.

Secure AKE models. Bellare and Rogaway first proposed a security model for AKE [16], also in a symmetric setting. Their framework was later extended with the contribution of Pointcheval [15]. A further model for generic session-oriented protocols was proposed in [17] and extended in [25]. Although we use BPR-methodologies in our analysis, we cannot simply "import" their model, and use a slightly modified version thereof.

Privacy models. The privacy model due to Vaudenay [26] was the first to define several degrees of untraceability in the presence of corruptions. Though this framework was specifically designed for RFID privacy, his approach features attacks which are universal in authentication scenarios. In particular, Vaudenay captured adaptive corruptions, and classified adversaries in terms of their behaviour upon corruption. The AKA protocol is all the more adapted to his framework since it uses only symmetric keys, as does RFID authentication.

His framework was later refined and extended by Hermans et al. [12], who used Vaudenay's classification of adversaries according to two criteria: (i) whether the adversary is aware of the result of authentication sessions (wide adversaries) or not (narrow adversaries); (ii) any constraints in the corruption behaviour (the adversaries range from weak –no corruptions– to forward –corruptions "end" the game– and strong). In our work, we use the game-based definition of Hermans et al.

Two important impossibility results indicate that strong privacy *requires* key exchange [26] and that in symmetric-key authentication protocols, if the server authenticates to the client during the protocol, there exists an attack that allows an adversary to distinguish between two clients [24]. In particular, simply dropping the last message of the server to one of two clients, then corrupting the clients, and finally simulating them with the and dropped message lets the adversary distinguish between those clients. This attack, however, tacitly assumes that (a) clients do not "reset" if the authentication session is aborted, or alternatively that (b) the adversary can actually corrupt the clients before they time out.

B Proof Sketches

We prove each security statement in several game hops:

Proof of ww-unlink.

$\mathbb{G}_0$: the ww-unlink-game stipulated in Section 2.

$\mathbb{G}_0 \Rightarrow \mathbb{G}_1$**:** In $\mathbb{G}_1$, we abort if two honest server instances output the same random values R or $\mathsf{R_{id}}$. The two games are equivalent up to a collision term $\frac{(q_{\mathsf{exec}}+q_{\mathsf{serv}}\cdot q_{\mathsf{Op}})^2}{2^{|\mathsf{R}|}} + \frac{(q_{\mathsf{exec}}q_{\mathsf{id}})^2}{2^{|\mathsf{R_{id}}|}}$ (note that $\mathsf{R_{id}}$ is chosen by the server, whereas R is chosen by the operator, and can be batch-queried on corruption).

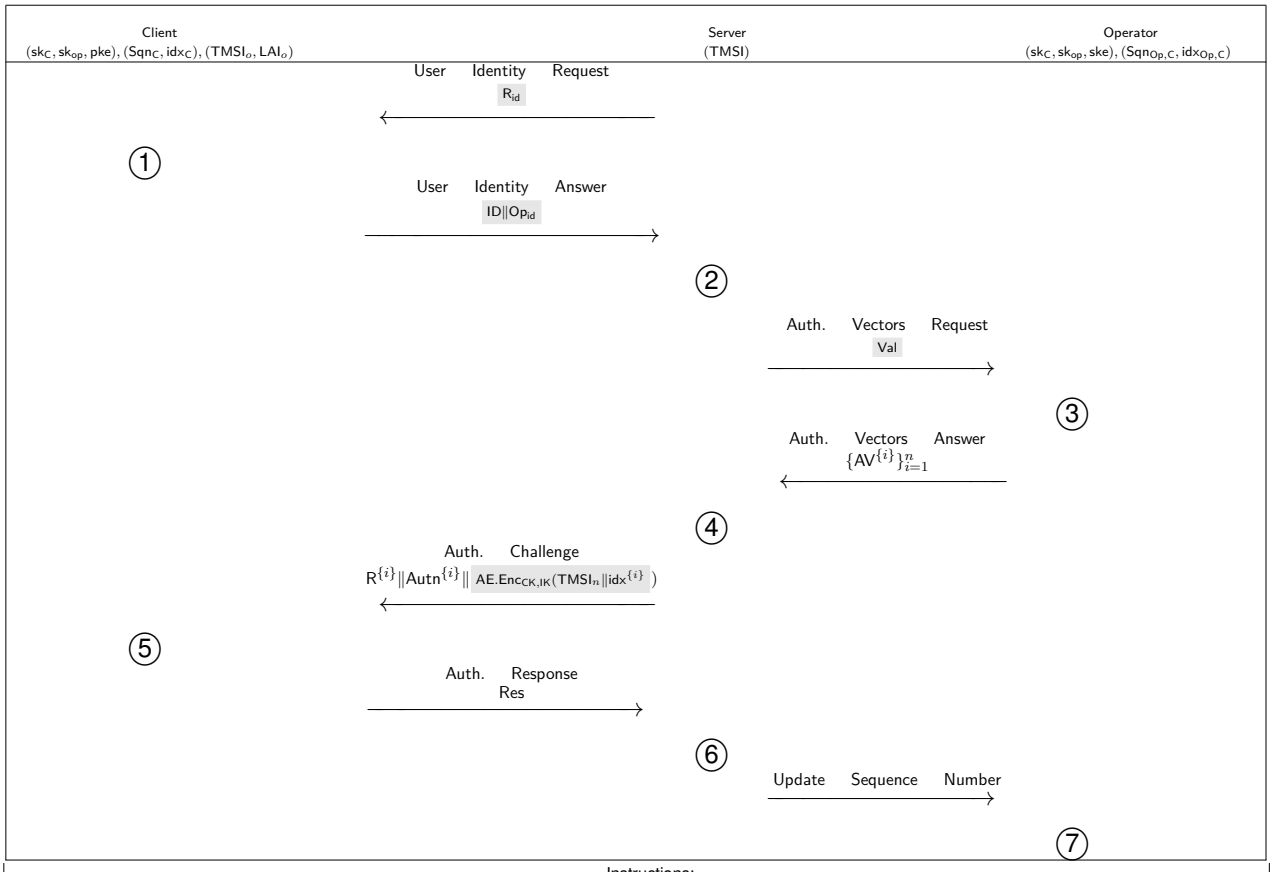
$\mathbb{G}_1 \Rightarrow \mathbb{G}_2$**:** In $\mathbb{G}_2$, we abort if one of three events happens:

- **Event 1:** The adversary $\mathcal{A}$ has forged the user identification answer on behalf of a target client C.
- **Event 2:** The adversary $\mathcal{A}$ has forged the authentication challenge for a client C.
- **Event 3:** The adversary $\mathcal{A}$ has forged the authentication response of an honest client C.

Any of these events can lead to the following attack: the adversary $\mathcal{A}$ chooses one of two clients (C_0 or C_1) to input to the DrawCl oracle, receiving the virtual handle VC. Then $\mathcal{A}$ acts as a MiM between the client VC and the server. For each event, the adversary will forge one specific message with respect to one of the two clients (say C_0), then forward the rest of the messages in that session. If the client is authenticated, then $\mathcal{A}$ guesses it was the client with respect to which the message was forged (in this case, C_0). Else, $\mathcal{A}$ guesses it was the other client.

We calculate the probability of causing each of the three events. To prompt (event 1), $\mathcal{A}$ can either guess/forge the identifier TMSI allocated through authenticated encryption (AE), or guess/forge the symmetric encryption of the corresponding IMSI and a unique (honestly generated) $\mathsf{R_{id}}$. Both possibilities imply breaking either the key-indistinguishability property, or the pseudorandomness of G.

For events 2,3, $\mathcal{A}$ could always try to replay an old authentication challenge. This is only accepted by the client if the recovered sequence number is either: (i) the client's current sequence number; (ii) the previous sequence number, but with an incremented index. For (i), the challenge



Instructions:

Client

①: Compute the identifier:
If $flag_{TMSI} := 0$ then ID = TMSI.
Else, $ID = PKE.Enc_{pke}(f_5(keys, R_{id}, IMSI, idx_C)||R_{id}||IMSI||idx_C)$.
$flag_{TMSI} := 1$.

⑤: Compute AK using $R^{\{i\}}$.
Recover $Sqn^{\{i\}}$ (from AK).
Check Mac_S value.
Compute: IK, CK;
Retrieve the received index and the new TMSI.
If abort caused or the AE does not verify, set $flag_{TMSI} := 1$ and increment: $idx_C := idx_C + 1$.

Else, check validity of $Sqn^{\{i\}}$, i.e if one of the following conditions is correct:

- $Sqn_C = Sqn^{\{i\}}$.
- $Sqn_C = inc(Sqn^{\{i\}})$ and $idx^{\{i\}} = idx_C + 1$.

If the first condition is accepted: reset the index idx_C, update the sequence number $Sqn_C = inc(Sqn_C)$.
If the second condition is accepted: $idx_C = idx_C + 1$.

Compute $Res := \mathcal{F}_1^*(keys, R^{\{i\}}, Sqn^{\{i\}}, Res_S, AMF)$.
Update the internal index. Allocate the new TMSI.
$flag_{TMSI} := 0$.

Server

②: Process the identifier ID:
If the identifier is a TMSI then Val = IMSI. Otherwise, $Val = (ID, R_{id})$.

④: Store $\{AV^{\{i\}}\}_{i=1}^{n}$.
Choose $AV^{\{i\}}$ one by one in order. Then, it sends the authentication challenge and the new couple $(TMSI_n, idx^{\{i\}})$ encrypted and authenticated by the session keys.

⑥: If the authentication of the client is verified ($Res \stackrel{?}{=} Mac_C$), then they ask to the server the update of its sequence number. Otherwise, the protocol is aborted.

Operator

③: Verify the identity of the client with Val.
If this holds, retrieve idx_C, set $idx_{Op,C} := idx_C$
Generate $(R^{\{1\}}, ..., R^{\{n\}})$. Denote: $keys := (sk_C, sk_{op})$.
For each $i = 1, \ldots, n$, compute:
$Mac_S \leftarrow \mathcal{F}_1(keys, R^{\{i\}}, Sqn^{\{i\}}, Res_S, AMF)$,
$Mac_C \leftarrow \mathcal{F}_1^*(keys, R^{\{i\}}, Sqn^{\{i\}}, Res_S, AMF)$,
$CK \leftarrow \mathcal{F}_3(keys, R^{\{i\}}, Sqn^{\{i\}}, Res_S, AMF)$,
$IK \leftarrow \mathcal{F}_4(keys, R^{\{i\}}, Sqn^{\{i\}}, Res_S, AMF)$,
$AK \leftarrow \mathcal{F}_5(keys, R^{\{i\}}, Ress)$,
$Autn^{\{i\}} \leftarrow (Sqn^{\{i\}} \oplus AK)||AMF||Mac_S$,
$Sqn^{\{i\}} \leftarrow inc(Sqn^{\{i-1\}})$,
$AV^{\{i\}} := (R^{\{i\}}, CK, IK, Autn^{\{i\}}, Mac_C, idx^{\{i\}})$, with $Sqn^{\{1\}} := Sqn_{Op,C}$,
$idx^{\{1\}} := idx_{Op,C}$, $\forall i \neq 1, idx^{\{i\}} = 0$.
End for.

⑦: Update the sequence number:
$Sqn_{Op,C} \leftarrow inc(Sqn_{Op,C})$. Reset the index $idx_{Op,C}$.

Fig. 6. Our fixed AKA Procedure.

is fresh and the client may at most guess it. Achieving (ii) implies that $\mathcal{A}$ must forge the authentication of an incremented index starting from AE output either with the wrong index, or with the wrong session keys. Finally, for

fresh R, the best $\mathcal{A}$ can do to forge Res is to guess it or break the pseudorandomness of G. This yields:

$$\left|\Pr[\mathcal{A}_{\mathbb{G}_1}\mathsf{wins}] - \Pr[\mathcal{A}_{\mathbb{G}_2}\mathsf{wins}]\right| \leq 2 \cdot \frac{1+(q_{\mathsf{exec}}\cdot q_{\mathsf{id}})^2}{2^{|\mathsf{TMSI}|}} + 4n_{\mathsf{C}} \cdot (\mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}) + 2\mathsf{Adv}^{\mathsf{mac}}_{G}(\mathcal{A})) + 3\mathsf{Adv}^{\mathsf{K.Ind}}_{\Pi}(\mathcal{A}) \quad .$$

$\mathbb{G}_2 \Rightarrow \mathbb{G}_3$**:** In $\mathbb{G}_3$ we replace the output of all the internal functions $(G, \mathsf{AE}, \mathsf{PKE})$ by truly random, but consistent values. The security loss is related to the advantage against the pseudorandomness of G and the related security of the functions PKE and AE.

$\mathbb{G}_3$**:** At this point, protocol transcripts are consistent, but random, and with unique randomness, thus $\mathcal{A}$ can at most guess the bit b.

Security Statement. This yields the following result:

$$\mathsf{Adv}^{\mathsf{ww\text{-}unlink}}_{\Pi}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{K.Ind}}_{\Pi}(\mathcal{A}_4) + 2 \cdot \frac{1+(q_{\mathsf{exec}}\cdot q_{\mathsf{id}})^2}{2^{|\mathsf{TMSI}|}} + \frac{q^2_{\mathsf{exec}}}{2^{|\mathsf{R}|}} + \frac{(q_{\mathsf{exec}}\cdot q_{\mathsf{id}})^2}{2^{|\mathsf{R_{id}}|}} + n_{\mathsf{C}} \cdot (10 \cdot \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_1) + 5 \cdot \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}_3)).$$

Proof of K.Ind.

$\mathbb{G}_0$**:** We consider the K.Ind game with server corruptions.

$\mathbb{G}_0 \Rightarrow \mathbb{G}_1$**:** In $\mathbb{G}_1$ we extend the corruption oracle to give the adversary access to the operator key $\mathsf{sk_{op}}$. There is no security loss (this is trivial since in game $\mathbb{G}_1$, $\mathcal{A}$ has more information).

$\mathbb{G}_1 \Rightarrow \mathbb{G}_2$**:** In $\mathbb{G}_2$, $\mathcal{A}$ can only interact with a single client. For the reduction, the adversary in $\mathbb{G}_2$ can simulate queries perfectly for the adversary in $\mathbb{G}_1$ as long as it guesses the target (tested) client. This yields a security loss of $\frac{1}{n_{\mathsf{C}}}$.

$\mathbb{G}_2 \Rightarrow \mathbb{G}_3$**:** In $\mathbb{G}_3$, we abort if two honest server instances output same random values R or $\mathsf{R_{id}}$. As before, we lose $\frac{(q_{\mathsf{exec}}+q_{\mathsf{serv}}\cdot q_{\mathsf{Op}})^2}{2^{|\mathsf{R}|}} + \frac{(q_{\mathsf{exec}}q_{\mathsf{id}})^2}{2^{|\mathsf{R_{id}}|}}$.

$\mathbb{G}_3 \Rightarrow \mathbb{G}_4$**:** In $\mathbb{G}_4$ $\mathcal{A}$ only plays against one server. Thus $\mathcal{A}$ can no longer corrupt servers and query operators; however, note that server corruptions are not used to the adversary anyway, because they contain a different $\mathsf{Res_S}$. Thus, we lose at most the advantage of breaking the pseudorandomness of G.

$\mathbb{G}_4 \Rightarrow \mathbb{G}_5$**:** In $\mathbb{G}_5$, we replace the output of $(G, \mathsf{AE}, \mathsf{PKE})$ by truly random, consistent values. The reduction goes as in the previous proof.

$\mathbb{G}_5$**:** In this game, $\mathcal{A}$ must distinguish real from random keys if the transcripts are truly random, for unique $\mathsf{R}, \mathsf{R_{id}}$ values. Thus, $\mathcal{A}$ can at most guess.

Security Statement. This yields the following bound:

$$\mathsf{Adv}^{\mathsf{K.Ind}}_{\Pi}(\mathcal{A}_{\mathbb{G}_0}) \leq n_{\mathsf{C}} \cdot (\frac{(q_{\mathsf{exec}}\cdot q_{\mathsf{id}})^2}{2^{|\mathsf{R_{id}}|}} + \frac{(q_{\mathsf{exec}}+q_{\mathsf{serv}}\cdot q_{\mathsf{Op}})^2}{2^{|\mathsf{R}|}} + 2\mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_1) + +\mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}_3).$$

Proof of C.Imp.

$\mathbb{G}_0$**:** This is the C.Imp game outlined in the full version [10]. This is a classical client-impersonation attack by an active MiM that can corrupt servers.

$\mathbb{G}_0 \Rightarrow \mathbb{G}_2$**:** Use the same game hops as for K.Ind. In $\mathbb{G}_2$, $\mathcal{A}$ interacts with only one client.

$\mathbb{G}_2 \Rightarrow \mathbb{G}_3$**:** In $\mathbb{G}_3$ $\mathcal{A}$ only interacts with a single uncorruptible server. The security loss is given by the collision on two outputs of the same function G with two different inputs (at least $\mathsf{Res_S}$ differs). The security loss is thus bounded by the advantage of a best distinguisher against the pseudorandomness of G.

$\mathbb{G}_3 \Rightarrow \mathbb{G}_4$**:** In $\mathbb{G}_3$ we replace the output of $(G, \mathsf{AE}, \mathsf{PKE})$ by truly random, consistent values, as in the previous proof.

$\mathbb{G}_4 \Rightarrow \mathbb{G}_5$**:** In $\mathbb{G}_5$, we abort if two honest server instances output the same values R or $\mathsf{R_{id}}$. The loss is $\frac{(q_{\mathsf{exec}})^2}{2^{|\mathsf{R}|}} + \frac{(q_{\mathsf{exec}}q_{\mathsf{id}})^2}{2^{|\mathsf{R_{id}}|}}$.

$\mathbb{G}_5$**:** Now $\mathcal{A}$ plays the game against a single client and server, with only truly random, but consistent values, for unique R and $\mathsf{R_{id}}$ values. There are three options: (1) re-using a value already received from the honest client; (2) guessing the key $\mathsf{sk_C}$; (3) guessing the response Res. The first option yields no result, since it implies there exists a previous client instance with the same session id sid as the client. The second option happens with a probability of $2^{-|\mathsf{sk_C}|}$. The third option occurs with a probability of $2^{-|\mathsf{Res}|} + 2^{-|\mathsf{ID}|}$ per session, thus a total of $q_{\mathsf{exec}} \cdot (2^{-|\mathsf{Res}|} + q_{\mathsf{id}} \cdot 2^{-|\mathsf{ID}|})$.

Security Statement. This yields the following result:

$$\mathsf{Adv}^{\mathsf{C.Imp}}_{\Pi}(\mathcal{A}_{\mathbb{G}_0}) \leq n_{\mathsf{C}} \cdot (\mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}_3) + \frac{(q_{\mathsf{exec}}\cdot q_{\mathsf{id}})^2}{2^{|\mathsf{R_{id}}|}} + \frac{(q_{\mathsf{exec}})^2}{2^{|\mathsf{R}|}} + \frac{q_{\mathsf{exec}}}{2^{|\mathsf{Res}|}} + \frac{1}{2^{\kappa}} + \frac{q_{\mathsf{exec}}\cdot q_{\mathsf{id}}}{2^{|\mathsf{ID}|}}).$$

Proof of S.Imp.

Game $\mathbb{G}_0$: This is the S.Imp-game with server corruptions, equivalent to the C.Imp game.

$\mathbb{G}_0 \Rightarrow \mathbb{G}_4$**:** We use the same game hops from the previous proof of the C.Imp-security to reach to $\mathbb{G}_4$, in which $\mathcal{A}$ can interact with a single client and a single server, and all output from $(G, \mathsf{AE}, \mathsf{PKE})$ is truly random and consistent.

Game $\mathbb{G}_4$: Now $\mathcal{A}$ plays the game with a single client C_i, which only accepts $\mathcal{A}_{\mathbb{G}_4}$, if the authentication challenge is verified. Assume that this happens against instance C_i of the target client, for some target session sid. The MAC value $\mathsf{Mac_S}$ computed by C_i is purely random, but consistent. Now $\mathcal{A}$ can: (a) forward a value already received from the honest server for the same R; Sqn; $\mathsf{sk_{op}}$; $\mathsf{sk_C}$, of which $\mathsf{sk_C}$ is unknown; (b) guess $\mathsf{sk_C}$; or (c) guess the response. As in the previous proof, this yields:

$$\Pr[\mathcal{A}_{\mathbb{G}_4}\mathsf{wins}] = 2^{-|\mathsf{sk_C}|} + q_{\mathsf{exec}} \cdot 2^{-|\mathsf{Mac_S}|}.$$

Security statement: This yields the following result:

$$\mathsf{Adv}^{\mathsf{S.Imp}}_{\Pi}(\mathcal{A}_{\mathbb{G}_0}) \leq n_{\mathsf{C}} \cdot (\frac{q_{\mathsf{exec}}}{2^{|\mathsf{Mac_S}|}} + \frac{1}{2^{\kappa}} + \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}_3)).$$

Proof of St.conf.

$\mathbb{G}_0$**:** This is the St.conf game as outlined before.

$\mathbb{G}_0 \Rightarrow \mathbb{G}_1$**:** In $\mathbb{G}_1$ we only include one operator. The security loss is the advantage against the pseudorandomness of G when keyed with $\mathsf{sk_C}$.

$\mathbb{G}_1 \Rightarrow \mathbb{G}_2$**:** In $\mathbb{G}_2$, $\mathcal{A}$ can only interact with a single client. An interesting difficulty is simulating queries for non-target clients affiliated to the target operator. We can do this by requiring that the AKA algorithms behave as a PRF when keyed with $\mathsf{sk_{op}}$ (this function is called G^*). The adversary requires oracles to both G^* and G.

$\mathbb{G}_2 \Rightarrow \mathbb{G}_3$**:** In $\mathbb{G}_3$, we replace G and G^* by truly random, consistent values. We lose the advantage against the pseudorandomness of G or G^*.

$\mathbb{G}_3$**:** Now $\mathcal{A}$ plays the game against a single client C, using truly random values. She can output at least one correct long-term secret for C only by guessing it, namely with probability: $\frac{1}{2^{|\mathsf{sk_C}|}} + \frac{1}{2^{|\mathsf{sk_{Op}}|}} + \frac{2}{2^{|\mathsf{Sqn}|}}$.

Security Statement. This yields the following result:

$$\mathsf{Adv}^{\mathsf{St.conf}}_{\Pi}(\mathcal{A}_{\mathbb{G}_0}) \leq n_{\mathsf{C}} \cdot 2^{-|\mathsf{sk_C}|} + 2^{-|\mathsf{sk_{op}}|} + 2 \cdot 2^{-|\mathsf{Sqn_C}|} + \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}_3)) + \mathsf{Adv}^{\mathsf{prf}}_{G^*}(\mathcal{A}_4).$$

Proof of S.sound.

$\mathbb{G}_0$**:** This game resembles S.Imp, but we play against a malicious server.

$\mathbb{G}_0 \Rightarrow \mathbb{G}_1$**:** We define game $\mathbb{G}_1$ as the modification of the S.Imp game in which $\mathcal{A}$ can corrupt operators and get up to q_{Op} authentication vectors; the adversary's goal is to authenticate $q_{\mathsf{Op}} + 1$ times. We show that for every adversary winning game $\mathbb{G}_0$ with some probability, there exists an adversary winning game $\mathbb{G}_1$ with the same probability.

$\mathbb{G}_1$**:** By the construction of $\mathbb{G}_1$ the success probability of the adversary is now only at most the S.Imp bound.

Security Statement. This yields the following result:

$$\mathsf{Adv}^{\mathsf{S.sound}}_{\Pi}(\mathcal{A}_{\mathbb{G}_0}) \leq n_{\mathsf{C}} \cdot (\frac{q_{\mathsf{exec}}}{2^{|\mathsf{Mac_S}|}} + \frac{1}{2^{\kappa}} + 2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{G}(\mathcal{A}_1) + \mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE}}(\mathcal{A}_2) + \mathsf{Adv}^{\mathsf{ind-cca2}}_{\mathsf{PKE}}(\mathcal{A}_3)).$$

C Evaluation

In proposing our variant of AKA we explicitly or implicitly addressed several attacks. We discuss these below, referring the reader to Figure 7 for a better overview.

Server Corruptions : The original AKA protocol only offers a degree of key-indistinguishability and impersonation security, only in the absence of server corruptions. Since servers are trusted to run the authenticated key-exchange step, corrupting a server compromises any security of a channel this server establishes; however, in the AKA routine, this flaw is exacerbated, since the corruption results can be re-used later in non-vulnerable areas. This is an active, and rather complex attack, but it is highly parallelizable and has a great security impact. To mitigate this risk, we added a server-specific, unique, publicly-known identifier $\mathsf{Res_S}$, which is now given as input to all the cryptographic functions.

Client Confidentiality : IMSI catcher attacks compromise the client's identity in a direct way (the adversary learns a static identifier). This attack can be run (with a reduced success probability) even by passive attackers, and it is highly parallelizable. The consequence is that multiple clients can be tracked simultaneously in a mass-surveillance operation. We mitigate such risks by ensuring that no static identifier is leaked through, by using a PKE scheme, in which only the operators have secret and public keys.

Client Unlinkability : Even if the adversary cannot track a user back to a permanent identifier, she can still try to distinguish between two chosen users, e.g. by causing some unusual protocol steps. Attacks like distinguishing between two different failure messages (which are actively triggered by the adversary), injecting a message and then seeing its effect in a protocol run (which is accepted if

Added countermeasures	Cost	Attacks it Prevents	Attack Impact
Client sends encrypted IMSI	- Needs IND-CCA PKE encryption - Simple PKI (only operators)	Client Confidentiality: (IMSI Catchers)	Trace many users - Parallelizable - Passive/Active
	Large TMSI size	Client unlinkability: Distinguish TMSI/IMSI msg.	Trace 1 user: - Non-parallelizable - Active only
Authenticate TMSI reallocation (see also: index)	New reallocation alg.	Client unlinkability: (Denial-of-Service)	Trace many users - Parallelizable - Active only
		Client unlinkability: Distinguish TMSI/IMSI	Trace 1 user - Non-parallelizable - Active only
Index idx_C, idx_S	New 1-bit state variable	Client unlinkability: Prompt resynch, distinguish	Trace 1 user - Non-parallelizable - Active only
		S.Imp-resistance: Challege is un-replayable	Impersonate servers - Parallelizable - Active only
Introducing Res_S	- New server identifier - Changed crypto algs.	S.Imp-resistance k.ind-security S.sound-security (Server Corruptions)	Break sec. channel - Parallelizable - Needs corruptions - Active only
Use only current LAI	- Clients must know LAI - Clients store Res_S	Location privacy: (Track past LAI)	Trace 1 user per LAI - Non-Parallelizable - Passive

Fig. 7. Assessment of our AKA variant: cost and effect of coutermeasures.

the chosen client is compatible with the injected message, and rejected otherwise), or distinguishing between messages of distinct lengths allow client linkability. While not as versatile, nor as parallelizable as client confidentiality attacks, these threat nevertheless allow an insidious adversary to track a user that is singled-out for mass surveillance. In our variant, we make protocol executions for different users indistinguishable from one another, at the cost of larger TMSI values, a new index variable, using IND-CCA PKE encryption, and making the operator intervene in the case of an error.

Denial of Service : Apart from being a means of breaking client-unlinkability, DoS attacks can also facilitate IMSI catchers, and add to the complexity of AKA. One way of causing a DoS in AKA is to send a random string instead of the TMSI reallocation message. The client will parse this as some TMSI value, which, when reused, will not be traceable by the server. The latter then requests the IMSI in clear. We mitigate this DoS attack by using authenticated encryption for the TMSI reallocation and ensuring that no desynchronizations can occur.

Itinerary tracking : One disadvantage of AKA is that the client's past location is revealed during the protocol, allowing to track up to 1 user per LAI at any one time. We bypass this difficulty by only using current LAI values.