The 5G-AKA Authentication Protocol Privacy

Adrien Koutsos
LSV, CNRS, ENS Paris-Saclay, Université Paris-Saclay
Cachan, France
adrien.koutsos@lsv.fr

***Abstract*—We study the 5G-AKA authentication protocol described in the 5G mobile communication standards. This version of AKA tries to achieve a better privacy than the 3G and 4G versions through the use of asymmetric randomized encryption. Nonetheless, we show that except for the IMSI-catcher attack, all known attacks against 5G-AKA privacy still apply.**

Next, we modify the 5G-AKA protocol to prevent these attacks, while satisfying 5G-AKA efficiency constraints as much as possible. We then formally prove that our protocol is σ-unlinkable. This is a new security notion, which allows for a fine-grained quantification of a protocol privacy. Our security proof is carried out in the Bana-Comon indistinguishability logic. We also prove mutual authentication as a secondary result.

***Index Terms*—AKA, Unlinkability, Privacy, Formal Methods.**

I. Introduction

Mobile communication technologies are widely used for voice, text and Internet access. These technologies allow a subscriber's device, typically a mobile phone, to connect wirelessly to an antenna, and from there to its service provider. The two most recent generations of mobile communication standards, the 3G and 4G standards, have been designed by the 3GPP consortium. The *fifth generation* (5G) of mobile communication standards is being finalized, and drafts are now available [1]. These standards describe protocols that aim at providing security guarantees to the subscribers and service providers. One of the most important such protocol is the *Authentication and Key Agreement* (AKA) protocol, which allows a subscriber and its service provider to establish a shared secret key in an authenticated fashion. There are different variants of the AKA protocol, one for each generation.

In the 3G and 4G-AKA protocols, the subscriber and its service provider share a long term secret key. The subscriber stores this key in a cryptographic chip, the *Universal Subscriber Identity Module* (*USIM*), which also performs all the cryptographic computations. Because of the *USIM* limited computational power, the protocols only use symmetric key cryptography without any pseudo-random number generation on the subscriber side. Therefore the subscriber does not use a random challenge to prevent replay attacks, but instead relies on a sequence number SQN. Since the sequence number has to be tracked by the subscriber and its service provider, the AKA protocols are stateful.

Because a user could be easily tracked through its mobile phone, it is important that the AKA protocols provide privacy guarantees. The 3G and 4G-AKA protocols try to do that using temporary identities. While this provides some privacy against a *passive adversary*, this is not enough against an *active adversary*. Indeed, these protocols allow an antenna to ask for a user permanent identity when it does not know its temporary identity (this naturally happens in roaming situations). This mechanism is abused by IMSI-catchers [2] to collect the permanent identities of all mobile devices in range.

The IMSI-catcher attack is not the only known attack against the privacy of the AKA protocols. In [3], the authors show how an attacker can obtain the least significant bits of a subscriber's sequence number, which allows the attacker to monitor the user's activity. The authors of [4] describe a linkability attack against the 3G-AKA protocol. This attack is similar to the attack on the French e-passport [5], and relies on the fact that 3G-AKA protocol uses different error messages if the authentication failed because of a bad Mac or because a de-synchronization occurred.

The 5G standards include changes to the AKA protocol to improve its privacy guarantees. In 5G-AKA, a user never sends its permanent identity in plain-text. Instead, it encrypts it using a *randomized asymmetric encryption* with its service provider public key. While this prevents the IMSI-catcher attack, this is not sufficient to get unlinkability. Indeed, the attacks from [3], [4] against the 3G and 4G-AKA protocols still apply. Moreover, the authors of [6] proposed an attack against a variant of the AKA protocol introduced in [4], which uses the fact that an encrypted identity can be replayed. It turns out that their attack also applies to 5G-AKA.

a) Objectives: Our goal is to improve the privacy of 5G-AKA while satisfying its design and efficiency constraints. In particular, our protocol should be as efficient as the 5G-AKA protocol, have a similar communication complexity and rely on the same cryptographic primitives. Moreover, we want formal guarantees on the privacy provided by our protocol.

b) Formal Methods: Formal methods are the best way to get a strong confidence in the security provided by a protocol. They have been successfully applied to prove the security of crucial protocols, such as Signal [7] and TLS [8], [9]. There exist several approaches to formally prove a protocol security.

In the *symbolic* or *Dolev-Yao* (DY) model, protocols are modeled as members of a formal process algebra [10]. In this model, the attacker controls the network: he reads all messages and he can forge new messages using capabilities granted to him through a fixed set of rules. While security in this model can be automated (e.g. [11]–[14]), it offers limited guarantees: we only prove security against an attacker that has the designated capabilities.

The *computational model* is more realistic. The attacker also controls the network, but is not limited by a fixed set of rules. Instead, the attacker is any Probabilistic Polynomial-time Turing Machine (PPTM for short). Security proofs in this model are typically sequences of game transformations [15] between a game stating the protocol security and cryptographic hypotheses. This model offers strong security guarantees, but proof automation is much harder. For instance, CRYPTOVERIF [16] cannot prove the security of stateful cryptographic protocols (such as the AKA protocols).

There is a third model, the *Bana-Comon* (BC) model [17], [18]. In this model, messages are terms and the security property is a first-order formula. Instead of granting the attacker capabilities through rules, as in the symbolic approach, we state what the adversary *cannot* do. This model has several advantages. First, since security in the BC model entails computational security, it offers strong security guarantees. Then, there is no ambiguity: the adversary can do anything which is not explicitly forbidden. Finally, this approach is well-suited to model stateful protocols.

c) Related Work: There are several formal analysis of AKA protocols in the symbolic models. In [12], the authors use the DEEPSEC tool to prove unlinkability of the protocol for three sessions. In [4] and [19], the authors use PROVERIF to prove unlinkability of AKA variants for, respectively, three sessions and an unbounded number of sessions. In these three works, the authors abstracted away several key features of the protocol. Because DEEPSEC and PROVERIF do not support the xor operator, they replaced it with a symmetric encryption. Moreover, sequence numbers are modeled by nonces in [4] and [12]. While [19] models the sequence number update, they assume it is always incremented by one, which is incorrect. Finally, none of these works modeled the re-synchronization or the temporary identity mechanisms. Because of these inaccuracies in their models, they all miss attacks.

In [20], the authors use the TAMARIN prover to analyse multiple properties of 5G-AKA. For each property, they either find a proof, or exhibit an attack. To our knowledge, this is the most precise symbolic analysis of an AKA protocol. For example, they correctly model the xor and the re-synchronization mechanisms, and they represent sequence numbers as integers (which makes their model stateful). Still, they decided not to include the temporary identity mechanism. Using this model, they successfully rediscover the linkability attack from [4].

We are aware of two analysis of AKA protocols in the computational model. In [6], the authors present a significantly modified version of AKA, called PRIV-AKA, and claim it is unlinkable. However, we discovered a linkability attack against the protocol, which falsifies the authors claim. In [21], the authors study the 4G-AKA protocol *without its first message*. They show that this reduced protocol satisfies a form of anonymity (which is weaker than unlinkability). Because they consider a weak privacy property for a reduced protocol, they fail to capture the linkability attacks from the literature.

To summarize, there is currently no computational security proof of a complete version of an AKA protocol.

d) Contributions: Our contributions are:

- We study the privacy of the 5G-AKA protocol described in the 3GPP draft [1]. Thanks to the introduction of asymmetric encryption, the 5G version of AKA is not vulnerable to the IMSI-catcher attack. However, we show that the linkability attacks from [3], [4], [6] against older versions of AKA still apply to 5G-AKA.
- We present a new linkability attack against PRIV-AKA, a significantly modified version of the AKA protocol introduced and claimed unlinkable in [6]. This attack exploits the fact that, in PRIV-AKA, a message can be delayed to yield a state update later in the execution of the protocol, where it can be detected.
- We propose the AKA$^+$ protocol, which is a modified version of 5G-AKA with better privacy guarantees and satisfying the same design and efficiency constraints.
- We introduce a new privacy property, called σ-unlinkability, inspired from [22] and Vaudenay's Privacy [23]. Our property is parametric and allows us to have a fine-grained quantification of a protocol privacy.
- We formally prove that AKA$^+$ satisfies the σ-unlinkability property in the computational model. Our proof is carried out in the BC model, and holds for any number of agents and sessions that are not related to the security parameter. We also show that AKA$^+$ provides mutual authentication.

e) Outline: In Section II and III we describe the 5G-AKA protocol and the known linkability attacks against it. We present the AKA$^+$ protocol in Section IV, and we define the σ-unlinkability property in Section V. Finally, we show how we model the AKA$^+$ protocol using the BC logic in Section VI, and we state and sketch the proofs of the mutual authentication and σ-unlinkability of AKA$^+$ in Section VII. This is an extended abstract without the full proofs, which can be found in the technical report [24].

II. THE 5G-AKA PROTOCOL

We present the 5G-AKA protocol described in the 3GPP standards [1]. This is a three-party authentication protocol between:

- The *User Equipment* (*UE*). This is the subscriber's physical device using the mobile communication network (e.g. a mobile phone). Each *UE* contains a cryptographic chip, the *Universal Subscriber Identity Module* (*USIM*), which stores the user confidential material (such as secret keys).
- The *Home Network* (*HN*), which is the subscriber's service provider. It maintains a database with the necessary data to authenticate its subscribers.
- The *Serving Network* (*SN*). It controls the base station (the antenna) the *UE* is communicating with through a wireless channel.

If the *HN* has a base station nearby the *UE*, then the *HN* and the *SN* are the same entity. But this is not always the case (e.g. in roaming situations). When no base station from the user's *HN* are in range, the *UE* uses another network's base station.

The *UE* and its corresponding *HN* share some confidential key material and the *Subscription Permanent Identifier* (SUPI), which uniquely identifies the *UE*. The *SN* does not have access to the secret key material. It follows that all cryptographic computations are performed by the *HN*, and sent to the *SN* through a secure channel. The *SN* also forwards all the information it gets from the *UE* to the *HN*. But the *UE* permanent identity is not kept hidden from the *SN*: after a successful authentication, the *HN* sends the SUPI to the *SN*. This is not technically needed, but is done for legal reasons. Indeed, the *SN* needs to know whom it is serving to be able to answer to *Lawful Interception* requests.

Therefore, privacy requires to trust both the *HN* and the *SN*. Since, in addition, they communicate through a secure channel, we decided to model them as a single entity and we include the *SN* inside the *HN*. A description of the protocol with three distinct parties can be found in [20].

A. Description of the Protocol

The 5G standard proposes two authentication protocols, EAP-AKA′ and 5G-AKA. Since their differences are not relevant for privacy, we only describe the 5G-AKA protocol.

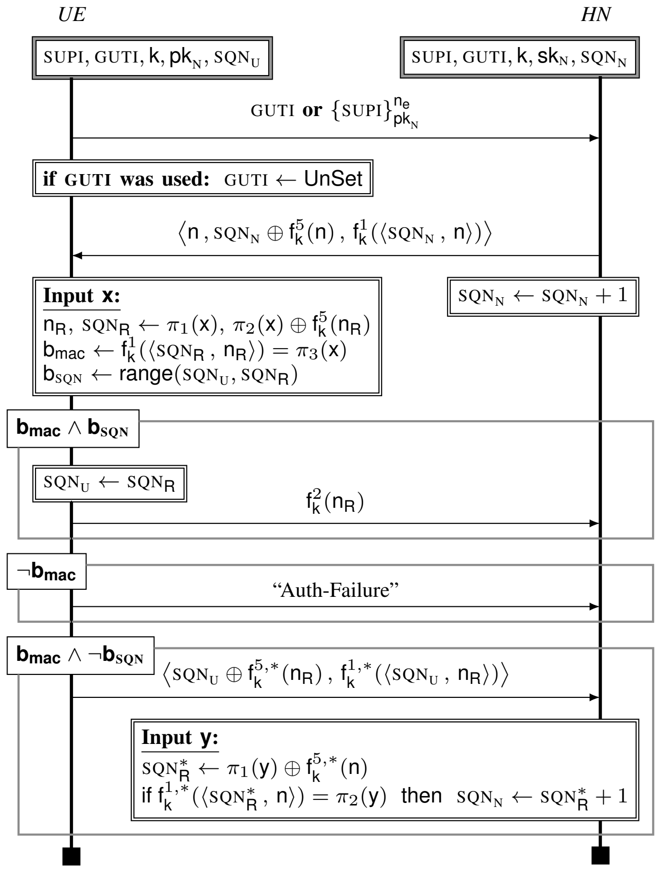
a) Cryptographic Primitives: As in the 3G and 4G variants, the 5G-AKA protocol uses several keyed cryptographic one-way functions: f^1, f^2, f^5, $\mathsf{f}^{1,*}$ and $\mathsf{f}^{5,*}$. These functions are used both for integrity and confidentiality, and take as input a long term secret key k (which is different for each subscriber).

A major novelty in 5G-AKA is the introduction of an asymmetric randomized encryption $\{\cdot\}_{\mathsf{pk}}^{\mathsf{n_e}}$. Here pk is the public key, and $\mathsf{n_e}$ is the encryption randomness. Previous versions of AKA did not use asymmetric encryption because the *USIM*, which is a cryptographic micro-processor, had no randomness generation capabilities. The asymmetric encryption is used to conceal the identity of the *UE*, by sending $\{\text{SUPI}\}_{\mathsf{pk}}^{\mathsf{n_e}}$ instead of transmitting the SUPI in clear (as in 3G and 4G-AKA).

b) Temporary Identities: After a successful run of the protocol, the *HN* may issue a temporary identity, a *Globally Unique Temporary Identity* (GUTI), to the *UE*. Each GUTI can be used in *at most one session* to replace the encrypted identity $\{\text{SUPI}\}_{\mathsf{pk}}^{\mathsf{n_e}}$. It is renewed after each use. Using a GUTI allows to avoid one asymmetric encryption. This saves a pseudo-random number generation and the expensive computation of an asymmetric encryption.

c) Sequence Numbers: The 5G-AKA protocol prevents replay attacks using a sequence number SQN instead of a random challenge. This sequence number is included in the messages, incremented after each successful run of the protocol, and must be tracked and updated by the *UE* and the *HN*. As it may get de-synchronized (e.g. because a message is lost), there are two versions of it: the *UE* sequence number SQN_U, and the *HN* sequence number SQN_N.

d) State: The *UE* and *HN* share the *UE* identity SUPI, a long-term symmetric secret key k, a sequence number SQN_U and the *HN* public key $\mathsf{pk_N}$. The *UE* also stores in GUTI the value of the last temporary identity assigned to it (if there is one). Finally, the *HN* stores the secret key $\mathsf{sk_N}$ corresponding to $\mathsf{pk_N}$, its version SQN_N of every *UE*'s sequence number and a mapping between the GUTIs and the SUPIs.



Conventions: $\leftarrow$ is used for assignments, and has a lower priority than the equality comparison operator $=$.

Fig. 1. The 5G-AKA Protocol

e) Authentication Protocol: The 5G-AKA protocol is represented in Fig. 1. We now describe an honest execution of the protocol. The *UE* initiates the protocol by identifying itself to the *HN*, which it can do in two different ways:

- It can send a temporary identity GUTI, if one was assigned to it. After sending the GUTI, the *UE* sets it to UnSet to ensure that it will not be used more than once. Otherwise, it would allow an adversary to link sessions together.
- It can send its concealed permanent identity $\{\text{SUPI}\}_{\mathsf{pk_N}}^{\mathsf{n_e}}$, using the *HN* public key $\mathsf{pk_N}$ and a fresh randomness $\mathsf{n_e}$.

Upon reception of an identifying message, the *HN* retrieves the permanent identity SUPI: if it received a temporary identity GUTI, this is done through a database look-up; and if a concealed permanent identity was used, it uses $\mathsf{sk_N}$ to decrypt it. It can then recover SQN_N and the key k associated to the identity SUPI from its memory. The *HN* then generates a fresh nonce n. It masks the sequence number SQN_N by xoring it with $\mathsf{f}^5_\mathsf{k}(\mathsf{n})$, and mac the message by computing $\mathsf{f}^1_\mathsf{k}(\langle \text{SQN}_\text{N}, \mathsf{n}\rangle)$ (we use $\langle \dots \rangle$ for tuples). It then sends the message $\langle \mathsf{n}, \text{SQN}_\text{N} \oplus \mathsf{f}^5_\mathsf{k}(\mathsf{n}), \mathsf{f}^1_\mathsf{k}(\langle \text{SQN}_\text{N}, \mathsf{n}\rangle)\rangle$.

When receiving this message, the *UE* computes $\mathsf{f}^5_\mathsf{k}(\mathsf{n})$. With it, it unmasks SQN_N and checks the authenticity of the message by re-computing $\mathsf{f}^1_{\mathsf{k}}(\langle \text{SQN}_{\text{N}}\, , \, \mathsf{n} \rangle)$ and verifying that it is equal to the third component of the message. It also checks whether SQN_{N} and SQN_{U} are in range[1]. If both checks succeed, the *UE* sets SQN_{U} to SQN_{N}, which prevents this message from being accepted again. It then sends $\mathsf{f}^2_{\mathsf{k}}(\mathsf{n})$ to prove to *HN* the knowledge of k. If the authenticity check fails, an "Auth-Failure" message is sent. Finally, if the authenticity check succeeds but the range check fails, *UE* starts the re-synchronization sub-protocol, which we describe below.

f) Re-synchronization: The re-synchronization protocol allows the *HN* to obtain the current value of SQN_{U}. First, the *UE* masks SQN_{U} by xoring it with $\mathsf{f}^{5,*}_{\mathsf{k}}(\mathsf{n})$, mac the message using $\mathsf{f}^{1,*}_{\mathsf{k}}(\langle \text{SQN}_{\text{U}}\, , \, \mathsf{n} \rangle)$ and sends the pair $\langle \text{SQN}_{\text{U}} \oplus \mathsf{f}^{5,*}_{\mathsf{k}}(\mathsf{n})\, , \, \mathsf{f}^{1,*}_{\mathsf{k}}(\langle \text{SQN}_{\text{U}}\, , \, \mathsf{n} \rangle) \rangle$. When receiving this message, the *HN* unmasks SQN_{U} and checks the mac. If the authentication test is successful, *HN* sets the value of SQN_{N} to $\text{SQN}_{\text{U}} + 1$. This ensures that *HN* first message in the next session of the protocol is in the correct range.

g) GUTI *Assignment:* There is a final component of the protocol which is not described in Fig. 1 (as it is not used in the privacy attacks we present later). After a successful run of the protocol, the *HN* generates a new temporary identity GUTI and links it to the *UE*'s permanent identity in its database. Then, it sends the masked fresh GUTI to the *UE*.

III. Unlinkability Attacks Against 5G-AKA

We present in this section several attacks against AKA that appeared in the literature. All these attacks but one (the IMSI-catcher attack) carry over to 5G-AKA. Moreover, several fixes of the 3G and 4G versions of AKA have been proposed. We discuss the two most relevant fixes, the first by Arapinis et al. [4], and the second by Fouque et al. [6].

None of these fixes are satisfactory. The modified AKA protocol given in [4] has been shown flawed in [6]. The authors of [6] then propose their own protocol, called PRIV-AKA, and claim it is unlinkable (they only provide a proof sketch). While analyzing the PRIV-AKA protocol, we discovered an attack allowing to permanently de-synchronize the *UE* and the *HN*. Since a de-synchronized *UE* can be easily tracked (after being de-synchronized, the *UE* rejects all further messages), our attack is also an unlinkability attack. This is in direct contradiction with the security property claimed in [6]. This is a novel attack that never appeared in the literature.

A. IMSI-*Catcher Attack*

All the older versions of AKA (4G and earlier) are vulnerable to the IMSI-catcher attack [2]. This attack simply relies on the fact that, in these versions of AKA, the permanent identity (called the *International Mobile Subscriber Identity* or IMSI in the 4G specifications) is not encrypted but sent in plain-text. Moreover, even if a temporary identity is used (a *Temporary Mobile Subscriber Identity* or TMSI), an attacker can simply send a Permanent-ID-Request message to obtain the *UE*'s permanent identity. The attack is depicted in Fig. 2.

[1]The specification is loose here: it only requires that $\text{SQN}_{\text{U}} < \text{SQN}_{\text{N}} \leq \text{SQN}_{\text{U}} + C$, where C is some constant chosen by the *HN*.

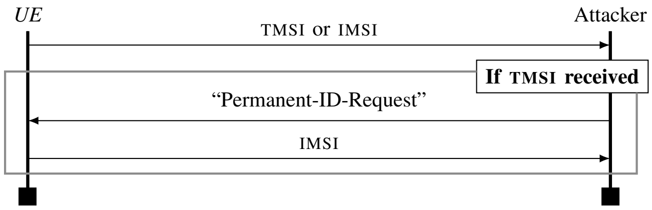


Fig. 2. An IMSI-Catcher Attack

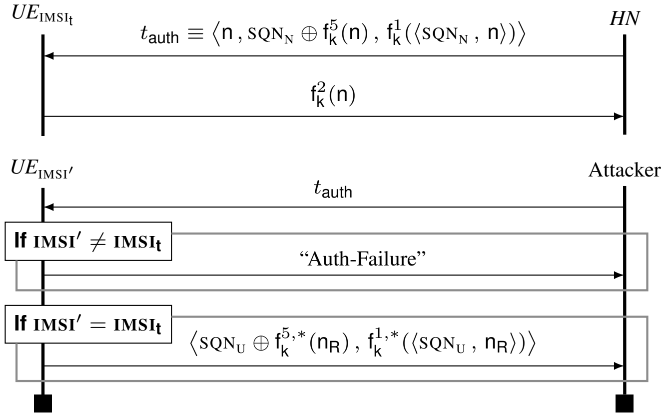


Fig. 3. The Failure Message Attack by [4]

This necessitates an active attacker with its own base station. At the time, this required specialized hardware, and was believed to be too expensive. This is no longer the case, and can be done for a few hundreds dollars (see [25]).

B. *The Failure Message Attack*

In [4], Arapinis et al. propose to use an asymmetric encryption to protect against the IMSI-catcher attack: each *UE* carries the public-key of its corresponding *HN*, and uses it to encrypt its permanent identity. This is basically the solution that was adopted by 3GPP for the 5G version of AKA. Interestingly, they show that this is not enough to ensure privacy, and give a linkability attack that does not rely on the identification message sent by *UE*. While their attack is against the 3G-AKA protocol, it is applicable to the 5G-AKA protocol.

a) The Attack: The attack is depicted in Fig. 3, and works in two phases. First, the adversary eavesdrops a successful run of the protocol between the *HN* and the target *UE* with identity IMSI_{t}, and stores the authentication message t_{auth} sent by *HN*. In a second phase, the attacker $\mathcal{A}$ tries to determine whether a *UE* with identity IMSI' is the initial *UE* (i.e. whether $\text{IMSI}' = \text{IMSI}_{\mathsf{t}}$). To do this, $\mathcal{A}$ initiates a new session of the protocol and replays the message t_{auth}. If $\text{IMSI}' \neq \text{IMSI}_{\mathsf{t}}$, then the mac test fails, and $UE_{\text{IMSI}'}$ answers "Auth-Failure". If $\text{IMSI}' = \text{IMSI}_{\mathsf{t}}$, then the mac test succeeds but the range test fails, and $UE_{\text{IMSI}'}$ sends a re-synchronization message.

The adversary can distinguish between the two messages, and therefore knows if it is interacting with the original or a different *UE*. Moreover, the second phase of the attack can

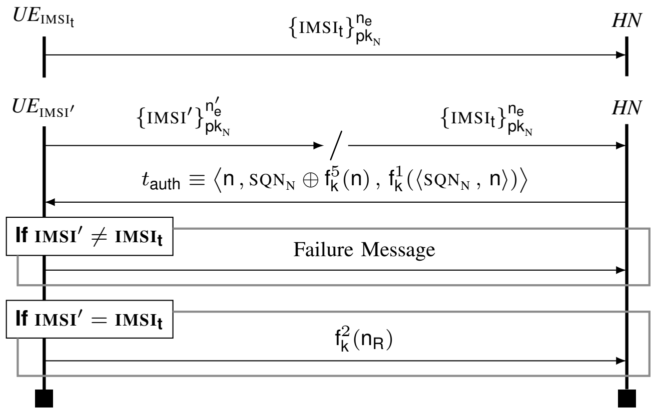


Fig. 4. The Encrypted IMSI Replay Attack by [6]

be repeated every time the adversary wants to check for the presence of the tracked user IMSI_t in its vicinity.

b) Proposed Fix: To protect against the failure message attack, the authors of [4] propose that the *UE* encrypts both error messages using the public key pk_N of the *HN*, making them indistinguishable. To the adversary, there is no distinctions between an authentication and a de-synchronization failure. The fixed AKA protocol, *without the identifying message* $\{\text{IMSI}\}^{n_e}_{pk_N}$, was formally checked in the symbolic model using the PROVERIF tool. Because this message was omitted in the model, an attack was missed. We present this attack next.

C. The Encrypted IMSI Replay Attack

In [6], Fouque et al. give an attack against the fixed AKA proposed by Arapinis et al. in [4]. Their attack, described in Fig. 4, uses the fact the identifying message $\{\text{IMSI}_t\}^{n_e}_{pk_N}$ in the proposed AKA protocol by Arapinis et al. can be replayed.

In a first phase, the attacker $\mathcal{A}$ eavesdrops and stores the identifying message $\{\text{IMSI}_t\}^{n_e}_{pk_N}$ of an honest session between the user UE_{IMSI_t} it wants to track and the *HN*. Then, every time $\mathcal{A}$ wants to determine whether some user $UE_{\text{IMSI}'}$ is the tracked user UE_{IMSI_t}, it intercepts the identifying message $\{\text{IMSI}'\}^{n'_e}_{pk_N}$ sent by $UE_{\text{IMSI}'}$, and replaces it with the stored message $\{\text{IMSI}_t\}^{n_e}_{pk_N}$. Finally, $\mathcal{A}$ lets the protocol continue without further tampering. We have two possible outcomes:

- If $\text{IMSI}' \neq \text{IMSI}_t$ then the message t_{auth} sent by *HN* is mac-ed using the wrong key, and the *UE* rejects the message. Hence the attacker observes a failure message.
- If $\text{IMSI}' = \text{IMSI}_t$ then t_{auth} is accepted by $UE_{\text{IMSI}'}$, and the attacker observes a success message.

Therefore the attacker can deduce whether it is interacting with UE_{IMSI_t} or not, which breaks unlinkability.

D. Attack Against The PRIV-AKA Protocol

The authors of [6] then propose the PRIV-AKA protocol, which is a significantly modified version of AKA. The authors claim that their protocol achieves authentication and client unlinkability. But we discovered a de-synchronization attack: it is possible to permanently de-synchronize the *UE* and the *HN*. Our attack uses the fact that in PRIV-AKA, the *HN* sequence number is incremented only upon reception of the confirmation message from the *UE*. Therefore, by intercepting the last message from the *UE*, we can prevent the *HN* from incrementing its sequence number. We now describe the attack.

We run a session of the protocol, but we intercept the last message and store it for later use. Note that the *HN*'s session is not closed. At that point, the *UE* and the *HN* are de-synchronized by one. We re-synchronize them by running a full session of the protocol. We then re-iterate the steps described above: we run a session of the protocol, prevent the last message from arriving at the *HN*, and then run a full session of the protocol to re-synchronize the *HN* and the *UE*. Now the *UE* and the *HN* are synchronized, and we have two stored messages, one for each uncompleted session. We then send the two messages to the corresponding *HN* sessions, which accept them and increment the sequence number. In the end, it is incremented by *two*.

The problem is that the *UE* and the *HN* cannot recover from a de-synchronization by two. We believe that this was missed by the authors of [6][2]. Remark that this attack is also an unlinkability attack. To attack some user UE_{IMSI}'s privacy, we permanently de-synchronize it. Then each time UE_{IMSI} tries to run the PRIV-AKA protocol, it will abort, which allows the adversary to track it.

Remark 1. Our attack requires that the *HN* does not close the first session when we execute the second session. At the end of the attack, before sending the two stored messages, there are two *HN* sessions simultaneously opened for the same *UE*. If the *HN* closes any un-finished sessions when starting a new session with the same *UE*, our attack does not work.

But this make another unlinkability attack possible. Indeed, closing a session because of some later session between the *HN* and the same *UE* reveals a link between the two sessions. We describe the attack. First, we start a session i between a user UE_A and the *HN*, but we intercept and store the last message t_A from the user. Then, we let the *HN* run a full session with some user UE_X. Finally, we complete the initial session i by sending the stored message t_A to the *HN*. Here, we have two cases. If $X = A$, then the *HN* closed the first session when it completed the second. Hence it rejects t_A. If $X \neq A$, then the first session is still opened, and it accepts t_A.

Closing a session may leak information to the adversary. Protocols which aim at providing unlinkability must explicit when sessions can safely be closed. By default, we assume a session stays open. In a real implementation, a timeout *tied to the session* (and not the user identity) could be used to avoid keeping sessions opened forever.

E. Sequence Numbers and Unlinkability

We conjecture that it is not possible to achieve functionality (i.e. honest sessions eventually succeed), authentication and unlinkability at the same time when using a sequence number

[2]"the two sequence numbers may become desynchronized by one step [...]. Further desynchronization is prevented [...]" (p. 266 [6])

based protocol with no random number generation capabilities in the *UE* side. We briefly explain our intuition.

In any sequence number based protocol, the agents may become de-synchronized because they cannot know if their last message has been received. Furthermore, the attacker can cause de-synchronization by blocking messages. The problem is that we have contradictory requirements. On the one hand, to ensure authentication, an agent must reject a replayed message. On the other hand, in order to guarantee unlinkability, an honest agent has to behave the same way when receiving a message from a synchronized agent or from a de-synchronized agent. Since functionality requires that a message from a synchronized agent is accepted, it follows that a message from a de-synchronized agent must be accepted. Intuitively, it seems to us that an honest agent cannot distinguish between a protocol message which is being replayed and an honest protocol message from a de-synchronized agent. It follows that a replayed message should be both rejected and accepted, which is a contradiction.

This is only a conjecture. We do not have a formal statement, or a proof. Actually, it is unclear how to formally define the set of protocols that rely on sequence numbers to achieve authentication. Note however that all requirements can be satisfied simultaneously if we allow *both* parties to generate random challenges in each session (in AKA, only *HN* uses a random challenge). Examples of challenge based unlinkable authentication protocols can be found in [26].

IV. The AKA$^+$ Protocol

We now describe our principal contribution, which is the design of the AKA$^+$ protocol. This is a fixed version of the 5G-AKA protocol offering some form of privacy against an *active* attacker. First, we explicit the efficiency and design constraints. We then describe the AKA$^+$ protocol, and explain how we designed this protocol from 5G-AKA by fixing all the previously described attacks. As we mentioned before, we think unlinkability cannot be achieved under these constraints. Nonetheless, our protocol satisfies some weaker notion of unlinkability that we call σ-unlinkability. This is a new security property that we introduce. Finally, we will show a subtle attack, and explain how we fine-tuned AKA$^+$ to prevent it.

A. *Efficiency and Design Constraints*

We now explicit the protocol design constraints. These constraints are necessary for an efficient, in-expensive to implement and backward compatible protocol. Observe that, in a mobile setting, it is very important to avoid expensive computations as they quickly drain the *UE*'s battery.

a) Communication Complexity: In 5G-AKA, authentication is achieved using only three messages: two messages are sent by the *UE*, and one by the *HN*. We want our protocol to have a similar communication complexity. While we did not manage to use only three messages in all scenarios, our protocol achieves authentication in less than four messages.

b) Cryptographic primitives: We recall that all cryptographic primitives are computed in the *USIM*, where they are implemented in hardware. It follows that using more primitives in the *UE* would make the *USIM* more voluminous and expensive. Hence we restrict AKA$^+$ to the cryptographic primitives used in 5G-AKA: we use only symmetric keyed one-way functions and asymmetric encryption. Notice that the *USIM* cannot do asymmetric *decryption*. As in 5G-AKA, we use some in-expensive functions, e.g. xor, pairs, by-one increments and boolean tests. We believe that relying on the same cryptographic primitives helps ensuring backward compatibility, and would simplify the protocol deployment.

c) Random Number Generation: In 5G-AKA, the *UE* generates at most one nonce per session, which is used to randomize the asymmetric encryption. Moreover, if the *UE* was assigned a GUTI in the previous session then there is no random number generation. Remark that when the *UE* and the *HN* are de-synchronized, the authentication fails and the *UE* sends a re-synchronization message. Since the session fails, no fresh GUTI is assigned to the *UE*. Hence, the next session of the protocol has to conceal the SUPI using $\{\text{SUPI}\}^{\mathsf{n_e}}_{\mathsf{pk_N}}$, which requires a random number generation. Therefore, we constrain our protocol to use at most one random number generation by the *UE* per session, and only if no GUTI has been assigned or if the *UE* and the *HN* have been de-synchronized.

d) Summary: We summarize the constraints for AKA$^+$:

- It must use at most four messages per sessions.
- The *UE* may use only keyed one-way functions and asymmetric *encryption*. The *HN* may use these functions, plus asymmetric *decryption*.
- The *UE* may generate at most one random number per session, and only if no GUTI is available, or if re-synchronization with the *HN* is necessary.

B. *Key Ideas*

In this section, we present the two key ideas used in the design of the AKA$^+$ protocol.

a) Postponed Re-Synchronization Message: We recall that whenever the *UE* and the *HN* are de-synchronized, the authentication fails and the *UE* sends a re-synchronization message. The problem is that this message can be distinguished from a mac failure message, which allows the attack presented in Section III-B. Since the session fails, no GUTI is assigned to the *UE*, and the next session will use the asymmetric encryption to conceal the SUPI. The first key idea is to piggy-back on the randomized encryption of the *next session* to send a concealed re-synchronization message. More precisely, we replace the message $\{\text{SUPI}\}^{\mathsf{n_e}}_{\mathsf{pk_N}}$ by $\{\langle \text{SUPI}\, ,\, \text{SQN}_\text{U} \rangle\}^{\mathsf{n_e}}_{\mathsf{pk_N}}$. This has several advantages:

- We can remove the re-synchronization message that lead to the unlinkability attack presented in Section III-B. In AKA$^+$, whenever the mac check or the range check fails, the same failure message is sent.
- This does not require more random number generation by the *UE*, since a random number is already being generated to conceal the SUPI in the next session.

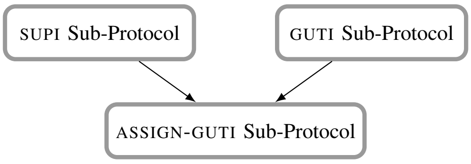


Fig. 5. General Architecture of the AKA$^+$ Protocol

The 3GPP technical specification (see [1], Annex C) requires that the asymmetric encryption used in the 5G-AKA protocol is the ECIES encryption scheme, which is an hybrid encryption scheme. Hybrid encryption schemes use a randomized asymmetric encryption to conceal a temporary key. This key is then used to encrypt the message using a symmetric encryption, which is in-expensive. Hence encrypting the pair $\langle \text{SUPI}, \text{SQN}_\text{U}\rangle$ is almost as fast as encrypting only SUPI, and requires the *UE* to generate the same amount of randomness.

b) HN Challenge Before Identification: To prevent the Encrypted IMSI Replay Attack of Section III-C, we add a random challenge n from the *HN*. The *UE* initiates the protocol by requesting a challenge without identifying itself. When requested, the *HN* generates and sends a fresh challenge n to the *UE*, which includes it in its response by mac-ing it with the SUPI using a symmetric one-way function Mac^1 with key $\mathsf{k}_\mathsf{m}^{\text{ID}}$. The *UE* response is now:

$$\left\langle \{\langle \text{SUPI}, \text{SQN}_\text{U}\rangle\}_{\mathsf{pk}_\text{N}}^{\mathsf{n_e}}, \mathsf{Mac}^1_{\mathsf{k}_\mathsf{m}^{\text{ID}}}(\langle \{\langle \text{SUPI}, \text{SQN}_\text{U}\rangle\}_{\mathsf{pk}_\text{N}}^{\mathsf{n_e}}, \mathsf{n}\rangle)\right\rangle$$

This challenge is only needed when the encrypted permanent identity is used. If the *UE* uses a temporary identity GUTI, then we do not need to use a random challenge. Indeed, temporary identities can only be used once before being discarded, and are therefore not subject to replay attacks. By consequence we split the protocol in two sub-protocols:

- The SUPI sub-protocol uses a random challenge from the *HN*, encrypts the permanent identity and allows to re-synchronize the *UE* and the *HN*.
- The GUTI sub-protocol is initiated by the *UE* using a temporary identity.

In the SUPI sub-protocol, the *UE*'s answer includes the challenge. We use this to save one message: the last confirmation step from the *UE* is not needed, and is removed. The resulting sub-protocol has four messages. Observe that the GUTI sub-protocol is faster, since it uses only three messages.

C. Architecture and States

Instead of a monolithic protocol, we have three sub-protocols: the SUPI and GUTI sub-protocols, which handle authentication; and the ASSIGN-GUTI sub-protocol, which is run after authentication has been achieved and assigns a fresh temporary identity to the *UE*. A full session of the AKA$^+$ protocol comprises a session of the SUPI or GUTI sub-protocols, followed by a session of the ASSIGN-GUTI sub-protocol. This is graphically depicted in Fig. 5.

Since the GUTI sub-protocol uses only three messages and does not require the *UE* to generate a random number or compute an asymmetric encryption, it is faster than the SUPI sub-protocol. By consequence, the *UE* should always use the GUTI sub-protocol if it has a temporary identity available.

The *HN* runs concurrently an arbitrary number of sessions, but a subscriber cannot run more than one session at the same time. Of course, sessions from *different* subscribers may be concurrently running. We associate a unique integer, the session number, to every session, and we use $HN(j)$ and $UE_{\text{ID}}(j)$ to refer to the j-th session of, respectively, the *HN* and the *UE* with identity ID.

a) One-Way Functions: We separate functions that are used only for confidentiality from functions that are also used for integrity. We have two confidentiality functions f and f^r, which use the key k, and five integrity functions Mac^1–Mac^5, which use the key k_m. We require that f and f^r (resp. Mac^1–Mac^5) satisfy jointly the PRF assumption.

This is a new assumption, which requires that these functions are *simultaneously* computationally indistinguishable from random functions.

Definition 1 (Jointly PRF Functions)**.** *Let $H_1(\cdot,\cdot),\dots,H_n(\cdot,\cdot)$ be a finite family of keyed hash functions from $\{0,1\}^*\times\{0,1\}^\eta$ to $\{0,1\}^\eta$. The functions $H_1,\dots,H_n$ are* Jointly Pseudo Random Functions *if, for any PPTM adversary $\mathcal{A}$ with access to oracles $\mathcal{O}_{f_1},\dots,\mathcal{O}_{f_n}$:*

$$|\mathbf{Pr}(k:\ \mathcal{A}^{\mathcal{O}_{H_1(\cdot,k)},\dots,\mathcal{O}_{H_n(\cdot,k)}}(1^\eta)=1)- \\ \mathbf{Pr}(g_1,\dots,g_n:\ \mathcal{A}^{\mathcal{O}_{g_1(\cdot)},\dots,\mathcal{O}_{g_n(\cdot)}}(1^\eta)=1)|$$

is negligible, where:

- *k is drawn uniformly in $\{0,1\}^\eta$.*
- *$g_1,\dots,g_n$ are drawn uniformly in the set of all functions from $\{0,1\}^*$ to $\{0,1\}^\eta$.*

Observe that if $H_1,\dots,H_n$ are jointly PRF then, in particular, every individual H_i is a PRF.

Remark 2*.* While this is a non-usual assumption, it is simple to build a set of functions $H_1,\dots,H_n$ which are jointly PRF from a single PRF H. For example, let $\mathsf{tag}_1,\dots,\mathsf{tag}_n$ be non-ambiguous tags, and let $H_i(m,\mathsf{k}) = H(\mathsf{tag}_i(m),\mathsf{k})$. Then, $H_1,\dots,H_n$ are jointly PRF whenever H is a PRF (see [24]).

b) UE Persistent State: Each UE_{ID} with identity ID has a state $\mathsf{state}_\text{U}^{\text{ID}}$ persistent across sessions. It contains the following immutable values: the permanent identity $\text{SUPI} = \text{ID}$, the confidentiality key k^{ID}, the integrity key $\mathsf{k}_\mathsf{m}^{\text{ID}}$ and the *HN*'s public key pk_N. The states also contain mutable values: the sequence number SQN_U, the temporary identity GUTI_U and the boolean $\mathsf{valid\text{-}guti}_\text{U}$. We have $\mathsf{valid\text{-}guti}_\text{U} = \mathsf{false}$ whenever no valid temporary identity is assigned to the *UE*. Finally, there are mutable values that are not persistent across sessions. E.g. $\mathsf{b\text{-}auth}_\text{U}$ stores *HN*'s random challenge, and $\mathsf{e\text{-}auth}_\text{U}$ stores *HN*'s random challenge *when the authentication is successful*.

c) HN Persistent State: The *HN* state state_N contains the secret key sk_N corresponding to the public key pk_N. Also, for every subscriber with identity ID, it stores the keys k^{ID} and $\mathsf{k}_\mathsf{m}^{\text{ID}}$, the permanent identity $\text{SUPI} = \text{ID}$, the *HN* version of the sequence number $\text{SQN}_\text{N}^{\text{ID}}$ and the temporary identity $\text{GUTI}_\text{N}^{\text{ID}}$. It

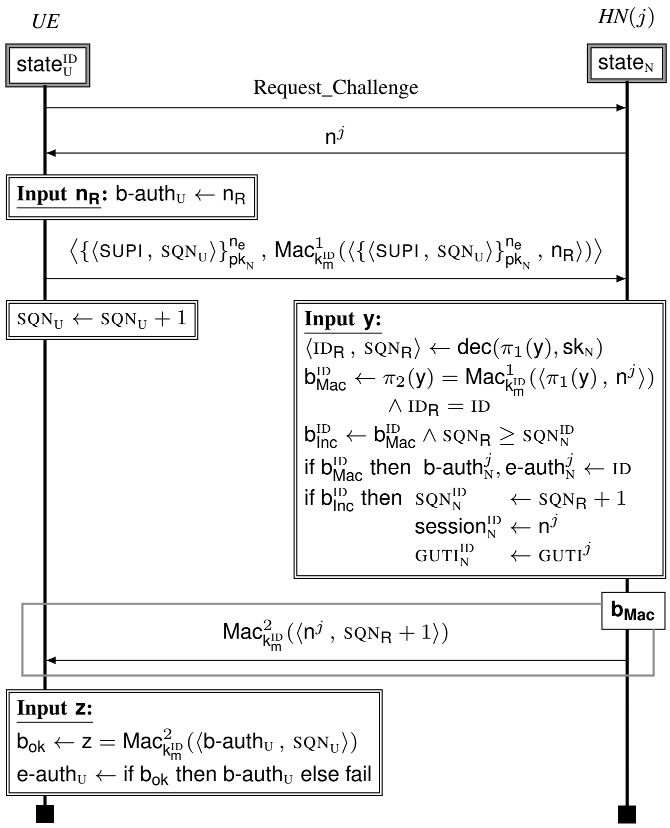


Fig. 6. The SUPI Sub-Protocol of the AKA$^+$ Protocol

stores in $\mathsf{session}^{ID}_N$ the random challenge of the last session that was either a successful SUPI session which modified the sequence number, or a GUTI session which authenticated ID. This is used to detect and prevent some subtle attacks, which we present later. Finally, every session $HN(j)$ stores in $\mathsf{b\text{-}auth}^j_N$ the identity claimed by the *UE*, and in $\mathsf{e\text{-}auth}^j_N$ the identity of the *UE* it authenticated.

D. The SUPI, GUTI *and* ASSIGN-GUTI *Sub-Protocols*

We describe honest executions of the three sub-protocols of the AKA$^+$ protocol. An honest execution is an execution where the adversary dutifully forwards the messages without tampering. Each execution is between a *UE* and $HN(j)$.

a) The SUPI *Sub-Protocol:* This protocol uses the *UE*'s permanent identity, re-synchronizes the *UE* and the *HN* and is expensive to run. The protocol is sketched in Fig. 6.

The *UE* initiates the protocol by requesting a challenge from the network. When asked, $HN(j)$ sends a fresh random challenge n^j. After receiving n^j, the *UE* stores it in $\mathsf{b\text{-}auth}_U$, and answers with the encryption of its permanent identity together with the current value of its sequence number, using the *HN* public key pk_N. It also includes the mac of this encryption and of the challenge, which yields the message:

$$\langle \{\langle \mathsf{SUPI}, \mathrm{SQN}_U \rangle\}^{\mathsf{n_e}}_{\mathsf{pk}_N}, \mathsf{Mac}^1_{\mathsf{k}^{ID}_{\mathsf{m}}}(\langle \{\langle \mathsf{SUPI}, \mathrm{SQN}_U \rangle\}^{\mathsf{n_e}}_{\mathsf{pk}_N}, \mathsf{n}^j \rangle)\rangle$$

Then the *UE* increments its sequence number by one. When it gets this message, the *HN* retrieves the pair $\langle \mathrm{SUPI}, \mathrm{SQN}_U \rangle$ by decrypting the encryption using its secret key sk_N. For every identity ID, it checks if $\mathrm{SUPI} = \mathrm{ID}$ and if the mac is correct. If this is the case, *HN* authenticated ID, and it stores ID in $\mathsf{b\text{-}auth}^j_N$ and $\mathsf{e\text{-}auth}^j_N$. After having authenticated ID, *HN* checks whether the sequence number SQN_U it received is greater than or equal to SQN^{ID}_N. If this holds, it sets SQN^{ID}_N to $\mathrm{SQN}_U + 1$, stores n^j in $\mathsf{session}^{ID}_N$, generates a fresh temporary identity GUTI^j and stores it into GUTI^{ID}_N. This additional check ensures that the *HN* sequence number is always increasing, which is a crucial property of the protocol.

If the *HN* authenticated ID, it sends a confirmation message $\mathsf{Mac}^2_{\mathsf{k}^{ID}_{\mathsf{m}}}(\langle \mathsf{n}^j, \mathrm{SQN}_U + 1 \rangle)$ to the *UE*. This message is sent even if the received sequence number SQN_U is smaller than SQN^{ID}_N. When receiving the confirmation message, if the mac is valid then the *UE* authenticated the *HN*, and it stores in $\mathsf{e\text{-}auth}_U$ the initial random challenge (which it keeps in $\mathsf{b\text{-}auth}_U$). If the mac test fails, it stores in $\mathsf{e\text{-}auth}_U$ the special value fail.

b) The GUTI *Sub-Protocol:* This protocol uses the *UE*'s temporary identity, requires synchronization to succeed and is inexpensive. The protocol is sketched in Fig. 7.

When $\mathsf{valid\text{-}guti}_U$ is true, the *UE* can initiate the protocol by sending its temporary identity GUTI_U. The *UE* then sets $\mathsf{valid\text{-}guti}_U$ to false to guarantee that this temporary identity is not used again. When receiving a temporary identity x, *HN* looks if there is an ID such that GUTI^{ID}_N is equal to x and is not UnSet. If the temporary identity belongs to ID, it sets GUTI^{ID}_N to UnSet and stores ID in $\mathsf{b\text{-}auth}^j_N$. Then it generates a random challenge n^j, stores it in $\mathsf{session}^{ID}_N$, and sends it to the *UE*, together with the xor of the sequence number SQN^{ID}_N with $\mathsf{f}_{\mathsf{k}^{ID}}(\mathsf{n}^j)$, and a mac:

$$\langle \mathsf{n}^j, \mathrm{SQN}^{ID}_N \oplus \mathsf{f}_{\mathsf{k}^{ID}}(\mathsf{n}^j), \mathsf{Mac}^3_{\mathsf{k}^{ID}_{\mathsf{m}}}(\langle \mathsf{n}^j, \mathrm{SQN}^{ID}_N, \mathrm{GUTI}^{ID}_N \rangle)\rangle$$

When it receives this message, the *UE* retrieves the challenge n^j at the beginning of the message, computes $\mathsf{f}_{\mathsf{k}^{ID}}(\mathsf{n}^j)$ and uses this value to unconceal the sequence number SQN^{ID}_N. It then computes $\mathsf{Mac}^3_{\mathsf{k}^{ID}_{\mathsf{m}}}(\langle \mathsf{n}^j, \mathrm{SQN}^{ID}_N, \mathrm{GUTI}_U \rangle)$ and compares it to the mac received from the network. If the macs are not equal, or if the range check $\mathsf{range}(\mathrm{SQN}_U, \mathrm{SQN}^{ID}_N)$ fails, it puts fail into $\mathsf{b\text{-}auth}_U$ and $\mathsf{e\text{-}auth}_U$ to record that the authentication was not successful. If both tests succeed, it stores in $\mathsf{b\text{-}auth}_U$ and $\mathsf{e\text{-}auth}_U$ the random challenge, increments SQN_U by one and sends the confirmation message $\mathsf{Mac}^4_{\mathsf{k}^{ID}_{\mathsf{m}}}(\mathsf{n}^j)$. When receiving this message, the *HN* verifies that the mac is correct. If this is the case then the *HN* authenticated the *UE*, and stores ID into $\mathsf{e\text{-}auth}^{ID}_N$. Then, *HN* checks whether $\mathsf{session}^{ID}_N$ is still equal to the challenge n^j stored in it at the beginning of the session. If this is true, the *HN* increments SQN^{ID}_N by one, generates a fresh temporary identity GUTI^j and stores it into GUTI^{ID}_N.

c) The ASSIGN-GUTI *Sub-Protocol:* The ASSIGN-GUTI sub-protocol is run after a successful authentication, regardless of the authentication sub-protocol used. It assigns a fresh temporary identity to the *UE* to allow the next AKA$^+$ session to run the faster GUTI sub-protocol. It is depicted in Fig. 8.

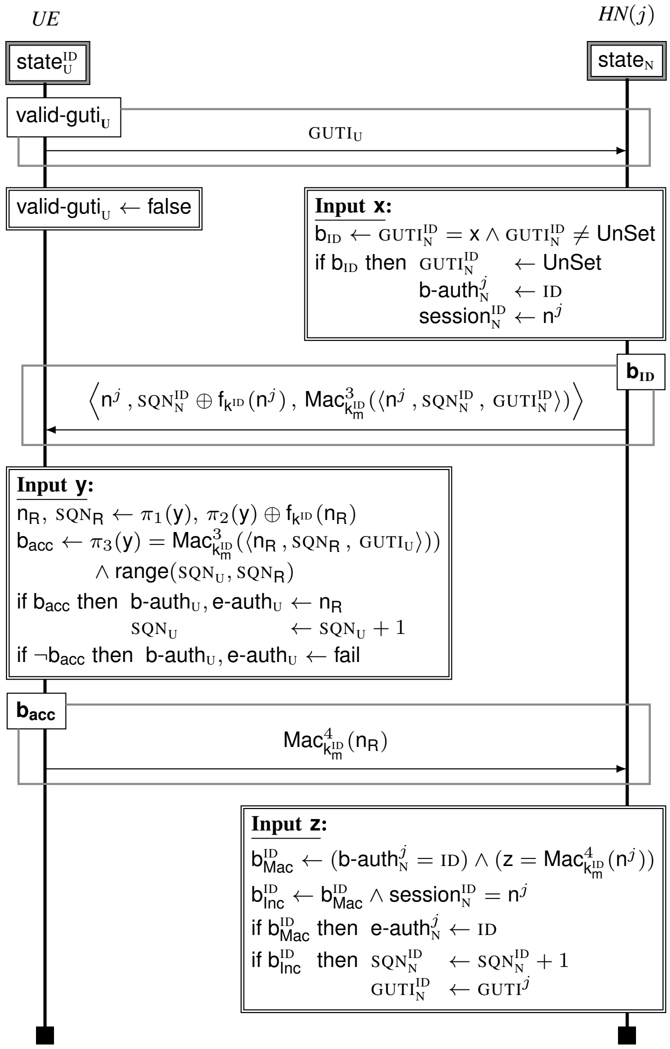


Fig. 7. The GUTI Sub-Protocol of the AKA$^+$ Protocol

The *HN* conceals the temporary identity GUTI^j generated by the authentication sub-protocol by xoring it with $\mathsf{f}^{\mathsf{r}}_{\mathsf{k}^{\mathrm{ID}}}(\mathsf{n}^j)$, and macs it. When receiving this message, *UE* unconceals the temporary identity $\mathrm{GUTI}_{\mathrm{N}}^{\mathrm{ID}}$ by xoring its first component with $\mathsf{f}^{\mathsf{r}}_{\mathsf{k}_\mathsf{m}^{\mathrm{ID}}}(\text{e-auth}_{\mathrm{U}})$ (since $\text{e-auth}_{\mathrm{U}}$ contains the *HN*'s challenge after authentication). Then *UE* checks that the mac is correct and that the authentication was successful. If it is the case, it stores $\mathrm{GUTI}_{\mathrm{N}}^{\mathrm{ID}}$ in $\mathrm{GUTI}_{\mathrm{U}}$ and sets $\text{valid-guti}_{\mathrm{U}}$ to true.

V. Unlinkability

We now define the unlinkability property we use, which is inspired from [22] and Vaudenay's privacy [23].

a) Definition: The property is defined by a game in which an adversary tries to link together some subscriber's sessions. The adversary is a PPTM which interacts, through oracles, with N different subscribers with identities $\mathrm{ID}_1, \dots, \mathrm{ID}_N$, and with the *HN*. The adversary cannot use a subscriber's permanent identity to refer to it, as it may not know it. Instead, we associate a virtual handler vh to any subscriber currently

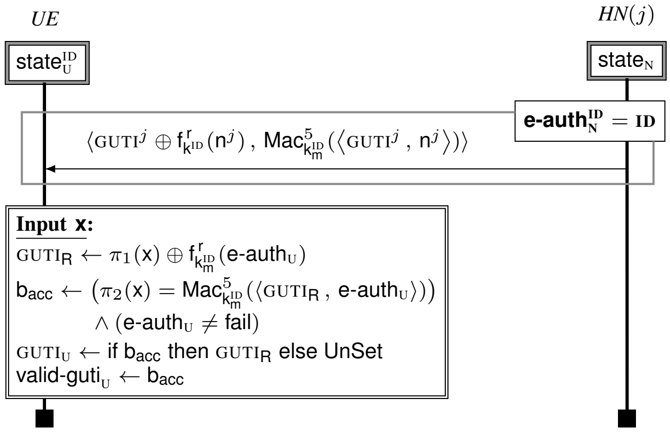


Fig. 8. The ASSIGN-GUTI Sub-Protocol of the AKA$^+$ Protocol

running a session of the protocol. We maintain a list l_{free} of all subscribers that are ready to start a session. We now describe the oracles $\mathcal{O}_b$:

- `StartSession()`: starts a new *HN* session and returns its session number j.
- `SendHN`(m, j) (resp. `SendUE`(m, vh)): sends the message m to $HN(j)$ (resp. the *UE* associated with vh), and returns $HN(j)$ (resp. vh) answer.
- `ResultHN`(j) (resp. `ResultUE`(vh)): returns true if $HN(j)$ (resp. the *UE* associated with vh) has made a successful authentication.
- `DrawUE`$(\mathrm{ID}_{i_0}, \mathrm{ID}_{i_1})$: checks that ID_{i_0} and ID_{i_1} are both in l_{free}. If that is the case, returns a new virtual handler pointing to ID_{i_b}, depending on an internal secret bit b. Then, it removes ID_{i_0} and ID_{i_1} from l_{free}.
- `FreeUE`(vh): makes the virtual handler vh no longer valid, and adds back to l_{free} the two identities that were removed when the virtual handler was created.

We recall that a function is negligible if and only if it is asymptotically smaller than the inverse of any polynomial. An adversary $\mathcal{A}$ interacting with $\mathcal{O}_b$ is winning the q-unlinkability game if: $\mathcal{A}$ makes less than q calls to the oracles; and it can guess the value of the internal bit b with a probability better than $1/2$ by a non-negligible margin, i.e. if the following quantity is non negligible in η:

$$\left|2 \times \mathbf{Pr}\left(b : \mathcal{A}^{\mathcal{O}_b}(1^\eta) = b\right) - 1\right|$$

Finally, a protocol is q-unlinkable if and only if there are no winning adversaries against the q-unlinkability game.

b) Corruption: In [22], [23], the adversary is allowed to corrupt some tags using a `Corrupt` oracle. Several classes of adversary are defined by restricting its access to the corruption oracle. A *strong* adversary has unrestricted access, a *destructive* adversary can no longer use a tag after corrupting it (it is destroyed), a *forward* adversary can only follow a `Corrupt` call by further `Corrupt` calls, and finally a *weak* adversary cannot use `Corrupt` at all. A protocol is $\mathcal{C}$ unlinkable if no

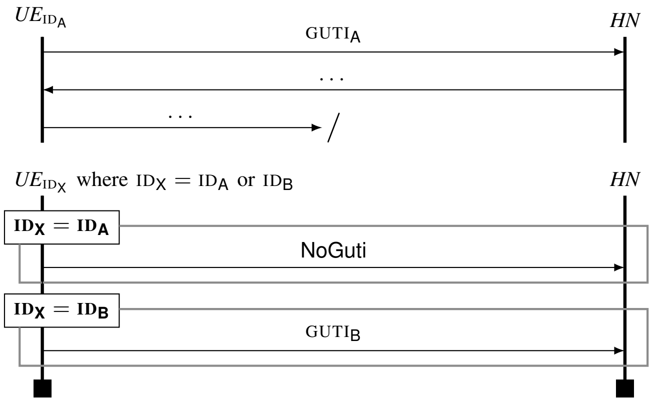


Fig. 9. Consecutive GUTI Sessions of AKA^+ Are Not Unlinkable.

adversary in $\mathcal{C}$ can win the unlinkability game. Clearly, we have the following relations:

$$\textit{strong} \Rightarrow \textit{destructive} \Rightarrow \textit{forward} \Rightarrow \textit{weak}$$

The 5G-AKA protocol does not provide forward secrecy: indeed, obtaining the long-term secret of a *UE* allows to decrypt all its past messages. By consequence, the best we can hope for is *weak* unlinkability. Since such adversaries cannot call `Corrupt`, we removed the oracle from our definition.

c) Wide Adversary: Note that the adversary knows if the protocol was successful or not using the `ResultUE` and `ResultHN` oracles (such an adversary is called *wide* in Vaudenay's terminology [23]). Indeed, in an authenticated key agreement protocol, this information is always available to the adversary: if the key exchange succeeds then it is followed by another protocol using the newly established key; while if it fails then either a new key-exchange session is initiated, or no message is sent. Hence the adversary knows if the key exchange was successful by passive monitoring.

A. σ-Unlinkability

In accord with our conjecture in Section III-E, the AKA^+ protocol is not unlinkable. Indeed, an adversary $\mathcal{A}$ can easily win the linkability game. First, $\mathcal{A}$ ensures that ID_A and ID_B have a valid temporary identity assigned: $\mathcal{A}$ calls $\texttt{DrawUE}(ID_A, ID_A)$ to obtain a virtual handler for ID_A, and runs a SUPI and ASSIGN-GUTI sessions between ID_A and the *HN* with no interruptions. This assigns a temporary identity to ID_A. We use the same procedure for ID_B.

Then, $\mathcal{A}$ executes the attack described in Fig. 9. It starts a GUTI session with ID_A, and intercepts the last message. At that point, ID_A no longer has a temporary identity, while ID_B still does. Then, it calls $\texttt{DrawUE}(ID_A, ID_B)$, which returns a virtual handler vh to ID_A or ID_B. The attacker then start a new GUTI session with vh. If vh is a handler for ID_A, the *UE* returns NoGuti. If vh aliases ID_B, the *UE* returns the temporary identity $GUTI_A$. The adversary $\mathcal{A}$ can distinguish between these two cases, and therefore wins the game.

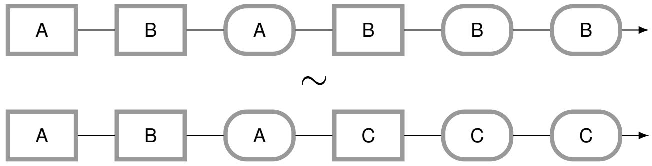


Fig. 10. Two indistinguishable executions. Square (resp. round) nodes are executions of the SUPI (resp. GUTI) sub-protocol. Each time the SUPI sub-protocol is used, we can change the subscriber's identity.

a) σ-unlinkability: To prevent this, we want to forbid `DrawUE` to be called on de-synchronized subscribers. We do this by modifying the state of the user chosen by `DrawUE`. We let σ be an update on the state of the subscribers. We then define the oracle $\texttt{DrawUE}_\sigma(ID_{i_0}, ID_{i_1})$: it checks that ID_A and ID_B are both free, then *applies the update σ* to ID_{i_b}'s state, and returns a new virtual handler pointing to ID_{i_b}. The (q, σ)-unlinkability game is the q-unlinkability game in which we replace `DrawUE` with $\texttt{DrawUE}_\sigma$. A protocol is (q, σ)-unlinkable if and only if there is no winning adversary against the (q, σ)-unlinkability game. Finally, a protocol is σ-unlinkable if it is (q, σ)-unlinkable for any q.

b) Application to AKA^+: The privacy guarantees given by the σ-unlinkability depend on the choice of σ. The idea is to choose a σ that allows to establish privacy in *some scenarios* of the standard unlinkability game[3].

We illustrate this on the AKA^+ protocol. Let $\sigma_{\mathsf{ul}} = \mathsf{valid\text{-}guti}_{\mathsf{U}} \mapsto \mathsf{false}$ be the function that makes the *UE*'s temporary identity not valid. This simulates the fact that the GUTI has been used and is no longer available. If the *UE*'s temporary identity is not valid, then it can only run the SUPI sub-protocol. Hence, if the AKA^+ protocol is σ_{ul}-unlinkable, then no adversary can distinguish between a normal execution and an execution where we change the identity of a subscriber each time it runs the SUPI sub-protocol. We give in Fig. 10 an example of such a scenario. We now state our main result:

Theorem 1. *The AKA^+ protocol is σ_{ul}-unlinkable for an arbitrary number of agents and sessions when the asymmetric encryption $\{\_\}_{\_}^{\_}$ is IND-CCA1 secure and f and f^{r} (resp. Mac^1–Mac^5) satisfy jointly the PRF assumption.*

This result is shown later in the paper. Still, the intuition is that no adversary can distinguish between two sessions of the SUPI protocol. Moreover, the SUPI protocol has two important properties. First, it re-synchronizes the user with the *HN*, which prevents the attacker from using any prior de-synchronization. Second, the AKA^+ protocol is designed in such a way that no message sent by the *UE* before a successful SUPI session can modify the *HN*'s state after the SUPI session. Therefore, any time the SUPI protocol is run, we get a "clean slate" and we can change the subscriber's identity. Note that we have a trade-off between efficiency and privacy: the SUPI protocol is more expensive to run, but provides more privacy.

[3]Remark that when σ is the empty state update, the σ-unlinkability and unlinkability properties coincide.

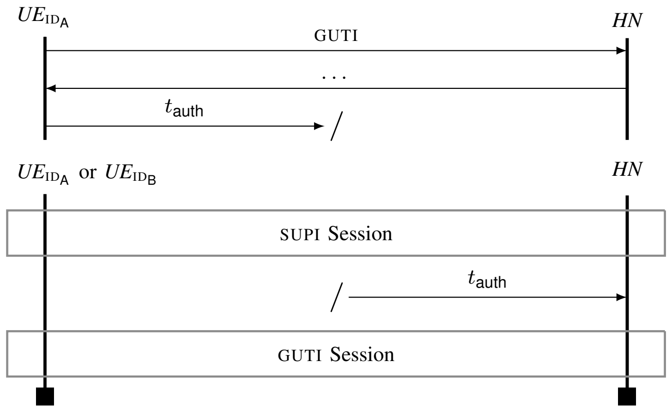


Fig. 11. A Subtle Attack Against The $\text{AKA}^{+}_{\text{no-inc}}$ Protocol

B. A Subtle Attack

We now explain what is the role of $\mathsf{session}_{\text{N}}^{\text{ID}}$, and how it prevents a subtle attack against the σ_{ul}-unlinkability of AKA^{+}. We let $\text{AKA}^{+}_{\text{no-inc}}$ be the AKA^{+} protocol where we modify the GUTI sub-protocol we described in Fig. 7: in the state update of the *HN*'s last input, we remove the check $\mathsf{session}_{\text{N}}^{\text{ID}} = \mathsf{n}^j$ (i.e. $\mathsf{b}_{\mathsf{Inc}}^{\text{ID}} = \mathsf{b}_{\mathsf{Mac}}^{\text{ID}}$). The attack is described in Fig. 11.

First, we run a session of the GUTI sub-protocol between UE_{ID_A} and the *HN*, but we do not forward the last message t_{auth} to the *HN*. We then call $\mathtt{DrawUE}_{\sigma_{\mathsf{ul}}}(\text{ID}_A, \text{ID}_B)$, which returns a virtual handler vh to ID_A or ID_B. We run a full session using the SUPI sub-protocol with vh, and then send the message t_{auth} to the *HN*. We can check that, because we removed the condition $\mathsf{session}_{\text{N}}^{\text{ID}} = \mathsf{n}^j$ from $\mathsf{b}_{\mathsf{Inc}}^{\text{ID}}$, this message causes the *HN* to increment $\text{SQN}_{\text{N}}^{\text{ID}_A}$ by one. At that point, UE_{ID_A} is de-synchronized but UE_{ID_B} is synchronized. Finally, we run a session of the GUTI sub-protocol. The session has two possible outcomes: if vh aliases to A then it fails, while if vh aliases to B, it succeeds. This leads to an attack.

When we removed the condition $\mathsf{session}_{\text{N}}^{\text{ID}} = \mathsf{n}^j$, we broke the "clean slate" property of the SUPI sub-protocol: we can use a message from a session that started *before* the SUPI session to modify the state *after* the SUPI session. $\mathsf{session}_{\text{N}}^{\text{ID}}$ allows to detect whether another session has been executed since the current session started, and to prevent the update of the sequence number when this is the case.

VI. Modeling in The Bana-Comon Logic

We prove Theorem 1 using the Bana-Comon model introduced in [18]. This is a first order logic, in which protocol messages are represented by terms using special function symbols for the adversary's inputs. It has only one predicate, $\sim$, which represents computational indistinguishability. To use this model, we first build a set of axioms Ax specifying what the adversary *cannot* do. This set of axiom comprises computationally valid properties, cryptographic hypotheses and implementation assumptions. Then, given a protocol and a security property, we compute a formula ϕ expressing the protocol security. Finally, we show that the security property ϕ can be deduced from the axioms Ax. If this is the case, this entails *computational security*.

A. Syntax and Semantics

We quickly recall the syntax and semantics of the logic.

a) Terms: Terms are built using function symbols in $\mathcal{F}$, names in $\mathcal{N}$ (representing random samplings) and variables in $\mathcal{X}$. The set $\mathcal{F}$ of function symbols contains a countable set of *adversarial* function symbols $\mathcal{G}$, which represent the adversary inputs, and protocol function symbols. The protocol function symbols are the functions used in the protocol, e.g. the pair $\langle \_ , \_ \rangle$, the i-th projection π_i, encryption $\{\_\}_{\_}^{\_}$, decryption $\mathsf{dec}(\_, \_)$, $\mathsf{if\_then\_else\_}$, true, false, equality $\mathsf{eq}(\_, \_)$, integer greater or equal $\mathsf{geq}(\_, \_)$ and length $\mathsf{len}(\_)$.

b) Formulas: For every integer n, we have one predicate symbol $\sim_n$ of arity $2n$, which represents equivalence between two vectors of terms of length n. We use an infix notation for $\sim_n$, and omit n when not relevant. Formulas are built using the usual Boolean connectives and first-order quantifiers.

c) Semantics: We use the classical semantics of first-order logic. Given an interpretation domain, we interpret terms, function symbols and predicates as, respectively, elements, functions and relations of this domain.

We focus on a particular class of models, called the *computational models* (see [18] for a formal definition). In a computational model $\mathcal{M}_c$, terms are interpreted in the set of PPTMs equipped with a working tape and two random tapes ρ_1, ρ_2. The tape ρ_1 is used for the protocol random values, while ρ_2 is for the adversary's random samplings. The adversary cannot access directly the random tape ρ_1, although it may obtain part of ρ_1 through the protocol messages. A key feature is to let the interpretation of an adversarial function g be *any* PPTM, which soundly models an attacker *arbitrary probabilistic polynomial time computation*. Moreover, the predicates $\sim_n$ are interpreted using *computational indistinguishability* $\approx$. Two families of distributions of bit-string sequences $(m_\eta)_\eta$ and $(m'_\eta)_\eta$, indexed by η, are indistinguishable iff for every PPTM $\mathcal{A}$ with random tape ρ_2, the following quantity is negligible in η:

$$\big|\, \mathbf{Pr}(\rho_1, \rho_2 : \ \mathcal{A}(m_\eta(\rho_1, \rho_2), \rho_2) = 1) - \mathbf{Pr}(\rho_1, \rho_2 : \ \mathcal{A}(m'_\eta(\rho_1, \rho_2), \rho_2) = 1) \,\big|$$

B. Modeling of the AKA^{+} Protocol States and Messages

We now use the Bana-Comon logic to model the σ_{ul}-unlinkability of the AKA^{+} protocol. We consider a setting with N identities $\text{ID}_1, \dots, \text{ID}_N$, and we let $\mathcal{S}_{\mathsf{id}}$ be the set of all identities. To improve readability, protocol descriptions often omit some details. For example, in Section IV we sometimes omitted the description of the error messages. The failure message attack of [4] demonstrates that such details may be crucial for security. An advantage of the Bana-Comon model is that it requires us to fully formalize the protocol, and to make all assumptions explicit.

a) Symbolic State: For every identity $\text{ID} \in \mathcal{S}_{\text{id}}$, we use several variables to represent UE_{ID}'s state. E.g., $\text{SQN}_{\text{U}}^{\text{ID}}$ and $\text{GUTI}_{\text{U}}^{\text{ID}}$ store, respectively, UE_{ID}'s sequence number and temporary identity. Similarly, we have variables for *HN*'s state, e.g. $\text{SQN}_{\text{N}}^{\text{ID}}$. We let $\mathcal{S}_{\text{var}}$ be the set of variables used in AKA^+:

$$\bigcup_{\substack{j\in\mathbb{N},\text{A}\in\{\text{U},\text{N}\}\\ \text{ID}\in\mathcal{S}_{\text{id}}}} \left\{ \begin{array}{l} \text{SQN}_{\text{A}}^{\text{ID}}, \text{GUTI}_{\text{A}}^{\text{ID}}, \text{e-auth}_{\text{U}}^{\text{ID}}, \text{b-auth}_{\text{U}}^{\text{ID}}, \text{e-auth}_{\text{N}}^{j} \\ \text{b-auth}_{\text{N}}^{j}, \text{s-valid-guti}_{\text{U}}^{\text{ID}}, \text{valid-guti}_{\text{U}}^{\text{ID}}, \text{session}_{\text{N}}^{\text{ID}} \end{array} \right\}$$

A symbolic state σ is a mapping from $\mathcal{S}_{\text{var}}$ to terms. Intuitively, $\sigma(\mathsf{x})$ is a term representing (the distribution of) the value of x.

Example 1. To avoid confusion with the *semantic* equality $=$, we use $\equiv$ to denote *syntactic* equality. Then, we can express the fact that $\text{GUTI}_{\text{U}}^{\text{ID}}$ is unset in a symbolic state σ by having $\sigma(\text{GUTI}_{\text{U}}^{\text{ID}}) \equiv \mathsf{UnSet}$. Also, given a state σ, we can state that σ' is the state σ in which we incremented $\text{SQN}_{\text{U}}^{\text{ID}}$ by having $\sigma'(\mathsf{x})$ be the term $\sigma(\text{SQN}_{\text{U}}^{\text{ID}}) + 1$ if x is $\text{SQN}_{\text{U}}^{\text{ID}}$, and $\sigma(\mathsf{x})$ otherwise.

b) Symbolic Traces: We explain how to express (q, σ_{ul})-unlinkability in the BC model. In the (q, σ_{ul})-unlinkability game, the adversary chooses dynamically which oracle it wants to call. This is not convenient to use in proofs, as we do not know statically the i-th action of the adversary. We prefer an alternative point-of-view, in which the trace of oracle calls is fixed (w.l.o.g., as shown later in Proposition 1). Then, there are no winning adversaries against the σ_{ul}-unlinkability game with a fixed trace of oracle calls if the adversary's interactions with the oracles when $b = 0$ are indistinguishable from the interactions with the oracles when $b = 1$.

We use the following action identifiers to represent symbolic calls to the oracle of the (q, σ_{ul})-unlinkability game:

- $\text{NS}_{\text{ID}}(j)$ represents a call to $\texttt{DrawUE}_{\sigma_{\text{ul}}}(\text{ID}, \_)$ when $b = 0$ or $\texttt{DrawUE}_{\sigma_{\text{ul}}}(\_, \text{ID})$ when $b = 1$.
- $\text{PU}_{\text{ID}}(j, i)$ (resp. $\text{TU}_{\text{ID}}(j, i)$) is the i-th user message in the session $UE_{\text{ID}}(j)$ of the SUPI (resp. GUTI) sub-protocol.
- $\text{FU}_{\text{ID}}(j)$ is the only user message in the session $UE_{\text{ID}}(j)$ of the ASSIGN-GUTI sub-protocol.
- $\text{PN}(j, i)$ (resp. $\text{TN}(j, i)$) is the i-th network message in the session $HN(j)$ of the SUPI (resp. GUTI) sub-protocol.
- $\text{FN}(j)$ is the only network message in the session $HN(j)$ of the ASSIGN-GUTI sub-protocol.

The remaining oracle calls either have no outputs and do not modify the state (e.g. $\texttt{StartSession}$), or can be simulated using the oracles above. E.g., since the *HN* sends an error message whenever the protocol is not successful, the output of $\texttt{ResultHN}$ can be deduced from the protocol messages.

A *symbolic trace* τ is a finite sequence of action identifiers. We associate, to any execution of the (q, σ_{ul})-unlinkability game with a fixed trace of oracle calls, a pair of symbolic traces (τ_l, τ_r), which corresponds to the adversary's interactions with the oracles when b is, respectively, 0 and 1. We let $\mathcal{R}_{\text{ul}}$ be the set of such pairs of traces.

Example 2. We give the symbolic traces corresponding to the honest execution of AKA^+ between $UE_{\text{ID}}(i)$ and $HN(j)$. If the SUPI protocol is used, we have the trace $\tau_{\text{SUPI}}^{i,j}(\text{ID})$:

$$\text{PU}_{\text{ID}}(i, 0),\ \text{PN}(j, 0), \text{PU}_{\text{ID}}(i, 1), \text{PN}(j, 1), \text{PU}_{\text{ID}}(i, 2), \text{FN}(j), \text{FU}_{\text{ID}}(i)$$

And if the GUTI sub-protocol is used, the trace $\tau_{\text{GUTI}}^{i,j}(\text{ID})$:

$$\text{TU}_{\text{ID}}(i, 0), \text{TN}(j, 0), \text{TU}_{\text{ID}}(i, 1), \text{TN}(j, 1), \text{FN}(j), \text{FU}_{\text{ID}}(i)$$

Which such notations, the left trace τ_l of the attack described in Fig. 11, in which the adversary only interacts with A, is:

$$\text{TU}_{\mathsf{A}}(0, 0), \text{TN}(0, 0), \text{TU}_{\mathsf{A}}(0, 1), \tau_{\text{SUPI}}^{1,1}(\mathsf{A}), \text{TN}(0, 1), \tau_{\text{GUTI}}^{2,2}(\mathsf{A})$$

Similarly, we can give the right trace τ_r in which the adversary interacts with A and B:

$$\text{TU}_{\mathsf{A}}(0, 0), \text{TN}(0, 0), \text{TU}_{\mathsf{A}}(0, 1), \tau_{\text{SUPI}}^{0,1}(\mathsf{B}), \text{TN}(0, 1), \tau_{\text{GUTI}}^{1,2}(\mathsf{B})$$

c) Symbolic Messages: We define, for every action identifier ai, the term representing the output observed by the adversary when ai is executed. Since the protocol is stateful, this term is a function of the prefix trace of actions executed since the beginning. We define by mutual induction, for any symbolic trace $\tau = \tau_0, \mathsf{ai}$ whose last action is ai:

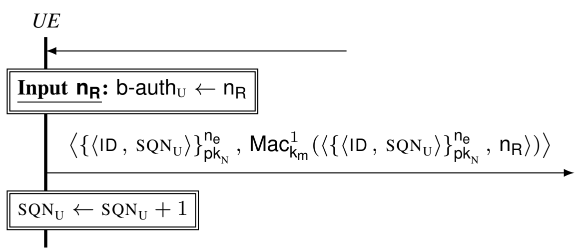
- The term t_τ representing the last message observed during the execution of τ.
- The symbolic state σ_τ representing the state after the execution of τ.
- The frame ϕ_τ representing the sequence of all messages observed during the execution of τ.

Some syntactic sugar: we let $\sigma_\tau^{\text{in}} = \sigma_{\tau_0}$ be the symbolic state before the execution of the last action; and $\phi_\tau^{\text{in}} = \phi_{\tau_0}$ be the sequence of all messages observed during the execution of τ, except for the last message.

The frame ϕ_τ is simply the frame ϕ_τ^{in} extended with t_τ, i.e. $\phi_\tau \equiv \phi_\tau^{\text{in}}, t_\tau$. Moreover the initial frame contains only pk_{N}, i.e. $\phi_\epsilon \equiv \mathsf{pk}_{\text{N}}$. When executing an action ai, only a subset of the symbolic state is modified. For example, if the adversary interacts with UE_{ID} then the state of the *HN* and of all the other users is unchanged. Therefore instead of defining σ_τ, we define the *symbolic state update* σ_τ^{up}, which is a *partial* function from $\mathcal{S}_{\text{var}}$ to terms. Then σ_τ is the function:

$$\sigma_\tau(\mathsf{x}) \equiv \begin{cases} \sigma_\tau^{\text{in}}(\mathsf{x}) & \text{if } \mathsf{x} \notin \mathsf{dom}(\sigma_\tau^{\text{up}}) \\ \sigma_\tau^{\text{up}}(\mathsf{x}) & \text{if } \mathsf{x} \in \mathsf{dom}(\sigma_\tau^{\text{up}}) \end{cases}$$

where dom gives the domain of a function. Now, for every action ai, we define t_τ and σ_τ^{up} using ϕ_τ^{in} and σ_τ^{in}. As an example, we describe the second message and state update of the session $UE_{\text{ID}}(j)$ for the SUPI sub-protocol, which corresponds to the action $\text{PU}_{\text{ID}}(j, 1)$. We recall the relevant part of Fig. 6:



First, we need a term representing the value inputted by UE_{ID} from the network. As we have an active adversary, this value can be anything that the adversary can compute using the knowledge it accumulated since the beginning of the protocol. The knowledge of the adversary, or the frame, is the sequence of all messages observed during the execution of τ, except for the last message. This is exactly ϕ_τ^{in}. Finally, we use a special function symbol $g \in \mathcal{G}$ to represent the arbitrary polynomial time computation done by the adversary. This yields the term $g(\phi_\tau^{\mathsf{in}})$, which symbolically represents the input.

We now need to build a term representing the asymmetric encryption of the pair containing the *UE*'s permanent identity ID and its sequence number. The permanent identity ID is simply represented using a constant function symbol ID (we omit the parenthesis ()). UE_{ID}'s sequence number is stored in the variable $\text{SQN}_{\text{U}}^{\text{ID}}$. To retrieve its value, we just do a look-up in the symbolic state $\sigma_\tau^{\mathsf{in}}$, which yields $\sigma_\tau^{\mathsf{in}}(\text{SQN}_{\text{U}}^{\text{ID}})$. Finally, we use the asymmetric encryption function symbol to build the term $t_\tau^{\mathsf{enc}} \equiv \{\langle \text{ID} , \sigma_\tau^{\mathsf{in}}(\text{SQN}_{\text{U}}^{\text{ID}})\rangle\}_{\mathsf{pk}_{\text{N}}}^{\mathsf{n}_{\mathsf{e}}^{j}}$. Notice that the encryption is randomized using a nonce $\mathsf{n}_{\mathsf{e}}^{j}$, and that the freshness of the randomness is guaranteed by indexing the nonce with the session number j. Finally, we can give t_τ and $\sigma_\tau^{\mathsf{up}}$:

$$t_\tau \equiv \langle t_\tau^{\mathsf{enc}} , \mathsf{Mac}^1_{\mathsf{k}_{\mathsf{m}}^{\text{ID}}}(\langle t_\tau^{\mathsf{enc}} , g(\phi_\tau^{\mathsf{in}})\rangle)\rangle$$

$$\sigma_\tau^{\mathsf{up}} \equiv \begin{cases} \text{SQN}_{\text{U}}^{\text{ID}} \mapsto \mathsf{suc}(\sigma_\tau^{\mathsf{in}}(\text{SQN}_{\text{U}}^{\text{ID}})) & \mathsf{e\text{-}auth}_{\text{U}}^{\text{ID}} \mapsto \mathsf{fail} \\ \mathsf{b\text{-}auth}_{\text{U}}^{\text{ID}} \mapsto g(\phi_\tau^{\mathsf{in}}) & \text{GUTI}_{\text{U}}^{\text{ID}} \mapsto \mathsf{UnSet} \\ \mathsf{valid\text{-}guti}_{\text{U}}^{\text{ID}} \mapsto \mathsf{false} & \end{cases}$$

Remark that we omitted some state updates in the description of the protocol in Fig. 6. For example, UE_{ID} temporary identity $\text{GUTI}_{\text{U}}^{\text{ID}}$ is reset when starting the SUPI sub-protocol. In the BC model, these details are made explicit.

The description of t_τ and $\sigma_\tau^{\mathsf{up}}$ for the other actions can be found in Fig. 12 and Fig. 13. Observe that we describe one more message for the SUPI and GUTI protocols than in Section IV . This is because we add one message ($\text{PU}_{\text{ID}}(j, 2)$ for SUPI and $\text{TN}(j, 1)$ for GUTI) for proof purposes, to simulate the `ResultUE` and `ResultHN` oracles. Also, notice that in the GUTI protocol, when *HN* receives a GUTI that is not assigned to anybody, it sends a decoy message to a special dummy identity ID_{dum}.

The following soundness theorem states that security in the BC model implies computationally security:

Proposition 1. *The* AKA$^+$ *protocol is* σ_{ul}*-unlinkable in any computational model satisfying the axioms* Ax *if, for every* $(\tau_l, \tau_r) \in \mathcal{R}_{\mathsf{ul}}$*, we can derive* $\phi_{\tau_l} \sim \phi_{\tau_r}$ *using* Ax*.*

The proof of this result is basically the proof that Fixed Trace Privacy implies Bounded Session Privacy in [27]. We omit the details.

C. Axioms

Using Proposition 1, we know that to prove Theorem 1 we need to derive $\phi_{\tau_l} \sim \phi_{\tau_r}$, for every $(\tau_l, \tau_r) \in \mathcal{R}_{\mathsf{ul}}$, using a set of inference rules Ax. Moreover, we need the axioms Ax to be valid in any computational model where the asymmetric encryption $\{\_\}_{\_}^{\_}$ is IND-CCA1 secure and f and f^{r} (resp. Mac^1–Mac^5) satisfy jointly the PRF assumption.

Remark that the AKA$^+$ protocol described in Section IV is under-specified. E.g., we never specified how the $\langle \_ , \_ \rangle$ function should be implemented. Instead of giving a complex specification of the protocol, we are going to put requirements on AKA$^+$ implementations through the set of axioms Ax. Then, if we can derive $\phi_{\tau_l} \sim \phi_{\tau_r}$ using Ax for every $(\tau_l, \tau_r) \in \mathcal{R}_{\mathsf{ul}}$, we know that any implementation of AKA$^+$ satisfying the axioms Ax is secure.

Our axioms are of two kinds. First, we have *structural axioms*, which are properties that are valid in any computational model. For example, we have axioms stating that $\sim$ is an equivalence relation. Second, we have *implementation axioms*, which reflect implementation assumptions on the protocol functions. For example, we can declare that different identity symbols are never equal by having an axiom $\mathsf{eq}(\text{ID}_1, \text{ID}_2) \sim \mathsf{false}$ for every $\text{ID}_1 \not\equiv \text{ID}_2$. For space reasons, we only describe a few of them here (the full set of axioms Ax is given in [24]).

a) Equality Axioms: If $\mathsf{eq}(s, t) \sim \mathsf{true}$ holds in any computational model then we know that the interpretations of s and t are always equal except for a negligible number of samplings. Let $s \doteq t$ be a shorthand for $\mathsf{eq}(s, t) \sim \mathsf{true}$. We use $\doteq$ to specify functional correctness properties of the protocol function symbols. For example, the following rules state that the i-th projection of a pair is the i-th element of the pair, and that the decryption with the correct key of a

<u>**Case** $\mathsf{ai} = \text{PU}_{\text{ID}}(j, 0)$.</u> $t_\tau \equiv$ Request_Challenge

<u>**Case** $\mathsf{ai} = \text{PN}(j, 0)$.</u> $t_\tau \equiv \mathsf{n}^j$

<u>**Case** $\mathsf{ai} = \text{PU}_{\text{ID}}(j, 1)$.</u> Let $t_\tau^{\mathsf{enc}} \equiv \{\langle \text{ID} , \sigma_\tau^{\mathsf{in}}(\text{SQN}_{\text{U}}^{\text{ID}})\rangle\}_{\mathsf{pk}_{\text{N}}}^{\mathsf{n}_{\mathsf{e}}^{j}}$, then:

$$t_\tau \equiv \langle t_\tau^{\mathsf{enc}} , \mathsf{Mac}^1_{\mathsf{k}_{\mathsf{m}}^{\text{ID}}}(\langle t_\tau^{\mathsf{enc}} , g(\phi_\tau^{\mathsf{in}})\rangle)\rangle$$

$$\sigma_\tau^{\mathsf{up}} \equiv \begin{cases} \text{SQN}_{\text{U}}^{\text{ID}} \mapsto \mathsf{suc}(\sigma_\tau^{\mathsf{in}}(\text{SQN}_{\text{U}}^{\text{ID}})) & \mathsf{e\text{-}auth}_{\text{U}}^{\text{ID}} \mapsto \mathsf{fail} \\ \mathsf{b\text{-}auth}_{\text{U}}^{\text{ID}} \mapsto g(\phi_\tau^{\mathsf{in}}) & \text{GUTI}_{\text{U}}^{\text{ID}} \mapsto \mathsf{UnSet} \\ \mathsf{valid\text{-}guti}_{\text{U}}^{\text{ID}} \mapsto \mathsf{false} & \end{cases}$$

<u>**Case** $\mathsf{ai} = \text{PN}(j, 1)$.</u> Let $t_{\mathsf{dec}} \equiv \mathsf{dec}(\pi_1(g(\phi_\tau^{\mathsf{in}})), \mathsf{sk}_{\text{N}})$, and let:

$$\begin{aligned} \mathsf{accept}_\tau^{\text{ID}_i} &\equiv \mathsf{eq}(\pi_2(g(\phi_\tau^{\mathsf{in}})), \mathsf{Mac}^1_{\mathsf{k}_{\mathsf{m}}^{\text{ID}_i}}(\langle \pi_1(g(\phi_\tau^{\mathsf{in}})) , \mathsf{n}^j\rangle)) \\ &\quad \wedge \mathsf{eq}(\pi_1(t_{\mathsf{dec}}), \text{ID}_i) \\ \mathsf{inc\text{-}accept}_\tau^{\text{ID}_i} &\equiv \mathsf{accept}_\tau^{\text{ID}_i} \wedge \mathsf{geq}(\pi_2(t_{\mathsf{dec}}), \sigma_\tau^{\mathsf{in}}(\text{SQN}_{\text{N}}^{\text{ID}_i})) \\ t_\tau &\equiv \text{if } \mathsf{accept}_\tau^{\text{ID}_1} \text{ then } \mathsf{Mac}^2_{\mathsf{k}_{\mathsf{m}}^{\text{ID}_1}}(\langle \mathsf{n}^j , \mathsf{suc}(\pi_2(t_{\mathsf{dec}}))\rangle) \\ &\quad \text{else if } \mathsf{accept}_\tau^{\text{ID}_2} \text{ then } \mathsf{Mac}^2_{\mathsf{k}_{\mathsf{m}}^{\text{ID}_2}}(\langle \mathsf{n}^j , \mathsf{suc}(\pi_2(t_{\mathsf{dec}}))\rangle) \\ &\quad \dots \\ &\quad \text{else } \mathsf{UnknownId} \end{aligned}$$

$$\sigma_\tau^{\mathsf{up}} \equiv \begin{cases} \mathsf{session}_{\text{N}}^{\text{ID}_i} \mapsto \text{if } \mathsf{inc\text{-}accept}_\tau^{\text{ID}_i} \text{ then } \mathsf{n}^j \text{ else } \sigma_\tau^{\mathsf{in}}(\mathsf{session}_{\text{N}}^{\text{ID}_i}) \\ \text{GUTI}_{\text{N}}^{\text{ID}_i} \mapsto \text{if } \mathsf{inc\text{-}accept}_\tau^{\text{ID}} \text{ then } \text{GUTI}^j \text{ else } \sigma_\tau^{\mathsf{in}}(\text{GUTI}_{\text{N}}^{\text{ID}_i}) \\ \text{SQN}_{\text{N}}^{\text{ID}_i} \mapsto \text{if } \mathsf{inc\text{-}accept}_\tau^{\text{ID}_i} \text{ then } \mathsf{suc}(\pi_2(t_{\mathsf{dec}})) \text{ else } \sigma_\tau^{\mathsf{in}}(\text{SQN}_{\text{N}}^{\text{ID}_i}) \\ \mathsf{b\text{-}auth}_{\text{N}}^{j}, \mathsf{e\text{-}auth}_{\text{N}}^{j} \mapsto \text{if } \mathsf{accept}_\tau^{\text{ID}_1} \text{ then } \text{ID}_1 \\ \qquad \text{else if } \mathsf{accept}_\tau^{\text{ID}_2} \text{ then } \text{ID}_2 \\ \qquad \dots \\ \qquad \text{else } \mathsf{UnknownId} \end{cases}$$

<u>**Case** $\mathsf{ai} = \text{PU}_{\text{ID}}(j, 2)$.</u>

$$\begin{aligned} \mathsf{accept}_\tau^{\text{ID}} &\equiv \mathsf{eq}(g(\phi_\tau^{\mathsf{in}}), \mathsf{Mac}^2_{\mathsf{k}_{\mathsf{m}}^{\text{ID}}}(\langle \sigma_\tau^{\mathsf{in}}(\mathsf{b\text{-}auth}_{\text{U}}^{\text{ID}}) , \sigma_\tau^{\mathsf{in}}(\text{SQN}_{\text{U}}^{\text{ID}})\rangle)) \\ t_\tau &\equiv \text{if } \mathsf{accept}_\tau^{\text{ID}} \text{ then ok else error} \\ \sigma_\tau^{\mathsf{up}} &\equiv \mathsf{e\text{-}auth}_{\text{U}}^{\text{ID}} \mapsto \text{if } \mathsf{accept}_\tau^{\text{ID}} \text{ then } \sigma_\tau^{\mathsf{in}}(\mathsf{b\text{-}auth}_{\text{U}}^{\text{ID}}) \text{ else fail} \end{aligned}$$

Fig. 12. The Symbolic Terms and State Updates for the SUPI Sub-Protocol.

$\underline{\textbf{Case } \mathsf{ai} = \mathrm{NS}_{\mathrm{ID}}(j).}$ $\sigma^{\mathsf{up}}_\tau \equiv \mathsf{valid\text{-}guti}^{\mathrm{ID}}_{\mathrm{U}} \mapsto \mathsf{false}$

$\underline{\textbf{Case } \mathsf{ai} = \mathrm{TU}_{\mathrm{ID}}(j,0).}$

$$
\begin{aligned}
t_\tau &\equiv \text{ if } \sigma^{\mathsf{in}}_\tau(\mathsf{valid\text{-}guti}^{\mathrm{ID}}_{\mathrm{U}}) \text{ then } \sigma^{\mathsf{in}}_\tau(\mathrm{GUTI}^{\mathrm{ID}}_{\mathrm{U}}) \text{ else } \mathsf{NoGuti} \\
\sigma^{\mathsf{up}}_\tau &\equiv \begin{cases} \mathsf{valid\text{-}guti}^{\mathrm{ID}}_{\mathrm{U}} \mapsto \mathsf{false} & \mathsf{e\text{-}auth}^{\mathrm{ID}}_{\mathrm{U}} \mapsto \mathsf{fail} \\ \mathsf{s\text{-}valid\text{-}guti}^{\mathrm{ID}}_{\mathrm{U}} \mapsto \sigma^{\mathsf{in}}_\tau(\mathsf{valid\text{-}guti}^{\mathrm{ID}}_{\mathrm{U}}) & \mathsf{b\text{-}auth}^{\mathrm{ID}}_{\mathrm{U}} \mapsto \mathsf{fail} \end{cases}
\end{aligned}
$$

$\underline{\textbf{Case } \mathsf{ai} = \mathrm{TN}(j,0).}$ Let $t^{\mathrm{ID}_i}_{\oplus} \equiv \sigma^{\mathsf{in}}_\tau(\mathrm{SQN}^{\mathrm{ID}_i}_{\mathrm{N}}) \oplus \mathsf{f}_{\mathsf{k}^{\mathrm{ID}_i}}(\mathsf{n}^j)$, then:

$$
\begin{aligned}
\mathsf{msg}^{\mathrm{ID}_i}_\tau &\equiv \langle \mathsf{n}^j, t^{\mathrm{ID}_i}_{\oplus}, \mathsf{Mac}^3_{\mathsf{k}^{\mathrm{ID}_i}_{\mathsf{m}}}(\langle \mathsf{n}^j, \sigma^{\mathsf{in}}_\tau(\mathrm{SQN}^{\mathrm{ID}_i}_{\mathrm{N}}), \sigma^{\mathsf{in}}_\tau(\mathrm{GUTI}^{\mathrm{ID}_i}_{\mathrm{N}})\rangle)\rangle \\
\mathsf{accept}^{\mathrm{ID}_i}_\tau &\equiv \mathsf{eq}(\sigma^{\mathsf{in}}_\tau(\mathrm{GUTI}^{\mathrm{ID}_i}_{\mathrm{N}}), g(\phi^{\mathsf{in}}_\tau)) \wedge \neg\mathsf{eq}(\sigma^{\mathsf{in}}_\tau(\mathrm{GUTI}^{\mathrm{ID}_i}_{\mathrm{N}}), \mathsf{UnSet}) \\
t_\tau &\equiv \text{ if } \mathsf{accept}^{\mathrm{ID}_1}_\tau \text{ then } \mathsf{msg}^{\mathrm{ID}_1}_\tau \\
&\quad\ \text{ else if } \mathsf{accept}^{\mathrm{ID}_2}_\tau \text{ then } \mathsf{msg}^{\mathrm{ID}_2}_\tau \\
&\qquad\quad \cdots \\
&\quad\ \text{ else } \mathsf{msg}^{\mathrm{ID}_{\mathsf{dum}}}_\tau \\
\sigma^{\mathsf{up}}_\tau &\equiv \begin{cases} \mathrm{GUTI}^{\mathrm{ID}_i}_{\mathrm{N}} \mapsto \text{if } \mathsf{accept}^{\mathrm{ID}_i}_\tau \text{ then } \mathsf{UnSet} \text{ else } \sigma^{\mathsf{in}}_\tau(\mathrm{GUTI}^{\mathrm{ID}_i}_{\mathrm{N}}) \\ \mathsf{session}^{\mathrm{ID}_i}_{\mathrm{N}} \mapsto \text{if } \mathsf{accept}^{\mathrm{ID}_i}_\tau \text{ then } \mathsf{n}^j \text{ else } \sigma^{\mathsf{in}}_\tau(\mathsf{session}^{\mathrm{ID}_i}_{\mathrm{N}}) \\ \mathsf{b\text{-}auth}^j_{\mathrm{N}} \mapsto \text{if } \mathsf{accept}^{\mathrm{ID}_1}_\tau \text{ then } \mathrm{ID}_1 \\ \qquad\qquad \text{else if } \mathsf{accept}^{\mathrm{ID}_2}_\tau \text{ then } \mathrm{ID}_2 \\ \qquad\qquad\quad \cdots \\ \qquad\qquad \text{else } \mathsf{UnknownId} \end{cases}
\end{aligned}
$$

$\underline{\textbf{Case } \mathsf{ai} = \mathrm{TU}_{\mathrm{ID}}(j,1).}$ Let $t_{\mathrm{SQN}} \equiv \pi_2(g(\phi^{\mathsf{in}}_\tau)) \oplus \mathsf{f}_{\mathsf{k}^{\mathrm{ID}}}(\pi_1(g(\phi^{\mathsf{in}}_\tau)))$, then:

$$
\begin{aligned}
\mathsf{accept}^{\mathrm{ID}}_\tau &\equiv \mathsf{eq}(\pi_3(g(\phi^{\mathsf{in}}_\tau)), \mathsf{Mac}^3_{\mathsf{k}^{\mathrm{ID}}_{\mathsf{m}}}(\langle \pi_1(g(\phi^{\mathsf{in}}_\tau)), t_{\mathrm{SQN}}, \sigma^{\mathsf{in}}_\tau(\mathrm{GUTI}^{\mathrm{ID}}_{\mathrm{U}})\rangle)) \\
&\qquad \wedge\ \sigma^{\mathsf{in}}_\tau(\mathsf{s\text{-}valid\text{-}guti}^{\mathrm{ID}}_{\mathrm{U}}) \ \wedge\ \mathsf{range}(\sigma^{\mathsf{in}}_\tau(\mathrm{SQN}^{\mathrm{ID}}_{\mathrm{U}}), t_{\mathrm{SQN}}) \\
t_\tau &\equiv \text{ if } \mathsf{accept}^{\mathrm{ID}}_\tau \text{ then } \mathsf{Mac}^4_{\mathsf{k}^{\mathrm{ID}}_{\mathsf{m}}}(\pi_1(g(\phi^{\mathsf{in}}_\tau))) \text{ else } \mathsf{error} \\
\sigma^{\mathsf{up}}_\tau &\equiv \begin{cases} \mathsf{b\text{-}auth}^{\mathrm{ID}}_{\mathrm{U}}, \mathsf{e\text{-}auth}^{\mathrm{ID}}_{\mathrm{U}} \mapsto \text{if } \mathsf{accept}^{\mathrm{ID}}_\tau \text{ then } \pi_1(g(\phi^{\mathsf{in}}_\tau)) \text{ else } \mathsf{fail} \\ \mathrm{SQN}^{\mathrm{ID}}_{\mathrm{U}} \mapsto \text{if } \mathsf{accept}^{\mathrm{ID}}_\tau \text{ then } \mathsf{suc}(\sigma^{\mathsf{in}}_\tau(\mathrm{SQN}^{\mathrm{ID}}_{\mathrm{U}})) \text{ else } \sigma^{\mathsf{in}}_\tau(\mathrm{SQN}^{\mathrm{ID}}_{\mathrm{U}}) \end{cases}
\end{aligned}
$$

$\underline{\textbf{Case } \mathsf{ai} = \mathrm{TN}(j,1).}$

$$
\begin{aligned}
\mathsf{accept}^{\mathrm{ID}_i}_\tau &\equiv \mathsf{eq}(g(\phi^{\mathsf{in}}_\tau), \mathsf{Mac}^4_{\mathsf{k}^{\mathrm{ID}_i}_{\mathsf{m}}}(\mathsf{n}^j)) \wedge \mathsf{eq}(\sigma^{\mathsf{in}}_\tau(\mathsf{b\text{-}auth}^j_{\mathrm{N}}), \mathrm{ID}_i) \\
\mathsf{inc\text{-}accept}^{\mathrm{ID}_i}_\tau &\equiv \mathsf{accept}^{\mathrm{ID}_i}_\tau \wedge \mathsf{eq}(\sigma^{\mathsf{in}}_\tau(\mathsf{session}^{\mathrm{ID}_i}_{\mathrm{N}}), \mathsf{n}^j) \\
t_\tau &\equiv \text{ if } \textstyle\bigvee_i \mathsf{accept}^{\mathrm{ID}_i}_\tau \text{ then } \mathsf{ok} \text{ else } \mathsf{error} \\
\sigma^{\mathsf{up}}_\tau &\equiv \begin{cases} \mathrm{SQN}^{\mathrm{ID}_i}_{\mathrm{N}} \mapsto \text{if } \mathsf{inc\text{-}accept}^{\mathrm{ID}_i}_\tau \text{ then } \mathsf{suc}(\sigma^{\mathsf{in}}_\tau(\mathrm{SQN}^{\mathrm{ID}_i}_{\mathrm{N}})) \\ \qquad\qquad\qquad\qquad\qquad \text{else } \sigma^{\mathsf{in}}_\tau(\mathrm{SQN}^{\mathrm{ID}_i}_{\mathrm{N}}) \\ \mathrm{GUTI}^{\mathrm{ID}_i}_{\mathrm{N}} \mapsto \text{if } \mathsf{inc\text{-}accept}^{\mathrm{ID}_i}_\tau \text{ then } \mathrm{GUTI}^j \text{ else } \sigma^{\mathsf{in}}_\tau(\mathrm{GUTI}^{\mathrm{ID}_i}_{\mathrm{N}}) \\ \mathsf{e\text{-}auth}^j_{\mathrm{N}} \mapsto \text{if } \mathsf{accept}^{\mathrm{ID}_1}_\tau \text{ then } \mathrm{ID}_1 \\ \qquad\qquad \text{else if } \mathsf{accept}^{\mathrm{ID}_2}_\tau \text{ then } \mathrm{ID}_2 \\ \qquad\qquad\quad \cdots \\ \qquad\qquad \text{else } \mathsf{UnknownId} \end{cases}
\end{aligned}
$$

$\underline{\text{Case } \mathsf{ai} = \mathrm{FN}(j).}$

$$
\begin{aligned}
\mathsf{msg}^{\mathrm{ID}_i}_\tau &\equiv \langle \mathrm{GUTI}^j \oplus \mathsf{f}^{\mathsf{r}}_{\mathsf{k}^{\mathrm{ID}_i}}(\mathsf{n}^j), \mathsf{Mac}^5_{\mathsf{k}^{\mathrm{ID}_i}_{\mathsf{m}}}(\langle \mathrm{GUTI}^j, \mathsf{n}^j\rangle)\rangle \\
t_\tau &\equiv \text{ if } \mathsf{eq}(\sigma^{\mathsf{in}}_\tau(\mathsf{e\text{-}auth}^j_{\mathrm{N}}), \mathrm{ID}_1) \text{ then } \mathsf{msg}^{\mathrm{ID}_1}_\tau \\
&\quad\ \text{ else if } \mathsf{eq}(\sigma^{\mathsf{in}}_\tau(\mathsf{e\text{-}auth}^j_{\mathrm{N}}), \mathrm{ID}_2) \text{ then } \mathsf{msg}^{\mathrm{ID}_2}_\tau \\
&\qquad\quad \cdots \\
&\quad\ \text{ else } \mathsf{UnknownId}
\end{aligned}
$$

$\underline{\textbf{Case } \mathsf{ai} = \mathrm{FU}_{\mathrm{ID}}(j).}$ Let $t_{\mathrm{GUTI}} \equiv \pi_1(g(\phi^{\mathsf{in}}_\tau)) \oplus \mathsf{f}^{\mathsf{r}}_{\mathsf{k}^{\mathrm{ID}}}(\sigma^{\mathsf{in}}_\tau(\mathsf{e\text{-}auth}^{\mathrm{ID}}_{\mathrm{U}}))$, then:

$$
\begin{aligned}
\mathsf{accept}^{\mathrm{ID}}_\tau &\equiv \mathsf{eq}(\pi_2(g(\phi^{\mathsf{in}}_\tau)), \mathsf{Mac}^5_{\mathsf{k}^{\mathrm{ID}}_{\mathsf{m}}}(\langle t_{\mathrm{GUTI}}, \sigma^{\mathsf{in}}_\tau(\mathsf{e\text{-}auth}^{\mathrm{ID}}_{\mathrm{U}})\rangle)) \\
&\qquad \wedge\ \neg\mathsf{eq}(\sigma^{\mathsf{in}}_\tau(\mathsf{e\text{-}auth}^{\mathrm{ID}}_{\mathrm{U}}), \mathsf{fail}) \ \wedge\ \neg\mathsf{eq}(\sigma^{\mathsf{in}}_\tau(\mathsf{e\text{-}auth}^{\mathrm{ID}}_{\mathrm{U}}), \bot) \\
t_\tau &\equiv \text{ if } \mathsf{accept}^{\mathrm{ID}}_\tau \text{ then } \mathsf{ok} \text{ else } \mathsf{error} \\
\sigma^{\mathsf{up}}_\tau &\equiv \begin{cases} \mathsf{valid\text{-}guti}^{\mathrm{ID}}_{\mathrm{U}} \mapsto \mathsf{accept}^{\mathrm{ID}}_\tau \\ \mathrm{GUTI}^{\mathrm{ID}}_{\mathrm{U}} \mapsto \text{if } \mathsf{accept}^{\mathrm{ID}}_\tau \text{ then } t_{\mathrm{GUTI}} \text{ else } \mathsf{UnSet} \end{cases}
\end{aligned}
$$

Fig. 13. The Symbolic Terms and State Updates for $\mathrm{NS}_{\mathrm{ID}}(j)$ and the GUTI and ASSIGN-GUTI Sub-Protocols.

cipher-text is equal to the message in plain-text:

$$
\frac{}{\pi_i(\langle x_1, x_2\rangle) \doteq x_i} \quad \text{for } i \in \{1,2\} \qquad \frac{}{\mathsf{dec}(\{x\}^z_{\mathsf{pk}(y)}, \mathsf{sk}(y)) \doteq x}
$$

b) Structural Axioms: Structural axioms are axioms which are valid in any computational model, e.g.:

$$
\frac{\vec{u}_1, \vec{v}_1 \sim \vec{u}_2, \vec{v}_2}{f(\vec{u}_1), \vec{v}_1 \sim f(\vec{u}_2), \vec{v}_2}\ \mathsf{FA} \qquad \frac{\vec{u}, t \sim \vec{v} \qquad s \doteq t}{\vec{u}, s \sim \vec{v}}\ R
$$

The axiom FA states that to show that two function applications are indistinguishable, it is sufficient to show that their arguments are indistinguishable. The axiom R states that if $s \doteq t$ holds then we can safely replace s by t.

c) Cryptographic Assumptions: We now explain how cryptographic assumptions are translated into axioms. We illustrate this on the unforgeability property of the functions Mac^1–Mac^5. Recall that UE_{ID} uses the same secret key $\mathsf{k}^{\mathrm{ID}}_{\mathsf{m}}$ for these five functions. Therefore, instead of the standard PRF assumption, we assume that these functions are *jointly* PRF, i.e. Mac^1–Mac^5 are *simultaneously* computationally indistinguishable from random functions.

It is well-known that if H is a PRF then H is unforgeable against an adversary with oracle access to $H(\cdot, \mathsf{k}_{\mathsf{m}})$. Similarly, we can show that if $H, H_1, \ldots, H_l$ are jointly PRF, then no adversary can forge a mac of $H(\cdot, \mathsf{k}_{\mathsf{m}})$, even if it has oracle access to $H(\cdot, \mathsf{k}_{\mathsf{m}}), H_1(\cdot, \mathsf{k}_{\mathsf{m}}), \ldots, H_l(\cdot, \mathsf{k}_{\mathsf{m}})$. We translate this property as follows: let s, m be ground terms where k_{m} appears only in subterms of the form $\mathsf{Mac}^{\_}_{\mathsf{k}_{\mathsf{m}}}(\_)$, then for every $1 \leq j \leq 5$, if S is the set of subterms of s, m of the form $\mathsf{Mac}^j_{\mathsf{k}_{\mathsf{m}}}(\_)$ then we have an instance of EUF-MAC^j:

$$
\frac{}{s = \mathsf{Mac}^j_{\mathsf{k}_{\mathsf{m}}}(m) \rightarrow \bigvee_{u \in S} s = \mathsf{Mac}^j_{\mathsf{k}_{\mathsf{m}}}(u)} \qquad (\text{EUF-MAC}^j)
$$

where $u = v$ denotes the term $\mathsf{eq}(u,v)$. Basically, if s is a valid Mac then s must have been honestly generated. Similarly, we can build a set of axioms reflecting the fact that some functions are jointly collision-resistant. Indeed, if $H, H_1, \ldots, H_l$ are jointly PRF, then no adversary can build a collision for $H(\cdot, \mathsf{k}_{\mathsf{m}})$, even if it has oracle access to $H(\cdot, \mathsf{k}_{\mathsf{m}}), H_1(\cdot, \mathsf{k}_{\mathsf{m}}), \ldots, H_l(\cdot, \mathsf{k}_{\mathsf{m}})$. This translates as follows: let m_1, m_2 be ground terms, if k_{m} appears only in subterms of the form $\mathsf{Mac}^{\_}_{\mathsf{k}_{\mathsf{m}}}(\_)$ then we have an instance of CR^j:

$$
\frac{}{\mathsf{Mac}^j_{\mathsf{k}_{\mathsf{m}}}(m_1) = \mathsf{Mac}^j_{\mathsf{k}_{\mathsf{m}}}(m_2) \rightarrow m_1 = m_2} \qquad (\text{CR}^j)
$$

These axioms are sound (the proof is given in [24]).

Proposition 2. *For every $1 \leq j \leq 5$, the* EUF-MAC^j *and* CR^j *axioms are valid in any computational model where the $(\mathsf{Mac}^i)_i$ functions are interpreted as jointly* PRF *functions.*

VII. Security Proofs

We now state the authentication and σ_{ul}-unlinkability lemmas. For space reasons, we only sketch the proofs (the full proofs are given in the technical report [24]).

A. Mutual Authentication of the AKA$^+$ Protocol

Authentication is modeled by a correspondence property [28] of the form "in any execution, if event A occurs, then event B occurred". This can be translated in the BC logic.

a) Authentication of the User by the Network: AKA^+ guarantees authentication of the user by the network if in any execution, if $HN(j)$ believes it authenticated UE_{ID}, then UE_{ID} stated earlier that it had initiated the protocol with $HN(j)$.

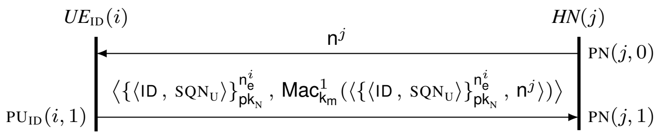
We recall that $\mathsf{e\text{-}auth}_{\mathrm{N}}^{j}$ stores the identity of the *UE* authenticated by $HN(j)$, and that UE_{ID} stores in $\mathsf{b\text{-}auth}_{\mathrm{U}}^{\text{ID}}$ the random challenge it received. Moreover, the session $HN(j)$ is uniquely identified by its random challenge n^j. Therefore, authentication of the user by the network is modeled by stating that, for any symbolic trace $\tau \in \mathsf{dom}(\mathcal{R}_{\mathsf{ul}})$, if $\sigma_\tau^{\mathsf{in}}(\mathsf{e\text{-}auth}_{\mathrm{N}}^{j}) = \text{ID}$ then there exists some prefix τ' of τ such that $\sigma_{\tau'}^{\mathsf{in}}(\mathsf{b\text{-}auth}_{\mathrm{U}}^{\text{ID}}) = \mathsf{n}^j$. Let $\preceq$ be the prefix ordering on symbolic traces, then:

Lemma 1. *For every $\tau \in \mathsf{dom}(\mathcal{R}_{ul})$, $\text{ID} \in \mathcal{S}_{id}$ and $j \in \mathbb{N}$, there is derivation using Ax of:*

$$\sigma_\tau^{in}(\mathsf{e\text{-}auth}_{\mathrm{N}}^{j}) = \text{ID} \;\rightarrow\; \bigvee\nolimits_{\tau' \preceq \tau}\; \sigma_{\tau'}^{in}(\mathsf{b\text{-}auth}_{\mathrm{U}}^{\text{ID}}) = \mathsf{n}^j$$

The key ingredients to show this lemma are *necessary conditions* for a message to be accepted by the network. Basically, a message can be accepted only if it was honestly generated by a subscriber. These necessary conditions rely on the unforgeability and collision-resistance of $(\mathsf{Mac}^j)_{1 \leq j \leq 5}$.

b) Necessary Acceptance Conditions: Using the EUF-MAC^j and CR^j axioms, we can find necessary conditions for a message to be accepted by a user. We illustrate this on the *HN*'s second message in the SUPI sub-protocol. We depict a part of the execution between session $UE_{\text{ID}}(i)$ and session $HN(j)$ below:



We then prove that if a message is accepted by $HN(j)$ as coming from UE_{ID}, then the first component of this message must have been honestly generated by a session of UE_{ID}. Moreover, we know that this session received the challenge n^j.

Lemma 2. *Let $\text{ID} \in \mathcal{S}_{id}$ and $\tau \in \mathsf{dom}(\mathcal{R}_{ul})$ be a trace ending with $\text{PN}(j,1)$. There is a derivation using Ax of:*

$$\mathsf{accept}_\tau^{\text{ID}} \rightarrow \bigvee_{\tau_1 = \_,\text{PU}_{\text{ID}}(\_,1) \preceq \tau} \left(\pi_1(g(\phi_\tau^{in})) = t_{\tau_1}^{enc} \wedge g(\phi_{\tau_1}^{in}) = \mathsf{n}^j\right)$$

Proof sktech. Let t_{dec} be the term $\mathsf{dec}(\pi_1(g(\phi_\tau^{\mathsf{in}})), \mathsf{sk_N})$. Then $HN(j)$ accepts the last message iff the following test succeeds:

$$\underline{\pi_2(g(\phi_\tau^{\mathsf{in}})) = \mathsf{Mac}^1_{\mathsf{k_m^{ID}}}(\langle \pi_1(g(\phi_\tau^{\mathsf{in}})), \mathsf{n}^j \rangle)} \wedge \pi_1(t_{\mathsf{dec}}) = \text{ID}$$

By applying EUF-MAC^1 to the underlined part above, we know that if the test holds then $\pi_2(g(\phi_\tau^{\mathsf{in}}))$ is equal to one of the honest $\mathsf{Mac}^1_{\mathsf{k_m^{ID}}}$ subterms of $\pi_2(g(\phi^{\mathsf{in}}(\tau)))$, which are the terms:

$$\left(\mathsf{Mac}^1_{\mathsf{k_m^{ID}}}(\langle t_{\tau_1}^{\mathsf{enc}}, g(\phi_{\tau_1}^{\mathsf{in}}) \rangle)\right)_{\tau_1 = \_,\text{PU}_{\text{ID}}(\_,1) \prec \tau} \tag{1}$$

$$\left(\mathsf{Mac}^1_{\mathsf{k_m^{ID}}}(\langle \pi_1(g(\phi_{\tau_1}^{\mathsf{in}})), \mathsf{n}^{j_1} \rangle)\right)_{\tau_1 = \_,\text{PN}(j_1,1) \prec \tau} \tag{2}$$

Where $\prec$ is the strict version of $\preceq$. We know that $\text{PN}(j,1)$ cannot appear twice in τ. Hence for every $\tau_1 = \_, \text{PN}(j_1,1) \prec \tau$, we know that $j_1 \neq j$. Using the fact that two distinct nonces are never equal except for a negligible number of samplings, we can derive that $\mathsf{eq}(\mathsf{n}^{j_1}, \mathsf{n}^j) = \mathsf{false}$. Using an axiom stating that the pair is injective and the CR^1 axiom, we can show that $\pi_2(g(\phi_\tau^{\mathsf{in}}))$ cannot by equal to one of the terms in (2).

Finally, for every $\tau_1 = \_, \text{PU}_{\text{ID}}(\_,1) \prec \tau$, using the CR^1 and the pair injectivity axioms we can derive that:

$$\begin{aligned}\mathsf{Mac}^1_{\mathsf{k_m^{ID}}}(\langle \pi_1(g(\phi_\tau^{\mathsf{in}})), \mathsf{n}^j \rangle) = \mathsf{Mac}^1_{\mathsf{k_m^{ID}}}(\langle t_{\tau_1}^{\mathsf{enc}}, g(\phi_{\tau_1}^{\mathsf{in}}) \rangle) \\ \rightarrow \pi_1(g(\phi_\tau^{\mathsf{in}})) = t_{\tau_1}^{\mathsf{enc}} \wedge \mathsf{n}^j = g(\phi_{\tau_1}^{\mathsf{in}})\end{aligned} \quad \blacksquare$$

We prove a similar lemma for $\text{TN}(j,1)$. The proof of Lemma 1 is straightforward using these two properties.

c) Authentication of the Network by the User: The AKA^+ protocol also provides authentication of the network by the user. That is, in any execution, if UE_{ID} believes it authenticated session $HN(j)$ then $HN(j)$ stated that it had initiated the protocol with UE_{ID}. Formally:

Lemma 3. *For every $\tau \in \mathsf{dom}(\mathcal{R}_{ul})$, $\text{ID} \in \mathcal{S}_{id}$ and $j \in \mathbb{N}$, there is derivation using Ax of:*

$$\sigma_\tau^{in}(\mathsf{e\text{-}auth}_{\mathrm{U}}^{\text{ID}}) = \mathsf{n}^j \;\rightarrow\; \bigvee\nolimits_{\tau' \preceq \tau}\; \sigma_{\tau'}^{in}(\mathsf{b\text{-}auth}_{\mathrm{N}}^{j}) = \text{ID}$$

This is shown using the same techniques than for Lemma 1.

B. σ-Unlinkability of the AKA^+ Protocol

Lemma 2 gives a necessary condition for a message to be accepted by $\text{PN}(j,1)$ as coming from ID. We can actually go further, and show that a message is accepted by $\text{PN}(j,1)$ as coming from ID *if and only if* it was honestly generated by a session of UE_{ID} which received the challenge n^j.

Lemma 4. *Let $\text{ID} \in \mathcal{S}_{id}$ and $\tau \in \mathsf{dom}(\mathcal{R}_{ul})$ be a trace ending with $\text{PN}(j,1)$. There is a derivation using Ax of:*

$$\mathsf{accept}_\tau^{\text{ID}} \leftrightarrow \bigvee_{\tau_1 = \_,\text{PU}_{\text{ID}}(\_,1) \preceq \tau} \left(g(\phi_\tau^{in}) = t_{\tau_1} \wedge g(\phi_{\tau_1}^{in}) = \mathsf{n}^j\right)$$

We prove similar lemmas for most actions of the AKA^+ protocol. Basically, these lemmas state that a message is accepted if and only if it is part of an honest execution of the protocol between UE_{ID} and *HN*. This allow us to replace each acceptance conditional $\mathsf{accept}_\tau^{\text{ID}}$ by a disjunction over all possible honest partial transcripts of the protocol.

We now state the σ_{ul}-unlinkability lemma:

Lemma 5. *For every $(\tau_l, \tau_r) \in \mathcal{R}_{ul}$, there is a derivation using Ax of the formula $\phi_{\tau_l} \sim \phi_{\tau_r}$.*

The full proof is long and technical. It is shown by induction over τ. Let $(\tau_l, \tau_r) \in \mathcal{R}_{\mathsf{ul}}$, we assume by induction that there is a derivation of $\phi_{\tau_l}^{\mathsf{in}} \sim \phi_{\tau_r}^{\mathsf{in}}$. We want to build a derivation of $\phi_{\tau_l}^{\mathsf{in}}, t_{\tau_l} \sim \phi_{\tau_r}^{\mathsf{in}}, t_{\tau_r}$ using the inference rules in Ax.

First, we rewrite t_{τ_l} using the acceptance characterization lemmas such as Lemma 4. This replaces each $\mathsf{accept}_{\tau_l}^{\text{ID}}$ by a case disjunction over all honest executions *on the left side*. Similarly, we rewrite t_{τ_r} as a case disjunction over honest executions *on the right side*. Our goal is then to find a matching between left and right transcripts such that matched transcripts are indistinguishable. If a left and right transcript correspond to the same trace of oracle calls, this is easy. But since the left and right traces of oracle calls may differ, this is not always possible. E.g., some left transcript may not have a corresponding right transcript. When this happens, we have two possibilities: instead of a one-to-one match we build a many-to-one match, e.g. matching a left transcript to several right transcripts; or we show that some transcripts always result in a failure of the protocol. Showing the latter is complicated, as it requires to precisely track the possible values of $\mathrm{SQN}_{\mathrm{U}}^{\mathrm{ID}}$ and $\mathrm{SQN}_{\mathrm{N}}^{\mathrm{ID}}$ across multiple sessions of the protocol to prove that some transcripts always yield a desynchronization between UE_{ID} and HN.

VIII. Conclusion

We studied the privacy provided by the 5G-AKA authentication protocol. While this protocol is not vulnerable to IMSI catchers, we showed that several privacy attacks from the literature apply to it. We also discovered a novel desynchronization attack against PRIV-AKA, a modified version of AKA, even though it had been claimed secure.

We then proposed the AKA$^+$ protocol. This is a fixed version of 5G-AKA, which is both efficient and has improved privacy guarantees. To study AKA$^+$'s privacy, we defined the σ-unlinkability property. This is a new parametric privacy property, which requires the prover to establish privacy only for a subset of the standard unlinkability game scenarios. Finally, we formally proved that AKA$^+$ provides mutual authentication and σ_{ul}-unlinkability for any number of agents and sessions. Our proof is carried out in the Bana-Comon model, which is well-suited to the formal analysis of stateful protocols.

Acknowledgment

This research has been partially funded by the French National Research Agency (ANR) under the project TECAP (ANR-17-CE39-0004-01).

References

[1] *TS 33.501: Security architecture and procedures for 5G system*, 3GPP Technical Specification, Rev. 15.2.0, September 2018.

[2] D. Strobel, "IMSI catcher," *Ruhr-Universität Bochum, Seminar Work*, 2007.

[3] R. Borgaonkar, L. Hirshi, S. Park, A. Shaik, A. Martin, and J.-P. Seifert, "New adventures in spying 3G & 4G users: Locate, track, monitor," 2017, briefing at BlackHat USA 2017. [Online]. Available: https://www.blackhat.com/us-17/briefings.html#new-adventures-in-spying-3g-and-4g-users-locate-track-and-monitor

[4] M. Arapinis, L. I. Mancini, E. Ritter, M. Ryan, N. Golde, K. Redon, and R. Borgaonkar, "New privacy issues in mobile telephony: fix and verification," in *the ACM Conference on Computer and Communications Security, CCS'12*. ACM, 2012, pp. 205–216.

[5] M. Arapinis, T. Chothia, E. Ritter, and M. Ryan, "Analysing unlinkability and anonymity using the applied pi calculus," in *Proceedings of the 23rd IEEE Computer Security Foundations Symposium, CSF 2010*. IEEE Computer Society, 2010, pp. 107–121.

[6] P. Fouque, C. Onete, and B. Richard, "Achieving better privacy for the 3gpp AKA protocol," *PoPETs*, vol. 2016, no. 4, pp. 255–275, 2016.

[7] N. Kobeissi, K. Bhargavan, and B. Blanchet, "Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach," in *2017 IEEE European Symposium on Security and Privacy, EuroS&P*. IEEE, 2017, pp. 435–450.

[8] K. Bhargavan, C. Fournet, and M. Kohlweiss, "mitls: Verifying protocol implementations against real-world attacks," *IEEE Security & Privacy*, vol. 14, no. 6, pp. 18–25, 2016.

[9] C. Cremers, M. Horvat, J. Hoyland, S. Scott, and T. van der Merwe, "A comprehensive symbolic analysis of TLS 1.3," in *ACM Conference on Computer and Communications Security, CCS'17*. ACM, 2017, pp. 1773–1788.

[10] M. Abadi, B. Blanchet, and C. Fournet, "The applied pi calculus: Mobile values, new names, and secure communication," *J. ACM*, vol. 65, no. 1, pp. 1:1–1:41, 2018.

[11] B. Blanchet, ProVerif*: Cryptographic protocols verifier in the formal model*, available at http://proseccco.gforge..inria.fr/personal/bblanchet/proverif/.

[12] V. Cheval, S. Kremer, and I. Rakotonirina, "DEEPSEC: deciding equivalence properties in security protocols theory and practice," in *2018 IEEE Symposium on Security and Privacy, SP 2018*. IEEE, 2018, pp. 529–546.

[13] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The tamarin prover for the symbolic analysis of security protocols," in *25th International Conference on Computer Aided Verification, CAV'13*. Springer-Verlag, 2013, pp. 696–701.

[14] V. Cortier, N. Grimm, J. Lallemand, and M. Maffei, "A type system for privacy properties," in *ACM Conference on Computer and Communications Security, CCS'17*. ACM, 2017, pp. 409–423.

[15] V. Shoup, "Sequences of games: a tool for taming complexity in security proofs," *IACR Cryptology ePrint Archive*, vol. 2004, p. 332, 2004.

[16] B. Blanchet, "A computationally sound mechanized prover for security protocols," *IEEE Trans. Dependable Sec. Comput.*, vol. 5, no. 4, pp. 193–207, 2008.

[17] G. Bana and H. Comon-Lundh, "Towards unconditional soundness: Computationally complete symbolic attacker," in *Principles of Security and Trust, 2012*, ser. LNCS, vol. 7215. Springer, 2012, pp. 189–208.

[18] G. Bana and H. Comon-Lundh, "A computationally complete symbolic attacker for equivalence properties," in *2014 ACM Conference on Computer and Communications Security, CCS '14*. ACM, 2014, pp. 609–620.

[19] F. van den Broek, R. Verdult, and J. de Ruiter, "Defeating IMSI catchers," in *ACM Conference on Computer and Communications Security, CCS'15*. ACM, 2015, pp. 340–351.

[20] D. A. Basin, J. Dreier, L. Hirschi, S. Radomirovi'c, R. Sasse, and V. Stettler, "A formal analysis of 5G authentication," in *the ACM Conference on Computer and Communications Security, CCS'18*. ACM, 2018.

[21] M. Lee, N. P. Smart, B. Warinschi, and G. J. Watson, "Anonymity guarantees of the UMTS/LTE authentication and connection protocol," *Int. J. Inf. Sec.*, vol. 13, no. 6, pp. 513–527, 2014.

[22] J. Hermans, A. Pashalidis, F. Vercauteren, and B. Preneel, "A new RFID privacy model," in *ESORICS*, ser. Lecture Notes in Computer Science, vol. 6879. Springer, 2011, pp. 568–587.

[23] S. Vaudenay, "On privacy models for RFID," in *ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security*, ser. LNCS. Springer, 2007, pp. 68–87.

[24] A. Koutsos, "The 5G-AKA authentication protocol privacy," *CoRR*, vol. abs/1811.06922, 2018.

[25] A. Shaik, J. Seifert, R. Borgaonkar, N. Asokan, and V. Niemi, "Practical attacks against privacy and availability in 4g/lte mobile communication systems," in *23rd Annual Network and Distributed System Security Symposium, NDSS*. The Internet Society, 2016.

[26] L. Hirschi, D. Baelde, and S. Delaune, "A method for verifying privacy-type properties: The unbounded case," in *IEEE Symposium on Security and Privacy, SP 2016*. IEEE Computer Society, 2016, pp. 564–581.

[27] H. Comon and A. Koutsos, "Formal computational unlinkability proofs of RFID protocols," in *30th Computer Security Foundations Symposium, 2017*. IEEE Computer Society, 2017, pp. 100–114.

[28] T. Y. C. Woo and S. S. Lam, "A semantic model for authentication protocols," in *Proceedings 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, May 1993, pp. 178–194.