Formal Analysis of 5G AKMA

Tengshun Yang[a,b], Shuling Wang[a,b], Bohua Zhan[a,b], Naijun Zhan[a,b], Jinghui Li[c], Shuangqing Xiang[c], Zhan Xiang[c], Bifei Mao[c]

[a] *SKLCS, Institute of Software, CAS, Beijing, China*
[b] *University of Chinese Academy of Sciences, Beijing, China*
[c] *Trustworthiness Theory Research Center, Huawei Technologies Co., Ltd., Shenzhen, China*

Abstract

Security and privacy of users' information in mobile communication networks have drawn increasing attention. The development of 5G system has demanded new protocols to realize authentication and key management service. AKMA (Authentication and Key Management for Application) service aims at establishing authenticated communication between users and application functions. For this purpose, the 3GPP group has standardized 5G AKMA service in Technical Specifications defining the 5G AKMA security architecture and procedures. To ensure security of communication between users and applications, AKMA service should meet strong security properties. In this paper, we apply formal methods to model and analyze the AKMA service. We construct a formal model of AKMA in the Tamarin verification tool, and specify the authentication, secrecy, and privacy properties extracted from informal descriptions given in the Technical Specifications. We identify assumptions for each security property during the modeling process. We prove that some properties are not satisfied, and by analyzing the counterexamples constructed by Tamarin, put forward some potential attacks. Moreover, we propose some suggestions for the 5G AKMA service.

Keywords: 5G AKMA, security, formal modeling, verification

1. Introduction

With mobile communication networks widely used across the world, more and more people subscribe to home networks and communicate with each other or use online services, such as phone calls, emails, and entertainment

applications. Much of these communications occur through public channels, which can be intercepted or suffer from other kinds of attacks. In order to ensure security and privacy of subscribers and application providers communicating along insecure channels, 3GPP (3rd Generation Partnership Project) has been specifying the security architecture, i.e. security features and mechanisms, for the 5G System and the 5G Core, and the security procedures performed within the 5G System including 5G Core and 5G New Radio in the Technical Specification (TS)[1]. One of the main mechanisms is to support authentication and key management aspects for applications, that is mutual authentication between users and application providers. Specifically, a major aim of this service is to allow application providers to authenticate users without knowing the users' identifier, with the home network of the user as an intermediary.

5G AKMA (Authentication and Key Management for Application) is a novel cellular-network-based delegated authentication service. This service, specified in 3GPP TS 33.535 [2], aims to provide a protocol to support authentication and key management aspects for applications based on subscription credentials. In AKMA, application provider, denoted by AKMA Application Function (AF), delegates the authentication of application user (UE) to the corresponding home network (HN) where the user subscribes. In this way, application provider could verify the identity of the user through home network without having chance to acquire knowledge and information of the user, especially, the real identifier of the user. The standardization of 5G AKMA service started with Release 16 in 2019 and the latest version was specified in Release 17. In this paper, according to the version 17.4.0 of Release 17 of the Technical Specification (TS) [2], we will provide the first formal model of 5G AKMA and also verify formally the security requirements using Tamarin.

In this paper, we apply formal methods to analyze the AKMA service, using the Tamarin verification tool [3]. Tamarin specifies protocols as a set of rewrite rules acting on a multiset of facts, and properties as two-sorted first-order logic assertions. By writing appropriate actions in the rules and in the trace, it is possible to formulate various threat models, such as Dolev-Yao [4] and eCK [5], as well as various authentication specifications [6]. Using a backward-search style algorithm [7], Tamarin attempts to prove the properties or find a counterexample. The counterexamples help users find potential attacks of protocols.

Contribution. In this work, we formally specify the standard's security assumptions and requirements of 5G AKMA, and build the first formal model of 5G AKMA for a precise security analysis. First, we construct a formal model of 5G AKMA, as specified in TS 33.535 [2], as a set of rewrite rules in Tamarin. As we describe in Section 4, the model contains main features and functions in the protocol. During the modeling process, we identity the security assumptions about the protocol for guaranteeing the security properties, which are implicitly stated in the standard documents. Next, we model the classical properties (e.g. secrecy, authentication and privacy) and check them in Tamarin. During the verification, for some of these security properties, Tamarin returns a counterexample showing that the model does not satisfy the given property. We then analyze the attacks according to the counterexamples and put forward the potential security and privacy problems about AKMA protocol. Also, we give suggestions to fix these problems.

This paper is an extension of our work in SETTA 2021 [8], which contains the following updates and extensions:

1. We modify and update our model according to version 17.4.0 of the Technical Specification [2] to make our model closer to reality and better describe the protocol, while the model given in the conference paper was based on version 17.1.0 of the Technical Specification. For example, the version 17.4.0 adds the identity of the user in the message that the home network sends to the application, and we add a corresponding identifier to our model.
2. We use the observational equivalence mode of Tamarin to specify and verify some common privacy properties of the AKMA service. Hence, we classify the properties we consider into authentication, secrecy and privacy properties. Moreover, we give a more detailed introduction to the theory underlying the Tamarin Prover, including its observational equivalence mode.
3. Due to the improvements to our model, in particular adding identifiers to relevant messages, we excluded from the model one counterexample that is considered unnatural in the conference paper.
4. Besides, we analyze the implicit authentication phase, i.e., key confirmation round trip, between the user and the application added at the end of the protocol, and analyze the verification results of the security properties with/without this phase. The verification results show that the addition of the key confirmation will promote the authentication

level of the protocol. We also present and explain more critical rules that construct the model.

1.1. Related Work

In the earlier generations of mobile network, the corresponding services were also specified by 3GPP. GBA (Generic Bootstrapping Architecture) [9] and BEST (Battery Efficient Security for very low Throughput Machine Type Communication (MTC) devices) [10], served use cases similar to that of AKMA in the 3rd and 4th generation respectively. 5G AKMA inherits and evolves features of GBA and BEST, performs better in all kinds of requirements (refering to 3GPP TR33.835 [11]). In [12], Khan *et al* analyzed potential AKMA requirements and compared AKMA with GBA and BEST. Beyond that, they put forward two new privacy requirements arose from AKMA applications, developed a privacy mode for fulfilling them and analyzed the security and privacy of their solution informally. In another work [13], they introduced delegated authentication system and summarized recent work about AKMA.

There are lots of work on formal modeling and verification of security systems. For adversaries, the most important models are Dolev-Yao model [4], eCK model [5], and its extension SeCK model [14]. The adversaries are given different powers for each of them. Especially, the eCK model inherits the spirit of Bellare and Rogaway [15] and Canetti and Krawczyk [16, 17] by an experiment in which the adversary is given many corruption powers for various key exchange sessions and must solve a challenge on a test session. Formal modeling languages and logics are used for modeling security protocols, and for capturing security properties, facilitating verification and debugging. These work include the process algebra CSP [18, 19, 6, 20], BAN logic [21], applied π-calculus [22], Horn clauses [23], TLA [24, 25], rewriting system [3] and so on.

Some security protocol verification tools are developed based on these theories, such as Tamarin [3], Maude-NPA [26], ProVerif [23], and so on. Tamarin will be introduced in Section 3. The Maude-NPA tool [26] supports protocols specified as linear role-scripts and properties specified as symbolic states [27]. ProVerif [23] models a protocol as a set of Horn clauses, analyzes them using a two-phase resolution algorithm, and uses abstractions to obtain an efficient analysis method.

There are lots of work on verification of security protocols. Protocols with loops and non-monotonic mutable global states such as TESLA proto-

cols, YubiKey and YubiHSM protocols were considered in [28, 29]. In [30], ARPKI protocol with many messages and multiple parties was modeled and analyzed. The group protocols STR and GDH based on Diffie-Hellman were verified on security and privacy. TLS 1.3 and 5G AKA protocol were analyzed in [31, 32, 33], which are important for Internet security and also widely used to establish secure channels in a variety of contexts. Significantly, 3GPP [34] formally analyzes the 3G AKA protocol using TLA [25] on the absence of failure scenarios and uses BAN logic [21] on proving security goals respectively.

There are also a large number of studies to conceptualize privacy, both informally and formally, in the literature. The definition of privacy was discussed and distinguished as hard privacy and soft privacy in [35, 36]. The work [37] presented a hierarchy of privacy notions that covers multiple anonymity and unlinkability variants. In [38], they proposed the notion of enforced privacy, meaning that a user's privacy is preserved even if the user collaborates with the adversary, and formalized the notions using applied π-calculus.

2. AKMA in 5G system

In this section, we give an informal introduction to the 5G AKMA service. We first describe the main entities of the service, and then present the steps of the protocol in detail. Our models are based on version 17.4.0 of the Technical Specification. See the specification document [2] for further information.

2.1. General Architecture

There are three main entities (roles) in the 5G AKMA service, as shown in Figure 1. We explain them below.

1. User Equipment (UE): represents user of the service, consisting of two parts: Mobile Equipment (ME) and Universal Integrated Circuit Card (UICC).
2. Home Network (HN): represents the mobile network provider. HN has all of the information about its subscribers, and is always considered to be credible. Home network plays the role of authenticating users and helps application providers to reach an agreement with the user on session keys in the AKMA service. There are several functions located within the HN, as follows:

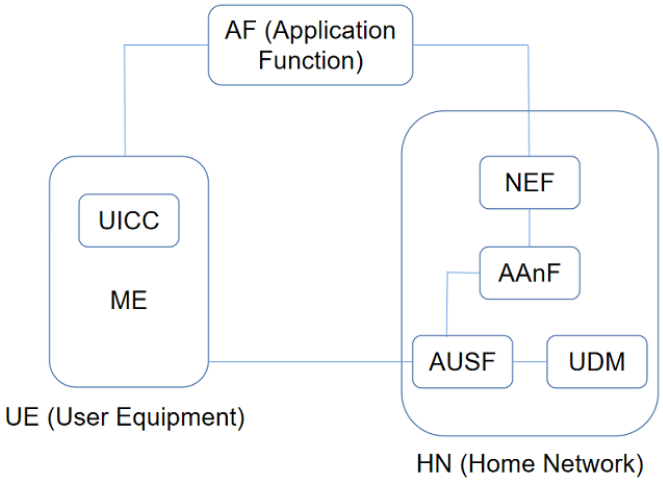

Figure 1: AKMA Architecture

- UDM (Unified Data Management): stores information about all subscribers of the home network.
- AAnF (AKMA Anchor Function): manages temporary information about subscribers, and generates temporary session keys K_{AF} for the application functions.
- AUSF (Authentication Server Function): connection between UDM and AAnF, obtains the 5G authentication vector from UDM and generates relative AKMA materials.
- NEF (Network Exposure Function): when the target AF is located outside the HN, establishes connection between AAnF and AF.

In general, there is also a Serving Network (SN) which the users connects directly to when roaming. Well, the roaming aspect is not considered in the technical specification of 5G AKMA service[2]. Therefore, in this paper, we consider only the case when the user is not roaming, that is, SN is part of the HN. We do not consider SN separately.

3. Application Function (AF or AApF): also called application provider or service provider, represents the online services that the user may wish to use. The goal of AKMA is to help to establish a secure channel (exchange a secret key) between AF and UE, with authentication of UE delegated to its corresponding HN.

Every user in the cellular network subscribes to a home network and has a unique long-term identifier SUPI (Subscription Permanent Identifier) and a long-term key K. These are stored at both UE and HN.

It is worth noting that the mutual authentication between HN and AF is not part of the AKMA service. That is, it should be prepared before the execution of the protocol. According to TS 33.501 [1], mutual authentication based on client and server certificates shall be performed between the HN and AF using TLS protocol. In our modeling of the protocol in Section 4, we will model their communication in a private channel.

2.2. 5G AKMA Protocol

5G AKMA protocol specifies the functions and behaviors of the AKMA service. We will begin by introducing the primary authentication step, which is a prerequisite but not a key part of the protocol. Next, we will present the interactions between UE, HN and AF step by step.

2.2.1. Primary Authentication

Before AKMA service can start, UE and HN must execute mutual authentication. This primary authentication step is known as 5G Authentication and Key Agreement (5G AKA [1]). Prior generations of cellular networks have different AKA protocols: 3G has UMTS AKA protocol [39]; 4G has LTE AKA protocol [40]; in 5G, besides AKA protocol, there exists EAP-AKA′ [1]. Whether to use 5G AKA or EAP-AKA′ is decided by HN.

As mentioned above, UE has its unique and permanent identifier SUPI and secret key K, which are also stored in HN. Roughly speaking, UE starts the 5G AKA protocol by encrypting its SUPI and sends to the corresponding HN. With the random number that HN sends to the UE, some authentication information and the sequence number, HN and UE agree on the authentication and freshness. When the primary authentication is finished, both UE and AUSF in HN side would generate K_{AUSF}, which will be used for generating subsequent keys during AKMA service.

2.2.2. Deriving AKMA Materials

The steps for deriving AKMA materials are shown in Figure 2. After UE finishes primary authentication with HN, and before it initiates communication with an AKMA Application Function (AF), it generates the AKMA Anchor Key K_{AKMA} and A-KID from K_{AUSF} (Steps 3, 4). The A-KID (AKMA

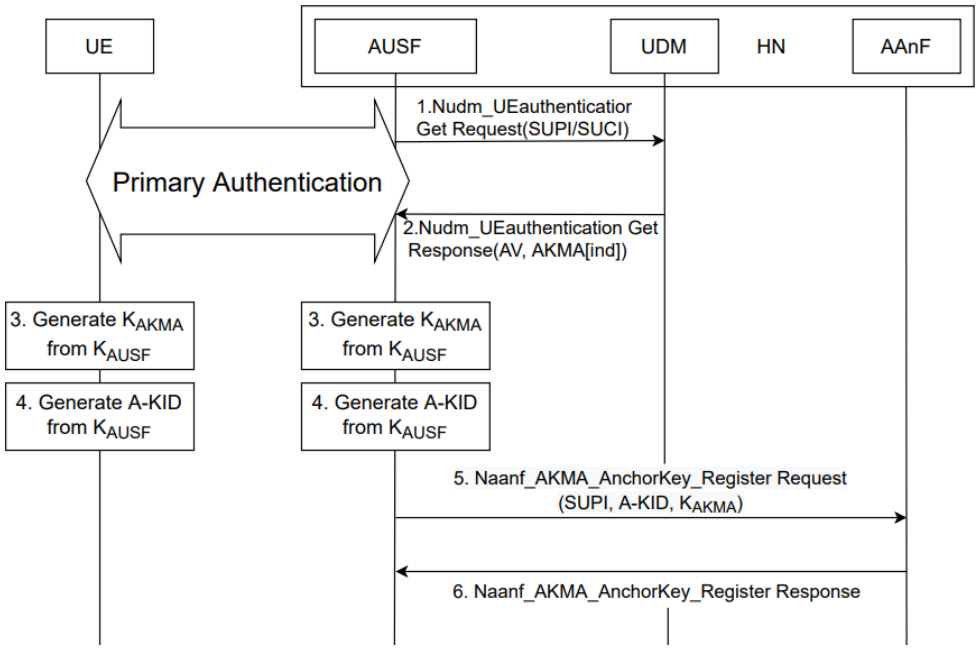

Figure 2: Deriving AKMA Materials

Key Identifier) consists of A-TID (AKMA Temporary UE Identifier) and HN-ID (identity of home network).

After receiving K_{AUSF} from UDM, AUSF stores this key and generates the AKMA Anchor Key K_{AKMA} and A-KID from K_{AUSF} (Steps 3, 4). Then AUSF sends the AKMA key materials (K_{AKMA}, A-KID) together with the SUPI of UE to AAnF (Step 5). AUSF does not need to store any AKMA key materials after sending them to AAnF.

When AAnF receives the AKMA key materials from AUSF, it first deletes the old materials with the same SUPI (if there exists any). This means, if re-authentication runs, AAnF only stores the latest materials from AUSF, and each UE only has one AKMA key material at any time in AAnF. Then AAnF would give a response back to AUSF (Step 6).

2.2.3. Deriving AKMA application key for a specific AF

The steps for deriving AKMA application key are shown in Figure 3 and 4. If UE attempts to connect to AF without initiating AKMA protocol, AF would reject the request with an AKMA initiation message. Then UE would re-send the request in accordance to AKMA.

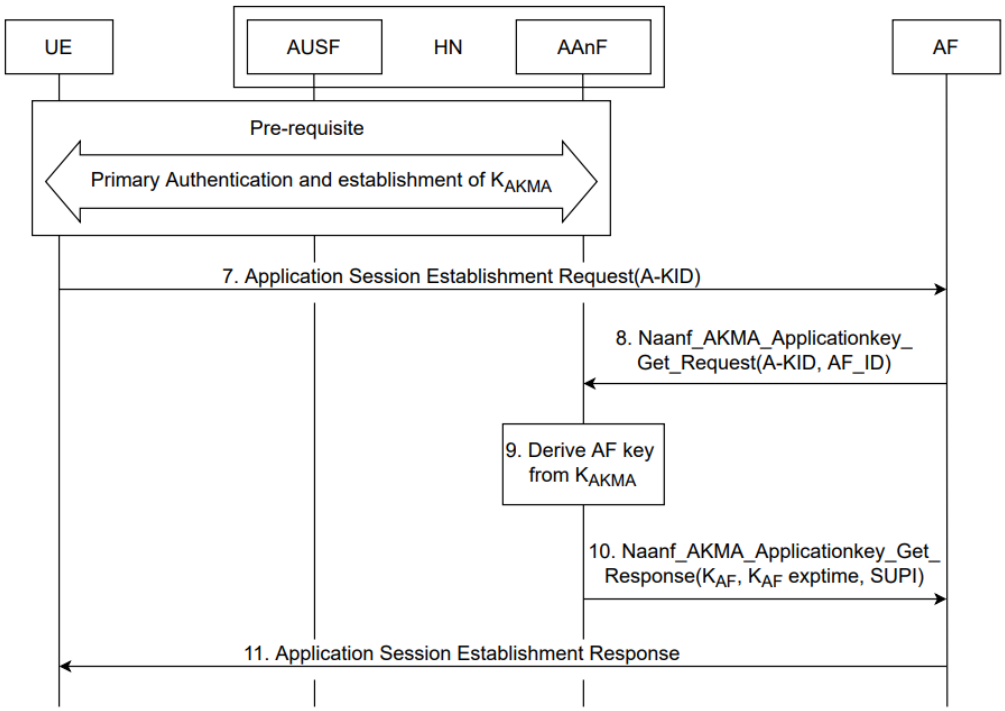

Figure 3: Deriving AKMA application key for a specific AF located within HN

UE initiates the AKMA protocol by sending the A-KID to AF (Step 7). Since the A-KID contains identity of HN, AF would attempt to establish connection with the HN. The following steps are divided into two cases, depending on whether AF is located within HN or not.

If AF is located within HN(see figure 3), it connects with AAnF directly. AF forwards the A-KID together with its own identity (AF-ID) to the AAnF in the HN (Step 8). Then AAnF checks the presence of the UE specific K_{AKMA} key corresponding to the received A-KID. If the material does not exist, AAnF returns an error message. Otherwise, according to the AF-ID received and the AKMA key material, AAnF generates K_{AF} (Step 9). Moreover, AAnF decides an expiration time for the key. It then sends the corresponding SUPI, the session key K_{AF} with its expiration time as a response back to AF (Step 10). The sending of identity SUPI is added in version 17.4.0 of the specification. If any step in the procedure fails, UE would receive a reject response and need to re-request with the latest A-KID.

If AF is located outside HN (see Figure 4), it connects to NEF rather than

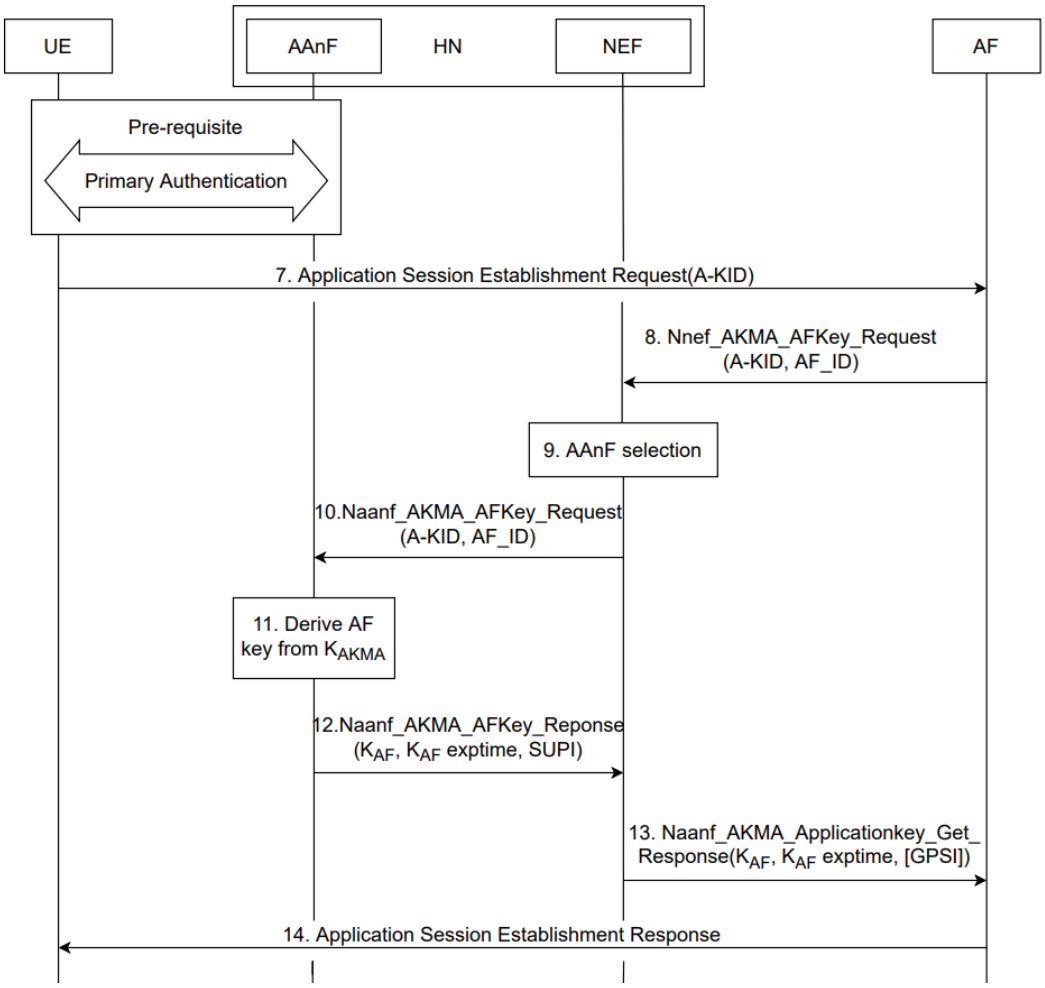

Figure 4: Deriving AKMA application key for a specific AF located outside HN

AAnF, which enables and authorizes external AF accessing AKMA service and forwards the request to AAnF. NEF plays a role of intermediary between AF and AAnF and performs AAnF selection(Step 9). Most of the procedure is the same as above. The major difference is that HN will translate SUPI to GPSI (external ID) and optionally include GPSI (external ID) in the response (Steps 12 and 13). Similar to the case when AF is located within HN, the addition of SUPI, the translation of SUPI to GPSI and the optional inclusion of GPSI in the response are added in version 17.4.0 of the specification. The NEF shall not send the SUPI to the AF directly otherwise it will result in

leakage of the most important private information.

When AF receives the session key K_{AF} and K_{AF} expiration time, it responds to UE. Since UE has all AKMA key materials, i.e. the latest K_{AKMA}, it can also generate K_{AF} by itself. Significantly, when the session key expires, AF ends the session with UE, but UE has a chance to refresh K_{AF}, depending on the protocol at the interface between AF and UE, i.e. the Ua* protocol. If this protocol supports refresh of K_{AF}, AF may refresh K_{AF} at any time using the Ua* protocol.

The case when AF is located within HN is secure, as no external communication occurs. So we will model the case that AF is located outside HN, to be presented in Section 4.2.

2.3. KDFs in the protocol

There are several Key Derivation Functions (KDFs) involved in the AKMA protocol. Each KDF accepts a number of input arguments. For generating each kind of key, some of the arguments are constant, while others depend on identifiers and existing keys. The key K_{AKMA} is derived from SUPI and K_{AUSF}. The temporary identifier A-TID is also derived from SUPI and K_{AUSF}, but with different settings of constants. The key K_{AF} is derived from identifiers for AF and K_{AKMA}. See [2] for more details.

3. Tamarin Prover

In this section, we give a brief introduction to the Tamarin verification tool [41]. Tamarin is a powerful tool for symbolic modeling and analysis of security protocols. It takes as input a security protocol model, specifying the actions taken by agents running the protocol in different roles (e.g., the protocol initiator, the responder, and the trusted server), a specification of the adversary, and a specification of the protocol's desired properties [41]. With the above inputs, Tamarin verifies whether the protocol satisfies the properties. Tamarin supports verification when there are an arbitrary number of sessions. This is reflected in modeling the state as a multiset of facts, where each new session is modeled by applying the corresponding initialization rule and adding new (linear) facts to the state. Hence, the state space is potentially infinite. Tamarin deals with the infinite state space using a backward-search style algorithm, starting from the violation of the property to be verified, and checking how the violation can result from applying the rules. The search does not always terminate as the verification problem can

be shown to be undecidable. If the search terminates, Tamarin either proves that the property is satisfied, or finds a trace as counterexample against the property. The user interface shows the trace as a visual chart, which can be examined, to analyze for possible mistakes in the constructed model, the statement of properties, or the protocol itself. Since the verification problem is undecidable, to partially remedy the situation that does not terminate, Tamarin also provides an interactive mode where the user can guide the tool through the verification. We now introduce the usage of Tamarin from two aspects: modeling and property specification.

3.1. Modeling

In Tamarin, messages are described using *terms*, which are formed from variables, constants, and functions. For example, the theory of symmetric encryption is given by two functions *dec* and *enc*. The term $enc(m, k)$ denotes encryption of message m with key k, and the term $dec(m, k)$ denotes decryption. Moreover, a set of identities specify the equational theory. For example, symmetric encryption has the equation $dec(enc(m, k), k) = m$.

The protocol is specified using an expressive language based on multiset rewriting rules. These rules construct a labeled transition system whose states are multisets of facts, which give a symbolic representation of the current state of the protocol, messages on the network, and adversary knowledge. In Tamarin, the sort of a variable is expressed using the following prefixes: `~` for fresh variables, `$` for public variables, `#` for temporal variables, indicating the order of actions. There are three types of builtin fact symbols: `Fr` for generating a fresh value, `In` for receiving a message from the untrusted network, `Out` for sending a message to the untrusted network. As Tamarin assumes Dolev-Yao style attackers [4], the adversary can intercept any message that is output through `Out`, and insert any message as `In`. The adversary can construct new terms from existing knowledge (modulo rewriting rules), but cannot break the cryptography. For example, in the symmetric encryption theory above, the adversary cannot derive m if he knows only $enc(m, k)$, but will be able to do so if he additionally knows k, by constructing $dec(enc(m, k), k)$ and rewriting to m. In addition to the three builtin fact symbols, Tamarin allows defining any number of custom fact symbols. By default, a fact symbol is *linear*, meaning each fact with that symbol can be used only once. A fact symbol can be declared as *permanent* by prepending an exclamation sign (`!`).

Each rule consists of a list of premises, a list of conclusions, and a list of actions. A rule can be executed if each premise in the rule is present in the current multiset. The transition corresponding to executing this rule removes all premises from the multiset (except the permanent facts), and inserts conclusions into the multiset. The actions of the rule are appended together to form the trace of execution.

We illustrate these concepts with an example, in which agents A and B share a long-term key k, and A uses this key to send an encrypted message to B.

Example 3.1 *In the protocol, A encrypts m with k and sends it to B.*

```
rule Initial: [Fr(k)] --> [!Ltk($A, k), !Ltk($B, k)]
rule Send_A: [!Ltk($A, k), Fr(m)] --[Send(A, m)]-> [Out(enc(m, k))]
rule Recv_B: [!Ltk($B, k), In(enc(m, k))] --[Recv(B, m)]-> []
```

In the above code, each line specifies a rule of the protocol. If there are no actions in the rule, the premises and conclusions are joined by `-->`. Otherwise, the list of actions is written in the middle of the arrow. Terms preceded by the symbol $ are public terms (known to everyone including the adversary).

3.2. Property Specification

Tamarin Prover provides two kinds of property specifications: **Trace** and **Observational Equivalence** properties. Trace properties are given as guarded first-order logic formulas and observational equivalence properties are specified using the diff operator, which we will introduce below.

3.2.1. Trace Properties

Security properties are defined over traces, formulated in terms of many-sorted first-order logic formulas over messages and timepoints, and checked against traces of the transition system. Using this logic, we can specify various secrecy and authentication properties.

Continuing Example 3.1, we show how to describe various levels of authentication specifications according to [6]. The following lemma specifies non-injective agreement between two agents A and B, meaning whenever B completes a run of the protocol, apparently with A, then A has been previously running the protocol, apparently with B, and they agree on the message m:

```
lemma Non_injective_agreement:
  "All m #i. Recv(B, m) @ i ==> (EX #j. Send(A, m) @ j & j < i)"
```

This property holds for the above example. The only way `Recv(B,m)` can appear in the trace is for rule `Recv_B` to be executed. This can occur only if a term `enc(m,k)` is input. Since the adversary does not know `k`, there is no way for him to construct the message `enc(m,k)`. So the input can only come from rule `Send_A`, which creates the action `Send(A,m)` at an earlier timepoint.

However, the following stronger property, injective agreement, does not hold:

```
lemma Injective_agreement:
    "All m #i. Recv(B, m) @ i
    ==> (Ex #j. Send(A, m) @ j & j < i
        & not (Ex #i2. Recv(B, m) @ i2 & not (#i2 = #i)))
```

This is because the adversary can intercept the message `enc(m,k)` and resend it, resulting in another execution of the rule `Recv_B`. Clearly this protocol is too weak to guard against replay attacks.

3.2.2. Observational Equivalence

In contrast to trace properties, i.e., properties that are defined on each trace independently, observational equivalence properties reason about two similar systems (for example two instances of a protocol), by showing that a specific adversary can not distinguish between these two systems. This kind of property is used to specify privacy properties, indistinguishability property, etc.

Tamarin can prove properties for two systems that only differ in those terms specified using the operator `diff( , )`. Any model containing the `diff` operator specifies two models: one model where each `diff(x,y)` is replaced with `x`, and the other where each `diff(x,y)` is replaced with `y`. We call these two systems LHS (left hand side) and RHS (right hand side), respectively. In the observational equivalence mode, there is a new builtin Lemma `Observational_Equivalence`, which is used to represent the observational equivalence property between LHS and RHS models. To prove this lemma, Tamarin computes for each rule all possible executions on both sides, and verifies whether an "equivalent" execution exists on the other side. Here we show the toy example in the Tamarin manual [41]:

Example 3.2 *The entity generates a public key and outputs it and chooses two fresh values a and b and reveals one of it. Then it encrypts either a or b and outputs the ciphertext. The observational equivalence property is whether adversaries can know which value is encrypted in the last message.*

```
rule example:
  [ Fr(~a),
    Fr(~b),
    Fr(~ltk) ]
  --[ Secret(b) ]->
  [ Out(pk(~ltk),
    Out(~a),
    Out(aenc(diff(~a,~b), pk(~ltk))) ]
```

Actually in this example, the observational equivalence is not satisfied since the adversary can take the output value a, encrypt it with the public key and compare it to the published ciphertext. Then the adversary knows whether a or b is encryted. The counterexample provided by Tamarin shows the execution.

4. Modeling and Specifying Properties of AKMA

In this section, we describe the detailed model of AKMA protocol and specify its properties of interest in Tamarin.

4.1. Threat Model

As we mentioned above, Tamarin assumes Dolev-Yao model for attackers. Adversary obeys the assumption of encryption, i.e., they can decrypt the secret messages only when having the corresponding key. In addition, we consider more advanced security properties corresponding to more powerful adversaries or compromised parties, following the eCK model [5]. In particular, we take into account the possibility of key reveal and the possibility that some of the entities have been compromised. In our protocol, the SUPI and K of a compromised UE could be revealed and the adversary would impersonate its identity to communicate with HN and AF. If HN is compromised, the information in UDM would be revealed and all information of the subscribers would be leaked, together with their asymmetric encryption key pairs, which play an important role in other protocols such as 5G AKA. Following [5], we define the concept of *clean session* as follows:

Definition 4.1 (Clean session) *We say a session is* clean *if neither of the following conditions holds:*

1. *One of the parties is an adversary-controlled party. This means in particular that adversary could reveal all private information known to the party, and perform all communications and computations on behalf of the compromised party;*
2. *Any of the long-term, temporary and session keys is revealed by adversary.*

Considering the following lemma:

```
All x #i.  Secret(X) @i ==> not (Ex #j.  K(x) @j)
```

it would be unsatisfiable when the agent is compromised. We call an agent is *Honest*(written as `Honest(X)`) if and only if the agent is not compromised. We indicate assumptions on honest agents by labeling the corresponding rule that the required action fact appears in with an `Honest(A)` action fact, where we assume A is honest. Intuitively, we explain the meaning of *Honest* by comparing the case where Honest is present in the properties and actions, and the case where it is not. If Honest is not present, then the meaning is that secrecy (or some other desired property) can be violated when *any* agent is compromised, whereas if Honest is present, then the meaning is that the desired property can be violated only when an agent participating in the protocol is compromised.

Therefore, following standard techniques of modeling using Tamarin [7, 31], we model *Honest* participants and key reveals as follows. For each long-term, temporary, and session key that could be revealed, we add a rule which outputs the key (so it becomes known to the adversary), with an action of the form `Reveal(X,type)`, where `X` is the participant who owns the key, and `type` specifies the type of the key. Moreover, at steps of the protocol where `Running`, `Commit` and `Confirmation` actions are inserted (see the protocol rules in 4.2.4), we also insert actions of the form `Honest(X)`, which indicates that `X` should be an honest participant of the protocol, i.e., should not be compromised. Hence, `Ex X m #r.  Reveal(X, m) @ r & Honest(X) @ i` means some participant of the protocol who is running (or finished) at time `i` has its secret key revealed (the session is not clean) at some time `r`. With this proposition, the considered lemma would be modified:

```
All x #i. Secret(X) @i
==> not (Ex #j. K(x) @j) | (Ex X m #r. Reveal(X, m) @r & Honest(X)@i)
```

Propositions of this form will appear frequently in the properties stated below, which are usually of the form *either security conditions are satisfied, or the session is not clean.*

4.2. Modeling the AKMA Service in Tamarin

In this part, we analyze the functions and behaviors of AKMA service, including some of the underlying assumptions, then describe the model of the protocol in Tamarin.

4.2.1. Assumptions

As mentioned in Section 2, we make several reasonable assumptions about AKMA service:

1. Communication between UE and AF occurs along public channels. Hence it is subject to eavesdropping, interception and injection by the adversary. The protocol should remain secure under such attacks.
2. We assume that the communication inside HN is always clean and credible. See the definition of clean in Section 4.1.
3. As mentioned previously, we consider the case where AF is located outside the HN. However, we do not include the NEF function and AAnF selection step in our model. Relaxing this assumption requires only dividing the communication between AF and AAnF to two communication steps, which should not affect the security arguments about the protocol.
4. Mutual authentication between AAnF and AF occurs before running AKMA using the TLS protocol [1], which provides integrity, replay, and secrecy protection of communication along a private channel. Following previous work [31, 42], we abstract this to a secure channel between HN and AF. In Tamarin, the channel is modeled with four rules, representing four behaviors respectively: sending messages into the channels, receiving messages from the channel, injecting messages into the channel, and eavesdropping messages from the channel (the latter two describe the behavior of the adversary):

```
rule send_secure:
  [ SndS(~cid,A,B,m) ] --> [ Sec(~cid,A,B,m) ]
rule receive_secure:
  [ Sec(~cid,A,B,m) ] --> [ RcvS(~cid,A,B,m) ]
rule secureChannel_compromised_in:
```

```
  [ In(<~cid,A,B,x>) ]
  --[Reveal(A,'secureChannel'), Injected(x)]->
  [ Sec(~cid,A,B,x) ]
rule secureChannel_compromised_out:
  [ Sec(~cid,A,B,m) ]
  --[ Reveal(B,'secureChannel') ]->
  [ Out(<~cid, m>) ]
```

5. Primary authentication using AKA is a prerequisite but not a proper part of AKMA service, and there are already a lot of work on analyzing the 5G AKA protocol [31, 33]. Therefore, we assume the communication between HN and UE to be secure and private.

Significantly, we make some assumptions about permanent information: the subscriber credentials, i.e. SUPI, K of the UE, which are shared between UE and HN, should initially be secret, provided they are not compromised.

We also make some assumptions about compromised entities. In our model, there are no private and permanent information related to AFs. Therefore, we only need to consider compromised UE and HN. For compromised UEs, adversaries would know all secret information like SUPI and K. Likewise, adversaries could access SUPI and K of all subscribers from compromised HNs (Although in a very small probability).

4.2.2. KDFs in the protocol

Parameters of each key derivation function have been specified by 3GPP. These are abstracted for convenience of modeling. We define the KDF of K_{AUSF} with three parameters: identity of HN, K of UE and the random number HN sent to UE, while actually the parameters of K_{AUSF} derivative function contains $\langle CK, IK \rangle$ generated from K of UE, identity of HN and the random number; The A-KID and K_{AKMA} are generated from the same key K_{AUSF}, and the only difference is the setting of constants, so the parameters are SUPI of UE, K_{AUSF} and C_1 (or C_2); The parameters of the KDF of K_{AF} contain K_{AKMA} and identity of the AF; The extenal ID GPSI translate function GPSI(ID).

4.2.3. Implicit Authentication

We equip the model with an optional key-confirmation round trip where the UEs and AFs confirm their target session key K_{AF} by MACing different

constants. Although the key-confirmation phase is not mentioned in the requirements and specification of the protocol, we find that the AKMA service fails to meet several security goals, especially authentication, without the key-confirmation phase. We introduce the details of the key-confirmation in the protocol rules in Section 4.2.4 and later when analyzing the verification results, we will show the importance of this phase.

4.2.4. Protocol rules

We list some important rules in the protocol and the corresponding Tamarin code below, in order to illustrate the modeling process.

Primary Authentication. Before UE starts a communication with AF, UE should execute primary authentication with HN. As it is not a key part of this protocol, we assume it is carried out successfully and only show the results of primary authentication, i.e., both UE and HN (the AUSF function) possess the key K_{AUSF}. And after the AKMA materials are derived, AUSF will transfer the AKMA materials to AAnF.

```
rule Init_Pri_Auth:
  let
    K_AUSF = KDF_AUSF(~K, ~id_HN, ~R)
  in
  [ Fr(~R),
    !Sub_K(~SUPI, ~id_HN, ~K)]
  --[ Pri_Auth(K_AUSF, ~SUPI, ~id_HN) ]->
  [ UE_Auth(~SUPI, K_AUSF, ~id_HN),
    AUSF(~SUPI, ~id_HN, K_AUSF) ]

rule K_AKMA_Register:
  [ AUSF_KEY(~SUPI, K_AUSF, ~id_HN, K_AKMA, <A_TID, ~id_HN>)]
  --[ K_AKMA_Register(~id_HN) ]->
  [ AAnF(~id_HN, ~SUPI, <A_TID, ~id_HN>, K_AKMA) ]
```

Re-primary Authentication. When AAnF receives a new AKMA key via fact AUSF_KEY, it deletes the old AKMA key materials by removing AAnF1 (to be produced by Rule K_AF_Generation_AAnF presented later this section) and only stores the latest message from AUSF by adding AAnF. The restriction in the action indicates that the rule would only trigger when K_AKMA_new does not equal K_AKMA and A_TID_new does not equal A_TID.

```
rule Re_pri_auth:
  [ AAnF1(~id_HN, ~SUPI, <A_TID, ~id_HN>, K_AKMA),
    AUSF_KEY(~SUPI, K_AUSF, ~id_HN, K_AKMA_new, <A_TID_new, ~id_HN>) ]
  --[ _restrict(NotEqual(K_AKMA_new, K_AKMA)),
      _restrict(NotEqual(A_TID_new, A_TID)),
      K_AKMA_Re_Register(~id_HN) ]->
  [ AAnF(~id_HN, ~SUPI, <A_TID_new, ~id_HN>, K_AKMA_new) ]
```

Application Session Establishment Request. After UE and HN generated AKMA key materials, UE starts a session request to AF with its A-KID (containing the AKMA Temporary UE Identifier A-TID and id-HN). We show the complete rule UE_send_request constructed in Tamarin as follows. Fact UE_Auth_KEY indicates that UE possesses all the information defined by the parameters. Meanwhile, UE will generate corresponding session key K_{AF} and a new fact UE_KEY, meaning the UE possesses more information containing the K_{AF}, should be added into the conclusion of this rule. `~tid` is the session id from the UE's point of view.

For two-party protocols, to analyze the desired authentication properties, we label the appropriate rules in the responder party B with an action fact `Commit(b, a, <‘A’, ‘B’, t>)` and in the initiator party A with the corresponding action fact `Running(a, b, <‘A’, ‘B’, t>)`. Likewise, `Confirmation(a, b, <‘A’, ’B’, t>)` is added into the action fact in appropriate rules. These actions are used for defining agreement between different parties required by the protocol.

In the following rule, for the actions, `Secret(<‘A_KID’, <A_TID, ~id_HN>>, ~SUPI)` means that the information `SUPI` should be kept secret; `Running(<A_TID, ~id_HN>, ~id_AF, <‘UE’, ‘AF’, <‘A_KID’, <A_TID, ~id_HN>>>)` means that the UE (here has the temporary ID A-KID, which includes A-TID and the entity HN) sends the A-KID to the AF; `Honest(~id_AF)` means that the entity AF, which is assumed honest, participates in this rule. The meaning of other actions can be understood similarly.

```
rule UE_send_request:
  let
    K_AF = KDF_AF(K_AKMA, ~id_AF)
  in
  [ UE_Auth_KEY(~SUPI, K_AUSF, K_AKMA, <A_TID, ~id_HN>),
    Fr(~tid),
    !Sub(~SUPI, ~id_HN),
```

```
    !AF(~id_AF) ]
  --[ UE_send_request(~SUPI),
      Secret(<‘A_KID’, <A_TID, ~id_HN>>, ~SUPI),
      Running(<A_TID, ~id_HN>, ~id_AF,
      <‘UE’, ‘AF’, <‘A_KID’, <A_TID, ~id_HN>>>),
      Running(~SUPI, ~id_HN, <‘UE’, ‘HN’, <‘A_KID’, <A_TID, ~id_HN>>>),
      Running(<A_TID, ~id_HN>, ~id_AF, <‘UE’, ‘AF’, <‘K_AF’, K_AF>>),
      Honest(<A_TID, ~id_HN>),
      Honest(~id_AF),
      Honest(~id_HN) ]->
  [ Out(<A_TID, ~id_HN>),
    UE_KEY(~tid, ~SUPI, K_AUSF, K_AKMA, <A_TID, ~id_HN>, K_AF, ~id_AF) ]
```

Naanf AKMA ApplicationKey Get Request. After receiving the communication request from UE, AF forwards the request of UE with the identity of AF to HN, indicated by msg, via a secure channel cid. The action `Commit(~id_AF, <A_TID, ~id_HN>, <‘UE’, ‘AF’, <‘A_KID’, <A_TID, ~id_HN>>>)` is corresponding to the `Running` action in Rule `UE_send_request`, and it means that the AF responds to UE with the same message.

```
rule AF_send_KeyRequest:
  let
    msg = < <A_TID, ~id_HN>, ~id_AF >
  in
  [ !AF(~id_AF),
    In(<A_TID, ~id_HN>),
    Fr(~cid),
    Fr(~tid) ]
  --[ AF_send_KeyRequest(~id_AF),
      Secret(<‘A_KID’, <A_TID, ~id_HN>>, ~id_AF),
      Commit(~id_AF, <A_TID, ~id_HN>,
      <‘UE’, ‘AF’, <‘A_KID’, <A_TID, ~id_HN>>>),
      Running(~id_AF, ~id_HN, <‘AF’, ‘HN’, <‘A_KID’, <A_TID, ~id_HN>>>),
      Running(~id_AF, ~id_HN, <‘AF’, ‘HN’, <‘id_AF’, ~id_AF>>),
      Honest(~id_AF),
      Honest(<A_TID, ~id_HN>),
      Honest(~id_HN) ]->
  [ SndS(~cid, ~id_AF, ~id_HN, msg),
    AF_request(~tid, ~id_AF, <A_TID, ~id_HN>, ~cid)  ]
```

Derive AF key. HN generates the session key K_{AF} based on the AKMA key and AF, using function `KDF_AF`. After K_{AF} is generated, a new fact `AAnF1` is produced to indicate that the AKMA key materials have already been used for the generation of the session key.

```
rule K_AF_Generation_AAnF:
  let
    K_AF = KDF_AF(K_AKMA, ~id_AF)
  in
  [ AAnF(~id_HN, ~SUPI, <A_TID, ~id_HN>, K_AKMA),
    !AF(~id_AF) ]
  --[ Session_Key_Generation_AAnF(~id_HN) ]->
  [ AAnF_KEY(~id_HN, ~SUPI, <A_TID, ~id_HN>, K_AKMA, K_AF, ~id_AF),
    AAnF1(~id_HN, ~SUPI, <A_TID, ~id_HN>, K_AKMA) ]
```

Naanf AKMA ApplicationKey Get Response. HN generates the session key K_{AF} and sends it through the secure channel cid with the expiration time, the GPSI identity (indicated by session_msg together) back to AF as a response.

```
rule AAnF_Send_K_AF:
  let
    session_msg = < K_AF, ~exptime, GPSI(<A_TID, ~id_HN>) >
    msg_In = < <A_TID, ~id_HN>, ~id_AF >
  in
  [ Fr(~exptime),
    AAnF_KEY(~id_HN, ~SUPI, <A_TID, ~id_HN>, K_AKMA, K_AF, ~id_AF),
    RcvS(~cid, ~id_AF, ~id_HN, msg_In) ]
  --[ HN_Response(~id_HN, K_AF),
      Secret(<'K_AF', K_AF>, ~id_HN),
      Commit(~id_HN, ~SUPI, <'UE', 'HN', <'A_KID', <A_TID, ~id_HN>>>),
      Commit(~id_HN, ~id_AF, <'AF', 'HN', <'A_KID', <A_TID, ~id_HN>>>),
      Commit(~id_HN, ~id_AF, <'AF', 'HN', <'id_AF', ~id_AF>>),
      Running(~id_HN, ~id_AF, <'HN', 'AF', <'K_AF', K_AF>>),
      Running(~id_HN, ~id_AF, <'HN', 'AF', <'K_AF_exptime', ~exptime>>),
      Honest(~id_HN),
      Honest(~id_AF) ]->
  [ SndS(~cid, ~id_HN, ~id_AF, session_msg) ]
```

Application Session Establishment Response. In the specification [2], after receiving the session key together with other information from HN, AF would return a response without any parameters to UE. As mentioned above, in order to let UE and AF confirm the session key, we suggest that there should be an implicit authentication started by AF. As shown by the following rule, AF hashes the session key with "AF" via function `f` and sends the hash value to UE.

```
rule AF_Response_Key:
  let
    confmess1 = f(K_AF, 'AF')
    msg_IN = < K_AF, ~exptime, GPSI(<A_TID, ~id_HN>) >
  in
  [ RcvS(~cid, ~id_HN, ~id_AF, msg_IN),
    AF_request(~tid, ~id_AF, <A_TID, ~id_HN>, ~cid) ]
  --[ AF_session_response(~id_AF, K_AF),
      Secret(<'K_AF', K_AF>, ~id_AF),
      Commit(~id_AF, <A_TID, ~id_HN>, <'UE', 'AF', <'K_AF', K_AF>>),
      Commit(~id_AF, ~id_HN, <'HN', 'AF', <'K_AF', K_AF>>),
      Commit(~id_AF, ~id_HN, <'HN', 'AF', <'K_AF_exptime', ~exptime>>),
      Honest(~id_AF),
      Honest(~id_HN),
      Honest(<A_TID, ~id_HN>)
  ]->
  [ Out(confmess1),
    AF_Confirmation(~tid, ~id_AF, K_AF, <A_TID, ~id_HN>, ~cid) ]
```

Implicit Authentication. Here we present the rest rules for implicit authentication. After AF sends the hash to UE, UE would confirm the hash value, then hash the session key with "UE" and send the hash value to AF. The implicit authentication is finished when UE and AF have both confirmed the hash values. The key confirmation phase is listed as followed:

```
rule UE_Key_Confirmation:
  [ In(f(K_AF, 'AF')),
    UE_KEY(~tid, ~SUPI, K_AUSF, K_AKMA, <A_TID, ~id_HN>, K_AF, ~id_AF),
    !AF(~id_AF) ]
  --[ UE_Key_Confirmation(~SUPI, K_AF),
      Confirmation(<'UE', ~SUPI>, <'AF', ~id_AF>,
      <'UE', 'AF', <'K_AF', K_AF>>) ]->
  [ Out(f(K_AF, 'UE')) ]
```

```
rule AF_Key_Confirmation:
  [ In(f(K_AF, 'UE')),
    AF_Confirmation(~tid, ~id_AF, K_AF, <A_TID, ~id_HN>, ~cid) ]
  --[ End_protocol(~id_AF, K_AF),
      Confirmation(<'AF', ~id_AF>, <'UE', <A_TID, ~id_HN>>,
      <'UE', 'AF', <'K_AF', K_AF>>)]->
  []
```

4.3. Security Requirements and Properties Specification

Now we introduce common security requirements and describe them as properties in the Tamarin prover. In 5G system, the requirements in the Technical Specification and Requirements are mainly divided into three parts: Authentication, Secrecy and Privacy. In the following, we will show the properties that we concern and its specification in Tamarin.

4.3.1. Authentication

Before specifying authentication properties, we first introduce Lowe's taxonomy of authentication properties [6], which specifies four authentication levels from one party's view and many security properties are extended from these four basic properties. Considering the authentication of the given two parties A and B, from party A's point of view, the authentication levels are defined as follows:

1. **Aliveness:** Whenever A completes a run of the protocol, apparently with B, then B has previously been running the protocol (not necessarily with A);
2. **Weak agreement:** Whenever A completes a run of the protocol, apparently with B, then B has previously been running the protocol, apparently with A (but not necessary agreeing on the same messages);
3. **Non-injective agreement:** In addition to the condition for weak agreement, the parties A and B also agree on the same message;
4. **Injective agreement:** In addition to the conditions for non-injective agreement, there is a unique matching partner instance for each completed run of an agent, which effectively prevents replay attacks.

In Technical Specifications and Technical Requirements by 3GPP [11, 1, 2], we find that many security requirements are based on these four authentication properties, as well as secrecy of messages. Therefore, we will mainly characterize security of AKMA service in terms of these properties.

Property 4.1 (Agreement between UE and AF) *By the end of the protocol execution, AF must obtain injective agreement on K_{AF}, and weak agreement with UE.*

This one is the most important property as the target of the protocol is to make UE and a specific AF perform authentication and agree on a session key K_{AF}, therefore the UE can communicate with the AF. As we add an optional key-confirmation round trip between UE and AF to the protocol, there would be two models, one with the key confirmation and one without. Here we consider the agreement properties with respect to both cases. The first three properties are weak agreement, non-injective agreement, injective agreement without key confirmation, and the last three are the properties with key confirmation, respectively. We select two representative lemmas to display here.

This lemma states that AF satisfies injective agreement on `K_AF` with UE, in the case without key confirmation.

```
lemma Injective_agreement_UE_AF_without_KC:
    all-traces
    "All A B t #i. Commit(A, B, <‘UE’, ‘AF’, <‘K_AF’, t>>) @i
    ==> (Ex #j. Running(B, A, <‘UE’, ‘AF’, <‘K_AF’, t>>) @j
        & not(Ex A2 B2 #i2.
        Commit(<‘AF’, A2>, <‘UE’, B2>, <‘UE’, ‘AF’, <‘K_AF’, t>>) @i2
        & not(#i2 = #i)))
        | (Ex D m #l. Reveal(D, m) @l & Honest(D) @i )
        | (Ex D #l. Reveal(D, ‘secureChannel’)@l & Honest(D) @i)"
```

This lemma states that AF satisfies weak agreement with UE, in the case with key confirmation.

```
lemma weak_agreement_UE_AF_with_KC:
    all-traces
    "All A B t1 #i.
    Confirmation(<‘AF’, A>, <‘UE’, B>, t1) @i
    ==> (Ex t2 #j . Running(B, A, t2) @j
        & j < i)
        | (Ex D m #l. Reveal(D, m) @l & Honest(D) @i )"
```

Property 4.2 (Agreement between UE and HN) *By the end of the protocol execution, HN must obtain weak agreement with UE.*

```
lemma weakagreement_UE_HN:
    all-traces
    "All A B t #i. Commit(A, B, <‘UE’, ‘HN’, t>)@i
    ==> (Ex t2 #j. Running(B, A, t2)@j)
        | (Ex X m #r. Reveal(X, m)@r & Honest(X)@i)
        | (Ex D #l. Reveal(D, ‘secureChannel’)@l & Honest(D) @i) "
```

Property 4.3 (Agreement between AF and HN) *By the end of protocol execution, the AF and HN must both obtain injective agreement on K_{AF} and A-KID, and weak agreement with each other.*

This property ensures that HN and AF participate in the same session started by one UE. Injective agreement between AF and HN (agreeing on the target session key K_{AF}) is defined by the following lemma. The inverse property, i.e. injective agreement between HN and AF, can be defined similarly.

```
lemma Injective_agreement_AF_HN_K_AF:
  all-traces
    "All A B t #i. Commit(A, B, <‘HN’, ‘AF’, <‘K_AF’, t>>) @i
    ==> (Ex #j. Running(B, A, <‘HN’, ‘AF’, <‘K_AF’, t>>) @j
        & not(Ex A2 B2 #i2.
        Commit(<‘AF’, A2>, <‘HN’, B2>, <‘HN’, ‘AF’, <‘K_AF’, t>>) @i2
        & not(#i2 = #i)))
        | (Ex D m #l. Reveal(D, m) @l & Honest(D) @i )
        | (Ex D #l. Reveal(D, ‘secureChannel’)@l & Honest(D) @i)"
```

Injective agreement between AF and HN (agreeing on the same A-KID) is defined by the following lemma. The inverse property, i.e. injective agreement between HN and AF, can be defined similarly.

```
lemma Injective_agreement_AF_HN_A_KID:
    all-traces
    "All A B t #i. Commit(A, B, <‘HN’, ‘AF’, <‘A_KID’, t>>) @i
    ==> (Ex #j. Running(B, A, <‘HN’, ‘AF’, <‘A_KID’, t>>) @j
        & not(Ex A2 B2 #i2.
        Commit(<‘AF’, A2>, <‘HN’, B2>, <‘HN’, ‘AF’, <‘A_KID’, t>>) @i2
        & not(#i2 = #i)))
        | (Ex D m #l. Reveal(D, m) @l & Honest(D) @i )
        | (Ex D #l. Reveal(D, ‘secureChannel’)@l & Honest(D) @i)"
```

4.3.2. Secrecy

We mainly focus on the secrecy of the target session key, i.e., the K_{AF}, as the privacy goal usually implies the secrecy of some important information, such as, session key and long-term user identity, which we will mention later. This protocol must prevent the session key K_{AF} from being revealed, i.e., adversaries will never know the session key. The property is specified as follows:

Property 4.4 (Session Key Secrecy) *K_{AF} must be kept secret.*

```
lemma secure_K_AF:
  all-traces
    "All n A #i. Secret(<'K_AF', n>, A) @i
    ==> (not (Ex #j. K(n) @j))
        | (Ex X data #r. Reveal(X, data) @r & Honest(X) @i)"
```

Moreover, we verify that A-KID will not be kept secret by the protocol, as the secrecy of A-KID is not a goal in its design. However, we note that the counterexample to be described later exposing a security problem in AKMA contains the leakage of A-KID as one of the initial steps. Hence, it may be advantageous to modify the protocol to keep A-KID secret as well.

Property 4.5 (A-KID Secrecy) *A_KID may be kept secret.*

```
lemma secure_A_KID:
  all-traces
    "All n A #i. Secret(<'A_KID', n>, A) @i
    ==> (not (Ex #j. K(n) @j))
        | (Ex X data #r. Reveal(X, data) @r & Honest(X) @i)"
```

4.3.3. Privacy

Privacy is also an important part in the AKMA study phase, which concentrates on protecting security of user identity and personal information, e.g., anonymity, unlinkability, etc. Privacy issues in delegated authentication systems have been identified by many researchers. In [12, 13], Khan *et al* summarized three of the most frequently identified privacy issues in 5G AKMA service, and they are specified in natural language as follows:

1. The AFs should not know a user's identity at the HN;

2. One AF cannot link one of its users with a user of another AF, even when those two AFs collude;
3. The HN should not know the name of the AFs that a UE connects to.

For the first privacy issue, we decompose this issue into two properties that can be formally described: secrecy of UE's identity SUPI and UE indistinguishability. The former one, which is also one of the most important requirements in communication system, can be described as maintaining secrecy of identity SUPI, while the latter one is specified as an observational equivalence property, and will be verified in the observational equivalence mode in Tamarin.

Property 4.6 (Secrecy of SUPI) *SUPI must be kept secret.*

```
lemma secure_SUPI:
  all-traces
    "All n A #i. Secret(<'SUPI', n>, A) @i
    ==> (not (Ex #j. K(n) @j))
        | (Ex X data #r. Reveal(X, data) @r & Honest(X) @i)"
```

The secrecy of SUPI obviously holds as SUPI is never transferred along any channel in the protocol.

Property 4.7 (UE indistinguishability) *Given two UE entities denoted by UE_1 and UE_2, and an AKMA execution started by UE_1 (or UE_2), no attacker can determine whether it is interacting with UE_1 or UE_2.*

In the observational equivalence mode in Tamarin, the model will have two instances corresponding to two UE entities respectively, and then Property 4.7 can be specified by the default built-in property `Observational_Equivalence`.

For the second privacy issue, it is not able to be specified in Tamarin. But later in next section, we will describe the linkability attack, showing the unlinkablity property is not satisfied by AKMA service.

For the last privacy issue, actually this one can not be satisfied as when UE starts an AKMA service, HN would know the AKMA key materials (containing K_{AKMA} and A-KID), and when AF forwards the request of UE (containing A-KID) with its own identity, HN would connect A-KID and the corresponding AF. As HN has generated A-KID from SUPI, therefore HN can connect SUPI and AF, knowing the name of the AFs that a UE connects

to. [12] provided a privacy mode of AKMA service to fix this flaw, but it is rather complex and has not yet been widely used.

Except for all the above properties, we specify the executability of the AKMA protocol, i.e., it is possible to complete the protocol and agree on a session key for the first time and more than once.

We specified our model and properties through Tamarin[1]. The total number of lines of code is approximately 700.

5. Results and Analysis

We verify the properties listed in Section 4.3 using Tamarin. The verification results are shown in Table 1 and Table 2. For the properties that fail to hold, we analyze the counterexamples returned by Tamarin. Moreover, based on the analysis, we put forward some potential attacks and suggestions.

Agreement	Weak		Non-Injective		Injective	
Results	SAT	Time	SAT	Time	SAT	Time
Without KC	×	3s	×	2s	×	2s
With KC	✓	23s	✓	24s	-	∞

Table 1: Agreement between UE and AF

5.1. Verification Results and Analysis

Figure 1 lists the verification results for agreement between UE and AF, and Figure 2 lists the verification results for the other security properties. We will explain them below. First of all, the executability of the protocol (existence of one execution), the weak agreement between UE and HN turn out to be correct. The mutual agreement between AF and HN (containing weak agreement, injective agreement on K_{AF} and A-KID) turns out to be correct, showing that the session they participate in is the same one at a time.

The secrecy of K_{AF} turns out to be correct, indicating that the protocol indeed protects the secrecy of the session key. The secrecy of A-KID turns

[1]The code is publicly available at https://github.com/TengshunYang/5G-AKMA.

Properties	Satisfaction	Time
Injective agreement between AF and HN on A-KID	✓	2.5s
Injective agreement between AF and HN on K_{AF}	✓	2.3s
Injective agreement between HN and AF on A-KID	✓	2.2s
Injective agreement between HN and AF on K_{AF}	✓	2s
Weak agreement between UE and HN on A-KID	✓	6.6s
Executability without re-primary authentication	✓	16.5s
Executability with re-primary authentication	✓	5s
Secrecy of K_{AF}	✓	24s
Secrecy of A-KID	×	1.7s
UE indistinguishability	✓	5s

Table 2: Other Security Properties

out to be incorrect, which is obvious because A-KID is transferred along public channels.

About the privacy property, the UE indistinguishability turns out to be correct, indicating that any adversary, even the AF, could not distinguish two UEs just according to the messages transferred in the channel all alone.

Next we discuss the main properties, i.e. agreement between UE and AF, as presented in Table 1.

Agreement between UE and AF. For the agreement between UE and AF without key confirmation, there are three properties respectively corresponding to three authentication levels: weak agreement, non-injective agreement and injective agreement. All of them turn out to be incorrect and Tamarin returns the same counterexample: (1) UE starts a session request to an AF (denoted by AF_1) with A-KID of UE; (2) the leakage of A-KID occurs, then adversary M connects another AF (denoted by AF_2) with this A-KID; (3) AF_2 thinks that UE should have connected AF_2 before and asks HN for the session key K_{AF}, while UE only connects to AF_1 and generates K_{AF1}. Therefore M would not communicate with AF_2 and could not complete the protocol. In conclusion, if there is no implicit authentication to confirm the session key, weak agreement between UE and AF would not be satisfied.

According to the verification results, the weak agreement and non-injective agreement on K_{AF} between UE and AF with key confirmation turn out to be correct. Compared with the case that without key confirmation, the authentication level is promoted. Therefore, we suggest adding a key confirmation

round trip in the protocol, which will improve the security and authentication level. Unfortunately, we fail to verify the injective agreement between UE and AF in the case with key confirmation as the verification process does not terminate. May be we need to turn to the interactive mode provided by Tamarin.

linkability between AFs. Now we will review a violated privacy issue mentioned in Section 4.3.3. According to the counterexamples provided by Tamarin, we find that the main problem is the leakage of A-KID. In real life, adversaries would eavesdrop the A-KID, or a malicious AF would play the role of adversary and forward the received A-KID to another AF, i.e. linkability between AFs, which is also mentioned as a privacy violation in [12, 13]. Adversary could impersonate UE's identity and start a session with AF. Although the adversary has no way to obtain the session key except by stealing from the UE, it would result in waste of trust and materials. Here we describe the situation of *linkability between AFs* as follows: (1) UE starts a session with an AF (denoted by AF_1), and completes AKMA service with AF_1 successfully; (2) With the possession of A-KID, AF_1 would forward it to another AF (denoted by AF_2). Knowing the A-KID helps AF_2 distinguish the UEs, even without knowing the user's true identity. AFs in the collusion group would share all the information of users with the same A-KID with each other, which would result in leakage of users such as history, hobbies and habits, etc. After combining all the information, the user's true identity could be revealed.

Moreover, in our conference paper [8], we propose a fix to add A-KID to the message sent from HN to AF. In the newest version of Technical Specification [2], as we mentioned in Section 2, a similar information GPSI (the extenal ID of UE) is added into the message sent from HN to AF.

5.2. Suggestions

According to the results of verification using Tamarin, several of the security properties that we expect to hold actually fail for the initial model we constructed for AKMA. We find that leakage of A-KID plays an important role in disturbing the protocol, such as, waste of materials and causing linkability between AFs, which are harmful to users' privacy. So we suggest adding protection for the communication of A-KID. For example, pre-constructing a channel for UE and AF with asymmetric encryption, or using TLS protocol,

can both achieve this. Aiming at resolving the collusion among AFs, dynamic A-KID or increasing the frequency of primary authentication is worth considering.

The second suggestion we propose is the addition of key confirmation. The protocol containing this implicit authentication can satisfy more authentication properties, improve authentication and security level.

At last, in the technical specification [2], the session key K_{AF} can still be used while UE restarts a primary authentication. We find that the leakage of K_{AF} would result in the situation where more than one dishonest UEs (impersonating the original UE) connect to one AF with the leaked K_{AF}, which would use the service from AF or even steal private information, although these dishonest UEs have never started primary authentication. We suggest that HN could inform the AF when the session key K_{AF} expires ahead of the time when UE re-starts a primary authentication. It would reduce the risk of leakage, at the price of only one message.

6. Conclusion

We have formalized for the first time the 5G AKMA service specified in TS 33.535 [2], using Tamarin verification tool. The formalization includes the formal model of the AKMA service, the properties including authentication, secrecy and privacy that are expected to hold, the verification, the potential attacks of the AKMA service and some suggestions for fixing the problems. During the modeling, we identify formally the assumptions for the security properties to hold. For the security properties that do not hold, we analyze the corresponding counterexamples and construct the potential attacks, and at the end, suggest some fixes for the model to resolve the attacks and weaknesses. For future work, we will continue to follow the future development of the AKMA standard and update the formalization. We will also consider more privacy requirements of 5G AKMA and their formalization, e.g. the privacy caused by the linkability between AFs mentioned in this paper deserving further consideration.

References

[1] 3GPP, TS33.501 v17.1.0 Security architecture and procedures for 5G system (Release 17), `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3169`, 2021.

[2] 3GPP, TS33.535 v17.4.0 Authentication and Key Management for Applications (AKMA) based on 3GPP credentials in the 5G System (5GS), https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3690, 2021.

[3] S. Meier, B. Schmidt, C. Cremers, D. A. Basin, The TAMARIN prover for the symbolic analysis of security protocols, in: Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings, 2013, pp. 696–701.

[4] D. Dolev, A. C. Yao, On the security of public key protocols, IEEE Trans. Inf. Theory 29 (1983) 198–207.

[5] B. A. LaMacchia, K. E. Lauter, A. Mityagin, Stronger security of authenticated key exchange, in: Provable Security, First International Conference, ProvSec 2007, Wollongong, Australia, November 1-2, 2007, Proceedings, 2007, pp. 1–16.

[6] G. Lowe, A hierarchy of authentication specification, in: 10th Computer Security Foundations Workshop (CSFW '97), June 10-12, 1997, Rockport, Massachusetts, USA, 1997, pp. 31–44.

[7] B. Schmidt, S. Meier, C. J. F. Cremers, D. A. Basin, Automated analysis of diffie-hellman protocols and advanced security properties, in: 25th IEEE Computer Security Foundations Symposium, CSF 2012, Cambridge, MA, USA, June 25-27, 2012, 2012, pp. 78–94.

[8] T. Yang, S. Wang, B. Zhan, N. Zhan, J. Li, S. Xiang, Z. Xiang, B. Mao, Formal analysis of 5g AKMA, in: Dependable Software Engineering. Theories, Tools, and Applications - 7th International Symposium, SETTA 2021, volume 13071 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 102–121.

[9] 3GPP, TS33.220 v17.2.0 Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA), https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2280, 2021.

[10] 3GPP, TS33.163 v16.2.0 Battery Efficient Security for very low throughput Machine Type Communication (MTC) devices (BEST),

https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3128, 2019.

[11] 3GPP, TR33.835 v16.1.0 Study on authentication and key management for applications based on 3GPP credential in 5g, https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3483, 2020.

[12] M. Khan, P. Ginzboorg, V. Niemi, Privacy preserving AKMA in 5g, in: M. Mehrnezhad, T. van der Merwe, F. Hao (Eds.), Proceedings of the 5th ACM Workshop on Security Standardisation Research Workshop, London, UK, November 11, 2019, ACM, 2019, pp. 45–56.

[13] M. Khan, P. Ginzboorg, V. Niemi, AKMA: delegated authentication system of 5g, IEEE Commun. Stand. Mag. 5 (2021) 56–61.

[14] A. P. Sarr, P. Elbaz-Vincent, J. Bajard, A new security model for authenticated key agreement, in: J. A. Garay, R. D. Prisco (Eds.), Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. Proceedings, volume 6280 of *Lecture Notes in Computer Science*, Springer, 2010, pp. 219–234.

[15] M. Bellare, P. Rogaway, Entity authentication and key distribution, in: D. R. Stinson (Ed.), Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings, volume 773 of *Lecture Notes in Computer Science*, Springer, 1993, pp. 232–249.

[16] R. Canetti, H. Krawczyk, Analysis of key-exchange protocols and their use for building secure channels, in: B. Pfitzmann (Ed.), Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding, volume 2045 of *Lecture Notes in Computer Science*, Springer, 2001, pp. 453–474.

[17] H. Krawczyk, HMQV: A high-performance secure diffie-hellman protocol, in: V. Shoup (Ed.), Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings, volume 3621 of *Lecture Notes in Computer Science*, Springer, 2005, pp. 546–566.

[18] S. A. Schneider, Security properties and CSP, in: 1996 IEEE Symposium on Security and Privacy, May 6-8, 1996, Oakland, CA, USA, IEEE Computer Society, 1996, pp. 174–187.

[19] S. Schneider, R. Holloway, Using CSP for protocol analysis: the Needham-Schroeder Public-Key Protocol, Technical Report, 1996.

[20] B. Donovan, P. Norris, G. Lowe, Analyzing a library of security protocols using casper and fdr, in: In Workshop on Formal Methods and Security Protocols, 1999.

[21] M. Burrows, M. Abadi, R. M. Needham, A logic of authentication, ACM Trans. Comput. Syst. 8 (1990) 18–36.

[22] M. Abadi, B. Blanchet, C. Fournet, The applied pi calculus: Mobile values, new names, and secure communication, J. ACM 65 (2018) 1:1–1:41.

[23] B. Blanchet, An efficient cryptographic protocol verifier based on prolog rules, in: 14th IEEE Computer Security Foundations Workshop (CSFW-14 2001), 11-13 June 2001, Cape Breton, Nova Scotia, Canada, IEEE Computer Society, 2001, pp. 82–96.

[24] A. Armando, D. A. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. H. Drielsma, P. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, L. Vigneron, The AVISPA tool for the automated validation of internet security protocols and applications, in: K. Etessami, S. K. Rajamani (Eds.), Computer Aided Verification, 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, July 6-10, 2005, Proceedings, volume 3576 of *Lecture Notes in Computer Science*, Springer, 2005, pp. 281–285.

[25] L. Lamport, The temporal logic of actions, ACM Trans. Program. Lang. Syst. 16 (1994) 872–923.

[26] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, C. L. Talcott (Eds.), All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic, volume 4350 of *Lecture Notes in Computer Science*, Springer, 2007.

[27] S. Escobar, C. A. Meadows, J. Meseguer, A rewriting-based inference system for the NRL protocol analyzer and its meta-logical properties, Theor. Comput. Sci. 367 (2006) 162–202.

[28] S. Meier, Advancing Automated Security Protocol Verification, Ph.D. thesis, ETH, 2013.

[29] R. Künnemann, G. Steel, Yubisecure? formal security analysis results for the yubikey and yubihsm, in: A. Jøsang, P. Samarati, M. Petrocchi (Eds.), Security and Trust Management - 8th International Workshop, STM 2012, Pisa, Italy, September 13-14, 2012, Revised Selected Papers, volume 7783 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 257–272.

[30] D. A. Basin, C. Cremers, T. H. Kim, A. Perrig, R. Sasse, P. Szalachowski, Design, analysis, and implementation of ARPKI: an attack-resilient public-key infrastructure, IEEE Trans. Dependable Secur. Comput. 15 (2018) 393–408.

[31] D. A. Basin, J. Dreier, L. Hirschi, S. Radomirovic, R. Sasse, V. Stettler, A formal analysis of 5g authentication, in: D. Lie, M. Mannan, M. Backes, X. Wang (Eds.), Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018, ACM, 2018, pp. 1383–1396.

[32] C. Cremers, M. Horvat, J. Hoyland, S. Scott, T. van der Merwe, A comprehensive symbolic analysis of TLS 1.3, in: B. M. Thuraisingham, D. Evans, T. Malkin, D. Xu (Eds.), Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017, ACM, 2017, pp. 1773–1788.

[33] Y. Wang, Z. Zhang, Y. Xie, Privacy-preserving and standard-compatible AKA protocol for 5g, in: M. Bailey, R. Greenstadt (Eds.), 30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021, USENIX Association, 2021, pp. 3595–3612.

[34] 3GPP, TR33.902 v4.0.0 3g Security; Formal Analysis of the 3g Authentication Protocol, `https://portal.3gpp.org/desktopmodules/`

Specifications/SpecificationDetails.aspx?specificationId=2337, 2001.

[35] M. Deng, K. Wuyts, R. Scandariato, B. Preneel, W. Joosen, A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements, Requir. Eng. 16 (2011) 3–32.

[36] D. J. Solove, A taxonomy of privacy, University of Pennsylvania Law Review 154 (2006) 477–560.

[37] J. Bohli, A. Pashalidis, Relations among privacy notions, in: R. Dingledine, P. Golle (Eds.), Financial Cryptography and Data Security, 13th International Conference, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected Papers, volume 5628 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 362–380.

[38] N. Dong, H. Jonker, J. Pang, Enforcing privacy in the presence of others: Notions, formalisations and relations, in: J. Crampton, S. Jajodia, K. Mayes (Eds.), Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings, volume 8134 of *Lecture Notes in Computer Science*, Springer, 2013, pp. 499–516.

[39] 3GPP, TS33.102 v16.0.0 3G Security; Security architecture, https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2262, 2020.

[40] 3GPP, TS33.401 v16.3.0 3GPP System Architecture Evolution (SAE); Security architecture, https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2296, 2020.

[41] Tamarin Team, Tamarin-Prover Manual: Security Protocol Analysis in the Symbolic Model, https://tamarin-prover.github.io/manual/, Accessed: January 7, 2021.

[42] D. A. Basin, S. Radomirovic, L. Schmid, Modeling human errors in security protocols, in: IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016, IEEE Computer Society, 2016, pp. 325–340.