Formal Analysis of 5G AKMA

Tengshun Yang[1,2], Shuling Wang[1,2], Bohua Zhan[1,2(✉)], Naijun Zhan[1,2], Jinghui Li[3], Shuangqing Xiang[3], Zhan Xiang[3], and Bifei Mao[3]

[1] SKLCS, Institute of Software, CAS, Beijing, China
{yangts,wangsl,bzhan,znj}@ios.ac.cn
[2] University of Chinese Academy of Sciences, Beijing, China
[3] Trustworthiness Theory Research Center, Huawei Technologies Co., Ltd., Shenzhen, China
{jinghui.li,xiangshuangqing,xiangzhan1,maobifei}@huawei.com

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 security properties extracted from informal descriptions given in the Technical Specifications. We identify the security 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 and fixes for the 5G AKMA service.

1 Introduction

With mobile communication networks widely used across the world, more and more people subscribe to their 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) [7]. One of the main mechanisms is to support authentication and key management aspects for applications, that is mutual authentication between users and

S. Qin et al. (Eds.): SETTA 2021, LNCS 13071, pp. 102–121, 2021.
https://doi.org/10.1007/978-3-030-91265-9_6

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 [8], 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 17.1.0 of Release 17 of the Technical Specification (TS) [8], we will provide the first formal model of 5G AKMA and also verify formally the security requirements using Tamarin.

Formal Methods. In this paper, we apply formal methods to analyze the AKMA service, using the Tamarin verification tool [31]. 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 [20] and eCK [27], as well as various authentication specifications [29]. Using a backward-search style algorithm [33], 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 [8], as a set of rewrite rules in Tamarin. As we describe in Sect. 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, weak agreement, non-injective agreement) 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.

Related Work. In the earlier generations of mobile network, the corresponding services were also specified by 3GPP. GBA (Generic Bootstrapping Architecture) [5] and BEST (Battery Efficient Security for very low Throughput Machine Type Communication (MTC) devices) [4], 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 [1]). In [23], 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 [24], they introduced designated 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 [20], eCK model [27], and its extension SeCK model [32]. The adversaries are given different powers for each of them. Especially, the eCK model inherits the spirit of Bellare and Rogaway [14] and Canetti and Krawczyk [17,25] 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 [21,29,34,35], BAN logic [16], applied π-calculus [9], Horn clauses [15], TLA [10,28], rewriting system [31] and so on. Some security protocol verification tools are developed based on these theories, such as Tamarin [31], Maude-NPA [18], ProVerif [15], and so on. Tamarin will be introduced in Sect. 3. The Maude-NPA tool [18] supports protocols specified as linear role-scripts and properties specified as symbolic states [22]. ProVerif [15] 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 protocols, YubiKey and YubiHSM protocols were considered in [26,30]. In [11], 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 [12,19], which are important for Internet security and also widely used to establish secure channels in a variety of contexts. Significantly, 3GPP [2] formally analyzes the 3G AKA protocol using TLA [28] on the absence of failure scenarios and uses BAN logic [16] on proving security goals respectively.

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. See the Technical Specification [8] for further information.

2.1 General Architecture

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

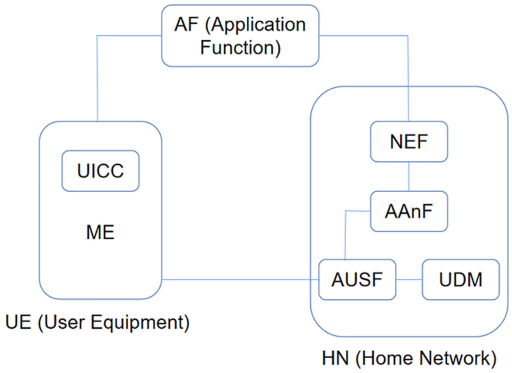

Fig. 1. AKMA architecture

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 users on session keys in the AKMA service. There are several functions located within the HN, as follows:
   - 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 user connects to when roaming. In this paper, we consider only the case when the user is not roaming, that is, SN is part of the HN, so 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 [7], 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 Sect. 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.

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 [7]). Prior generations of cellular networks have different AKA protocols: 3G has UMTS AKA protocol [3]; 4G has LTE AKA protocol [6]; in 5G, besides AKA protocol, there exists EAP-AKA′ [7]. 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, when 5G AKA protocol runs, HN sends a random number to UE. With the random number and information of the UE, both UE and AUSF in HN side would generate K_{AUSF}, which will be used for generating subsequent keys during AKMA.

Deriving AKMA Materials. The steps for deriving AKMA materials are shown in Fig. 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 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).

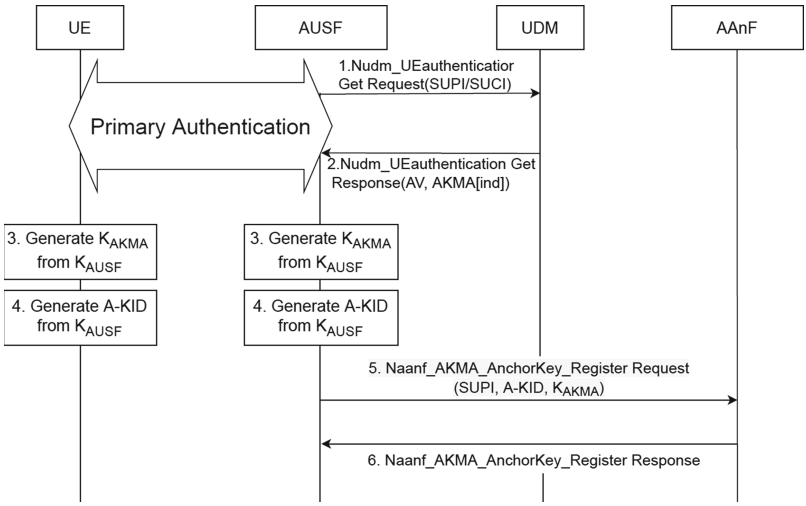

Fig. 2. Deriving AKMA materials (taken from [8])

Deriving AKMA Application Key for a Specific AF. The steps for deriving AKMA application key are shown in Fig. 3. 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.

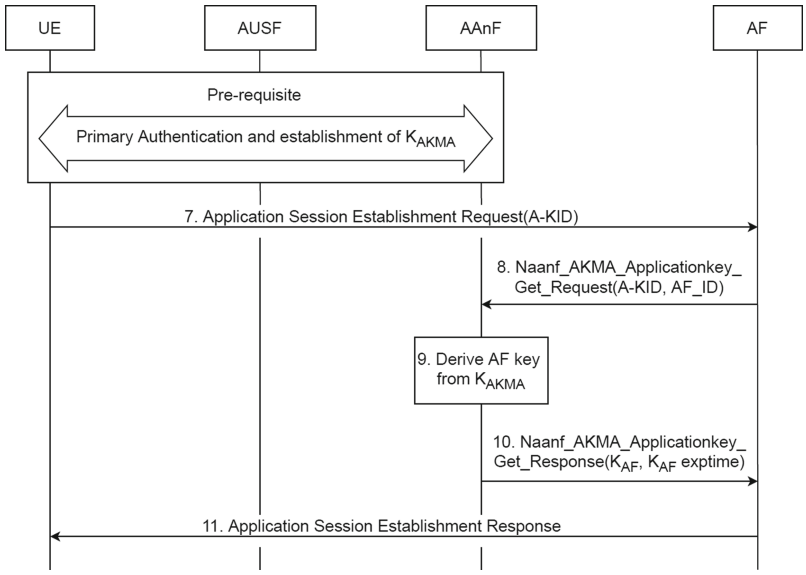

Fig. 3. Deriving AKMA application key for a specific AF (taken from [8])

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 inside HN or not.

If AF is located within HN, 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 key K_{AF} with its expiration time as a response back to AF (Step 10). 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, it connects to NEF rather than 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. Most of the procedure is the same as above.

When AF receives the session key K_{AF} and K_{AF} expiration time, it responds to UE (Step 11). 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.

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 [8] for more details.

3 Tamarin Prover

In this section, we give a brief introduction to the Tamarin verification tool [36]. 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 [36]. 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 [20], 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 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($A, k), Fr(m)] --[Send_mes(A, m)]-> [Out(enc(m, k))]
rule Recv_B: [!Ltk($B, k), In(enc(m, k))] --[Recv_mes(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

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 1, we show how to describe various levels of authentication specifications according to [29]. 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_mes(B, m) @ i ==> (EX #j. Send_mes(A, m) @ j & j < i)"
```

This property holds for the above example. The only way `Recv_mes(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_mes(A,m)` at an earlier timepoint.

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

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

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 [27]. 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 [27], we define the concept of *clean session* as follows:

Definition 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 [12,33], 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 Sect. 4.2), 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.

Assumptions. As mentioned in Sect. 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, as detailed in Sect. 4.1.
3. We only consider the case where AF can communicate directly with AAnF, without NEF as an intermediary. Hence, we do not include NEF in our model. Relaxing this assumption requires only changing the communication between AF and AAnF to taking two steps instead of only one step, 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 [7], which provides integrity, replay, and confidentiality protection of communication along a private channel. Following previous work [12,13], 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, eavesdropping messages from the channel, injecting messages into the channel (the latter two describe the behavior of the adversary).
5. Primary authentication using AKA is a prerequisite but not a proper part of AKMA service, and there are already a lot of work analyzing the 5G AKA protocol. 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. As we show in Sect. 5.1, the failure of non-injective agreement property is due to compromised 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.

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.

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

- We model the process of redoing primary authentication. When AAnF receives a new AKMA key via fact AUSF_KEY, it deletes the old AKMA key materials by removing AAnF1 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 HN-ID according to the TS [8]). Fact UE_KEY indicates that UE possesses all the information defined by the parameters. Fact UE_KEY1 is produced to indicate that these information does not disappear after this transition.
  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. We show the complete rule UE_send_request constructed in Tamarin as follows, but due to limited space, we will not list these actions in the remaining rules of this section.

```
rule UE_send_request:
  [ UE_KEY(~SUPI, K_AUSF, K_AKMA, <A_TID, ~id_HN>, K_AF, ~id_AF),
    !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_KEY1(~SUPI, K_AUSF, K_AKMA, <A_TID, ~id_HN>, K_AF, ~id_AF) ]
```

- Naanf AKMA ApplicationKey Get Request: AF forwards the request of UE with the identity of AF to HN, indicated by msg, via a secure channel cid.
- Naanf AKMA ApplicationKey Get Response: HN generates the session key K_{AF} and sends it through a message (indicated by session_msg) back to AF as a response.

```
rule AAnF_Send_K_AF:
  [ Fr(~exptime),
    AAnF_KEY(~id_HN, ~SUPI, <A_TID, ~id_HN>, K_AKMA, K_AF, ~id_AF),
    !AF(~id_AF),
    RcvS(~cid, ~id_AF, ~id_HN, < <A_TID, ~id_HN>, ~id_AF >) ]
  --[HN_Response(~id_HN, K_AF)]->
  [ SndS(~cid, ~id_HN, ~id_AF, < K_AF, ~exptime >) ]
```

- Application Session Establishment Response: After receiving the session key together with other information, AF would start an implicit authentication. In the specification [8], when AF receives a request from UE with its A-KID, AF would return a response without any parameters to UE. In order to let UE and AF confirm the session key, we add a key-confirmation round trip. When AF obtains the session key and the expiration time from AAnF, it would hash the session key with "AF" and send the hash value 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. We list the case for UE key confirmation.

```
rule UE_Key_Confirmation:
  [ In(f(K_AF, 'AF')),
    UE_KEY1(~SUPI, K_AUSF, K_AKMA, <A_TID, ~id_HN>, K_AF, ~id_AF),
    !AF(~id_AF) ]
  --[UE_Key_Confirmation(~SUPI, K_AF)]->
  [ Out(f(K_AF, 'UE')) ]
```

4.3 Specifying Properties

Now we introduce the properties of interest and describe them in the Tamarin prover. First, we introduce Lowe's taxonomy of authentication properties [29], which consists of 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 [1,7,8], we find that many security requirements are based on these four authentication properties, as well as confidentiality of some messages. Therefore, we will mainly characterize security of AKMA service in terms of these properties.

- Weak agreement between UE and AF is defined by the following lemma.

```
lemma weakagreement_UE_AF:
  all-traces
    "All A B t #i. Commit(A, B, <'UE', 'AF', t>) @i
      ==> (Ex t2 #j. Running(B, A, t2) @j)
        | (Ex X m #r. Reveal(X, m) @r & Honest(X) @i)"
```

  The weak agreement between AF and HN, HN and AF, UE and HN can be defined similarly.
- Non-injective agreement between UE and AF (agreeing on the target session key K_{AF}):

```
lemma Non_injective_agreement:
  all-traces
    "All A B t #i.
      Confirmation(<'AF', A>, <'UE', B>, <'UE', 'AF', <'K_AF', t>>) @i
      ==> (Ex #j. Running(B, A, <'UE', 'AF', <'K_AF', t>>) @j & j < i)
        | (Ex D m #l. Reveal(D, m) @l & Honest(D) @i)"
```

- Confidentiality of A-KID and K_{AF}. We find the leakage of A-KID will result in lots of security problems and we check its security. Meanwhile, The protocol must prevent the session key K_{AF} from being revealed, i.e., adversaries will never know the session key. We list the latter case.

```
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 describe 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 500.

5 Results and Analysis

We verify the properties listed in Sect. 4.3 using Tamarin. Except that the verification time of non-injective agreement between UE and AF is close to 30 s and the verification time of confidentiality of K_{AF} is about 15 s, the verification time of resting properties is less than 6 s. We present the verification results, and for the properties that fail to hold, analyze the counterexamples returned by Tamarin. For each counterexample, we put forward some potential attacks and propose suggestions.

5.1 Verification Results and Analysis

First of all, the executability of the protocol, the weak agreement between AF and HN, HN and AF, UE and HN, and confidentiality of K_{AF} turn out to be correct. The protocol does protect the secrecy of the session key. The confidentiality of A-KID turns out to be incorrect, which is obvious because A-KID is transferred along public channels. Next we mainly discuss the main properties that do not hold for the service.

Weak Agreement Between UE and AF. For weak agreement between UE and AF, we construct two lemmas, one with implicit authentication, and one without. The first one turns out to be correct, but the second fails. For the second case, Tamarin returns the following 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, although nothing harmful would actually happen.

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

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 mentioned as a privacy violation in [23,24]. 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.

Non-injective Agreement. Non-injective agreement property between UE and AF turns out to be incorrect, indicating that either weak-agreement between UE and AF does not hold, or UE and AF could not agree on the session key K_{AF}. Tamarin returns a counterexample: (1) UE starts a session request to the AF with A-KID_1; (2) AF forwards this message to HN and expects a session key as a response; (3) The interchange between HN and AF occurs. HN sends back to AF another A-KID_2 (actually consisting of A-TID and identifier of AF) together with identifier of HN. As a result, this A-KID_2 plays the role of K_{AF}, which could be computed by adversaries as a hash value for confirmation. Therefore, the confirmation in the protocol would execute successfully.

The reason for the above situation comes from the interchange between AF and HN and the leakage of A-KID. Considering the practical situation, the probability of HN being compromised is small and the AKMA service is assumed to trust HNs. Therefore, the interchange between HN and AF is not likely to happen. We conclude that the counterexample is unreasonable. However, to eliminate the counterexample, we make a simple fix to the rule AAnF_send_K_AF as follows: when AAnF sends the session key together with expiration time, AAnF also adds A-KID into the message.

```
rule AAnF_Send_K_AF:
  let
    session_msg = < K_AF, ~exptime, <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),
    !AF(~id_AF),
    RcvS(~cid, ~id_AF, ~id_HN, msg_In) ]
  --[_restrict(Equal(fst(msg_In), <A_TID, ~id_HN>)),
     HN_Response(~id_HN, K_AF)]->
  [ SndS(~cid, ~id_HN, ~id_AF, session_msg) ]
```

With this fix, the property is satisfied. Significantly, this fix helps AF distinguish between K_{AF} for different users. Actually in the execution of the protocol, the message would contain the session id, which is a default setting in mobile network. In a word, we prove the importance and value of the session id.

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 is harmful to users' privacy. So we suggest adding protection for the communication of A-KID. For example, pre-construct a channel for UE and AF with asymmetric encryption, or use TLS protocol. Aiming at resolving the collusion among AFs, dynamic A-KID or increasing the frequency of primary authentication are worth considering.

Moreover, in the technical specification [8], the session key K_{AF} could 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 properties and private information, even though 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 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 [8], using Tamarin verification tool. The formalization includes the formal model of the AKMA service, the security properties 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 follow the future development of the AKMA standard and update the formalization. We will also consider the privacy requirements of 5G AKMA and their formalization, e.g. the privacy caused by the linkability between AFs mentioned in this paper deserving consideration.

Acknowledgements. This work is supported in part by the NSFC under grants No. 61625206, 61972385, 62002351 and 61732001, and by the CAS Pioneer Hundred Talents Program under grant No. Y9RC585036.

References

1. 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
2. 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
3. 3GPP: TS33.102 v16.0.0 3G Security; Security architecture. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2262
4. 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
5. 3GPP: TS33.220 v17.1.0 Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA). https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2280
6. 3GPP: TS33.401 v16.3.0 3GPP System Architecture Evolution (SAE); Security architecture. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2296
7. 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
8. 3GPP: TS33.535 v17.1.0 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
9. Abadi, M., Blanchet, B., Fournet, C.: The applied pi calculus: mobile values, new names, and secure communication. J. ACM **65**(1), 1:1–1:41 (2018)
10. Armando, A., et al.: The AVISPA tool for the automated validation of internet security protocols and applications. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 281–285. Springer, Heidelberg (2005). https://doi.org/10.1007/11513988_27
11. Basin, D.A., Cremers, C., Kim, T.H., Perrig, A., Sasse, R., Szalachowski, P.: Design, analysis, and implementation of ARPKI: an attack-resilient public-key infrastructure. IEEE Trans. Dependable Secur. Comput. **15**(3), 393–408 (2018)
12. Basin, D.A., Dreier, J., Hirschi, L., Radomirovic, S., Sasse, R., Stettler, V.: A formal analysis of 5G authentication. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, 15–19 October 2018, pp. 1383–1396. ACM (2018)
13. Basin, D.A., Radomirovic, S., Schmid, L.: Modeling human errors in security protocols. In: IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, 27 June–1 July 2016, pp. 325–340. IEEE Computer Society (2016)
14. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48329-2_21
15. Blanchet, B.: An efficient cryptographic protocol verifier based on prolog rules. In: 14th IEEE Computer Security Foundations Workshop (CSFW-14 2001), Cape Breton, Nova Scotia, Canada, 11–13 June 2001, pp. 82–96. IEEE Computer Society (2001)

16. Burrows, M., Abadi, M., Needham, R.M.: A logic of authentication. ACM Trans. Comput. Syst. **8**(1), 18–36 (1990)
17. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_28
18. Clavel, M., et al.: All About Maude - A High-Performance Logical Framework. LNCS, vol. 4350. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71999-1
19. Cremers, C., Horvat, M., Hoyland, J., Scott, S., van der Merwe, T.: A comprehensive symbolic analysis of TLS 1.3. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, 30 October–03 November 2017, pp. 1773–1788. ACM (2017)
20. Dolev, D., Yao, A.C.: On the security of public key protocols. IEEE Trans. Inf. Theory **29**(2), 198–207 (1983)
21. Donovan, B., Norris, P., Lowe, G.: Analyzing a library of security protocols using Casper and FDR. In: In Workshop on Formal Methods and Security Protocols (1999)
22. Escobar, S., Meadows, C.A., Meseguer, J.: A rewriting-based inference system for the NRL protocol analyzer and its meta-logical properties. Theor. Comput. Sci. **367**(1–2), 162–202 (2006)
23. Khan, M., Ginzboorg, P., Niemi, V.: Privacy preserving AKMA in 5G. In: Mehrnezhad, M., van der Merwe, T., Hao, F. (eds.) Proceedings of the 5th ACM Workshop on Security Standardisation Research Workshop, London, UK, 11 November 2019, pp. 45–56. ACM (2019)
24. Khan, M., Ginzboorg, P., Niemi, V.: AKMA: Delegated Authentication System of 5G (2021). https://doi.org/10.13140/RG.2.2.28186.36804
25. Krawczyk, H.: HMQV: a high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_33
26. Künnemann, R., Steel, G.: YubiSecure? Formal security analysis results for the Yubikey and YubiHSM. In: Jøsang, A., Samarati, P., Petrocchi, M. (eds.) STM 2012. LNCS, vol. 7783, pp. 257–272. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38004-4_17
27. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75670-5_1
28. Lamport, L.: The temporal logic of actions. ACM Trans. Program. Lang. Syst. **16**(3), 872–923 (1994)
29. Lowe, G.: A hierarchy of authentication specification. In: 10th Computer Security Foundations Workshop (CSFW 1997), Rockport, Massachusetts, USA, 10–12 June 1997, pp. 31–44 (1997)
30. Meier, S.: Advancing automated security protocol verification. Ph.D. thesis, ETH (2013)
31. Meier, S., Schmidt, B., Cremers, C., Basin, D.: The TAMARIN prover for the symbolic analysis of security protocols. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 696–701. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_48

32. Sarr, A.P., Elbaz-Vincent, P., Bajard, J.-C.: A new security model for authenticated key agreement. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 219–234. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15317-4_15
33. Schmidt, B., Meier, S., Cremers, C.J.F., Basin, D.A.: Automated analysis of Diffie-Hellman protocols and advanced security properties. In: 25th IEEE Computer Security Foundations Symposium, CSF 2012, Cambridge, MA, USA, 25–27 June 2012, pp. 78–94 (2012)
34. Schneider, S., Holloway, R.: Using CSP for protocol analysis: the Needham-Schroeder public-key protocol. Technical report (1996)
35. Schneider, S.A.: Security properties and CSP. In: 1996 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 6–8 May 1996, pp. 174–187. IEEE Computer Society (1996)
36. Tamarin Team: Tamarin-Prover Manual: Security Protocol Analysis in the Symbolic Model. https://tamarin-prover.github.io/manual/. Accessed 7 Jan 2021