ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Formalizing and Verifying Generations of AKA Protocols

Master Thesis

Angela Rellstab

October 9, 2019

Supervisor: Dr. Ralf Sasse
Professor: Prof. Dr. David Basin

Department of Computer Science, ETH Zürich

Abstract

Mobile networks are connecting the world. More and more mobile network subscribers rely on a secure connection for their communication. The Authentication and Key Agreement (AKA) protocol variants used in mobile networks are crucial to ensure integrity and confidentiality of communication. In this thesis, we analyze different AKA protocol variants currently deployed by leveraging formal models and the Tamarin security protocol verification tool. Specifically, we first formally model individual AKA protocol variants and present the necessary requirements, i.e., minimal assumptions, to satisfy certain security properties. Second, we provide a comparison of the resulting security guarantees of individual AKA protocol variants. Finally, we formally analyze combinations of AKA protocol variants to model the co-existence of multiple mobile network generations. The analysis shows that newer AKA protocol variants improve security guarantees compared to older variants. However, the newest standard is still unable to satisfy certain security properties without extra assumptions that are not part of the actual protocol specification. When combining multiple AKA protocol variants, as happens in the real world, stronger assumptions must be made to satisfy the same security properties.

Contents

Chapter 1

Introduction

Mobile networks have evolved considerable over the years. Old standards are replaced by newer generations. Currently, the fifth generation (5G) is being deployed all over the world. During this transition to 5G, 3G as well as 4G still exist alongside 5G. On the other hand, the older second generation 2G will be shut down in near future. For instance, in Switzerland, Swisscom announced to end the support for 2G by the end of 2020, whereas 3G will be supported at least until 2024 [1]. The 3rd Generation Partnership Project (3GPP) [4] group is the standardization organization for mobile network generations. Subscribers rely on these standards to communicate securely.

A mobile network includes three parties, the device used by the subscriber, the home network, and the serving network. Each device is subscribed to one home network. The device connects to a serving network that may belong to the home network of the device. A critical building block of mobile networking is the Authentication and Key Agreement (AKA) protocol that is responsible to enable secure communication. The protocol is used to mutually authenticate device and home network, and to establish a session key shared between the device and serving network. Because of the co-existence of multiple mobile network generations for the next years, not only the security properties of the new standard 5G, but also the security properties of 3G and 4G are of interest. Additionally, security properties of an AKA variant should not be violated when running concurrently to other AKA protocol variants.

In general, security protocols are difficult to get right. Small mistakes in a protocol can jeopardize its security guarantees. Moreover, to reason about the correctness of security protocols is hard. Mistakes in a proof of a protocol may lead to incorrect results, and thus missed attacks. We leverage formal methods and Tamarin to analyze the protocols in detail.

Related Work. Because authentication and key establishment is an essential piece, the AKA protocol variants have been studied before. We present an high-level overview of previous work here, and revisit it in more detail in Chapter 6. 3GPP analyzes AKA of 3G manually using enhanced BAN logic [5]. An automated analysis of 3G AKA, using ProVerif [15], is presented in [12]. ProVerif is another security verification tool to automatically reason about security protocols. AKA of 4G is analyzed in [19], also using ProVerif. The previous analyses of the AKA protocols of 3G and 4G lack accuracy because they use at least two of the following three simplifications. First, The AKA protocol is analyzed as a stateless protocol abstracting the sequence numbers to nonces. Thus, the resynchronization mechanism is not taken into account since it is not possible to get out of sync. Second, the models do not include the exclusive-or (XOR) operator. Finally, the home and serving networks are combined into one single party, which is not necessarily the case. Due to the oversimplified protocol models, flaws and attacks can easily be missed.

The fifth generation of mobile networks introduces two AKA options: 5G AKA and 5G EAP AKA'. 5G AKA has been analyzed more thoroughly than other AKA variants. A detailed analysis of 5G AKA tackling the above mentioned issues in the analysis of 3G and 4G is presented by Basin et al. [14]. Basin et al. introduce a stateful model that includes sequence numbers. Thus, also the resynchronization mechanism is modeled. Additionally, they make use of the XOR operator which is supported in Tamarin. 5G divides the home network in two sub-parties, which is not taken into account by Basin et al. This shortcoming is tackled by Cremers and Dehnel-Wild [17]. Cremers and Dehnel-Wild model 5G AKA using four parties, dividing the home network in two separate parties. However, Cremers and Dehnel-Wild do not model 5G AKA as stateful protocol. Thus, resynchronization is not modeled. Additionally, they do not consider the XOR operator.

Contributions. The focus of this thesis lies on the comparison of AKA protocol variants used in different mobile network generations. We are interested in a detailed model and security analysis of AKA variants currently in use. We present the following main contributions:

- We present a generic flow of the AKA protocol variants currently used in 3G, 4G, and 5G.
- We formally model AKA protocol variants used in 3G, 4G, and 5G: 3G AKA, 4G EPS AKA, 5G AKA, and 5G EAP AKA'. The individual AKA protocol models build on the same generic model that includes sequence numbers as state, the resynchronization mechanism, and the XOR operator. Additionally, the serving and home network are mod-

eled as separate parties. To our knowledge, we are the first to introduce such a detailed model for 3G AKA and 4G EPS AKA.

- We introduce the first formal models that consider multiple AKA protocol variants at once. This allows us to take the co-existence of multiple mobile network generations into account.
- We introduce proof strategies to guide the TAMARIN prover during the proof search. All proofs and attacks run automatically with the help of an oracle.
- We analyze each individual AKA protocol variant and compare the resulting security guarantees between AKA protocol variants. The individual variants are based on the same generic flow, which enables us to consistently compare them.
- We analyze the combined AKA protocol models. We compare security guarantees of an AKA variant modeled separately with combined models.
- The analysis shows that newer AKA protocol variants improve security guarantees compared to older AKA variants. When combining multiple AKA variants, as happens in the real world, stronger assumptions must be made to satisfy the same security properties.

Outline. The thesis is structured as follows. The second chapter introduces the analyzed AKA protocol variants. The third chapter is dedicated to introduce TAMARIN. The fourth chapter discusses TAMARIN modeling decisions for the AKA variants. The fifth chapter presents the security analysis of the individual and combined models. The sixth chapter presents multiple comparisons of our results with previous work. Finally, chapter seven concludes the thesis.

Chapter 2

Authentication and Key Agreement Protocol

This chapter introduces four Authentication and Key Agreement (AKA) protocol variants. AKA is used in mobile networking to authenticate users and networks, and to establish a secure channel between them. The 3rd Generation Partnership Project (3GPP) is in charge of standardizing mobile networks, including the specification of the AKA protocol variants. Across mobile network generations different AKA variants are used. 3G and 4G use 3G AKA [7] and 4G EPS AKA [11], respectively. 5G on the other hand introduces two AKA options, namely 5G AKA [10] and 5G EAP AKA' [13]. Multiple parties are involved in an AKA protocol run:

User Equipment (UE) describes the device used by the subscriber, e.g., a mobile phone, including its USIM. The USIM stores the identity of the UE as well as a long-term secret K which is shared between the UE and its home network.

Home Network (HN) Each UE is subscribed to one HN. The HN knows the identifying information and the long-term secret K for each subscribed UE, and is in charge of authenticating them. Additionally, the HN operates a serving network.

Serving Network (SN) The UE uses an insecure channel to connect to the SN that may belong to the HN. Each SN is connected to the HNs via a secure channel, i.e., confidential and authentic. In case the UE is unable to reach the SN operated by its HN, the SN and HN are two different entities. An instance of this is roaming.

The involved parties including the connecting channels are shown in Figure 2.1. In contrast to older generations, 5G splits the HN into multiple subsystems. The idea is to separate sensitive data from the communication with a SN. However, in order to consistently compare the two 5G authentication

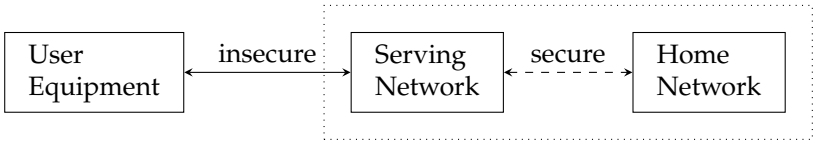

Figure 2.1: Overview of parties involved in the AKA protocol variants. The channel between the user equipment is insecure, whereas the channel between the serving and home network is secure.

protocols with the older mobile network generations, we combine the subsystems into one party. We introduce four different AKA protocol variants in this chapter. In order to get an overview of AKA, we first introduce the generic protocol flow of AKA in Section 2.1. The Overview is followed by the actual protocol variants in Section 2.2.

2.1 General AKA Flow

This section introduces the general flow of an AKA protocol. The goal of AKA is to achieve mutual authentication between the user equipment and the serving or home network. The protocol is summarized in Figure 2.2. AKA is a challenge-response authentication protocol that relies on sequence numbers for replay protection. The UE and HN have their own sequence number SQN that must be synchronized to provide freshness for challenges. The HN shares a long-term symmetric key K with the UE that is used to generate and verify message authentication codes (MAC). Additionally, K is used to compute the response for a challenge.

We divide the AKA protocol into three phases. In the initial phase, the UE starts the authentication process. In the second phase, a challenge-response mechanism is used to authenticate the parties to each other and to agree on a session key. In the last phase, failure handling is run in case the challenge contains an invalid MAC or the sequence numbers are out of sync. The three phases are discussed in more detail below. Some message details differ between AKA variants and are omitted.

1. The UE initiates the protocol by sending an attach request to the SN, which in turn sends an authentication initiation request (AIR) to the HN. The attach request indicates to the HN which UE wants to start authentication and is part of the AIR message.

2. The HN generates an authentication vector (AV) and sends the AV back to the SN. The AV contains a challenge and expected response that the SN uses in order to authenticate the UE. In case the UE receives a fresh challenge which contains a valid MAC, the UE first updates its own sequence number and derives the master key (MK). Then, the UE computes the response and sends it to the SN. After the

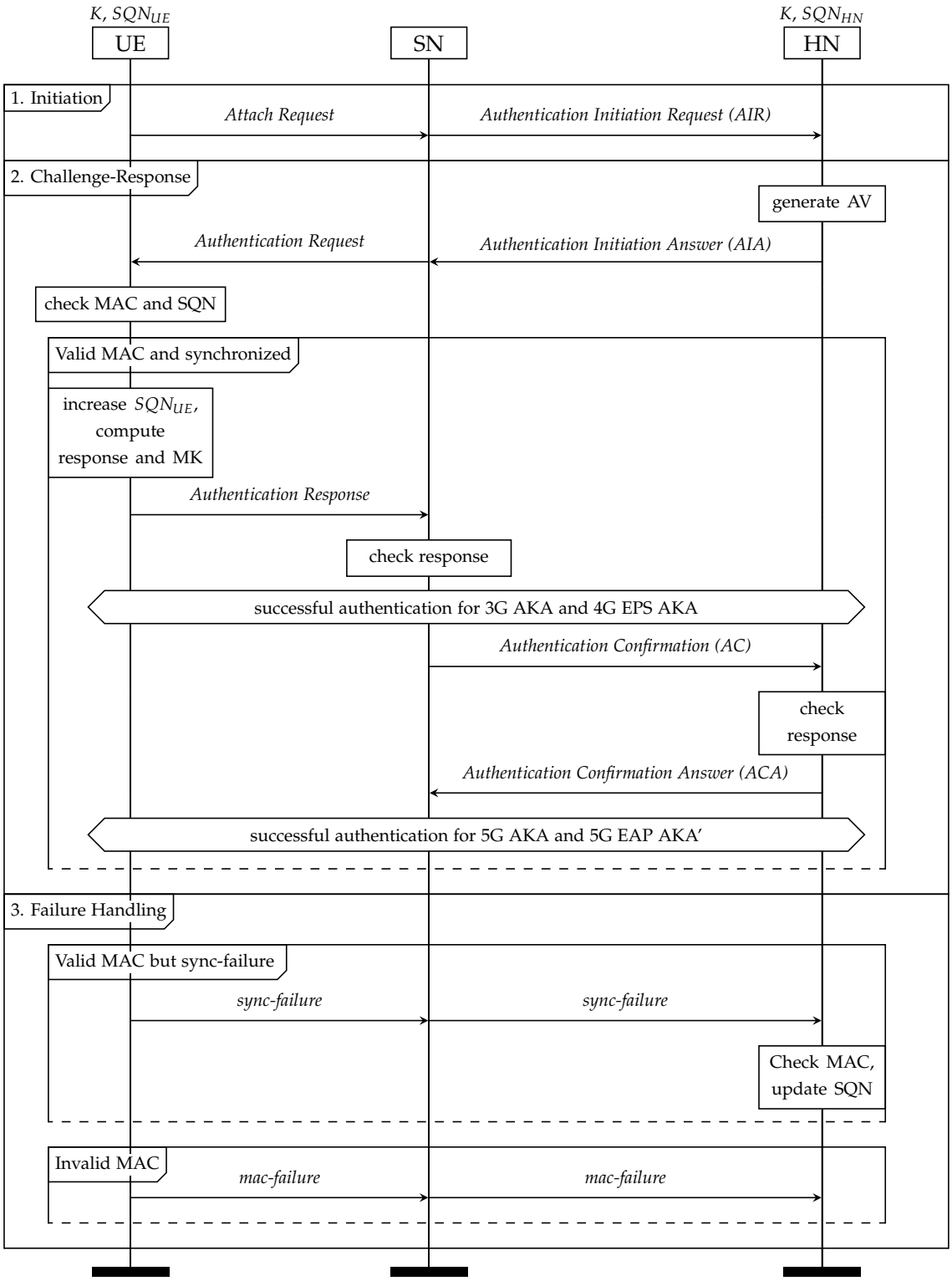

Figure 2.2: General flow overview of the AKA protocol generations. Note that 3G AKA and 4G EPS AKA finish earlier than 5G AKA and 5G EAP AKA'.

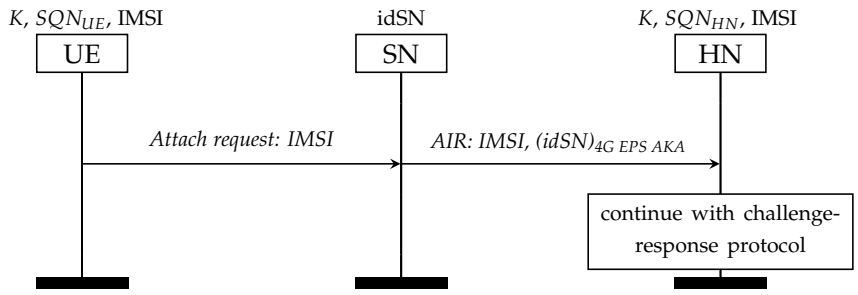

Figure 2.3: Authentication initiation of 3G AKA and 4G EPS AKA. The attach request contains the identity of the UE (IMSI). The SN identity (idSN) is only sent in 4G EPS AKA.

SN successfully verifies the response by comparing it to the expected response, authentication is complete in 3G AKA and 4G EPS AKA. As a result, the UE and SN, as well as the UE and HN have mutually authenticated each other. In addition, the UE shares a session key with the SN. In 5G AKA and 5G EAP AKA' authentication is complete only after the HN successfully verifies the response as well.

3. In case the UE receives a challenge with an invalid MAC, it sends a MAC failure message to the SN, which in turn forwards it to the HN. In case the UE receives an authentication request containing an old challenge, i.e., with a correct MAC but the SQN is unsynchronized, the UE sends an out of sync message to the SN. The out of sync failure message includes enough information for the HN to update its sequence number.

The three protocol phases of AKA are described in more detail for each variant in the following section.

2.2 AKA Generations

This section introduces 3G AKA, 4G EPS AKA, 5G AKA as well as 5G EAP AKA'. We divide the protocols into three phases: Section 2.2.1 covers initiation, Section 2.2.2 introduces the challenge-response, and Section 2.2.3 presents failure handling mechanisms. Each of the three sections highlights notable variations for each AKA variant.

2.2.1 Initiation

Initiation works similarly for all AKA variants. The UE sends an attach request to the SN, which contains a unique UE identifier. The UE identity is called international mobile subscriber identity (IMSI) in 3G AKA and 4G EPS AKA, and subscriber permanent identifier (SUPI) in 5G AKA and 5G

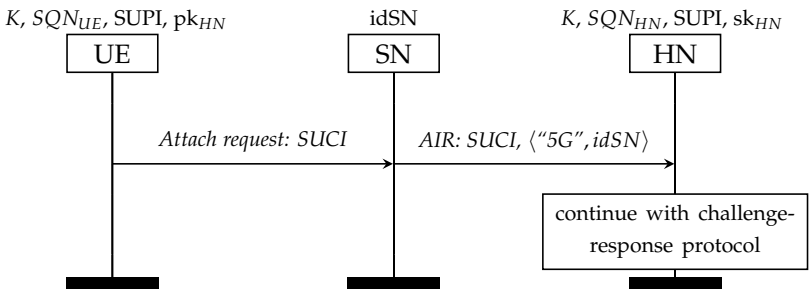

Figure 2.4: Authentication initiation of 5G AKA and 5G EAP AKA'. The UE sends a concealed UE identity to the SN. Note that the SN identity is tagged with "5G".

EAP AKA'. Although the name of the UE identity differs between generations, the identity remains the same. IMSI and SUPI contain a subscriber identifier, and a home network identifier. The home network identifier is a tuple itself, containing a mobile country code and a mobile network code.

Figure 2.3 shows initiation of 3G AKA and 4G EPS AKA. The UE sends an attach request to the SN containing IMSI. The SN forwards the authentication initiation request, containing IMSI, to the HN. In case of 4G EPS AKA, the SN additionally includes the SN identity in the authentication initiation request to the HN. Figure 2.4 shows initiation of 5G AKA and 5G EAP AKA', which is the same for both 5G variants. The UE sends a concealed UE identifier over the network. The subscription concealed identifier (SUCI) randomly conceals SUPI using the public key of the HN (pk_{HN}) as follows:

$$SUCI = \langle \{SUPI, R\}_{pk_{HN}}, idHN \rangle,$$

where R denotes a freshly generated random number, and $idHN$ denotes the identity of the HN. Additionally, the SN identity is tagged with "5G".

2.2.2 Challenge-Response

After a successful challenge-response phase, the involved parties agree on a master key MK. The MK can then be used to establish a secure channel between the UE and SN.

Functions used in this section are defined in the technical specifications [7, 8] as follows. $f1$ is a message authentication function. $f2$ is a (possibly truncated) message authentication function. $f3$, $f4$, $f5$ are key derivation functions, (or $f5 = 0$). $HMAC\text{-}SHA\text{-}256(key, term)$ is used as key derivation function $kdf(key, term)$.

We introduce 3G AKA in more detail and discuss the differences and improvements for the newer generations. For instance, the master key derivation gets more involved for each generation.

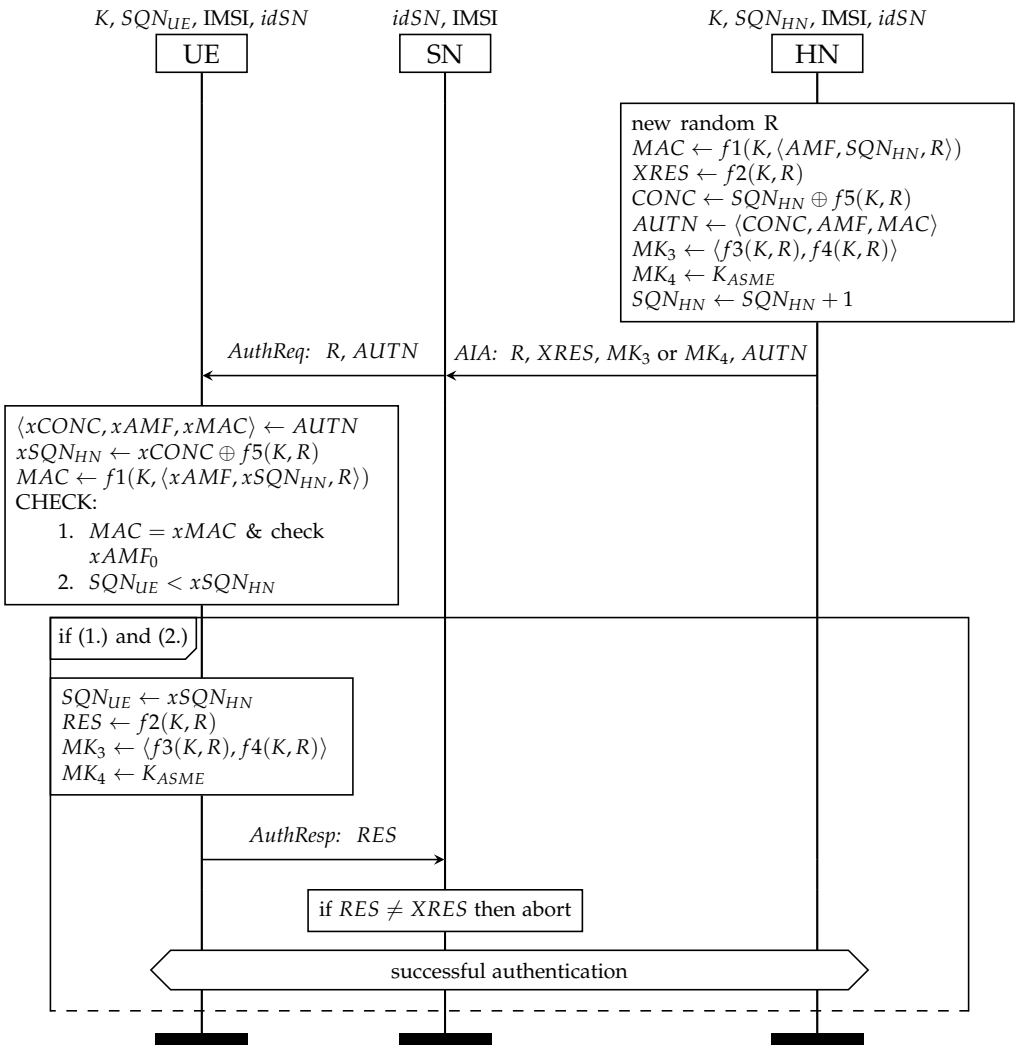

Figure 2.5: 3G AKA and 4G EPS AKA challenge-response phase continuing initiation shown in Figure 2.3. Note that MK_3 is the master key derived for 3G AKA, whereas MK_4 is derived for 4G EPS AKA. Only the MK of the current AKA variant is derived. See Section 2.2.2 for more detail on the MK_4 derivation. Failing checks (1.) or (2.) at the UE are discussed in Section 2.2.3.

3G AKA

Figure 2.5 shows 3G AKA (and 4G EPS AKA) continuing after the authentication initiation. The HN generates an authentication vector AV:

$$\text{AV} \leftarrow \langle R, XRES, MK_3, AUTN \rangle.$$

The AV contains a random nonce R, the expected challenge response $XRES$, the master key MK_3 and an authentication token $AUTN$. The terms are

derived using the shared long-term secret K and R as follows:

$$XRES \leftarrow f2(K, R),$$

$$MK_3 \leftarrow \langle f3(K, R), f4(K, R)\rangle, \text{ and}$$
$$AUTN \leftarrow \langle CONC, AMF, MAC\rangle.$$

$AUTN$ is a triple. First, $AUTN$ contains the concealed sequence number $CONC$ of the HN derived by xoring SQN_{HN} with the anonymity key $f5(K, R)$, i.e., $CONC \leftarrow SQN_{HN} \oplus f5(K, R)$. Second, the authentication management field AMF is used to signal further parameters. The first bit of the AMF is called AMF separation bit AMF_0, which is always set to 0 in 3G AKA. We will not discuss AMF in more detail here, because its use is not fully standardized [6]. Finally, MAC defines a message authentication code over AMF, SQN_{HN}, and R:

$$MAC \leftarrow f1(K, \langle AMF, SQN_{HN}, R\rangle).$$

The HN sends the AV to the SN. The AV contains everything that the SN needs in order to authenticate the UE and to communicate with the UE after successful authentication. Thus, the HN is not included in the further authentication process, except if a MAC or synchronization failure happens at the UE (see Section 2.2.3). The SN uses the random number R, $AUTN$ and $XRES$ to authenticate the UE. R and $AUTN$ pose the challenge for the UE.

The UE uses $AUTN$ to check that the challenge is authentic and fresh. As mentioned above, the $AUTN$ is a triple:

$$\langle xCONC, xAMF, xMAC\rangle \leftarrow AUTN.$$

The UE computes MAC as follows:

$$MAC \leftarrow f1(K, \langle xAMF, xSQN_{HN}, R\rangle), \text{ where}$$

$xAMF$ denotes the received authentication management field included in $AUTN$ and

$$xSQN_{HN} \leftarrow xCONC \oplus f5(K, R)$$

denotes the unmasked sequence number of the HN. Note that

$$(a \oplus b) \oplus b = a.$$

In order to continue with authentication, the received MAC ($xMAC$) and computed MAC (MAC) have to be equal, i.e., $MAC = xMAC$. Further, the AMF separation bit has to be set to 0. Additionally, the sequence number at the UE has to be smaller than the received sequence number from the HN,

i.e., $SQN_{UE} < SQN_{HN}$. Otherwise, the UE considers the challenge not to be fresh. If any of the mentioned checks fail, the UE continues as described in Section 2.2.3. In case all checks are successful, the UE first updates its own sequence number

$$SQN_{UE} \leftarrow xSQN_{HN},$$

and derives MK_3 in the same way as the HN. Further, the UE computes the response for the challenge RES, and sends it to the SN.

$$RES \leftarrow f2(K, R)$$

The SN is able to verify the response of the UE by comparing it with $XRES$. In case $RES = XRES$, the SN considers authentication to be a success. The SN uses MK_3, received from the HN in the AV, for further communication with the UE.

4G EPS AKA

4G EPS AKA is summarized alongside 3G AKA in Figure 2.5. The protocol is very similar to 3G AKA with two adjustments. First, the AMF separation bit is always set to 1. Second, 4G EPS AKA uses a more involved derivation scheme for the master key $MK_4 \leftarrow K_{ASME}$. The derivation does not only include the long-term secret K and randomness R, but also the identity of the SN $idSN$ and SQN_{HN}. K_{ASME} is derived as follows

$$K_{ASME} \leftarrow kdf(\langle CK, IK\rangle, \langle idSN, CONC\rangle), \text{ where}$$

$$CK \leftarrow f3(K, R),$$

$$IK \leftarrow f4(K, R), \text{ and}$$

$$CONC \leftarrow SQN_{HN} \oplus f5(K, R).$$

Note that the key $\langle CK, IK\rangle$ used in the key derivation function is the same as MK_3.

5G

5G includes two different protocol variants to use for authentication and key establishment. After initiation, the HN chooses either 5G AKA or 5G EAP AKA' to continue with the challenge-response phase. The flow of 5G AKA and 5G EAP AKA' are still similar to the previous generations (see Figure 2.6 and 2.7). However, multiple improvements happen in 5G. First, the HN is actively included in the protocol even after providing the AV. As a result, the HN knows if authentication is successful. Second, the SN receives the unconcealed UE identity SUPI only in the last message from the HN. This message also includes the master key. Previous generations attach the master key to the AV. The goal is that a SN is unable to fake visits from

a UE. An example scenario where that would be useful is roaming. A SN is unable to bill a UE that did not actively connect to the SN. Additionally, all SN identities are tagged with "5G". Like in 4G EPS AKA, the AMF separation bit is set to 1. Following we present the improvements that only concern the specific AKA protocol variant of 5G.

5G AKA The HN does not include the expected response in the AV. Instead, the HN includes the hash of the expected response. The goal is to ensure that the UE is indeed included in the protocol run, since the SN is unable to compute the response. The SN is still able to check the validity of the response received from the UE by checking its hash.

The response RES^* and the expected response $XRES^*$ are computed in a more involved manner than in 4G EPS AKA:

$$RES^* \leftarrow Challenge(K, idSN, R),$$

where the *Challenge* function is defined as follows:

$$Challenge(K, idSN, R) := kdf(\langle CK, IK\rangle, payload).$$

The key $\langle CK, IK\rangle$ used in the key derivation function is the same as MK_3, which is also used in the derivation of the master key MK_5 below. The payload is a triple containing the SN identity, R and the response RES that is also used in previous generations.

$$payload \leftarrow \langle \text{"5G"} : idSN, R, RES\rangle, \text{ where}$$

$$RES \leftarrow f2(K, R).$$

As mentioned above, the HN sends the hash of the expected response, called $HXRES^*$, to the SN which is computed as follows:

$$HXRES^* \leftarrow SHA256(\langle R, XRES^*\rangle).$$

The SN computes the hash of the response received from the UE. The computed and received hash have to be equal in order to continue the protocol, i.e.,

$$SHA256(\langle R, RES^*\rangle) \stackrel{?}{=} HXRES^*.$$

In case the check is successful, the SN sends the response to the HN. The HN also verifies the response, i.e.,

$$RES^* \stackrel{?}{=} XRES^*.$$

After the validation of the response, the HN considers the authentication successful. The HN sends the master key MK_5 to the SN. The derivation of $MK_5 \leftarrow K_{SEAF}$ is defined as follows:

$$K_{SEAF} \leftarrow kdf(K_{AUSF}, \text{"5G"} : idSN), \text{ where}$$

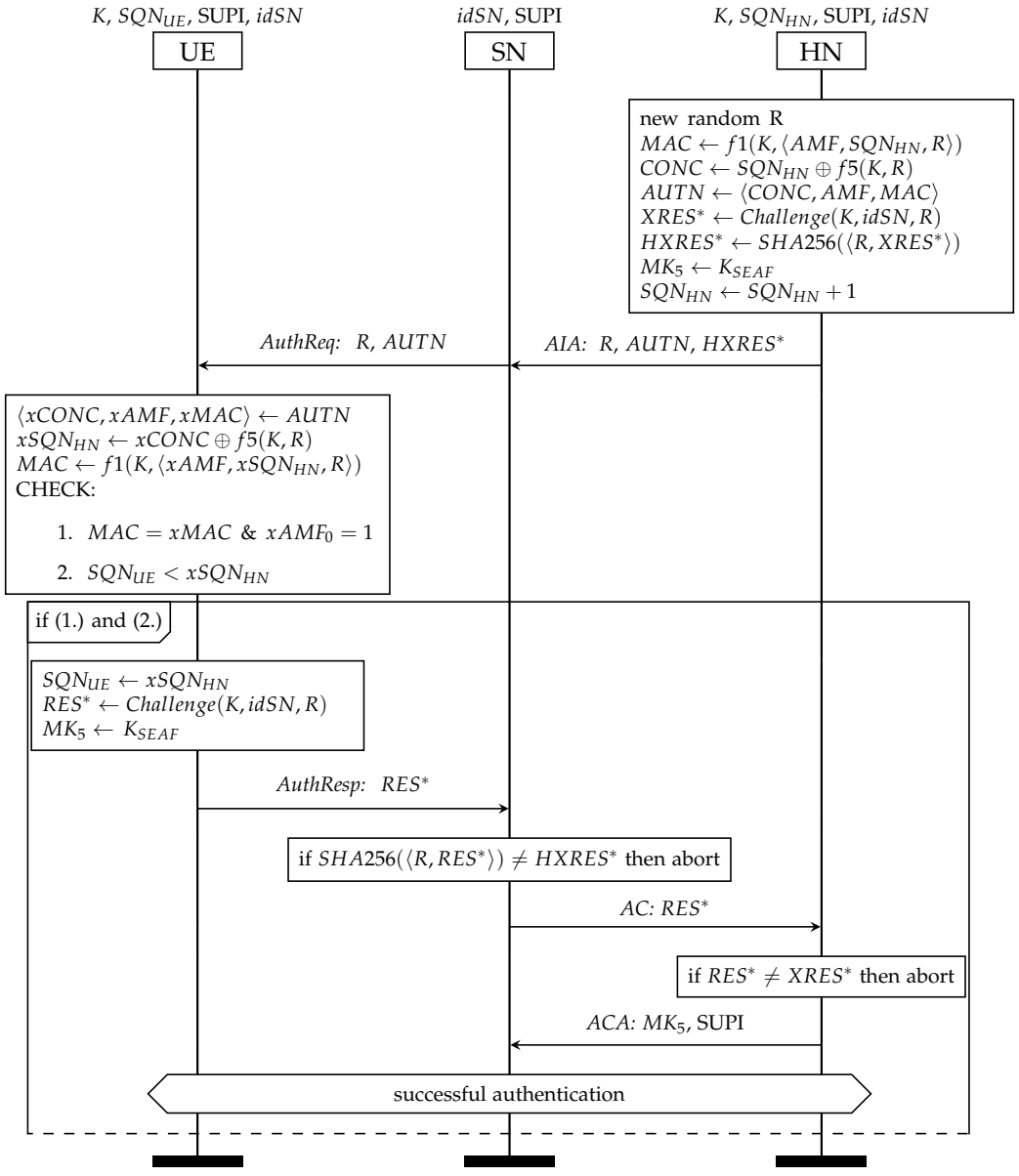

Figure 2.6: 5G AKA challenge-response phase continuing initiation shown in Figure 2.4. See Section 2.2.2 for more details on the *Challenge* function and derivation of MK_5. Failing checks (1.) or (2.) at the UE are discussed in Section 2.2.3.

$$K_{AUSF} \leftarrow kdf(\langle CK, IK\rangle, \langle \text{“5G”} : idSN, CONC\rangle).$$

The intermediate key, K_{AUSF}, is derived in the same way as K_{ASME} (MK_4) in 4G EPS AKA, except that the SN identity is tagged with “5G”. $CONC$ denotes the concealed sequence number defined for previous generations.

5G EAP AKA’ 5G EAP AKA′ is based on the extensible authentication protocol (EAP). During authentication the SN is used as a pass-through only, i.e., the SN forwards messages to the UE or to the HN until authentication is complete (see Figure 2.7). In contrast to the other AKA variants, the SN does not check the validity of the response received from the UE. If the protocol run is successful, the SN receives SUPI and $MK_{5'}$ in the last message from the HN. The UE and HN derive $MK_{5'} \leftarrow K_{SEAF}$ as follows:

$$K_{SEAF} \leftarrow kdf(K_{AUSF}, \text{“5G”} : idSN),$$

which is the same for 5G AKA. 5G EAP AKA′ differs to 5G AKA in the derivation of K_{AUSF}:

$$K_{AUSF} \leftarrow MasterKey\big[1152..1663\big].$$

K_{AUSF} takes bit range starting at 1152 to 1663 from $MasterKey$. Be aware that $MasterKey$ is not the same as $MK_{5'}$. $MasterKey$ not only includes K_{AUSF} but also K_{AUTH}. K_{AUTH} is used for the MAC computation, which we will introduce in the next paragraph. $MasterKey$ computation works as follows:

$$MasterKey \leftarrow PRF'(\langle IK', CK'\rangle, \langle \text{”EAP-AKA’”}, peer\_identity\rangle),$$

where PRF' is a pseudo random function defined in [13, 10]. The pseudo random function uses SUPI as peer_identity. The key, $\langle IK', CK'\rangle$, is derived as follows:

$$CK' \leftarrow kdf(K, \langle R, SQN, \text{“5G”} : idSN\rangle)\big[0..127\big], \text{ and}$$

$$IK' \leftarrow kdf(K, \langle R, SQN, \text{“5G”} : idSN\rangle)\big[128..255\big],$$

where $\big[0..127\big]$ denote the 128 most significant bits.

The UE and HN add an additional MAC to each message called AT_MAC that contains the whole message. Hence, no party in between the HN and UE is able to alter or inject messages. The computation of AT_MAC uses an authentication key K_{AUTH} which is part of the $MasterKey$ defined above:

$$K_{AUTH} \leftarrow MasterKey\big[128..383\big].$$

AT_MAC for a message m is computed as follows:

$$AT\_MAC_m \leftarrow mac(K_{AUTH}, \langle m, \text{message-tag}\rangle),$$

where mac denotes a MAC function defined in [13]. AT_MAC of the challenge and response is tagged with ”1” and ”0”, respectively, in order to distinguish the messages.

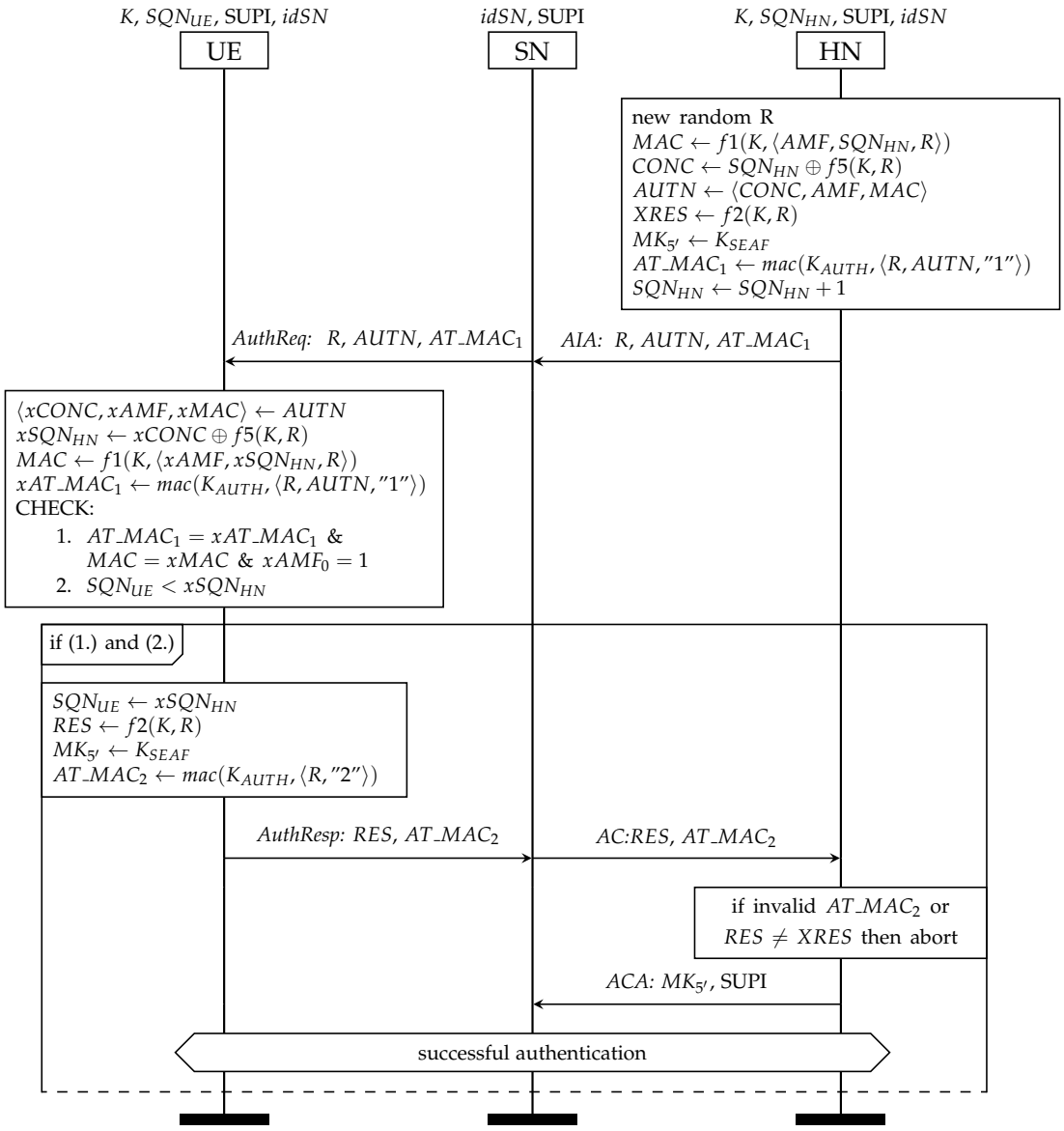

Figure 2.7: 5G EAP AKA' challenge-response phase continuing initiation shown in Figure 2.4. See Section 2.2.2 for more details on derivation of $MK_{5'}$ and $AT\_MAC_{1/2}$. Failing checks (1.) or (2.) at the UE are discussed in Section 2.2.3.

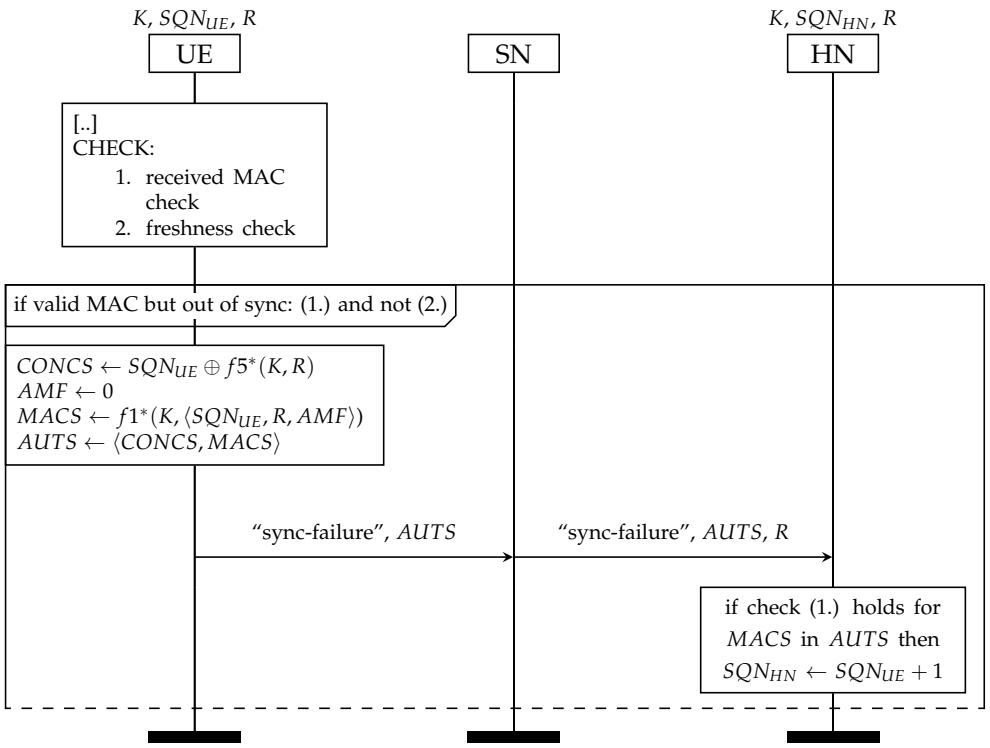

Figure 2.8: Resynchronization mechanism used if the UE receives a challenge with a valid MAC but with an old sequence number. Check (1.) and (2.) are introduced for each AKA variant in Section 2.2.2.

2.2.3 Invalid Challenge

In case the UE receives an authentication request which either contains an invalid MAC, or an invalid sequence number, the challenge is flawed. The MAC and sequence number validation is presented for each variant in Section 2.2.2. We first introduce the resynchronization mechanism and afterwards MAC failure.

Resynchronization

The resynchronization mechanism is used if the challenge contains a valid MAC but the sequence numbers of the UE and HN are out of sync. Figure 2.8 shows the resynchronization procedure. The procedure is the same for all AKA variants. It is used when the UE receives an authentication token that contains a valid MAC but an invalid sequence number, i.e., $SQN_{UE} \geq SQN_{HN}$. In order to synchronize the sequence number of the UE and HN, the UE creates an authentication failure message with synchronization fail-

ure $AUTS$:

$$AUTS \leftarrow \langle CONCS, MACS \rangle.$$

$AUTS$ is a tuple which contains the concealed sequence number of the UE as well as a MAC. The concealment of SQN_{UE} works as follows:

$$CONCS \leftarrow SQN_{UE} \oplus f5^*(K, R).$$

$MACS$ is a MAC over SQN_{UE}, R and AMF_{dummy}.

$$MACS \leftarrow f1^*(K, \langle SQN_{UE}, R, AMF_{dummy} \rangle)$$

The AMF is set to a dummy value of all zeros, such that it does not have to be retransmitted again. The SN forwards $AUTS$ to the HN which checks that $AUTS$ contains a valid MAC. The validation works in the same way as check (1.) at the UE. In case $MACS$ is valid, the HN updates SQN_{HN} to $SQN_{UE} + 1$. The resynchronization mechanism replaces functions $f1$ and $f5$ with $f1^*$ and $f5^*$, respectively. $f1^*$ is a message authentication function, whereas $f5^*$ is a key derivation function to conceal the sequence number. Neither from $f1^*$ nor from $f5^*$ can valuable information be inferred about $f1$-$f5$.

MAC Failure

In case the UE receives a challenge with an invalid MAC, the UE sends a MAC failure message to the SN, which in turn forwards the failure message to the HN. The HN decides if the failure is indeed justified. It is the task of the HN to decide whether to reattempt authentication or to cancel. We do not take a closer look at MAC failures and the following procedures. For our purpose it suffices to assume that after the occurrence of such a failure the UE eventually starts another authentication initiation attempt.

Chapter 3

Tamarin Prover

Security protocols are critical to get right. However, it is hard to reason about such protocols, because of a huge search space even for small protocols. As a matter of fact, correctness of security protocols in case of unbounded number of messages, nonces, and sessions is undecidable. We leverage Tamarin to formally prove properties of the AKA protocol variants introduced in the previous chapter. Tamarin is a symbolic verification tool, which means that it argues over terms and not over bit strings. Tamarin is proven sound and complete, but may not terminate because of the undecidability of the problem. In case Tamarin terminates, it either returns a proof, or a counter example.

This chapter introduces the Tamarin prover defined in [22, 21]. Throughout the chapter we will refer to the small toy protocol defined in Figure 3.1 which is modeled in Tamarin in Listing 3.1. The toy protocol includes two roles X and Y. X encrypts a nonce n_X using the shared symmetric key k and sends it to Y.

We first present how protocols are specified in Tamarin in Section 3.1. We introduce the threat model in Section 3.2. Protocol execution is covered in Section 3.3, which is followed by an introduction of the representation of security properties in Section 3.4. Finally, we introduce the proof mechanism of Tamarin in Section 3.5.

3.1 Protocol Model

A protocol model in Tamarin defines rules that describe the actions of protocol participants. First, initialization rules create the setting in which the protocol participants communicate. That includes generation of keys, identities and other prearrangements to enable communication. Second, protocol rules describe the communication of participants, i.e., sending and receiv-

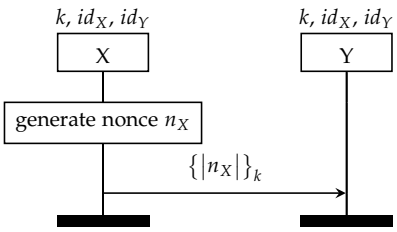

Figure 3.1: Toy protocol. Role *X* generates a nonce n_X, encrypts it with the shared symmetric key *k*, and sends the cyphertext to *Y*. $\{|n_X|\}_k$ denotes the symmetric encryption of n_X using key *k*.

Listing 3.1: Tamarin Model of the Toy Protocol introduced in Figure 3.1.

```
1  theory example
2  begin
3
4  builtins:
5  symmetric-encryption
6
7  // Initialization rules
8  rule init_ltk:
9   [Fr(~k),
10   Fr(~id_X),
11   Fr(~id_Y)]
12  --[Create(~id_X),
13     Create(~id_Y)]->
14  [!Ltk(~id_X, ~id_Y, ~k)]
15
16 // Protocol rules
17 rule x_send_nonce:
18  [Fr(~n),
19   !Ltk(~id_X, ~id_Y, ~k)]
20  --[Running(~id_X, ~id_Y, <'Y','X',~n>)]->
21  [St_1_X(~id_X, ~k, ~n),
22   Out( senc(~n, ~k))]
23
24 rule y_receive:
25  [!Ltk(~id_X, ~id_Y, ~k),
26   In(senc(n, ~k)) ]
27  --[Commit(~id_Y, ~id_X, <'Y','X',n>),
28     Secret(~id_Y, n),
29     Honest(~id_X),
30     Honest(~id_Y)]->
31  [St_1_Y(~id_Y, ~k, n)]
32
33 end
```

ing of messages for each participant. To model a protocol TAMARIN distinguishes between *terms*, *multiset rewrite rules*, and *facts*.

Term A message sent between roles is defined as term. A term is either a constant, a variable, or a function over terms. The n-ary function f with terms $t_1, \ldots, t_n$ is denoted by $f(t_1, t_2, \ldots, t_n)$. TAMARIN provides multiple built-in functions, like asymmetric encryption, hash, xor, and more. Additional functions can be provided by the user. The toy model uses symmetric encryption which is specified in line 5. The built-in symmetric encryption defines functions for encryption *senc* and decryption *sdec*. Additionally, for a message m and key k the following equation holds

$$m = sdec(senc(m, k), k) \ .$$

The toy protocol uses multiple terms, for instance both n_X, and $\{|n_X|\}_k$ are terms.

Multiset Rewrite Rule A multiset rewrite rule denotes a transition from a state to the next state. It is of the form

$$\texttt{rule}\text{ example: } l \texttt{ --}[a]\texttt{-> } r \ ,$$

where `rule` is a keyword that denotes the start of the rule called *example*. The rule name has to be unique in a protocol. l, r, and a are multisets of facts. Facts are introduced below. A multiset denotes a set that allows elements to occur multiple times. l and r are multisets of state facts, describing the progress of a protocol participant. a contains action facts, which describe a protocol trace. We introduce protocol traces in Section 3.4. The multiset rewrite rule consumes the facts defined in l from the current state, and produces all facts defined in r. Transitions and protocol executions are covered more thoroughly in Section 3.3.

Fact A fact is described with a fact symbol and terms. For instance for the fact `St_1_X(~id_X, ~k, ~n)` in Listing 3.1, `St_1_X` is the fact symbol which describes that agent X is at its internal state 1. `~id_X`, `~k`, and `~n` are the terms. *In*, *Out*, and *Fr* are reserved fact symbols. The *In* and *Out* facts are used to receive and send messages, respectively, see Section 3.2 for more details. *Fr* generates a fresh random value. To model a random value we use $\sim d$, which describes that variable d contains a random value. The toy protocol uses random values for the nonce, the symmetric key k, and the two identities.

TAMARIN distinguishes between linear and persistent facts. Linear facts are consumed by a rule, e.g., no two rules can use the same linear fact on the left-hand side of the multiset rewrite rule. On the other

Listing 3.2: Secure Send and Receive

```
1 rule send_secure:
2  [SndS(A,B,m)]
3  -->
4  [Sec(A,B,m)]
5
6 rule receive_secure:
7  [Sec(A,B,m)]
8  -->
9  [RcvS(A,B,m)]
```

hand, persistent facts are allowed to be consumed multiple times by rules, e.g., it is possible that the same fact occurs multiple times on the left-hand side of a rule. Tamarin uses an exclamation mark (!) to label persistent facts. The toy protocol uses the *Ltk* fact as persistent fact. Thus, it can be consumed multiple times. This is necessary, because both send and receive rule (line 17 and 24) use *Ltk* on the left-hand side.

The toy protocol has three rules (see lines 8-31). The first rule, `init_ltk`, starting on line 8 initializes a long-term key $\sim k$ and two identities $\sim id\_X$ and $\sim id\_Y$. The rule creates random values via the fresh fact. The fact $!Ltk$ binds $\sim id\_X$, $\sim id\_Y$, and $\sim k$ together. In the second rule starting on line 17, X generates the nonce and sends it encrypted to the network. The protocol rule starting on line 24 models Y receiving the encrypted nonce.

3.2 Threat Model

Tamarin uses as default a Dolev-Yao adversary [18]. The adversary controls the network, e.g., the adversary is able to read, intercept, and send messages. Additionally, the adversary can apply functions to terms in his knowledge and generate fresh values. Tamarin models the capabilities of the adversary using multiset rewrite rules. The fact $K(t)$ describes that the adversary knows term t. The *Out(m)* fact, in line 22 of the toy protocol, describes that the message is sent to the network, i.e., the adversary knowledge contains m. The *In(m)* fact on the other hand (line 26), describes that the participant receives message m from the network. As a consequence, the adversary knows all messages that are sent and received via the *Out* and *In* fact, respectively.

In case a protocol relies on authentic or confidential channels between protocol participants, they have to be modeled explicitly. We introduce the rules for a secure channel following the Tamarin manual [2] in Listing 3.2. A secure channel is both authentic and confidential. Instead of *Out(m)*, agent A uses *SndS(A, B, m)* to securely send message m to agent B. On the other hand, to securely receive the message the *RcvS* fact is applied. Note that

the introduced rules model a channel that is automatically replay protected because the *Sec* fact is not persistent. In case replay should be allowed we can use a persistent fact *!Sec(A,B,m)* instead of *Sec(A,B,m)*.

The Dolve-Yao adversary is unable to break cryptographic functions, i.e., cryptography is assumed perfect. But, it is possible to enhance the adversary with more capabilities by modeling reveal rules. Thus, we are able to equip the protocol model with various compromise scenarios. For the toy protocol in Listing 3.1 we add the following rule, to compromise the symmetric key k. The rule takes the long-term key k shared by *id_X* and *id_Y* and sends it to the network. As a result, the adversary knows k.

Listing 3.3: Reveal Rule

```
1 rule reveal_ltk:
2  [!Ltk(~id_X, ~id_Y, ~k) ]
3  --[Rev(~id_X, ~k),
4     Rev(~id_Y, ~k) ]->
5  [Out(~k) ]
```

3.3 Protocol Execution

The multiset rewrite rules introduced in Section 3.1 define a transition system. Each rule defines a transition from a state S to another state T. A state itself is a multiset of facts, containing all state facts, and adversary knowledge. In order to be able to apply the multiset rewrite rule l --$[a]$-> r to the state S, S has to be a superset of l. The rule consumes all facts specified in l. Additionally, the rule generates the facts specified in r. As a result, the following state T contains all facts of S that were not consumed by l, plus all facts generated by r.

A protocol execution is defined as an alternate sequence of states and labeled multiset rewriting rule instances.

$$S_0, l_1 \xrightarrow{a_1} r_1, S_1, \ldots, l_k \xrightarrow{a_k} r_k, S_k$$

S_i describes the current state of the transition system at the time point i of the execution. S_0 denotes the initial state which is empty. Each applied rule at step i has to be applicable to state S_{i-1} and result in state S_i.

3.4 Security Properties

We define security properties in a first-order logic formula over traces. A trace of an execution is defined as the sequence of action facts of the labeled multiset rewriting rules, i.e., for the execution

$$S_0, l_1 \xrightarrow{a_1} r_1, S_1, l_2 \xrightarrow{a_2} r_2, \ldots, l_k \xrightarrow{a_k} r_k, S_k \ ,$$

the trace is defined as

$$a_1, a_2, \ldots, a_k \ .$$

A trace property is defined in a first-order logic formula which contains events and timepoints. The formula defines properties using the following syntax.

- $F@i$ defines the action fact F at timepoint i. The predicate is true in case $F@i \in a_i$ for the trace $a_1, a_2, \ldots, a_n$.
- $\#i = \#j$ defines equality of timepoints i and j.
- $x = y$ defines equality of term x and y.

We differ between the following action facts.

Protocol Events describe events that happen in the protocol. For instance *Create(*id_X*)* denotes that an agent with identity id_X has been created.

Claim Events describe assumptions an agent claims. *Running*, *Commit*, and *Secret* are examples for claim events. For instance if an agent claims $Secret(m)$, the agent assumes that the term m is unknown to the adversary at any time.

Honesty Events describe that certain agents are expected to be honest.

Reveal Events describe compromise scenarios.

Adversary Knowledge is described by $K(t)$, which denotes that the adversary knows term t.

We first introduce authentication properties, which is followed by secrecy properties. We translate each property definition into a first-order logic formula, called lemma. The lemmas can be applied to the toy protocol modeled in Listing 3.1. Note, the model of the toy protocol is only equipped with action facts to prove properties from the point of view of role Y to make the listing more readable.

3.4.1 Authentication Properties

We introduce authentication properties defined by Lowe [20] in Definitions 3.1-3.4. The authentication properties are ordered from weakest, i.e., aliveness, to strongest, i.e., injective agreement.

Definition 3.1 *(Aliveness) We say that a protocol guarantees to an initiator A aliveness of another agent B if, whenever A (acting as initiator) completes a run of the protocol, apparently with responder B, then B has previously been running the protocol.*

Informally, aliveness means that whenever A finishes a protocol run with B, then B has been running the protocol at least once with someone. The

communication partners of B do not matter for aliveness. The listing below introduces aliveness. The lemma shows aliveness of agent x to y.

Listing 3.4: Aliveness

```
1 lemma aliveness:
2 "All  x y t #i. Commit(y, x, t)@i
3  ==> (Ex #j. Create(x)@j)
4    | (Ex z k #r. Rev(z, k)@r & Honest(z)@i) "
```

The lemma uses the action facts specified in the protocol rules. It means that for every commit claim `Commit(y, x, t)` of agent y with agent x on term t, either agent x has been created before `Create(x)`, or there was a reveal `Rev(z, k)` of agent z that was supposed to be an honest protocol participant `Honest(z)` at timepoint i. Line 4 in aliveness specifies, that protocol participants that are called honest at i are uncompromised. The ordering of the commit claim and create event, i.e., the ordering of the timepoints i and j, is implicitly given to be $j < i$. Since a trace that has the commit claim before the create claim can be cutoff after the commit claim, which results in a counterexample for the lemma. Thus, the right-hand side of the implication has to causally precede the left-hand side of the implication.

Definition 3.2 *(Weak Agreement) We say that a protocol guarantees to an initiator A weak agreement with another agent B if, whenever A (acting as initiator) completes a run of the protocol, apparently with responder B, then B has previously been running the protocol, apparently with A.*

Weak agreement is stronger than aliveness. It ensures that whenever A finishes a protocol run with B, B has been running it at least once with A. However, weak agreement does not include roles. Thus, the commit claim includes term t, whereas the running claim may include a term $t2$ that differs from t. It also may be that both, A and B, think they acted as initiator. The listing below presents weak agreement as lemma.

Listing 3.5: Weak Agreement

```
1 lemma weakagreement:
2 "All  x y t #i. Commit(y, x, t)@i
3  ==> (Ex t2 #j. Running(x, y, t2)@j)
4    | (Ex z k #r. Rev(z, k)@r & Honest(z)@i) "
```

Definition 3.3 *(Non-injective Agreement) We say that a protocol guarantees to an initiator A non-injective agreement with a responder B on a term t if, whenever A (acting as initiator) completes a run of the protocol, apparently with responder B, then B has previously been running the protocol, apparently with A, and B was acting as responder in his run, and the two agents agreed on t.*

Non-injective agreement includes roles of the agents. Informally, non-injective agreement on term t of A with B ensures that whenever A finishes the protocol run with B, they agree on their identities, their roles and term t. Non-

injective agreement does not protect against replay attacks. In case A finishes two protocol runs with B, but B only finishes the protocol once, it is possible that non-injective agreement is satisfied. The listing below shows non-injective agreement as lemma.

Listing 3.6: Non-injective Agreement

```
1 lemma noninjectiveagreement_y_with_x:
2 "All  x y t #i. Commit(y, x, <'Y','X', t>)@i
3  ==> (Ex  #j. Running(x, y, <'Y','X',t>)@j)
4     | (Ex z k #r. Rev(z, k)@r & Honest(z)@i) "
```

Injective agreement on the other hand is only satisfied if each protocol run of A corresponds to a unique run of B.

Definition 3.4 *(Injective Agreement) We say that a protocol guarantees to an initiator A agreement with a responder B on a set of data items ds if, whenever A (acting as initiator) completes a run of the protocol, apparently with responder B, then B has previously been running the protocol, apparently with A, and B was acting as responder in his run, and the two agents agreed on t, and each such run of A corresponds to a unique run of B.*

Informally, injective agreement on term t of A with B ensures that non-injective agreement is satisfied and additionally each run of A corresponds to a unique run of B. The listing below introduces an injective agreement lemma. Line 4 ensures that for the running claim at timepoint j with the term $\langle 'Y', 'X', t \rangle$ exists only one matching commit claim, i.e., no other commit claim exist with term $\langle 'Y', 'X', t \rangle$ than the one at timepoint i.

Listing 3.7: Injective Agreement

```
1 lemma injectiveagreement_b_with_a:
2 "All  x y t #i. Commit(y, x, <'Y','X', t>)@i
3  ==> (Ex  #j. Running(x, y, <'Y','X',t>)@j
4     & not (Ex a2 b2 #i2. Commit(a2, b2, <'Y','X', t>)@i2 & not (#i2 = #i)))
5     | (Ex z k #r. Rev(z, k)@r & Honest(z)@i) "
```

3.4.2 Secrecy

Secrecy on a term t is satisfied in case the adversary is unable to learn t. We express the secrecy lemma as follows.

Listing 3.8: Secrecy

```
1 lemma secrecy:
2 "All y t #i. Secret(y, t)@i
3  ==> not (Ex #j. K(t)@j)
4     | (Ex z k #r. Rev(z, k)@r & Honest(z)@i)"
```

The lemma denotes that for all secret claims of agent y on term t, t is not part of the adversary knowledge. A trace that contains *K(t)* violates the secrecy property of t.

Perfect forward secrecy (PFS) of a session key is stronger than secrecy. A session key that satisfies perfect forward secrecy remains secret even in case long-term secrets are revealed after the session key was established. Perfect forward secrecy as lemma is defined as follows.

Listing 3.9: Perfect Forward Secrecy

```
1 lemma perfect_forward_secrecy:
2 "All a t #i. Secret(a, t)@i
3  ==> not (Ex #j. K(t)@j)
4    | (Ex X data #r. Rev(X, data)@r & Honest(z)@i & r < i)"
```

The lemma states that the term t has to be secret even in case a reveal happens after the secrecy claim. The time constraint is enforced with $r < i$.

3.5 Proof Search in Tamarin

The Tamarin prover uses symbolic backwards search to prove a property defined as lemma. Backwards search means that Tamarin starts from the negated property and searches backwards for a trace with a valid initial state. In case Tamarin is able to reach an initial state, the property is violated. Additionally, the found trace poses the counter example for the property. Otherwise, if Tamarin terminates and is unable to find an initial state, the property is satisfied.

For example, the proof search for the aliveness lemma shown in Listing 3.4 starts at the commit claim `Commit(y,x,t)`. Tamarin searches backwards a trace to an initial state. The trace is neither allowed to include a create claim of x nor a reveal of the honest participants x and y. This is not possible since the only initial state includes a create claim. Thus, from the point of view of Y, aliveness of X is satisfied.

The Tamarin prover supports two modes to prove properties. We can guide a proof manually using the interactive mode. The automatic mode relies on heuristics to automatically prove a property. The heuristics prioritize certain proof goals. In case the Tamarin prover is slow to prove a property or does not terminate at all it is convenient to first manually inspect the proof search. The heuristics can be tweaked by the use of an oracle, which allows to rank goals differently. With the use of an oracle Tamarin may be able to automatically prove the property.

Chapter 4

Tamarin Model

This chapter presents our formal models of the AKA protocol variants. The models are available at [3]. Our AKA models are based on the Tamarin model of 5G AKA presented by Basin et al. [14], henceforth referred to as `5g-aka-2018`.

We first summarize modeling choices introduced in `5g-aka-2018` in Section 4.1. Second, we introduce additional model choices in Section 4.2. Last, we discuss the automatic generation of the various Tamarin models in Section 4.3.

4.1 Model Choices of previously analyzed 5G AKA

The following listing summarizes the model decisions of `5g-aka-2018`.

Channels The channel between a SN and HN is authentic, confidential and replay protected. `5g-aka-2018` uses the secure channel rules introduced in Listing 3.2, called non-binding channel. `5g-aka-2018` is modeled once using the non-binding channel and once using a binding channel. The binding channel additionally preserves the order of messages, i.e., requests from the SN and answers from the HN are not confused. Listing 4.1 shows the rules of a secure channel which binds send and receive rules together. $\sim cid$ denotes a channel identifier.

Listing 4.1: Binding Channel

```
1 rule send_secure:
2  [SndS(~cid,A,B,m)]
3  -->
4  [Sec(~cid,A,B,m)]
5
6 rule receive_secure:
7  [Sec(~cid,A,B,m)]
8  -->
9  [RcvS(~cid,A,B,m)]
```

All messages between the SN and HN are tagged with its message type in order to prevent type flaw attacks. For instance, the authentication initiation request (AIR) is tagged with "air".

Roles `5g-aka-2018` includes three roles, the UE, the SN, and the HN. Each UE is restricted to be subscribed to only one HN.

Sequence Number SQNs are not allowed to repeat, otherwise it is possible that the same MK is derived multiple times. In order to tackle this issue, SQNs are modeled as monotonically increasing natural numbers. In that way SQNs are automatically protected against wrapping around. `5g-aka-2018` assumes that an attacker is not able to follow the UE from creation. Thus, the SQN starts as random nonce, called the SQN root. Additionally, the model allows arbitrary increase of the SQN_{UE}. The increase of SQN_{HN} is trivially possible since a SN is able to repeat an authentication initiation request to the HN arbitrarily many times. SQNs are concealed using the built-in XOR function.

Compromise scenarios `5g-aka-2018` covers several compromise scenarios. Consequently it is possible to detect minimal assumptions that have to hold in order to satisfy a certain property. Potential compromises are listed below.

- Secure channel reveal between the SN and HN, i.e., it is possible to read or write to the secure channel.
- Subscriber long-term secret key K
- SQN root
- Permanent identity of the UE
- Private key of the HN sk_{HN}

Parameter Lengths Key derivation functions with parameters $\langle t1, t2, .. \rangle$ are specified to include the lengths of each term: $\langle t1, l1, t2, l2, .. \rangle$, where l1 and l2 denote the length of term t1 and t2, respectively. However, since type flaw attacks based on parameter lengths are not possible in a symbolic model, `5g-aka-2018` omits lengths as parameter in all key derivation functions.

Constants `5g-aka-2018` ignores all constants, except they are useful as tags in a key derivation function.

Key Identifier The key set identifier (KSI) is used after a successful AKA run in order to identify an already established key without invoking the authentication procedure again. Such an identifier is not modeled in `5g-aka-2018`, because the KSI is used only after the AKA protocol.

MAC Failure `5g-aka-2018` does not include MAC failure messages. The model assumes that the UE starts another authentication attempt with an initiation request to the SN in case of an invalid MAC.

Temporary Identities The SN is able to recognize a UE using a temporary identity instead of its permanent identity for successive authentication requests. However, `5g-aka-2018` does not model temporary identifier of subscribers. Security properties should be satisfied even when using a permanent identity.

Expiration In `5g-aka-2018` the authentication token ($AUTN$) that the SN uses to authenticate the UE does not expire, which is usual in symbolic models.

Key confirmation `5g-aka-2018` includes an additional round called key confirmation, which takes place after successful authentication (see Figure 4.1). The SN and UE exchange two messages that make use of the agreed master key MK. The key confirmation is not a part of the actual AKA protocol, but `5g-aka-2018` adds the extra round to compare resulting agreement guarantees before and after key confirmation. The different agreements are distinguished using two distinct lemma types. Below are two examples of weak agreement from the point of view of a UE with a SN. The first shows weak agreement before key confirmation.

Listing 4.2: Before Key Confirmation

```
1 lemma weakagreement_ue_sn_noRev:
2 "All ue sn t #i.
3  Commit(ue, sn, <'UE','SN', t>)@i
4  ==> (Ex t2 #j. Running(sn, ue, t2)@j)
5    | (Ex X data #r. Rev(X, data)@r & Honest(X)@i)"
```

Weak agreement after key confirmation is presented below. Instead of `Commit` the lemma uses `CommitConf`. Additionally, the lemma name is tagged with `keyConf`. The right-hand side of the implication, i.e., line 4 and 5, is the same for both lemma types.

Listing 4.3: After Key Confirmation

```
1 lemma weakagreement_ue_sn_keyConf_noRev:
2 "All ue sn t #i.
3  CommitConf(ue, sn, <'UE','SN', t>)@i
4  ==> (Ex t2 #j. Running(sn, ue, t2)@j)
5    | (Ex X data #r. Rev(X, data)@r & Honest(X)@i)"
```

4.2 Model Choices

As suggested in `5g-aka-2018` and already mentioned in Chapter 2, we introduce three roles, the UE, SN, and HN. In addition to the model choices presented in `5g-aka-2018`, we introduce the following decisions:

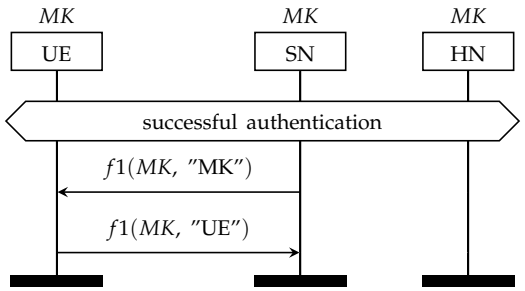

Figure 4.1: After successful authentication the SN and UE use the shared MK in order to exchange two messages.

Multiple AVs In 3G AKA and 4G EPS AKA the SN receives multiple AVs from the HN at once to reduce the communication overhead between the SN and HN. The derived MK is only valid for one session with the UE. Afterwards, the SN has to generate a new MK by challenging the UE again. In order to consistently compare all protocol variants, we model the SN to receive one AV at a time. Agreement and secrecy properties should not be affected by this modeling choice.

Bit Range Key derivations in 5G EAP AKA′ make use of bit ranges, for instance $Key\big[x \ldots y\big]$ in order to access bit x to y of Key. Tamarin does not argue over bit strings, thus we tag the derivation of terms that rely on a bit range. For instance, the derivation of CK' is defined as

$$CK' \leftarrow kdf(K, \langle R, SQN, \text{“5G”} : idSN \rangle)\big[0..127\big].$$

We tag the key derivation with “1” to model CK' in Tamarin:

$$CK' \leftarrow kdf(K, \langle R, SQN, \text{“5G”} : idSN, \text{“1”} \rangle).$$

IK' on the other hand is tagged with “2”.

AMF In contrast to `5g-aka-2018` we model parts of the authentication management field (AMF), namely the separation bit. The AMF separation bit allows to clearly distinguish 3G AKA from newer generations. The AMF separation bit is set to zero in 3G AKA, while it is set to one in newer generations. The AMF is protected by a MAC, thus only the UE or the HN are able to change it.

Same K and UE Identity We assume that each AKA variant uses the same shared secret key K and UE identity, which describes the worst case scenario. This decision does not impact the separate AKA protocols, but may have an impact on combined models.

4.3 Automatic Model Generation

We manually generate the generic AKA flow presented in Figure 2.2 using the model choices introduced in the previous two sections. From the generic AKA model we automatically instantiate the specific AKA variants introduced in Chapter 2. Characteristics that do not concern all AKA variants are labeled with keywords such as *ADDLINE*, *TOREPLACE*, and many more. The full list of keywords is explained in more detail in `README.md` in [3]. The main benefit of the automatic workflow is that we modify all models in one place only. Thus, changes and updates for one model are automatically included in the other models as well. As consequence, not every model has to be adjusted manually, which is not only tedious work but also prone to failures.

4.3.1 Combined AKA Model

We automatically instantiate combined models using keywords in the same way as an instantiation of a separate AKA variant. Initialization rules, compromise rules, and restrictions, which do not differ across AKA variants, are included once. We add the specific protocol rules and lemmas for every AKA variant once. Three main points have to be adjusted in every protocol rule and lemma for each AKA variant. First, Tamarin requires rule and lemma names to be unique. Thus, we label names with its AKA version. For instance in 3G AKA, the rule called `ue_send_attach_req` turns into `3G_AKA_ue_send_attach_req`. Second, we tag claims with its version. Instead of `Commit`, we use `Commit_3G_AKA` for 3G AKA. This allows to enforce that the protocol participants use the same AKA versions. As a result, disagreement on the protocol versions are not allowed. Finally, each state fact includes the current AKA variant, e.g., `St_1_UE(..., ''3g_aka'')`. As a consequence, it is not possible that a running instance of a role switches the protocol version during a protocol execution.

We define $A \cup B$ to be the model that combines protocol A and protocol B in one model. $A \cup B$ contains all rules and lemmas for both A and B. We generate the following protocol combinations:

- 3G AKA $\cup$ 4G EPS AKA
- 4G EPS AKA $\cup$ 5G AKA
- 3G AKA $\cup$ 4G EPS AKA $\cup$ 5G AKA

We present the security analysis of the separate and combined models in the next chapter.

Chapter 5

Security Analysis

This chapter presents our analysis of the AKA protocol variants. Each of our Tamarin models [3] is generated automatically using the same modeling decisions and abstractions. This workflow enables a consistent comparison of the resulting security guarantees of different AKA protocol variants. Table 5.1 summarizes the modeled protocols presenting the time Tamarin requires to prove all lemmas for each protocol. The lemmas are proven on a machine running Ubuntu 16.04 with an Intel Xeon CPU E5-2650 v4 @ 2.20 GHz. Tamarin is restricted to 10 threads and runs in version 1.5.1[1].

Before we take a look at the security properties of each individual AKA protocol variant in Section 5.3, we introduce notation that we use in this chapter in Section 5.1 and special lemma types in Section 5.2. Section 5.4 compares communication complexity and cost across AKA variants. We conclude this chapter by presenting the analysis of the combined models in Section 5.5.

Table 5.1: Summary of modeled protocols presenting the rough time Tamarin prover requires to prove all lemmas per model. "$A \cup B$" denotes that protocol A and B are combined in one model.

	Non-Binding Channel		Binding Channel	
Tamarin Model	Time	# Lemmas	Time	# Lemmas
3G AKA	19 min	98	26 min	133
4G EPS AKA	28 min	113	48 min	174
5G AKA	59 min	175	1h 11 min	175
5G EAP AKA'	2h 24 min	253	2h 44 min	253
3G AKA ∪ 4G EPS AKA	1h 13 min	147	2h 18 min	284
4G EPS AKA ∪ 5G AKA	2h 36 min	265	3h 12 min	330
3G AKA ∪ 4G EPS AKA ∪ 5G AKA	4h 32 min	295	6h 12 min	431

[1]Git revision hash: c3c3cec55eabe3f4cd808858b0da4e79f5549f99

5.1 Notation

This section explains the notation and abbreviations used in this chapter. First we introduce how security properties are presented. ✗ denotes that the security property is violated under the assumption that all protocol participants are honest. We annotate a satisfied security property with its minimal assumption. Minimal means that the property is violated if we make the attacker stronger in any way. We represent minimal assumptions as a formula. For example, let us assume we have a property p and three assumptions A, B, and C. Further, we assume p requires the assumptions A and B to be satisfied. The assumption for p is then described with the formula $A \wedge B$. We ensure that the assumption $A \wedge B$ is minimal by providing two attack lemmas, one that uses $A \wedge C$, and the other uses $B \wedge C$.

The following listing summarizes possible statements to be used in a minimal assumption formula.

$\neg K$ does not allow the shared long-term secret K between the UE and HN to be revealed.

$\neg sqn$ does not allow the sequence number of the UE or HN to be revealed.

$\neg ch$ denotes that the channel between the SN and HN has to be uncompromised. Note, in this case only channel reveal of honest participants are disallowed.

$\neg CH$ is a stronger assumption than $\neg ch$. It denotes that all channels from and to an honest HN have to be uncompromised. This assumption is still weaker than requesting all channels to be uncompromised.

$\neg idUE$ denotes that the UE identity has to be uncompromised. In versions 3G AKA and 4G EPS AKA this is represented by $\neg imsi$, whereas in 5G AKA and 5G EAP AKA' it is represented by $\neg supi$. In the combined models 4G EPS AKA $\cup$ 5G AKA and 3G AKA $\cup$ 4G EPS AKA $\cup$ 5G AKA we use $\neg idUE$.

$\neg sk_{HN}$ is the condition that the private key of the HN is not revealed, which only makes sense in 5G AKA or 5G EAP AKA'.

$secretSupi$ is an abbreviation for $\neg supi \wedge \neg sk_{HN} \wedge \neg CH$. It is used in order to make the minimal assumptions used in this chapter more readable. The statement includes the assumptions that are needed to ensure that SUPI remains secret.

B denotes the need of a binding channel between the SN and HN. In this case a session ID identifies a running session.

kc denotes that an extra key confirmation round is needed in order to satisfy the property.

✓ denotes that the property is satisfied no matter what compromise scenarios occur.

We label a property that we do not prove actively surrounded with $[\dots]$, which means the property is already implied by another property. For instance, if aliveness is violated, then a stronger property, e.g., weak agreement, is violated as well. Weak agreement is then marked with $[\text{✗}]$. Additionally, the chapter uses the following abbreviations:

WA – Weak agreement

NIA – Non-injective agreement

IA – Injective agreement

idUE – UE identity, either IMSI or SUPI

idSN – SN identity

MK – Master key

5.2 Lemmas

In some cases we have to rely on strong assumptions in order to satisfy some security properties. We first introduce a lemma type that uses the $\neg CH$ assumption. Second, we present anonymous lemmas to accept a mismatch of the identities of protocol participants.

5.2.1 No HN Channel Reveal

$\neg CH$ denotes that all channels from and to an honest HN have to be uncompromised. That means, $\neg CH$ does not allow any channel compromise from or to the HN where the UE is subscribed. This assumption is weaker than requesting all channels between an HN and a SN to be uncompromised, but stronger than $\neg ch$ that does not allow a channel compromise of honest protocol participants.

First we show an example of an aliveness lemma of the UE with the SN using the weaker assumption $\neg ch$.

Listing 5.1: Example Lemma using $\neg ch$

```
1 lemma aliveness_ue_sn_noChanRev:
2 "All ue sn t #i. Commit(ue, sn, <'UE','SN', t>)@i
3 ==> (Ex #j. ServNet(sn)@j)
4  | (Ex X #r. Rev(X, 'secureChannel')@r & Honest(X)@i) "
```

The lemma is satisfied in case either for every commit of *ue* with *sn*, *sn* was initiated before, or there exists a channel reveal of an honest protocol participant. $\neg CH$ additionally assumes every channel from and to the HN

hn where *ue* is subscribed not to be revealed. The two statements which include 'secureChannelBoth' disallow a channel reveal from another SN *sn2* while communicating with *hn*, and *ue* is subscribed at *hn*. We have to use both statements to cover both channel directions.

Listing 5.2: Example Lemma using $\neg CH$

```
1 lemma aliveness_ue_sn_noHNChanRev:
2 "All ue sn t #i. Commit(ue, sn, <'UE','SN', t>)@i
3 ==> (Ex #j. ServNet(sn)@j)
4   | (Ex X #r. Rev(X, 'secureChannel')@r & Honest(X)@i)
5   | (Ex hn sn2 #k #r. Rev(<hn, sn2>, 'secureChannelBoth')@r
6    & Subscribe(ue, hn)@k & Honest(hn)@i)
7   | (Ex hn sn2 #k #r. Rev(<sn2, hn>, 'secureChannelBoth')@r
8    & Subscribe(ue, hn)@k & Honest(hn)@i) "
```

5.2.2 Anonymous Agreement

We introduce anonymous agreement lemmas to allow a certain mix-up between the identities of protocol participants. This is necessary because in some cases even weak agreement properties fail because of a disagreement on the agent identities. The anonymous agreement lemmas are based on the bachelor's thesis of Vincent Stettler [23]. The following example lemma allows disagreement on the UE identity.

Listing 5.3: Anonymous Initiator

```
1 lemma anonymous_ue_weakagrement_ue_sn_noRev:
2 "All ue sn t #i. Commit(ue, sn, <'UE','SN', t>)@i
3 ==> (Ex ue2 t2 #j. Running(sn, ue2, t2)@j)
4   | (Ex X data #r. Rev(X, data)@r & Honest(X)@i) "
```

Note that the identity of the initiator, the UE, is allowed to be different in the commit and running claim. The lemma states that in case *ue* commits on talking with *sn* on term *t*, then either *sn* was running with someone (*ue2*) on term *t2*, or there was a reveal from an honest agent. In this example we are able to allow a disagreement on the identity of the UE, see *ue2*. Be aware that weak agreement does not include roles in the running claim. Consequently anonymous weak agreement of the UE with a SN is not able to ensure that the partner of the UE is indeed an instance of a SN.

In the following example, we are not able to allow a disagreement on the responder, which is explained in the following example.

Listing 5.4: Anonymous Responder (attempt)

```
1 lemma anonymous_sn_weakagrement_ue_sn_noRev:
2 "All ue sn t #i. Commit(ue, sn, <'UE','SN', t>)@i
3 ==> (Ex sn2 t2 #j. Running(sn2, ue, t2)@j)
4   | (Ex X data #r. Rev(X, data)@r & Honest(X)@i) "
```

At first glance, the lemma ensures that the running claim originates from some SN *sn2*. However, *ue* is not able to ensure that *sn2* is honest, since

ue does not know the identity of *sn2*. As a result, reveal rules for *sn2* are applied and the lemma fails.

We will not consider anonymous aliveness, because anonymous aliveness is the same as aliveness. As already mentioned above, the idea of anonymous lemmas is that two different identities of the initiator are allowed to be used on the left and right-hand side of the implication. However, in aliveness, the communication partners of the responder are irrelevant. Thus, it does not make a difference if the initiator is anonymous or not. We explain it in more detail with the following example of an aliveness lemma from the point of view of the UE with the SN.

Listing 5.5: Aliveness

```
1 lemma aliveness_ue_sn_noRev:
2 "All ue sn t #i. Commit(ue, sn, <'UE','SN', t>)@i
3 ==> (Ex #j. ServNet(sn)@j)
4  | (Ex X data #r. Rev(X, data)@r & Honest(X)@i) "
```

The anonymous lemma of the example would allow two different UE identities, i.e., *ue* on the left-hand side and *ue2* on the right-hand side of the implication. But, aliveness only checks that the SN *sn* was initialized at some point using *ServNet(sn)*. The identity of the UE is irrelevant in the initialization of the SN. Thus, anonymous aliveness and aliveness lemmas are the same.

5.3 AKA Security Analysis

We present satisfied or violated security properties of each AKA protocol variant. This section focuses on the comparison of AKA protocol variants. In order to get an overview of a single AKA variant refer to the tables in the Appendix A. We first discuss agreement properties for each pair of roles, which includes agreement on the MK and on the identities of the UE and SN. We do not consider agreement on the HN identity. The UE identity already contains the identity of its HN. Be aware that injective agreement *on identities* is always violated, because they are constants. Following the agreement properties, we present secrecy properties for each role.

5.3.1 UE Agreement with SN

Table 5.2 summarizes the security properties that the UE achieves with the SN. Table 5.3 summarizes anonymous agreement properties, which only shows 3G AKA and 4G EPS AKA. 5G AKA and 5G EAP AKA' are not included because they already satisfy the stronger agreement properties shown in the first table without the use of a binding channel (B). First, we summarize agreement properties *before* the extra key confirmation round. Second, we present agreement properties *after* key confirmation.

Table 5.2: Summary of security properties of the **UE with SN**.

Security Property	3G AKA	4G EPS AKA	5G AKA	5G EAP AKA'
Aliveness	✗	$kc \wedge \neg K \wedge \neg ch$	$kc \wedge \neg K \wedge \neg ch$	$(\neg K \wedge \neg ch) \vee secretSupi$
WA	[✗]	$B \wedge kc \wedge \neg K \wedge \neg ch$	$kc \wedge \neg K \wedge \neg ch$	$kc \wedge ((\neg K \wedge \neg ch) \vee secretSupi)$
NIA on idUE	[✗]	$B \wedge kc \wedge \neg K \wedge \neg ch$	$kc \wedge \neg K \wedge \neg ch$	$kc \wedge ((\neg K \wedge \neg ch) \vee secretSupi)$
IA on idUE *(const.)*	[✗]	✗	✗	✗
IA on MK	[✗]	$B \wedge kc \wedge \neg K \wedge \neg ch$	$kc \wedge \neg K \wedge \neg ch$	$kc \wedge ((\neg K \wedge \neg ch) \vee secretSupi)$
NIA on idSN	[✗]	✗	$kc \wedge \neg K \wedge \neg ch$	$kc \wedge ((\neg K \wedge \neg ch) \vee secretSupi)$
IA on idSN *(const.)*	[✗]	[✗]	✗	✗

Table 5.3: Summary of **anonymous** agreement properties of the **UE with SN**.

Security Property	3G AKA	4G EPS AKA
Anonymous WA	✗	$kc \wedge \neg K \wedge \neg ch$
Anonymous NIA on idUE	[✗]	$B \wedge kc \wedge \neg K \wedge \neg ch$
Anonymous IA on idUE *(const.)*	[✗]	✗
Anonymous IA on MK	[✗]	$kc \wedge \neg K \wedge \neg ch$
Anonymous NIA on idSN	[✗]	✗
Anonymous IA on idSN *(const.)*	[✗]	[✗]

Before Key Confirmation

Aliveness fails for 3G AKA, 4G EPS AKA and 5G AKA. This is due to a mismatch of the SN identity. The UE *ue* receives a wrong, non-existing SN identity $idSN_{fake}$. The issue is that there does not exist an initialization of a SN with the identity $idSN_{fake}$, hence aliveness fails. Figure 5.1 shows the described trace including relevant claims. Opposite to 3G AKA, in 4G EPS AKA and 5G AKA the UE and HN will compute a different MK because the SN identity is used in the MK derivation. However, that does not matter since the MK is not used.
Aliveness is only satisfied in 5G EAP AKA'. The attack presented for the other versions is not possible for the following reason: The UE receives the *Authentication Request* message from the SN, which not only contains the challenge but also a MAC over the challenge called AT_MAC. The identity of the SN is used in the derivation of the MAC key K_{auth}. The UE only proceeds if the received message contains a valid AT_MAC. Thus, an intermediate party is not able to change the SN identity without the UE noticing.

Weak agreement fails trivially for all versions for various reasons. The failing aliveness property for versions 3G AKA, 4G EPS AKA and 5G AKA implies that weak agreement and anonymous weak agreement fail as well. In 5G EAP AKA' on the other hand, it is not possible to have a running claim at the SN which causally precedes the commit claim of the UE. See Figure 5.2 for a trace which includes relevant claims. The SN *sn* receives the unconcealed UE identity *idUE* only in the last message from the HN *hn*. Thus, the earliest point for *sn* to make a running claim is after it receives the last message

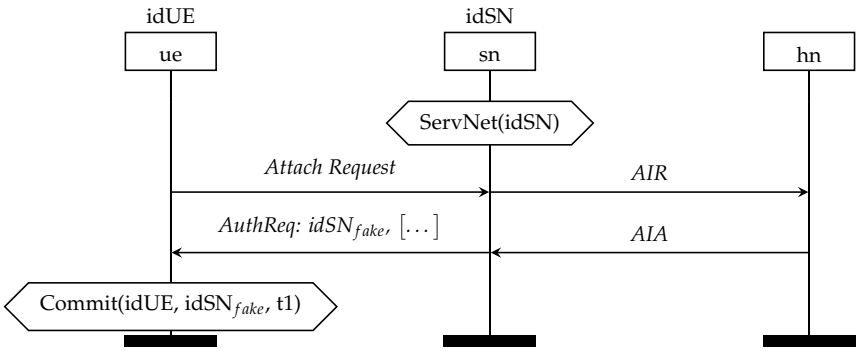

Figure 5.1: Violation of the aliveness property before key confirmation, from the perspective of the UE with the SN in versions 3G AKA, 4G EPS AKA and 5G AKA. The issue is that the UE *ue* receives a non-existing SN identity idSN_{fake}.

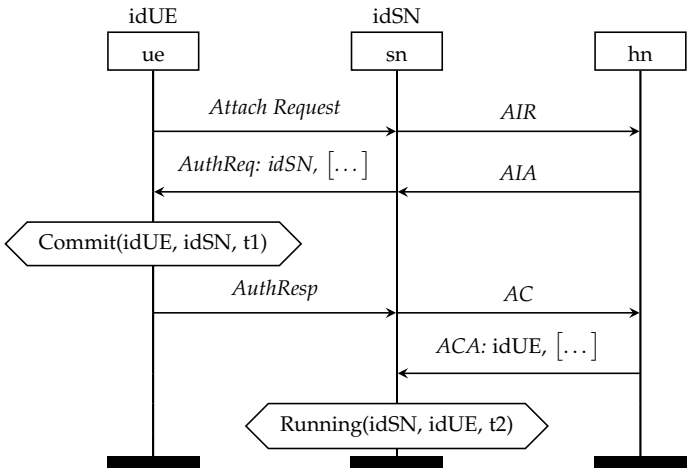

Figure 5.2: Commit and running claims for weak agreement of the UE with the SN for versions 5G AKA and 5G EAP AKA'. It is not possible that the running claim causally precedes the commit claim. The running claim cannot be earlier, because *sn* does not know the identity of *ue*. As a result, weak agreement fails trivially.

from *hn*. Note, this is also the reason why 5G EAP AKA' does not satisfy anonymous weak agreement.

After Key Confirmation

With the use of a key confirmation step we are able to satisfy more agreement properties for 4G EPS AKA, 5G AKA and 5G EAP AKA'. However, key confirmation does not help to satisfy aliveness in 3G AKA. The UE still receives a non-existing SN identity, which allows an attack like the one on

Table 5.4: Summary of agreement properties of the **UE with HN**.

Security Property	3G AKA	4G EPS AKA	5G AKA	5G EAP AKA'
WA	$\neg K$	$\neg K$	$\neg K$	$\neg K \vee (\neg supi \wedge \neg sk_{HN})$
NIA on idUE	$\neg K$	$\neg K$	$\neg K$	$\neg K \vee (\neg supi \wedge \neg sk_{HN})$
IA on idUE *(const.)*	✗	✗	✗	✗
IA on MK	$\neg K$	$kc \wedge \neg K$	$kc \wedge \neg K$	$\neg K \vee secretSupi$
NIA on idSN	✗	$kc \wedge \neg K$	$kc \wedge \neg K$	$\neg K \vee secretSupi$
IA on idSN *(const.)*	[✗]	✗	✗	✗

aliveness before key confirmation, mentioned in Figure 5.1. In 4G EPS AKA and 5G AKA, the mentioned attack on aliveness is not possible anymore. This is because the SN identity is used in the MK derivation, and the MK is used in the key confirmation step.

4G EPS AKA, 5G AKA and 5G EAP AKA' satisfy non-injective agreement on the UE identity and injective agreement on the MK. Non-injective agreement on the SN identity is only satisfied in the 5G versions of AKA. Differences between the AKA variants especially concern the minimal assumptions that are needed to satisfy the agreement properties. 5G EAP AKA' relies on the weakest assumption of the three. Surprisingly it is still possible to satisfy the named properties even in case the shared secret K is revealed, as long as the identity of the UE remains secret. Another party is only able to compute $MK_{5'}$ or AT_MAC if it has both, the shared secret K and the UE identity. 4G EPS AKA and 5G AKA on the other hand do not satisfy any properties if K is compromised. 5G AKA relies on the long-term secret K and the channel between the SN and HN to be uncompromised. 4G EPS AKA relies on the strongest assumptions. Non-injective agreement on the UE identity requires an uncompromised and binding channel. Otherwise a mix-up of UE identities is possible. The same holds for injective agreement on the MK, where a binding channel must be used. On the other hand, anonymous injective agreement on the MK is satisfied without a binding channel using the same assumptions 5G AKA requires for injective agreement on the MK.

5.3.2 UE Agreement with HN

Table 5.4 summarizes the security properties that the UE achieves with the HN. All AKA variations satisfy non-injective agreement on the UE identity and injective agreement on the MK. Surprisingly only 3G AKA and 5G EAP AKA' satisfy injective agreement on the MK *without* key confirmation. 4G EPS AKA and 5G AKA are vulnerable to an attack similar to the attack on aliveness of the UE with the SN introduced previously (see Figure 5.1). The UE receives a wrong SN identity. That is also the reason why 4G EPS AKA and 5G AKA rely on an extra key confirmation to satisfy non-injective

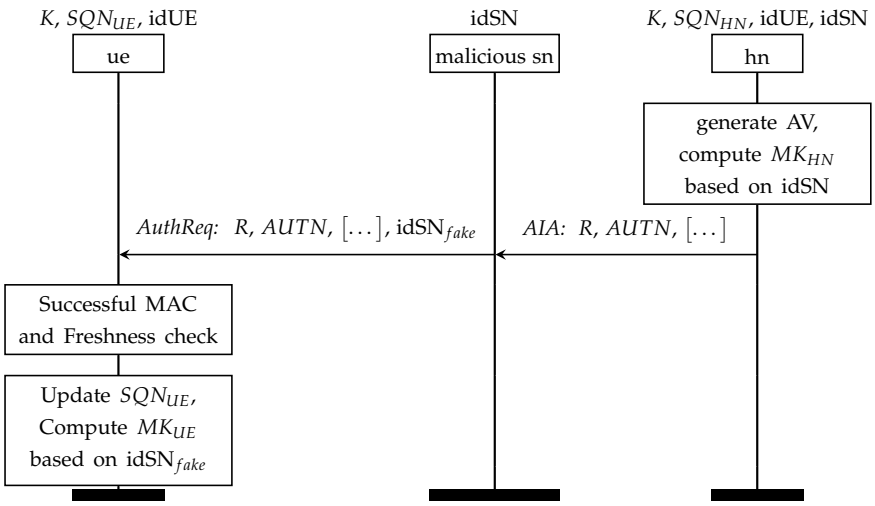

Figure 5.3: Violation of non-injective agreement on the MK *without* key confirmation of the UE with the HN in versions 4G EPS AKA and 5G AKA. The trace only shows messages that are relevant for the attack, initiation messages are ignored. The UE and HN use different SN identities to derive the MK.

agreement on the SN identity and 3G AKA does not satisfy non-injective agreement on the SN identity at all. Figure 5.3 shows the violation of the agreement on the MK before key confirmation in 4G EPS AKA and 5G AKA. Note that the trace does not include initiation messages. Ultimately, the UE *ue* and HN *hn* derive two different MKs. In more detail, a malicious SN does not send its own identity, but sends $idSN_{fake}$. This is possible since the SN identity is not protected with a MAC. *ue* computes MK_{UE} based on $idSN_{fake}$, whereas *hn* computes MK_{HN} based on idSN. As a result, *ue* and *hn* do not agree on the master key. Nevertheless, the malicious SN is not able to derive MK_{UE}. Consequently *ue* is not able to communicate with the malicious SN using MK_{UE}, which weakens the attack. That is also the reason why this attack is not possible after key confirmation. The UE and SN have to be able to communicate using the derived MK.
Despite the fact that 3G AKA does not satisfy non-injective agreement on the SN identity, it is not vulnerable to the mentioned attack. This is due to the MK derivation, which does not include the identity of the SN in 3G AKA. 5G EAP AKA' on the other hand is not vulnerable since the SN identity is protected with AT_MAC.

While other AKA variants rely on the assumption that K is not compromised, 5G EAP AKA' satisfies properties using weaker assumptions. Either K or the UE identity has to be secret. As described before, 5G EAP AKA' includes an additional MAC, which protects messages from tampering. The assumption for non-injective agreement on the identity of the UE as well as

Table 5.5: Summary of agreement properties of the **SN with UE**. See Table 5.6 for anonymous properties.

Security Property	3G AKA	4G EPS AKA	5G AKA	5G EAP AKA'
Aliveness	$B \wedge \neg ch$	$B \wedge \neg ch$	$\neg ch$	$\neg ch$
WA	✗	$B \wedge kc \wedge \neg K \wedge \neg ch$	$\neg K \wedge \neg ch$	$(\neg K \wedge \neg ch) \vee secretSupi$
NIA on idUE	[✗]	$B \wedge kc \wedge \neg K \wedge \neg ch$	$\neg K \wedge \neg ch$	$(\neg K \wedge \neg ch) \vee secretSupi$
IA on idUE *(const.)*	[✗]	✗	✗	✗
IA on MK	[✗]	$B \wedge kc \wedge \neg K \wedge \neg ch$	$\neg K \wedge \neg ch$	$(\neg K \wedge \neg ch) \vee secretSupi$
NIA on idSN	[✗]	$B \wedge kc \wedge \neg K \wedge \neg ch$	$\neg K \wedge \neg ch$	$(\neg K \wedge \neg ch) \vee secretSupi$
IA on idSN *(const.)*	[✗]	✗	✗	✗

weak agreement have an even weaker assumption than injective agreement on the MK, namely $\neg K \vee (\neg supi \wedge \neg sk_{HN})$. The assumption makes sure that either K is uncompromised, or the UE identity is secret as long as the UE has not yet finished one protocol run. After a successful protocol run from the point of view of the UE, the UE identity can be leaked in a channel reveal. In that way, for each commit of the HN that follows, there exist at least one running claim on the UE identity.

5.3.3 SN Agreement with UE

We present the agreement properties that the SN achieves with the UE as partner. We divide the discussion into three parts. First, we discuss agreement properties before key confirmation, which is followed by a discussion of properties after key confirmation. Those results are summarized in Table 5.5. Last we present results of anonymous agreement properties.

In case of 3G AKA and 4G EPS AKA, we must use a binding channel to satisfy any agreement properties. Otherwise it is possible that the SN confuses AVs received from the HN. This is not possible in the 5G variants of the AKA protocol because the HN also checks the response of the UE.

Before Key Confirmation

All AKA variants satisfy aliveness, if the channel is uncompromised. $\neg ch$ has to be enforced because the SN is not able to distinguish a fake AV from a real AV originating from the HN. 3G AKA and 4G EPS AKA do not satisfy further agreement properties. Weak agreement is violated because the UE receives a non-existing SN identity. As result, the UE uses a non-existing SN identity in the running claim. Thus, no running claim from a UE exist that contains the real SN identity. The 5G variants of the AKA protocol satisfy injective agreement on the MK and non-injective agreement on the identities of the SN and UE. As we have already seen in Section 5.3.1 5G EAP AKA' relies on weaker assumptions in order to satisfy the properties.

Table 5.6: Summary of **anonymous** agreement properties of the **SN with UE**. Note that 5G AKA and 5G EAP AKA' are not included in this table, since the security properties are implied by the properties shown in Table 5.5.

Security Property	3G AKA	4G EPS AKA
Anonymous WA	$B \wedge \neg ch \wedge (\neg K \vee \neg imsi)$	$B \wedge \neg ch \wedge (\neg K \vee \neg imsi)$
Anonymous NIA on idUE	$B \wedge \neg K \wedge \neg ch$	$B \wedge \neg K \wedge \neg ch$
Anonymous IA on idUE *(const.)*	✗	✗
Anonymous IA on MK	$B \wedge \neg K \wedge \neg ch$	$B \wedge kc \wedge \neg K \wedge \neg ch$
Anonymous NIA on idSN	✗	$B \wedge kc \wedge \neg K \wedge \neg ch$
Anonymous IA on idSN *(const.)*	[✗]	✗

After Key Confirmation

While 3G AKA does not improve with an extra key confirmation round, 4G EPS AKA now satisfies injective agreement on the MK and non-injective agreement on the identities of the SN and UE. The improvement on satisfied properties is due to the MK derivation which includes the SN identity.

Anonymous Agreements

Anonymous agreement properties are summarized in Table 5.6 which only includes 3G AKA and 4G EPS AKA. Anonymous agreement properties for 5G AKA and 5G EAP AKA' are implied by the stronger non anonymous agreement properties. In 3G AKA the anonymous agreement lemmas are satisfied because the UE is allowed to run with a wrong SN identity. In that way it is possible to achieve anonymous injective agreement on the MK and non-injective agreement on the identity of the UE. In order to satisfy injective agreement on the MK in 4G EPS AKA, the SN has to agree also on the SN identity. Otherwise the UE derives a different MK than the SN receives from the HN. This is only possible if an extra key confirmation round is enforced. In both versions, anonymous weak agreement is satisfied using $(B \wedge \neg ch \wedge \neg K) \vee (B \wedge \neg ch \wedge \neg imsi)$. Be aware of the fact that $B \wedge \neg ch \wedge \neg imsi$ is only sufficient because weak agreement does not include roles in the running claim. That means, the running claim does not originate from the UE, however, it originates from the HN. If we would redefine the weak agreement to includes role names in the term, e.g., change *Running(b, c, t2)* to *Running(b, c,⟨'SN','UE', t2⟩)*, the lemma fails. The minimal assumption would then just be $B \wedge \neg ch \wedge \neg K$.

5.3.4 SN Agreement with HN

Table 5.7 summarizes the security properties that the SN achieves with the HN as partner. All AKA variants satisfy injective agreement on the MK and non-injective agreement on the identities of the UE and SN. Since the SN

Table 5.7: Summary of agreement properties of the **SN with HN**.

Security Property	3G AKA	4G EPS AKA	5G AKA	5G EAP AKA'
Aliveness	$\neg ch$	$\neg ch$	$\neg ch$	$\neg ch$
WA	$\neg ch$	$\neg ch$	$\neg ch$	$\neg ch$
NIA on idUE	$B \wedge \neg ch$	$B \wedge \neg ch$	$\neg ch$	$\neg ch$
IA on idUE *(const.)*	✗	✗	✗	✗
NIA on MK	$\neg ch$	$\neg ch$	$\neg ch$	$\neg ch$
IA on MK	$\neg K \wedge \neg ch$	$\neg K \wedge \neg ch$	$\neg K \wedge \neg ch$	$(\neg K \wedge \neg ch) \vee secretSupi$
NIA on idSN	$\neg ch$	$\neg ch$	$\neg ch$	$\neg ch$
IA on idSN *(const.)*	✗	✗	✗	✗

Table 5.8: Summary of early security properties of the **HN with UE**.

Security Property	3G AKA	4G EPS AKA	5G AKA	5G EAP AKA'
early Aliveness	✓	✓	✓	✓
early WA	$\neg imsi$	$\neg imsi$	$\neg supi$	$\neg supi$
early NIA on idUE	$\neg imsi$	$\neg imsi$	$\neg supi$	$\neg supi$
early IA on idUE *(const.)*	✗	✗	✗	✗
early NIA on MK	✗	✗	✗	✗
early NIA on idSN	✗	✗	✗	✗

is not able to differentiate a valid AV received from the HN and an invalid AV received from an attacker, $\neg ch$ has to hold for all agreement properties. In addition, 3G AKA and 4G EPS AKA must use a binding channel in order to satisfy non-injective agreement on the UE identity. Otherwise it is possible that the HN confuses two sessions of the SN. This is possible since the channel between the SN and HN is not order preserving. The attack is summarized in Figure 5.4. The HN *hn* receives two attach requests from two different instances of the same SN, *sn.1* and *sn.2*. *hn* computes the AV for the request of *sn.1*, but sends it to *sn.2*. *sn.2* authenticates *ue* but thinks it talks with the UE with identity idUE_{fake}.

In contrast to non-injective agreement on MK, $\neg ch$ is not enough to satisfy injective agreement. $\neg K$ has to hold in addition to $\neg ch$ (or $secretSupi$ in case of 5G EAP AKA'). After listening to a valid protocol run, an attacker can use the same messages to trick a different SN. The attacker fakes both messages from the UE and HN. Since the attacker knows K, she can compute necessary input for the SN in order to persuade the SN to commit on the same MK.

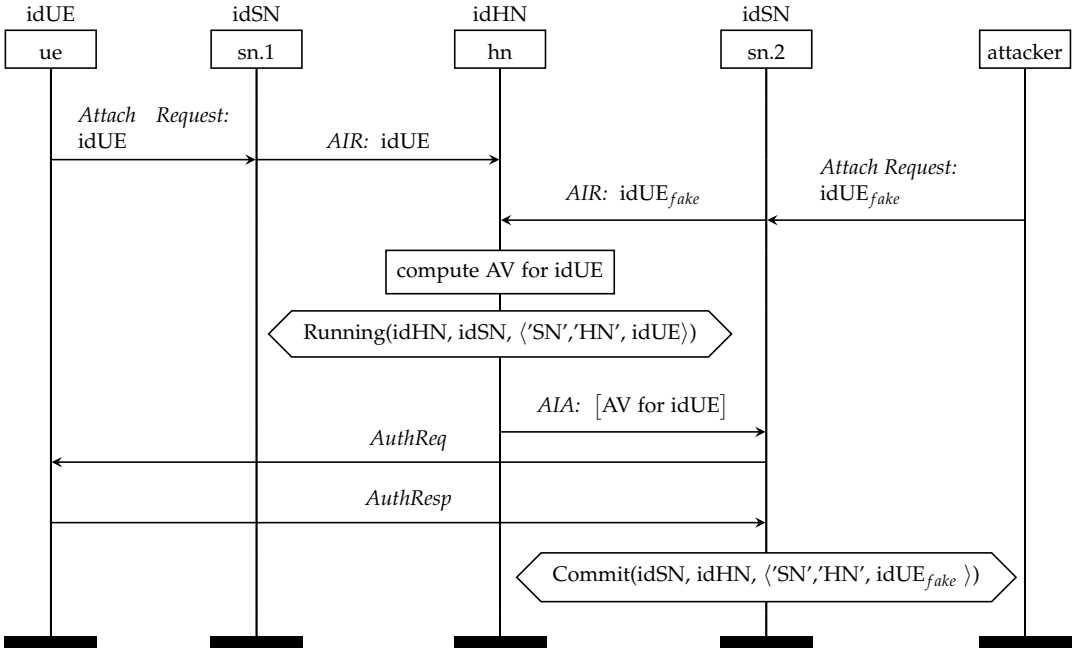

Figure 5.4: Attack on 3G AKA non-injective agreement of SN with HN on the UE identity. No binding channel is used. The requests from *sn.1* and *sn.2* are confused. HN accidentally sends the AV for *ue* to *sn.2* instead of *sn.1*. *sn.2* commits on idUE$_{fake}$, although idUE$_{fake}$ does not exist. The attack for 4G EPS AKA is the same, except that the attach request also includes the identity of the SN.

Table 5.9: Summary of late security properties of the **HN with UE**.

Security Property	5G AKA	5G EAP AKA'
late Aliveness	✓	✓
late WA	$\neg K \vee \neg supi$	$\neg K \vee \neg supi$
late NIA on idUE	$\neg K \vee \neg supi$	$\neg K \vee \neg supi$
late IA on idUE *(const.)*	✗	✗
late IA on MK	$\neg K$	$\neg K \vee secretSupi$
late NIA on idSN	$\neg K$	$\neg K \vee secretSupi$
late IA on idSN *(const.)*	✗	✗

Table 5.10: Summary of agreement properties of the **HN with SN**.

Security Property	3G AKA	4G EPS AKA	5G AKA	5G EAP AKA'
WA	$\neg ch$	$\neg ch$	$\neg ch$	$\neg ch$
NIA on idUE	$\neg ch$	$\neg ch$	✗	✗
IA on idUE *(const.)*	✗	✗	[✗]	[✗]
NIA on MK	✗	✗	✗	✗
NIA on idSN	$\neg ch$	$\neg ch$	$\neg ch$	$\neg ch$
IA on idSN *(const.)*	✗	✗	✗	✗

5.3.5 HN Agreement with UE

From the point of view of the HN we do not distinguish *before* and *after* key confirmation. Key confirmation happens without the inclusion of the HN and only after the HN has sent its last message. Thus, commitConf claims would be at the same place as commit claims. As a result, the same security properties hold for both scenarios. However, for the HN with the UE as partner we distinguish *early* and *late* agreement. Early commit claims are placed after the HN receives the first message from the SN. Late commit claims are placed when the HN receives the second message from the SN, which is only possible for the 5G variants of AKA. Table 5.8 summarizes early agreement properties, whereas Table 5.9 summarizes late agreement properties. The late agreement properties only show 5G AKA and 5G EAP AKA'.

Early agreements on the MK and SN identity trivially fail for all AKA variations. It is not possible to have a running claim at the UE which causally precedes the early commit claim at the HN. In early non-injective agreement on the UE identity, an uncompromised UE identity is sufficient. Early and late aliveness are always satisfied, no matter what compromise scenario occurs. The reason is that the HN only accepts authentication initiation requests which include a UE identity that is subscribed at the HN. Thus, the UE has to be initialized and subscribed at the HN before the HN accepts an authentication initiation request. 5G AKA and 5G EAP AKA' satisfy late injective agreement on the MK and late non-injective agreement on the identities of the UE and SN.

5.3.6 HN Agreement with SN

As already mentioned in the previous section, we do not distinguish before and after key confirmation for the HN. Table 5.10 summarizes the security properties that the HN achieves with the SN as partner. The satisfied properties rely on an uncompromised channel between the SN ans HN. Non-injective agreement on the SN identity is satisfied in all AKA versions. Non-

Table 5.11: Summary of secrecy properties of the **UE**.

Secrecy	3G AKA	4G EPS AKA	5G AKA	5G EAP AKA′
idUE	✗	✗	$secretSupi$	$secretSupi$
MK	✗	$\neg K \wedge \neg ch$	$\neg K \wedge \neg ch$	$(\neg K \wedge \neg ch) \vee secretSupi$
PFS MK	$[\text{✗}]$	✗	✗	✗
K	$\neg K$	$\neg K$	$\neg K$	$\neg K$

Table 5.12: Summary of secrecy properties of the **SN**.

Secrecy	3G AKA	4G EPS AKA	5G AKA	5G EAP AKA′
idUE	✗	✗	$secretSupi$	$secretSupi$
MK	$\neg K \wedge \neg ch$	$\neg K \wedge \neg ch$	$\neg K \wedge \neg ch$	$(\neg K \wedge \neg ch) \vee secretSupi$
PFS MK	✗	✗	✗	✗

Table 5.13: Summary of secrecy properties of the **HN**.

Secrecy	3G AKA	4G EPS AKA	5G AKA	5G EAP AKA′
idUE	✗	✗	$secretSupi$	$secretSupi$
MK	$\neg K \wedge \neg ch$	$\neg K \wedge \neg ch$	$\neg K \wedge \neg ch$	$(\neg K \wedge \neg ch) \vee secretSupi$
PFS MK	✗	✗	✗	✗
K	$\neg K$	$\neg K$	$\neg K$	$\neg K$

injective agreement on the MK trivially fails since it is not possible to have a running claim on the MK at the SN which causally precedes the commit claim of the HN. In the same way non-injective agreement on the UE identity fails for 5G AKA and 5G EAP AKA′. The SN receives the unconcealed UE identity (SUPI) only after the HN has sent its last message. In 3G AKA and 4G EPS AKA, non-injective agreement on the UE identity is satisfied because the UE identity is not concealed.

5.3.7 Secrecy

Table 5.11, 5.12, and 5.13 summarize secrecy properties for each role. Secrecy of the identity of the UE is trivially violated in 3G AKA and 4G EPS AKA for all roles. The attach request contains the identity in clear. This is already mentioned in [7]. In the 5G variants of the AKA protocol the identity of the UE is secret, as long as all channels from and to the HN, where the UE is subscribed, are uncompromised. Obviously the identity of the UE itself and the private key of the HN have to be uncompromised as well. In 3G AKA, secrecy on the MK fails from point of view of the UE. The reason is that the UE does not agree with the SN on the identity of the SN. The UE claims that the SN has be honest, however, since the UE has a non-existing SN identity, the honest claim does not ensure that the SN does not reveal the MK. For

all other versions and roles, secrecy on the MK is satisfied. On the other hand, perfect forward secrecy of the MK is always violated. It is possible to compute the MK if an attacker records the exchanged messages, as well as compromises the long-term secret K. In case of 5G EAP AKA', the identity of the UE has to be compromised as well.

Secrecy lemma using ¬CH

Lemmas that use $\neg CH$ have to know the identity of the HN in order to prevent reveals on a channel from or to the HN. However, the secret claim at the SN only includes the identity of the SN as well as the secret term. The identity of the HN is unknown. In order to tackle this issue, we add a `SecretFrom` claim which includes the SN identity as well as the HN identity. The `SecretFrom` claim is used for secrecy lemmas at the SN that rely on the $\neg CH$ assumption.

5.3.8 Summary

We want to summarize and highlight certain properties we have encountered in this section. First, if a property is satisfied before key confirmation, it is always satisfied after key confirmation. While an extra key confirmation round helps to satisfy more properties for 4G EPS AKA, 5G AKA, and 5G EAP AKA', it does not make a difference in 3G AKA. In this case, all properties that are satisfied after key confirmation are already satisfied before key confirmation. Second, a binding channel does not improve the results for 5G AKA and 5G EAP AKA'. Thus, both versions use the same number of lemmas for both channel types (see Table 5.1). Finally, the sequence number is never included in the minimal assumptions. All properties that are satisfied do not require the sequence number to be uncompromised.

5.4 Complexity and Cost across AKA Variants

This section compares communication complexity and cost across AKA variants. We measure cost based on the number of messages sent, the number of generated nonces, as well as applied key derivation functions, encryptions, and hash functions.

The cost for 4G EPS AKA compared to 3G AKA increases only in the MK derivation. The additional key derivation function binds the SN identity to the MK. Hence, 4G EPS AKA satisfies more security properties compared to 3G AKA which justifies the cost.

The 5G variants of the AKA protocol increase communication complexity and cost compared to the earlier variants. First, two more messages are exchanged between the SN and HN. Second, one additional nonce is used.

Third, one asymmetric encryption and decryption is applied to conceal the UE identity. Finally, the MK derivation gets even more involved than in 4G EPS AKA. However, the 5G variants satisfy more agreement and secrecy properties with mostly weaker assumptions than 3G AKA and 4G EPS AKA, which justifies the increasing communication complexity and cost. 5G AKA applies a more involved derivation of the challenge, whereas 5G EAP AKA' uses the same mechanism as the older AKA variants. From the point of view of security properties, the more complex derivation seems unnecessary since 5G EAP AKA' satisfies more properties with weaker assumptions than 5G AKA. On the other hand, in case the SN receives a wrong response of the UE, the SN is able to cancel authentication in 5G AKA. In 5G EAP AKA', the SN is unable to recognize faulty responses, thus in every case the HN has to verify the response. 5G EAP AKA' attaches an additional MAC to each message, hence no messages can be altered or injected. Additionally, it binds the identities of the UE and SN to the message. The MAC adds computational cost to the HN and UE, however, as we have seen in the previous section, 5G EAP AKA' satisfies the most security properties relying on the weakest assumptions.

5.5 Combined Models

This section presents the security analysis of the combined AKA protocol models. We automatically generate the following model combinations. $A \cup B$ denotes that protocol A and B are combined in one model.

- 3G AKA $\cup$ 4G EPS AKA
- 4G EPS AKA $\cup$ 5G AKA
- 3G AKA $\cup$ 4G EPS AKA $\cup$ 5G AKA

We first introduce properties of 3G AKA $\cup$ 4G EPS AKA, which is followed by a discussion on 4G EPS AKA $\cup$ 5G AKA. The combination of 3G AKA, 4G EPS AKA and 5G AKA does not include new results that are not already contained in the smaller two combinations. Thus, we will not discuss our analysis of 3G AKA $\cup$ 4G EPS AKA $\cup$ 5G AKA. We only cover properties that change when comparing a separate AKA variant to a combined model. To get an overview of a specific AKA variant refer to Section 5.3 or to the Appendix A. Note that an attack that exists in an individual AKA variant exists for that AKA variant in a combined model as well. Thus, other AKA variants are excluded in already found attack traces in order to reduce the search space.

5.5.1 3G AKA ∪ 4G EPS AKA

This section covers changes in agreement properties of 3G AKA and 4G EPS AKA in case both are combined in one model. The agreement and secrecy properties of the following view points are the same in both the separate and combined models. We do not discuss them further.

- UE with HN
- HN with SN
- Secrecy for UE, SN, and HN

UE Agreement with SN

We recall the properties of the AKA variants in a separate model for the UE with the SN: 3G AKA does not satisfy aliveness, hence all other agreement properties are also violated. 4G EPS AKA satisfies weak agreement, non-injective agreement on the UE identity, and injective agreement on the MK. Non-injective agreement on the SN identity is violated.

Changes of agreement properties between separate and the combined model are shown in Table 5.14. The attack on aliveness in 3G AKA is also possible in the combined model. Thus, all properties are violated for 3G AKA in the combined model as well. On the other hand, 3G AKA ∪ 4G EPS AKA satisfies the same properties for 4G EPS AKA, however, a stronger minimal assumption is required to satisfy anonymous (non-)injective agreement on the MK. A binding channel is necessary, which means the assumption is as follows: $B \wedge kc \wedge \neg K \wedge \neg ch$. A non-binding channel does not suffice since the SN does not verify that the AMF separation bit is set according its version. Consequently, the SN does not realize that it talks with the HN that uses the wrong AKA version. A possible attack trace is shown in Figure 5.5. The SN *sn.2* runs on the MK using protocol version 3G AKA, while the UE *ue.2* uses 4G EPS AKA. As a result, *ue.2* does not agree with *sn.2* on the protocol version, although both share the same MK. The attack is possible since the channel between the SN and HN is not order preserving, e.g., it is possible that requests are confused. With the use of a binding channel such an attack is not possible anymore, because answers from the HN are bound to its request. A different solution to prevent such an attack would be to enforce a check on the AMF separation bit at the SN.

SN Agreement with UE

Table 5.15 summarizes changes from the separate models of 3G AKA and 4G EPS AKA to the combined model 3G AKA ∪ 4G EPS AKA. Non anonymous agreement properties are unchanged. On the other hand, anonymous

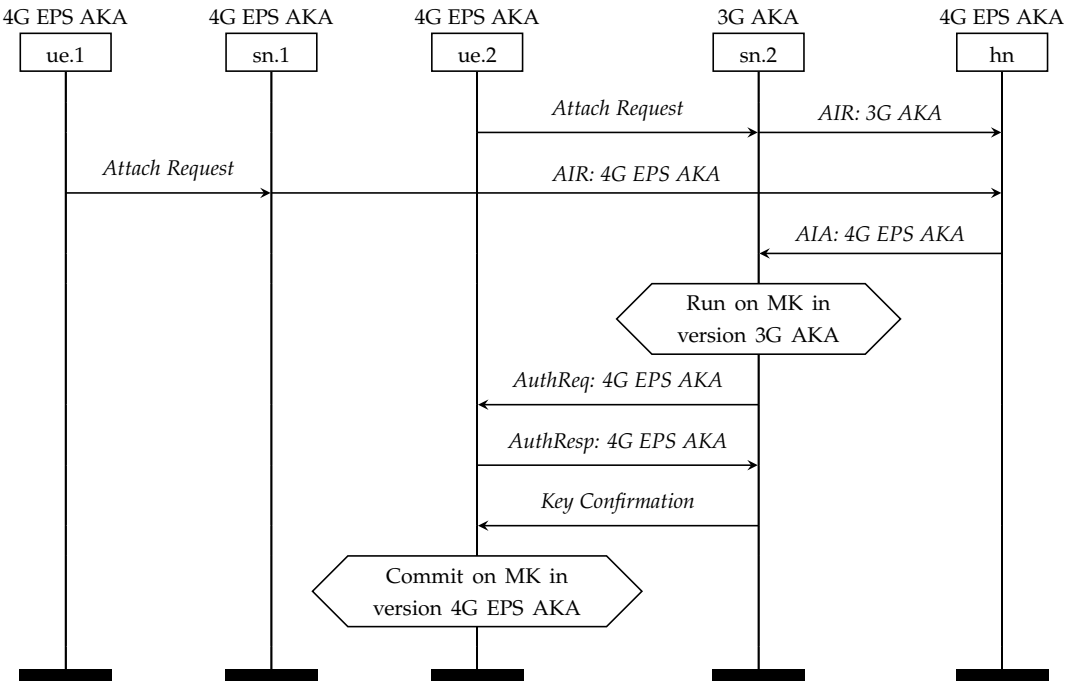

Figure 5.5: Attack trace of the UE with the SN on anonymous non-injective agreement on the MK after key confirmation in the combined model of 3G AKA and 4G EPS AKA. *ue.1* and *ue.2* are an instance of the same UE, whereas *sn.1* and *sn.2* are both an instance of the same SN. From the point of view of *ue.2* and *sn.2* it is a successful protocol run, however, *ue.2* does not agree with *sn.2* on the used protocol version.

Table 5.14: Summary of changes of agreement properties of the **UE with SN** in 3G AKA ∪ 4G EPS AKA. Differences between separate and combined models are marked in red.

	4G EPS AKA	
Security Property	separate	combined
Anonymous NIA on MK	$kc \wedge \neg K \wedge \neg ch$	$B \wedge kc \wedge \neg K \wedge \neg ch$
Anonymous IA on MK	$kc \wedge \neg K \wedge \neg ch$	$B \wedge kc \wedge \neg K \wedge \neg ch$

Table 5.15: Summary of changes of agreement properties of the **SN with UE** in 3G AKA ∪ 4G EPS AKA. Differences between separate and combined models are marked in red.

Security Property	3G AKA / 4G EPS AKA separate	combined
Anonymous WA	$B \wedge \neg ch \wedge (\neg K \vee \neg imsi)$	$B \wedge \neg ch \wedge \neg K$

weak agreement for 3G AKA and 4G EPS AKA requires a stronger minimal assumption in order to be satisfied. It is necessary that the shared secret K is not compromised, i.e., the minimal assumption changes from $B \wedge \neg ch \wedge (\neg K \vee \neg imsi)$ to $B \wedge \neg K \wedge \neg ch$. As already mentioned in the analysis of the individual models, using $B \wedge \neg ch \wedge \neg imsi$ is only valid because weak agreement does not include roles. The running claim actually originates from the HN instead of the UE. By compromising K, an attacker is able to create a valid 3G AKA authentication request from a valid 4G EPS AKA authentication request. As a result, the UE runs using version 3G AKA while the SN commits in version 4G EPS AKA. The attack is the same for 3G AKA, but with the AKA versions exchanged.

SN Agreement with HN

Weak agreement without a binding channel fails for 3G AKA and 4G EPS AKA because the SN does not agree with the HN on the used protocol version. Although a running claim exists, it originates from the wrong version. The attack is possible because the SN does not check if the AMF separation bit is set. The attack on 3G AKA weak agreement is nearly the same shown in Figure 5.5, except that running and commit claims are different. The HN *hn* runs with the SN *sn.2* in version 4G EPS AKA before sending AIA. *sn.2* commits after receiving AIA. As a result, *sn.2* does not agree with *hn* on the protocol version.

In order to satisfy weak agreement, agreement on the MK and SN identity, each minimal assumption has to be joined with B (see Table 5.16). Another solution already mentioned above, would be to enforce the check on the AMF separation bit at the SN.

HN Agreement with UE

Like in the separate models, the combined model 3G AKA ∪ 4G EPS AKA only considers early agreements for 3G AKA and 4G EPS AKA. Only early aliveness is satisfied. Early weak agreement fails because the attach request message originating from the UE is identical in both versions 3G AKA and 4G EPS AKA. The SN is not able to distinguish the protocol versions and forwards the message in the wrong version. Consequently, the HN does not

Table 5.16: Summary of changes of agreement properties of the **SN with HN** in 3G AKA ∪ 4G EPS AKA. Differences between separate and combined models are marked in red.

Security Property	3G AKA / 4G EPS AKA separate	combined
WA	$\neg ch$	$B \wedge \neg ch$
NIA on MK	$\neg ch$	$B \wedge \neg ch$
IA on MK	$\neg K \wedge \neg ch$	$B \wedge \neg K \wedge \neg ch$
NIA on idSN	$\neg ch$	$B \wedge \neg ch$

Table 5.17: Summary of changes of agreement properties of the **HN with UE** in 3G AKA ∪ 4G EPS AKA. Differences between separate and combined models are marked in red.

Security Property	3G AKA / 4G EPS AKA separate	combined
early WA	$\neg imsi$	✗
early NIA on idUE	$\neg imsi$	✗

agree with the UE on the used protocol version. Table 5.17 summarizes the changing agreement properties.

5.5.2 4G EPS AKA ∪ 5G AKA

This section discusses changes in security properties comparing the combination of 4G EPS AKA and 5G AKA to the separate models. We do not discuss the following view points.

- UE with SN
- SN with HN
- HN with SN

The agreement properties are the same in the separate and combined model. This section only covers differences between 4G EPS AKA ∪ 5G AKA to individual models of 4G EPS AKA and 5G AKA.

UE Agreement with HN

Surprisingly, weak agreement fails without an extra key confirmation step for both 4G EPS AKA and 5G AKA. The authentication request message from the SN to the UE are exchangeable. As a result, the UE does not agree with the HN on the used protocol version. And both, the UE and HN, derive different MKs according to their AKA version. While joining kc to the minimum assumption is sufficient for 4G EPS AKA, 5G AKA requires an even stronger minimal assumption, namely $\neg ch$. Figure 5.6 shows a trace

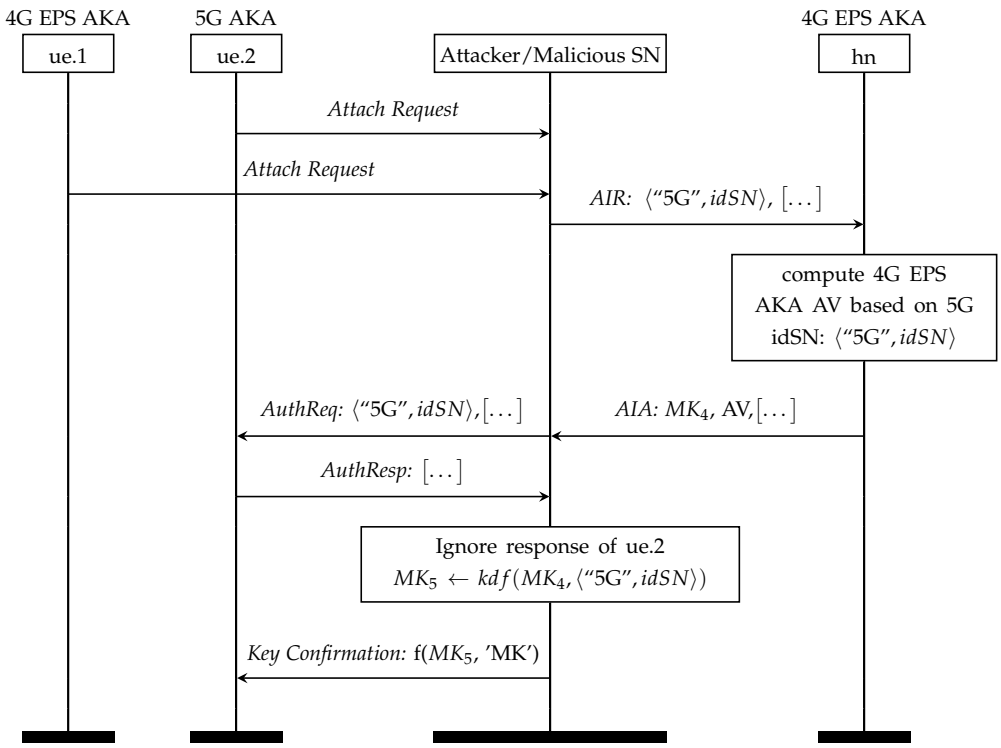

Figure 5.6: Attack on 5G AKA weak agreement of the UE with the HN in combined version of 4G EPS AKA and 5G AKA using only $kc \wedge \neg K$ as assumption. The malicious SN is able to derive MK_5 from MK_4.

that uses $kc \wedge \neg K$ assumption and violates weak agreement for 5G AKA. The issue is that the HN in 4G EPS AKA does not check if the received SN identity is valid, e.g., the SN identity in 4G EPS AKA should not be tagged with "5G". In the attack trace the HN *hn* runs a 4G EPS AKA instance, however it receives a SN identity that is of the form used in 5G AKA. *hn* uses the 5G AKA SN identity to derive MK_4, which actually results in an invalid 4G EPS AKA master key. However, the attacker is able to derive a valid MK_5 from it, which in turn is used in the key confirmation step. Such an attack is not possible in 4G EPS AKA because a 5G AKA HN checks if the received SN identity is of the form $\langle$"5G", $idSN\rangle$. Table 5.18 summarizes all changes of agreement properties between the separate and combined model. (Non-)injective agreement on the MK and identities of the UE and SN requires the same minimal assumption as weak agreement.

SN Agreement with UE

Table 5.19 summarizes the changes of agreement properties of the SN with the UE. The minimal assumption for anonymous weak agreement has to be

Table 5.18: Summary of changes of agreement properties of the **UE with HN** in 4G EPS AKA ∪ 5G AKA. Differences between separate and combined models are marked in red.

	4G EPS AKA		5G AKA	
Security Property	separate	combined	separate	combined
WA	$\neg K$	$kc \wedge \neg K$	$\neg K$	$kc \wedge \neg K \wedge \neg ch$
NIA on idUE	$\neg K$	$kc \wedge \neg K$	$\neg K$	$kc \wedge \neg K \wedge \neg ch$
IA on MK	$\neg K$	$kc \wedge \neg K$	$\neg K$	$kc \wedge \neg K \wedge \neg ch$
NIA on idSN	$kc \wedge \neg K$	$kc \wedge \neg K$	$kc \wedge \neg K$	$kc \wedge \neg K \wedge \neg ch$

Table 5.19: Summary of changes of agreement properties of the **SN with UE** in 4G EPS AKA ∪ 5G AKA. Differences between separate and combined models are marked in red.

	4G EPS AKA	
Security Property	separate	combined
Anonymous WA	$B \wedge \neg ch \wedge (\neg K \vee \neg idUE)$	$B \wedge \neg ch \wedge (\neg K \vee secretSupi)$

stronger for 4G EPS AKA. $B \wedge \neg ch \wedge \neg idUE$ is strengthened to $B \wedge \neg ch \wedge secretSupi$ because otherwise an earlier 5G AKA protocol run leaks the UE identity by compromising the HN private key. In order to keep the UE identity secret until the UE sends an attach request in 4G EPS AKA, we join $secretSupi$ to the minimal assumptions. On the other hand, $B \wedge \neg ch \wedge \neg K$ is also sufficient to satisfy anonymous weak agreement.

HN Agreement with UE

Changes in agreement properties are summarized in Table 5.20. 4G EPS AKA requires $secretSupi$ as minimal assumption in order to satisfy early weak agreement and early non-injective agreement on the identity of the UE. Otherwise a protocol it is possible to compromise the identity of the UE in a previous run in 5G AKA by compromising the private key of the HN. As a result, an attacker eavesdropping is able to forge an attach request in 4G EPS AKA. 5G AKA on the other hand does not satisfy any early agreement except of aliveness. This is due to 4G EPS AKA, which sends the identity of the UE in clear. This is also the reason why $\neg supi$ is not sufficient anymore for late agreement in 5G AKA. $\neg K$ has to be uncompromised. In that way, only the HN and the UE are able to compute challenges and the corresponding response.

Secrecy

Secrecy of the UE identity fails in 5G AKA for all roles, the UE, the SN, and HN if combined with 4G EPS AKA. The attach request of 4G EPS AKA

Table 5.20: Summary of changes of agreement properties of the **HN with UE** in 4G EPS AKA ∪ 5G AKA. Differences between separate and combined models are marked in red.

	4G EPS AKA		5G AKA	
Security Property	separate	combined	separate	combined
early WA	$\neg idUE$	$secretSupi$	$\neg supi$	✗
early NIA on idUE	$\neg idUE$	$secretSupi$	$\neg supi$	✗
late WA	-	-	$\neg K \vee \neg supi$	$\neg K$

Table 5.21: Summary of changes of secrecy properties in 4G EPS AKA ∪ 5G AKA. Differences between separate and combined models are marked in red.

	5G AKA	
Secrecy for the UE, HN and SN	separate	combined
idUE	$secretSupi$	✗

contains the UE identity in clear. After the UE sends an attached request using 4G EPS AKA, the identity of the UE is not secret anymore.

Chapter 6

Related Work

In this chapter we compare our analysis of AKA variants with multiple models introduced in previous work. We recall the limitations of previous models of AKA variants listed in Chapter 1:

1. The protocol is modeled as a stateless protocol.
2. The XOR operator is not modeled.
3. The HN and SN are modeled as one single party.

We refer to the limitations (1.)-(3.) in the following sections. This chapter is structured as follows. In Section 6.1, we compare the results of two formal analyses of 3G AKA with our results. In Section 6.2, we present a comparison of 4G EPS AKA. In Section 6.3, we introduce a detailed comparison of our 5G AKA model and a previous analyzed 5G AKA model. In Section 6.4, we comment on previous work of 5G EAP AKA'.

6.1 3G AKA

6.1.1 Enhanced BAN Logic

3GPP formally analyzes 3G AKA using enhanced BAN logic in [5]. The proofs are conducted manually. According to their proofs, 3G AKA satisfies many security properties. However, the proofs lack accuracy due to the proof methodology with an oversimplified protocol model. The enhanced BAN logic has three main limitations: First, the enhanced BAN logic is not sound [16]. Second, only a weak threat model is considered. The compromise of agents are outside of the logic's scope. Finally, type flaws are not detected. The model analyzed by 3GPP suffers from the limitations (1.) and (2.) presented in the beginning of this chapter.

Our analysis of 3G AKA shows that certain security properties that were proven as satisfied by 3GPP are violated. For instance, 3GPP proves agree-

ment on the established key of the UE and SN. In our model, from the perspective of the UE, aliveness of the SN is violated. Hence, agreement on the established key is violated as well. The attack is described in Section 5.3. We will not enumerate all attacks on the security properties of 3G AKA that were missed by 3GPP because of the known limitations on the BAN logic.

6.1.2 ProVerif

An automatic verification of 3G AKA using ProVerif with focus on subscriber privacy is presented by Arapinis et al. [12]. The model suffers from all limitations presented at the beginning of this chapter, i.e., (1.)-(3.). Instead of using the XOR operator, the SQNs are concealed with the use of randomized symmetric encryption.

Arapinis et al. present two attacks on privacy and anonymity of 3G AKA subscribers. Attacks on privacy of 3G AKA subscribers are not surprising, as the constant UE identity is sent in clear. Arapinis et al. do not discuss agreement properties, and whether they are satisfied. Hence, it is not possible to compare results on agreement properties in detail. Since they model the SN and HN as one party, the model makes it impossible to detect attacks that are based on session confusions. As a result, Arapinis et al. are not able to detect security properties that require a binding channel. For instance, the analysis of our 3G AKA model shows that from the perspective of the SN, aliveness of the UE requires a binding and uncompromised channel.

6.2 4G EPS AKA

Hussain et al. formally analyze the attach, detach, and paging procedures of 4G using ProVerif. They list multiple attacks on 4G EPS AKA that is part of the attach procedure. Our model of 4G EPS AKA finds the same attacks on agreement and secrecy properties. However, attacks resulting in denial of service of any kind are not covered by our models.

6.3 5G AKA

Basin et al. [14] present a detailed analysis of 5G AKA that does not suffer from any of the named limitations. They model 5G AKA as specified in the technical specification TS 33.501 V15.1.0, June 2018 [9], henceforth referred to as `5g-aka-2018`. Our model of 5G AKA is based on the updated specification: TS 33.501 V15.4.0, March 2019 [10]. In this section we will refer to our 5G AKA model as `5g-aka-2019`.

We are able to consistently compare the two models because `5g-aka-2019` builds on `5g-aka-2018` using mostly the same modeling decisions. The

Table 6.1: Summary of changing agreement properties when comparing `5g-aka-2018` to its successor `5g-aka-2019`. Only properties that change are shown. A non-binding channel is used.

Models	`5g-aka-2018`	`5g-aka-2019`
UE with SN		
WA	✗	$kc \land \neg K \land \neg ch$
NIA on idUE	✗	$kc \land \neg K \land \neg ch$
IA on MK	✗	$kc \land \neg K \land \neg ch$
NIA on idSN	✗	$kc \land \neg K \land \neg ch$
SN with UE		
WA	$\neg K \land \neg ch$	$\neg K \land \neg ch$
NIA on MK	✗	$\neg K \land \neg ch$
IA on MK	✗	$\neg K \land \neg ch$
HN with UE		
(late) WA	$\neg K$	$\neg K \lor \neg supi$
(late) NIA on idUE	$\neg K$	$\neg K \lor \neg supi$

comparison is summarized in Table 6.1. The table only includes agreement properties that differ between the two models. Basin et al. point out the missing binding of the UE identity and the MK for the SN. In `5g-aka-2018` the SN receives the MK and the unconcealed UE identity in two different messages. Consequently, the SN has no means to check that the received MK belongs to the UE. As a result, even weak agreement fails for the UE with the SN, and non-injective agreement on the MK is violated for the SN with the UE. To satisfy the same properties that hold in `5g-aka-2019`, Basin et al. introduce a binding channel between the SN and HN. The specification update of 5G AKA combines the MK and unconcealed UE identity in one message to bind the two terms together. Thus, a binding channel is not necessary in `5g-aka-2019`.

For the HN with the UE we differentiate between early and late agreement in `5g-aka-2019`. No such distinction is made in `5g-aka-2018`. However, we can compare late agreement of `5g-aka-2019` to the agreement properties of `5g-aka-2018`. In `5g-aka-2019`, weak agreement and non-injective agreement on the UE identity are satisfied with the assumption $\neg K \lor \neg supi$. In `5g-aka-2018`, $\neg K$ is required to remain uncompromised which is weaker than the minimal assumptions for `5g-aka-2019`.

6.4 5G EAP AKA'

To our knowledge, no previous work does analyze 5G EAP AKA' in detail. Most previous analyses consider 5G AKA and refer to 5G EAP AKA' as a similar protocol with similar properties and gloss over the differences. However, as our analysis shows, 5G EAP AKA' satisfies more security properties than 5G AKA, while requiring weaker assumptions.

Chapter 7

Conclusion

In this thesis, we formally modeled different AKA variants used across multiple mobile network generations based on one generic model. Additionally, we combined multiple protocol variants in one model. We leveraged the Tamarin prover and the help of oracles to automatically prove or disprove all security properties.

In general, we discovered that the AKA protocol variants of newer mobile network generations satisfy more security guarantees than the AKA protocol of older generations. However, also the newest AKA variants, which includes 5G AKA and 5G EAP AKA', satisfy many security properties only after an extra key confirmation round is applied. Key Confirmation is not a part of the actual protocol specification. Ultimately, 5G EAP AKA' satisfies the most security properties using the weakest minimal assumptions. The AKA variants of 5G conceal the subscriber identity to protect privacy of the subscribers. The combined models show that the subscriber identity does not remain secret when the same identity is used across multiple mobile network generations, because earlier generations leak the identity. Some messages do not differ across variants, which may lead to a disagreement on the used protocol variant. As a result, the AKA variants may require stronger assumptions in combined models. For instance, the combination of 3G AKA with 4G EPS AKA requires for many security guarantees stronger assumptions on the channel between the SN and HN. It must be binding.

We considered 3G AKA, 4G EPS AKA, and 5G AKA to generate combined models. We expect future work to combine 5G EAP AKA' with other AKA protocol variants. Especially, the combination of 5G AKA and 5G EAP AKA' is of interest, because the two AKA variants are specified to co-exist. The generation of a combined model using AKA variants already modeled in this thesis should be straightforward, thanks to our automatic instantiation. However, to prove the lemmas and adjust the oracle we expect some additional effort.

Appendix A

Appendix

We present an overview of security properties for each individual AKA protocol variant. Tables A.1-A.4 show agreement properties, whereas Tables A.5-A.8 show secrecy properties. Properties that we did not consider are labeled with –. The remaining notation is introduced in Section 5.1.

Table A.1: Agreement properties of 3G AKA. We only consider anonymous lemmas for the UE with the SN, and vice versa. The lemmas of the HN with the UE are the early agreement properties.

Point of View	UE		SN		HN	
Partner	SN	HN	UE	HN	UE early	SN
Agreement Properties						
Aliveness	✗	$[\neg K]$	$B \wedge \neg ch$	$\neg ch$	✓	$[\neg ch]$
WA	$[\text{✗}]$	$\neg K$	✗	$\neg ch$	$\neg imsi$	$\neg ch$
NIA on idUE	$[\text{✗}]$	$\neg K$	$[\text{✗}]$	$B \wedge \neg ch$	$\neg imsi$	$\neg ch$
IA on idUE *(const.)*	$[\text{✗}]$	✗	$[\text{✗}]$	✗	✗	✗
NIA on MK	$[\text{✗}]$	$\neg K$	$[\text{✗}]$	$\neg ch$	✗	✗
IA on MK	$[\text{✗}]$	$\neg K$	$[\text{✗}]$	$B \wedge \neg ch$	$[\text{✗}]$	$[\text{✗}]$
NIA on idSN	$[\text{✗}]$	✗	$[\text{✗}]$	$\neg K$	✗	$\neg ch$
IA on idSN *(const.)*	$[\text{✗}]$	$[\text{✗}]$	$[\text{✗}]$	✗	$[\text{✗}]$	✗
Anonymous Agreement Properties						
Anonymous WA	✗	–	$B \wedge \neg ch \wedge (\neg K \vee \neg imsi)$	–	–	–
Anonymous NIA on idUE	$[\text{✗}]$	–	$B \wedge \neg K \wedge \neg ch$	–	–	–
Anonymous IA on idUE *(const.)*	$[\text{✗}]$	–	✗	–	–	–
Anonymous NIA on MK	$[\text{✗}]$	–	$B \wedge \neg K \wedge \neg ch$	–	–	–
Anonymous IA on MK	$[\text{✗}]$	–	$B \wedge \neg K \wedge \neg ch$	–	–	–
Anonymous NIA on idSN	$[\text{✗}]$	–	✗	–	–	–
Anonymous IA on idSN *(const.)*	$[\text{✗}]$	–	$[\text{✗}]$	–	–	–

Table A.2: Agreement properties of 4G EPS AKA. We only consider anonymous lemmas for the UE with the SN, and vice versa. The lemmas of the HN with the UE are the early agreement properties.

Point of View	UE		SN		HN	
Partner	SN	HN	UE	HN	UE early	SN
Agreement Properties						
Aliveness	$kc \wedge \neg K \wedge \neg ch$	$[\neg K]$	$B \wedge \neg ch$	$\neg ch$	✓	$[\neg ch]$
WA	$B \wedge kc \wedge \neg K \wedge \neg ch$	$\neg K$	$B \wedge kc \wedge \neg K \wedge \neg ch$	$\neg ch$	$\neg imsi$	$\neg ch$
NIA on idUE	$B \wedge kc \wedge \neg K \wedge \neg ch$	$\neg K$	$B \wedge kc \wedge \neg K \wedge \neg ch$	$B \wedge \neg ch$	$\neg imsi$	$\neg ch$
IA on idUE *(const.)*	✗	✗	✗	✗	✗	✗
NIA on MK	$B \wedge kc \wedge \neg K \wedge \neg ch$	$kc \wedge \neg K$	$B \wedge kc \wedge \neg K \wedge \neg ch$	$\neg ch$	✗	✗
IA on MK	$B \wedge kc \wedge \neg K \wedge \neg ch$	$kc \wedge \neg K$	$B \wedge kc \wedge \neg K \wedge \neg ch$	$\neg K \wedge \neg ch$	[✗]	[✗]
NIA on idSN	✗	$kc \wedge \neg K$	$B \wedge kc \wedge \neg K \wedge \neg ch$	$\neg ch$	✗	$\neg ch$
IA on idSN *(const.)*	[✗]	✗	✗	✗	[✗]	✗
Anonymous Agreement Properties						
Anonymous WA	$kc \wedge \neg K \wedge \neg ch$	–	$B \wedge \neg ch \wedge (\neg K \vee \neg imsi)$	–	–	–
Anonymous NIA on idUE	$B \wedge kc \wedge \neg K \wedge \neg ch$	–	$B \wedge \neg K \wedge \neg ch$	–	–	–
Anonymous IA on idUE *(const.)*	✗	–	✗	–	–	–
Anonymous NIA on MK	$kc \wedge \neg K \wedge \neg ch$	–	$B \wedge kc \wedge \neg K \wedge \neg ch$	–	–	–
Anonymous IA on MK	$kc \wedge \neg K \wedge \neg ch$	–	$B \wedge kc \wedge \neg K \wedge \neg ch$	–	–	–
Anonymous NIA on idSN	✗	–	$B \wedge kc \wedge \neg K \wedge \neg ch$	–	–	–
Anonymous IA on idSN *(const.)*	[✗]	–	✗	–	–	–

Table A.3: Agreement properties of 5G AKA. The column of the HN with the UE is divided in early and late agreement.

Point of View	UE		SN		HN		
Partner	SN	HN	UE	HN	UE		SN
					early	late	
Aliveness	$kc \wedge \neg K \wedge \neg ch$	$[\neg K]$	$\neg ch$	$\neg ch$	✓	✓	$[\neg ch]$
WA	$kc \wedge \neg K \wedge \neg ch$	$\neg K$	$\neg K \wedge \neg ch$	$\neg ch$	$\neg supi$	$\neg K \vee \neg supi$	$\neg ch$
NIA on idUE	$kc \wedge \neg K \wedge \neg ch$	$\neg K$	$\neg K \wedge \neg ch$	$\neg ch$	$\neg supi$	$\neg K \vee \neg supi$	✗
IA on idUE *(const.)*	✗	✗	✗	✗	✗	✗	$[\neg ch]$
NIA on MK	$kc \wedge \neg K \wedge \neg ch$	$kc \wedge \neg K$	$\neg K \wedge \neg ch$	$\neg ch$	✗	$\neg K$	✗
IA on MK	$kc \wedge \neg K \wedge \neg ch$	$kc \wedge \neg K$	$\neg K \wedge \neg ch$	$\neg K \wedge \neg ch$	$[\neg ch]$	$\neg K$	$[\neg ch]$
NIA on idSN	$kc \wedge \neg K \wedge \neg ch$	$kc \wedge \neg K$	$\neg K \wedge \neg ch$	$\neg ch$	✗	$\neg K$	$\neg ch$
IA on idSN *(const.)*	$[\neg ch]$	✗	✗	✗	$[\neg ch]$	✗	✗

Table A.4: Agreement properties of 5G EAP AKA'. The column of the HN with the UE is divided in early and late agreement. $KeyOrSupi$ is short for $(\neg K \wedge \neg ch) \vee secretSupi$.

Point of View	UE		SN		HN		
Partner	SN	HN	UE	HN	UE		SN
					early	late	
Aliveness	$KeyOrSupi$	$[\neg K \vee (\neg supi \wedge \neg sk_{HN})]$	$\neg ch$	$\neg ch$	✓	✓	$[\neg ch]$
WA	$kc \wedge KeyOrSupi$	$\neg K \vee (\neg supi \wedge \neg sk_{HN})$	$KeyOrSupi$	$\neg ch$	$\neg supi$	$\neg K \vee \neg supi$	$\neg ch$
NIA on idUE	$kc \wedge KeyOrSupi$	$\neg K \vee (\neg supi \wedge \neg sk_{HN})$	$KeyOrSupi$	$\neg ch$	$\neg supi$	$\neg K \vee \neg supi$	✗
IA on idUE *(const.)*	✗	✗	✗	✗	✗	✗	[✗]
NIA on MK	$kc \wedge KeyOrSupi$	$\neg K \vee secretSupi$	$KeyOrSupi$	$\neg ch$	✗	$\neg K \vee secretSupi$	✗
IA on MK	$kc \wedge KeyOrSupi$	$\neg K \vee secretSupi$	$KeyOrSupi$	$KeyOrSupi$	[✗]	$\neg K \vee secretSupi$	[✗]
NIA on idSN	$kc \wedge KeyOrSupi$	$\neg K \vee secretSupi$	$KeyOrSupi$	$\neg ch$	✗	$\neg K \vee secretSupi$	$\neg ch$
IA on idSN *(const.)*	✗	✗	✗	✗	[✗]	✗	✗

Table A.5: Secrecy properties of 3G AKA.

Point of View	UE	SN	HN
idUE	✗	✗	✗
MK	✗	$\neg K \wedge \neg ch$	$\neg K \wedge \neg ch$
PFS MK	[✗]	✗	✗
K	$\neg K$	–	$\neg K$

Table A.6: Secrecy properties of 4G EPS AKA.

Point of View	UE	SN	HN
idUE	✗	✗	✗
MK	$\neg K \wedge \neg ch$	$\neg K \wedge \neg ch$	$\neg K \wedge \neg ch$
PFS MK	✗	✗	✗
K	$\neg K$	–	$\neg K$

Table A.7: Secrecy properties of 5G AKA.

Point of View	UE	SN	HN
idUE	$secretSupi$	$secretSupi$	$secretSupi$
MK	$\neg K \wedge \neg ch$	$\neg K \wedge \neg ch$	$\neg K \wedge \neg ch$
PFS MK	✗	✗	✗
K	$\neg K$	–	$\neg K$

Table A.8: Secrecy properties of 5G EAP AKA'.

Point of View	UE	SN	HN
idUE	$secretSupi$	$secretSupi$	$secretSupi$
MK	$(\neg K \wedge \neg ch) \vee secretSupi$	$(\neg K \wedge \neg ch) \vee secretSupi$	$(\neg K \wedge \neg ch) \vee secretSupi$
PFS MK	✗	✗	✗
K	$\neg K$	–	$\neg K$

Bibliography

[1] Goodbye 2G – hello 5G. https://www.swisscom.ch/en/about/company/portrait/network/2g-phase-out.html, accessed on 4.9.2019.

[2] Tamarin Manual. https://tamarin-prover.github.io/manual/index.html, accessed on 4.9.2019.

[3] Tamarin Models of AKA variants. https://www.ethz.ch/content/dam/ethz/special-interest/infk/inst-infsec/information-security-group-dam/research/software/aka-gens-rellstab.zip.

[4] Third Generation Partnership Project (3GPP). http://www.3gpp.org/, accessed on 4.9.2019.

[5] 3GPP. 3G Security: Formal Analysis of the 3G Authentication Protocol. TS 33.902, v4.0.0, 2001.

[6] 3GPP. 3G Security: Cryptographic algorithm requirements. TS 33.105, v15.0.0, 2018.

[7] 3GPP. 3G Security: Security architecture. TS 33.102, v15.1.0, 2018.

[8] 3GPP. Generic Authentication Architecture; Generic Bootstrapping Architecture. TS 33.220, v15.4.0, 2018.

[9] 3GPP. Security architecture and procedures for 5G system. TS 33.501, v15.1.0, 2018.

[10] 3GPP. Security architecture and procedures for 5G system. TS 33.501, v15.4.0, 2019.

[11] 3GPP. System Architecture Evolution (SAE): Security architecture. TS 33.401, v15.7.0, 2019.

[12] Myrto Arapinis, Loretta Mancini, Eike Ritter, Mark Ryan, Nico Golde, Kevin Redon, and Ravishankar Borgaonkar. New Privacy Issues in Mobile Telephony: Fix and Verification. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 205–216, New York, NY, USA, 2012. ACM.

[13] Jari Arkko, Vesa Lehtovirta, and Pasi Eronen. Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA'), 2009.

[14] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. A Formal Analysis of 5G Authentication. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 1383–1396, New York, NY, USA, 2018. ACM.

[15] Bruno Blanchet. Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif. *Found. Trends Priv. Secur.*, 1(1-2):1–135, October 2016.

[16] Colin Boyd and Wenbo Mao. On a Limitation of BAN Logic. In *Advances in Cryptology — EUROCRYPT '93*, pages 240–247, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[17] Cas J. F. Cremers and Martin Dehnel-Wild. Component-Based Formal Analysis of 5G-AKA: Channel Assumptions and Session Confusion. In *NDSS*, 2019.

[18] D. Dolev and A. C. Yao. On the Security of Public Key Protocols. In *Proceedings of the 22Nd Annual Symposium on Foundations of Computer Science*, SFCS '81, pages 350–357, Washington, DC, USA, 1981. IEEE Computer Society.

[19] Syed Hussain, Omar Chowdhury, Shagufta Mehnaz, and Elisa Bertino. LTEInspector: A Systematic Approach for Adversarial Testing of 4G LTE. 01 2018.

[20] Gavin Lowe. A Hierarchy of Authentication Specifications. In *Proceedings of the 10th IEEE Workshop on Computer Security Foundations*, CSFW '97, pages 31–, Washington, DC, USA, 1997. IEEE Computer Society.

[21] Simon Meier. *Advancing automated security protocol verification*. PhD, ETH Zurich, 2013.

[22] Benedikt Schmidt. *Formal analysis of key exchange protocols and physical protocols*. PhD, ETH Zurich, 2012.

[23] Vincent Stettler. *Formally Analyzing the TLS 1.3 proposal*. Bachelor Thesis, ETH Zurich, 2016.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Formalizing and Verifying Generations of AKA Protocols

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Rellstab	Angela

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zürich, 24.09.2019	A. Rellstab

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.