Provably-Secure Enhancement on 3GPP Authentication and Key Agreement Protocol

Muxiang Zhang
Verizon Laboratories
40 Sylvan Road, Waltham, MA 02451
muxiang.zhang@verizon.com

Abstract

This paper analyses the authentication and key agreement protocol adopted by Universal Mobile Telecommunication System (UMTS), an emerging standard for third generation (3G) wireless communications. The protocol, known as *3GPP AKA*, is based on the security framework of GSM and provides significant enhancement to address and correct real and perceived weaknesses in GSM and other wireless communication systems. In this paper, we show that 3GPP AKA is vulnerable to a variant of false base station attack. The vulnerability allows an adversary to re-direct user traffic to an unintended network. It also allows an adversary to use authentication vectors obtained from a corrupted network to impersonate all other networks. In addition, we show that the need of synchronization between a mobile station and its home network incurs considerable difficulty for the normal operation of 3GPP AKA. To provide further enhancement on 3GPP AKA, we present an authentication and key agreement protocol which defeats re-direction attack and drastically lowers the impact of network corruption. The proposed protocol also eliminates synchronization between a mobile station and its home network. Following the multi-party simulatability approach, we have developed a formal model of security for symmetric-key based authentication and key agreement protocols in the mobile setting. Within this model, we have analyzed the security of our protocol against a powerful adversary having full control of the communication channels between a user and a network.

1 Introduction

The movement towards ubiquitous wireless networking has brought about a number of security concerns among service providers and end users. The radio interface and the access to wireless service are two areas where wireless networks do not provide the same level of protection as wired networks unless additional security measures are taken. Two basic threats include interception of data on the radio interface and illegitimate access to wireless service. The interception of user data may result in loss of confidentiality of sensitive user information. The illegitimate use of service is not only of concern with respect to proper billing, but also of concern with respect to masquerading: impersonating a network operator or service provider to intercept user data on the radio interface.

Security issues were not properly addressed in the first-generation (1G) analogue systems. With low-cost equipment, an intruder could eavesdrop user traffic or even change the identity of mobile phones to gain fraudulent service. Given this background, security measures were taken into account in the design of second-generation (2G) digital cellular systems. The Global System for Mobile (GSM) communications was designed from the beginning with security in mind and has adopted several mechanisms [19] to provide user authentication and user data confidentiality. To prevent fraudulent use of wireless service, the GSM network authenticates the identity of a user through a challenge-response mechanism, that is, the user proves its identity by providing a response to a time-variant challenge raised by the network. When the user roams into a foreign network, the home network transfers a set of authentication data (called triplets) to the foreign network. Based on each triplet, the foreign network can authenticate the user without the involvement of the home network.

The GSM authentication and key agreement (hereafter called *GSM AKA*) is simple and has merits in several aspects. First of all, the cryptographic processing is confined to the mobile station and the home network. The serving network does not require the authentication key to compute cryptographic response and cipher key. This helps to minimize the trust that the home network needs to put on the serving network. Secondly, the home network can select its own algorithms used in the challenge-response protocol; the user only needs to implement those algorithms used by his home network. Thirdly, the home network is not on-line involved in every authentication process in the serving network. This lightens the burden on the home network and reduces the overhead caused by the interactions between the serving network and the home network. Nevertheless, weaknesses of the GSM challenge-response protocol have been uncovered over time [22, 23, 20]. Above all, authentication is only unidirectional; the user can not authenticate the serving network. The lack of authentication of serving network allows the so-called false base station attack [25]. Furthermore, triplets can be re-used indefinitely. There is no assurance provided to the user that authentication information and cipher keys are not being re-used.

To address and correct real and perceived security weaknesses in GSM and other 2G systems, the Universal Mobile Telecommunication System (UMTS), an emerging standard for third generation *3G*) wireless communications, has adopted an enhanced authentication and key agreement protocol resulted from the *Third Generation Partnership Project* (*3GPP*) [1]. The protocol, known as *3GPP AKA*, retains the framework of GSM AKA and provides significant enhancement to achieve additional goals such as mutual authentication, agreement on an integrity key between and user and the serving network, and freshness assurance of agreed cipher key and integrity key. As in GSM AKA, the serving network authenticates the user by using authentication data (called authentication vectors) transferred from the user's home network. In each authentication vector, a sequence number is included, which is verified by the user to achieve freshness assurance of agreed cipher and integrity keys. To facilitate sequence number generation and verification, two counters are maintained for each user: one in the mobile station and another one in the home network. Normally, the counter in the mobile station has a value less than or equal to the counter in the home network. When a mismatch occurs between the two counters, which may be caused by a failure in the home network, authentication vectors generated by the home network may not be acceptable by the mobile station. Such a phenomenon is called loss of synchronization and re-synchronization is needed to adjust the counter in the home network.

The 3GPP authentication and key agreement protocol has been scrutinized widely within wireless communications industry. To data, no serious flaw has ever been reported in public literature. Using a formal method known as BAN logic [16], the designers have claimed [2] that "the goals of the protocol as they are stated in ... are met by the protocol". In this paper, we show that 3GPP AKA is vulnerable to a variant of false base station attack. The flaw of 3GPP AKA allows an adversary to re-direct user traffic to an unintended network. It also allows an adversary to use authentication vectors obtained from a corrupted network to impersonate other networks; the corruption of one network jeopardizes the entire system. The re-direction attack represents a real threat since the security levels provided by different networks are not always the same. The re-direction attack could also cause billing problem as service rates offered by different networks are not always the same, either. In addition, the need of synchronization between a mobile station and its home network incurs considerable difficulty for the normal operation of the protocol.

To provide further enhancement on 3GPP AKA, we present an authentication and key agreement protocol which defeats re-direction attack and drastically lowers the impact of network corruption. The protocol, called AP-AKA, also eliminates synchronization between a mobile station and its home network. In AP-AKA, the home network does not maintain dynamic states for each individual subscriber. The user can verify whether authentication vectors were indeed ordered by a serving network and were not used before by the serving network. Following the multi-party simulatability approach [5], we have developed a formal model of security for symmetric-key based authentication and key agreement protocols in the mobile setting. Our model is adapted from Shoup's [29] formal model of security for authenticated key exchange in the three-party setting. For our interest, we extend Shoup's model to the mobile setting and formulate the definition of authentication. Within our formal model of security, we have analyzed the security of AP-AKA against a powerful adversary having full control of the communication channels between a user and a network.

This paper is organized as follows. Section 2 specifies the operation of 3GPP AKA and describes the re-synchronization process. In Section 3, we present two types of attacks against 3GPP AKA and discuss operational difficulty involved with sequence number management. Section 4 presents the protocol AP-AKA and specifies its operation in various environment. In Section 5, we analyze the security of AP-AKA against re-direction attack and examine the impact of network corruption. We also give an overview of provable security for AP-AKA and provide comparison with related protocols. Section 6 concludes the paper.

2 Description of 3GPP AKA

In 3GPP AKA, each user U, represented by a mobile station (MS), shares a secret key K and certain cryptographic algorithms with his home network, denoted by HN. In addition, the home network HN maintains a counter SQN_{HN} for each individual subscriber, and the user U maintains a counter SQN_U for himself. The initial values for SQN_{HN} and SQN_U are zero. The cryptographic algorithms shared between HN and U include three message authentication codes $f1, f1^*, f2$ and four key generation functions $f3, f4, f5, f5^*$. For generic requirements as well as an example algorithm set for these cryptographic algorithms, refer to [3].

There are three goals for 3GPP AKA: i) mutual authentication between the user and the network; ii) establishment of a cipher key and an integrity key upon successful authentication; and iii) freshness assurance to the user of the established cipher and integrity keys. To achieve these goals, 3GPP AKA combines a challenge-response protocol identical to the GSM authentication and key agreement protocol and a sequence number-based one-pass protocol derived from the ISO standard ISO/IEC 9798-4 [21]. 3GPP AKA consists of two phases. The first phase specifies the distribution of authentication data, called authentication vectors, from the home network to the serving network. The second phase specifies the authentication and key agreement procedure between the user and the serving network. When the protocol is executed in the home network or when the serving network has unused authentication vectors for the user, the first phase is not executed. For a concise description of the two phases, let's assume that a user U roams into a foreign network SN which does not have authentication vectors for the user. In this scenario, the protocol can be described as follows:

1.	$SN \to HN$:	*authentication data request*, ID_U
2.	$HN \to SN$:	*authentication data response*, $\{(RAND, XRES, CK, IK, AUTH), \ldots\}$ where $XRES = f2_K(RAND)$, $CK = f3_K(RAND)$, $IK = f4_K(RAND)$, $AUTN = SQN \oplus AK \| AMF \| MAC$, $AK = f5_K(RAND)$, $MAC = f1_K(SQN \| RAND \| AMF)$
3.	$SN \to U$:	*user authentication request*, $RAND, AUTH$
4.	$U \to SN$:	*user authentication response*, $RES = f2_K(RAND)$, or *user authentication reject*, $CAUSE$

The serving network SN starts the protocol by sending *authentication data request* to the home network HN, where *authentication data request* is the *type* of the message. Upon receipt of the message, HN sends back a batch of authentication vectors. For each authentication vector, HN generates a random number, $RAND$, and a sequence number, SQN, from the counter SQN_{HN} maintained for U. Then HN computes an expected response, RES, a cipher key, CK, an integrity key, IK, an authentication token, $AUTH$, and increases the counter SQN_{HN} by 1. Each authentication vector also includes an authentication and key management field *AMF*, which serves to define operator-specific options in the authentication process, e.g., the use of multiple authentication algorithms or a limitation of key lifetime.

After receiving the authentication vectors from HN, SN selects one of them and stores the rest in its database. Then SN sends to the user $RAND$ and $AUTN$ from the selected authentication vector. The user computes AK and retrieves SQN from the received $AUTH$. Next, the user verifies the correctness of the MAC included in $AUTH$. If the verification fails, the user rejects. Otherwise, the user further verifies if the sequence number SQN is in the correct range, i.e., $SQN > SQN_U$. If not, the user sends a *synchronization failure* message to SN and rejects. Otherwise, the user sets SQN_U to SQN, sends a response back to SN, and accepts.

Re-synchronization. A *synchronization failure* message includes a re-synchronization token $AUTS$, which has the following form

$$AUTS = Conc(SQN_{MS}) \| MAC - S,$$

where

$$Conc(SQN_{MS}) = SQN_{MS} \oplus f5^*_k(RAND),$$

and

$$MAC - S = f1^*_k(SQN_{MS}, RAND, AMF).$$

Upon receipt of the *synchronization failure* message from the user, SN sends $RAND$ and $AUTS$ to the home network with an indication of *synchronization failure*.

Upon receiving the *synchronization failure* indication, HN retrieves SQN_{MS} and then verifies whether the value of SQN_{MS} mandates that SQN_{HN} needs to be changed, i.e., $SQN_{HN} < SQN_{MS}$. If necessary, HN also verifies the re-synchronization token $AUTS$ and sets SQN_{HN} equal to SQN_{MS}. Subsequently, HN sends a new batch of authentication vectors to SN. When SN receives a new batch of authentication vectors, it deletes the old authentication vectors stored for the user.

3 Analysis of 3GPP-AKA

Through sequence numbers, the user is ensured that authentication information (i.e., $RAND$ and $AUTH$) can not be re-used by an adversary or by a serving network. The serving network authenticates the user by verifying the user authentication response, RES, to determine if the user has knowledge of the authentication key. The user, however, can only verify if an authentication vector was indeed generated by the home network. The user can not determine if an authentication vector was ordered by a specific serving network since the authentication vector could be ordered by any serving network. This fosters attacks as described below.

3.1 Re-direction Attack

Assume that a user U is in the territory of his home network HN and intends to establish a communication session with the home network. Also Assume that an adversary is operating a device having the functionality of a base station. Such a device is called a false base station. Once the adversary intercepts the connection attempt from the user, the adversary entices the user to camp on the radio channels of the false base station. Then the adversary sends a connection request to a foreign network SN on behalf of the user. Next, the adversary faithfully delivers message between he user and the foreign network. Authentication will be successful both in the user side and in the foreign network and subsequent communication will be protected via the established keys. In this way, the adversary can re-direct user traffic to an unintended network.

The re-direction attack may seem apparent since network identity is not included in the protocol. However, it can be proved (e.g., in Shoup's formal model of security [29]) that 3GPP AKA is a secure protocol in the two-party setting, although not secure in the mobile setting. It is worth pointing out that the re-direction attack has practical implications. According to [1], data encryption is not mandatory in every network, while data integrity is mandatory in all networks. To intercept user traffic, the adversary may re-direct user traffic to a network in which data encryption is either not provided or provided but very weak. The threat of this attack is particularly evident as the user roams to the border of two different networks. It might be argued that the risk could be mitigated if all the networks "trust" each other and use a commonly agreed strong encryption algorithm. Nevertheless, the re-direction attack could cause billing problem; the user is in the territory of his home network but gets charged by a foreign network based on a rate higher than that offered by the home network.

3.2 Active Attack in Corrupted Network

In 3GPP AKA, authentication vectors are transferred between and within networks. Each network is operated under a different administration. When a network is corrupted, an adversary could forge an authentication data request from the corrupted network to obtain authentication vectors for any user, independent of the actual location of the user. Then the adversary could use the obtained authentication vectors to impersonate uncorrupted networks and to mount false base station attack against legitimate users. In addition, by flooding authentication data requests to the home network, the adversary could force the counter SQN_{HN} maintained for a subscriber to be set to a high value. As the maximum value of SQN_{HN} is limited, this shortens the lifetime of the mobile station.

Since the corruption of one network jeopardizes the entire system, it is critical that security measures are in place in every network. Although mechanisms are currently being developed to provide security between and within operators' networks, it is unlikely that network wide security will be implemented in every operator's network at the same time. Situations where network operators A and B have reached agreement on the deployment of network wide security, but A and C have not, may persist for a long time. This fosters both passive and active attacks in those unprotected networks and impedes the normal operation of 3GPP AKA.

3.3 Operational Difficulty with Sequence Numbers

Sequence numbers are generated based on a counter SQN_{HN} maintained in the home network and verified based on another counter SQN_{MS} maintained in the mobile station. When a mismatch happens between the two counters, i.e., $SQN_{HN} < SQN_{MS}$, which may be caused by a failure in the home network, authentication vectors generated by the home network will be rejected by the user. In such a circumstance, re-synchronization is initiated to adjust the value of SQN_{HN}. In 3GPP AKA, re-synchronization is requested by the user, not by the home network. Whenever a sequence number is considered to be not in the correct range, the user decides that a synchronization failure has occurred in the home network and consequently initiates a re-synchronization request to the home network. This may produce spurious re-synchronization requests, as the fact that a sequence number is not in the correct range does not necessarily mean a failure in the counter SQN_{HN}. It might be caused by an adversary by replaying a pair of used $RAND$ and $AUTH$. Moreover, the out-of-order use of authentication vectors in the serving network could also cause synchronization failure. The user, however, cannot discern the actual reason for the sequence number being not in the correct range. Spurious re-synchronization adds extra cost to signaling and may cause deletion of unused authentication vectors.

4 Specification of AP-AKA

By adding sequence numbers to GSM AKA, 3GPP AKA intends to achieve mutual authentication between a user and his home network, not mutual authentication between a user and a foreign network. The assumption might be that each foreign network is trusted by the home network and will not impersonate other networks. This assumption, however, does not apply to an adversary. As shown in Section 3, the adversary could perform both re-direction attack and network corruption attack. In addition, synchronization between a user and his home network imposes considerable difficulty for the normal operation of 3GPP AKA.

In this section, we present an authentication and key agreement protocol (called AP-AKA) which addresses both security and operational issues involved with 3GPP AKA. The protocol AP-AKA retains the framework of 3GPP AKA, but eliminates synchronization between a user and his home network. In AP-AKA, the home network does not maintain dynamic states, e.g., counters, for each individual subscriber. The user can verify whether authentication vectors were ordered by a serving network and were not used before by the serving network. This helps to defeat re-direction attack. It also helps to lower the impact of network corruption. We specify the operation of AP-AKA in this section and analyze its security in next section.

Assume that each user and his home network share a secret key K of length k. Also assume that each user and his home network share three cryptographic algorithms F, G and H, where F and H are message authentication codes, G is a key generation function indexed by K. An overview of AP-AKA is described as follows:

1. $SN \to U$: *user data request*, $FRESH$
2. $U \to SN$: *user data response*, $RN, U_{MAC} = F_K(FRESH||RN||ID_{SN})$
3. $SN \to HN$: *authentication data request*, $ID_U, FRESH, RN, U_{MAC}$
4. $HN \to SN$: *authentication data response*, $\{(RAND, XRES, SK, AUTH), \ldots\}$, where $XRES = F_K(RAND), SK = G_K(RAND)$, $AUTH = idx \| RN_{idx} \| MAC$, $idx \geq 1$, $RN_{idx} = H_K(idx||RN), MAC = F_K(RAND||idx||RN_{idx})$.
5. $SN \to U$: *user authentication request*, $RAND, AUTH$
6. $U \to SN$: *user authentication response*, $RES = F_K(RAND)$

The protocol AP-AKA specifies a sequence of six flows. Each flow defines a message *type* and *format* sent or received by an entity. How the flows are actually carried out and under what conditions entities accept or reject are dependent on the execution environment. In the following, we specify the execution of AP-AKA in various scenarios.

Protocol execution in SN

When a user U roams into a foreign network SN, the protocol execution may be carried out in two different ways.

- If SN has unused authentication vectors for the user, it starts the protocol by executing the fifth flow, i.e., sending *user authentication request* to the user, including $RAND$ and $AUTH$ from the selected authentication vector. After receiving *user authentication response*, SN compares if $RES = XRES$. If not, SN rejects. Otherwise, SN computes the session key SK and accepts.

- If SN does not have unused authentication vectors for the user, then all the six flows will be carried out. SN starts by sending a random number, $FRESH$, to the user. After receiving *user data response*, SN requests authentication data from the home network HN. Next, HN verifies the correctness of the received U_{MAC}. If the verification fails, HN sends back a *reject notice* including $ID_U, FRESH$, and RN, where RN is the random number sent by the user. Upon receiving the notice, SN rejects. If the verification succeeds, HN sends back a batch of authentication vectors to SN. For each authentication vector, HN generates a random number, $RAND$, and computes an expected response, $XRES$, a session key, SK, and an authentication token $AUTH$, where SK may be a concatenation of a cipher key and an integrity key. Each authentication vector is indexed by an integer $idx = 1, 2, \ldots, m$, where m is the number of authentication vectors in the batch. After receiving authentication vectors from HN, SN proceeds as in the previous case.

User actions

The user U acts as a *responder* in the protocol. When the user receives a *user data request*, it generates a random number, RN, send replies back with RN and U_{MAC}. When the user receives a *user authentication request*, it retrieves $RAND$ and $AUTN$ from the request and verifies the correctness of the MAC included in $AUTN$. If the verification fails, the user rejects. Otherwise, the user further verifies if the RN_{idx} included in $AUTH$ is acceptable, that is, $RN_{idx} = H_k(idx||RN)$ and RN_{idx} was not used before by SN. If RN_{idx} is not acceptable, the user rejects. Otherwise the user computes the session key SK and proceeds as follows:

- If $idx > 0$, the user updates its internal state, replies back with RES, and then accepts.

- If $idx = 0$, the user updates its internal state and accepts.

To facilitate fast verification of RN_{idx}, the user maintains usage information on RN_{idx}. The usage information is used to verify that the authentication vector containing RN_{idx} was indeed ordered by the serving network and was not used before by the serving network. This helps to prevent re-direction attack. It also helps to defeat replay attack exploiting previously used authentication vectors.

Protocol execution in HN

If there are unused authentication vectors for the user, the protocol execution is perform in the same way as in the foreign network. Otherwise, HN executes a three-flow protocol as described below:

1. $HN \rightarrow U$: *user data request*, $FRESH$
2. $U \rightarrow HN$: *user data response*, $RN, U_{MAC} = F_K(FRESH||RN||ID_{HN})$
3. $HN \rightarrow U$: *user authentication request*, $RAND, AUTH$,
   where $AUTH = 0\|RN_0\|F_K(RAND||0||RN_0), RN_0 = H_K(0||RN)$.

In the three-flow protocol, HN verifies the correctness of U_{MAC}. If the verification fails, HN rejects. Otherwise, HN accepts. In the case of acceptance, HN may also generate a batch of authentication vectors for future use.

Remark 1. When HN receives an *authentication data request* containing an incorrect U_{MAC}, HN may simply drop the request and does not send back a *reject notice* to SN. SN will automatically reject if it does not receive *authentication data response* within certain mount of time. In a protocol execution, SN may only order authentication vectors for the user and not perform the authentication procedure. E.g., at the end of a communication session, SN may start the protocol again to order a new batch of authentication vectors for future use.

Remark 2. The protocol AP-AKA may seem more efficient if it is started by the user. For instance, when the user initiates a connection request to SN, the user also includes a random number RN and a message authentication code U'_{MAC} in the connection request, where $U'_{MAC} = F_K(RN, ID_{SN})$. Then the serving network SN requests authentication vectors by sending ID_U, RN, and U'_{MAC} to HN. This modification, however, makes it difficult for the network to re-authenticate the user, since the network can not order new authentication vectors during a communication session. Moreover, the modified protocol is susceptible to a denial-of-service attack, that is, an adversary impersonates the user and starts the protocol by sending a used pair (RN, U'_{MAC}) to SN. In subsequent protocol execution, the user will reject a newly ordered authentication vector by SN since RN_{idx} included in the authentication vector was used before. The denial-of-service attack causes considerable overhead both on HN and on SN and impedes the normal operation of the modified protocol.

Verification of RN_{idx}

In AP-AKA, each number RN_{idx} can only be used once. For convenience, we refer to each number RN_{idx} as a nonce. To support fast verification of RN_{idx}, the user maintains a list of unused nonces for every visited network. Each list is pointed by the identity of the network. Upon receiving *user data request* from a network N_i, the user replies back with RN and U_{MAC}. Then the user computes a sequence of nonces $RN_r = H_K(r, RN)$, $r = 0, 1, \ldots, m$, and adds the computed nonces to the list pointed by ID_{N_i}. When the user verifies a nonce RN_{idx} received from the network N_i, the user only needs to check if RN_{idx} is in the list pointed by ID_{N_i}. If not, RN_{idx} is not acceptable; otherwise, it is acceptable. In the case of acceptance, the user removes RN_{idx} from the list. To counter against denial-of-service attack as described in Remark 2, the user may re-send the random number RN for a newly received *user data request* from N_i until an authentication vector is accepted based on RN.

From the above description, we see that the protocol execution in HN is different than the protocol execution in SN. The purpose is to make AP-AKA efficient both in the home network and in foreign networks. Note that protocol executions in both types of networks could be made identical by slightly modifying the protocol execution in HN, that is, HN sends *user data request*, the user replies with *user data response*, HN then sends *user authentication request* and the user replies with *user authentication response*. This arrangement, however, needs to carry out four flows and is less efficient than the three-flow protocol. Also note that the home network does not have to generate authentication vectors for its subscribers. As the use of authentication vectors requires only two flows, the home network may decide to generate authentication vectors for those frequently visiting subscribers.

Assume that the home network generates a batch of m authentication vectors in response to every *authentication data request*. Also assume that all the authentication vectors will be used. In a foreign network, the average number of flows carried out in each protocol execution is $\eta_{SN} = (6 + 2(m-1))/m = 2 + 4/m$. E.g., $\eta_{SN} = 2.8$ when $m = 5$. In the home network, $\eta_{HN} = 3$ if the home network does not generate authentication vectors; otherwise, $\eta_{HN} = (3 + 2m)/(m+1)$. For 3GPP AKA, $\eta_{SN} = (4 + 2(m-1))/m = 2 + 2/m$ and $\eta_{HN} = 2$. So AP-AKA is slightly costly in traffic than 3GPP AKA.

5 Analysis of AP-AKA

In AP-AKA, the network authenticates the user either by verifying U_{MAC} included in *user data response* or by verifying RES included in *user authentication response*. The user authenticates the network through the MAC included in *user authentication request*. If the verification succeeds, the user in ensured that the network is either the home network or a foreign network authorized by the home network to provide him service. In addition, by verifying the RN_{idx} included in $AUTH$, the user is assured that the authentication vector was ordered by the serving network and was not used before by the serving network. In the following, we analyze the security of AP-AKA against re-direction attack and examine the impact of network corruption. We also give an overview of provable security for APA-AKA and provide comparison with other wireless security protocols.

5.1 Re-direction Attack

Whenever a serving network SN requests authentication vectors for a user, the serving network sends a random number, $FRESH$, to the user and also asks the user to provide another random number, RN. Together with the random number RN, the user sends back a message authentication code U_{MAC} providing integrity for $FRESH, RN$, and ID_{SN}. In addition, the user maintains a record (RN, ID_{SN}) in its data base. We assume that the user has a good random number generator such that value of RN will never repeat. By verifying U_{MAC}, the home network is ensured that the user is indeed in the territory of SN. When generating authentications vectors, the home network inserts a number RN_{idx}, which is derived from RN and an index, into each authentication vector. After receiving a user authentication request, the user can determine if the request is sent by the serving network SN, not by other serving networks, since the user can verify if the number RN_{idx} included in the request can be derived from the random number RN sent to the serving network SN. Thus, the protocol AP-AKA defeats re-direction attack.

5.2 Impact of Network Corruption

For 3GPP AKA, we showed that the corruption of one network (either a foreign network of a home network) affects the security of the entire system. We now examine the impact of network corruption on AP-AKA. Assume that every network has established a communication channel with every other network, that is, communications between two networks go through a dedicated circuit.

Assume that a serving network SN is corrupted. The adversary can eavesdrop any message sent or received by SN. In addition, the adversary can concoct a message and sends it to any other network through SN. Since the adversary can obtain authentication vectors generated or received by SN, it certainly can impersonate SN to establish a communication session with a user roamed into SN. She can also impersonate any subscriber of SN. Let us assume that a user U, whose home network HN is not corrupted, roams into a network SN' which is not corrupted, either. To impersonate the user U, the adversary must send back a correct response RES corresponding to a user authentication request sent by SN'. The adversary, however, can not derive RES from the corrupted network SN since the authentication vector containing RES was transferred between HN and SN'. So the adversary can not impersonate the user U. Next, let us see if the adversary can impersonate the network SN'. There are two possible scenarios:

- Assume that the user visited the corrupted network SN before and the adversary has an unused authentication vector which was ordered by SN. Then the adversary may decide to use the authentication vector to impersonate the network SN'. As has been discussed in Section 5.1, the impersonation attempt will fail as the user can verify that the authentication vector was not ordered by SN'.
- Assume that adversary does not have an authentication vector for the user. The adversary starts AP-AKA by sending $FRESH$ and $ID_{SN'}$ to the user, the user replies with RN and U_{MAC}. Then the adversary orders authentication vectors for the user through the corrupted network SN. By verifying U_{MAC}, however, the home network can determine that the user is in not in SN and thus refuse the request.

From the above analysis, we see that the corruption of a serving network only affects those users who either subscribes to or roam into the corrupted network; the adversary can not impersonate uncorrupted serving networks. Thus, the impact of network corruption is drastically lowered on AP-AKA in comparison with 3GPP AKA.

5.3 Provable Security

So far, we have showed that the protocol AP-AKA defeats re-direction attack and lowers the impact of network corruption. In fact, we have also analyzed the security of AP-AKA within a formal model of security. In Appendix A, we present a formal model of security for authentication and key agreement protocols in the mobile setting. The model consists of two systems, an ideal system and a real system. Security is based on simulatability of adversaries in the two systems. Our model is adapted from Shoup's [29] formal model of security for authenticated key exchange in the three-party setting. For our interest, we extend Shoup's model to the mobile setting and formulate the definition of authentication. In our model, we first consider the case that the adversary can not corrupt honest networks to obtain authentication data of users. Then, we consider the case that the real-world adversary can adaptively corrupt networks to obtain authentication data of users. In the following, we provide an overview of the security model, as well as the main result proved within the model. Details of the security model and security proofs can be found in Appendices A, B, and C.

Ideal System. In the ideal system, we do not distinguish between foreign networks and home networks; they are all called serving networks, or simply, networks. Each user or network is called an entity. Communications are between user entities and network entities. With this abstraction, authentication and key agreement protocols in the ideal system can be treated in the same way as in the two-party setting. This allows us to utilize the ideal system defined in Shoup's security model. In the ideal system, there is an *adversary*, who plays a game by issuing a sequence of operations to a *ring master*. All the adversary can do is create and connect entity instances according to some rules, whereby entity instances obtain random session keys from the ring master. Entity instances that are connected share a common session key. The adversary learns no more information about the session key other than that is leaked through its use. Basically, the ideal system describes the service that an authentication and key agreement protocol is supposed to provide. For further motivations on the ideal system, we refer the readers to [29].

In the ideal system, the adversary can execute six types of operations: *initialize entity, initialize entity instance, abort session, start session, application* and *implementation*. As the adversary executes operations, a transcript logging her activities in constructed in the ideal system.

Real System. In the real system, we have a powerful adversary who has full control of the communication channels between a user and a network. The real-world adversary can deliver messages out of order and to unintended recipients, concoct messages of her own choosing, and start up arbitrary number of sessions between a user and a network. Moreover, it is the adversary that drives everything forward; users and networks only follow the instructions of the real-world adversary. The objective of the real-world adversary is to defeat the goals set for an authentication and key agreement protocol. The adversary is not only interested in recovering session keys, but also interested in establishing communication sessions between unintended entities. The adversary executes four types of operations: *initialize entity, initialize entity instance, deliver message*, and *application*. As in the ideal system, operations of the adversary are logged in a real-world transcript

Definition of Security. Security is based on simulatability of adversaries in the two different systems. There are two requirements:

> **Completion.** For every efficient real-world adversary that faithfully delivers messages between two compatible entity instances, both entity instances accept and share the same session key.

Simulatability. For every efficient real-world adversary, there exists an ideal-world adversary such that their transcripts are computationally indistinguishable.

Within the security model, we have proved the following theorem:

Main Theorem. *Assume that G is a pseudorandom function family, F and H are secure message authentication codes, and F, G, and H are independent. Then AP-AKA is a secure authentication and key agreement protocol.*

5.4 Comparison with Other Proposals

Over the last decade, numerous protocols specifically designed for wireless networks have been proposed, e.g., [4, 9, 10, 11, 26, 27, 28, 30]. Most of the proposed protocols were designed based on ad-hoc approaches (i.e., breaking and fixing) and have been found containing various flaws [14, 17, 18]. Based on the underlying cryptographic primitives, the proposed protocols can be divided into two classes: public-key protocols and symmetric-key protocols. In public-key protocols, it is typically assumed that the user or the network can verify its partner's public key certificate. This requires the construction of large-scaled (or even, global) public key infrastructure in order to support global roaming. There are also concerns on the processing power of mobile stations and the bandwidth consumption in exchanging public keys of large size.

In most of the proposed symmetric-key protocols, e.g., [22, 11, 26, 28], a foreign network basically acts as a proxy of the home network. The home network is on-line involved in every authentication process in the foreign network. In addition, most of the symmetric-key protocols were designed in the three-party setting and might not be efficient in the home network. In [26], the authors suggested a construction which consists of two separate protocols; one protocol is designed in the two-party setting and will be running in the home network, another protocol is designed in the three-party setting and will be running in foreign networks. With this construction, the protocol can be optimized both for the home network and for foreign networks. This construction, however, requires that the mobile station recognize the identity of the home network. When the home network is merged into another network, the mobile station may not be able to authenticate the merged network although it has knowledge of the user's authentication key. Another concern on this construction is the interaction between the two component protocols. Such interaction may induce security implications (see [31]) on either of them.

As shown in Section 4, the protocol AP-AKA can be executed efficiently both in the home network and in foreign networks. The user does not need to determine whether he is in a foreign network or in the home network. Based on each authentication vector, the user and a foreign network can perform authentication and key agreement without the involvement of the home network. In addition, AP-AKA allows the home network to select its own algorithms used in the protocol; users only need to share algorithms with their home networks. The benefit is that a mobile station does not need to implement all the algorithms used by serving networks, it only needs to implement those algorithms used by its home network.

6 Conclusion

This paper investigates the security of 3GPP AKA and examines operational difficulty involved with sequence number management. 3GPP AKA is based on the framework of GSM AKA and intends to defeats real and perceived attacks, especially the so-called false base station attack. In this paper, we show that 3GPP AKA is vulnerable to a variant of false base station attack. The vulnerability allows an adversary to re-direct user traffic to an unintended network. It also allows an adversary to use authentication vectors obtained from a corrupted network to impersonate other networks. In addition, the need of synchronization between a user and his home network incurs considerable difficulty for the normal operation of 3GPP AKA.

To provide further enhancement on 3GPP AKA, we present an authentication and key agreement protocol which defeats re-direction attack and drastically lowers the impact of network corruption. The protocol AP-AKA also eliminates synchronization between a user and his home network. Following the multi-party simulatability approach, we have developed a formal model of security for symmetric-key based authentication and key agreement protocols in the mobile setting. Within this model, we have analyzed the security of AP-AKA against a powerful adversary having full control of the communication channels between a user and a network.

References

[1] 3GPP TS 21.102, 3rd Generation Partnership Project (3GPP); Technical Specification Group (TSG) SA; 3G Security;Security Architecture, version 4.2.0, Release 4, 2001.

[2] 3GPP TR 33.902, 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Formal Analysis of the 3G Authentication Protocol", version 3.1.0 (Release 1999).

[3] 3GPP TS 21.102, 3rd Generation Partnership Project (3GPP); Technical Specification Group (TSG) SA; 3G Security; Specification of the MILENAGE algorithm set: An example algorithm set for the 3GPP authentication and key generation functions $f1, f1^*, f2, f3, f4, f5$ and $f5^*$; Document 1: General, version 4.2.0, Release 4, 2001.

[4] A. Aziz and W. Diffie, Privacy and authentication for wireless local area networks, *IEEE Personal Communications*, vol. 1, 1994, pp. 25-31.

[5] D. Beaver, Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority, *Journal of Cryptology*, vol. 4, 1991, pp. 75–122.

[6] M. Bellare and P. Rogaway, Entity authentication and key distribution, *Advances in Cryptology - Crypto' 93 Proceedings*, Lecture Notes in Computer Science, vol. 773, Springer-Verlag, 1993. pp. 232-249.

[7] M. Bellare and P. Rogaway, Provably secure session key distribution–The three party case, *Proc. 27th ACM Symp. on Theory of Computing*, Las Vegas, NV, USA, May 1995, pp. 57-66.

[8] M. Bellare, R. Canetti, and H. Krawczyk, A modular approach to the design and analysis of authentication and key exchange protocols, *Proceedings of 30th Annual ACM Symposium on Theory of Computing*, ACM Press, new York, 1998.

[9] M. J. Beller, L.-F. Chang, and Y. Yacobi, Privacy and authentication on a portable communication system, *IEEE Journal on Selected Areas in Communications*, Vol. 11, 1993, pp. 82-829.

[10] M. Beller and Y. Yacobi, Fully-fledged two-way public key authentication and key agreement for low-cost terminals, *Electronics Letters*, vol. 29, 1993, pp. 999-1001.

[11] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, and M. Yung, The Kryptoknight family of light-weight protocols for authentication and key distribution, *IEEE/ACM Trans. on Networking*, vol.3, 1995, pp. 31-41

[12] S. Blake-Wilson, D. Johnson and A. Menezes, Key agreement protocols and their security analysis, *Cryptography and Coding*, Lecture Notes in Computer Science, vol. 1355, 1997, pp. 30-45.

[13] S. Blake-Wilson and A. Menezes, Entity authentication and key transport protocols employing asymmetric techniques", Proceedings of Security Protocols Workshop, 1997.

[14] C. Boyd and A. Mathuria, Key establishment protocols for secure mobile communications: A selective survey, *Proceedings of ACISP'98* , Lecture Notes in Computer Science, vol. 1438, 1998, pp. 344-355.

[15] V. Boyko, P. MacKenzie, and S. Patel, Provably secure password-authenticated key exchange using Diffie-Hellman, *Advances in Cryptology - Eurocrypt'2000 Proceedings*, Lecture Notes in Computer Science, vol. 1807, Springer-Verlag, 2000. pp. 156-171.

[16] M. Burrows, M. Abadi, and R. Needham, A logic of authentication, *ACM Transactions on Computer Systems*, vol. 8, 1990, pp.18-36.

[17] L. Buttyan, C. Gbaguidi, S. Sttmann, and U. Wilhelm, Extensions to an authentication technique proposed for global mobility network, *IEEE Transactions on Communications*, vol. 48, 2000, pp. 373-376.

[18] U. Carlsen, Optimal privacy and authentication on a portable communications system, *Operating Systems Review*, vol. 28, 1994, pp.16-23.

[19] European Telecommunications Standards Institute (ETSI), GSM 02.09: Security Aspects, June 1993.

[20] L. Harn and H. Lin, "Modifications to enhance the security of GSM", *Proceedings of 5th National Conference on Information Security*, Taiwan, pp. 74-76, May 1995.

[21] ISO/IEC 9798-4, Information technology-Security Techniques- Entity Authentication-Part 4: Mechanisms using a cryptographic check function.

[22] C.H. Lee, M.S. Hwang, and W.P. Yang, Enhanced privacy and authentication for the global system for mobile communications, *Wireless Networks*, vol.5, pp. 231-243, 1999.

[23] H. Lin and L. Harn, "Authentication protocols for personal communication system", *Proceedings of ACM SIGCOMM'95*, August 1995.

[24] M. Luby, *Pseudorandomness and Cryptographic Applications*, Princeton University Press, 1996.

[25] C. Mitchell, The security of the GSM air interface protocol, Technical Report, RHUL-MA-2001-3, Royal Holloway, University of London, 2001.

[26] R. Molva, D. Samfat, and G. Tsudik, Authentication of mobile users, *IEEE Network*, 1994, pp. 26-34.

[27] Yi Mu, V. Varadharajan, On the design of security protocols for mobile communications, *Proceedings of ACISP'96*, Lecture Notes in Computer Science, vol. 1172, Springer-Verlag, 1996, pp. 134-145.

[28] C. Park, K. Kurosawa, T. Okamoto and S. Tsujii, On key distribution and authentication in mobile radio networks, *Advances in Cryptology-Eurocrypt'93 Proceedings*, Lecture Notes in Computer Science, vol. 765, 1993, pp. 461-465.

[29] V. Shoup, On formal models for secure key exchange, *Proceedings of the Sixth Annual ACM Conference on Computer and Communications security (invited talk)*, 1999.

[30] M. Tatebayashi, N. Matsuzaki and D.B.J. Newman, Key distribution protocol for digital mobile communication systems, *Advances in Cryptology-Crypto '89 Proceedings*, Lecture Notes in Computer Science, vol. 435, 1989, pp. 324-334.

[31] W. Tzeng and C. Hu, Inter-protocol interleaving attacks on some authentication and key distribution protocols, *Information Processing Letters*, vol. 69, 1999, pp. 297-302.

A Security Model

Bellare and Rogaway [6] proposed the first formal model of security for symmetric-key based authentication and key distribution protocols in the two-party setting. In their subsequent work [7] Bellare and Rogaway extended their model to analyze key distribution protocols in the three-party setting. The Bellare-Rogaway model was also adapted to treat public-key based protocols by Blake-Wilson et al. [13, 12]. In [8] Bellare, Canetti, and Krawczyk presented a different approach to formal model of security for authentication and key exchange protocols. This approach stems from the multi-party simulatability tradition [5]. Shoup [29] extended this approach and proposed a formal model of security for key exchange protocols in the three-party setting. Based on the simulatability approach, Boyko, et al. [15] recently proposed a formal model of security for password-authenticated key exchange protocols.

In this paper, we utilize Shoup's formal model of security to analyze authentication and key agreement protocols in the mobile setting. To capture the security threats, we identify two types of communication channels in mobile networks. The first type consists of channels within and between networks and the second type consists of channels between the users and the networks. Channels of the first type connect fixed communication entities. In practice such channels are protected through dedicated communication circuits or through security mechanisms running in high layer, e.g., application layer. Channels of the second type connect dynamic communication entities, which are usually implemented via wireless media. Attacks on these channels represent a major threat in mobile networks. We assume that the adversary has full control on channels of the second type. She can deliver messages out of order and to unintended recipients, concoct messages of her own choosing, and start up arbitrary number of sessions between a user and a network. She can even acquire session keys and apply them through an arbitrary mathematical function.

In the following, we assume that the channels of the first type are secured. In Appendix C, we consider the case in which adversaries can adaptively corrupt channels of the first type. In the following, we specify the actions of adversaries in two different worlds: the ideal system and the real system. Security of authentication and key agreement protocol is based on simulatability in the two different worlds.

A.1 The Ideal System

Assume that there is a set of *users* who are served by a set of *networks*. Here, we do not distinguish between foreign networks and home networks; they are all called serving networks, or simply, networks. Each user or network is called an entity. Communications are between user entities and network entities. With the abstraction, authentication and key agreement protocols in the ideal system can be treated in the same way as in the two-party setting. This allows us to utilize the ideal system defined in Shoup's security model. For motivations behind the ideal system, we refer the readers to [29].

Let E_i, indexed $i = 1, 2, \ldots$, denote all the entities. Each entity E_i may have several *instances* I_{ij}, $j = 1, 2, \ldots$. There is also an *adversary*, who plays a game by issuing a sequence of operations to a *ring master*. All the adversary can do is create and connect entity instances, whereby entity instances obtain random session keys from the ring master. The adversary learns no more information about the session key other than that is leaked through its use. Each operation of the adversary produces a record, which is placed in a *transcript*. There are a total of six operations:

(**initialize entity**, $i, role_i, ID_i$): This operation specifies an identity ID_i, along with a value $role_i \in \{0.1\}$. The adversary assigns the identity ID_i to the i-th entity. When $role_i = 0$, the entity is initialized as a user, denoted by U_i; otherwise, the entity is initialized as a network, denoted by N_i. The identity ID_i may be an arbitrary bit string that has not been assigned to another entity. The execution of this operation produces a record of the form (`initialize entity`, $i, role_i, ID_i$).

(**initialize entity instance**, i, j, PID_{ij}): This operation specifies an entity instance I_{ij}, along with a partner identity PID_{ij}. The entity U_i or N_i must have been previously initialized, but I_{ij} should not haven been previously initialized. After the execution of this operation, the instance I_{ij} is *active* and remains active until the execution of either an *abort session* or *start session* operation on I_{ij}. This operation produces a record of the form (`initialize entity instance`, i, j, PID_{ij}).

(**abort session**, i, j): This operation specifies an initialized entity instance I_{ij}. The record for this operation is (`abort session`, i, j).

(**start session**, $i, j, connection\_assignment[, key]$): This operation specifies an active entity instance I_{ij}, along with a connection assignment that specifies how the session key K_{ij} is generated for I_{ij}. This operation produces a record (`start session`, i, j). There are three possible connection assignments: (`create`, i', j'), (`connect`, i', j'), and `compromise`. Each connection assignment is selected under certain rules. The optional *key* field is present only if the *connection_assignment* is `compromise`. The session key K_{ij} is determined according to *connection_assignment* as follows:

- (`create`, i', j'): the ring master generates a random bit string for K_{ij}. This assignment is legal if: (i) $I_{i'j'}$ is a previously initialized instance *compatible* with I_{ij}, that is,

$PID_{ij} = ID_{i'}, PID_{i'j'} = ID_i$ and $role_i \neq role_{i'}$; and (ii) the connection assignment (`create`, i', j') was not made before, either on I_{ij} or on other instances. When the *start session* operation completes, we say that I_{ij} is isolated for $I_{i'j'}$.

- (`connect`, i', j'): the ring master sets K_{ij} equal to $K_{i'j'}$. This assignment is legal if $I_{i'j'}$ has been isolated for I_{ij}. When the *start session* operation completes, $I_{i'j'}$ is no longer isolated.
- `compromise`: the ring master sets K_{ij} equal to *key*.

(**application**, f): This operation specifies a function f that is applied to the session keys $\{K_{ij}\}$ and a random input R. After the execution of this operation, the ring master gives the adversary $f(R, \{K_{ij}\})$. This operation generates a record of the form (`application`, $f, f(R, \{K_{ij}\})$).

(**implementation**, *comment*): This operation allows the adversary to add a *comment* to the transcript. The record for this operation is (`implementation`,*comment*).

As the adversary executes operations in the ideal system, a transcript logging his activities is constructed. For an adversary $\mathcal{A}^*$, we use $\mathcal{T}_{\mathcal{A}^*}$ to denote the transcript.

A.2 The Real System

We now describe the real system which models the operations executed by a real-world adversary who has full control of the communication channels between a user and a network. As mentioned earlier, the real-world adversary can deliver messages out of order and to unintended recipients, concoct messages of her own choosing, and start up arbitrary number of sessions between a user and a network. Moreover, it is the adversary that drives everything forward; users and networks only follow the instructions of the real-world adversary. The objective of the real-world adversary is to defeat the goal set for an authentication and key agreement protocol. In the following, we describes the operations of the real-world adversary.

As in the ideal system, the real-world adversary initializes an entity by executing the operation

$$(\textbf{initializeentity}, i, role_i, ID_i)$$

This operation is logged in the transcript as (`initialize entity`, $i, role_i, ID_i$). Likewise, the real-world adversary initializes an entity instance through the operation

$$(\textbf{initializeentityinstance}, i, j, PID_{ij})$$

The record for this operation is (`initialize entity instance`, i, j, PID_{ij}). Unlike in the ideal system, however, entities and entity instances in the real system are not just placeholders; they have internal states and can generate messages of their choosing.

When an entity is initialized as a network N_i with identity ID_i, a protocol-specific *initialization routine* is started to initialize the internal state of N_i and to establish service agreement with all previously initialized networks. To do so, the ring master in the real system provides to the initialization routine a list of previously initialized networks, the routine selects a network, say $N_{i'}$, from the list and sends the message (ID_i, *service agreement*) to $N_{i'}$. Upon receipt of the message, $N_{i'}$ updates its internal state and creates two mailboxes, $InMail_i$ and $OutMail_i$, for N_i. Next, $N_{i'}$ sends back an acknowledgement ($ID_{i'}$, *approval*) to N_i. After receiving the acknowledgement, N_i updates its internal state and also creates two mailboxes, $InMail_{i'}$ and $OutMail_{i'}$ for $N_{i'}$. Both N_i and its instances can write messages into $OutMail_{i'}$ and read messages from $InMail_{i'}$; the adversary is not allowed to access the mailboxes. Both *read* and *write* are atomic operations. When a message is written into $OutMail_{i'}$ by N_i or its instance, the ring master takes out the message and delivers it to the mailbox $InMail_i$ of $N_{i'}$. Through the ring master and the mailboxes, we model a secure channel between N_i and $N_{i'}$. All the instances of N_i and $N_{i'}$ share this secure channel.

In the real system, each user is associated with a network, i.e., the home network, which has been previously initialized. To specify the association between a user and his home network, we assume that the identity of the user is prefixed by identity of the home network. When an entity is initialized as a user U_i with identity ID_i, where ID_i is prefix by the identity $ID_{i'}$ of a previously initialized network $N_{i'}$, a protocol-specific *registration routine* is started to initialize the internal state of U_i and to register U_i's identity with $N_{i'}$. During the registration, the ring master generates a random bit string, denoted by K_i, and delivers K_i to both U_i and $N_{i'}$. Upon receipt of K_i, the user U_i stores K_i in the variable LTS_i, and the network $N_{i'}$ stores the pair (ID_i, K_i) in the variable $LTS_{i'}$.

An entity instance I_{ij} is a state machine having access to ID_i, $role_i$, and LTS_i. After starting in some initial state, the state of I_{ij} may be updated by a message delivered by the adversary. The message-delivery operation takes the form

$$(\textbf{delivermessage}, i, j, type, InMsg),$$

where a *type* is assigned to the message $InMsg$. In response to the message delivery, I_{ij} updates its state, generates a response message $OutMsg$, and reports its *status*. The response message and status information are given to the adversary. There are three possible status:

continue: I_{ij} is prepared to receive another message.

accept: I_{ij} is finished and has generated a session key SK_{ij}.

reject: I_{ij} is finished but refuses to generate a session key.

If the *status* is not `continue`, I_{ij} is no longer *active*. Dependent on the status, this operation generates one or two records. The first record is (`implementation, deliver message`, i, j, *type*, $InMsg$, $OutMsg$, *status*). The second record is (`start session`, i, j) if *status* =`accept`, or (`abort session`, i, j) if *status* =`reject`.

In the real system, the adversary may also execute the application operation (**application**, f). This is the same operation as described in the ideal system, except that the actual session keys $\{SK_{ij}\}$ are used. In addition, the random input R is independent of any bits used by entities or entity instances during initialization and during the execution of the protocol. The application operation produces the record (`application`, $f, f(R, \{SK_{ij}\})$).

A.3 Definition of Security

We now formulate the definition of security of an authentication and key agreement protocol in our model.

> **Completion.** For every efficient real-world adversary that faithfully delivers messages between two compatible entity instances, both entity instances accept and share the same session key.
>
> **Simulatability.** For every efficient real-world adversary $\mathcal{A}$, there exists an ideal-world adversary $\mathcal{A}^*$ such that their transcripts $\mathcal{T}_{\mathcal{A}}$ and $\mathcal{T}^*_{\mathcal{A}}$ are computationally indistinguishable.

The above requirements are essentially the same as those given by Shoup [29]. In Shoup's security model, the connection assignment `create` is always legal. So there may be multiple instances which are isolated for an initialized instance. In our model, we add restrictions to the connection assignment `create` to ensure that at most one instance is isolated for an initialized instance. When making the connection assignment (`create`, i', j'), we do not require that $I_{i'j'}$ is still *active*. This is different from the arrangement in [15]. The reason for our arrangement is because it is legal for the adversary to issue the operation (**abort session**, i', j') before executing the operation (**start session**, i, j, (`create`, i', j')).

B Security Proofs

Let $\{0,1\}^n$ denote the set of binary strings of length n and $\{0,1\}^{\leq n}$ denote the set of binary strings of length at most n. For two binary strings s_1 and s_2, the concatenation of s_1 and s_2 is denoted by $s_1 \| s_2$. A real-valued function $\epsilon(k)$ of non-negative integers is called *negligible* (in k) if for every $c > 0$, there exists $k_0 > 0$ such that $\epsilon(k) \leq 1/k^c$ for all $k > k_0$.

Let $X = \{X_k\}_{k \geq 0}$ and $Y = \{Y_k\}_{k \geq 0}$ be sequences of random variables, where X_k and Y_k take values in a finite set S_k. For a probabilistic polynomial-time algorithm D that outputs 0 or 1, we define the *distinguishing advantage* of D as the function

$$\mathbf{Adv}^{dist}_{X_k,Y_k}(D) = |Pr(D(X_k) = 1) - Pr(D(Y_k) = 1)|.$$

If for every probabilistic polynomial-time algorithm, the distinguishing advantage is negligible in k, we say that X and Y are *computationally indistinguishable.*

Let $G : \{0,1\}^k \times \{0,1\}^d \rightarrow \{0,1\}^s$ denote a family of functions and let $\mathcal{U}(L,l)$ denote the family of all functions from $\{0,1\}^d$ to $\{0,1\}^s$. For a probabilistic polynomial-time oracle machine A, the *prf-advantage* of A is defined as

$$\mathbf{Adv}^{prf}_{G}(A) = |Pr(g \stackrel{R}{\leftarrow} G : A^g = 1) - Pr(g \stackrel{R}{\leftarrow} \mathcal{U}(L,l) : A^g = 1)|,$$

where $g \stackrel{R}{\leftarrow} G$ denotes the operation of randomly selecting a function g from the family G. We associate to G an insecurity function:

$$\mathbf{Adv}^{prf}_{G}(t,q) = \max_{A \in \mathcal{A}(t,q)} \mathbf{Adv}^{prf}_{G}(A),$$

where $\mathcal{A}(t,q)$ denotes the set of adversaries that make at most q oracle queries and have running time at most t. Assume that d and s are polynomials in k. If for every probabilistic polynomial-time oracle machine A, $\mathbf{Adv}_G^{prf}(A)$ is negligible in k, then we say that G is a *pseudorandom* function family.

A *Message Authentication Code* is a family of functions $F : \{0,1\}^k \times Dom(F) \rightarrow \{0,1\}^l$, where $Dom(F)$ denotes the domain of F. In this paper, $Dom(F) = \{0,1\}^{\leq L}$. For $K \in \{0,1\}^k$ and $M \in \{0,1\}^{\leq L}$, let $\sigma = F(K,M)$. We refer to σ as the tag or MAC of M. For the security of F, we will use the notion of security against chosen message attacks. An adversary, called a forger in this context, is a probabilistic polynomial-time algorithm which has access to an oracle that computes MACs under a randomly chosen key K. We define the *mac-advantage* of an adversary A, denoted by $\mathbf{Adv}_F^{mac}(A)$, as the probability that $A^{F(K,)}$ outputs a pair (σ, M) such that $\sigma = F(K,M)$, and M was not a query of A to its oracle. We associate to F an insecurity function,

$$\mathbf{Adv}_F^{mac}(t,q) = \max_{A \in \mathcal{A}(t,q)} \mathbf{Adv}_F^{mac}(A),$$

where $\mathcal{A}(t,q)$ denotes the set of adversaries that make at most q oracle queries and have running time at most t. If for every polynomially bounded adversary A, $\mathbf{Adv}_F^{mac}(A)$ is negligible in k, we say that F is a *secure* message authentication code.

Definition 1 *Let I_{ij} be an entity instance in the real system. A stimulus on I_{ij} is a message such that the status of I_{ij} changes from continue to accept after receiving the message.*

Definition 2 *Let $\mathcal{A}$ be a real world adversary and let $\mathcal{T}_{\mathcal{A}}$ be the transcript of $\mathcal{A}$. For every accepted instance I_{ij}, if the stimulus on I_{ij} was output by a compatible instance, we say that $\mathcal{T}_{\mathcal{A}}$ is an authentic transcript.*

In AP-AKA, we assume that each entity has a random number generator which produces random numbers, e.g., $FRESH$, RN, and $RAND$, for the entity and its instances. Without confusion, we also say that the random numbers are generated by the entity or its instances.

Definition 3 *Let $\mathcal{A}$ be a real-world adversary and let $\mathcal{T}_{\mathcal{A}}$ be the transcript of $\mathcal{A}$. In the game of $\mathcal{A}$, if the random numbers generated by an entity and its instances are different, we say that $\mathcal{T}_{\mathcal{A}}$ is a collision-free transcript.*

Let $|FRESH|, |RN|$ and $|RAND|$ denote the length of $FRESH$, RN, and $RAND$ respectively. Assume that these numbers are randomly selected in the game of $\mathcal{A}$. Let $C_{\mathcal{A}}$ denote the event that $\mathcal{T}_{\mathcal{A}}$ is collision-free. Then

$$Pr(\overline{C}_{\mathcal{A}}) \leq \frac{n_i^2}{2}(2^{-|FRESH|} + 2^{-|RN|} + 2^{-|RAND|}), \tag{1}$$

where n_i denotes the number of instances initialized by $\mathcal{A}$. In the following, we assume that $|FRESH|, |RN|$ and $|RAND|$ are polynomials in k, then $Pr(\overline{C}_{\mathcal{A}})$ is negligible.

Definition 4 *Let $\mathcal{T}_{\mathcal{A}}$ be the transcript of a real-world adversary $\mathcal{A}$. Let $\sigma_1, \sigma_2, \ldots, \sigma_n$ denote all the tags which are computed under F by entities and entity instances. If $\sigma_i \neq \sigma_j$ for any $i \neq j$, we say that F is collision-resistant in $\mathcal{T}_{\mathcal{A}}$. Similarly, if all the tags computed under H are different, we say that H is collision-resistant in $\mathcal{T}_{\mathcal{A}}$.*

Lemma 1 *Let $\mathcal{A}$ be a real-world adversary and let $\mathcal{T}_{\mathcal{A}}$ be the transcript of $\mathcal{A}$. Assume that $\mathcal{T}_{\mathcal{A}}$ is collision-free. Also assume that F and H are independent function families and are collision-resistant in $\mathcal{T}_{\mathcal{A}}$. Let $M_{\mathcal{A}}$ denote the event that $\mathcal{T}_{\mathcal{A}}$ is authentic. Then*

$$Pr(\overline{M}_{\mathcal{A}}) \leq n_i(\mathbf{Adv}_F^{mac}(t,q) + \mathbf{Adv}_F^{mac}(t,q)\mathbf{Adv}_H^{mac}(t,q')),$$

where $t = O(T), q = O(3n_i)$, $q' = O(n_i)$, T is the running time of $\mathcal{A}$, and n_i is the number of instances initialized by $\mathcal{A}$.

Proof. If $\mathcal{T}_{\mathcal{A}}$ is not authentic, there exists at least one instance which has accepted, but the stimulus on this instance was not output by a compatible instance. We claim that the probability of such an event is upper-bounded by $\mathbf{Adv}_F^{mac}(t,q) + \mathbf{Adv}_F^{mac}(t,q)\mathbf{Adv}_H^{mac}(t,q')$. To prove our claim, we consider the following three cases.

(i). Let $I_{i'j'}$ be a network instance which has received the message (*user data response*, RN, U_{MAC}) and has accepted. Since the identity $ID_{i'}$ is used in the computation of U_{MAC}, the stimulus on $I_{i'j'}$ could not be output by a user instance not compatible with $I_{i'j'}$. If the stimulus was not output by a user instance compatible with $I_{i'j'}$, then we can construct an adversary $\mathcal{F}$ for the message authentication code F. The adversary $\mathcal{F}$ has oracle access to F_K and H_K, where K was chosen at random. Assume that $PID_{i'j'}$ is assigned to a user U, which may or may not be initialized by $\mathcal{A}$. The adversary $\mathcal{F}$ begins its experiment by selecting authentication keys for all users, except that the authentication key for user U is not chosen. Next, $\mathcal{F}$ runs $\mathcal{A}$ just as in the real system. In the game of $\mathcal{A}$, if an entity or entity instance needs to evaluate F and H under the key of U, $\mathcal{F}$ provides the evaluation by appealing to the oracles F_K and H_K. If an entity or entity instance needs to evaluate G under the key of U, $\mathcal{F}$ supplies a random number or even a constant for the evaluation. If at any point $I_{i'j'}$ accepts, $\mathcal{F}$ stops and outputs U_{MAC} as well as $FRESH||RN||ID_{i'}$, where $FRESH$ was sent out by $I_{i'j'}$ in *user data request*. Else, $\mathcal{F}$ stops at the end of the game of $\mathcal{A}$ and outputs an empty string.

Let $Succ(\mathcal{F}, F)$ denote the event that $\mathcal{F}$ outputs a MAC and a message and the message was not queried to the oracle F_K. Let $AS_{i'j'}$ denote the event that $I_{i'j'}$ has accepted, but the stimulus on $I_{i'j'}$ was not output by a user instance. If $AS_{i'j'} = 1$, then $\mathcal{F}$ has successfully forged the MAC for the message $FRESH||RN||ID_{i'}$ and this message was not queried to the oracle F_K. This implies that

$$Pr(AS_{i'j'} = 1) \leq Pr(Succ(\mathcal{F}, F) = 1).$$

For each instance of U and $N_{i'}$, $\mathcal{F}$ makes at most three oracle queries to F_K. So, the total number of oracle queries to F_K is $q = O(3n_i)$. Hence,

$$Pr(AS_{i'j'} = 1) \leq \mathbf{Adv}_F^{mac}(t,q), \tag{2}$$

where $t = O(T), q = O(3n_i)$.

(ii). Let I_{ij} be a user instance which has received *user authentication request* and has accepted. Let AS_{ij} denote the event that the stimulus on I_{ij} was not output by a network instance. Let IS_{ij} denote the event that the stimulus on I_{ij} was output by a network instance $I_{i'_1 j'_1}$, but not compatible with I_{ij}. If IS_{ij} is true, then the instance $I_{i'_1 j'_1}$ received the message (*user data response*, $RN, F_K(RN||FRESH||ID_{i'_1})$) before sending out $RAND$ and $AUTH$, where $AUTH = idx\|RN_{idx}\|MAC$ and $RN_{idx} = H_K(idx, RN)$. Since, $\mathcal{T}_{\mathcal{A}}$ is collision-free, RN could not be generated by a user instance other than I_{ij}. This implies that the adversary $\mathcal{A}$ has successfully concocted the MAC for the message $RN\|FRESH\|ID_{i'_1}$. By (2), we have

$$Pr(IS_{ij} = 1) \leq \mathbf{Adv}_F^{mac}(t, q). \tag{3}$$

Now suppose that AS_{ij} is true, then the adversary $\mathcal{A}$ has successfully concocted the MAC for the message $RAND\|idx\|RN_{idx}$. Running the adversary $\mathcal{A}$, we can construct an adversary $\mathcal{F}'$ for both F and H. The adversary $\mathcal{F}'$ works in the same way as F except that, when I_{ij} accepts, $\mathcal{F}'$ stops and outputs two pairs: $(MAC, RAND\|idx\|RN_{idx})$ and $(RN_{idx}, idx\|RN)$. Since $(RAND, AUTH)$ was not output by any instance, the message $RAND\|idx\|RN_{idx}$ was not queried to the oracle F_K in the experiment of $\mathcal{F}'$ and $idx\|RN$ was not queried to the oracle H_K either. Using the notation $Succ(\mathcal{F}', F)$ and $Succ(\mathcal{F}', H)$ as described above, we have

$$Pr(AS_{ij} = 1) \leq Pr((Succ(\mathcal{F}', F) = 1) \wedge (Succ(\mathcal{F}', H) = 1)).$$

Since F and H are independent,

$$Pr((Succ(\mathcal{F}', F) = 1) \wedge (Succ(\mathcal{F}', H) = 1)) = Pr(Succ(\mathcal{F}', F) = 1)Pr(Succ(\mathcal{F}', H) = 1).$$

In the experiment of $\mathcal{F}'$, the number of oracle queries to F_K is $q = O(3n_i)$ and the number of oracle access to H_K is $q' = O(n_i)$. Therefore,

$$Pr(AS_{ij} = 1) \leq \mathbf{Adv}_F^{mac}(t, q)\mathbf{Adv}_H^{mac}(t, q'). \tag{4}$$

By (3) and (4), the probability that the stimulus on a user instance I_{ij} was not output by a compatible network instance is upper-bounded by

$$Pr(AS_{ij} = 1) + Pr(IS_{ij} = 1) \leq \mathbf{Adv}_F^{mac}(t, q) + \mathbf{Adv}_F^{mac}(t, q)\mathbf{Adv}_H^{prf}(t, q').$$

(iii). Let $I_{i''j''}$ be a network instance which has received (*user authentication response*, RES) and has accepted, where $RAND$ was sent out by $I_{i''j''}$ in *user authentication request*. If the stimulus on $I_{i''j''}$ was not output by a user instance, then the adversary $\mathcal{A}$ has successfully concocted the MAC for the message $RAND$. Similar to (2), it can be proved that the probability of such an event is upper-bounded by $\mathbf{Adv}_F^{mac}(t, q)$. Next, if the stimulus on $I_{i''j''}$ was output by a user instance $I_{i_1 j_1}$ which is not compatible with $I_{i''j''}$. Then the user instance $I_{i_1 j_1}$ received $RAND$ and $AUTH$ before it output the stimulus. Since $\mathcal{T}_{\mathcal{A}}$ is collision-free, $RAND$ and $AUTH$ could not be output by a network instance other than $I_{i''j''}$. This means that it is the adversary who concocted the MAC for $RAND\|idx\|RN_{idx}$. By (4), the probability of such an event is upper-bounded by $\mathbf{Adv}_F^{mac}(t, q)\mathbf{Adv}_H^{prf}(t, q')$. So the probability that the stimulus on $I_{i''j''}$ was not output by a compatible user instance is at most $\mathbf{Adv}_F^{mac}(t, q) + \mathbf{Adv}_F^{mac}(t, q)\mathbf{Adv}_H^{prf}(t, q')$.

Based on the above analysis, it can be concluded that the probability that $\mathcal{T}_{\mathcal{A}}$ is not an authentic transcript is at most $n_i(\mathbf{Adv}_F^{mac}(t, q) + \mathbf{Adv}_F^{mac}(t, q)\mathbf{Adv}_H^{mac}(t, q'))$. $\square$

Lemma 2 *Let $\mathcal{A}$ be a real-world adversary and let $\mathcal{T}_{\mathcal{A}}$ be the transcript of $\mathcal{A}$. Assume that $\mathcal{T}_{\mathcal{A}}$ is authentic and collision-free. Also assume that G is a pseudorandom function family, independent of F and H, and F, H are collision-resistant in $\mathcal{T}_{\mathcal{A}}$. Then there exists an ideal-world adversary A^* such that for every distinguisher D with running time T,*

$$\mathbf{Adv}^{dist}_{\mathcal{T}_{\mathcal{A}},\mathcal{T}_{\mathcal{A}^*}}(D) \leq n_e \mathbf{Adv}^{prf}_{G}(t, q),$$

where n_e is the number of user entities initialized by $\mathcal{A}$ and n_i is the number of instances initialized by $\mathcal{A}$, $t = O(T), q = O(n_i)$.

Proof. We construct a *simulator* which takes the real-world adversary $\mathcal{A}$ as input and creates an ideal-world adversary $\mathcal{A}^*$. The simulator basically has $\mathcal{A}^*$ run the adversary $\mathcal{A}$ just as in the real system. For any `implementation` record in the real-world transcript, $\mathcal{A}^*$ copies this record into the ideal-world transcript by issuing an *implementation* operation. Corresponding to each (`start session,` i, j) record that $\mathcal{A}$'s action cause to be placed in the real-world transcript, $\mathcal{A}^*$ computes a connection assignment, and the ring master in the ideal system substitutes the session key SK_{ij} by an idealized session key K_{ij}, which is a random number. Corresponding to each (`abort session,` i, j) record that $\mathcal{A}$'s action cause to be placed in the real-world transcript, $\mathcal{A}^*$ executes the operation (*abort session*, i, j). For an *application* operation, the ring master in the ideal system makes the evaluations using the idealized session keys. This way, we have an ideal-world adversary whose transcript is almost identical to the transcript of the real-world adversary $\mathcal{A}$. The differences exist in the `application` records. In the following, we show that the connection assignments made by $\mathcal{A}^*$ are legal and the differences between the two transcript are computationally indistinguishable.

Case 1. Assume that a user instance $I_{i_1j_1}$ has received the message (*user authentication request*, $RAND$, $AUTH$) and has accepted, where $AUTH = idx \| RN_{idx} \| MAC$, $idx > 0$. Since $\mathcal{T}_{\mathcal{A}}$ is authentic, this message must be output by a network instance $I_{i'_1j'_1}$ compatible with $I_{i_1j_1}$. In this case, we let the adversary $\mathcal{A}^*$ make the connection assignment (`create`, i'_1, j'_1). We have to argue that this connection assignment was not made before. This is true because $RAND$ and $AUTH$ could not be a stimulus on other user instances, otherwise RN_{idx} would not be acceptable by $I_{i_1j_1}$. So it is legal for the adversary $\mathcal{A}^*$ to make the connection assignment. Consequently, it is also legal to substitute the session key $SK_{i_1j_1}$ by a random number $K_{i_1j_1}$.

Case 2. Assume that a network instance $I_{i'_2j'_2}$ has received the message (*user data response*, RN, U_{MAC}) from a user instance $I_{i_2j_2}$ and has accepted, where $U_{MAC} = F_{K_{i_2}}(FRESH||RN||ID_{i'_2})$. In this case, we let $\mathcal{A}^*$ make the connection assignment (`create`, i_2, j_2) and let the ring master substitute the session key $SK_{i'_2j'_2}$ by a random number $K_{i'_2j'_2}$. Since $\mathcal{T}_{\mathcal{A}}$ is collision-free, $FRESH$ could not be sent out by any instance other than $I_{i'_2j'_2}$. Also since F is collision-resistant in $\mathcal{T}_{\mathcal{A}}$, RN and U_{MAC} could not be a stimulus on any instances other than $I_{i'_2j'_2}$. So the connection assignment (`create`, i_2, j_2) was not made before.

Case 3. Assume that a network instance $I_{i'_3j'_3}$ has received the message (*user authentication response*, RES) from a user instance $I_{i_3j_3}$ and has accepted, where $RES = F_{K_{i_3}}(RAND)$, $RAND$ was sent out by $I_{i'_3j'_3}$. Under the assumption that $\mathcal{T}_{\mathcal{A}}$ is collision-free and F is collision-resistant in $\mathcal{T}_{\mathcal{A}}$, it can be concluded that $I_{i_3j_3}$ has accepted and the stimulus on $I_{i_3j_3}$ was output by $I_{i'_3j'_3}$. According to Case 1, $I_{i_3j_3}$ has been isolated for $I_{i'_3j'_3}$. So it is legal for $\mathcal{A}^*$ to make the connection assignment $(\texttt{connect}, i_3, j_3)$. Accordingly, the ring master set the session key $K_{i'_3 j'_3}$ by $K_{i_3 j_3}$.

Case 4. Assume that a user instance $I_{i_4 j_4}$ has received the message (*user authentication request*, $RAN, AUTH$) from a network instance $I_{i'_4 j'_4}$ and has accepted, where $AUTH = 0|RN_0\|MAC$, $RN_0 = H_{K_{i_4}}(0||RN)$, RN was sent out by $I_{i_4 j_4}$. By the assumption that $\mathcal{T}_\mathcal{A}$ is collision-free and H is collision-resistant in $\mathcal{T}_\mathcal{A}$, we can conclude that $I_{i'_4 j'_4}$ has accepted and the stimulus on $I_{i'_4 j'_4}$ was output by $I_{i_4 j_4}$. According to Case 2, $I_{i'_4 j'_4}$ has been isolated for $I_{i_4 j_4}$. So it is legal for $\mathcal{A}^*$ to make the connection assignment $(\texttt{connect}, i'_4, j'_4)$. Consequently, the ring master set the session key $K_{i_4 j_4}$ by $K_{i'_4 j'_4}$.

The above analysis shows that there exists a connection assignment for each `start session` record in $\mathcal{T}_{\mathcal{A}^*}$. Next, we show that the two transcripts $\mathcal{T}_\mathcal{A}$ and $\mathcal{T}_{\mathcal{A}^*}$ are computationally indistinguishable. Note that if we remove the `application` records in both $\mathcal{T}_\mathcal{A}$ and $\mathcal{T}_{\mathcal{A}^*}$, then the remaining transcripts are exactly the same. So we only need to consider the `application` records in both transcripts. First, let's assume that there is only one user entity initialized by $\mathcal{A}$. Let D be a distinguisher for $\mathcal{T}_\mathcal{A}$ and $\mathcal{T}_{\mathcal{A}^*}$. By running D on $\mathcal{T}_\mathcal{A}$ and $\mathcal{T}_{\mathcal{A}^*}$, we have an adversary D' for G such that

$$\mathbf{Adv}^{dist}_{\mathcal{T}_\mathcal{A}, \mathcal{T}_{\mathcal{A}^*}}(D) = \mathbf{Adv}^{prf}_{G}(D').$$

Thus,

$$\mathbf{Adv}^{dist}_{\mathcal{T}_\mathcal{A}, \mathcal{T}_{\mathcal{A}^*}}(D) \leq \mathbf{Adv}^{prf}_{G}(t, q),$$

where $t = O(T), q = O(n_i)$, n_i is the number of instances initialized by $\mathcal{A}$.

Now, assume that the number of user entities initialized by $\mathcal{A}$ is n_e. Let $K_1, K_2, \ldots, K_{n_e}$ denote the keys of these user entities. Then D and D' have access to the input-and-output pairs of $G_{K_1}, G_{K_2}, \ldots, G_{K_{n_e}}$. Following the hybrid argument given in [24], it can be concluded that

$$\mathbf{Adv}^{dist}_{\mathcal{T}_\mathcal{A}, \mathcal{T}_{\mathcal{A}^*}}(D) \leq n_e \mathbf{Adv}^{prf}_{G}(t, q),$$

which proves the lemma. □

Based on Lemma 1 and Lemma 2, we have the follow theorem.

Theorem 1 *Assume that G is a pseudorandom function family, F and H are secure message authentication codes, and F, G, and H are independent. Then AP-AKA is a secure authentication and key agreement protocol.*

Proof. The completion requirement follows directly by inspection. Now we prove that the simulatability requirement is also satisfied. Let $\mathcal{A}$ be a real world adversary and let $\mathcal{T}_\mathcal{A}$ be the transcript of $\mathcal{A}$. Since F and H are secure message authentication codes, the probability that F and H are not collision-resistant is negligible. Without loss of generality, let's assume that F and H are collision-resistant in $\mathcal{T}_\mathcal{A}$. By Lemma 2, there exists an ideal world adversary $\mathcal{A}^*$ such that for every distinguisher D with running time T,

$$|Pr(D(\mathcal{T}_\mathcal{A}) = 1|M_\mathcal{A} \wedge C_\mathcal{A}) - Pr(D(\mathcal{T}^*_\mathcal{A}) = 1|M_\mathcal{A} \wedge C_\mathcal{A})| \leq n_e \mathbf{Adv}^{prf}_{G}(t, q).$$

Thus, it follows that

$$\mathbf{Adv}^{dist}_{\mathcal{T}_\mathcal{A}, \mathcal{T}^*_\mathcal{A}}(D) \quad = \quad |Pr(D(\mathcal{T}_\mathcal{A}) = 1) - |Pr(D(\mathcal{T}^*_\mathcal{A}) = 1)|$$

$$
\begin{aligned}
&= \; |(Pr(D(\mathcal{T}_{\mathcal{A}}) = 1|M_{\mathcal{A}} \wedge C_{\mathcal{A}}) - Pr(D(\mathcal{T}^*_{\mathcal{A}}) = 1|M_{\mathcal{A}} \wedge C_{\mathcal{A}}))Pr(M_{\mathcal{A}} \wedge C_{\mathcal{A}}) \\
&\quad +(Pr(D(\mathcal{T}_{\mathcal{A}}) = 1|\overline{M}_{\mathcal{A}} \vee \overline{C}_{\mathcal{A}}) - Pr(D(\mathcal{T}^*_{\mathcal{A}}) = 1|\overline{M}_{\mathcal{A}} \vee \overline{C}_{\mathcal{A}}))Pr(\overline{M}_{\mathcal{A}} \vee \overline{C}_{\mathcal{A}})| \\
&\leq \; |(Pr(D(\mathcal{T}_{\mathcal{A}}) = 1|M_{\mathcal{A}} \wedge C_{\mathcal{A}}) - Pr(D(\mathcal{T}^*_{\mathcal{A}}) = 1|M_{\mathcal{A}} \wedge C_{\mathcal{A}})| + Pr(\overline{M}_{\mathcal{A}}) + Pr(\overline{C}_{\mathcal{A}}) \\
&\leq \; n_e \mathbf{Adv}^{prf}_G(t, q) + Pr(\overline{M}_{\mathcal{A}}) + Pr(\overline{C}_{\mathcal{A}})
\end{aligned}
$$

On the other hand,

$$
\begin{aligned}
Pr(\overline{M}_{\mathcal{A}}) &= Pr(\overline{M}_{\mathcal{A}}|C_{\mathcal{A}})Pr(C_{\mathcal{A}}) + Pr(\overline{M}_{\mathcal{A}}|\overline{C}_{\mathcal{A}})Pr(\overline{C}_{\mathcal{A}}) \\
&\leq Pr(\overline{M}_{\mathcal{A}}|C_{\mathcal{A}}) + Pr(\overline{C}_{\mathcal{A}})
\end{aligned}
$$

Therefore,

$$
\mathbf{Adv}^{dist}_{\mathcal{T}_{\mathcal{A}},\mathcal{T}}(D) \leq n_e \mathbf{Adv}^{prf}_G(t, q) + Pr(\overline{M}_{\mathcal{A}}|C_{\mathcal{A}}) + 2Pr(\overline{C}_{\mathcal{A}}).
$$

By (1), $Pr(\overline{C}_{\mathcal{A}})$ is negligible (in k). By Lemma 1, $Pr(\overline{M}_{\mathcal{A}}|C_{\mathcal{A}})$ is also negligible. Hence, $\mathbf{Adv}^{dist}_{\mathcal{T}_{\mathcal{A}},\mathcal{T}}(D)$ is negligible. □

C Security Against Network Corruption

In the formal security model as described in Appendix A, we use the ring master and the mailboxes to simulate the channels between networks; the adversary is not allowed to access the mailboxes. We now extend our formal security model to deal with *network corruptions*. In a network corruption, the adversary can access the mailboxes maintained by a network. The adversary can read messages from or write messages into the mailboxes of the corrupted network. To analyze protocols in the presence of network corruptions, we need to modify both the ideal system and the real system. The definition of security, defined in terms of completion and simulatability, will remain the same.

A modification to the ideal system is that the ideal-world adversary can execute the operation (**corrupt network**, i) to corrupt an initialized network N_i. This operation is logged in the transcript as (`corrupt network`, i). After the execution of this operation, the real-world adversary is given authentication vectors sent and received by N_i. More precisely, whenever the ring master writes a message Msg into a mailbox $InMail_{i'}$ of N_i, the ring master also delivers the message to the adversary. Consequently, a record of the form (`implementation, read message`, i, $InMail_{i'}$, Msg) is added to the transcript. Likewise, whenever the ring master takes out a message Msg from a mailbox $OutMail_{i'}$ of N_i, the ring master gives a copy of the message to the adversary. Accordingly, a record (`implementation, read message`, i, $OutMail_{i'}$, Msg) is added to the transcript.

After the corruption of N_i in the real system, the real-world adversary can also write messages to the mailboxes of N_i. Specifically, the real-world adversary can execute the operation (**write message**, i, $OutMail_{i'}$, Msg) to write a message Msg to the mailbox $OutMail_{i'}$ of N_i. This operation is logged in the transcript as (`implementation, write message`, i, $OutMail_{i'}$, Msg). Likewise, the real-world adversary can also write a message Msg into $InMail_{i'}$ of N_i by executing the operation (**write message**, i, $InMail_{i'}$, Msg). This operation is logged in the transcript as (`implementation, write message`, i, $InMail_{i'}$, Msg).

In the ideal system, we also need to modify the rules governing the legitimacy of the connection assignments. For a user instance I_{ij}, the connection assignment `compromise` is legal if either PID_{ij} is assigned to a corrupted network or the home network of U_i is corrupted. For a network instance $I_{i'j'}$, the connection assignment `compromise` is legal if either $N_{i'}$ is corrupted or $PID_{i'j'}$ is assigned to a user whose home network is corrupted.

It is clear that 3GPP AKA is insecure against network corruptions for the same reason that it was insecure against the re-direction attack. Moreover, it can also be attacked in another way. Suppose a network, say N_i, is corrupted. By writing *authentication data request* into the outgoing mailboxes of N_i, the adversary can obtain authentication vectors for any user, independent of the actual location of the user. The compromised authentication vectors allow the adversary to impersonate uncorrupted networks. This shows that the corruption of one operator's network jeopardize the entire network.

Definition 5 *Let I_{ij} be a user instance which has accepted in the real system. We say that I_{ij} is reliable if, at the time of acceptance, PID_{ij} is not assigned to a corrupted network and the home network of U_i is not corrupted. Similarly, Let $I_{i'j'}$ be a network instance which has accepted in the real system. We say that $I_{i'j'}$ is reliable if, at the time of acceptance, $N_{i'}$ is not corrupted and $PID_{i'j'}$ is assigned to a user whose home network is not corrupted.*

Definition 6 *Let $\mathcal{A}$ be a real-world adversary and let $\mathcal{T}_{\mathcal{A}}$ be the transcript of $\mathcal{A}$. We say that $\mathcal{T}_{\mathcal{A}}$ is authentic against network corruptions if the following two conditions are satisfied: i) for every accepted instance I_{ij}, PID_{ij} is assigned to an initialized entity whose role is not equal to $role_i$; and ii) for every reliable instance I_{ij}, the stimulus on I_{ij} is output by an instance compatible with I_{ij}.*

It can be proved that Lemma 1 holds for real-world transcripts which are authentic against network corruptions. Similar to Lemma 2, we have the following lemma:

Lemma 3 *Let $\mathcal{A}$ be a real-world adversary and let $\mathcal{T}_{\mathcal{A}}$ be the transcript of $\mathcal{A}$. Assume that $\mathcal{T}_{\mathcal{A}}$ is authentic against network corruptions and is collision-free. Also assume that G is a pseudorandom function family, independent of F and H, and F and H are collision-resistant in $\mathcal{T}_{\mathcal{A}}$. Then there exists an ideal-world adversary $\mathcal{A}^*$ such that, for every distinguisher D with running time T,*

$$\mathbf{Adv}^{dist}_{\mathcal{T}_{\mathcal{A}},\mathcal{T}_{\mathcal{A}^*}}(D) \leq n_e \mathbf{Adv}^{prf}_{G}(t,q),$$

where, n_e is the number of user entities initialized by $\mathcal{A}$ and n_i is the number of instances initialized by $\mathcal{A}$, $t = O(T), q = O(n_i)$.

Proof. Similar to the proof of Lemma 2, we construct a simulator, denoted by $Simulator_c$, for the real world-adversary $\mathcal{A}$. $Simulator_c$ basically has $\mathcal{A}^*$ run the adversary $\mathcal{A}$ just as in the real system. For any `implementation` record in the real-world transcript, $\mathcal{A}^*$ copies this record into the ideal-world transcript by issuing an *implementation* operation. Corresponding to each (`start session,` i,j) record that $\mathcal{A}$'s action cause to be placed in the real-world transcript, $\mathcal{A}^*$ computes a connection assignment, and the ring master in the ideal system substitutes the session key SK_{ij} by an idealized session key K_{ij}, which is a random number when the connection assignment is not *compromise*. When $\mathcal{A}^*$ chooses to make the connection assignment *compromise* for I_{ij}, it extracts the session key SK_{ij}

from the instance I_{ij} and executes the operation (*start session*, $i, j, compromise[SK_{ij}]$). Corresponding to each (`abort session,` i, j) record that $\mathcal{A}$'s action cause to be placed in the real-world transcript, $\mathcal{A}^*$ executes the operation (*abort session*, i, j). For *application* operations, the ring master in the ideal system makes the evaluations using the idealized session keys. This way, we have an ideal-world adversary whose transcript is almost identical to the transcript of the real-world adversary $\mathcal{A}$. The differences exist in some of the `application` records. In the following, we show that the connection assignments made by $\mathcal{A}^*$ are legal and the differences between the two transcript are not detectable (computationally).

Case 1. Let $I_{i'j'}$ be a network instance which has just received the message (*user data response*, RN, U_{MAC}) and has accepted. If $N_{i'}$ has been corrupted, the adversary $\mathcal{A}^*$ extracts $SK_{i'j'}$ from $I_{i'j'}$ and makes the connection assignment *compromise* for $I_{i'j'}$. If $N_{i'}$ has not been corrupted, $I_{i'j'}$ is a reliable instance. So the stimulus on $I_{i'j'}$ is output by a compatible user instance I_{ij}. In this case, $\mathcal{A}^*$ makes the connection assignment (*create*, i, j) and the ring master substitutes the session key $SK_{i'j'}$ by a random number $K_{i'j'}$. Under the assumption that $\mathcal{T}_{\mathcal{A}}$ is collision-free and F is collision-resistant in $\mathcal{T}_{\mathcal{A}}$, the message (*user data response*, RN, U_{MAC}) could not be stimulus on network instances other than $I_{i'j'}$. So the connection assignment (*create*, i, j) was not made before.

Case 2. Let I_{ij} be a user instance which has just received the message (*user authentication request*, $RAND, AUTH$) and has accepted. Assume that PID_{ij} is assigned to a network $N_{i'}$. Let idx denote the index included in $AUTH$.

Case 2a: $idx = 0$. In this case, $N_{i'}$ is the home network of U_i. If there is no network instance which has been isolated for I_{ij}, we claim that $N_{i'}$ has been corrupted. If $N_{i'}$ is not corrupted, I_{ij} is reliable and the stimulus on I_{ij} is output by an instance $I_{i'j'}$ of $N_{i'}$. According to Case 1, $I_{i'j'}$ has been isolated for I_{ij}, which leads to a contradiction. So it is legal to make the connection assignment `compromise` for I_{ij}. If there exists a network instance $I_{i'j'}$ which has been isolated for I_{ij}, then stimulus on $I_{i'j'}$ was output by I_{ij}. Moreover, at the point of time when $I_{i'j'}$ accepted, the network $N_{i'}$ was not corrupted. So it is legal for the adversary $\mathcal{A}^*$ to make the connection assignment (`connect`, i', j'). Accordingly, the ring master set the key K_{ij} to $K_{i'j'}$.

Case 2b: $idx > 0$. If either $N_{i'}$ or the home network of U_i is corrupted, the adversary $\mathcal{A}^*$ extracts the session key SK_{ij} and makes the connection assignment *compromise* for I_{ij}. Otherwise, the instance I_{ij} is reliable and the stimulus, i.e., $RAND$ and $AUTH$, on I_{ij} was output by a compatible instance $I_{i'j'}$. In this case, the adversary $\mathcal{A}^*$ makes the connection assignment (*create*, i', j') for I_{ij} and the ring master substitutes the session key SK_{ij} by a random number K_{ij}. Since $\mathcal{T}_{\mathcal{A}}$ is collision-free and H is collision-resistant in $\mathcal{T}_{\mathcal{A}}$, $RAND$ and $AUTH$ could not be stimulus on any instance other than I_{ij}, which implies that the connection assignment (*create*, i', j') was not made before.

Case 3. Let $I_{i'j'}$ be a network instance which has just received the message (*user authentication response*, RES) and has accepted. Assume that $PID_{i'j'}$ is assigned to a user entity U_i. If there exists a user instance I_{ij} which has been isolated for $I_{i'j'}$, $\mathcal{A}^*$ makes the connection assignment (*connect*, i, j) and the ring master set the $K_{i'j'}$ to K_{ij}. If there is no user instance which has been isolated for $I_{i'j'}$, then the network $N_{i'}$ or the home network of U_i must be corrupted. Otherwise, $I_{i'j'}$ is reliable and the stimulus on $I_{i'j'}$ is output a compatible instance of U_i. According to Case 1b, there exists a compatible instance of

U_i which is isolated for $I_{i'j'}$, which contradicts to our assumption. So it is legal for the adversary $\mathcal{A}^*$ to make the connection assignment *compromise* for $I_{i'j'}$.

The above analysis shows that, whenever an instance I_{ij} is isolated for a compatible instance $I_{i'j'}$, the instance $I_{i'j'}$ has not been or ever will be *compromised.* The only possible outcomes for $I_{i'j'}$ are that either it aborts or it connects to I_{ij}. Following the proof of Lemma 2, it can be proved that for every distinguisher D with running time T,

$$\mathbf{Adv}^{dist}_{\mathcal{T}_{\mathcal{A}},\mathcal{T}_{\mathcal{A}^*}}(D) \leq n_e \mathbf{Adv}^{prf}_{G}(t,q).$$

which completes the proof of the lemma. □

Based on Lemma 1 and Lemma 3, we have the following theorem.

Theorem 2 *Assume that G is a pseudorandom function family, F and H are secure message authentication codes, and F, G, and H are independent. Then AP-AKA is an authentication and key agreement protocol secure against network corruptions.*

In the ideal system, the adversary $\mathcal{A}^*$ may compromise the session keys of those users who either subscribe to or roam into a corrupted network. The adversary $\mathcal{A}^*$ can not impersonate uncorrupted networks to authenticate users whose home networks are not corrupted. By Theorem 2, the real-world adversary $\mathcal{A}$ could not do more harm than the ideal-world adversary $\mathcal{A}^*$. Hence, the threat of network corruptions is greatly reduced for AP-AKA in comparison with 3GPP AKA.