Stream: Internet Engineering Task Force (IETF)
RFC: 9048
Updates: 4187, 5448
Category: Informational
Published: October 2021
ISSN: 2070-1721
Authors: J. Arkko, *Ericsson*; V. Lehtovirta, *Ericsson*; V. Torvinen, *Ericsson*; P. Eronen, *Independent*

RFC 9048
Improved Extensible Authentication Protocol Method for 3GPP Mobile Network Authentication and Key Agreement (EAP-AKA')

Abstract

The 3GPP mobile network Authentication and Key Agreement (AKA) is an authentication mechanism for devices wishing to access mobile networks. RFC 4187 (EAP-AKA) made the use of this mechanism possible within the Extensible Authentication Protocol (EAP) framework. RFC 5448 (EAP-AKA') was an improved version of EAP-AKA.

This document is the most recent specification of EAP-AKA', including, for instance, details about and references related to operating EAP-AKA' in 5G networks.

EAP-AKA' differs from EAP-AKA by providing a key derivation function that binds the keys derived within the method to the name of the access network. The key derivation function has been defined in the 3rd Generation Partnership Project (3GPP). EAP-AKA' allows its use in EAP in an interoperable manner. EAP-AKA' also updates the algorithm used in hash functions, as it employs SHA-256 / HMAC-SHA-256 instead of SHA-1 / HMAC-SHA-1, which is used in EAP-AKA.

This version of the EAP-AKA' specification defines the protocol behavior for both 4G and 5G deployments, whereas the previous version defined protocol behavior for 4G deployments only. While EAP-AKA' as defined in RFC 5448 is not obsolete, this document defines the most recent and fully backwards-compatible specification of EAP-AKA'. This document updates both RFCs 4187 and 5448.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

Copyright Notice

Table of Contents

1. Introduction

The 3GPP mobile network Authentication and Key Agreement (AKA) is an authentication mechanism for devices wishing to access mobile networks. [RFC4187] (EAP-AKA) made the use of this mechanism possible within the Extensible Authentication Protocol (EAP) framework [RFC3748].

EAP-AKA' is an improved version of EAP-AKA. EAP-AKA' was defined in RFC 5448 [RFC5448], and it updated EAP-AKA [RFC4187].

This document is the most recent specification of EAP-AKA', including, for instance, details about and references related to operating EAP-AKA' in 5G networks. This document does not obsolete RFC 5448; however, this document is the most recent and fully backwards-compatible specification.

EAP-AKA' is commonly implemented in mobile phones and network equipment. It can be used for authentication to gain network access via Wireless LAN networks and, with 5G, also directly to mobile networks.

EAP-AKA' differs from EAP-AKA by providing a different key derivation function. This function binds the keys derived within the method to the name of the access network. This limits the effects of compromised access network nodes and keys. EAP-AKA' also updates the algorithm used for hash functions.

The EAP-AKA' method employs the derived keys CK' and IK' from the 3GPP specification [TS-3GPP.33.402] and updates the hash function that is used to SHA-256 [FIPS.180-4] and HMAC to HMAC-SHA-256. Otherwise, EAP-AKA' is equivalent to EAP-AKA. Given that a different EAP method Type value is used for EAP-AKA and EAP-AKA', a mutually supported method may be negotiated using the standard mechanisms in EAP [RFC3748].

> Note that any change of the key derivation must be unambiguous to both sides in the protocol. That is, it must not be possible to accidentally connect old equipment to new equipment and get the key derivation wrong or to attempt to use incorrect keys without getting a proper error message. See Appendix C for further information.
>
> Note also that choices in authentication protocols should be secure against bidding down attacks that attempt to force the participants to use the least secure function. See Section 4 for further information.

This specification makes the following changes from RFC 5448:

- Updates the reference that specifies how the Network Name field is constructed in the protocol. This update ensures that EAP-AKA' is compatible with 5G deployments. RFC 5448 referred to the Release 8 version of [TS-3GPP.24.302]. This document points to the first 5G version, Release 16.
- Specifies how EAP and EAP-AKA' use identifiers in 5G. Additional identifiers are introduced in 5G, and for interoperability, it is necessary that the right identifiers are used as inputs in the key derivation. In addition, for identity privacy it is important that when privacy-friendly identifiers in 5G are used, no trackable, permanent identifiers are passed in EAP-AKA', either.
- Specifies session identifiers and other exported parameters, as those were not specified in [RFC5448] despite requirements set forward in [RFC5247] to do so. Also, while [RFC5247] specified session identifiers for EAP-AKA, it only did so for the full authentication case, not for the case of fast re-authentication.

- Updates the requirements on generating pseudonym usernames and fast re-authentication identities to ensure identity privacy.
- Describes what has been learned about any vulnerabilities in AKA or EAP-AKA'.
- Describes the privacy and pervasive monitoring considerations related to EAP-AKA'.
- Adds summaries of the attributes.

Some of the updates are small. For instance, the reference update to [TS-3GPP.24.302] does not change the 3GPP specification number, only the version. But this reference is crucial for the correct calculation of the keys that result from running the EAP-AKA' method, so an RFC update pointing to the newest version was warranted.

> Note: Any further updates in 3GPP specifications that affect, for instance, key derivation is something that EAP-AKA' implementations need to take into account. Upon such updates, there will be a need to update both this specification and the implementations.

It is an explicit non-goal of this specification to include any other technical modifications, addition of new features, or other changes. The EAP-AKA' base protocol is stable and needs to stay that way. If there are any extensions or variants, those need to be proposed as standalone extensions or even as different authentication methods.

The rest of this specification is structured as follows. Section 3 defines the EAP-AKA' method. Section 4 adds support to EAP-AKA to prevent bidding down attacks from EAP-AKA'. Section 5 specifies requirements regarding the use of peer identities, including how 5G identifiers are used in the EAP-AKA' context. Section 6 specifies which parameters EAP-AKA' exports out of the method. Section 7 explains the security differences between EAP-AKA and EAP-AKA'. Section 8 describes the IANA considerations, and Appendix A and Appendix B explain the updates to RFC 5448 (EAP-AKA') and RFC 4187 (EAP-AKA) that have been made in this specification. Appendix C explains some of the design rationale for creating EAP-AKA'. Finally, Appendix D provides test vectors.

2. Requirements Language

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. EAP-AKA'

EAP-AKA' is an EAP method that follows the EAP-AKA specification [RFC4187] in all respects except the following:

- It uses the Type code 0x32, not 0x17 (which is used by EAP-AKA).
- It carries the AT_KDF_INPUT attribute, as defined in Section 3.1, to ensure that both the peer and server know the name of the access network.

- It supports key derivation function negotiation via the AT_KDF attribute (Section 3.2) to allow for future extensions.
- It calculates keys as defined in Section 3.3, not as defined in EAP-AKA.
- It employs SHA-256 / HMAC-SHA-256 [FIPS.180-4], not SHA-1 / HMAC-SHA-1 [RFC2104] (see Section 3.4).

Figure 1 shows an example of the authentication process. Each message AKA'-Challenge and so on represents the corresponding message from EAP-AKA, but with the EAP-AKA' Type code. The definition of these messages, along with the definition of attributes AT_RAND, AT_AUTN, AT_MAC, and AT_RES can be found in [RFC4187].

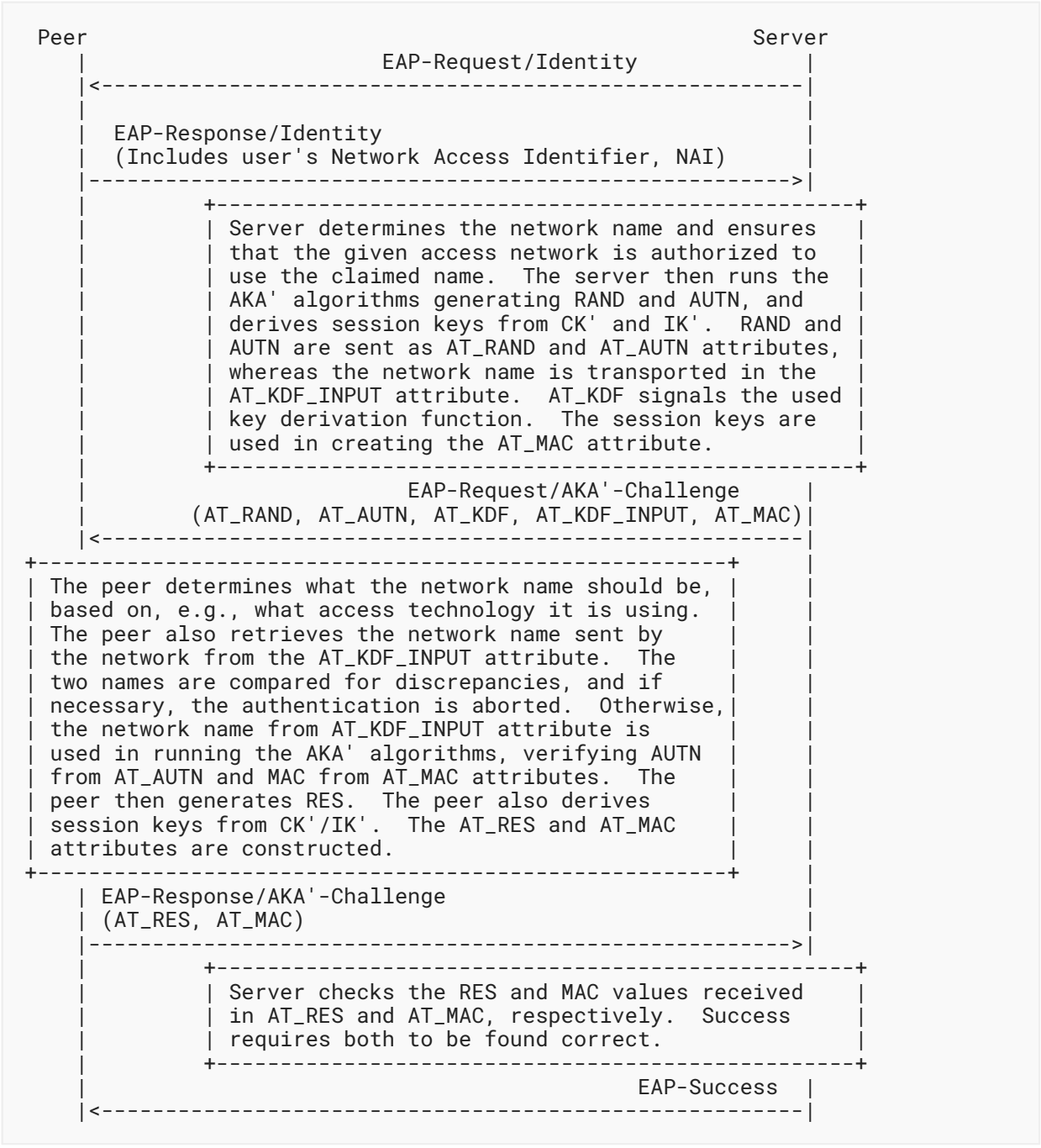


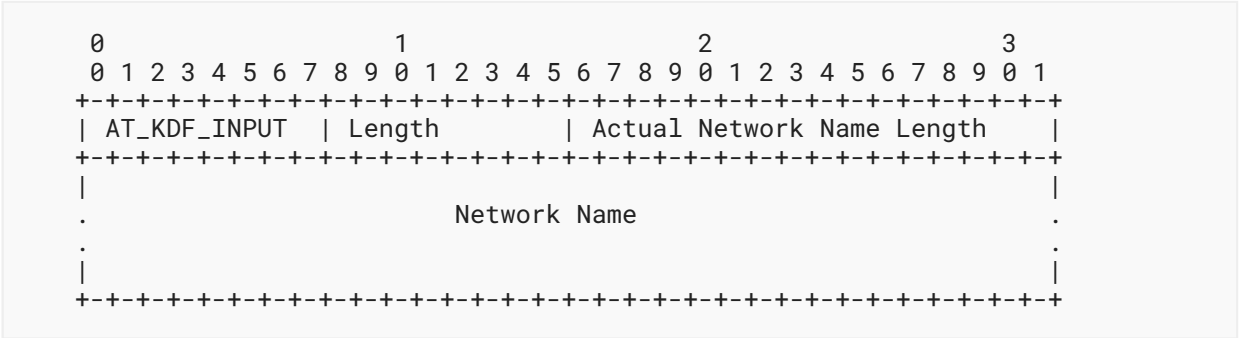
Figure 1: EAP-AKA' Authentication Process

EAP-AKA' can operate on the same credentials as EAP-AKA and employ the same identities. However, EAP-AKA' employs different leading characters than EAP-AKA for the conventions given in Section 4.1.1 of [RFC4187] for usernames based on International Mobile Subscriber Identifier (IMSI). For 4G networks, EAP-AKA' **MUST** use the leading character "6" (ASCII 36 hexadecimal) instead of "0" for IMSI-based permanent usernames. For 5G networks, the leading

character "6" is not used for IMSI-based permanent usernames. Identifier usage in 5G is specified in Section 5.3. All other usage and processing of the leading characters, usernames, and identities is as defined by EAP-AKA [RFC4187]. For instance, the pseudonym and fast re-authentication usernames need to be constructed so that the server can recognize them. As an example, a pseudonym could begin with a leading "7" character (ASCII 37 hexadecimal) and a fast re-authentication username could begin with "8" (ASCII 38 hexadecimal). Note that a server that implements only EAP-AKA may not recognize these leading characters. According to Section 4.1.4 of [RFC4187], such a server will re-request the identity via the EAP-Request/AKA-Identity message, making obvious to the peer that EAP-AKA and associated identity are expected.

3.1. AT_KDF_INPUT

The format of the AT_KDF_INPUT attribute is shown below.



The fields are as follows:

AT_KDF_INPUT
: This is set to 23.

Length
: The length of the attribute, calculated as defined in [RFC4187], Section 8.1.

Actual Network Name Length
: This is a 2-byte actual length field, needed due to the requirement that the previous field is expressed in multiples of 4 bytes per the usual EAP-AKA rules. The Actual Network Name Length field provides the length of the network name in bytes.

Network Name
: This field contains the network name of the access network for which the authentication is being performed. The name does not include any terminating null characters. Because the length of the entire attribute must be a multiple of 4 bytes, the sender pads the name with 1, 2, or 3 bytes of all zero bits when necessary.

Only the server sends the AT_KDF_INPUT attribute. The value is sent as specified in [TS-3GPP.24.302] for both non-3GPP access networks and for 5G access networks. Per [TS-3GPP.33.402], the server always verifies the authorization of a given access network to use a particular name before sending it to the peer over EAP-AKA'. The value of the AT_KDF_INPUT attribute from the

server **MUST** be non-empty, with a greater than zero length in the Actual Network Name Length field. If the AT_KDF_INPUT attribute is empty, the peer behaves as if AUTN had been incorrect and authentication fails. See Section 3 and Figure 3 of [RFC4187] for an overview of how authentication failures are handled.

In addition, the peer **MAY** check the received value against its own understanding of the network name. Upon detecting a discrepancy, the peer either warns the user and continues, or fails the authentication process. More specifically, the peer **SHOULD** have a configurable policy that it can follow under these circumstances. If the policy indicates that it can continue, the peer **SHOULD** log a warning message or display it to the user. If the peer chooses to proceed, it **MUST** use the network name as received in the AT_KDF_INPUT attribute. If the policy indicates that the authentication should fail, the peer behaves as if AUTN had been incorrect and authentication fails.

The Network Name field contains a UTF-8 string. This string **MUST** be constructed as specified in [TS-3GPP.24.302] for "Access Network Identity". The string is structured as fields separated by colons (:). The algorithms and mechanisms to construct the identity string depend on the used access technology.

On the network side, the network name construction is a configuration issue in an access network and an authorization check in the authentication server. On the peer, the network name is constructed based on the local observations. For instance, the peer knows which access technology it is using on the link, it can see information in a link-layer beacon, and so on. The construction rules specify how this information maps to an access network name. Typically, the network name consists of the name of the access technology or the name of the access technology followed by some operator identifier that was advertised in a link-layer beacon. In all cases, [TS-3GPP.24.302] is the normative specification for the construction in both the network and peer side. If the peer policy allows running EAP-AKA' over an access technology for which that specification does not provide network name construction rules, the peer **SHOULD** rely only on the information from the AT_KDF_INPUT attribute and not perform a comparison.

If a comparison of the locally determined network name and the one received over EAP-AKA' is performed on the peer, it **MUST** be done as follows. First, each name is broken down to the fields separated by colons. If one of the names has more colons and fields than the other one, the additional fields are ignored. The remaining sequences of fields are compared, and they match only if they are equal character by character. This algorithm allows a prefix match where the peer would be able to match "", "FOO", and "FOO:BAR" against the value "FOO:BAR" received from the server. This capability is important in order to allow possible updates to the specifications that dictate how the network names are constructed. For instance, if a peer knows that it is running on access technology "FOO", it can use the string "FOO" even if the server uses an additional, more accurate description, e.g., "FOO:BAR", that contains more information.

The allocation procedures in [TS-3GPP.24.302] ensure that conflicts potentially arising from using the same name in different types of networks are avoided. The specification also has detailed rules about how a client can determine these based on information available to the client, such

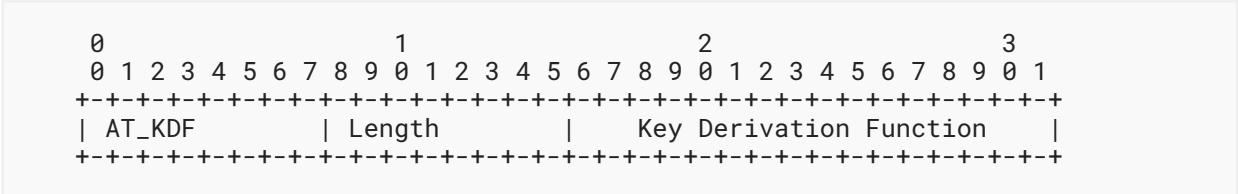
as the type of protocol used to attach to the network, beacons sent out by the network, and so on. Information that the client cannot directly observe (such as the type or version of the home network) is not used by this algorithm.

The AT_KDF_INPUT attribute **MUST** be sent and processed as explained above when AT_KDF attribute has the value 1. Future definitions of new AT_KDF values **MUST** define how this attribute is sent and processed.

3.2. AT_KDF

AT_KDF is an attribute that the server uses to reference a specific key derivation function. It offers a negotiation capability that can be useful for future evolution of the key derivation functions.

The format of the AT_KDF attribute is shown below.



The fields are as follows:

AT_KDF
: This is set to 24.

Length
: The length of the attribute, calculated as defined in [RFC4187], Section 8.1. For AT_KDF, the Length field **MUST** be set to 1.

Key Derivation Function
: An enumerated value representing the key derivation function that the server (or peer) wishes to use. Value 1 represents the default key derivation function for EAP-AKA', i.e., employing CK' and IK' as defined in Section 3.3.

Servers **MUST** send one or more AT_KDF attributes in the EAP-Request/AKA'-Challenge message. These attributes represent the desired functions ordered by preference, the most preferred function being the first attribute.

Upon receiving a set of these attributes, if the peer supports and is willing to use the key derivation function indicated by the first attribute, the function is taken into use without any further negotiation. However, if the peer does not support this function or is unwilling to use it, it does not process the received EAP-Request/AKA'-Challenge in any way except by responding with the EAP-Response/AKA'-Challenge message that contains only one attribute, AT_KDF with the value set to the selected alternative. If there is no suitable alternative, the peer behaves as if AUTN had been incorrect and authentication fails (see Figure 3 of [RFC4187]). The peer fails the

authentication also if there are any duplicate values within the list of AT_KDF attributes (except where the duplication is due to a request to change the key derivation function; see below for further information).

Upon receiving an EAP-Response/AKA'-Challenge with AT_KDF from the peer, the server checks that the suggested AT_KDF value was one of the alternatives in its offer. The first AT_KDF value in the message from the server is not a valid alternative since the peer should have accepted it without further negotiation. If the peer has replied with the first AT_KDF value, the server behaves as if AT_MAC of the response had been incorrect and fails the authentication. For an overview of the failed authentication process in the server side, see Section 3 and Figure 2 of [RFC4187]. Otherwise, the server re-sends the EAP-Response/AKA'-Challenge message, but adds the selected alternative to the beginning of the list of AT_KDF attributes and retains the entire list following it. Note that this means that the selected alternative appears twice in the set of AT_KDF values. Responding to the peer's request to change the key derivation function is the only legal situation where such duplication may occur.

When the peer receives the new EAP-Request/AKA'-Challenge message, it **MUST** check that the requested change, and only the requested change, occurred in the list of AT_KDF attributes. If so, it continues with processing the received EAP-Request/AKA'-Challenge as specified in [RFC4187] and Section 3.1 of this document. If not, it behaves as if AT_MAC had been incorrect and fails the authentication. If the peer receives multiple EAP-Request/AKA'-Challenge messages with differing AT_KDF attributes without having requested negotiation, the peer **MUST** behave as if AT_MAC had been incorrect and fail the authentication.

Note that the peer may also request sequence number resynchronization [RFC4187]. This happens after AT_KDF negotiation has already completed. That is, the EAP-Request/AKA'-Challenge and, possibly, the EAP-Response/AKA'-Challenge messages are exchanged first to determine a mutually acceptable key derivation function, and only then is the possible AKA'-Synchronization-Failure message sent. The AKA'-Synchronization-Failure message is sent as a response to the newly received EAP-Request/AKA'-Challenge, which is the last message of the AT_KDF negotiation. Note that if the first proposed KDF is acceptable, then the first EAP-Request/AKA'-Challenge message is also the last message. The AKA'-Synchronization-Failure message **MUST** contain the AUTS parameter as specified in [RFC4187] and a copy the AT_KDF attributes as they appeared in the last message of the AT_KDF negotiation. If the AT_KDF attributes are found to differ from their earlier values, the peer and server **MUST** behave as if AT_MAC had been incorrect and fail the authentication.

3.3. Key Derivation

Both the peer and server **MUST** derive the keys as follows.

AT_KDF parameter has the value 1
: In this case, MK is derived and used as follows:

```
MK = PRF'(IK'|CK',"EAP-AKA'"|Identity)
K_encr = MK[0..127]
K_aut  = MK[128..383]
K_re   = MK[384..639]
MSK    = MK[640..1151]
EMSK   = MK[1152..1663]
```

Here [n..m] denotes the substring from bit n to m, including bits n and m. PRF' is a new pseudorandom function specified in Section 3.4. The first 1664 bits from its output are used for K_encr (encryption key, 128 bits), K_aut (authentication key, 256 bits), K_re (re-authentication key, 256 bits), MSK (Master Session Key, 512 bits), and EMSK (Extended Master Session Key, 512 bits). These keys are used by the subsequent EAP-AKA' process. K_encr is used by the AT_ENCR_DATA attribute, and K_aut by the AT_MAC attribute. K_re is used later in this section. MSK and EMSK are outputs from a successful EAP method run [RFC3748].

IK' and CK' are derived as specified in [TS-3GPP.33.402]. The functions that derive IK' and CK' take the following parameters: CK and IK produced by the AKA algorithm, and value of the Network Name field comes from the AT_KDF_INPUT attribute (without length or padding).

The value "EAP-AKA'" is an eight-characters-long ASCII string. It is used as is, without any trailing NUL characters.

Identity is the peer identity as specified in Section 7 of [RFC4187] and in Section 5.3.2 of in this document for the 5G cases.

When the server creates an AKA challenge and corresponding AUTN, CK, CK', IK, and IK' values, it **MUST** set the Authentication Management Field (AMF) separation bit to 1 in the AKA algorithm [TS-3GPP.33.102]. Similarly, the peer **MUST** check that the AMF separation bit is set to 1. If the bit is not set to 1, the peer behaves as if the AUTN had been incorrect and fails the authentication.

On fast re-authentication, the following keys are calculated:

```
MK = PRF'(K_re,"EAP-AKA' re-auth"|Identity|counter|NONCE_S)
MSK  = MK[0..511]
EMSK = MK[512..1023]
```

MSK and EMSK are the resulting 512-bit keys, taking the first 1024 bits from the result of PRF'. Note that K_encr and K_aut are not re-derived on fast re-authentication. K_re is the re-authentication key from the preceding full authentication and stays unchanged over any fast re-authentication(s) that may happen based on it. The value "EAP-AKA' re-auth" is a sixteen-characters-long ASCII string, again represented without any trailing NUL characters. Identity is the fast re-authentication identity, counter is the value from the AT_COUNTER attribute, NONCE_S is the nonce value from the AT_NONCE_S attribute, all as specified in Section 7 of [RFC4187]. To prevent the use of compromised keys in other places, it is forbidden to change the network name when going from the full to the fast re-authentication process. The peer **SHOULD NOT** attempt fast re-authentication when it knows that the network name in the

current access network is different from the one in the initial, full authentication. Upon seeing a re-authentication request with a changed network name, the server **SHOULD** behave as if the re-authentication identifier had been unrecognized, and fall back to full authentication. The server observes the change in the name by comparing where the fast re-authentication and full authentication EAP transactions were received at the Authentication, Authorization, and Accounting (AAA) protocol level.

AT_KDF has any other value
: Future variations of key derivation functions may be defined, and they will be represented by new values of AT_KDF. If the peer does not recognize the value, it cannot calculate the keys and behaves as explained in Section 3.2.

AT_KDF is missing
: The peer behaves as if the AUTN had been incorrect and **MUST** fail the authentication.

If the peer supports a given key derivation function but is unwilling to perform it for policy reasons, it refuses to calculate the keys and behaves as explained in Section 3.2.

3.4. Hash Functions

EAP-AKA' uses SHA-256 / HMAC-SHA-256, not SHA-1 / HMAC-SHA-1 (see [FIPS.180-4] and [RFC2104]) as in EAP-AKA. This requires a change to the pseudorandom function (PRF) as well as the AT_MAC and AT_CHECKCODE attributes.

3.4.1. PRF'

The PRF' construction is the same one IKEv2 uses (see Section 2.13 of [RFC7296]; the definition of this function has not changed since [RFC4306], which was referenced by [RFC5448]). The function takes two arguments. K is a 256-bit value and S is a byte string of arbitrary length. PRF' is defined as follows:

```
PRF'(K,S) = T1 | T2 | T3 | T4 | ...

   where:
   T1 = HMAC-SHA-256 (K, S | 0x01)
   T2 = HMAC-SHA-256 (K, T1 | S | 0x02)
   T3 = HMAC-SHA-256 (K, T2 | S | 0x03)
   T4 = HMAC-SHA-256 (K, T3 | S | 0x04)
   ...
```

PRF' produces as many bits of output as is needed. HMAC-SHA-256 is the application of HMAC [RFC2104] to SHA-256.

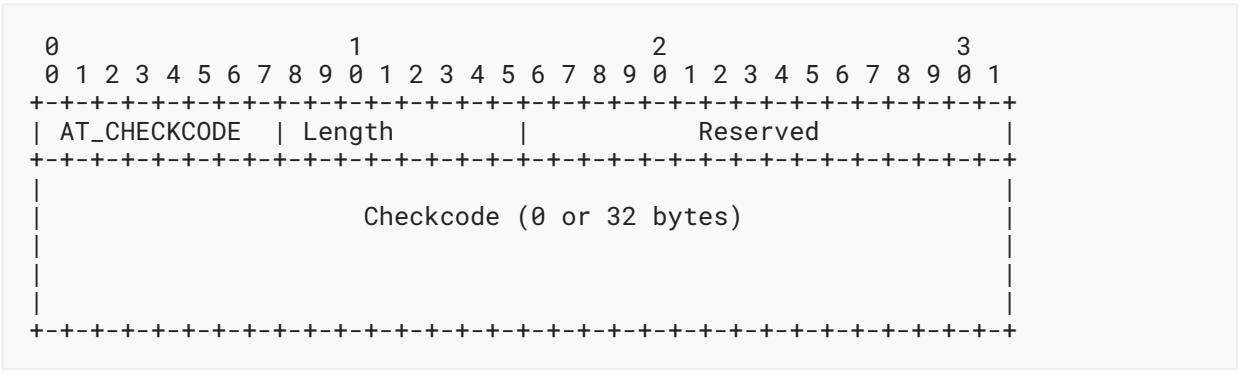
3.4.2. AT_MAC

When used within EAP-AKA', the AT_MAC attribute is changed as follows. The MAC algorithm is HMAC-SHA-256-128, a keyed hash value. The HMAC-SHA-256-128 value is obtained from the 32-byte HMAC-SHA-256 value by truncating the output to the first 16 bytes. Hence, the length of the MAC is 16 bytes.

Otherwise, the use of AT_MAC in EAP-AKA' follows Section 10.15 of [RFC4187].

3.4.3. AT_CHECKCODE

When used within EAP-AKA', the AT_CHECKCODE attribute is changed as follows. First, a 32-byte value is needed to accommodate a 256-bit hash output:



Second, the checkcode is a hash value, calculated with SHA-256 [FIPS.180-4], over the data specified in Section 10.13 of [RFC4187].

3.5. Summary of Attributes for EAP-AKA'

Table 1 identifies which attributes may be found in which kinds of messages, and in what quantity.

Messages are denoted with numbers as follows:

1 EAP-Request/AKA-Identity

2 EAP-Response/AKA-Identity

3 EAP-Request/AKA-Challenge

4 EAP-Response/AKA-Challenge

5 EAP-Request/AKA-Notification

6 EAP-Response/AKA-Notification

7 EAP-Response/AKA-Client-Error

8 EAP-Request/AKA-Reauthentication

9 EAP-Response/AKA-Reauthentication

10 EAP-Response/AKA-Authentication-Reject

11 EAP-Response/AKA-Synchronization-Failure

The column denoted with "E" indicates whether the attribute is a nested attribute that **MUST** be included within AT_ENCR_DATA.

In addition, the numbered columns indicate the quantity of the attribute within the message as follows:

"0" Indicates that the attribute **MUST NOT** be included in the message.

"1" Indicates that the attribute **MUST** be included in the message.

"0-1" Indicates that the attribute is sometimes included in the message

"0+" Indicates that zero or more copies of the attribute **MAY** be included in the message.

"1+" Indicates that there **MUST** be at least one attribute in the message but more than one **MAY** be included in the message.

"0*" Indicates that the attribute is not included in the message in cases specified in this document, but **MAY** be included in the future versions of the protocol.

The attribute table is shown below. The table is largely the same as in the EAP-AKA attribute table ([RFC4187], Section 10.1), but changes how many times AT_MAC may appear in an EAP-Response/AKA'-Challenge message as it does not appear there when AT_KDF has to be sent from the peer to the server. The table also adds the AT_KDF and AT_KDF_INPUT attributes.

Attribute	1	2	3	4	5	6	7	8	9	10	11	E
AT_PERMANENT_ID_REQ	0-1	0	0	0	0	0	0	0	0	0	0	N
AT_ANY_ID_REQ	0-1	0	0	0	0	0	0	0	0	0	0	N
AT_FULLAUTH_ID_REQ	0-1	0	0	0	0	0	0	0	0	0	0	N
AT_IDENTITY	0	0-1	0	0	0	0	0	0	0	0	0	N
AT_RAND	0	0	1	0	0	0	0	0	0	0	0	N
AT_AUTN	0	0	1	0	0	0	0	0	0	0	0	N
AT_RES	0	0	0	1	0	0	0	0	0	0	0	N
AT_AUTS	0	0	0	0	0	0	0	0	0	0	1	N
AT_NEXT_PSEUDONYM	0	0	0-1	0	0	0	0	0	0	0	0	Y
AT_NEXT_REAUTH_ID	0	0	0-1	0	0	0	0	0-1	0	0	0	Y
AT_IV	0	0	0-1	0*	0-1	0-1	0	1	1	0	0	N
AT_ENCR_DATA	0	0	0-1	0*	0-1	0-1	0	1	1	0	0	N

Attribute	1	2	3	4	5	6	7	8	9	10	11	E
AT_PADDING	0	0	0-1	0*	0-1	0-1	0	0-1	0-1	0	0	Y
AT_CHECKCODE	0	0	0-1	0-1	0	0	0	0-1	0-1	0	0	N
AT_RESULT_IND	0	0	0-1	0-1	0	0	0	0-1	0-1	0	0	N
AT_MAC	0	0	1	0-1	0-1	0-1	0	1	1	0	0	N
AT_COUNTER	0	0	0	0	0-1	0-1	0	1	1	0	0	Y
AT_COUNTER_TOO_SMALL	0	0	0	0	0	0	0	0	0-1	0	0	Y
AT_NONCE_S	0	0	0	0	0	0	0	1	0	0	0	Y
AT_NOTIFICATION	0	0	0	0	1	0	0	0	0	0	0	N
AT_CLIENT_ERROR_CODE	0	0	0	0	0	0	1	0	0	0	0	N
AT_KDF	0	0	1+	0+	0	0	0	0	0	0	1+	N
AT_KDF_INPUT	0	0	1	0	0	0	0	0	0	0	0	N

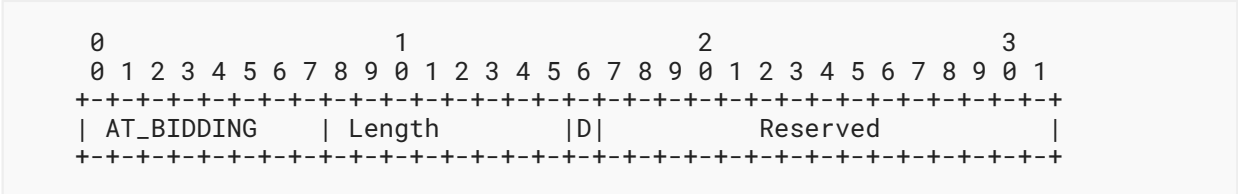
Table 1: The Attribute Table

4. Bidding Down Prevention for EAP-AKA

As discussed in [RFC3748], negotiation of methods within EAP is insecure. That is, a man-in-the-middle attacker may force the endpoints to use a method that is not the strongest that they both support. This is a problem, as we expect EAP-AKA and EAP-AKA' to be negotiated via EAP.

In order to prevent such attacks, this RFC specifies a mechanism for EAP-AKA that allows the endpoints to securely discover the capabilities of each other. This mechanism comes in the form of the AT_BIDDING attribute. This allows both endpoints to communicate their desire and support for EAP-AKA' when exchanging EAP-AKA messages. This attribute is not included in EAP-AKA' messages. It is only included in EAP-AKA messages, which are protected with the AT_MAC attribute. This approach is based on the assumption that EAP-AKA' is always preferable (see Section 7). If during the EAP-AKA authentication process it is discovered that both endpoints would have been able to use EAP-AKA', the authentication process **SHOULD** be aborted, as a bidding down attack may have happened.

The format of the AT_BIDDING attribute is shown below.



The fields are as follows:

AT_BIDDING
: This is set to 136.

Length
: The length of the attribute, calculated as defined in [RFC4187], Section 8.1. For AT_BIDDING, the Length **MUST** be set to 1.

D
: This bit is set to 1 if the sender supports EAP-AKA', is willing to use it, and prefers it over EAP-AKA. Otherwise, it should be set to zero.

Reserved
: This field **MUST** be set to zero when sent and ignored on receipt.

The server sends this attribute in the EAP-Request/AKA-Challenge message. If the peer supports EAP-AKA', it compares the received value to its own capabilities. If it turns out that both the server and peer would have been able to use EAP-AKA' and preferred it over EAP-AKA, the peer behaves as if AUTN had been incorrect and fails the authentication (see Figure 3 of [RFC4187]). A peer not supporting EAP-AKA' will simply ignore this attribute. In all cases, the attribute is protected by the integrity mechanisms of EAP-AKA, so it cannot be removed by a man-in-the-middle attacker.

Note that we assume (Section 7) that EAP-AKA' is always stronger than EAP-AKA. As a result, this specification does not provide protection against bidding "down" attacks in the other direction, i.e., attackers forcing the endpoints to use EAP-AKA'.

4.1. Summary of Attributes for EAP-AKA

The appearance of the AT_BIDDING attribute in EAP-AKA exchanges is shown below, using the notation from Section 3.5:

Attribute	1	2	3	4	5	6	7	8	9	10	11	E
AT_BIDDING	0	0	1	0	0	0	0	0	0	0	0	N

Table 2: AT_BIDDING Attribute Appearance

5. Peer Identities

EAP-AKA' peer identities are as specified in [RFC4187], Section 4.1, with the addition of some requirements specified in this section.

EAP-AKA' includes optional identity privacy support that can be used to hide the cleartext permanent identity and thereby make the subscriber's EAP exchanges untraceable to eavesdroppers. EAP-AKA' can also use the privacy-friendly identifiers specified for 5G networks.

The permanent identity is usually based on the IMSI. Exposing the IMSI is undesirable because, as a permanent identity, it is easily trackable. In addition, since IMSIs may be used in other contexts as well, there would be additional opportunities for such tracking.

In EAP-AKA', identity privacy is based on temporary usernames or pseudonym usernames. These are similar to, but separate from, the Temporary Mobile Subscriber Identities (TMSI) that are used on cellular networks.

5.1. Username Types in EAP-AKA' Identities

Section 4.1.1.3 of [RFC4187] specifies that there are three types of usernames: permanent, pseudonym, and fast re-authentication usernames. This specification extends this definition as follows. There are four types of usernames:

(1) Regular usernames. These are external names given to EAP-AKA' peers. The regular usernames are further subdivided into to categories:

   (a) Permanent usernames, for instance, IMSI-based usernames.

   (b) Privacy-friendly temporary usernames, for instance, 5G GUTI (5G Globally Unique Temporary Identifier) or 5G privacy identifiers (see Section 5.3.2) such as SUCI (Subscription Concealed Identifier).

(2) EAP-AKA' pseudonym usernames. For example, 2s7ah6n9q@example.com might be a valid pseudonym identity. In this example, 2s7ah6n9q is the pseudonym username.

(3) EAP-AKA' fast re-authentication usernames. For example, 43953754@example.com might be a valid fast re-authentication identity and 43953754 the fast re-authentication username.

The permanent, privacy-friendly temporary, and pseudonym usernames are only used with full authentication, and fast re-authentication usernames only with fast re-authentication. Unlike permanent usernames and pseudonym usernames, privacy-friendly temporary usernames and fast re-authentication usernames are one-time identifiers, which are not reused across EAP exchanges.

5.2. Generating Pseudonyms and Fast Re-Authentication Identities

This section provides some additional guidance to implementations for producing secure pseudonyms and fast re-authentication identities. It does not impact backwards compatibility because each server consumes only the identities that it generates itself. However, adherence to the guidance will provide better security.

As specified by [RFC4187], Section 4.1.1.7, pseudonym usernames and fast re-authentication identities are generated by the EAP server in an implementation-dependent manner. RFC 4187 provides some general requirements on how these identities are transported, how they map to the NAI syntax, how they are distinguished from each other, and so on.

However, to enhance privacy, some additional requirements need to be applied.

The pseudonym usernames and fast re-authentication identities **MUST** be generated in a cryptographically secure way so that it is computationally infeasible for an attacker to differentiate two identities belonging to the same user from two identities belonging to different users. This can be achieved, for instance, by using random or pseudorandom identifiers such as random byte strings or ciphertexts. See also [RFC4086] for guidance on random number generation.

Note that the pseudonym and fast re-authentication usernames also **MUST NOT** include substrings that can be used to relate the username to a particular entity or a particular permanent identity. For instance, the usernames cannot include any subscriber-identifying part of an IMSI or other permanent identifier. Similarly, no part of the username can be formed by a fixed mapping that stays the same across multiple different pseudonyms or fast re-authentication identities for the same subscriber.

When the identifier used to identify a subscriber in an EAP-AKA' authentication exchange is a privacy-friendly identifier that is used only once, the EAP-AKA' peer **MUST NOT** use a pseudonym provided in that authentication exchange in subsequent exchanges more than once. To ensure that this does not happen, the EAP-AKA' server **MAY** decline to provide a pseudonym in such authentication exchanges. An important case where such privacy-friendly identifiers are used is in 5G networks (see Section 5.3).

5.3. Identifier Usage in 5G

In EAP-AKA', the peer identity may be communicated to the server in one of three ways:

- As a part of link-layer establishment procedures, externally to EAP.
- With the EAP-Response/Identity message in the beginning of the EAP exchange, but before the selection of EAP-AKA'.
- Transmitted from the peer to the server using EAP-AKA' messages instead of EAP-Response/Identity. In this case, the server includes an identity-requesting attribute (AT_ANY_ID_REQ, AT_FULLAUTH_ID_REQ, or AT_PERMANENT_ID_REQ) in the EAP-Request/AKA-Identity

message, and the peer includes the AT_IDENTITY attribute, which contains the peer's identity, in the EAP-Response/AKA-Identity message.

The identity carried above may be a permanent identity, privacy-friendly identity, pseudonym identity, or fast re-authentication identity as defined in Section 5.1.

5G supports the concept of privacy identifiers, and it is important for interoperability that the right type of identifier is used.

5G defines the SUbscription Permanent Identifier (SUPI) and SUbscription Concealed Identifier (SUCI) [TS-3GPP.23.501] [TS-3GPP.33.501] [TS-3GPP.23.003]. SUPI is globally unique and allocated to each subscriber. However, it is only used internally in the 5G network and is privacy sensitive. The SUCI is a privacy-preserving identifier containing the concealed SUPI, using public key cryptography to encrypt the SUPI.

Given the choice between these two types of identifiers, EAP-AKA' ensures interoperability as follows:

- Where identifiers are used within EAP-AKA' (such as key derivation) determine the exact values of the identity to be used, to avoid ambiguity (see Section 5.3.1).
- Where identifiers are carried within EAP-AKA' packets (such as in the AT_IDENTITY attribute) determine which identifiers should be filled in (see Section 5.3.2).

In 5G, the normal mode of operation is that identifiers are only transmitted outside EAP. However, in a system involving terminals from many generations and several connectivity options via 5G and other mechanisms, implementations and the EAP-AKA' specification need to prepare for many different situations, including sometimes having to communicate identities within EAP.

The following sections clarify which identifiers are used and how.

5.3.1. Key Derivation

In EAP-AKA', the peer identity is used in the key derivation formula found in Section 3.3.

The identity needs to be represented in exactly the correct format for the key derivation formula to produce correct results.

If the AT_KDF_INPUT parameter contains the prefix "5G:", the AT_KDF parameter has the value 1, and this authentication is not a fast re-authentication, then the peer identity used in the key derivation **MUST** be as specified in Annex F.3 of [TS-3GPP.33.501] and Clause 2.2 of [TS-3GPP.23.003]. This is in contrast to [RFC5448], which uses the identity as communicated in EAP and represented as a NAI. Also, in contrast to [RFC5448], in 5G EAP-AKA' does not use the "0" nor the "6" prefix in front of the identifier.

For an example of the format of the identity, see Clause 2.2 of [TS-3GPP.23.003].

In all other cases, the following applies:

The identity used in the key derivation formula **MUST** be exactly the one sent in the EAP-AKA' AT_IDENTITY attribute, if one was sent, regardless of the kind of identity that it may have been. If no AT_IDENTITY was sent, the identity **MUST** be exactly the one sent in the generic EAP Identity exchange, if one was made.

If no identity was communicated inside EAP, then the identity is the one communicated outside EAP in link-layer messaging.

In this case, the used identity **MUST** be the identity most recently communicated by the peer to the network, again regardless of what type of identity it may have been.

5.3.2. EAP Identity Response and EAP-AKA' AT_IDENTITY Attribute

The EAP authentication option is only available in 5G when the new 5G core network is also in use. However, in other networks, an EAP-AKA' peer may be connecting to other types of networks and existing equipment.

When the EAP server is in a 5G network, the 5G procedures for EAP-AKA' apply. [TS-3GPP.33.501] specifies when the EAP server is in a 5G network.

> Note: Currently, the following conditions are specified: when the EAP peer uses the 5G Non-Access Stratum (NAS) protocol [TS-3GPP.24.501] or when the EAP peer attaches to a network that advertises 5G connectivity without NAS [TS-3GPP.23.501]. Possible future conditions may also be specified by 3GPP.

When the 5G procedures for EAP-AKA' apply, EAP identity exchanges are generally not used as the identity is already made available on previous link-layer exchanges.

In this situation, the EAP Identity Response and EAP-AKA' AT_IDENTITY attribute are handled as specified in Annex F.2 of [TS-3GPP.33.501].

When used in EAP-AKA', the format of the SUCI **MUST** be as specified in [TS-3GPP.23.003], Section 28.7.3, with the semantics defined in [TS-3GPP.23.003], Section 2.2B. Also, in contrast to [RFC5448], in 5G EAP-AKA' does not use the "0" nor the "6" prefix in front of the identifier.

For an example of an IMSI in NAI format, see [TS-3GPP.23.003], Section 28.7.3.

Otherwise, the peer **SHOULD** employ an IMSI, SUPI, or NAI [RFC7542] as it is configured to use.

6. Exported Parameters

When not using fast re-authentication, the EAP-AKA' Session-Id is the concatenation of the EAP-AKA' Type value (0x32, one byte) with the contents of the RAND field from the AT_RAND attribute followed by the contents of the AUTN field in the AT_AUTN attribute:

```
Session-Id = 0x32 || RAND || AUTN
```

When using fast re-authentication, the EAP-AKA' Session-Id is the concatenation of the EAP-AKA' Type value (0x32) with the contents of the NONCE_S field from the AT_NONCE_S attribute followed by the contents of the MAC field from the AT_MAC attribute from the EAP-Request/AKA-Reauthentication:

```
Session-Id = 0x32 || NONCE_S || MAC
```

The Peer-Id is the contents of the Identity field from the AT_IDENTITY attribute, using only the Actual Identity Length bytes from the beginning. Note that the contents are used as they are transmitted, regardless of whether the transmitted identity was a permanent, pseudonym, or fast EAP re-authentication identity. If no AT_IDENTITY attribute was exchanged, the exported Peer-Id is the identity provided from the EAP Identity Response packet. If no EAP Identity Response was provided either, the exported Peer-Id is the null string (zero length).

The Server-Id is the null string (zero length).

7. Security Considerations

A summary of the security properties of EAP-AKA' follows. These properties are very similar to those in EAP-AKA. We assume that HMAC SHA-256 is at least as secure as HMAC SHA-1 (see also [RFC6194]). This is called the SHA-256 assumption in the remainder of this section. Under this assumption, EAP-AKA' is at least as secure as EAP-AKA.

If the AT_KDF attribute has value 1, then the security properties of EAP-AKA' are as follows:

Protected ciphersuite negotiation
: EAP-AKA' has no ciphersuite negotiation mechanisms. It does have a negotiation mechanism for selecting the key derivation functions. This mechanism is secure against bidding down attacks from EAP-AKA' to EAP-AKA. The negotiation mechanism allows changing the offered key derivation function, but the change is visible in the final EAP-Request/AKA'-Challenge message that the server sends to the peer. This message is authenticated via the AT_MAC attribute, and carries both the chosen alternative and the initially offered list. The peer refuses to accept a change it did not initiate. As a result, both parties are aware that a change is being made and what the original offer was.

  Per assumptions in Section 4, there is no protection against bidding down attacks from EAP-AKA to EAP-AKA' should EAP-AKA' somehow be considered less secure some day than EAP-AKA. Such protection was not provided in RFC 5448 implementations and consequently neither does this specification provide it. If such support is needed, it would have to be added as a separate new feature.

In general, it is expected that the current negotiation capabilities in EAP-AKA' are sufficient for some types of extensions, including adding Perfect Forward Secrecy [EMU-AKA-PFS] and perhaps others. However, some larger changes may require a new EAP method type, which is how EAP-AKA' itself happened. One example of such change would be the introduction of new algorithms.

Mutual authentication
: Under the SHA-256 assumption, the properties of EAP-AKA' are at least as good as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12, for further details.

Integrity protection
: Under the SHA-256 assumption, the properties of EAP-AKA' are at least as good (most likely better) as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12, for further details. The only difference is that a stronger hash algorithm and keyed MAC, SHA-256 / HMAC-SHA-256, is used instead of SHA-1 / HMAC-SHA-1.

Replay protection
: Under the SHA-256 assumption, the properties of EAP-AKA' are at least as good as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12, for further details.

Confidentiality
: The properties of EAP-AKA' are exactly the same as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12, for further details.

Key derivation
: EAP-AKA' supports key derivation with an effective key strength against brute-force attacks equal to the minimum of the length of the derived keys and the length of the AKA base key, i.e., 128 bits or more. The key hierarchy is specified in Section 3.3.

  The Transient EAP Keys used to protect EAP-AKA packets (K_encr, K_aut, K_re), the MSK, and the EMSK are cryptographically separate. If we make the assumption that SHA-256 behaves as a pseudorandom function, an attacker is incapable of deriving any non-trivial information about any of these keys based on the other keys. An attacker also cannot calculate the pre-shared secret from IK, CK, IK', CK', K_encr, K_aut, K_re, MSK, or EMSK by any practically feasible means.

  EAP-AKA' adds an additional layer of key derivation functions within itself to protect against the use of compromised keys. This is discussed further in Section 7.4.

  EAP-AKA' uses a pseudorandom function modeled after the one used in IKEv2 [RFC7296] together with SHA-256.

Key strength
: See above.

Dictionary attack resistance
: Under the SHA-256 assumption, the properties of EAP-AKA' are at least as good as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12, for further details.

Fast reconnect
: Under the SHA-256 assumption, the properties of EAP-AKA' are at least as good as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12, for further details. Note that implementations **MUST** prevent performing a fast reconnect across method types.

Cryptographic binding
: Note that this term refers to a very specific form of binding, something that is performed between two layers of authentication. It is not the same as the binding to a particular network name. The properties of EAP-AKA' are exactly the same as those of EAP-AKA in this respect, i.e., as it is not a tunnel method, this property is not applicable to it. Refer to [RFC4187], Section 12, for further details.

Session independence
: The properties of EAP-AKA' are exactly the same as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12, for further details.

Fragmentation
: The properties of EAP-AKA' are exactly the same as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12, for further details.

Channel binding
: EAP-AKA', like EAP-AKA, does not provide channel bindings as they're defined in [RFC3748] and [RFC5247]. New skippable attributes can be used to add channel binding support in the future, if required.

  However, including the Network Name field in the AKA' algorithms (which are also used for other purposes than EAP-AKA') provides a form of cryptographic separation between different network names, which resembles channel bindings. However, the network name does not typically identify the EAP (pass-through) authenticator. See Section 7.4 for more discussion.

7.1. Privacy

[RFC6973] suggests that the privacy considerations of IETF protocols be documented.

The confidentiality properties of EAP-AKA' itself have been discussed above under "Confidentiality" (Section 7).

EAP-AKA' uses several different types of identifiers to identify the authenticating peer. It is strongly **RECOMMENDED** to use the privacy-friendly temporary or hidden identifiers, i.e., the 5G GUTI or SUCI, pseudonym usernames, and fast re-authentication usernames. The use of permanent identifiers such as the IMSI or SUPI may lead to an ability to track the peer and/or user associated with the peer. The use of permanent identifiers such as the IMSI or SUPI is strongly **NOT RECOMMENDED**.

As discussed in Section 5.3, when authenticating to a 5G network, only the SUCI identifier is normally used. The use of EAP-AKA' pseudonyms in this situation is at best limited because the SUCI already provides a stronger mechanism. In fact, reusing the same pseudonym multiple times will result in a tracking opportunity for observers that see the pseudonym pass by. To avoid this, the peer and server need to follow the guidelines given in Section 5.2.

When authenticating to a 5G network, per Section 5.3.1, both the EAP-AKA' peer and server need to employ the permanent identifier SUPI as an input to key derivation. However, this use of the SUPI is only internal. As such, the SUPI need not be communicated in EAP messages. Therefore, SUPI **MUST NOT** be communicated in EAP-AKA' when authenticating to a 5G network.

While the use of SUCI in 5G networks generally provides identity privacy, this is not true if the null-scheme encryption is used to construct the SUCI (see [TS-3GPP.33.501], Annex C). The use of this scheme makes the use of SUCI equivalent to the use of SUPI or IMSI. The use of the null scheme is **NOT RECOMMENDED** where identity privacy is important.

The use of fast re-authentication identities when authenticating to a 5G network does not have the same problems as the use of pseudonyms, as long as the 5G authentication server generates the fast re-authentication identifiers in a proper manner specified in Section 5.2.

Outside 5G, the peer can freely choose between the use of permanent, pseudonym, or fast re-authentication identifiers:

- A peer that has not yet performed any EAP-AKA' exchanges does not typically have a pseudonym available. If the peer does not have a pseudonym available, then the privacy mechanism cannot be used, and the permanent identity will have to be sent in the clear.

  The terminal **SHOULD** store the pseudonym in nonvolatile memory so that it can be maintained across reboots. An active attacker that impersonates the network may use the AT_PERMANENT_ID_REQ attribute ([RFC4187], Section 4.1.2) to learn the subscriber's IMSI. However, as discussed in [RFC4187], Section 4.1.2, the terminal can refuse to send the cleartext permanent identity if it believes that the network should be able to recognize the pseudonym.
- When pseudonyms and fast re-authentication identities are used, the peer relies on the properly created identifiers by the server.

  It is essential that an attacker cannot link a privacy-friendly identifier to the user in any way or determine that two identifiers belong to the same user as outlined in Section 5.2. The pseudonym usernames and fast re-authentication identities **MUST NOT** be used for other purposes (e.g., in other protocols).

If the peer and server cannot guarantee that SUCI can be used or that pseudonyms will be available, generated properly, and maintained reliably, and identity privacy is required, then additional protection from an external security mechanism such as tunneled EAP methods like Tunneled Transport Layer Security (TTLS) [RFC5281] or Tunnel Extensible Authentication Protocol (TEAP) [RFC7170] may be used. The benefits and the security considerations of using an external security mechanism with EAP-AKA are beyond the scope of this document.

Finally, as with other EAP methods, even when privacy-friendly identifiers or EAP tunneling is used, typically the domain part of an identifier (e.g., the home operator) is visible to external parties.

7.2. Discovered Vulnerabilities

There have been no published attacks that violate the primary secrecy or authentication properties defined for Authentication and Key Agreement (AKA) under the originally assumed trust model. The same is true of EAP-AKA'.

However, there have been attacks when a different trust model is in use, with characteristics not originally provided by the design, or when participants in the protocol leak information to outsiders on purpose, and there have been some privacy-related attacks.

For instance, the original AKA protocol does not prevent an insider supplying keys to a third party, e.g., as described by Mjølsnes and Tsay in [MT2012] where a serving network lets an authentication run succeed, but then it misuses the session keys to send traffic on the authenticated user's behalf. This particular attack is not different from any on-path entity (such as a router) pretending to send traffic, but the general issue of insider attacks can be a problem, particularly in a large group of collaborating operators.

Another class of attacks is the use of tunneling of traffic from one place to another, e.g., as done by Zhang and Fang in [ZF2005] to leverage security policy differences between different operator networks, for instance. To gain something in such an attack, the attacker needs to trick the user into believing it is in another location. If policies between locations differ, for instance, if payload traffic is not required to be encrypted in some location, the attacker may trick the user into opening a vulnerability. As an authentication mechanism, EAP-AKA' is not directly affected by most of these attacks. EAP-AKA' network name binding can also help alleviate some of the attacks. In any case, it is recommended that EAP-AKA' configuration not be dependent on the location of request origin, unless the location information can be cryptographically confirmed, e.g., with the network name binding.

Zhang and Fang also looked at denial-of-service attacks [ZF2005]. A serving network may request large numbers of authentication runs for a particular subscriber from a home network. While the resynchronization process can help recover from this, eventually it is possible to exhaust the sequence number space and render the subscriber's card unusable. This attack is possible for both original AKA and EAP-AKA'. However, it requires the collaboration of a serving network in an attack. It is recommended that EAP-AKA' implementations provide the means to track, detect, and limit excessive authentication attempts to combat this problem.

There have also been attacks related to the use of AKA without the generated session keys (e.g., [BT2013]). Some of those attacks relate to the use of HTTP Digest AKAv1 [RFC3310], which was originally vulnerable to man-in-the-middle attacks. This has since been corrected in [RFC4169]. The EAP-AKA' protocol uses session keys and provides channel binding, and as such, it is resistant to the above attacks except where the protocol participants leak information to outsiders.

Basin, et al. [Basin2018] have performed formal analysis and concluded that the AKA protocol would have benefited from additional security requirements such as key confirmation.

In the context of pervasive monitoring revelations, there were also reports of compromised long-term pre-shared keys used in SIM and AKA [Heist2015]. While no protocol can survive the theft of key material associated with its credentials, there are some things that alleviate the impacts in such situations. These are discussed further in Section 7.3.

Arapinis, et al. [Arapinis2012] describe an attack that uses the AKA resynchronization protocol to attempt to detect whether a particular subscriber is in a given area. This attack depends on the attacker setting up a false base station in the given area and on the subscriber performing at least one authentication between the time the attack is set up and run.

Borgaonkar, et al. discovered that the AKA resynchronization protocol may also be used to predict the authentication frequency of a subscriber if a non-time-based sequence number (SQN) generation scheme is used [Borgaonkar2018]. The attacker can force the reuse of the keystream that is used to protect the SQN in the AKA resynchronization protocol. The attacker then guesses the authentication frequency based on the lowest bits of two XORed SQNs. The researchers' concern was that the authentication frequency would reveal some information about the phone usage behavior, e.g., number of phone calls made or number of SMS messages sent. There are a number of possible triggers for authentication, so such an information leak is not direct, but it can be a concern. The impact of the attack differs depending on whether the SQN generation scheme that is used is time-based or not.

Similar attacks are possible outside AKA in the cellular paging protocols where the attacker can simply send application-layer data, send short messages, or make phone calls to the intended victim and observe the air interface (e.g., [Kune2012] and [Shaik2016]). Hussain, et al. demonstrated a slightly more sophisticated version of the attack that exploits the fact that the 4G paging protocol uses the IMSI to calculate the paging timeslot [Hussain2019]. As this attack is outside AKA, it does not impact EAP-AKA'.

Finally, bad implementations of EAP-AKA' may not produce pseudonym usernames or fast re-authentication identities in a manner that is sufficiently secure. While it is not a problem with the protocol itself, following the recommendations in Section 5.2 can mitigate this concern.

7.3. Pervasive Monitoring

As required by [RFC7258], work on IETF protocols needs to consider the effects of pervasive monitoring and mitigate them when possible.

As described in Section 7.2, after the publication of RFC 5448, new information has come to light regarding the use of pervasive monitoring techniques against many security technologies, including AKA-based authentication.

For AKA, these attacks relate to theft of the long-term, shared-secret key material stored on the cards. Such attacks are conceivable, for instance, during the manufacturing process of cards, through coercion of the card manufacturers, or during the transfer of cards and associated information to an operator. Since the publication of reports about such attacks, manufacturing and provisioning processes have gained much scrutiny and have improved.

In particular, it is crucial that manufacturers limit access to the secret information and the cards only to necessary systems and personnel. It is also crucial that secure mechanisms be used to store and communicate the secrets between the manufacturer and the operator that adopts those cards for their customers.

Beyond these operational considerations, there are also technical means to improve resistance to these attacks. One approach is to provide Perfect Forward Secrecy (PFS). This would prevent any passive attacks merely based on the long-term secrets and observation of traffic. Such a mechanism can be defined as a backwards-compatible extension of EAP-AKA' and is pursued separately from this specification [EMU-AKA-PFS]. Alternatively, EAP-AKA' authentication can be run inside a PFS-capable, tunneled authentication method. In any case, the use of some PFS-capable mechanism is recommended.

7.4. Security Properties of Binding Network Names

The ability of EAP-AKA' to bind the network name into the used keys provides some additional protection against key leakage to inappropriate parties. The keys used in the protocol are specific to a particular network name. If key leakage occurs due to an accident, access node compromise, or another attack, the leaked keys are only useful when providing access with that name. For instance, a malicious access point cannot claim to be network Y if it has stolen keys from network X. Obviously, if an access point is compromised, the malicious node can still represent the compromised node. As a result, neither EAP-AKA' nor any other extension can prevent such attacks; however, the binding to a particular name limits the attacker's choices, allows better tracking of attacks, makes it possible to identify compromised networks, and applies good cryptographic hygiene.

The server receives the EAP transaction from a given access network, and verifies that the claim from the access network corresponds to the name that this access network should be using. It becomes impossible for an access network to claim over AAA that it is another access network. In addition, if the peer checks that the information it has received locally over the network-access link-layer matches with the information the server has given it via EAP-AKA', it becomes impossible for the access network to tell one story to the AAA network and another one to the peer. These checks prevent some "lying NAS" (Network Access Server) attacks. For instance, a roaming partner, R, might claim that it is the home network H in an effort to lure peers to connect to itself. Such an attack would be beneficial for the roaming partner if it can attract more users, and damaging for the users if their access costs in R are higher than those in other alternative networks, such as H.

Any attacker who gets hold of the keys CK and IK, produced by the AKA algorithm, can compute the keys CK' and IK' and, hence, the Master Key (MK) according to the rules in Section 3.3. The attacker could then act as a lying NAS. In 3GPP systems in general, the keys CK and IK have been

distributed to, for instance, nodes in a visited access network where they may be vulnerable. In order to reduce this risk, the AKA algorithm **MUST** be computed with the AMF separation bit set to 1, and the peer **MUST** check that this is indeed the case whenever it runs EAP-AKA'. Furthermore, [TS-3GPP.33.402] requires that no CK or IK keys computed in this way ever leave the home subscriber system.

The additional security benefits obtained from the binding depend obviously on the way names are assigned to different access networks. This is specified in [TS-3GPP.24.302]. See also [TS-3GPP.23.003]. Ideally, the names allow separating each different access technology, each different access network, and each different NAS within a domain. If this is not possible, the full benefits may not be achieved. For instance, if the names identify just an access technology, use of compromised keys in a different technology can be prevented, but it is not possible to prevent their use by other domains or devices using the same technology.

8. IANA Considerations

IANA has updated the "Extensible Authentication Protocol (EAP) Registry" and the "EAP-AKA and EAP-SIM Parameters" registry so that entries that pointed to RFC 5448 now point to this RFC instead.

8.1. Type Value

IANA has updated the reference for EAP-AKA' (0x32) in the "Method Types" subregistry under the "Extensible Authentication Protocol (EAP) Registry" to point to this document. Per Section 6.2 of [RFC3748], this allocation can be made with Specification Required [RFC8126].

8.2. Attribute Type Values

EAP-AKA' shares its attribute space and subtypes with EAP-SIM [RFC4186] and EAP-AKA [RFC4187]. No new registries are needed.

IANA has updated the reference for AT_KDF_INPUT (23) and AT_KDF (24) in the "Attribute Types (Non-Skippable Attributes 0-127)" subregistry under the "EAP-AKA and EAP-SIM Parameters" registry to point to this document. AT_KDF_INPUT and AT_KDF are defined in Sections 3.1 and 3.2, respectively, of this document.

IANA has also updated the reference for AT_BIDDING (136) in the "Attribute Types (Skippable Attributes 128-255)" subregistry of the "EAP-AKA and EAP-SIM Parameters" registry to point to this document. AT_BIDDING is defined in Section 4.

8.3. Key Derivation Function Namespace

IANA has updated the reference for the "EAP-AKA' AT_KDF Key Derivation Function Values" subregistry to point to this document. This subregistry appears under the "EAP-AKA and EAP-SIM Parameters" registry. The references for following entries have also been updated to point to this document. New values can be created through the Specification Required policy [RFC8126].

Value	Description	Reference
0	Reserved	RFC 9048
1	EAP-AKA' with CK'/IK'	RFC 9048

Table 3: EAP-AKA' AT_KDF Key Derivation Function Values

9. References

9.1. Normative References

[FIPS.180-4] National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-4, DOI 10.6028/NIST.FIPS.180-4, August 2015, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <https://www.rfc-editor.org/info/rfc3748>.

[RFC4187] Arkko, J. and H. Haverinen, "Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)", RFC 4187, DOI 10.17487/RFC4187, January 2006, <https://www.rfc-editor.org/info/rfc4187>.

[RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, DOI 10.17487/RFC7542, May 2015, <https://www.rfc-editor.org/info/rfc7542>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[TS-3GPP.23.003] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Numbering, addressing and identification (Release 16)", Version 16.7.0, 3GPP Technical Specification 23.003, June 2021.

[TS-3GPP.23.501] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; System architecture for the 5G System (5GS); (Release 16)", Version 16.9.0, 3GPP Technical Specification 23.501, June 2021.

[TS-3GPP.24.302] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Access to the 3GPP Evolved Packet Core (EPC) via non-3GPP access networks; Stage 3; (Release 16)", Version 16.4.0, 3GPP Technical Specification 24.302, July 2020.

[TS-3GPP.24.501] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Non-Access-Stratum (NAS) protocol for 5G System (5GS); Stage 3; (Release 16)", Version 16.9.0, 3GPP Draft Technical Specification 24.501, June 2021.

[TS-3GPP.33.102] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security architecture (Release 16)", Version 16.0.0, 3GPP Technical Specification 33.102, July 2020.

[TS-3GPP.33.402] 3GPP, "3GPP System Architecture Evolution (SAE); Security aspects of non-3GPP accesses (Release 16)", Version 16.0.0, 3GPP Technical Specification 33.402, July 2020.

[TS-3GPP.33.501] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security architecture and procedures for 5G System (Release 16)", Version 16.7.1, 3GPP Technical Specification 33.501, July 2021.

9.2. Informative References

[Arapinis2012] Arapinis, M., Mancini, L., Ritter, E., Ryan, M., Golde, N., Redon, R., and R. Borgaonkar, "New Privacy Issues in Mobile Telephony: Fix and Verification", in CCS '12: Proceedings of the 2012 ACM Conference on Computer and Communications Security, Raleigh, North Carolina, USA, DOI 10.1145/2382196.2382221, October 2012, <https://doi.org/10.1145/2382196.2382221>.

[Basin2018] Basin, D., Dreier, J., Hirschi, L., Radomirović, S., Sasse, R., and V. Stettler, "A Formal Analysis of 5G Authentication", arXiv:1806.10360, DOI 10.1145/3243734.3243846, August 2018, <https://doi.org/10.1145/3243734.3243846>.

[Borgaonkar2018] Borgaonkar, R., Hirschi, L., Park, S., and A. Shaik, "New Privacy Threat on 3G, 4G, and Upcoming 5G AKA Protocols", in IACR Cryptology ePrint Archive, 2018.

[BT2013] Beekman, J. G. and C. Thompson, "Breaking Cell Phone Authentication: Vulnerabilities in AKA, IMS and Android", in 7th USENIX Workshop on Offensive Technologies, WOOT '13, August 2013.

[EMU-AKA-PFS] Arkko, J., Norrman, K., and V. Torvinen, "Perfect-Forward Secrecy for the Extensible Authentication Protocol Method for Authentication and Key Agreement (EAP-AKA' PFS)", Work in Progress, Internet-Draft, draft-ietf-emu-aka-pfs-05, 30 October 2020, <https://datatracker.ietf.org/doc/html/draft-ietf-emu-aka-pfs-05>.

[FIPS.180-1] National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-1, DOI 10.6028/NIST.FIPS.180-1, April 1995, <https://csrc.nist.gov/publications/detail/fips/180/1/archive/1995-04-17>.

[FIPS.180-2] National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-2, August 2002, <https://csrc.nist.gov/publications/detail/fips/180/2/archive/2002-08-01>.

[Heist2015] Scahill, J. and J. Begley, "How Spies Stole the Keys to the Encryption Castle", February 2015, <https://firstlook.org/theintercept/2015/02/19/great-sim-heist/>.

[Hussain2019] Hussain, S., Echeverria, M., Chowdhury, O., Li, N., and E. Bertino, "Privacy Attacks to the 4G and 5G Cellular Paging Protocols Using Side Channel Information", in the proceedings of NDSS '19, held 24-27 February, 2019, San Diego, California, 2019.

[Kune2012] Kune, D., Koelndorfer, J., Hopper, N., and Y. Kim, "Location Leaks on the GSM Air Interface", in the proceedings of NDSS '12, held 5-8 February, 2012, San Diego, California, 2012.

[MT2012] Mjølsnes, S. F. and J-K. Tsay, "A Vulnerability in the UMTS and LTE Authentication and Key Agreement Protocols", in Computer Network Security, Proceedings of the 6th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security, Lecture Notes in Computer Science, Vol. 7531, pp. 65-76, DOI 10.1007/978-3-642-33704-8_6, October 2012, <https://doi.org/10.1007/978-3-642-33704-8_6>.

[RFC3310] Niemi, A., Arkko, J., and V. Torvinen, "Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA)", RFC 3310, DOI 10.17487/RFC3310, September 2002, <https://www.rfc-editor.org/info/rfc3310>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC4169] Torvinen, V., Arkko, J., and M. Naslund, "Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA) Version-2", RFC 4169, DOI 10.17487/RFC4169, November 2005, <https://www.rfc-editor.org/info/rfc4169>.

[RFC4186] Haverinen, H., Ed. and J. Salowey, Ed., "Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)", RFC 4186, DOI 10.17487/RFC4186, January 2006, <https://www.rfc-editor.org/info/rfc4186>.

[RFC4284] Adrangi, F., Lortz, V., Bari, F., and P. Eronen, "Identity Selection Hints for the Extensible Authentication Protocol (EAP)", RFC 4284, DOI 10.17487/RFC4284, January 2006, <https://www.rfc-editor.org/info/rfc4284>.

[RFC4306] Kaufman, C., Ed., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, DOI 10.17487/RFC4306, December 2005, <https://www.rfc-editor.org/info/rfc4306>.

[RFC5113] Arkko, J., Aboba, B., Korhonen, J., Ed., and F. Bari, "Network Discovery and Selection Problem", RFC 5113, DOI 10.17487/RFC5113, January 2008, <https://www.rfc-editor.org/info/rfc5113>.

[RFC5247] Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, DOI 10.17487/RFC5247, August 2008, <https://www.rfc-editor.org/info/rfc5247>.

[RFC5281] Funk, P. and S. Blake-Wilson, "Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)", RFC 5281, DOI 10.17487/RFC5281, August 2008, <https://www.rfc-editor.org/info/rfc5281>.

[RFC5448] Arkko, J., Lehtovirta, V., and P. Eronen, "Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA')", RFC 5448, DOI 10.17487/RFC5448, May 2009, <https://www.rfc-editor.org/info/rfc5448>.

[RFC6194] Polk, T., Chen, L., Turner, S., and P. Hoffman, "Security Considerations for the SHA-0 and SHA-1 Message-Digest Algorithms", RFC 6194, DOI 10.17487/RFC6194, March 2011, <https://www.rfc-editor.org/info/rfc6194>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <https://www.rfc-editor.org/info/rfc6973>.

[RFC7170] Zhou, H., Cam-Winget, N., Salowey, J., and S. Hanna, "Tunnel Extensible Authentication Protocol (TEAP) Version 1", RFC 7170, DOI 10.17487/RFC7170, May 2014, <https://www.rfc-editor.org/info/rfc7170>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <https://www.rfc-editor.org/info/rfc7258>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <https://www.rfc-editor.org/info/rfc7296>.

[Shaik2016] Shaik, A., Seifert, J., Borgaonkar, R., Asokan, N., and V. Niemi, "Practical attacks against Privacy and Availability in 4G/LTE Mobile Communication Systems", in the proceedings of NDSS '16 held 21-24 February, 2016, San Diego, California, 2012.

[TS-3GPP.35.208] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 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 4: Design Conformance Test Data (Release 14)", Version 16.0.0, 3GPP Technical Specification 35.208, July 2020.

[ZF2005] Zhang, M. and Y. Fang, "Security analysis and enhancements of 3GPP authentication and key agreement protocol", IEEE Transactions on Wireless Communications, Vol. 4, No. 2, DOI 10.1109/TWC.2004.842941, March 2005, <https://doi.org/10.1109/TWC.2004.842941>.

Appendix A. Changes from RFC 5448

The change from RFC 5448 was to refer to a newer version of [TS-3GPP.24.302]. This RFC includes an updated definition of the Network Name field to include 5G.

Identifier usage for 5G has been specified in Section 5.3. Also, the requirements for generating pseudonym usernames and fast re-authentication identities have been updated from the original definition in RFC 5448, which referenced RFC 4187. See Section 5.

Exported parameters for EAP-AKA' have been defined in Section 6, as required by [RFC5247], including the definition of those parameters for both full authentication and fast re-authentication.

The security, privacy, and pervasive monitoring considerations have been updated or added. See Section 7.

The references to [RFC2119], [RFC4306], [RFC7296], [FIPS.180-1] and [FIPS.180-2] have been updated to their most recent versions, and language in this document has been changed accordingly. However, these are merely reference updates to newer specifications; the actual protocol functions are the same as defined in the earlier RFCs.

Similarly, references to all 3GPP technical specifications have been updated to their 5G versions (Release 16) or otherwise most recent version when there has not been a 5G-related update.

Finally, a number of clarifications have been made, including a summary of where attributes may appear.

Appendix B. Changes to RFC 4187

In addition to specifying EAP-AKA', this document also mandates a change to another EAP method -- EAP-AKA that was defined in RFC 4187. This change was already mandated in RFC 5448 but repeated here to ensure that the latest EAP-AKA' specification contains the instructions about the necessary bidding down prevention feature in EAP-AKA as well.

The changes to RFC 4187 relate only to the bidding down prevention support defined in Section 4. In particular, this document does not change how the Master Key (MK) is calculated or any other aspect of EAP-AKA. The provisions in this specification for EAP-AKA' do not apply to EAP-AKA, outside of Section 4.

Appendix C. Importance of Explicit Negotiation

Choosing between the traditional and revised AKA key derivation functions is easy when their use is unambiguously tied to a particular radio access network, e.g., Long Term Evolution (LTE) as defined by 3GPP or evolved High Rate Packet Data (eHRPD) as defined by 3GPP2. There is no possibility for interoperability problems if this radio access network is always used in conjunction with new protocols that cannot be mixed with the old ones; clients will always know whether they are connecting to the old or new system.

However, using the new key derivation functions over EAP introduces several degrees of separation, making the choice of the correct key derivation functions much harder. Many different types of networks employ EAP. Most of these networks have no means to carry any information about what is expected from the authentication process. EAP itself is severely limited in carrying any additional information, as noted in [RFC4284] and [RFC5113]. Even if these networks or EAP were extended to carry additional information, it would not affect millions of deployed access networks and clients attaching to them.

Simply changing the key derivation functions that EAP-AKA [RFC4187] uses would cause interoperability problems with all of the existing implementations. Perhaps it would be possible to employ strict separation into domain names that should be used by the new clients and networks. Only these new devices would then employ the new key derivation function. While this can be made to work for specific cases, it would be an extremely brittle mechanism, ripe to result in problems whenever client configuration, routing of authentication requests, or server configuration does not match expectations. It also does not help to assume that the EAP client and server are running a particular release of 3GPP network specifications. Network vendors often provide features from future releases early or do not provide all features of the current release. And obviously, there are many EAP and even some EAP-AKA implementations that are not bundled with the 3GPP network offerings. In general, these approaches are expected to lead to hard-to-diagnose problems and increased support calls.

Appendix D. Test Vectors

Test vectors are provided below for four different cases. The test vectors may be useful for testing implementations. In the first two cases, we employ the MILENAGE algorithm and the algorithm configuration parameters (the subscriber key K and operator algorithm variant configuration value OP) from test set 19 in [TS-3GPP.35.208].

The last two cases use artificial values as the output of AKA, which are useful only for testing the computation of values within EAP-AKA', not AKA itself.

Case 1

```
   The parameters for the AKA run are as follows:

      Identity:     "0555444333222111"

      Network name: "WLAN"

      RAND:         81e9 2b6c 0ee0 e12e bceb a8d9 2a99 dfa5

      AUTN:         bb52 e91c 747a c3ab 2a5c 23d1 5ee3 51d5

      IK:           9744 871a d32b f9bb d1dd 5ce5 4e3e 2e5a

      CK:           5349 fbe0 9864 9f94 8f5d 2e97 3a81 c00f

      RES:          28d7 b0f2 a2ec 3de5

   Then the derived keys are generated as follows:

      CK':          0093 962d 0dd8 4aa5 684b 045c 9edf fa04

      IK':          ccfc 230c a74f cc96 c0a5 d611 64f5 a76c

      K_encr:       766f a0a6 c317 174b 812d 52fb cd11 a179

      K_aut:        0842 ea72 2ff6 835b fa20 3249 9fc3 ec23
                    c2f0 e388 b4f0 7543 ffc6 77f1 696d 71ea

      K_re:         cf83 aa8b c7e0 aced 892a cc98 e76a 9b20
                    95b5 58c7 795c 7094 715c b339 3aa7 d17a

      MSK:          67c4 2d9a a56c 1b79 e295 e345 9fc3 d187
                    d42b e0bf 818d 3070 e362 c5e9 67a4 d544
                    e8ec fe19 358a b303 9aff 03b7 c930 588c
                    055b abee 58a0 2650 b067 ec4e 9347 c75a

      EMSK:         f861 703c d775 590e 16c7 679e a387 4ada
                    8663 11de 2907 64d7 60cf 76df 647e a01c
                    313f 6992 4bdd 7650 ca9b ac14 1ea0 75c4
                    ef9e 8029 c0e2 90cd bad5 638b 63bc 23fb
```

Case 2

```
   The parameters for the AKA run are as follows:

      Identity:     "0555444333222111"

      Network name: "HRPD"

      RAND:         81e9 2b6c 0ee0 e12e bceb a8d9 2a99 dfa5

      AUTN:         bb52 e91c 747a c3ab 2a5c 23d1 5ee3 51d5

      IK:           9744 871a d32b f9bb d1dd 5ce5 4e3e 2e5a

      CK:           5349 fbe0 9864 9f94 8f5d 2e97 3a81 c00f

      RES:          28d7 b0f2 a2ec 3de5

   Then the derived keys are generated as follows:

      CK':          3820 f027 7fa5 f777 32b1 fb1d 90c1 a0da

      IK':          db94 a0ab 557e f6c9 ab48 619c a05b 9a9f

      K_encr:       05ad 73ac 915f ce89 ac77 e152 0d82 187b

      K_aut:        5b4a caef 62c6 ebb8 882b 2f3d 534c 4b35
                    2773 37a0 0184 f20f f25d 224c 04be 2afd

      K_re:         3f90 bf5c 6e5e f325 ff04 eb5e f653 9fa8
                    cca8 3981 94fb d00b e425 b3f4 0dba 10ac

      MSK:          87b3 2157 0117 cd6c 95ab 6c43 6fb5 073f
                    f15c f855 05d2 bc5b b735 5fc2 1ea8 a757
                    57e8 f86a 2b13 8002 e057 5291 3bb4 3b82
                    f868 a961 17e9 1a2d 95f5 2667 7d57 2900

      EMSK:         c891 d5f2 0f14 8a10 0755 3e2d ea55 5c9c
                    b672 e967 5f4a 66b4 bafa 0273 79f9 3aee
                    539a 5979 d0a0 042b 9d2a e28b ed3b 17a3
                    1dc8 ab75 072b 80bd 0c1d a612 466e 402c
```

Case 3

```
   The parameters for the AKA run are as follows:

      Identity:     "0555444333222111"

      Network name: "WLAN"

      RAND:         e0e0 e0e0 e0e0 e0e0 e0e0 e0e0 e0e0 e0e0

      AUTN:         a0a0 a0a0 a0a0 a0a0 a0a0 a0a0 a0a0 a0a0

      IK:           b0b0 b0b0 b0b0 b0b0 b0b0 b0b0 b0b0 b0b0

      CK:           c0c0 c0c0 c0c0 c0c0 c0c0 c0c0 c0c0 c0c0

      RES:          d0d0 d0d0 d0d0 d0d0 d0d0 d0d0 d0d0 d0d0

   Then the derived keys are generated as follows:

      CK':          cd4c 8e5c 68f5 7dd1 d7d7 dfd0 c538 e577

      IK':          3ece 6b70 5dbb f7df c459 a112 80c6 5524

      K_encr:       897d 302f a284 7416 488c 28e2 0dcb 7be4

      K_aut:        c407 00e7 7224 83ae 3dc7 139e b0b8 8bb5
                    58cb 3081 eccd 057f 9207 d128 6ee7 dd53

      K_re:         0a59 1a22 dd8b 5b1c f29e 3d50 8c91 dbbd
                    b4ae e230 5189 2c42 b6a2 de66 ea50 4473

      MSK:          9f7d ca9e 37bb 2202 9ed9 86e7 cd09 d4a7
                    0d1a c76d 9553 5c5c ac40 a750 4699 bb89
                    61a2 9ef6 f3e9 0f18 3de5 861a d1be dc81
                    ce99 1639 1b40 1aa0 06c9 8785 a575 6df7

      EMSK:         724d e00b db9e 5681 87be 3fe7 4611 4557
                    d501 8779 537e e37f 4d3c 6c73 8cb9 7b9d
                    c651 bc19 bfad c344 ffe2 b52c a78b d831
                    6b51 dacc 5f2b 1440 cb95 1552 1cc7 ba23
```

Case 4

```
   The parameters for the AKA run are as follows:

      Identity:     "0555444333222111"

      Network name: "HRPD"

      RAND:         e0e0 e0e0 e0e0 e0e0 e0e0 e0e0 e0e0 e0e0

      AUTN:         a0a0 a0a0 a0a0 a0a0 a0a0 a0a0 a0a0 a0a0

      IK:           b0b0 b0b0 b0b0 b0b0 b0b0 b0b0 b0b0 b0b0

      CK:           c0c0 c0c0 c0c0 c0c0 c0c0 c0c0 c0c0 c0c0

      RES:          d0d0 d0d0 d0d0 d0d0 d0d0 d0d0 d0d0 d0d0

   Then the derived keys are generated as follows:

      CK':          8310 a71c e6f7 5488 9613 da8f 64d5 fb46

      IK':          5adf 1436 0ae8 3819 2db2 3f6f cb7f 8c76

      K_encr:       745e 7439 ba23 8f50 fcac 4d15 d47c d1d9

      K_aut:        3e1d 2aa4 e677 025c fd86 2a4b e183 61a1
                    3a64 5765 5714 63df 833a 9759 e809 9879

      K_re:         99da 835e 2ae8 2462 576f e651 6fad 1f80
                    2f0f a119 1655 dd0a 273d a96d 04e0 fcd3

      MSK:          c6d3 a6e0 ceea 951e b20d 74f3 2c30 61d0
                    680a 04b0 b086 ee87 00ac e3e0 b95f a026
                    83c2 87be ee44 4322 94ff 98af 26d2 cc78
                    3bac e75c 4b0a f7fd feb5 511b a8e4 cbd0

      EMSK:         7fb5 6813 838a dafa 99d1 40c2 f198 f6da
                    cebf b6af ee44 4961 1054 02b5 08c7 f363
                    352c b291 9644 b504 63e6 a693 5415 0147
                    ae09 cbc5 4b8a 651d 8787 a689 3ed8 536d
```

Acknowledgments

The authors would like to thank Guenther Horn, Joe Salowey, Mats Naslund, Adrian Escott, Brian Rosenberg, Laksminath Dondeti, Ahmad Muhanna, Stefan Rommer, Miguel Garcia, Jan Kall, Ankur Agarwal, Jouni Malinen, John Mattsson, Jesus De Gregorio, Brian Weis, Russ Housley, Alfred Hoenes, Anand Palanigounder, Michael Richardson, Roman Danyliw, Dan Romascanu, Kyle Rose, Benjamin Kaduk, Alissa Cooper, Erik Kline, Murray Kucherawy, Robert Wilton, Warren Kumari, Andreas Kunz, Marcus Wong, Kalle Jarvinen, Daniel Migault, and Mohit Sethi for their in-depth reviews and interesting discussions in this problem space.

Contributors

The test vectors in Appendix D were provided by Yogendra Pal and Jouni Malinen, based on two independent implementations of this specification.

Jouni Malinen provided suggested text for Section 6. John Mattsson provided much of the text for Section 7.1. Karl Norrman was the source of much of the information in Section 7.2.

Authors' Addresses

Jari Arkko
Ericsson
FI-02420 Jorvas
Finland
Email: jari.arkko@piuha.net

Vesa Lehtovirta
Ericsson
FI-02420 Jorvas
Finland
Email: vesa.lehtovirta@ericsson.com

Vesa Torvinen
Ericsson
FI-02420 Jorvas
Finland
Email: vesa.torvinen@ericsson.com

Pasi Eronen
Independent
Finland
Email: pe@iki.fi