Network Working Group
Request for Comments: 5448
Updates: 4187
Category: Informational

J. Arkko
V. Lehtovirta
Ericsson
P. Eronen
Nokia

May 2009

# Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA')

## Status of This Memo



## Copyright Notice



## Abstract

This specification defines a new EAP method, EAP-AKA', which is a small revision of the EAP-AKA (Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement) method. The change is a new key derivation function that binds the keys derived within the method to the name of the access network. The new key derivation mechanism has been defined in the 3rd Generation Partnership Project (3GPP). This specification allows its use in EAP in an interoperable manner. In addition, EAP-AKA' employs SHA-256 instead of SHA-1.

This specification also updates RFC 4187, EAP-AKA, to prevent bidding down attacks from EAP-AKA'.

Table of Contents

1. Introduction . . . . . 2
2. Requirements Language . . . . . 3
3. EAP-AKA' . . . . . 3
3.1. AT_KDF_INPUT . . . . . 6
3.2. AT_KDF . . . . . 8
3.3. Key Generation . . . . . 10
3.4. Hash Functions . . . . . 12
3.4.1. PRF' . . . . . 12
3.4.2. AT_MAC . . . . . 13
3.4.3. AT_CHECKCODE . . . . . 13
4. Bidding Down Prevention for EAP-AKA . . . . . 14
5. Security Considerations . . . . . 15
5.1. Security Properties of Binding Network Names . . . . . 18
6. IANA Considerations . . . . . 19
6.1. Type Value . . . . . 19
6.2. Attribute Type Values . . . . . 19
6.3. Key Derivation Function Namespace . . . . . 19
7. Contributors . . . . . 20
8. Acknowledgments . . . . . 20
9. References . . . . . 20
9.1. Normative References . . . . . 20
9.2. Informative References . . . . . 21
Appendix A. Changes from RFC 4187 . . . . . 23
Appendix B. Importance of Explicit Negotiation . . . . . 23
Appendix C. Test Vectors . . . . . 24

## 1. Introduction

This specification defines a new Extensible Authentication Protocol (EAP)[RFC3748] method, EAP-AKA', which is a small revision of the EAP-AKA method originally defined in [RFC4187]. What is new in EAP-AKA' is that it has a new key derivation function, specified in [3GPP.33.402]. 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. This specification defines the EAP encapsulation for AKA when the new key derivation mechanism is in use.

3GPP has defined a number of applications for the revised AKA mechanism, some based on native encapsulation of AKA over 3GPP radio access networks and others based on the use of EAP.

For making the new key derivation mechanisms usable in EAP-AKA, additional protocol mechanisms are necessary. Given that RFC 4187 calls for the use of CK (the encryption key) and IK (the integrity key) from AKA, existing implementations continue to use these. 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 attempt to use wrong keys without getting a proper error message. The change must also be secure against bidding down attacks that attempt to force the participants to use the least secure mechanism.

This specification therefore introduces a variant of the EAP-AKA method, called EAP-AKA'. This method can employ the derived keys CK' and IK' from the 3GPP specification and updates the used hash function to SHA-256 [FIPS.180-2.2002]. But it is otherwise equivalent to RFC 4187. 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: Appendix B explains why it is important to be explicit about the change of semantics for the keys, and why other approaches would lead to severe interoperability problems.

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 explains the security differences between EAP-AKA and EAP-AKA'. Section 6 describes the IANA considerations and Appendix A explains what updates to RFC 4187 EAP-AKA have been made in this specification. Appendix B explains some of the design rationale for creating EAP-AKA'. Finally, Appendix C provides test vectors.

## 2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

## 3. EAP-AKA'

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

- It uses the Type code 50, not 23 (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 [FIPS.180-2.2002], not SHA-1 [FIPS.180-1.1995] (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 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].

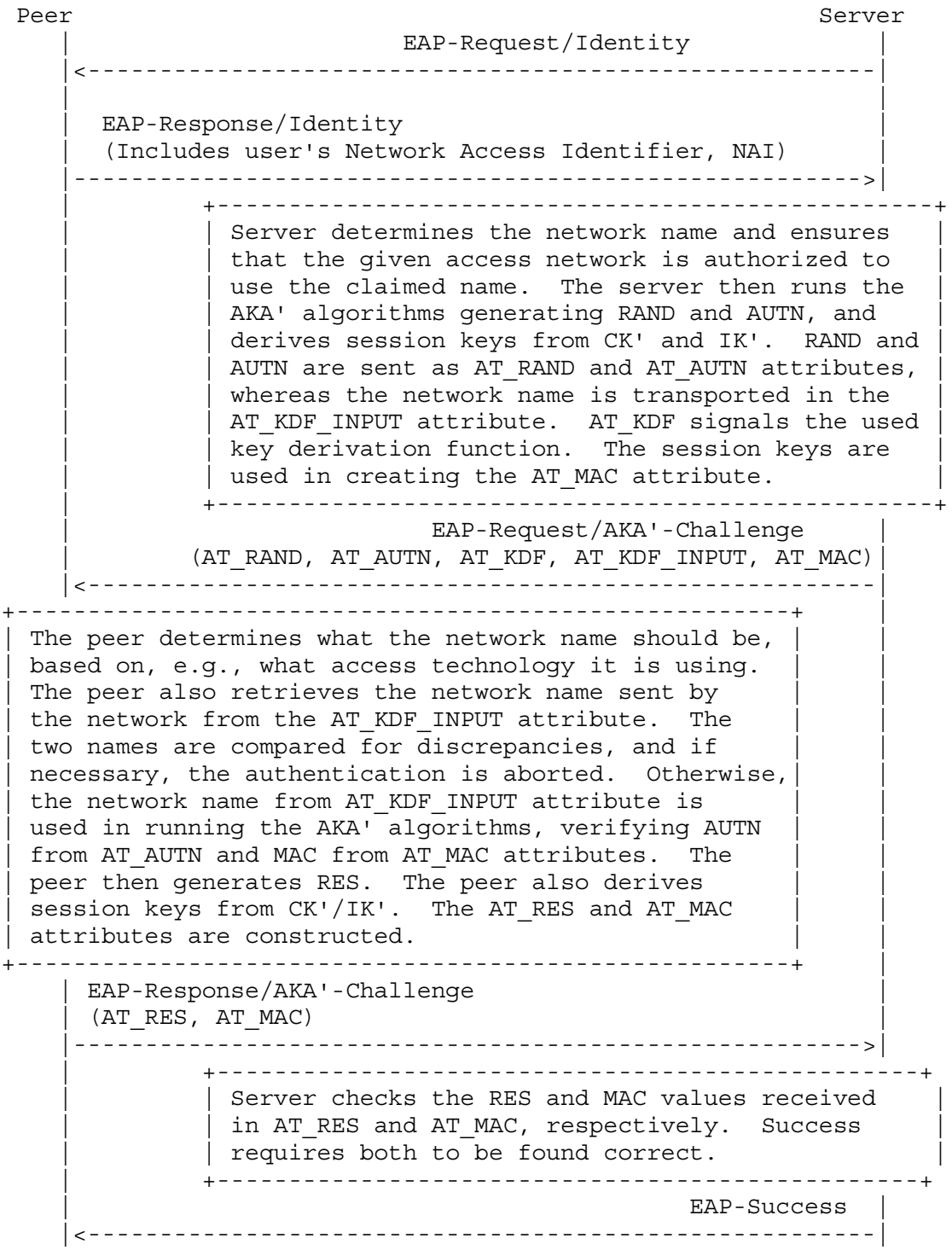
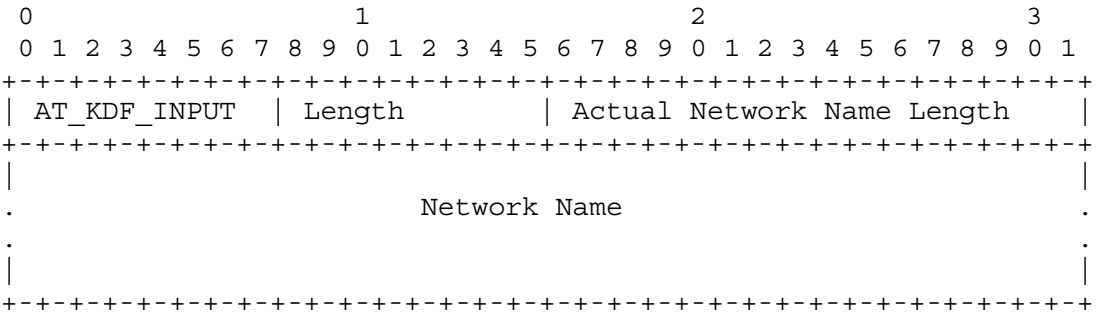



Figure 1: EAP-AKA' Authentication Process

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 International Mobile Subscriber Identifier (IMSI) based usernames. EAP-AKA' MUST use the leading character "6" (ASCII 36 hexadecimal) instead of "0" for IMSI-based permanent usernames. 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. Per [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. If it 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 [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, [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.

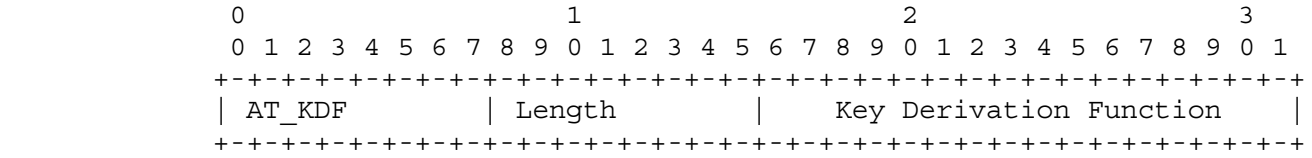
The allocation procedures in [3GPP.24.302] ensure that conflicts potentially arising from using the same name in different types of networks are avoided. The specification also has detailed rules about how a client can determine these based on information available to the client, such as the type of 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, 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. 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. An AKA'-Synchronization-Failure message is sent as a response to the newly received EAP-Request/AKA'-Challenge (the last message of the AT_KDF negotiation). 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 Generation

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

AT_KDF set to 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. PRF' is a new pseudo-random 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 [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].

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 [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 [FIPS.180-2.2002], not SHA-1 [FIPS.180-1.1995] as in EAP-AKA. This requires a change to the pseudo-random 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 [RFC4306]). The function takes two arguments. K is a 256-bit value and S is an octet 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.

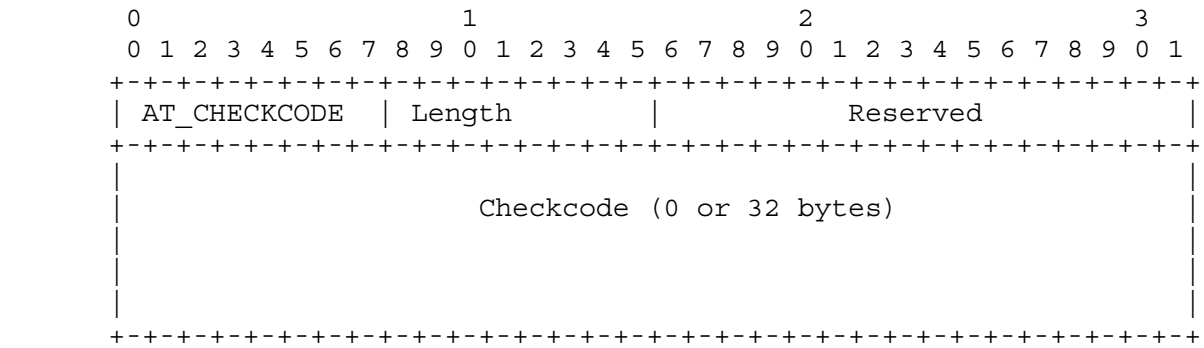
### 3.4.2. AT_MAC

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:



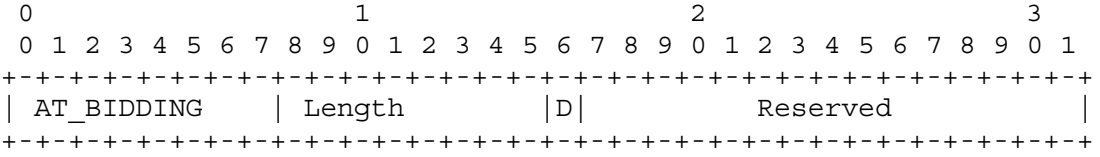

Second, the checkcode is a hash value, calculated with SHA-256 [FIPS.180-2.2002], over the data specified in Section 10.13 of [RFC4187].

## 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 new 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 as defined in this RFC. It is only included in EAP-AKA messages. This is based on the assumption that EAP-AKA' is always preferable (see Section 5). 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, 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 5) that EAP-AKA' is always stronger than EAP-AKA. As a result, there is no need to prevent bidding "down" attacks in the other direction, i.e., attackers forcing the endpoints to use EAP-AKA'.

## 5. 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 SHA-256 is at least as secure as SHA-1. 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. 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.

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, SHA-256, is used instead of 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 pseudo-random 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 5.1.

EAP-AKA' uses a pseudo-random function modeled after the one used in IKEv2 [RFC4306] 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 the following section for more discussion.

### 5.1. 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, [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 [3GPP.24.302]. See also [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.

## 6. IANA Considerations

### 6.1. Type Value

EAP-AKA' has the EAP Type value 50 in the Extensible Authentication Protocol (EAP) Registry under Method Types. Per Section 6.2 of [RFC3748], this allocation can be made with Designated Expert and Specification Required.

### 6.2. Attribute Type Values

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

However, a new Attribute Type value (23) in the non-skippable range has been assigned for AT_KDF_INPUT (Section 3.1) in the EAP-AKA and EAP-SIM Parameters registry under Attribute Types.

Also, a new Attribute Type value (24) in the non-skippable range has been assigned for AT_KDF (Section 3.2).

Finally, a new Attribute Type value (136) in the skippable range has been assigned for AT_BIDDING (Section 4).

### 6.3. Key Derivation Function Namespace

IANA has also created a new namespace for EAP-AKA' AT_KDF Key Derivation Function Values. This namespace exists under the EAP-AKA and EAP-SIM Parameters registry. The initial contents of this namespace are given below; new values can be created through the Specification Required policy [RFC5226].

| Value | Description | Reference |
|---|---|---|
| 0 | Reserved | [RFC5448] |
| 1 | EAP-AKA' with CK'/IK' | [RFC5448] |
| 2-65535 | Unassigned | |

## 7. Contributors

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

## 8. 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, Brian Weis, Russ Housley, and Alfred Hoenes for their in-depth reviews and interesting discussions in this problem space.

## 9. References

### 9.1. Normative References

[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 8)", 3GPP Technical Specification 24.302, December 2008.

[3GPP.33.102] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security architecture (Release 8)", 3GPP Technical Specification 33.102, December 2008.

[3GPP.33.402] 3GPP, "3GPP System Architecture Evolution (SAE); Security aspects of non-3GPP accesses; Release 8", 3GPP Technical Specification 33.402, December 2008.

[FIPS.180-2.2002] National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-2, August 2002, <http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC4187] Arkko, J. and H. Haverinen, "Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)", RFC 4187, January 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

## 9.2. Informative References

[3GPP.23.003] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Numbering, addressing and identification (Release 8)", 3GPP Draft Technical Specification 23.003, December 2008.

[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 8)", 3GPP Technical Specification 35.208, December 2008.

[FIPS.180-1.1995] National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-1, April 1995, <http://www.itl.nist.gov/fipspubs/fip180-1.htm>.

[RFC4186] Haverinen, H. and J. Salowey, "Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)", RFC 4186, January 2006.

[RFC4284] Adrangi, F., Lortz, V., Bari, F., and P. Eronen, "Identity Selection Hints for the Extensible Authentication Protocol (EAP)", RFC 4284, January 2006.

[RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[RFC5113] Arkko, J., Aboba, B., Korhonen, J., and F. Bari, "Network Discovery and Selection Problem", RFC 5113, January 2008.

[RFC5247] Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, August 2008.

## Appendix A. Changes from RFC 4187

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 in RFC 4187 (it uses CK and IK, not CK' and IK'); neither is any processing of the AMF bit added to RFC 4187.

## Appendix B. 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 mechanism. 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 C. 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 [3GPP.35.208].

The last two cases use artificial values as the output of AKA, and is 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
```

Authors' Addresses

Jari Arkko
Ericsson
Jorvas 02420
Finland

EMail: jari.arkko@piuha.net

Vesa Lehtovirta
Ericsson
Jorvas 02420
Finland

EMail: vesa.lehtovirta@ericsson.com

Pasi Eronen
Nokia Research Center
P.O. Box 407
FIN-00045 Nokia Group
Finland

EMail: pasi.eronen@nokia.com