

Workgroup: Internet Engineering Task Force  
Internet-Draft: draft-rieckers-emu-eap-ute-00  
Published: 7 March 2022  
Intended Status: Standards Track  
Expires: 8 September 2022  
Authors: J.-F. Rieckers  
DFN

## **User-assisted Trust Establishment (EAP-UTE)**

### **Abstract**

The Extensible Authentication Protocol (EAP) provides support for multiple authentication methods. This document defines the EAP-UTE authentication method for a User-assisted Trust Establishment between the peer and the server. The EAP method is intended for bootstrapping Internet-of-Things (IoT) devices without preconfigured authentication credentials. The trust establishment is achieved by transmitting a one-directional out-of-band (OOB) message between the peer and the server to authenticate the in-band exchange. The peer must have a secondary input or output interface, such as a display, camera, microphone, speaker, blinking light, or light sensor, so that dynamically generated messages with tens of bytes in length can be transmitted or received.

### **About This Document**

This note is to be removed before publishing as an RFC.

Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-rieckers-emu-eap-ute/>.

Discussion of this document takes place on the EAP Method Update (emu) Working Group mailing list (<mailto:emu@ietf.org>), which is archived at <https://mailarchive.ietf.org/arch/browse/emu/>.

### **Status of This Memo**

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 8 September 2022.

## Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

- [1. Introduction](#)
- [2. Conventions and Definitions](#)
- [3. EAP-UTE protocol](#)
  - [3.1. Protocol Overview](#)
  - [3.2. Messages](#)
    - [3.2.1. General Message format](#)
    - [3.2.2. Server greeting](#)
    - [3.2.3. Client greeting](#)
    - [3.2.4. Server Keyshare](#)
    - [3.2.5. Client Finished](#)
    - [3.2.6. Client Completion Request](#)
    - [3.2.7. Server Completion Response](#)
    - [3.2.8. Client Keyshare](#)
  - [3.3. Protocol Sequence](#)
    - [3.3.1. Initial Exchange](#)
    - [3.3.2. User-assisted out-of-band step](#)
    - [3.3.3. Waiting Exchange](#)
    - [3.3.4. Completion Exchange](#)
    - [3.3.5. Reconnect Exchange](#)
  - [3.4. MAC and OOB calculation and Key derivation](#)
  - [3.5. Error handling](#)
- [4. Security Considerations](#)
  - [4.1. EAP Security Claims](#)
- [5. IANA Considerations](#)
- [6. Implementation Status](#)
- [7. Differences to RFC 9140 \(EAP-NOOB\)](#)
  - [7.1. Different encoding](#)
  - [7.2. Implicit transmission of peer state](#)
  - [7.3. Extensibility](#)

## [8. References](#)

### [8.1. Normative References](#)

### [8.2. Informative References](#)

### [Acknowledgements](#)

### [Author's Address](#)

## **1. Introduction**

This document describes a method for registration, authentication, and key derivation for network-connected devices, especially with low computational power and small or no interaction interfaces, such as devices that are part of the Internet of Things (IoT). These devices may come without preconfigured trust anchors or have no possibility to receive a network configuration that enables them to connect securely to a network.

This document uses the basic design principle behind the EAP-NOOB method described in [[RFC9140](#)] and aims to improve some key elements of the protocol to better address the needs for IoT devices. This is mainly achieved by using CBOR with numeric keys instead of JSON to encode the exchanged messages.

TODO: The EAP-UTE protocol also allows extensions, they are still TBD. Basically, the messages can just include additional fields with newly defined meanings.

The possible problems of EAP-NOOB are discussed in [[I-D.draft-rieckers-emu-eap-noob-observations](#)]. This document provides a specification which aims to address these concerns.

## **2. Conventions and Definitions**

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.

TODO frequently used terms

authenticator

peer

server

## **3. EAP-UTE protocol**

This section defines the EAP-UTE method.

### 3.1. Protocol Overview

TODO: The introduction text is basically copied from RFC9140. Should be reworded.

The EAP-UTE method execution spans two or more EAP conversations, called Exchanges in this specification. Each Exchange consists of several EAP request-response pairs. In order to give the user time to deliver the OOB message between the peer and the server, at least two separate EAP conversations are needed.

The overall protocol starts with a version and cryptosuite negotiation and peer detection. Depending of the current state of the peer and server, different exchanges are selected.

If the server or the peer are in the unregistered state, peer and server exchange nonces and keys for the Ephemeral Elliptic Curve Diffie-Hellman. This is called the Initial Exchange. The Initial Exchange results in an EAP-Failure, since neither the server nor the peer are authenticated.

After the Initial Exchange, the user-assisted step of trust establishment takes place. The user delivers one OOB message either from the peer to the server or from the server to the peer.

While peer and server are waiting for completion of the OOB Step, the peer **MAY** probe the server by reconnecting, to check for successful transmission of the OOB message. This probe request will result in a Waiting Exchange and EAP-Failure, if the server has not yet received the OOB message.

If either the server or the peer have received the OOB message, the probe request will result in a Completion Exchange. In the Completion Exchange, peer and server exchange message authentication codes over the previous in-band messages and the OOB message. The Completion Exchange may result in EAP-Success. Once the peer and server have performed a successful Completion Exchange, both endpoints store the created association in persistent storage.

After a successful Completion Exchange, the peer and server can use the Reconnect Exchange, to create a new association with new cryptographic bindings. The user-assisted OOB step is not necessary, since the peer and server can infer the mutual authentication by using the persistent data stored after the Completion Exchange.

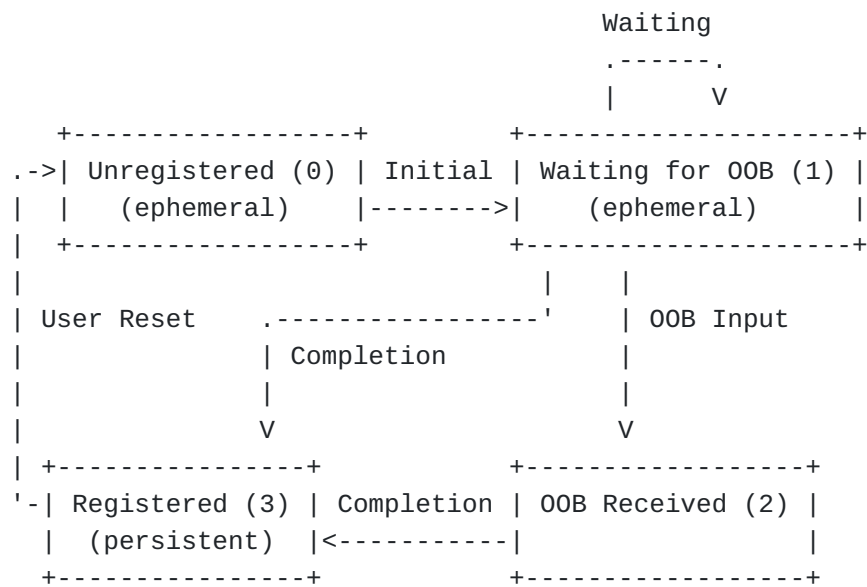


Figure 1: EAP-UTE Server-Peer Association State Machine

## 3.2. Messages

### 3.2.1. General Message format

All EAP-UTE messages consist of the following parts:

**type:** one octet to indicate the type of the message

**length:** two octets indicating the length of the following message payload, not including the optional MAC value

**payload:** the CBOR encoded message

**MAC:** (optional) the message authentication code

Remark from the author:

This format is just a first draft. It allows a very simple MAC calculation, since the MACs can just consist of the concatenated previous messages. This also allows an easy addition of extensions, since the extension payloads are automatically included in the MAC calculation, if they are part of the CBOR payload.

The message payloads are encoded in CBOR [[RFC8949](#)] as maps.

In [Table 1](#) the different message fields, their assigned mapkey and the type are listed.

Mapkey	Type	Label	Description
1	Array of Integers	Versions	

Mapkey	Type	Label	Description
			The versions supported by the server. For this document the version is 1
2	Integer	Version	The version selected by the peer
3	Array?	Ciphers	The ciphers supported by the server. TODO: Not yet sure how to define them.
4	Integer?	Cipher	The cipher selected by the peer
5	Integer	Directions	The OOB-Directions supported by the server. 0x01 for peer-to-server, 0x02 for server-to-peer, 0x03 for both
6	Integer	Direction	The OOB-Direction selected by the peer. <b>SHOULD</b> be either 0x01 or 0x02, but <b>MAY</b> be 0x03 for both directions
7	Map	ServerInfo	Information about the server, e.g. a URL for OOB-message-submission
8	Map	PeerInfo	Information about the peer, e.g. manufacturer/serial number
9	bytes	Nonce_P	Peer Nonce
10	bytes	Nonce_S	Server Nonce
11	?	Key_P	Peer's ECDHE key according to the chosen cipher
12	?	Key_S	Server's ECDHE key
13	null	MAC_S	Indication that Server MAC is included
14	null	MAC_P	Indication that Peer MAC is included
15	text	PeerId	Peer Identifier
16	bytes	OOB-Id	Identifier of the OOB message
17	int	RetryInterval	Number of seconds to wait after a failed Completion Exchange
18	Map	AdditionalServerInfo	Additional information about the server. TODO: not sure about this yet.

Table 1: Mapkeys for CBOR encoding

The inclusion of MAC\_S or MAC\_P indicate that the MAC value is appended to the message. The length of the MAC field is determined by the used cryptosuite. A message **MUST NOT** contain both MAC\_S and MAC\_P, only one of these values can be present in a message.

TODD: Depending on the definition of the Cipher Suites, the format for Ciphers and Cipher might change, as well as Key\_P and Key\_S. The most immediate choice would be COSE [[RFC8152](#)]. But maybe there are better choices out there.

#### **3.2.1.1. Thoughts about the message format**

EAP-NOOB [[RFC9140](#)] uses JSON as encoding. Problems of using JSON are discussed in section 2.1 of [[I-D.draft-rieckers-emu-eap-noob-observations](#)].

For this specification, the following encodings have been considered:

- \*Static encoding

This allows a minimal number of bytes and requires a minimal amount of parsing, since the format and order of the message fields is specified exactly. However, this encoding severely affects the extensibility, unless a specific extension format is used. The specification of this protocol also has optional fields in some message types, so this would also have to be addressed.

- \*CBOR with static fields (e.g. Array)

This approach has a slightly higher number of bytes than the static encoding, but allows an easier extensibility. The required fields can be specified, so the order of the protocol field is static and a parser has minimal effort to parse the protocol fields. However, this might be problematic in future protocol versions, when new fields are introduced. Like with static encoding, this also requires a mechanism for optional fields in the different message types.

- \*CBOR map with numeric keys

To mitigate the problems of optional fields while keeping the parsing effort low, CBOR maps with numeric keys can be used. All protocol fields are identified by a unique identifier, specified in this document. A parser can simply loop through the CBOR map. Since CBOR maps have a canonical order, minimal implementations may rely on this fact to parse the information needed.

On the basis of this discussion, this draft will use a CBOR map as message encoding. However, this is just a first draft and suggestions for other message formats are highly welcome.

### **3.2.2. Server greeting**

\*Message Type: 1

\*Required Attributes:

- Versions

- Ciphers

- ServerInfo

- Directions

\*Optional Attributes:

- RetryInterval?

### **3.2.3. Client greeting**

\*Message Type: 2

\*Required Attributes:

- Version

- Cipher

- PeerInfo

- Direction

- Nonce\_P

- Key\_P

\*Optional Attributes:

- PeerId

### **3.2.4. Server Keyshare**

\*Message Type: 3

\*Required Attributes:

- Key\_S

- Nonce\_S

- MAC\_S



\*Optional Attributes:

-PeerId

-AdditionalServerInfo?

-RetryInterval?

TODO: Maybe make MAC\_S optional, if used in Initial Exchange

### **3.2.5. Client Finished**

\*Message Type: 4

\*Required Attributes:

-MAC\_P

TODO: Maybe make MAC\_P optional, if used in Initial Exchange

### **3.2.6. Client Completion Request**

\*Message Type: 5

\*Required Attributes:

-Nonce\_P

-PeerId

\*Optional Attributes:

-OOB-Id

### **3.2.7. Server Completion Response**

\*Message Type: 6

\*Required Attributes:

-Nonce\_S

-MAC\_S

\*Optional Attributes:

-OOB-Id

### **3.2.8. Client Keyshare**

\*Message Type: 7

\*Required Attributes:

-PeerId

-Nonce\_P

-Key\_P

### 3.3. Protocol Sequence

After reception of the EAP-Response/Identity packet, the server always answers with a Server Greeting packet (Type 1). This Server Greeting contains the supported protocol versions, ciphers and OOB directions along with the ServerInfo.

Depending on the peer state, the peer chooses the next packet. If the peer is in the unregistered state and does not yet have an ephemeral or persistent state, it chooses the Client Greeting, which starts the Initial Handshake.

If the peer is in the Waiting for OOB or OOB Received state, the Initial Exchange has completed and the OOB step needs to take place. If the negotiated direction is from server to peer, the peer **SHOULD NOT** try to reconnect until the peer received an OOB message. If the negotiated direction is from peer to server, the peer can probe the server at regular intervals to check if the OOB message to the server has been delivered. The peer will send a Client Completion Request to initiate the Waiting/Completion Exchange.

If the peer is in the Registered state, it may choose between three different Reconnect Exchanges. If the peer wants a reconnect without new key exchanges, it will send a Client Completion Request, starting the Reconnect Exchange without ECDHE. If the peer wants to reconnect with new key exchanges, it will send a Client Key Share packet, which starts the Reconnect Exchange with new ECDHE exchange. The third option is a reconnect with a new version or cipher, this is TBD.

#### 3.3.1. Initial Exchange

The Initial Exchange comprises of the following packets:

After the Server Greeting common to all exchanges, the peer sends a Client Greeting packet. The Client Greeting contains the peer's chosen protocol version, cipher and direction of the OOB message. The peer **MUST** only choose values for these fields offered by the server in it's Server Greeting. For Direction the peer **SHOULD** choose either 0x01 or 0x02 if the server offered 0x03. Additionally, the Client Greeting contains PeerInfo, a nonce and the peer's ECDHE public key.

The server will then answer with a Server Keyshare packet. The packet contains a newly allocated PeerId, the server's nonce and ECDHE public key and the message authentication code MAC\_S.

The peer then answers with a Client Finished packet, containing the peer's message authentication code MAC\_P.

Since no authentication has yet been achieved, the server then answers with an EAP-Failure.

```

EAP Peer                               Authenticator EAP Server
|                                         |
|<- EAP-Request/Identity -|
|
|-- EAP-Response/Identity ----->|
|   (NAI=new@eap-ute.arpa)
|
|<- EAP-Request/EAP-UTE -----|
|   SERVER GREETING (1)
|   Versions, Ciphers, ServerInfo,
|   Directions
|
|-- EAP-Response/EAP-UTE ----->|
|   CLIENT GREETING (2)
|   Version, Cipher, PeerInfo,
|   Direction, Nonce_P, Key_P
|
|<- EAP-Request/EAP-UTE -----|
|   SERVER KEYSHARE (3)
|   PeerId, Key_S, Nonce_S, MAC_S
|
|-- EAP-Response/EAP-UTE ----->|
|   CLIENT FINISHED (4)
|   MAC_P
|
|<- EAP-Failure -----|
|

```

Figure 2: Initial Exchange

TODO: Do I need MACs here? What are they really for? (see [Section 3.4](#) for more thoughts on this)

### 3.3.2. User-assisted out-of-band step

After the completed Initial Exchange, the peer or the server, depending on the negotiated direction, will generate an OOB message.

Details still TBD.

### 3.3.3. Waiting Exchange

The Waiting Exchange is performed if neither the server nor the peer have received an OOB message yet.

The peer probes the server with a Client Completion Request. In this packet the peer omits the optional OOB-Id field. If the OOB message is delivered from the peer to the server, the server may have received an OOB message already. To allow the server to complete the association, the peer includes a nonce, along with the allocated PeerId. The nonce **MAY** be repeated for all Client Completion Requests while waiting for the completion.

If the server did not receive an OOB message, it answers with an EAP-Failure.

EAP Peer	Authenticator	EAP Server
<- EAP-Request/Identity -		
-- EAP-Response/Identity ----->		
(NAI=waiting@eap-ute.arpa)		
<- EAP-Request/EAP-UTE -----		
SERVER GREETING (1)		
Versions, Ciphers, Server Info,		
Directions		
-- EAP-Response/EAP-UTE ----->		
CLIENT COMPLETION REQUEST (5)		
PeerId, Nonce_P		
<- EAP-Failure -----		

Figure 3: Waiting Exchange

### 3.3.4. Completion Exchange

The Completion Exchange is performed to finish the mutual trust establishment.

As in the Waiting Exchange, the peer probes the server with a Client Completion Request. The nonce of the previous Client Completion Requests which did not lead to a completion **MAY** be repeated. If the peer has received an OOB message, the peer will include the OOB-Id in the Completion Request. If the peer did not include an OOB-Id, the server will include the OOB-Id of its received OOB message. In the unlikely case that both directions are negotiated and an OOB

message is delivered from the peer to the server and from the server to the peer at the same time, as a tiebreaker, the 00B message from the server to the peer is chosen.

The server generates a new nonce, calculates MAC\_S according to [Section 3.4](#) and sends a Server Completion Response to the peer.

The peer will then calculate the MAC\_P value and send a Client Finished message to the server.

The server then answers with an EAP-Success.

EAP Peer	Authenticator	EAP Server
<- EAP-Request/Identity -		
-- EAP-Response/Identity ----->		
(NAI=waiting@eap-ute.arpa)		
<- EAP-Request/EAP-UTE -----		
SERVER GREETING (1)		
Versions, Ciphers, Server Info,		
Directions		
-- EAP-Response/EAP-UTE ----->		
CLIENT COMPLETION REQUEST (5)		
PeerId, Nonce_P, [00B-Id]		
<- EAP-Request/EAP-UTE -----		
SERVER COMPLETION RESPONSE (6)		
[00B-Id], Nonce_S, MAC_S		
-- EAP-Response/EAP-UTE ----->		
CLIENT FINISHED (4)		
MAC_P		
<- EAP-Success -----		

Figure 4: Completion Exchange

### 3.3.5. Reconnect Exchange

The Reconnect Exchange is performed if both the peer and the server are in the registered state.

For a reconnect without new exchanging of ECDHE keys, the peer will answer to the Server Greeting with a Client Completion Request, including the PeerId and a nonce.

To distinguish a Reconnect Exchange from a Waiting/Completion Exchange, the server will look up the saved states for the transmitted PeerId. If the server has a persistent state saved, it will choose the Reconnect Exchange, otherwise it will choose the Waiting Exchange.

The server will then generate a nonce and the MAC\_S value according to [Section 3.4](#) and send a Server Completion Response with the nonce and MAC\_S value.

The peer then sends a Client Finished message, containing the computed MAC\_P value.

The server then answers with an EAP-Success.

EAP Peer	Authenticator	EAP Server
<- EAP-Request/Identity -		
-- EAP-Response/Identity ----->		
<- EAP-Request/EAP-UTE -----		
SERVER GREETING (1)		
Versions, Ciphers, Server Info,		
Directions		
-- EAP-Response/EAP-UTE ----->		
CLIENT COMPLETION REQUEST (5)		
PeerId, Nonce_P		
<- EAP-Request/EAP-UTE -----		
SERVER COMPLETION RESPONSE (6)		
Nonce_S, MAC_S		
-- EAP-Response/EAP-UTE ----->		
CLIENT FINISHED (4)		
MAC_P		
<- EAP-Success -----		

Figure 5: Reconnect Exchange without new ECDHE exchange

For a Reconnect Exchange with new ECDHE exchange, the peer will send a Client Keyshare in response to the Server Greeting. The Client Keyshare will include the PeerId, a nonce and a new ECDHE key.

The server will also generate a new ECDHE key, a nonce and compute MAC\_S according to [Section 3.4](#).

The peer will then calculate the MAC\_P value and send a Client Finished message to the server.

The server then answers with an EAP-Success.

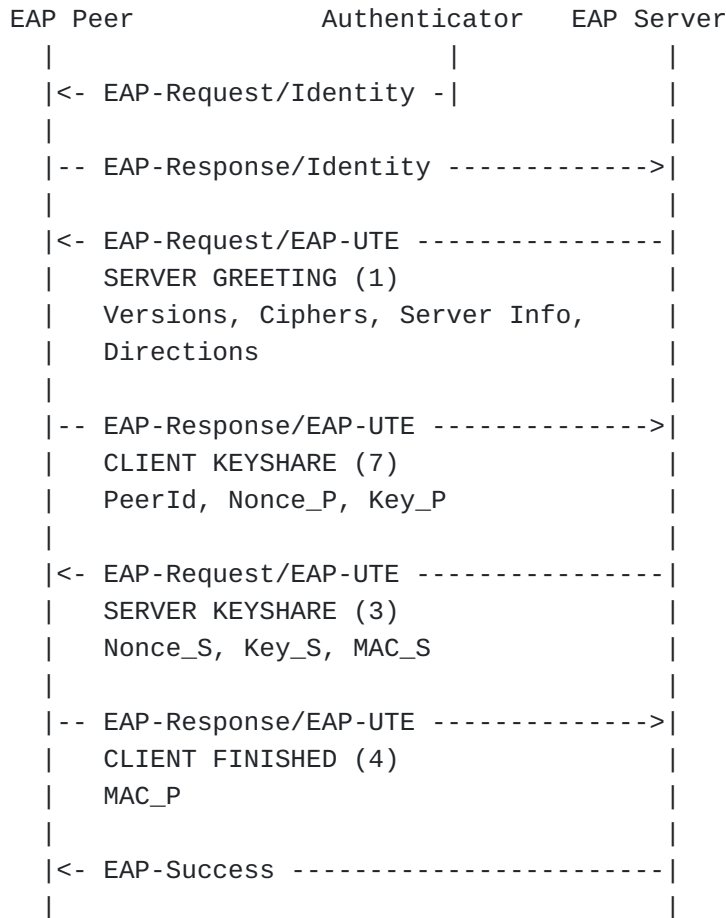


Figure 6: Reconnect Exchange with new ECDHE exchange

TODO: Reconnect exchange with updated version or cipher suite

### 3.4. MAC and OOB calculation and Key derivation

For the MAC calculation, the exchanged messages up to the current message are concatenated into the "Messages" field. This field consists for each message of the one octet message type, the two octet encoding of the length and the CBOR encoded message payload. The optional MAC value at the end of the message is omitted for the MAC calculation. For the calculation of the MAC\_S value, the Messages field also includes the Server Keyshare/Server Completion Response message. For MAC\_P the Client Finished message is omitted, so both MAC\_P and MAC\_S have the same input.

For the following definition || denotes a concatenation.

Messages = Type\_1 || Length\_1 || Payload\_1 || ... || Type\_n ||  
Length\_n || Payload\_n

OOB-Id = H(OOB-Nonce || Messages)

TODO: Calculation of MAC\_S/MAC\_P

Idea: For initial exchange MAC\_S/MAC\_P = HMAC(key, Messages), and  
for completion exchange MAC\_S/MAC\_P = HMAC(key, prev\_MAC\_S ||  
prev\_MAC\_P || Messages)

TODO: Key derivation. Here I have a problem. If I want to send MACs  
in the initial exchange, I somehow have to make a key derivation  
already. Maybe this is too costly. Maybe it would only be necessary  
to save a Hash of the previous messages during the InitialExchange  
and include it in the KDF to cryptographically bind the Server/  
PeerInfo to the connection. This way, the Initial Exchange wont have  
MACs, the integrity check is done completely by exchanging of MACs  
during the Completion Exchange. This will probably be more clear in  
the -01 draft version.

### **3.5. Error handling**

TBD

## **4. Security Considerations**

This document has a lot of security considerations, however they  
remain TBD

### **4.1. EAP Security Claims**

TODO. See [[RFC3748](#)], section 7.2.1

## **5. IANA Considerations**

This document has IANA actions, if approved. What they are exactly  
needs to be defined in detail.

The EAP Method Type number for EAP-UTE needs to be assigned. The  
reference implementation will use 255 (Experimental) for now.

Like EAP-NOOB, this draft will probably use a .arpa domain, in this  
case probably eap-ute.arpa, as default NAI realm.

Additionally, the IANA should create registries for the message  
types and the message field mapkeys.



## 6. Implementation Status

Note to RFC Editor: Please remove this entire section before publication.

There are no implementations yet.

## 7. Differences to RFC 9140 (EAP-NOOB)

In this section the main differences between EAP-NOOB and EAP-UTE are discussed. Some problems of [\[RFC9140\]](#) are discussed in [\[I-D.draft-rieckers-emu-eap-noob-observations\]](#).

### 7.1. Different encoding

EAP-UTE uses CBOR instead of JSON. More text TBD.

### 7.2. Implicit transmission of peer state

In EAP-NOOB all EAP exchanges start with the same common handshake, which mainly serves the purpose of detecting the current peer state.

The server initiates the EAP conversation by sending a Type 1 message without any further content, to which the peer responds by sending its PeerId, if it was assigned, and its PeerState.

In EAP-UTE, this peer state transmission is done implicitly by the peer's choice of response to the Server Greeting.

This adds probably unnecessary bytes in the first packet from the server to the peer, since the peer already knows the server's supported versions, ciphers and the ServerInfo in the later exchanges, especially in the Waiting/Completion Exchange. However, this increased number of bytes is negligible in comparison to the elevated expense of an additional roundtrip, since this would significantly increase the authentication time, especially if the EAP packets are routed through a number of proxies.

### 7.3. Extensibility

The EAP-NOOB standard does not specify how to deal with unexpected labels in the message, which could be used to extend the protocol. This specification will explicitly allow extensions. They are still TBD.

## 8. References

### 8.1. Normative References

[\[RFC2119\]](#)

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. Levkowitz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <<https://www.rfc-editor.org/info/rfc3748>>.

[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>>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <<https://www.rfc-editor.org/info/rfc8949>>.

## 8.2. Informative References

### [I-D.draft-rieckers-emu-eap-noob-observations]

Rieckers, J., "Observations about EAP-NOOB (RFC 9140)", Work in Progress, Internet-Draft, draft-rieckers-emu-eap-noob-observations-00, 7 March 2022, <<https://www.ietf.org/archive/id/draft-rieckers-emu-eap-noob-observations-00.txt>>.

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <<https://www.rfc-editor.org/info/rfc8152>>.

[RFC9140] Aura, T., Sethi, M., and A. Peltonen, "Nimble Out-of-Band Authentication for EAP (EAP-NOOB)", RFC 9140, DOI 10.17487/RFC9140, December 2021, <<https://www.rfc-editor.org/info/rfc9140>>.

## Acknowledgements

TBD

## Author's Address

Jan-Frederik Rieckers  
Deutsches Forschungsnetz | German National Research and Education  
Network  
Alexanderplatz 1  
10178 Berlin  
Germany

Email: [rieckers@dfn.de](mailto:rieckers@dfn.de)

URI: [www.dfn.de](http://www.dfn.de)