Network Working Group
Internet-Draft
Expires: April 22, 2009

Y. Ohba (Editor)
Toshiba

October 19, 2008

Derivation, delivery and management of EAP based keys for handover and re-authentication
draft-ietf-hokey-key-mgm-04

Status of this Memo

Abstract

This document describes a mechanism for delivering a usage-specific root key (USRK), a domain-specific root key (DSRK) and a usage-specific domain-specific root key (USDSRK) using RADIUS. The root keys are derived as part of an Extended Master Session Key (EMSK) hierarchy in Extensible Authentication Protocol (EAP), and delivered from a server to an intended third-party key holder. The mechanism supports different scenarios for key delivery, depending on the type of keys being delivered. The mechanism description includes the definition for a key distribution exchange (KDE) protocol.

Table of Contents

1. Introduction

The ability of the Extensible Authentication Protocol (EAP) framework [RFC3748] to incorporate desired authentication methods and generate master session keys (an MSK and an EMSK) [RFC5247] has led to the idea of using the MSK and/or the EMSK for bootstrapping further keys for a variety of security mechanisms. The MSK has been widely used for bootstrapping the wireless link security associations between the peer and the network attachment points. Issues arising from the use of the MSK and the current bootstrapping methods when it comes to mobility performance and security are described in [RFC5169]. The EMSK is used for bootstrapping of keys for use cases that are not covered by the use case of the MSK [RFC5295]. For instance [RFC5295] defines a way to create a usage-specific root key (USRK) for bootstrapping security for a specific use case. [RFC5295] also defines ways to create domain-specific root keys for bootstrapping security of a set of services within a domain.

Along with those lines, this document provides a specification of a mechanism for secure delivery of such EMSK child keys from the EAP server, holding the EMSK, to the intended third-party destinations by using RADIUS [RFC2865], [RFC3579]. This document also describes security requirements on delivery for this keying material over RADIUS.

2. Terminology

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].

USRK: Usage-Specific Root Key. A root key generated from the EMSK and is used for a specific usage that is authorized for a peer, following an EAP authentication. USRKs are domain independent.

USR-KH: USRK Holder. The USR-KH is responsible for receiving, holding and protecting the USRK derived directly from the EMSK.

DSRK: Domain-Specific Root Key. A root key generated from the EMSK and is used within a specific domain that EAP-authenticated peer is authorized to receive services from or roam into. DSRKs are usage independent.

DSR-KH: DSRK holder. The DSRK holder is responsible for receiving, holding and protecting the DSRK derived directly from the EMSK.

DSUSRK: Domain-Specific Usage-Specific Root Key. A root key generated from the DSRK and is used for a specific usage within a specific domain that an EAP-authenticated peer is authorized to receive services from. DSUSRKs are both usage and domain dependent.

DSUSR-KH: DSRK holder. The DSUSRK holder is responsible for receiving, holding and protection of the DSUSRK.

3. Key Delivery Architecture

The EAP server is only responsible for performing EAP authentication and is not expected to be involved in any service authorization decisions, nor is the EAP server aware of the future service requests at the time of authentication. The authorization decisions based on the user service profile and provisioning of services including support for service security is expected to happen by third-parties, such as AAA servers or service servers. When EAP-based keying is used, such servers will cache and use the USRKs, DSRKs or DSUSRKs, generated from the EMSK, as root keys for derivation of further keys to secure the services they are providing. Thus they are considered third-party key holders (KH) with respect to the initial two EAP parties (EAP peer and server). However, since the EMSK cannot be exported from the EAP server, such third-parties need to request the EAP server to generate the relevant root key (e.g., a USRK) from the EMSK and deliver the requested key to them. The third-party needs to provide the required input data to be used along with the pseudo random function (PRF) to the EAP server to generate the requested key. The following types of top level key holders can be envisioned:

USRK holder (USR-KH): An entity acting as a recipient and then holder of the usage-specific root key (USRK). The USR-KH is responsible for derivation and distribution of any child keys derived from the USRK for that specific usage. We assume that the USR-KH and the EAP server are separate entities, logically if not physically, and we specify the delivery of the USRK from EAP server to USR-KH accordingly.

DSRK holder (DSR-KH): An entity acting as a recipient and then holder of the domain-specific root key (DSRK). The DSR-KH is responsible for derivation and distribution of any child keys derived from the DSRK for that specific domain. The most likely realization of a DSR-KH is a AAA server in the corresponding domain, responsible for setting the policies for usage of the DSRK within the domain.

DSUSRK holder (DSUSR-KH): An entity acting as a recipient and then holder of the domain-specific usage-specific root key (DSUSRK) delivered from the EAP server. The DSUSR-KH is responsible for derivation and distribution of any child keys derived from the DSUSRK for that specific domain and usage. The most likely realization of a DSUSR-KH is a AAA server in the corresponding domain, responsible for the service offered within the domain for the specific usage at hand.

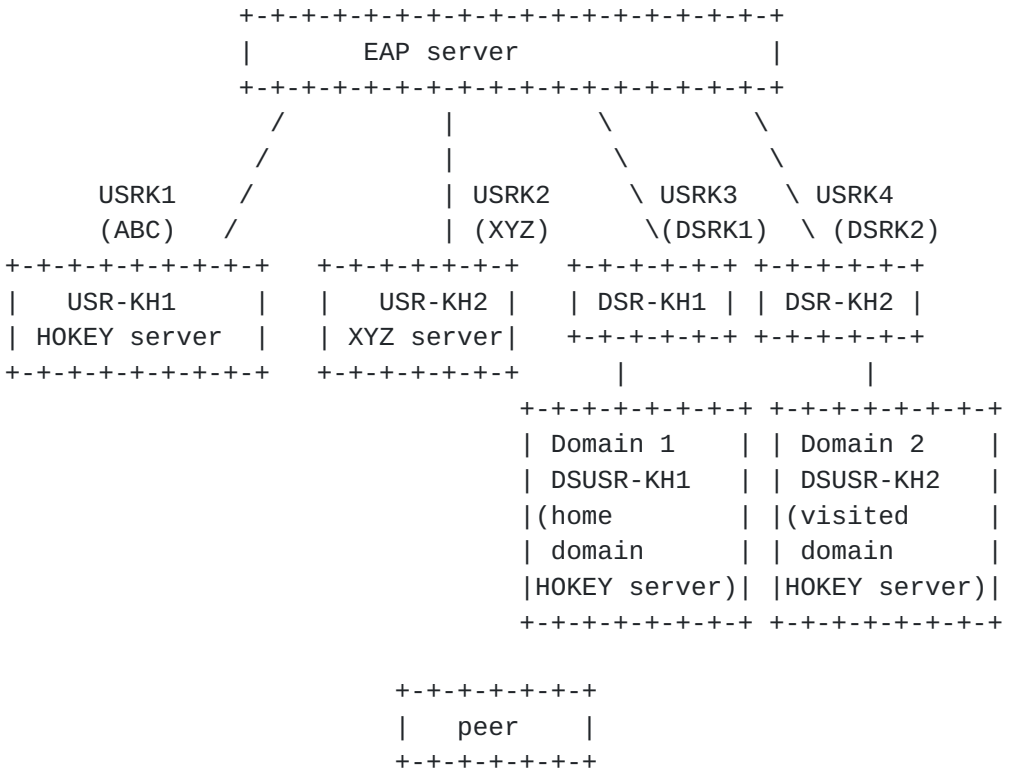


Figure 1: Key delivery for various EMSK child key cateories

USR-KHs and DSR-KHs are the top-level key holders where each key holder has an interface with the EAP server. The interface between a USR-KH and the EAP server is used for delivering usage-specific data needed for generating a USRK to the EAP server and receiving the resulting root key from the EAP server. Similary, the interface between a DSR-KH and the EAP server is used for delivering domain-specific data needed for generating a DSRK to the EAP server and receiving the resulting root key from the EAP server. DSUSR-KHs are considered a second-level key holder and has an interface with a DSR-KH.

4. Key Distribution Exchange (KDE)

In the following we describe the generic mechanism for a key distribution exchange (KDE), where a key is distributed from a network server (with parent key holder) to a third-party. The following shows a generic trust model for the key distribution mechanism to the third-party. The peer (P) and a parent key holder, called "server" (S) in this model share a parent key. The goal of the keying solution is to use the parent key and generate a child key (Kpt) to be shared between the peer and the third-party intermediary (T). The peer is able to generate Kpt, but Kpt needs to be distributed to a third-party intermediary (T). The goal of this section is to provide a the general description of the KDE over RADIUS for distribution of Kpt from S to T. It is required that the communication path between the server (S) and the third-party (T) is protected by use of an appropriate RADIUS transport security mechanism (see Section 7).

Figure 2: Distribution of a child key from a parent key key holder to a 3rd party child key key holder

The key distribution to a third-party consists of 1 exchange, i.e. 2 messages between the third-party and the server. A RADIUS KDE attribute is introduced for this exchange (see Section 5. A KDE-Request is sent by the third-party as a RADIUS Access-Request message with a KDE attribute with K-flag cleared. A KDE-Response is sent by the server as a RADIUS Access-Accept message with a KDE attribute with K-flag set.

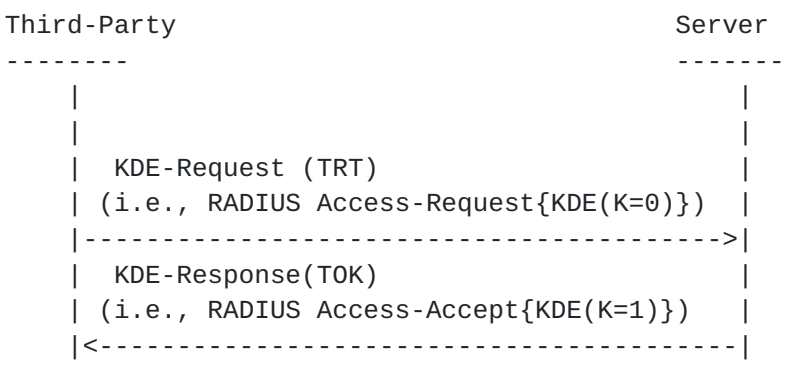


Figure 3: KDE Exchange

KDE-Request: Third-party sends a third party request token (TRT) to the server. The contents of TRT are detailed below.

KDE-Response: Server sends the Kpt to third-party wrapped inside a token called Key Token (TOK).

TRT : (PID, KT, KL)

TRT (Third-party Request Token) carries the identifiers of the peer (PID) as well as the key type (KT) and the key label (KL). See [RFC5295] for the specification of key labels.

TOK : (KT, KL, Kpt, KN_Kpt, LT_Kpt)

TOK (Key Token) carries the key to be distributed to the third-party (Kpt). LT_Kpt is the key lifetime for a Kpt.

4.1. Context and scope for distributed keys

The key lifetime of each distributed key MUST NOT be greater than that of its parent key.

The key context of each distributed key is determined by the sequence of KTs in the key hierarchy. When a DSRK is being delivered and the DSRK applies to only a specific set of services, the service types may need to be carried as part of context for the key. Carrying such a specific set of services are outside the scope of this document.

The key scope of each distributed key is determined by the sequence of (PID, KT, KL)-tuples in the key hierarchy. The Key Derivation Function (KDF) used to generate the keys includes context and scope information that binds the key to the specific channel [RFC5295].

4.2. Key distribution exchange scenarios

There are three scenarios for distribution of EMSK child keys third-parties. We assume that the peer and the EAP server have mutually authenticated to each other using an EAP method and have generated an EMSK. For all scenarios, the trigger and mechanism for key delivery may involve a specific request from the peer and another intermediary (such as authenticator). For simplicity, we assume that USR-KHs reside in the same domain as the EAP server.

Scenario 1: EAP server to USR-KH: In this scenario, an EAP server delivers a USRK to a USR-KH.

Scenario 2: EAP server to DSR-KH: In this scenario, an EAP server delivers a DSRK to a DSR-KH.

Scenario 3: DSR-KH to DSUSR-KH: In this scenario, a DSR-KH in a specific domain delivers keying material to the DSUSR-KH in the same domain.

The key distribution exchanges for Scenario 3 can be combined with the key distribution exchanges for Scenario 2 into a single roundtrip exchange as shown in Figure 4. KDE-Request and KDE-Response are messages for Scenarios 2. KDE-Request' and KDE-Response' are messages for Scenarios 3.

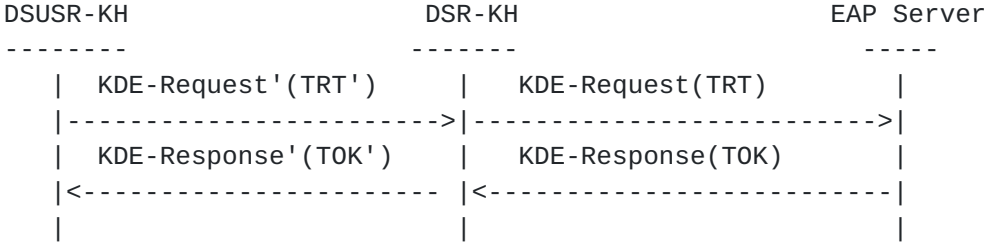
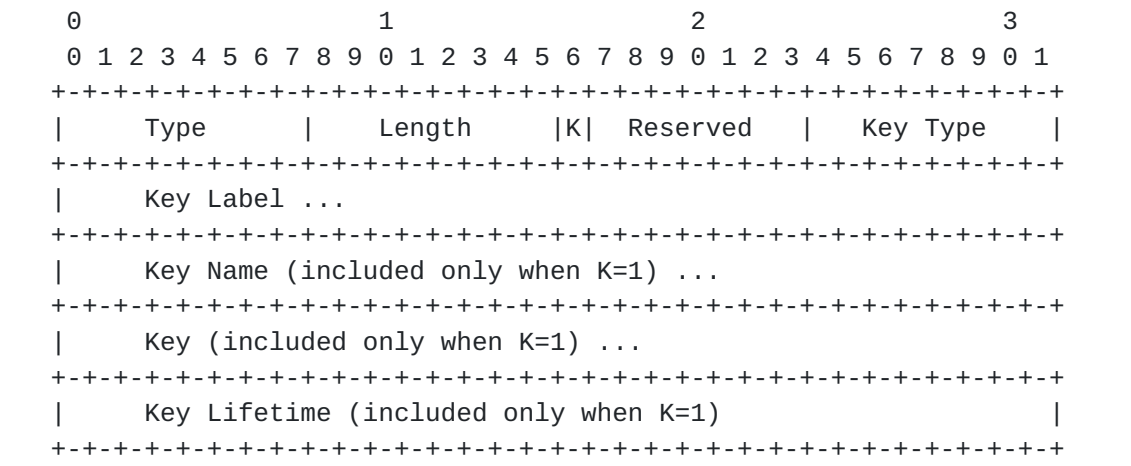


Figure 4: Combined Message Exchange

5. RADIUS KDE Attribute

A RADIUS Key Distribution Exchange (KDE) attribute has the following format. See Section 7 for security requirements on transporting this RADIUS attribute.



Type

= X (KDE) [X to be assigned by IANA].

Length

>4

K (Key included)

A flag to indicate whether this attribute contains a Key field. This flag is set for a KDE-Response. This flag is cleared for a KDE-Request.

Reserved

Reserved bits. All reserved bits MUST be set to 0 by the sender and ignored by the recipient.

Key Type

A field to contain a KT. The following KT values are defined: 0 (DSRK), 1 (USRK) and 2 (DSUSRK).

Key Label

A field to contain a key label (KL). The first octet contains the length of the rest of this field in octets.

Key Name

A field to contain a KN_Kpt. The first octet contains the length of the rest of this field in octets. This field is contained if

and only if K-flag is set.

Key

A field to contain a Kpt. The first octet contains the length of the rest of this field in octets. This field is contained if and only if K-flag is set.

Key Lifetime

A 4-octet unsigned integer to indicate a LT_Kpt. This field is contained if and only if K-flag is set.

A KDE-Request, i.e., an RADIUS Access-Request message carrying a KDE attribute with K-flag cleared, is transmitted by the third-party in one of the following cases. One case is for explicit ERP bootstrapping [RFC5296] in which an EAP-Initiate and EAP-Finish exchange is performed between the peer and the home AAA server, with the bootstrapping flag in the EAP-Initiate message set. In this case, the third-party that requests a Kpt MUST include a KDE attribute with K-bit cleared in a RADIUS Access-Request message that carries an EAP-Initiate message with the bootstrapping flag turned on [RFC5296]. Another case is implicit ERP bootstrapping [RFC5296] in which Kpt key distribution occurs during initial EAP authentication. In this case, the third-party that requests a Kpt MUST include a KDE attribute with K-flag cleared in the RADIUS Access-Request message that carries the first EAP-Response message from the peer. In both cases, a value of the RADIUS User-Name attribute is used as the PID.

A KDE-Response, i.e., an RADIUS Access-Accept message carrying a KDE attribute with K-flag set, is transmitted by a server in one of the following ways. In the case of explicit ERP bootstrapping, the server MUST include a KDE attribute with K-flag set in a RADIUS Access-Accept message that carries an EAP-Finish message [RFC5296] for which the bootstrapping flag is set. In the case of implicit ERP bootstrapping, the server that received a valid KDE-Request includes a KDE attribute with K-flag set in a RADIUS Access-Accept message that carries an EAP-Success.

6. Conflicting Messages

In addition to the rules specified in Section 2.6.3. of [RFC3579], the following combinations SHOULD NOT be sent by a RADIUS Server:

Access-Accept/EAP-Message/EAP-Finish with 'R' flag set to 1

Access-Reject/EAP-Message/EAP-Finish with 'R' flag set to 0

Access-Reject/Keying-Material

Access-Reject/KDE

Access-Challenge/EAP-Message/EAP-Initiate

Access-Challenge/EAP-Message/EAP-Finish

Access-Challenge/KDE

7. Security Considerations

This section provides security requirements and an analysis on transporting EAP keying material using RADIUS.

7.1. Requirements on RADIUS Key Transport

RADIUS messages that carry a KDE attribute MUST be encrypted and integrity and replay protected with a security association created by a RADIUS transport protocol such as TLS [I-D.ietf-radext-radsec]. When there is an intermediary such as a RADIUS proxy on the path between the third-party and the server, there will be a series of hop-by-hop security associations along the path. The use of hop-by-hop security associations implies that the intermediary on each hop can access the distributed keying material. Hence the use of hop-by-hop security SHOULD be limited to an environment where an intermediary is trusted not to use the distributed key material.

7.2. Distributing Kpt without Peer Consent

When a KDE-Request message is sent as a result of explicit ERP bootstrapping [RFC5296], cryptographic verification of peer consent on distributing a Kpt is provided by the integrity checksum of the EAP-Initiate message with the bootstrapping flag turned on.

When a KDE-Request message is sent as a result of implicit ERP bootstrapping [RFC5296], cryptographic verification of peer consent on distributing a Kpt is not provided. As a result, it is possible for a third-party to request a Kpt from the server and obtain the Kpt even if a peer actually does not support ERP, which can lead to an unintended use of a Kpt.

8. IANA consideration

This document defines a new namespace for maintaining Key Type used to identify the type of Kpt. The range of values 0 - 255 are for permanent, standard message types, allocated by IETF Review [IANA]. This document defines the values 0 (DSRK) 1 (USRK) and 2 (DSUSRK).

This document defines a new RADIUS Attribute Type for KDE defined in Section 5.

9. Acknowledgements

The author would like to thank Dan Harkins, Chunqiang Li, Rafael Marin Lopez and Charles Clancy for their valuable comments.

10. Contributors

The following people contributed to this document.

Madjid Nakhjiri (madjid.nakhjiri@motorola.com)

Yoshihiro Ohba (yohba@tari.toshiba.com)

Kedar Gaonkar (kgaonkar3@gatech.edu)

Lakshminath Dondeti (ldondeti@qualcomm.com)

Vidya Narayanan (vidyan@qualcomm.com)

Glen Zorn (glenzorn@comcast.net)

11. References

11.1. Normative References

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

[RFC2548] Zorn, G., "Microsoft Vendor-specific RADIUS Attributes", RFC 2548, March 1999.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC3579] Aboba, B. and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, September 2003.

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

[RFC5295] Salowey, J., Dondeti, L., Narayanan, V., and M. Nakhjiri, "Specification for the Derivation of Root Keys from an Extended Master Session Key (EMSK)", RFC 5295, August 2008.

[RFC5296] Narayanan, V. and L. Dondeti, "EAP Extensions for EAP Re-authentication Protocol (ERP)", RFC 5296, August 2008.

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

11.2. Informative references

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

[RFC5169] Clancy, T., Nakhjiri, M., Narayanan, V., and L. Dondeti, "Handover Key Management and Re-Authentication Problem Statement", RFC 5169, March 2008.

[I-D.ietf-radext-radsec] Winter, S., McCauley, M., Venaas, S., and K. Wierenga, "TLS encryption for RADIUS over TCP (RadSec)", draft-ietf-radext-radsec-01 (work in progress), August 2008.

Author's Address

Yoshihiro Ohba
Toshiba America Research, Inc.
1 Telcordia Drive
Piscataway, NJ 08854
USA

Phone: +1 732 699 5305
Email: yohba@tari.toshiba.com

Full Copyright Statement

Intellectual Property