Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 6, 2010

K. Hoeper, Ed.
M. Nakhjiri
Motorola
Y. Ohba, Ed.
Toshiba

December 3, 2009

Distribution of EAP based keys for handover and re-authentication draft-ietf-hokey-key-mgm-13

Abstract

This document describes an abstract mechanism for delivering root keys from an Extensible Authentication Protocol (EAP) server to another network server that requires the keys for offering security protected services, such as re-authentication, to an EAP peer. The distributed root key can be either a usage-specific root key (USRK), a domain-specific root key (DSRK) or a domain-specific usage-specific root key (DSUSRK) that has been derived from an Extended Master Session Key (EMSK) hierarchy previously established between the EAP server and an EAP peer. The document defines a template for a key distribution exchange (KDE) protocol that can distribute these different types of root keys using an AAA (Authentication, Authorization and Accounting) protocol and discusses its security requirements. The described protocol template does not specify message formats, data encoding, or other implementation details. It thus needs to be instantiated with a specific protocol (e.g. RADIUS or Diameter) before it can be used.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on June 6, 2010.

Copyright Notice

Table of Contents

1. Introduction

The Extensible Authentication Protocol (EAP) [RFC3748] is an authentication framework supporting authentication methods that are specified in EAP methods. By definition, any key-generating EAP method derives a Master Session Key (MSK) and an Extended Master Session Key (EMSK). [RFC5295] reserves the EMSK for the sole purpose of deriving root keys that can be used for specific purposes called usages. In particular, [RFC5295] defines how to create a usage-specific root key (USRK) for bootstrapping security in a specific application, a domain-specific root key (DSRK) for bootstrapping security of a set of services within a domain, and a usage-specific DSRK (DSUSRK) for a specific application within a domain. [RFC5296] defines a re-authentication root key (rRK) that is a USRK designated for re-authentication.

The MSK and EMSK may be used to derive further keying material for a variety of security mechanisms [RFC5247]. For example, the MSK has been widely used for bootstrapping the wireless link security associations between the peer and the network attachment points. However, performance as well as security issues arise when using the MSK and the current bootstrapping methods in mobile scenarios that require handovers, as described in [RFC5169]. To address handover latencies and other shortcomings, [RFC5296] specifies an EAP re-authentication protocol (ERP) that uses keys derived from the EMSK or DSRK to enable efficient re-authentications in handover scenarios. [RFC5295] and [RFC5296] both do not specify how root keys are delivered to the network server requiring the key. Such a key delivery mechanism is essential because the EMSK cannot leave the EAP server ([RFC5295]) but root keys are needed by other network servers disjoint with the EAP server. For example, in order to enable an EAP peer to re-authenticate to a network during a handover, certain root keys need to be made available by the EAP server to the server carrying out the re-authentication.

This document specifies an abstract mechanism for the delivery of the EMSK child keys from the server holding the EMSK or a root key to another network server that requests a root key for providing protected services (such as re-authentication and other usage and domain-specific services) to EAP peers. In the remainder of this document, a server delivering root keys is referred to as Key Delivering Server (KDS) and a server authorized to request and receive root keys from a KDS is referred to as Key Requesting Server (KRS). The Key Distribution Exchange (KDE) mechanism defined in this document runs over an AAA (Authentication, Authorization and Accounting) protocol, e.g., RADIUS [RFC2865], [RFC3579] or Diameter [RFC3588], and has several variants depending on the type of key that is requested and delivered (i.e., DRSK, USRK, and DSUSRK). The

presented KDE mechanism is a protocol template that must be instantiated for a particular protocol, such as RADIUS or Diameter, to specify the format and encoding of the abstract protocol messages. Only after such an instantiation can the KDE mechanism described in this document be implemented. The document also describes security requirements for the secure key delivery over AAA.

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].

AAA
: Authentication, Authorization and Accounting. AAA protocols with EAP support include RADIUS [RFC2865], [RFC3579] and Diameter [RFC3588].

USRK
: Usage-Specific Root Key. A root key that is derived from the EMSK, see [RFC5295].

USR-KH
: USRK Holder. A network server that is authorized to request and receive a USRK from the EAP server. The USR-KH can be an AAA server or dedicated service server.

DSRK
: Domain-Specific Root Key. A root key that is derived from the EMSK, see [RFC5295].

DSR-KH
: DSRK Holder. A network server that is authorized to request and receive a DSRK from the EAP server. The most likely implementation of a DSR-KH is an AAA server in a domain, enforcing the policies for the usage of the DSRK within this domain.

DSUSRK
: Domain-Specific Usage-Specific Root Key. A root key that is derived from the DSRK, see [RFC5295].

DSUSR-KH
: DSUSRK holder. A network server authorized to request and receive a DSUSRK from the DSR-KH. The most likely implementation of a DSUSR-KH is an AAA server in a domain, responsible for a particular service offered within this domain.

RK
Root Key. An EMSK child key, i.e., a USRK, DSRK, or DSUSRK.

KDS
Key Delivering Server. A network server that holds an EMSK or DSRK and delivers root keys to KRS requesting root keys. The EAP server together with the AAA server it exports the keys to for delivery and the DSR-KH can both act as KDS.

KRS
Key Requesting Server. A network server that shares an interface with a KDS and is authorized to request root keys from the KDS. USR-KH, DSR-KH, and DSUSR-KH can all act as KRS.

3. Key Delivery Architecture

An EAP server carries out normal EAP authentications with EAP peers but is typically not involved in potential handovers and re-authentication attempts by the same EAP peer. Other servers are typically in place to offer these requested services. These servers can be AAA servers or other service network servers. Whenever EAP-based keying material is used to protect a requested service, the respective keying material has to be available to the server providing the requested service. For example, the first time a peer requests a service from a network server, this server acts as a KRS. The KRS requests the root keys needed to derive the keys for protecting the requested service from the respective KDS. In subsequent requests from the same peer and as long as the root key has not expired, the KRS can use the same root keys to derive fresh keying material to protect the requested service. These kinds of key requests and distributions are necessary because an EMSK cannot leave the EAP server ([RFC5295]). Hence, any root key that is directly derived from an EMSK can only be derived by the EAP server itself. The EAP server then exports these keys to a server that can distribute the keys to the KRS. In the remainder of this document, the KDS consisting of the EAP server that derives the root keys together with the AAA server that distributes these keys is denoted EAP/AAA server. Root keys derived from EMSK child keys, such as a DSUSRK, can be requested from the respective root key holder. Hence, a KDS can be either the EAP/AAA server or a DSRK holder (DSR-KH), whereas a KRS can be either a USRK holder (USR-KH), a DSR-KH or a DSUSRK holder (DSUSR-KH).

The KRS needs to share an interface with the KDS to be able to send all necessary input data to derive the requested key and to receive the requested key. The provided data includes the Key Derivation Function (KDF) that should be used to derive the requested key. The

KRS uses the received root key to derive further keying material in order to secure its offered services. Every KDS is responsible for storing and protecting the received root key as well as the derivation and distribution of any child key derived from the root key. An example of a key delivery architecture is illustrated in Figure 1 showing the different types of KRS and their interfaces to the KDS.

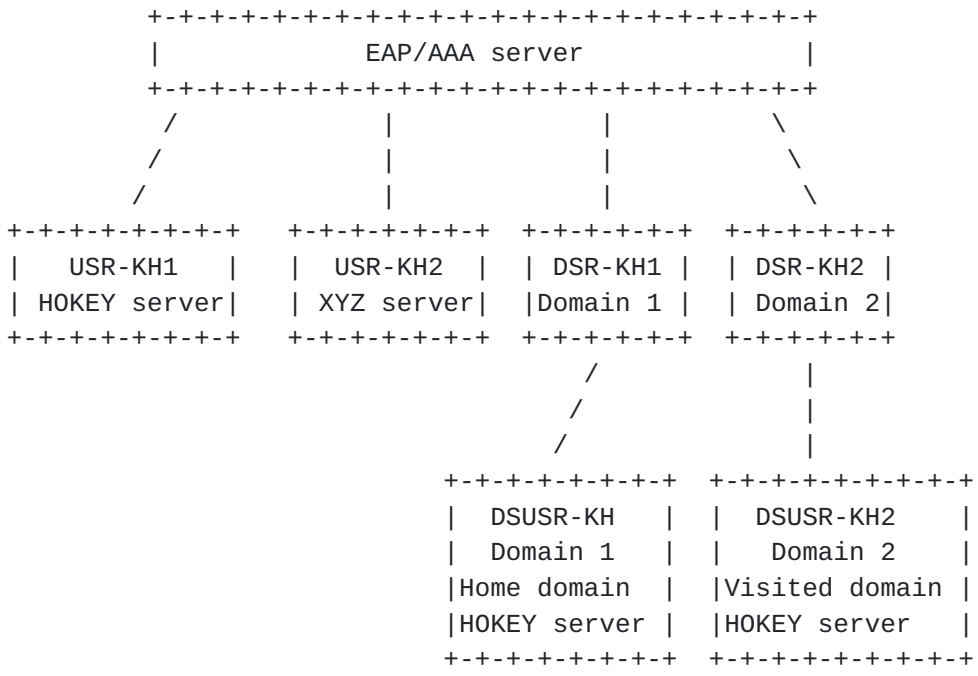

Figure 1: Example Key Delivery Architecture for the Different KRS and KDS

4. Key Distribution Exchange (KDE)

In this section, a generic mechanism for a key distribution exchange (KDE) over AAA is described in which a root key (RK) is distributed from a KDS to a KRS. It is required that the communication path between the KDS and the KRS is protected by the use of an appropriate AAA transport security mechanism (see Section 6 for security requirements). Here, it is assumed that the KRS and the KDS are separate entities, logically if not physically, and the delivery of the requested RK is specified accordingly.

The key distribution exchange consists of one round-trip, i.e., two messages between the KRS and the KDS, as illustrated in Figure 2. First, the KRS sends a KDE-Request carrying a Key Request Token (KRT). As a response, the KDS sends a KDE-Response carrying a Key

Delivery Token (KDT). Both tokens are encapsulated in AAA messages. The definition of the AAA attributes depends on the implemented AAA protocol and is out of scope of this document. However, the security requirements for AAA messages carrying KDE messages are discussed in Section 6. The contents of KRT and KDT are defined in the following.

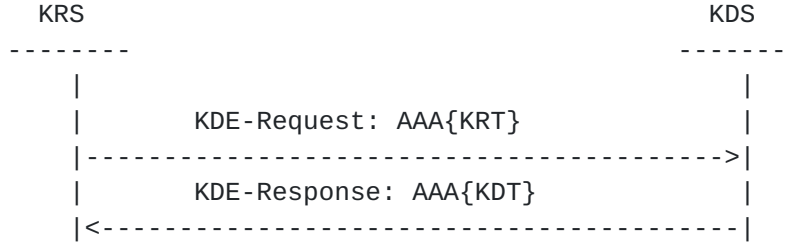

Figure 2: KDE Message Flow

KRT : (PID, KT, KL)

KRT carries the identifiers of the peer (PID), the key type (KT) and the key label (KL). The key type specifies which type of root key is requested, e.g., DSRK, USRK and DSUSRK. The encoding rules for each key type are left to the protocol developers who define the instantiation of the KDE mechanism for a particular protocol. For the specification of key labels and the associated IANA registries, please refer to [RFC5295] which specifies key labels for USRKs and establishes an IANA registry for them. The same specifications can be applied to other root keys.

KDT : (KT, KL, RK, KN_RK, LT_RK)

KDT carries the root key (RK) to be distributed to the KRS, as well as the key type (KT) of the key, the key label (KL), the key name (KN_RK) and the lifetime of RK (LT_RK). The key lifetime of each distributed key MUST NOT be greater than that of its parent key.

4.1. Context and Scope for Distributed Keys

The key context of each distributed key is determined by the sequence of KTs in the key hierarchy. The key scope of each distributed key is determined by the sequence of (PID, KT, KL)-tuples in the key hierarchy and the identifier of the KRS. The KDF used to generate the requested keys includes context and scope information, thus, binding the key to the specific channel [RFC5295].

4.2. Key Distribution Exchange Scenarios

Given the three types of KRS, there are three scenarios for the distribution of the EMSK child keys. For all scenarios, the trigger and mechanism for key delivery may involve a specific request from an EAP peer and/or another intermediary (such as an authenticator). For simplicity, it is assumed that USR-KHs reside in the same domain as the EAP server.

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

Scenario 2: EAP/AAA server to DSR-KH: In this scenario, the EAP/AAA 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 a 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 round-trip exchange as shown in Figure 3. Here, KDE-Request and KDE-Response are messages for Scenarios 2, whereas KDE-Request' and KDE-Response' are messages for Scenarios 3.

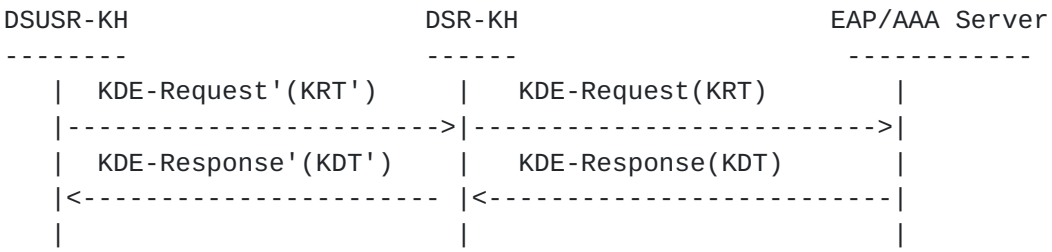

Figure 3: Combined Message Exchange

5. KDE used in the EAP Re-authentication Protocol (ERP)

This section describes how the presented KDE mechanism should be used to request and deliver the root keys used for re-authentication in the EAP Re-authentication Protocol (ERP) defined in [RFC5296]. ERP supports two forms of bootstrapping, implicit as well as explicit bootstrapping, and KDE is discussed for both cases in the remainder of this section.

In implicit bootstrapping the local EAP Re-authentication (ER) server requests the DSRK from the home AAA server during the initial EAP

exchange. Here, the local ER server acts as the KRS and the home AAA server as the KDS. In this case, the local ER server requesting the DSRK includes a KDE-Request in the AAA packet encapsulating the first EAP-Response message from the peer. Here, an AAA User-Name attribute is used as the PID. If the EAP exchange is successful, the home AAA server includes a KDE-Response in the AAA message that carries the EAP-Success message.

Explicit bootstrapping is initiated by peers that do not know the domain. Here, the peer sends an EAP-Initiate message with the bootstrapping flag turned on. The local ER server (acting as KRS) includes a KDE-Request message in the AAA message that carries the peer's EAP-Initiate message and sends it to the peer's home AAA server. Here, an AAA User-Name attribute is used as the PID. In its response, the home AAA server (acting as KDS) includes a KDE-Response in the AAA message that carries the EAP-Finish message with the bootstrapping flag set.

6. Security Considerations

This section provides security requirements and a discussion of distributing RK without peer consent.

6.1. Requirements on AAA Key Transport Protocols

Any KDE attribute that is exchanged as part of a KDE-Request or KDE-Response MUST be integrity-protected and replay-protected by the underlying AAA protocol that is used to encapsulate the attributes. Additionally, a secure key wrap algorithm MUST be used by the AAA protocol to protect the RK in a KDE-Response. Other confidential information as part of the KDE messages (e.g., identifiers if privacy is a requirement) SHOULD be encrypted by the underlying AAA protocol.

When there is an intermediary, such as an AAA proxy, on the path between the KRS and the KDS, 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 abuse the distributed key material. If such a trusted AAA infrastructure does not exist, other means must be applied at a different layer to ensure the end-to-end security (i.e., between KRS and KDS) of the exchanged KDE messages. The security requirements for such a protocol are the same as previously outlined for AAA protocols and MUST hold when encapsulated in AAA messages.

6.2. Distributing RK without Peer Consent

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

On the other hand, when a KDE-Request is sent as a result of implicit ERP bootstrapping [RFC5296], cryptographic verification of peer consent on distributing an RK is not provided. A peer is not involved in the process and, thus, not aware of a key delivery requests for root keys derived from its established EAP keying material. Hence, a peer has no control where keys derived from its established EAP keying material are distributed to. A possible consequence of this is that a KRS may request and obtain an RK from the home server even if the peer does not support ERP. EAP-Initiate/Re-auth-Start messages send to the peer will be silently dropped by the peer causing further waste of resources.

7. IANA consideration

This document contains no IANA considerations.

8. Acknowledgments

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

9. Contributors

The following people contributed to this document: Kedar Gaonkar, Lakshminath Dondeti, Vidya Narayanan, and Glen Zorn.

10. References

10.1. Normative References

[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.

[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.

10.2. Informative references

[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.

[RFC3588] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, September 2003.

[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.

Authors' Addresses

Katrin Hoeper (editor)
Motorola, Inc.
1301 E Algonquin Road
Schaumburg, IL 60196
USA

Phone: +1 847 576 4714
Email: khoeper@motorola.com

Madjid F Nakhjiri
Motorola, Inc.
6450 Sequence Drive
San Diego, CA 92121
USA

Email: madjid.nakhjiri@motorola.com

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

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