Network Working Group
Internet-Draft
Expires: August 3, 2008

M. Nakhjiri
Motorola
Y. Ohba
Toshiba

January 31, 2008

Derivation, delivery and management of EAP based keys for handover and re-authentication
draft-ietf-hokey-key-mgm-02

Status of this Memo

Copyright Notice

Abstract

This document describes a delivery framework for usage and domain specific root keys (USRK and DSRK), derived as part of an EAP EMSK hierarchy, and delivered from an EAP server to an intended third party key holder. The framework description includes different scenarios for key delivery, depending on the type of keys being delivered. It also includes, specification of derivation of keys

required for security protection of key requests and delivery signaling.

Table of Contents

1. Introduction and Problem statement

The ability of Extensible Authentication Protocol (EAP) framework [RFC3748] in incorporating desired authentication methods and generating master session keys (MSK and EMSK) [I-D.ietf-eap-keying] has led to the idea of using MSK and/or EMSK for bootstrapping further keys for a variety of security mechanisms. Especially, 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 MSK and the current bootstrapping methods when it comes to mobility performance and security are described in [I-D.ietf-hokey-reauth-ps]. Thus new efforts are under way to use EMSK instead of MSK for bootstrapping of keys for future use cases [I-D.ietf-hokey-emsk-hierarchy], [I-D.ietf-hokey-erx]. For instance [I-D.ietf-hokey-emsk-hierarchy] defines ways to create usage specific root keys (USRK) for bootstrapping security of a specific use case. [I-D.ietf-hokey-emsk-hierarchy] 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 on the other hand 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. This is to address the following concerns:

1. EAP authentication is a 2 party protocol executed between an EAP peer and an EAP server and the EMSK is only generated and held at these two parties [I-D.ietf-eap-keying], while USRK, DSRK and DSUSRK are also generated only by these two parties, but they typically need to be stored and utilized at third party key holders (e.g. AAA servers/entities) that are logically or even physically separate from the EAP server or peer. For instance, handover keying and re-authentication service requires distribution of keys a variety of intermediaries. This would mean these root keys need to be delivered to these third party key holders (KH) in a secure manner, while considering the requirements stated in [RFC4962]

2. EAP authentication and EMSK generation process is oblivious to the service and authorization requests following the initial EAP authentication. Thus at the time of EAP authentication, the EAP parties do not have access to the input data required for creation of the USRK, DSRK or DSUSRK [I-D.ietf-hokey-emsk-hierarchy]. Such input data is typically acquired and delivered to the EAP server at a later stage. The EAP server then performs the derivation function, followed by a secure delivery of the resulting keys to these third party key holders.

The purpose of this document is to show how the required input data for root key derivation can be delivered to the EAP server, and how the generated key material is delivered to the third party key holder in a secure manner. The specification also includes derivation of key material required for secure delivery and channel binding procedures for these key materials to ensure that not only the keys are not exposed to unintended parties during delivery, but also the scope and usage context for the key is properly understood and agreed upon by the initial parties.

The purpose of this document is not to provide exact syntax for the signaling, only the general semantics for the parameters involved are defined. The exact syntax for these parameters when carried by specific protocols, such as AAA protocols [RFC3579] [RFC4072] or EAP protocols are out of scope of this document.

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 EMSK and is used for a specific usage that is authorized for a peer, following an EAP authentication. USRK is domain independent.

USR-KH: USRK holder. The USR-KH is responsible for receiving, holding and protection of the USRK derived directly from EMSK.

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

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

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

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

HOKEY server (HS): This is essentially a usage specific server, that deals with re-authentication as specific usage (USR-KH for HOKEY).

IK and CK: Integrity and cipher keys, used to protect the key delivery signaling between the peer and the EAP server. These two keys are some times referred to as key delivery keys.

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, neither 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 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 EMSK cannot be exported from EAP server, such third parties need to request the EAP server to generate the relevant root key (USRK, DSRK, or DSUSRK) 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 possibly responsible for derivation and distribution of any child keys derived from USRK for that specific usage. The USR-KH by definition is domain independent, which means the USR-KH can use USRK to generate DSUSRK for each domain deploying that service. It is possible that this USR-KH server is not physically disjunct from the EAP server but is simply considered as a separate logic to off-load the EAP server from the need to handle usage specific services, such as HOKEY service. Security Requirements for delivery of USRK from EAP server to a collocated USR-KH is currently TBD. However, to keep the security specifications generic here, we assume that USR-KH and EAP server are physically separate and specify the delivery of 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 possibly responsible for derivation and distribution of any child keys derived from DSRK for that specific domain. The most likely realization of DSR-KH is a AAA server in the corresponding domain, responsible for setting the policies for usage of DSRK within the domain.

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

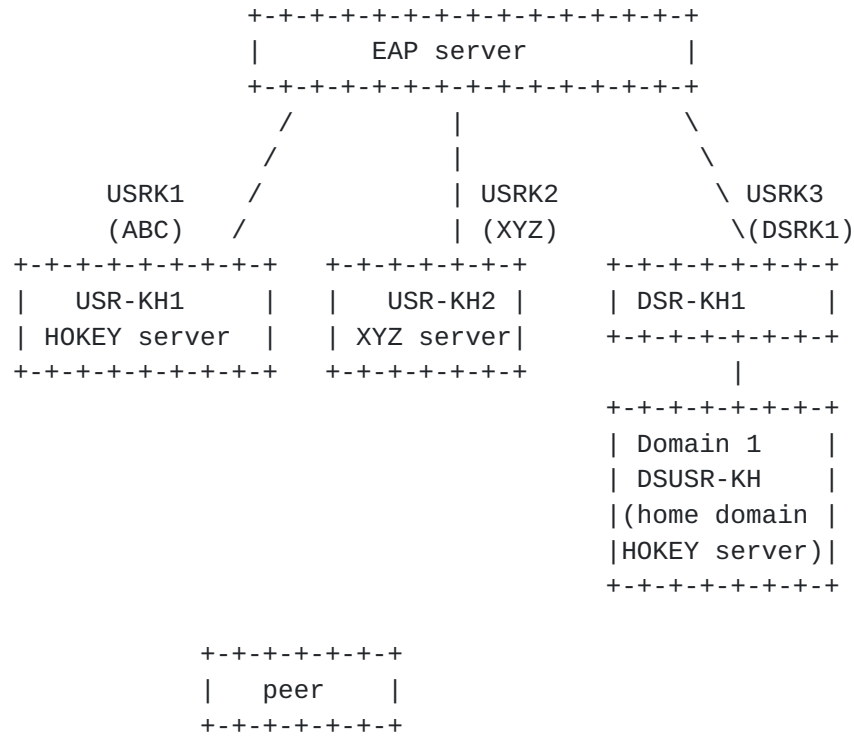

Figure 1: Key delivery for various EMSK child key cateories

As one can see, depending on the type of key being delivered, different third party key holders are involved. Each of the top level key holders (USR-KH, DSR-KH has a interface with the EAP server, for delivering usage specific (and/or domain specific) input data needed for root key generation (USRK, DSRK) to the EAP server and receiving the resulting root key from the EAP server. The

DSUSR-KH is considered a top level key holder and has an interface with EAP server, only when the DSUSRK is directly derived from the EMSK and by the EAP server.

Regardless of the type of key being delivered, the model for EAP based key derivation and delivery interface can be generalized as a 3 party key distribution model, since EAP authentication method signaling and the following EMSK generation is performed between the peer and the EAP server in a manner that is almost transparent to all intermediaries, while the EMSK is used to derive the top level root keys and deliver those to a third party key holder, such as USR-KH or DSR-KH.

3.1. Three Party Key distribution Exchange (KDE)

In the following we describe the generic mechanism for a 3 party 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 3 party key distribution mechanism. The peer (P) and a parent key holder, called "server" (S) in this model share a parent key (Kps) and a set of security associations (SA1) for integrity and privacy protection of signaling between the peer and the server (KIps and KCps). The goal of the keying solution is to use the parent key (Kps) 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 (key distribution exchange) for distribution of Kpt from S to T. We also assume that the server (S) and the third party (T) share a similar set security association, SA2 (KIts, KCts).

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

The key distribution exchange described here is a modified version of the Otway-Rees protocol that meets the requirements for such 3-party

lay-out, providing channel binding and avoids the lying intermediary scenario. The exchange proposed below is to perform a channel binding and avoid the lying intermediary scenario. The description below can be carried over a generic transport and thus is independent of the exact type of protocol that is used. However for the purpose of this document the assumption is that the 3 party mechanism parameters are carried for EAP messages that are themselves encapsulated over a lower layer such as a AAA protocol or an access protocol.

The 3 party key distribution basically consists of 1 exchange, i.e. 2 messages between the peer and the server. However, in most scenarios each message traverses through the intermediary, i.e. Over two logical hops (peer-third party) and (third party-server) even though the exchange seems to consist of 4 logical messages. It should be noted that the information in message 0 is typically conveyed as an advertisement through other means.

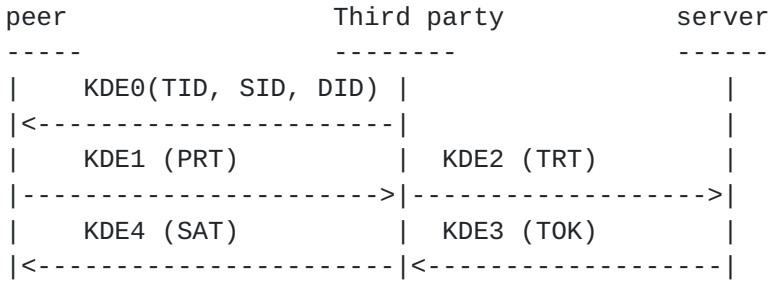

Figure 3: Handover using EAP-HR

KDE message 0: Third party sends its own identifier (TID), the Server ID (SID) and the Domain ID (DID) to peer. These identifiers need to be recognizable by the server S and when AAA signaling is used may be carried as AAA attributes. Domain ID is used to create domain specific keys or to assist in key distribution.

KDE message 1: Peer sends a peer request token (PRT) to the third party including the TID reported by the third party and a freshness value. The contents of PRT are detailed below.

KDE message 2: Third party uses the PRT and creates a third party request token (TRT) and sends it to the server. The contents of TRT are detailed below.

KDE message 3: Server sends the Kpt to third party wrapped inside a token called Key Token (TOK). The TOK carries a Server Authorization Token (SAT) destined for the peer, carrying assurance for the peer that the server has sent the key Kpt to the properly identified third party (identified by TID)

KDE message 4: The third party extracts the SAT from the server and forwards it to the peer. The successful receipt of message 4 by peer means that the third party has successfully verified integrity of message 3 and decrypted Kpt.

{X}K: X encrypted with key K

Int[K, X]: X || MIC (K, X), where MIC Message Integrity Code over X with key K.

PRT : Int[KIps,(PID, TID, SID, DID, FVp, KT, KN_KIps)]

PRT (Peer Request Token) carries the identities of peer (PID), server (SID), third party (TID) and domain (DID) along with the signature. The signature is called the peer request authenticator (PRA). KIps is a symmetric key shared between peer and Server for signing and identified by KN_KIps. FVx is the Freshness Value provided by the party X. A Freshness Value can be a nonce or a time stamp. Use of nonce for FV may require additional mechanism for the server to maintain the freshness. KT is the Key Type requested by the peer. The KT is an 2-octet unsigned integer which uniquely identifies the type of the key.

TRT : Int[KIts, (PID, TID), PRT]

TRT (Third party Request Token) carries the token from the peer along with the third party and peer IDs and a signature for integrity protection. KIts is the shared key used for signing purposes. Providing third party identifier both explicitly by the third party and both implicitly through PRT allows the server to detect a lying third party.

TOK : Int[KIts, {Kpt}KCts, SAT]

TOK(Key Token) carries the key to be distributed to the third party (Kpt) wrapped with an encryption key (KCts). KL_Kx is the key lifetime for key Kx.

SAT : Int[KIps,(PID, TID, SID, DID, FVp+1, KN_Kpt, KL_Kpt, KN_KIps)]

SAT (Server Authorization Token) carries assurance (in form of signature on the incremented nonce value) for the peer that the

server has sent the key Kpt to the properly identified third party (identified by TID).

The exchange proposed above can avoid the lying intermediary scenario, as follows: if an intermediary decided to announce two different identifiers to the peer versus to the server, e.g. a down link ID to the peer (DTID) and a different uplink ID to the server (UTID). The peer uses DTID in its token towards the server, while the intermediary uses its UTID in its token to the server. Server must use the UTID from peer token to calculate the MIC in the third party token ([PID, UTID]KIts) and if there is a match, then the server can verify that DTID and UTID are the same as the TID and proceed with generating and provisioning of Kpt, otherwise the server MUST return a failure code instead of generating an Kpt.

3.2. Derivation of keys protecting the KDE

As shown in the generic description of the key distribution exchange, to protect the exchange, at least one (or two) keys are required to protect the exchange. These keys are an integrity and a cipher key. These keys are generated from the EMSK hierarchy themselves. However, as discussed when enumerating the various KDE use case scenarios, the KDE can and need to be used in many different scenarios for delivering keys. Depending on the key that is being delivered, the integrity and cipher keys can be generated at different levels of the key hierarchy as well. For instance to protect the KDE performed to deliver an EMSK child key (USRK, DSRK, or DSUSRK), these two keys are generated directly from EMSK.

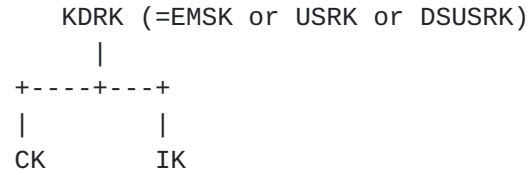

Figure 4: Key delivery keys as EMSK Child keys

Cipher key (CK) and Integrity Key (IK) are used to protect KDE for delivery of USRKs, DSRKs, DSUSRKs, per-authenticator master keys and any other keys generated from EMSK. CK and IK are generated from KDRK (Key Distribution Root Key) which is EMSK or DSRK. When KDRK is EMSK, CK and IK are defined as USRKs. When KDRK is DSRK, CK and IK are defined as DSUSRKs.

We refer to PRF used to generate the USRKs, USRK-PRF, and assume that it can generate Z bits of output.

CK and IK, and their key names are defined using the algorithms

defined in [I-D.ietf-hokey-emsk-hierarchy] as follows:

IK = KDF(KDRK, IK_Label | NULL | Key_length)

IK_name = prf-64(Name_Base, IK_Label)

CK = KDF(KDRK, CK_Label | NULL | Key_length)

CK_name = prf-64(Name_Base, CK_Label)

The IK_Label and CK_Label are IANA-assigned ASCII strings "Key Distribution Integrity Key" and "Key Distribution Cipher Key" assigned from the Key Label name space in accordance with [I-D.ietf-hokey-emsk-hierarchy]. This document specifies IANA registration for the IK_Label and CK_Label above.

When KDRK is EMSK, Name_Base is EAP Session ID.

When KDRK is a USRK or a DSUSRK, Name_Base is the name of the USRK or DSUSRK, respectively.

3.3. Specification of 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.

The key scope of each distributed key is determined by the sequence of (PID, SID, TID, DID)-tuples in the key hierarchy.

3.4. Automated Key Management for KIts and KCts

KIts and KCts require automated key management [RFC4107] since these are long-term session keys used by more than two parties. It is RECOMMENDED that Kerberos [RFC4120] be used as an automated key management protocol for distributing KIts and KCts. If there is no direct trust relationship between the third-party and the server, then inter-realm Kerberos SHOULD be used to create a direct trust relationship between the third-party and the server from a chain of trust relationships.

3.5. 3 party key exchange scenarios

As mentioned earlier, EMSK can be used to generate any of the USRKs, DSRKs and DSUSRKs. The following scenarios can be envisioned for distribution of a key to a 3rd party. All scenarios assume the peer and the EAP server have mutually authenticated to each other using an EAP method and have generated an EMSK. Since the EAP server performing EAP method authentication and EMSK generation resides in peer's home domain, for practical purposes, for the mechanisms described in this document, the USR-KH MUST reside in this domain. Note that other key distribution scenarios may also be possible since the key distribution protocol is designed to be generic.

Scenario 1: EAP server to USR-KH: The specific use case considered in this document is HOKEY re-authentication service: The USR-KH is a HOKEY server and USRK is rRK. The EAP server delivers the rRK to the USR-KH. The trigger and mechanism for key delivery may involve a specific request from the peer and another intermediary (such as authenticator).

Scenario 2: EAP server to DSR-KH: In this scenario, a domain server, typically a domain AAA server requires delivery of a DSRK for a set of prespecified usages within the domain. DSR-KH and EAP server are physically disjunct. Thus deployment of 3 party key distribution exchange is required.

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 mapping between the protocol parameters in each scenario to the protocol parameters of the KDE protocol defined in Section 3.1 is given below, where IK_X and CK_X are IK and CK derived from key X, respectively.

	Scenarios		
KDE Param.	1	2	3
PID	EAP Peer ID		
SID	EAP Server ID		DSR-KH ID
TID	HS ID(*1)	DSR-KH ID	HS ID(*2)
Kpt	rRK	DSRK	rRK
KIps	IK_EMSK		IK_DSRK
KCps	CK_EMSK		CK_DSRK
KIts	Any pre-existing key		
KCts	Any pre-existing key		

(*1) HS ID is USR-KH ID for HOKEY
(*2) HS ID is DSUSR-KH ID for HOKEY

The key distribution exchanges for some of the above scenarios can be recursively combined into a single 1.5-roundtrip exchange. For example, a combined key distribution exchange for Scenarios 3 and 6 is illustrated in Figure 5 where KDE[0-4] and KDE'[0-4] are messages for Scenarios 3 and 6, respectively. It is assumed in Figure 5 that DSR-KH and DSUSR-KH are co-located in the same HOKEY server (i.e., DSR-KH and DSUSR-KH have the same identity).

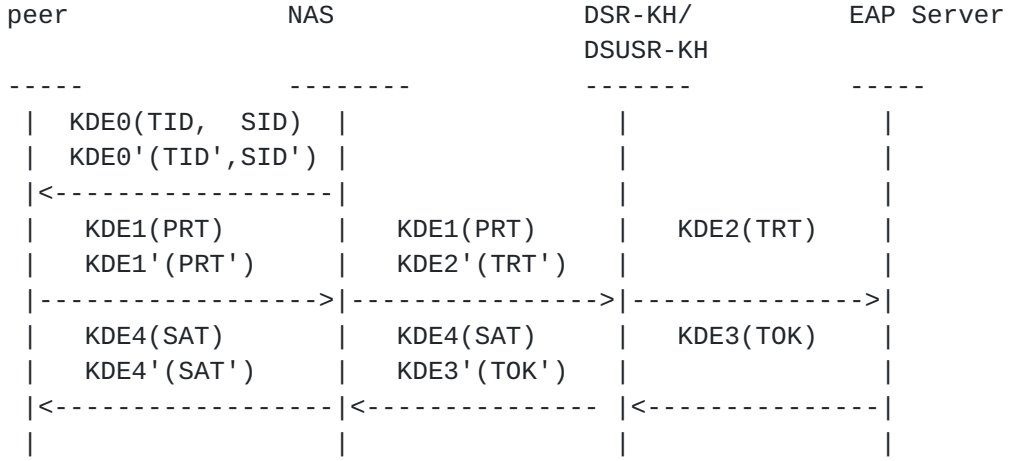

Figure 5: Combined Message Exchange

4. Security Considerations

The key distribution mechanism described in this document assumes existence of a direct trust relationship between the server and the third party key holder. Such a direct trust relationship may be dynamically created from a chain of transitive trust relationships with the use of inter-realm Kerberos to distribute KIts and KCts as described in Section 3.4. Therefore, the key distribution method described in this document eliminates the need for hop-by-hop security associations along the transitive trust relationship.

5. IANA consideration

This document defines new usage labels, such as those used in generation of CK and IK. The corresponding labels (e.g. "Key Distribution Integrity Key" and "Key Distribution Cipher key") need to be assigned numerical values by IANA.

This document defines KT (Key Type) which is an IANA-assigned 2-octet unsigned integer. The following Key Type values are allocated in this document:

0 (DSRK)

1 (rRK)

6. Acknowledgements

The author would like to thank Dan Harkins, Chunqiang Li and Rafael Marin Lopez for their valuable contribution to the formation of the KDE.

7. References

7.1. 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.

[I-D.ietf-hokey-emsk-hierarchy]
Salowey, J., Dondeti, L., Narayanan, V., and M. Nakhjiri, "Specification for the Derivation of Root Keys from an Extended Master Session Key (EMSK)", draft-ietf-hokey-emsk-hierarchy-03 (work in progress), January 2008.

[I-D.ietf-hokey-erx]
Narayanan, V. and L. Dondeti, "EAP Extensions for EAP Re-authentication Protocol (ERP)", draft-ietf-hokey-erx-08 (work in progress), November 2007.

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

[RFC4072] Eronen, P., Hiller, T., and G. Zorn, "Diameter Extensible Authentication Protocol (EAP) Application", RFC 4072, August 2005.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.

[RFC4962] Housley, R. and B. Aboba, "Guidance for Authentication, Authorization, and Accounting (AAA) Key Management", BCP 132, RFC 4962, July 2007.

7.2. Informative references

[RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, June 2005.

[I-D.ietf-eap-keying]
Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", draft-ietf-eap-keying-22 (work in progress), November 2007.

[I-D.ietf-hokey-reauth-ps]
Clancy, C., Nakhjiri, M., Narayanan, V., and L. Dondeti, "Handover Key Management and Re-authentication Problem Statement", draft-ietf-hokey-reauth-ps-07 (work in progress), November 2007.

Authors' Addresses

Madjid Nakhjiri
Motorola

Email: madjid.nakhjiri@motorola.com

Yoshihiro Ohba
Toshiba

Email: yohba@tari.toshiba.com

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).