EAP Working Group
Internet-Draft
Expires: December 3, 2005

Y. Ohba
Toshiba
M. Yanagiya
NTT
Jun 2005

AAA-Key Derivation with Lower-Layer Parameter Binding
draft-ohba-eap-aaakey-binding-01

Status of this Memo

Copyright Notice

Abstract

This document describes an alternative method for deriving a AAA-Key. The method cryptographically binds EAP lower-layer parameters to the AAA-Key without need to carry those parameters in EAP methods.

Table of Contents

1. Introduction

EAP (Extensible Authentication Protocol) is an authentication framework which supports multiple authentication algorithms known as "EAP methods" [RFC3748]. EAP lower- layers use a key generated and exported by an EAP method to bootstrap their ciphersuites. This key is referred to as AAA-Key. A framework for the generation, transport and usage of AAA-Key is described in [I-D.ietf-eap-keying].

Each EAP lower-layer has its own parameters that are carried at the lower-layer. EAP lower-layer end-point identifiers are one of such parameters. Those parameters would need to be cryptographically bound to the AAA-Key to avoid possible security flaws.

A mechanism that is described in [RFC3748] to create such a binding is based on communicating lower-layer parameters over a protected channel of an EAP method to help the EAP peer and the EAP server detect a mismatch between the parameters exchanged over the protected channel and the ones advertised at an unprotected lower-layer. There have been several solutions [I-D.arkko-eap-service-identity-auth] [I-D.tschofenig-eap-ikev2] that are based on this mechanism.

This document describes an alternate mechanism for creating a binding between a AAA-Key and EAP lower-layer parameters without need for an EAP method to carry the EAP lower-layer parameters.

1.1 Specification of Requirements

In this document, several words are used to signify the requirements of the specification. These words are often capitalized. 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].

2. Problem Description

When an authentication and authorization procedure for a network access service succeeds using EAP with an EAP authentication method that is capable of generating a Master Session Key (MSK), a AAA-Key is derived from the MSK and transferred from the EAP server and the EAP authenticator [I-D.ietf-eap-keying] for use by the EAP lower layer to further derive keys used for protecting the lower layer. However, if the AAA-Key is not bound to EAP lower layer parameters, the key may be used for a different context of the EAP lower layer or even for a different EAP lower layer. Detailed examples on this problem is described in section 2 of [I-D.arkko-eap-service-identity-auth].

The problem has been originally known as channel binding problem [RFC3748] [I-D.ietf-eap-keying]. However, the solution that is suggested in those documents is to use EAP methods that can carry EAP lower layer parameters and expecting the EAP server and EAP peer to validate them by comparing with the EAP lower layer parameters carried via the EAP lower layer protocol between the EAP peer and EAP authenticator or a AAA protocol between the EAP authenticator or the EAP server. The solution, however, requires each EAP method to carry EAP lower layer parameters, which is not actually needed for a standalone EAP authenticator while binding the EAP lower layer parameters to the AAA-Key is still somehow needed for both standalone and pass-through EAP authenticators. This leads to a need for a more generic key binding mechanism that does not require any change in existing EAP methods and can work with both standalone and pass-through EAP authenticators in the same way.

3. Prerequisites

There are following prerequisites for the solution proposed in this document to work.

3.1 Trust Relationship

In this document, trust relationship is characterized with a "trust level" to cover a wide range of scenarios including roaming scenarios. A full level of trust relationship between two or more entities is defined where each entity trusts whatever other entities claim. There are following prerequisites with regard to trust relationship between an EAP peer, an EAP authenticator and an EAP server involving in an EAP conversation.

- There is a full level of trust relationship between the EAP peer and the EAP server.

- There is a trust relationship between the EAP server and the EAP authenticator. The trust level between them may vary depending on the deployment. For example, if they are implemented on the same physical box, it is highly likely that there is a full level of trust relationship between them. On the other hand, if they are implemented on different physical boxes belonging to different service providers, there may not be a full level of trust relationship between them, even if there is some sort of roaming agreement between the service providers. In this case, the EAP server does not always trust what the EAP authenticator claims.

- There is a trust relationship between the EAP peer and the EAP authenticator, but most likely the trust relationship is not a full level. The level of the trust relationship is high enough for the EAP peer to use the AAA-Key in the service provided by the EAP lower-layer.

3.2 Securty Association

In general, a security association is a relationship established between two or more entities to enable them to protect data they exchange [RFC2828]. A security association needs to be established between such entities unless the underlying communication path is already protected by some other mechanism. The following security associations needs to be established prior to execute EAP.

- A security association between the EAP peer and the EAP server in order to establish an MSK (Master Session Key) [I-D.ietf-eap-keying].

- A security association between the EAP authenticator and the EAP server in case an untrusted third party may interfere the communication between them.

4. Solution Framework

In the proposed solution, it is the EAP lower-layer entities that make final decision as to whether the the lower-layer parameters are successfully bound to the AAA-Key, regardless of the location of the EAP server. To this end, the EAP server that has a trust relationship with both the EAP peer and the EAP authenticator needs to involve in the process of binding the EAP lower-layer parameters to the AAA-Key. The proposed solution, however, does not require EAP and EAP methods to carry EAP lower-layer parameters. The proposed solution is designed such that the AAA-Key derivation algorithm is agnostic to specific EAP lower-layer parameters.

4.1 Key Binding Blob for Lower-Layer Parameters

Each EAP lower-layer must define a blob that is an octet-string used for carrying lower-layer parameters that need to be bound to the AAA-Key. Such a blob is referred to as a "key-binding-blob". A key-binding-blob contains parameters that persist long-term such as the identification of the EAP authenticator. It is the responsibility of each EAP lower-layer to define how the EAP lower-layer parameters are encoded in the blob.

When the EAP authenticator and the EAP server are implemented in different physical boxes, a key-binding-blob is carried in a AAA protocol as described in Section 6.

4.2 AAA-Key Derivation Algorithm

As a result of successful authentication, the EAP peer and EAP server derives a AAA-Key from the MSK [I-D.ietf-eap-keying] exported by the EAP method as follows.

AAA-Key = KDF(MSK, AAA-Key-name|key-binding-blob)

KDF is a key derivation function. The definition of KDF is TBD. AAA-Key-name is the name of the AAA-Key. The format of the name of AAA-Key is described in Section 4.4

4.3 AAA-Key Scope

The scope of a AAA-Key is between the pair of a particular EAP peer and a particular EAP authenticator. The AAA-Key MUST NOT be shared among multiple EAP authenticators or multiple EAP peers.

4.4 AAA-Key Name

TBD.

4.5 Key Binding Procedure

During an EAP authentication run, the EAP authenticator constructs a key-binding-blob from the EAP lower-layer parameters and sends the key-binding-blob to the EAP server. When the EAP authenticator is acting as a pass-through authenticator, a AAA protocol would be used for communicating the key-binding-blob to the EAP server.

Depending on the level of the trust relationship between the EAP authenticator and EAP server, the EAP server MAY validate the key-binding-blob as described in Section 5 and the authentication procedure MUST fail when the validation fails.

Upon successful EAP authentication, the EAP peer and the EAP server are expected to compute the AAA-Key using the above algorithm. The computed AAA-Key is delivered from the EAP server to the EAP authenticator.

Finally, the EAP peer and the EAP authenticator verify the possession of the AAA-Key via a secure association protocol to establish a secure association. For the verification process to succeed, it is required for the EAP authenticator to have obtained the same AAA-Key from the EAP server as the EAP peer has. This actually requires the EAP authenticator to have sent the same key-binding-blob to the EAP server as the one the EAP peer constructs from the lower-layer parameters obtained via the lower-layer protocol.

5. Validating Key Binding Blob

As described in Section 3.1, the level of trust relationship between an EAP authenticator and an EAP server may vary depending on the deployment. When there is a full level of trust relationship between the EAP authenticator and EAP server, the EAP server can trust information sent by the EAP authenticator and thus it is not necessary for the EAP server to verify the EAP lower-layer parameters encoded in the key-binding-blob.

On the other hand, when there is not a full level of trust relationship between the EAP authenticator and EAP server, the EAP server may not fully trust information sent by the EAP authenticator and thus it would need to verify the EAP lower-layer parameters encoded in the key-binding-blob. The verification of the key-binding-blob can be performed based on simple string comparison with the expected value that may be pre-configured on the EAP server, meaning that the EAP server would need to know the structure and semantics of the key-binding-blob when the key-binding-blob is pre-configured. However, the EAP server can still be agnostic to the structure and semantics of the key-binding-blob during the execution of the verification procedure.

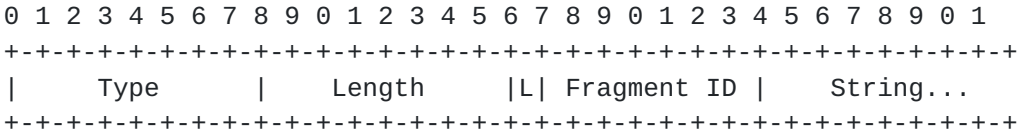
6. AAA Protocol Consideration

When a AAA protocol such as RADIUS [RFC2865] and Diameter [RFC3588] is used for carrying EAP messages between an EAP authenticator and the EAP server, a key-binding-blob is carried in a AAA protocol. The Key-Binding-Blob attribute, which is a new RADIUS attribute, is defined for this purpose. Since Diameter has a backward compatibility with RADIUS, it is possible to automatically convert one or more RADIUS Key-Binding-Blob attribute to a corresponding Diameter AVP and vise versa. The Key-Binding-Blob attribute MUST NOT be modified by an intermediary AAA node such as a AAA proxy.

Considering that the maximum size of a RADIUS attribute value is 253 octets and that the size of the key-binding-blob can exceed 253 octets, the Key-Binding-Blob attribute is defined to allow fragmentation in the following way.

When the size of the key-binding-blob is X octets (X>0) and RADIUS is used for carrying the key-binding-blob, the key-binding-blob is split into (X div 253) octet-strings, where each octet-string is carried in a separate RADIUS Key-Binding-Blob attribute with a distinct fragment identifier (Fragment ID) which starts from zero (0) and incremented by one (1). A Key-Binding-Blob attribute also has a flag (L-flag) to indicate the last fragment. The L-flag MUST be set for a Key-Binding-Blob attribute carrying the last fragment, otherwise it MUST be unset.

When Diameter is used for carrying the key-binding-blob, the entire key-binding-blob is carried in a single Key-Binding-Blob AVP, with the L-flag set and the Fragment ID set to zero (0).



Type: TBD

Represents Key-Binding-Blob.

Length: >=4

L(ast flagment):

Indicates whether this is the last fragment. If there is only one fragment, the L-flag MUST be set.

Fragment ID:

Fragment identifier used for identifying a fragment of a key-binding-blob. The identifier starts from zero (0) and incremented by one (1).

String:

Contains a fragment of a key-binding-blob.

7. Co-existence with Legacy AAA-Key Derivation Algorithm

The following consideration is needed in order to interwork with EAP lower layers that use the legacy AAA-Key derivation algorithm (i.e., AAA-Key=MSK(0,63) [I-D.ietf-eap-keying]) instead of the algorithm described in Section 4.2.

An EAP authenticator for an EAP lower layer using the legacy AAA-Key derivation algorithm is allowed to not send a key-binding-blob to the EAP server. If an EAP authenticator does not send a key-binding-blob to the EAP server, and the EAP server is configured to use the legacy AAA-Key derivation algorithm for the EAP authenticator, a AAA-Key is derived based on the algorithm described in [I-D.ietf-eap-keying]. If an EAP authenticator does not send a key-binding-blob to the EAP server and the EAP server is configured to use the AAA-Key derivation algorithm described in Section 4.2 for the EAP authenticator, the authentication procedure MUST fail. If an EAP authenticator sends a key-binding-blob to the EAP server and the EAP server is configured to use the legacy AAA-Key derivation algorithm for the EAP authenticator, the authentication procedure MUST fail.

8. Discussion

The solution described in this document makes EAP methods totally agnostic to EAP lower-layers. EAP methods do not need to carry EAP lower-layer parameters even in the form of a blob.

The solution does not require the the EAP server to know about the structure and the semantics of the key-binding-blob during the execution of the EAP authentication. However, the EAP server may need to identify at least the type of the EAP lower layer in order to avoid the situation where the EAP authenticator sends a key-binding-blob for an EAP lower layer that is different from what is expected by the EAP server, but the content of the key-binding-blob happens to match the expected value. How the EAP lower layer type information is carried from the EAP authenticator to the EAP peer (e.g., carrying it in the key-binding-blob or in the AAA-Key-name or by some other means) is an open issue.

The solution works regardless of whether an EAP authenticator is acting as a pass-through authenticator or not.

Although the solution requires a change in the AAA-Key derivation algorithm described in section 2.3 of [I-D.ietf-eap-keying], the solution can co-exist with the legacy AAA-Key derivation algorithm as described in Section 7.

9. Security Considerations

The solution described in this document improves the security characteristics of the EAP key management framework in that a secure association is never established if there is a difference in EAP lower-layer parameters recognized by the EAP peer and the EAP authenticator. This is in contrast to existing parameter binding methods described in [I-D.arkko-eap-service-identity-auth] [I-D.tschofenig-eap-ikev2] in which an EAP peer can still establish a secure association even when a mismatch in EAP lower-layer parameters is detected by the EAP peer, as the EAP peer can ignore the mismatch and continue the EAP conversation to succeed.

10. Acknowledgments

The authors would like to thank Jari Arkko and Bernard Aboba for discussing this issue on the EAP mailing list and giving insights. The authors would like to thank Rafa Marin Lopez and Julien Bournelle for reviewing the document.

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.

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

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

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

[I-D.ietf-eap-keying] Aboba, B., "Extensible Authentication Protocol (EAP) Key Management Framework", draft-ietf-eap-keying-06 (work in progress), April 2005.

11.2 Informative References

[RFC2828] Shirey, R., "Internet Security Glossary", RFC 2828, May 2000.

[I-D.arkko-eap-service-identity-auth] Arkko, J. and P. Eronen, "Authenticated Service Information for the Extensible Authentication Protocol (EAP)", draft-arkko-eap-service-identity-auth-02 (work in progress), May 2005.

[I-D.tschofenig-eap-ikev2] Tschofenig, H., "EAP IKEv2 Method (EAP-IKEv2)", draft-tschofenig-eap-ikev2-06 (work in progress), May 2005.

Authors' Addresses

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

Phone: +1 732 699 5365
Email: yohba@tari.toshiba.com

Mayumi Yanagiya
NTT Network service systems laboratories, NTT Corporation
9-11, Midori-Cho, 3-Chome
Musashino-Shi, Tokyo 180-8585
Japan

Email: yanagiya.mayumi@lab.ntt.co.jp

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.