Network Working Group
Request for Comments: 4650
Category: Standards Track

M. Euchner

September 2006

HMAC-Authenticated Diffie-Hellman for Multimedia Internet KEYing (MIKEY)

Status of This Memo

Copyright Notice

Abstract

This document describes a lightweight point-to-point key management protocol variant for the multimedia Internet keying (MIKEY) protocol MIKEY, as defined in RFC 3830. In particular, this variant deploys the classic Diffie-Hellman key agreement protocol for key establishment featuring perfect forward secrecy in conjunction with a keyed hash message authentication code for achieving mutual authentication and message integrity of the key management messages exchanged. This protocol addresses the security and performance constraints of multimedia key management in MIKEY.

Table of Contents

1. Introduction

There is work done in IETF to develop key management schemes. For example, IKE [12] is a widely accepted unicast scheme for IPsec, and the MSEC WG is developing other schemes, addressed to group communication [17], [18]. For reasons discussed below, there is, however, a need for a scheme with low latency, suitable for demanding cases such as real-time data over heterogeneous networks and small interactive groups.

As pointed out in MIKEY (see [2]), secure real-time multimedia applications demand a particular adequate lightweight key management scheme that takes care to establish dynamic session keys securely and efficiently in a conversational multimedia scenario.

In general, MIKEY scenarios cover peer-to-peer, simple one-to-many, and small-sized groups. MIKEY in particular describes three key

management schemes for the peer-to-peer case that all finish their task within one roundtrip:

- a symmetric key distribution protocol (MIKEY-PS) based on pre-shared master keys

- a public-key encryption-based key distribution protocol (MIKEY-PK and reverse-mode MIKEY-RSA-R [33]) assuming a public-key infrastructure with RSA-based (Rivest, Shamir and Adleman) private/public keys and digital certificates

- a Diffie-Hellman key agreement protocol (MIKEY-DHSIGN) deploying digital signatures and certificates.

All of these three key management protocols are designed so that they complete their work within just one roundtrip. This requires depending on loosely synchronized clocks and deploying timestamps within the key management protocols.

However, it is known [6] that each of the three key management schemes has its subtle constraints and limitations:

- The symmetric key distribution protocol (MIKEY-PS) is simple to implement; however, it was not intended to scale to support any configurations beyond peer-to-peer, simple one-to-many, and small-size (interactive) groups, due to the need for mutually pre-assigned shared master secrets.

  Moreover, the security provided does not achieve the property of perfect forward secrecy; i.e., compromise of the shared master secret would render past and even future session keys susceptible to compromise.

  Further, the generation of the session key happens just at the initiator. Thus, the responder has to fully trust the initiator to choose a good and secure session secret; the responder is able neither to participate in the key generation nor to influence that process. This is considered a specific limitation in less trusted environments.

- The public-key encryption scheme (MIKEY-PK and MIKEY-RSA-R [33]) depends upon a public-key infrastructure that certifies the private-public keys by issuing and maintaining digital certificates. While such key management schemes provide full scalability in large networked configurations, public-key infrastructures are still not widely available, and, in general, implementations are significantly more complex.

Further, additional roundtrips and computational processing might be necessary for each end system in order to ascertain verification of the digital certificates. For example, typical operations in the context of a public-key infrastructure may involve extra network communication handshakes with the public-key infrastructure and with certification authorities and may typically involve additional processing steps in the end systems. These operations would include validating digital certificates (RFC 3029, [24]), ascertaining the revocation status of digital certificates (RFC 2560, [23]), asserting certificate policies, construction of certification path(s) ([26]), requesting and obtaining necessary certificates (RFC 2511, [25]), and management of certificates for such purposes ([22]). Such steps and tasks all result in further delay of the key agreement or key establishment phase among the end systems, which negatively affects setup time. Any extra PKI handshakes and processing are not in the scope of MIKEY, and since this document only deploys symmetric security mechanisms, aspects of PKI, digital certificates, and related processing are not further covered in this document.

Finally, as in the symmetric case, the responder depends completely upon the initiator's choosing good and secure session keys.

- The third MIKEY-DHSIGN key management protocol deploys the Diffie-Hellman key agreement scheme and authenticates the exchange of the Diffie-Hellman half-keys in each direction by using a digital signature. This approach has the same advantages and deficiencies as described in the previous section in terms of a public-key infrastructure.

  However, the Diffie-Hellman key agreement protocol is known for its subtle security strengths in that it is able to provide full perfect forward secrecy (PFS) and further have to both parties actively involved in session key generation. This special security property (despite the somewhat higher computational costs) makes Diffie-Hellman techniques attractive in practice.

In order to overcome some of the limitations as outlined above, a special need has been recognized for another efficient key agreement protocol variant in MIKEY. This protocol variant aims to provide the capability of perfect forward secrecy as part of a key agreement with low latency without dependency on a public-key infrastructure.

This document describes a fourth lightweight key management scheme for MIKEY that could somehow be seen as a synergetic optimization between the pre-shared key distribution scheme and the Diffie-Hellman key agreement.

The idea of the protocol in this document is to apply the Diffie-Hellman key agreement, but rather than deploy a digital signature for authenticity of the exchanged keying material, it instead uses a keyed-hash for symmetrically pre-assigned shared secrets. This combination of security mechanisms is called the HMAC-authenticated Diffie-Hellman (DH) key agreement for MIKEY (DHHMAC).

The DHHMAC variant closely follows the design and philosophy of MIKEY and reuses MIKEY protocol payload components and MIKEY mechanisms to its maximum benefit and for best compatibility.

Like the MIKEY Diffie-Hellman protocol, DHHMAC does not scale beyond a point-to-point constellation; thus, both MIKEY Diffie-Hellman protocols do not support group-based keying for any group size larger than two entities.

1.1. Definitions

The definitions and notations in this document are aligned with MIKEY; see [2] sections 1.3 - 1.4.

All large integer computations in this document should be understood as being mod p within some fixed group G for some large prime p; see [2] section 3.3. However, the DHHMAC protocol is also applicable generally to other appropriate finite, cyclical groups as well.

It is assumed that a pre-shared key s is known by both entities (initiator and responder). The authentication key auth_key is derived from the pre-shared secret s using the pseudo-random function PRF; see [2] sections 4.1.3 and 4.1.5.

In this text, [X] represents an optional piece of information. Generally throughout the text, X SHOULD be present unless certain circumstances MAY allow X to be optional and not to be present, thereby potentially resulting in weaker security. Likewise, [X, Y] represents an optional compound piece of information where the pieces X and Y either SHOULD both be present or MAY optionally both be absent. {X} denotes zero or more occurrences of X.

1.2. Abbreviations

auth_key	Pre-shared authentication key, PRF-derived from pre-shared key s.
DH	Diffie-Hellman
DHi	Public Diffie-Hellman half key g^(xi) of the Initiator
DHr	Public Diffie-Hellman half key g^(xr) of the Responder
DHHMAC	HMAC-authenticated Diffie-Hellman
DoS	Denial-of-service
G	Diffie-Hellman group
HDR	MIKEY common header payload
HMAC	Keyed Hash Message Authentication Code
HMAC-SHA1	HMAC using SHA1 as hash function (160-bit result)
IDi	Identity of initiator
IDr	Identity of receiver
IKE	Internet Key Exchange
IPsec	Internet Protocol Security
MIKEY	Multimedia Internet KEYing
MIKEY-DHHMAC	MIKEY Diffie-Hellman key management protocol using HMAC
MIKEY-DHSIGN	MIKEY Diffie-Hellman key agreement protocol
MIKEY-PK	MIKEY public-key encryption-based key distribution protocol
MIKEY-PS	MIKEY pre-shared key distribution protocol
p	Diffie-Hellman prime modulus
PKI	Public-key Infrastructure
PRF	MIKEY pseudo-random function (see [2] section 4.1.3)
RSA	Rivest, Shamir, and Adleman
s	Pre-shared key
SDP	Session Description Protocol
SOI	Son-of-IKE, IKEv2
SP	MIKEY Security Policy (Parameter) Payload
T	Timestamp
TEK	Traffic Encryption Key
TGK	MIKEY TEK Generation Key, as the common Diffie-Hellman shared secret
TLS	Transport Layer Security
xi	Secret, (pseudo) random Diffie-Hellman key of the Initiator
xr	Secret, (pseudo) random Diffie-Hellman key of the Responder

1.3. Conventions Used in This Document

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 RFC 2119 [1].

2. Scenario

The HMAC-authenticated Diffie-Hellman key agreement protocol (DHHMAC) for MIKEY addresses the same scenarios and scope as the other three key management schemes in MIKEY address.

DHHMAC is applicable in a peer-to-peer group where no access to a public-key infrastructure can be assumed to be available. Rather, pre- shared master secrets are assumed to be available among the entities in such an environment.

In a pair-wise group, it is assumed that each client will be setting up a session key for its outgoing links with its peer using the DH-MAC key agreement protocol.

As is the case for the other three MIKEY key management protocols, DHHMAC assumes, at least, loosely synchronized clocks among the entities in the small group.

To synchronize the clocks in a secure manner, some operational or procedural means are recommended. MIKEY-DHHMAC does not define any secure time synchronization measures; however, sections 5.4 and 9.3 of [2] provide implementation guidance on clock synchronization and timestamps.

2.1. Applicability

MIKEY-DHHMAC and the other MIKEY key management protocols are intended for application-level key management and are optimized for multimedia applications with real-time session setup and session management constraints.

As the MIKEY-DHHMAC key management protocol terminates in one roundtrip, DHHMAC is applicable for integration into two-way handshake session or call signaling protocols such as

a) SIP [13] and SDP, where the encoded MIKEY messages are encapsulated and transported in SDP containers of the SDP offer/answer see RFC 3264 [27]) handshake, as described in [4]; and

b) H.323 (see [15]), where the encoded MIKEY messages are transported in the H.225.0 fast start call signaling handshake. Appendix A outlines the usage of MIKEY-DHHMAC within H.235.

MIKEY-DHHMAC is offered as an option to the other MIKEY key management variants (MIKEY-pre-shared, MIKEY-public-key and MIKEY-DH-SIGN) for all those cases where DHHMAC has its particular strengths (see section 5).

2.2. Relation to GKMARCH

The Group key management architecture (GKMARCH) [19] describes a generic architecture for multicast security group key management protocols. In the context of this architecture, MIKEY-DHHMAC may operate as a registration protocol; see also [2] section 2.4. The main entities involved in the architecture are a group controller/key server (GCKS), the receiver(s), and the sender(s). Due to the pair-wise nature of the Diffie-Hellman operation and the 1-roundtrip constraint, usage of MIKEY-DHHMAC rules out any deployment as a group key management protocol with more than two group entities. Only the degenerate case with two peers is possible where, for example, the responder acts as the group controller.

Note that MIKEY does not provide re-keying in the GKMARCH sense, only updating of the keys by normal unicast messages.

3. DHHMAC Security Protocol

The following figure defines the security protocol for DHHMAC:

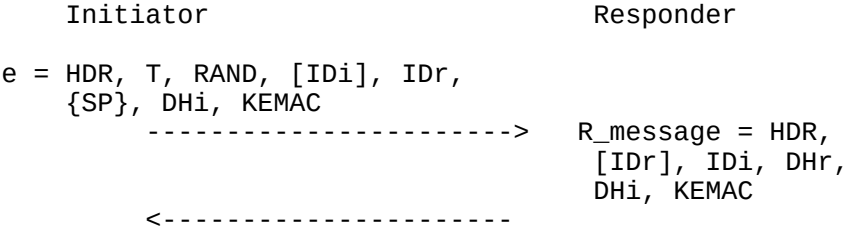

Figure 1: HMAC-authenticated Diffie-Hellman key-based exchange, where xi and xr are (pseudo) randomly chosen, respectively, by the initiator and the responder.

The DHHMAC key exchange SHALL be done according to Figure 1. The initiator chooses a (pseudo) random value, xi, and sends an HMACed message including g^(xi) and a timestamp to the responder. It is recommended that the initiator SHOULD always include the identity

payloads IDi and IDr within the I_message; unless the receiver can defer the initiator's identity by some other means, IDi MAY optionally be omitted. The initiator SHALL always include the recipient's identity.

The group parameters (e.g., the group G) are a set of parameters chosen by the initiator. Note that like in the MIKEY protocol, both sender and receiver explicitly transmit the Diffie-Hellman group G within the Diffie-Hellman payload DHi or DHr through an encoding (e.g., OAKLEY group numbering; see [2] section 6.4). The actual group parameters g and p, however, are not explicitly transmitted but can be deduced from the Diffie-Hellman group G. The responder chooses a (pseudo) random positive integer, xr, and sends an HMACed message including g^(xr) and the timestamp to the initiator. The responder SHALL always include the initiator's identity IDi regardless of whether the I_message conveyed any IDi. It is RECOMMENDED that the responder SHOULD always include the identity payload IDr within the R_message; unless the initiator can defer the responder's identity by some other means, IDr MAY optionally be left out.

Both parties then calculate the TGK as g^(xi * xr).

The HMAC authentication provides authentication of the DH half-keys and is necessary to avoid man-in-the-middle attacks.

This approach is less expensive than digitally signed Diffie-Hellman in that both sides compute one exponentiation and one HMAC first, then one HMAC verification, and finally another Diffie-Hellman exponentiation.

With off-line pre-computation, the initial Diffie-Hellman half-key MAY be computed before the key management transaction and thereby MAY further reduce the overall roundtrip delay, as well as the risk of denial-of-service attacks.

Processing of the TGK SHALL be accomplished as described in MIKEY [2] section 4.

The computed HMAC result SHALL be conveyed in the KEMAC payload field where the MAC fields holds the HMAC result. The HMAC SHALL be computed over the entire message, excluding the MAC field using auth_key; see also section 4.2.

3.1. TGK Re-keying

TGK re-keying for DHHMAC generally proceeds as described in [2] section 4.5. Specifically, Figure 2 provides the message exchange for the DHHMAC update message.

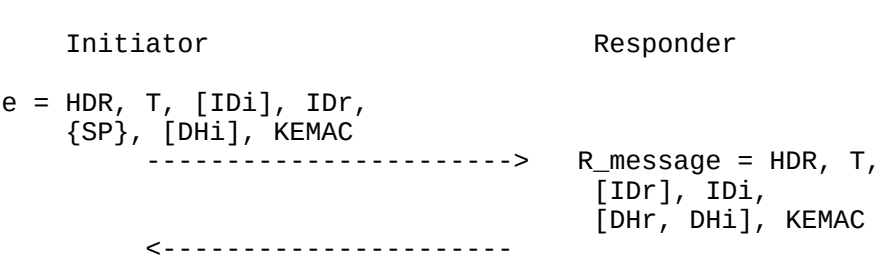

Figure 2: DHHMAC update message

TGK re-keying supports two procedures:

a) True re-keying by exchanging new and fresh Diffie-Hellman half-keys. For this, the initiator SHALL provide a new, fresh DHi, and the responder SHALL respond with a new, fresh DHr and the received DHi.

b) Non-key related information update without including any Diffie-Hellman half-keys in the exchange. Such a transaction does not change the actual TGK but updates other information such as security policy parameters. To update the non-key related information only, [DHi] and [DHr, DHi] SHALL be left out.

4. DHHMAC Payload Formats

This section specifies the payload formats and data type values for DHHMAC; see also [2] section 6, for a definition of the MIKEY payloads.

This document does not define new payload formats but re-uses MIKEY payloads for DHHMAC as referenced:

* Common header payload (HDR); see section 4.1 and [2] section 6.1.

* SRTP ID sub-payload; see [2] section 6.1.1.

* Key data transport payload (KEMAC); see section 4.2 and [2] section 6.2.

* DH data payload; see [2] section 6.4.

* Timestamp payload; see [2] section 6.6.

* ID payload; [2] section 6.7.

* Security Policy payload (SP); see [2] section 6.10.

* RAND payload (RAND); see [2] section 6.11.

* Error payload (ERR); see [2] section 6.12.

* General Extension Payload; see [2] section 6.15.

4.1. Common Header Payload (HDR)

Referring to [2] section 6.1, the following data types SHALL be used for DHHMAC:

Data type	Value	Comment
DHHMAC init	7	Initiator's DHHMAC exchange message
DHHMAC resp	8	Responder's DHHMAC exchange message
Error	6	Error message; see [2] section 6.12

Table 4.1.a

Note: A responder is able to recognize the MIKEY DHHMAC protocol by evaluating the data type field as 7 or 8. This is how the responder can differentiate between MIKEY and MIKEY DHHMAC.

The next payload field SHALL be one of the following values:

Next payload	Value	Section
Last payload	0	-
KEMAC	1	section 4.2 and [2] section 6.2
DH	3	[2] section 6.4
T	5	[2] section 6.6
ID	6	[2] section 6.7
SP	10	[2] section 6.10
RAND	11	[2] section 6.11
ERR	12	[2] section 6.12
General Ext.	21	[2] section 6.15

Table 4.1.b

Other defined next payload values defined in [2] SHALL not be applied to DHHMAC.

In case of a decoding error or of a failed HMAC authentication verification, the responder SHALL apply the Error payload data type.

4.2. Key Data Transport Payload (KEMAC)

DHHMAC SHALL apply this payload for conveying the HMAC result along with the indicated authentication algorithm. When used in conjunction with DHHMAC, KEMAC SHALL not convey any encrypted data; thus, Encr alg SHALL be set to 2 (NULL), Encr data len SHALL be set to 0, and Encr data SHALL be left empty. The AES key wrap method (see [16]) SHALL not be applied for DHHMAC.

For DHHMAC, this key data transport payload SHALL be the last payload in the message. Note that the Next payload field SHALL be set to Last payload. The HMAC is then calculated over the entire MIKEY message, excluding the MAC field using auth_key as described in [2] section 5.2, and then stored within the MAC field.

MAC alg	Value	Comments
HMAC-SHA-1	0	Mandatory, Default (see [3])
NULL	1	Very restricted use; see [2] section 4.2.4

Table 4.2.a

HMAC-SHA-1 is the default hash function that MUST be implemented as part of the DHHMAC. The length of the HMAC-SHA-1 result is 160 bits.

4.3. ID Payload (ID)

For DHHMAC, this payload SHALL only hold a non-certificate-based identity.

4.4. General Extension Payload

For DHHMAC, to avoid bidding-down attacks, this payload SHALL list all key management protocol identifiers of a surrounding encapsulation protocol, such as SDP [4]. The General Extension Payload SHALL be integrity protected with the HMAC using the shared secret.

Type	Value	Comments
SDP IDs	1	List of SDP key management IDs (allocated for use in [4]); see also [2] section 6.15.

Table 4.4.a

5. Security Considerations

This document addresses key management security issues throughout. For a comprehensive explanation of MIKEY security considerations, please refer to MIKEY [2] section 9.

In addition, this document addresses security issues according to [7], where the following security considerations apply in particular to this document:

5.1. Security Environment

The DHHMAC security protocol described in this document focuses primarily on communication security; i.e., the security issues concerned with the MIKEY DHHMAC protocol. Nevertheless, some system security issues are also of interest that are not explicitly defined by the DHHMAC protocol, but that should be provided locally in practice.

The system that runs the DHHMAC protocol entity SHALL provide the capability to generate (pseudo) random numbers as input to the Diffie-Hellman operation (see [8]). Furthermore, the system SHALL be capable of storing the generated (pseudo) random data, secret data, keys, and other secret security parameters securely (i.e., confidential and safe from unauthorized tampering).

5.2. Threat Model

The threat model, to which this document adheres, covers the issues of end-to-end security in the Internet generally, without ruling out the possibility that MIKEY DHHMAC can be deployed in a corporate, closed IP environment. This also includes the possibility that MIKEY DHHMAC can be deployed on a hop-by-hop basis with some intermediate trusted "MIKEY DHHMAC proxies" involved.

Since DHHMAC is a key management protocol, the following security threats are of concern:

* Unauthorized interception of plain TGKs: For DHHMAC, this threat does not occur since the TGK is not actually transmitted on the wire (not even in encrypted fashion).

* Eavesdropping of other, transmitted keying information: DHHMAC protocol does not explicitly transmit the TGK at all. Instead, by using the Diffie-Hellman "encryption" operation, which conceals the secret (pseudo) random values, only partial information (i.e., the DH half-key) for construction of the TGK is transmitted. It is fundamentally assumed that availability of such Diffie-Hellman

half-keys to an eavesdropper does not result in any substantial security risk; see 5.4. Furthermore, the DHHMAC carries other data such as timestamps, (pseudo) random values, identification information or security policy parameters; eavesdropping of any such data is not considered to yield any significant security risk.

* Masquerade of either entity: This security threat must be avoided, and if a masquerade attack would be attempted, appropriate detection means must be in place. DHHMAC addresses this threat by providing mutual peer entity authentication.

* Man-in-the-middle attacks: Such attacks threaten the security of exchanged, non-authenticated messages. Man-in-the-middle attacks usually come with masquerade and or loss of message integrity (see below). Man-in-the-middle attacks must be avoided and, if present or attempted, must be detected appropriately. DHHMAC addresses this threat by providing mutual peer entity authentication and message integrity.

* Loss of integrity: This security threat relates to unauthorized replay, deletion, insertion, and manipulation of messages. Although any such attacks cannot be avoided, they must at least be detected. DHHMAC addresses this threat by providing message integrity.

* Bidding-down attacks: When multiple key management protocols, each of a distinct security level, are offered (such as those made possible by SDP [4]), avoiding bidding-down attacks is of concern. DHHMAC addresses this threat by reusing the MIKEY General Extension Payload mechanism, where all key management protocol identifiers are to be listed within the MIKEY General Extension Payload.

Some potential threats are not within the scope of this threat model:

* Passive and off-line cryptanalysis of the Diffie-Hellman algorithm: Under certain reasonable assumptions (see 5.4, below), it is widely believed that DHHMAC is sufficiently secure and that such attacks are infeasible, although the possibility of a successful attack cannot be ruled out.

* Non-repudiation of the receipt or of the origin of the message: These are not requirements within the context of DHHMAC in this environment, and thus related countermeasures are not provided at all.

* Denial-of-service or distributed denial-of-service attacks: Some considerations are given on some of those attacks, but DHHMAC does not claim to provide full countermeasure against any of those attacks. For example, stressing the availability of the entities is not thwarted by means of the key management protocol; some other local countermeasures should be applied. Further, some DoS attacks are not countered, such as interception of a valid DH- request and its massive instant duplication. Such attacks might at least be countered partially by some local means that are outside the scope of this document.

* Identity protection: Like MIKEY, identity protection is not a major design requirement for MIKEY-DHHMAC, either; see [2]. No security protocol is known so far that is able to provide the objectives of DHHMAC as stated in section 5.3, including identity protection within just a single roundtrip. MIKEY-DHHMAC trades identity protection for better security for the keying material and shorter roundtrip time. Thus, MIKEY-DHHMAC does not provide identity protection on its own but may inherit such property from a security protocol underneath that actually features identity protection.

  The DHHMAC security protocol (see section 3) and the TGK re-keying security protocol (see section 3.1) provide the option not to supply identity information. This option is only applicable if some other means are available to supply trustworthy identity information; e.g., by relying on secured links underneath MIKEY that supply trustworthy identity information some other way. However, it is understood that without identity information, the MIKEY key management security protocols might be subject to security weaknesses such as masquerade, impersonation, and reflection attacks, particularly in end-to-end scenarios where no other secure means of assured identity information are provided.

  Leaving identity fields optional (if doing so is possible) thus should not be seen as a privacy method, either, but rather as a protocol optimization feature.

5.3. Security Features and Properties

With the security threats in mind, this document provides the following security features and yields the following properties:

* Secure key agreement with the establishment of a TGK at both peers: This is achieved using an authenticated Diffie-Hellman key management protocol.

* Peer-entity authentication (mutual): This authentication corroborates that the host/user is authentic in that possession of a pre-assigned secret key is proven using keyed HMAC. Authentication occurs on the request and on the response message; thus authentication is mutual.

  The HMAC computation corroborates for authentication and message integrity of the exchanged Diffie-Hellman half-keys and associated messages. The authentication is absolutely necessary in order to avoid man-in-the-middle attacks on the exchanged messages in transit and, in particular, on the otherwise non-authenticated exchanged Diffie-Hellman half-keys.

  Note: This document does not address issues regarding authorization; this feature is not provided explicitly. However, DHHMAC authentication means support and facilitate realization of authorization means (local issue).

* Cryptographic integrity check: The cryptographic integrity check is achieved using a message digest (keyed HMAC). It includes the exchanged Diffie-Hellman half-keys but covers the other parts of the exchanged message as well. Both mutual peer entity authentication and message integrity provide effective countermeasures against man-in-the-middle attacks.

  The initiator may deploy a local timer that fires when the awaited response message did not arrive in a timely manner. This is intended to detect deletion of entire messages.

* Replay protection of the messages is achieved using embedded timestamps: In order to detect replayed messages, it is essential that the clocks among initiator and sender be roughly synchronized. The reader is referred to [2] section 5.4, and [2] section 9.3, which provide further considerations and give guidance on clock synchronization and timestamp usage. Should the clock synchronization be lost, end systems cannot detect replayed messages anymore, and the end systems cannot securely establish keying material. This may result in a denial-of-service; see [2] section 9.5.

* Limited DoS protection: Rapid checking of the message digest allows verifying the authenticity and integrity of a message before launching CPU intensive Diffie-Hellman operations or starting other resource consuming tasks. This protects against some denial-of-service attacks: malicious modification of messages and spam attacks with (replayed or masqueraded) messages. DHHMAC probably does not explicitly counter sophisticated distributed, large-scale denial-of-service attacks that compromise system availability, for

example. Some DoS protection is provided by inclusion of the initiator's identity payload in the I_message. This allows the recipient to filter out those (replayed) I_messages that are not targeted for him and to avoid creating unnecessary MIKEY sessions.

* Perfect-forward secrecy (PFS): Other than the MIKEY pre-shared and public-key-based key distribution protocols, the Diffie-Hellman key agreement protocol features a security property called perfect forward secrecy. That is, even if the long-term pre-shared key is compromised at some point in time, this does not compromise past or future session keys.

  Neither the MIKEY pre-shared nor the MIKEY public-key protocol variants are able to provide the security property of perfect-forward secrecy. Thus, none of the other MIKEY protocols is able to substitute the Diffie-Hellman PFS property.

  As such, DHHMAC and digitally signed DH provide a far superior security level to that of the pre-shared or public-key-based key distribution protocol in that respect.

* Fair, mutual key contribution: The Diffie-Hellman key management protocol is not a strict key distribution protocol per se, in which the initiator distributes a key to its peers. Actually, both parties involved in the protocol exchange are able to contribute to the common Diffie-Hellman TEK traffic generating key equally. This reduces the risk of either party cheating or unintentionally generating a weak session key. This makes the DHHMAC a fair key agreement protocol. One may view this property as an additional distributed security measure that increases security robustness over that of the case where all the security depends just on the proper implementation of a single entity.

  For Diffie-Hellman key agreement to be secure, each party SHALL generate its xi or xr values using a strong, unpredictable pseudo-random generator if a source of true randomness is not available. Further, these values xi or xr SHALL be kept private. It is RECOMMENDED that these secret values be destroyed once the common Diffie-Hellman shared secret key has been established.

* Efficiency and performance: Like the MIKEY-public key protocol, the MIKEY DHHMAC key agreement protocol securely establishes a TGK within just one roundtrip. Other existing key management techniques, such as IPsec-IKE [12], IPsec-IKEv2 [14], TLS [11], and other schemes, are not deemed adequate in addressing those real-time and security requirements sufficiently; they all use more than a single roundtrip. All the MIKEY key management protocols are able to complete their task of security policy parameter

negotiation, including key-agreement or key distribution, in one roundtrip. However, the MIKEY pre-shared and MIKEY public-key protocol are both able to complete their task even in a half-roundtrip when the confirmation messages are omitted.

Using HMAC in conjunction with a strong one-way hash function (such as SHA1) may be achieved more efficiently in software than expensive public-key operations. This yields a particular performance benefit of DHHMAC over signed DH or the public-key encryption protocol.

If a very high security level is desired for long-term secrecy of the negotiated Diffie-Hellman shared secret, longer hash values may be deployed, such as SHA256, SHA384, or SHA512 provide, possibly in conjunction with stronger Diffie-Hellman groups. This is left as for further study.

For the sake of improved performance and reduced roundtrip delay, either party may pre-compute its public Diffie-Hellman half-key off-line.

On the other side and under reasonable conditions, DHHMAC consumes more CPU cycles than the MIKEY pre-shared key distribution protocol. The same might hold true quite likely for the MIKEY public-key distribution protocol (depending on choice of the private and public key lengths). As such, it can be said that DHHMAC provides sound performance when compared with the other MIKEY protocol variants.

The use of optional identity information (with the constraints stated in section 5.2) and optional Diffie-Hellman half-key fields provides a means to increase performance and shorten the consumed network bandwidth.

* Security infrastructure: This document describes the HMAC-authenticated Diffie-Hellman key agreement protocol, which completely avoids digital signatures and the associated public-key infrastructure, as would be necessary for the X.509 RSA public-key-based key distribution protocol or the digitally signed Diffie-Hellman key agreement protocol as described in MIKEY. Public-key infrastructures may not always be available in certain environments, nor may they be deemed adequate for real-time multimedia applications when additional steps are taken for certificate validation and certificate revocation methods with additional roundtrips into account.

DHHMAC does not depend on PKI, nor do implementations require PKI standards. Thus, it is believed to be much simpler than the more complex PKI facilities.

DHHMAC is particularly attractive in those environments where provisioning of a pre-shared key has already been accomplished.

* NAT-friendliness: DHHMAC is able to operate smoothly through firewall/NAT devices as long as the protected identity information of the end entity is not an IP/transport address.

* Scalability: Like the MIKEY signed Diffie-Hellman protocol, DHHMAC does not scale to any larger configurations beyond peer-to-peer groups.

5.4. Assumptions

This document states a couple of assumptions upon which the security of DHHMAC significantly depends. The following conditions are assumed:

* The parameters xi, xr, s, and auth_key are to be kept secret.

* The pre-shared key s has sufficient entropy and cannot be effectively guessed.

* The pseudo-random function (PRF) is secure, yields the pseudo-random property, and maintains the entropy.

* A sufficiently large and secure Diffie-Hellman group is applied.

* The Diffie-Hellman assumption holds saying basically that even with knowledge of the exchanged Diffie-Hellman half-keys and knowledge of the Diffie-Hellman group, it is infeasible to compute the TGK or to derive the secret parameters xi or xr. The latter is also called the discrete logarithm assumption. Please see [6], [9], or [10] for more background information regarding the Diffie-Hellman problem and its computational complexity assumptions.

* The hash function (SHA1) is secure; i.e., it is computationally infeasible to find a message that corresponds to a given message digest, or to find two different messages that produce the same message digest.

* The HMAC algorithm is secure and does not leak the auth_key. In particular, the security depends on the message authentication property of the compression function of the hash function H when it is applied to single blocks (see [5]).

* A source capable of producing sufficiently many bits of (pseudo) randomness is available.

* The system upon which DHHMAC runs is sufficiently secure.

5.5. Residual Risk

Although these detailed assumptions are non-negligible, security experts generally believe that all these assumptions are reasonable and that the assumptions made can be fulfilled in practice with little or no expenses.

The mathematical and cryptographic assumptions of the properties of the PRF, the Diffie-Hellman algorithm (discrete log-assumption), the HMAC algorithm, and the SHA1 algorithms have been neither proven nor disproven at this time.

Thus, a certain residual risk remains, which might threaten the overall security at some unforeseeable time in the future.

The DHHMAC would be compromised as soon as any of the listed assumptions no longer hold.

The Diffie-Hellman mechanism is a generic security technique that is not only applicable to groups of prime order or of characteristic two. This is because of the fundamental mathematical assumption that the discrete logarithm problem is also a very hard one in general groups. This enables Diffie-Hellman to be deployed also for GF(p)*, for sub-groups of sufficient size, and for groups upon elliptic curves. RSA does not allow such generalization, as the core mathematical problem is a different one (large integer factorization).

RSA asymmetric keys tend to become increasingly lengthy (1536 bits and more) and thus very computationally intensive. Nevertheless, Elliptic Curve Diffie-Hellman (ECDH) allows key lengths to be cut down substantially (say 170 bits or more) while maintaining at least the security level and providing even more significant performance benefits in practice. Moreover, it is believed that elliptic-curve techniques provide much better protection against side channel attacks due to the inherent redundancy in the projective coordinates. For all these reasons, one may view elliptic-curve-based Diffie-Hellman as being more "future-proof" and robust against potential threats than RSA is. Note that Elliptic Curve Diffie-Hellman variants of MIKEY are defined in [31].

HMAC-SHA1 is a key security mechanism within DHHMAC on which the overall security of MIKEY DHHMAC depends. MIKEY DHHMAC uses HMAC-SHA1 in combination with the classic Diffie-Hellman key agreement scheme. HMAC-SHA1 is a keyed one-way hash function that involves a secret in its computation. DHHMAC applies HMAC-SHA1 for protection of the MIKEY payload. Likewise, the pseudo-random function PRF within MIKEY [2] uses the HMAC-SHA1 mechanism as a key derivation function. While certain attacks have been reported against SHA1 and MD5 (see [29]), with current knowledge (see [29], [30]), no attacks have been reported against the HMAC-SHA1 security mechanism. In fact, [32] proves that HMAC possesses the property of a pseudo-random function PRF assuming solely that the (SHA1) hash function is a pseudo-random function. [32] also provides evidence that HMAC is robust against collision attacks on the underlying hash function. It is believed that MIKEY DHHMAC should be considered secure enough for the time being. Thus, there is no need to change the underlying security mechanism within the MIKEY DHHMAC protocol.

It is not recommended to deploy DHHMAC for any other use than that depicted in section 2. Any misapplication might lead to unknown, undefined properties.

5.6. Authorization and Trust Model

Basically, similar remarks on authorization as those stated in [2] section 4.3.2 hold also for DHHMAC. However, as noted before, this key management protocol does not serve full groups.

One may view the pre-established shared secret as yielding some pre-established trust relationship between the initiator and the responder. This results in a much simpler trust model for DHHMAC than would be the case for some generic group key management protocol and potential group entities without any pre-defined trust relationship. In conjunction with the assumption of a shared key, the common group controller simplifies the communication setup of the entities.

One may view the pre-established trust relationship through the pre-shared secret as some means for pre-granted, implied authorization. This document does not define any particular authorization means but leaves this subject to the application.

6. Acknowledgments

This document incorporates kindly, valuable review feedback from Steffen Fries, Hannes Tschofenig, Fredrick Lindholm, Mary Barnes, and Russell Housley and general feedback by the MSEC WG.

7. IANA Considerations

This document does not define its own new name spaces for DHHMAC, beyond the IANA name spaces that have been assigned for MIKEY; see [2] sections 10 and 10.1 and IANA MIKEY payload name spaces [37].

In order to align Table 4.1.a with Table 6.1.a in [2], IANA is requested to add the following entries to their MIKEY Payload Name Space:

Data Type	Value	Reference
DHHMAC init	7	RFC 4650
DHHMAC resp	8	RFC 4650

8. References

8.1. Normative References

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

[2] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, August 2004.

[3] NIST, FIBS-PUB 180-2, "Secure Hash Standard", April 1995, http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf.

[4] Arkko, J., Lindholm, F., Naslund, M., Norrman, K., and E. Carrara, "Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP)", RFC 4567, July 2006.

[5] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

8.2. Informative References

[6] A.J. Menezes, P. van Oorschot, S. A. Vanstone: "Handbook of Applied Cryptography", CRC Press 1996.

[7] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

[8] Eastlake 3rd, D., Crocker, S., and J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1994.

[9] Ueli M. Maurer, S. Wolf: "The Diffie-Hellman Protocol", Designs, Codes, and Cryptography, Special Issue Public Key Cryptography, Kluwer Academic Publishers, vol. 19, pp. 147-171, 2000. ftp://ftp.inf.ethz.ch/pub/crypto/publications/MauWol00c.ps.

[10] Discrete Logarithms and the Diffie-Hellman Protocol, http://www.crypto.ethz.ch/research/ntc/dldh/.

[11] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[12] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[13] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[14] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[15] ITU-T Recommendation H.235.7: " H.323 Security framework: Usage of the MIKEY Key Management Protocol for the Secure Real Time Transport Protocol (SRTP) within H.235"; 9/2005.

[16] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, September 2002.

[17] Baugher, M., Weis, B., Hardjono, T., and H. Harney, "The Group Domain of Interpretation", RFC 3547, July 2003.

[18] Harney, H., Meth, U., Colegrove, A., and G. Gross, "GSAKMP: Group Secure Association Key Management Protocol", RFC 4535, June 2006.

[19] Baugher, M., Canetti, R., Dondeti, L., and F. Lindholm, "Multicast Security (MSEC) Group Key Management Architecture", RFC 4046, April 2005.

[20] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[21] ITU-T Recommendation H.235.0, " H.323 Security framework: Security framework for H-series (H.323 and other H.245 based) multimedia systems", (09/2005).

[22] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, September 2005.

[23] Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 2560, June 1999.

[24] Adams, C., Sylvester, P., Zolotarev, M., and R. Zuccherato, "Internet X.509 Public Key Infrastructure Data Validation and Certification Server Protocols", RFC 3029, February 2001.

[25] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", RFC 4211, September 2005.

[26] Cooper, M., Dzambasow, Y., Hesse, P., Joseph, S., and R. Nicholas, "Internet X.509 Public Key Infrastructure: Certification Path Building", RFC 4158, September 2005.

[27] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[37] IANA MIKEY Payload Name Spaces per RFC 3830, see http://www.iana.org/assignments/mikey-payloads.

[29] Hoffman, P. and B. Schneier, "Attacks on Cryptographic Hashes in Internet Protocols", RFC 4270, November 2005.

[30] Bellovin, S.M. and E.K. Rescorla: "Deploying a New Hash Algorithm", October 2005, http://www.cs.columbia.edu/~smb/papers/new-hash.pdf.

[31] Milne, A., Blaser, M., Brown, D., and L. Dondetti, "ECC Algorithms For MIKEY", Work in Progress, June 2005.

[32] Bellare, M.: "New Proofs for NMAC and HMAC: Security Without Collision-Resistance", http://eprint.iacr.org/2006/043.pdf, November 2005.

[33] Ignjatic, D., Dondeti, L., Audet, F., and P. Lin, "An additional mode of key Distribution in MIKEY: MIKEY-RSA-R", Work in Progress, August 2006.

Appendix A. Usage of MIKEY-DHHMAC in H.235

This appendix provides informative overview how MIKEY-DHHMAC can be applied in some H.323-based multimedia environments. Generally, MIKEY is applicable for multimedia applications including IP telephony. [15] describes various use cases of the MIKEY key management protocols (MIKEY-PS, MIKEY-PK, MIKEY-DHSIGN and MIKEY-DHHMAC) with the purpose to establish TGK keying material among H.323 endpoints. The TGKs are then used for media encryption by applying SRTP [20]. Addressed scenarios include point-to-point with one or more intermediate gatekeepers (trusted or partially trusted) in between.

One particular use case addresses MIKEY-DHHMAC to establish a media connection from an endpoint B calling (through a gatekeeper) to another endpoint A that is located within that same gatekeeper zone. While EP-A and EP-B typically do not share any auth_key a priori, some separate protocol exchange means are achieved outside the actual call setup procedure to establish an auth_key for the time while endpoints are being registered with the gatekeeper; such protocols exist [15] but are not shown in this document. The auth_key between the endpoints is being used to authenticate and integrity protect the MIKEY-DHHMAC messages.

To establish a call, it is assumed that endpoint B has obtained permission from the gatekeeper (not shown). Endpoint B as the caller builds the MIKEY-DHHMAC I_message (see section 3) and sends the I_message encapsulated within the H.323-SETUP to endpoint A. A routing gatekeeper (GK) would forward this message to endpoint B; in case of a non-routing gatekeeper, endpoint B sends the SETUP directly to endpoint A. In either case, H.323 inherent security mechanisms [21] are applied to protect the (encapsulation) message during transfer. This is not depicted here. The receiving endpoint A is able to verify the conveyed I_message and can compute a TGK. Assuming that endpoint A would accept the call, EP-A then builds the MIKEY-DHHMAC R_message and sends the response as part of the CallProceeding-to-Connect message back to the calling endpoint B (possibly through a routing gatekeeper). Endpoint B processes the conveyed R_message to compute the same TGK as the called endpoint A.

1.) EP-B -> (GK) -> EP-A: SETUP(I_fwd_message [, I_rev_message])
2.) EP-A -> (GK) -> EP-B: CallProceeding-to-CONNECT(R_fwd_message [, R_rev_message])

Notes: If it is necessary to establish directional TGKs for full-duplex links in both directions B->A and A->B, then the calling endpoint B instantiates the DHHMAC protocol twice: once in the direction B->A using I_fwd_message and another run

in parallel in the direction A->B using I_rev_message. In that case, two MIKEY-DHHMAC I_messages are encapsulated within SETUP (I_fwd_message and I_rev_message) and two MIKEY-DHHMAC R_messages (R_fwd_message and R_rev_message) are encapsulated within CallProceeding-to-CONNECT. The I_rev_message corresponds with the I_fwd_message. Alternatively, the called endpoint A may instantiate the DHHMAC protocol in a separate run with endpoint B (not shown); however, this requires a third handshake to complete.

For more details on how the MIKEY protocols may be deployed with H.235, please refer to [15].

Author's Address

Martin Euchner
Hofmannstr. 51
81359 Munich, Germany

Phone: +49 89 722 55790
Fax: +49 89 722 62366
EMail: martin_euchner@hotmail.com

Full Copyright Statement

Intellectual Property

Acknowledgement

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