| | | |
|---|---|---|
| Host Identity Protocol | T. Heer | |
| Internet-Draft | Distributed Systems Group, RWTH | |
| Intended status: Experimental | Aachen University | |
| Expires: May 14, 2008 | November 11, 2007 | |

TOC

**End-Host Authentication for HIP Middleboxes**
**draft-heer-hip-middle-auth-00**

### Status of this Memo



### Abstract

The Host Identity Protocol is a signaling protocol for secure communication, mobility, and multihoming by introducing a cryptographic namespace. This document specifies an extension for HIP that enables middleboxes to unambiguously verify the identities of hosts that communicate across them. This extension enables middleboxes to verify the liveness and freshness of a HIP association and, thus, enables reliable and secure access control in middleboxes.

**Requirements Language**

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.

**Notation**

| | |
|---|---|
| [x] | indicates that x is optional. |
| {x} | indicates that x is under signature. |
| Initiator | is the host which initiates a HIP association (cf. HIP base protocol). |
| Responder | is the host which responds to the INITIATOR (cf. HIP base protocol). |
| --> | signifies "Initiator to Responder" communication. |
| <-- | signifies "Responder to Initiator" communication. |

---

**Table of Contents**



---

## 1. Introduction

TOC

The Host Identity Protocol (HIP) introduces a new cryptographic namespace, based on public keys, in order to secure Internet communication. This namespace allows hosts to authenticate their peers. HIP was designed to be middlebox-friendly and allows middleboxes to inspect HIP control traffic. Such middleboxes are e.g. firewalls and Network Address Translators (NATs).

In this context, one can distinguish HIP-aware middleboxes, which were designed to process HIP packets, and other middleboxes, which are not aware of the Host Identity Protocol. This document addresses only on HIP-aware middleboxes while the behavior of HIP in combination with non-HIP-aware middleboxes is specified elsewhere [I-D.ietf-hip-nat-traversal] (Komu, M., Henderson, T., Tschofenig, H., Melen, J., and A. Keranen, "Basic HIP Extensions for Traversal of Network Address Translators," October 2009.). Moreover, the scope of this document is restricted to middleboxes that use HIP in order to enforce access regulation and, thus, need to authenticate the communicating peers that send traffic over the middlebox. The class of middleboxes, this document focuses on, does not require explicit registration via a handshake with the middlebox. HIP behavior for interacting and registering to such middleboxes is specified in [I-D.ietf-hip-registration] (Laganier, J., "Host Identity Protocol (HIP) Registration Extension," June 2006.). Thus, we focus on middleboxes that build their state-base from packets it forwards.

An example for such a middlebox is a firewall that only allows traffic from certain hosts to traverse. We assume that access regulation is performed based on Host Identities (HIs). Such an authenticating middlebox needs to observe the HIP Base EXchange (BEX) or a HIP mobility update [I-D.ietf-hip-mm] (Henderson, T., "End-Host Mobility and Multihoming with the Host Identity Protocol," March 2007.)" and check the Host Identifiers (HIs) in the packets.

Along the lines of [I-D.tschofenig-hiprg-hip-natfw-traversal] (Tschofenig, H. and M. Shanmugam, "Traversing HIP-aware NATs and Firewalls: Problem Statement and Requirements," July 2007.), an authentication solution for middleboxes must have some vital properties. For one, the middlebox must be able to unambiguously identify one or both of the communicating peers. For another, the solution must not allow for new attacks against the middlebox. This document specifies a HIP extension that allows middleboxes to participate in the HIP handshake and the HIP update process in order to enable these devices to reliably verify the identities of the communicating peers. To this end, this HIP extension defines how middleboxes can interact with end-hosts in order to verify the identity of the end-hosts.

Verifying public-key (PK) signatures is costly in terms of CPU cycles. Thus, in addition to authentication capabilities, it is also necessary to provide middleboxes with a way of defending against resource-exhaustion attacks that target PK signature verification. This document defines how middleboxes can utilize the HIP puzzle mechanism defined in [I-D.ietf-hip-base] (Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, “Host Identity Protocol,” October 2007.) to slow down resource-exhaustion attacks.

---

### 1.1.  Authentication and Replay Attacks

TOC

Middleboxes need to be able to verify the HIs in the HIP base exchange messages to perform access control based on Host Identities. However, passive verification of identifiers in the messages is not sufficient to verify the identity of an end-host. Moreover, it is necessary to also ensure the freshness and authenticity of the communication to prevent replay attacks. The basic HIP protocol as specified in [I-D.ietf-hip-base] (Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, “Host Identity Protocol,” October 2007.) does not provide adequate protection against these attacks. To illustrate the need for additional security features, we briefly outline a possible replay attack targeted at middleboxes:
Assume that a middlebox M checks HIP HIs in order to restrict traffic passing through the box. Further assume that the legitimate owner of HIT X establishes a HIP association with the legitimate owner of HIT Y at some point in time and an attacker A overhears the base exchange and records it. Note that it is not required that the middlebox M is on the communication path between the peers at that time.
At some later point in time, A collaborates with another attacker B. They replay the very same BEX with the middlebox M on the communication path. The middlebox has no way to distinguish X and Y A and B as it can only overhear the BEX passively and does not participate in the authentication process. If A and B have agreed on a shared secret beforehand, they can make fake ESP traffic traverse the middlebox by using the SPIs that A and B negotiated in the original BEX. This is problematic in cases for which the middlebox needs to know who is communicating across it. Examples for such cases are access restriction, logging of activities, and accounting for traffic volume or connection duration.
So far, this attack is not addressed by the HIP specifications. Therefore, this document specifies a HIP extension that allows middleboxes to defend against it.

---

## 2.  Protocol Overview

TOC

The following section gives an overview of the interaction between hosts and authenticating middleboxes.

### 2.1. Signed Middlebox Nonces

TOC

The aforementioned attack scenario clearly shows the necessity for unambiguous end-host identity verification by middleboxes. Relying on nonces generated by the end-hosts is not possible because middleboxes can not verify the freshness of these nonces. Introducing time-stamps restricts the attack to a certain time frame but requires global time synchronization.

The following sections specify how HIP hosts can prove their identity by performing a challenge-response protocol between the middlebox and the end-hosts. As the challenge, the middlebox add data (e.g. nonces) to HIP control packets which end-hosts must echo with applied PK signatures.

The challenge-response mechanism is similar to the ECHO_REQUEST/ ECHO_RESPONSE mechanism used by HIP end-hosts to authenticate their peers. Middleboxes may add ECHO_REQUEST_M parameters to HIP control packets and verify ECHO_RESPONSE_M parameters. By echoing the data in the ECHO_REQUEST_M parameter as ECHO_RESPONSE_M parameter in the signed part of its response, an end-host proves that it is in possession of the private key that corresponds to the HI it uses.

#### 2.1.1. ECHO_REQUEST_M

TOC

Middleboxes MAY add ECHO_REQUEST_M parameters to the the R1, I2, and to any UPDATE packet. This parameter contains an opaque data block of variable size which is used by the middlebox to carry arbitrary data. Each of the afore-mentioned HIP packets may contain multiple ECHO_REQUEST_M parameters. As all middleboxes on the path may need to add ECHO_REQUEST_M parameters, the length of the data field of each parameter SHOULD not exceed a maximum of 32 bytes. The total length of the packets SHOULD not exceed 1280 bytes to avoid IPv6 fragmentation (cf. Section Section 2.4 (Fragmentation)).

The ECHO_REQUEST_M parameter is added to the unprotected part of a HIP message. Thus it does not corrupt any HMAC or public-key signatures. However, it is necessary to recompute the IP- and HIP header checksums. The UDP headers of UDP encapsulated HIP packets MUST also be recomputed if UDP encapsulation, as defined in [I-D.ietf-hip-nat-traversal] (Komu, M., Henderson, T., Tschofenig, H., Melen, J., and A. Keranen, “Basic HIP Extensions for Traversal of Network Address Translators,” October 2009.), is applied.

An end-host that receives a HIP control packet containing one or multiple ECHO_REQUEST_M parameters must copy the contents of each parameter without modification to an ECHO_RESPONSE_M parameter. This

parameter MUST be sent within the signed part of its reply. Note that middleboxes MAY also rewrite the ECHO_REQUEST_UNSIGNED parameter as specified in [I-D.ietf-hip-base] (Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, “Host Identity Protocol,” October 2007.) when the receiver of the parameter is not required to sign the contents of the ECHO_REQUEST_M.
Middleboxes can delay state creation by utilizing the ECHO_RESPONSE_M and ECHO_REQUEST_M parameter. Encrypted or otherwise protected information about previous authentication steps can be hidden in the opaque blob.

---

### 2.1.2. ECHO_RESPONSE_M

TOC

When a middlebox injects an opaque blob of data via an ECHO_REQUEST_M parameter, it expects to receive the same data without modification as part of an ECHO_RESPONSE_M parameter in a subsequent packet. The opaque data MUST be copied as it is from the corresponding ECHO_REQUEST_M parameter. In case of multiple ECHO_REQUEST_M parameters, their order MUST be preserved by the corresponding ECHO_RESPONSE_M parameters.
The ECHO_REQUEST_M and ECHO_RESPONSE_M parameters MAY be used for any purpose, in particular when a middlebox needs to carry state or recognizable information in a HIP packet and receive it in a subsequent response packet. The ECHO_RESPONSE_M MUST be covered by the HIP_SIGNATURE.
The ECHO_RESPONSE_M parameter is non critical. Depending on its local policy, a middlebox can react differently on a missing ECHO_RESPONSE_M parameter. Possible actions range from degraded or restricted service such as bandwidth limitation up to refusing connections and reporting access violations.

---

### 2.1.3. Middlebox Puzzles

TOC

As public-key (PK) operations are costly in terms of CPU cycles, it is necessary to provide some way for the middlebox to defend against resource-exhaustion attacks. The HIP base protocol [I-D.ietf-hip-base] (Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, “Host Identity Protocol,” October 2007.) specifies a puzzle mechanism to protect the Responder from I2 floods that require numerous public-key operations. However, middleboxes can not utilize this mechanism as there is no defense against a collaborative replay attack, which involves a malicious Initiator and a malicious Responder. This section specifies how middleboxes can utilize the puzzle mechanism to add their own puzzles to R1, I2, and any UPDATE packets. This allows middleboxes to shelter against Service (DoS) attacks on PK verification.

To defend against attacks, a middlebox adds a puzzle in a PUZZLE_M parameter to I2, R2 and UPDATE packets. Depending on the packet to which the puzzle was added, either the Initiator or the Responder of a BEX or the receiver of an UPDATE packet must solve it.

A puzzle increases the delay and computational cost for establishing or updating a HIP association, a middlebox SHOULD only add puzzles to packets if it is under attack conditions. Moreover, middleboxes SHOULD distinguish attack directions. If the majority of the CPU load is caused by verifying HIP control messages that arrive from a certain interface, middleboxes MAY add puzzles with higher difficulty to HIP control packets that leave the interface.

Middleboxes MAY decide to use only the PUZZLE_M parameter instead of using PUZZLE_M in combination with ECHO_REQUEST_M because the PUZZLE_M parameter also contains an opaque data field that guarantees the freshness of the signature. However, the opaque data field in the PUZZLE_M and the corresponding SOLUTION_M parameter is restricted to 6 bytes which may not be sufficient for all purposes.

## 2.2. Identity Verification by Middleboxes

TOC

This section describes how middleboxes can interact with the BEX and the HIP update process in order to verify the identity of the HIP end-hosts.

### 2.2.1. Identity Verification During BEX

TOC

Middleboxes MAY add ECHO_REQUEST_M and PUZZLE_M parameters to R1 and I2 packets in order to verify the identities of the participating parties. Middleboxes can choose to either authenticate the Initiator, the Responder, or both. Middleboxes MUST NOT add ECHO_REQUEST_M or PUZZLE_M parameters to I1 messages because this would expose the Responder to DoS attacks. Thus, middleboxes MUST let unauthenticated minimal I1 packets traverse. Minimal means that the packet MUST NOT contain more than the minimal set of parameters specified by HIP standards or internet drafts. In particular, the I1 packet MUST NOT contain any attached payload. Figure 1 illustrates the authentication process during the BEX.

Figure 1: Middlebox authentication of a HIP base exchange.

```
Main path:

Initiator                 Middlebox                         Responder
                     .------------------.
 I1                  |                  |  I1
-------------------> |                  | ---------------------------->
                     |                  |
 R1, + EQ1, [PM1]    | Add EQ1, PM1     |  R1
<------------------- |                  | <----------------------------
                     |                  |
 I2, {ER1, SM1}      | Verify SM1, EQ1  |  I2, {ER1, SM1} + EQ2, [PM2]
-------------------> | Add EQ2, PM2     | --------------------------->
                     |                  |
                     |                  |
 R2, {ER2, SM2}      | Verify SM2, ER2  |  R2, {ER2, SM2}
-------------------> |                  | ---------------------------->
                     '------------------'

EQ: Middlebox Echo reQuest
ER: Middlebox Echo Response
PM: Puzzle of the Middlebox
SM: Solution of Middlebox puzzle
```

---

### 2.2.2. Identity Verification During Mobility Updates

TOC

Multihomed hosts may use multiple communication paths during an HIP mobility update. Depending on whether the middlebox is located on the communication path between the preferred locators or not, the middlebox forwards different packets and, thus, needs to interact differently with the updates. Figure 1 illustrates an update with Middlebox 1 on the path between the Initiator's and the RECEIVER's preferred locators and with Middlebox 2 on an alternative path.
Middlebox 1 receives the first UPDATE packet, which contains e.g. the set of new locators. As the middlebox has no adequate way of identifying replay attacks of U1 (first UPDATE message) and, moreover cannot defend against U1 flooding attacks, the middlebox may decide not to verify the signature in the U1 packet. In the case it is necessary to verify the identity of the Responder and the freshness of the UPDATE packets, the middlebox MAY add an ECHO_REQUEST_M (EQ1) to the U1.
The following figure illustrates the authentication for middleboxes on the path between the preferred locators (main path) and other paths between two HIP peers (alternative path).

Figure 1: Middlebox authentication of a HIP mobility update over different paths.

```
Main path:

Initiator                     Middlebox 1                  Responder
                               .------.
 U1                            |      |  U1 + EQ1, [PM1]
-----------------------------> |      | ----------------------------->
                               |      |
 U2, {ER1, [SM1]} + EQ2, [PM2] |      |  U2, {ER1, [SM1]}
<----------------------------- |OK    | <-----------------------------
                               |      |
 U3, {ER2, SM2}                |      |  U3, {ER2, SM2}
-----------------------------> |    OK| ----------------------------->
                               '------'

Alternative path:

Initiator                     Middlebox 2                  Responder
                               .------.
 U2, {ER1, [SM1]} + P3, [PM3]  |      |  U2, {ER1, [SM1]}
<----------------------------- | wrong| <-----------------------------
                               |      |
 U3', {ER3, SM3}               |      |  U3', {ER3, SM3} + EQ4, PM4
-----------------------------> |OK    | ------------------------------>
                               |      |
 U4, {ER4, [SM4]}              |      |  U4, {ER1, [SM1]}
<----------------------------- |    OK| <-----------------------------
                               '------'
EQ: Middlebox Echo reQuest
ER: Middlebox Echo Response
PM: Puzzle of the Middlebox
SM: Solution of Middlebox puzzle
```

Middlebox 1 can verify the identity of the Responder by checking its PK signature and the presence of the ECHO_RESPONSE_M in the U2 packet. If necessary, the middlebox MAY add an ECHO_REQUEST_M for the Initiator of the update. The middlebox can verify the Initiator's identity by verifying its signature and the ECHO_RESPONSE_M in the U3 packet.
A middlebox that is not located on the path between preferred locators of the HIP end-hosts does not receive the U1 message. Therefore, it will not recognize any ER1 or SM1 in the second UPDATE packet. Thus, if a middlebox encounters non-matching or missing ECHO_RESPONSE_M parameters, the middlebox SHOULD ignore these.
When receiving an UPDATE message with an ECHO_REQUEST_M, a HIP host SHOULD send an UPDATE message containing the corresponding

ECHO_RESPONSE_M covered by a HIP_SIGNATURE parameter. Otherwise the middlebox may refuse to make the communication path available to the HIP host.

---

### 2.2.3. UPDATE Verification

TOC

As middleboxes need to be able to rapidly verify and forward HIP packets, these devices need to be supplied with all information necessary to do so. If, due to host mobility, a new communication path is used, middleboxes need to be able to learn the Host Identifiers (HIs) from the UPDATE packets. Therefore, HIP hosts MUST include the HOST_ID parameter in all UPDATE packets that use combinations of locators that have not been used before. Thus, UPDATE packets that contain ECHO_REQUEST or ECHO_RESPONSE parameters MUST contain the HOST_ID parameter. Moreover, all packets that contain an ECHO_RESPONSE_M parameter MUST contain the HOST_ID parameter.

---

### 2.3. Failure Signaling

TOC

Middleboxes SHOULD inform the sender of a BEX or update message if it does not satisfy the requirements of the middlebox. Reasons for non-satisfactory packets are missing HOST_ID, ECHO_RESPONSE_M, and SOLUTION_M parameters. Options for expressing such shortcomings are ICMP or HIP_NOTIFY packets. Defining this signaling mechanism is future work.

---

### 2.4. Fragmentation

TOC

Analogously to the specification in [I-D.ietf-hip-base] (Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, “Host Identity Protocol,” October 2007.), HIP aware middleboxes SHOULD support IP-level fragmentation and reassembly for IPv6 and MUST support IP-level fragmentation and reassembly for IPv4. However, when adding ECHO_REQUEST_M and PUZZLE_M parameters, a middlebox SHOULD keep the total packet size below 1280 bytes to avoid packet fragmentation in IPv6.

---

TOC

## 3. HIP Parameters

This HIP extension specifies four new HIP parameters that allow middleboxes to authenticate HIP end-hosts and to protect against DoS attacks.

---

### 3.1. ECHO_REQUEST_M

TOC

The ECHO_REQUEST_M parameter MAY be added to R1, I2, and UPDATE packets by HIP-aware middleboxes. The structure of the ECHO_REQUEST_M parameter is depicted below:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Opaque data (variable length)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Type          65332
Length        Variable
Opaque data   Opaque data, supposed to be meaningful only to the
              middlebox that adds ECHO_REQUEST_M and receives a
              corresponding ECHO_RESPONSE_M.
```

---

### 3.2. ECHO_RESONSE_M

TOC

The ECHO_RESPONSE_M is the reply to the ECHO_REQUEST_M parameter. The receiver of an ECHO_RESPONSE_M parameter SHOULD reply with n ECHO_RESPONSE_M. If not, the middlebox that added the parameter MAY decide to degrade or deny its service. The contents of the ECHO_REQUEST_M parameter must be copied to the ECHO_RESPONSE_M parameter without any modification. The ECHO_RESPONSE_M parameter is non-critical and covered by the SIGNATURE. The structure of the ECHO_RESPONSE_M parameter is depicted below:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Opaque data (variable length)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Type         962
Length       Variable
Opaque data  Opaque data, supposed to be meaningful only to the
             middlebox that adds adds ECHO_REQUEST_M and receives a
             corresponding ECHO_RESPONSE_M.
```

---

### 3.3. PUZZLE_M

TOC

A middlebox MAY add a PUZZLE_M parameter to R1, I2, and UPDATE packets. A HIP packet may contain multiple PUZZLE_M parameters as multiple middleboxes may be located on a communication path. These puzzles serve as defense against DoS attacks. Hosts that receive a PUZZLE_M parameter SHOULD reply with a SOLUTION_M parameter in the subsequent I2, R2, or UPDATE packet. With the exception of an extended opaque field, the format and meaning of the puzzle are defined in [I-D.ietf-hip-base] (Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, “Host Identity Protocol,” October 2007.). The reader is advised to refer to that document for a detailed specification of the puzzle mechanism. The extended opaque data field helps middleboxes to recognize their puzzles and solutions, respectively, if a packet contains more than one puzzle. A middlebox MUST preserve the order of PUZZLE_M parameters in a packet and attach its own PUZZLE_M parameter after all other PUZZLE_M parameters. Preserving the order of PUZZLE_M parameters may help middleboxes to recognize the puzzles and solutions relevant to a middlebox.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| K, 1 byte     |    Lifetime   |        Opaque, 6 bytes        /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Random # I, 8 bytes                                           |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Type           65334
Length         16
K              K is the number of verified bits
Lifetime       Puzzle lifetime 2^(value-32) seconds
Opaque         Data set by the middlebox, indexing the middlebox
Random #I      Random number
```

---

### 3.4. SOLUTION_M

TOC

The SOLUTION_M parameter contains the solution for the corresponding PUZZLE_M parameter. End-hosts that receive a PUZZLE_M parameter SHOULD solve the puzzle according to the specification in [I-D.ietf-hip-base] (Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, "Host Identity Protocol," October 2007.) and send the resulting solution in the SOLUTION_M parameter. Exclusion of a solution MAY result in degraded or denied service by the middlebox that added the PUZZLE_M parameter. The format and meaning of the fields in the SOLUTION_M parameter resemble the specifications of the SOLUTION parameter in [I-D.ietf-hip-base] (Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, "Host Identity Protocol," October 2007.). The reader is advised to refer to that document for further details. The extended opaque data field helps middleboxes to recognize their puzzles and the resulting solutions, respectively, when a packet contains multiple puzzles.
The relative order of SOLUTION_M parameters in a HIP control packet MUST match the order of the PUZZLE_M parameters in the previously received packet. Preserving the order of PUZZLE_M for the corresponding SOLUTION_M parameters may help middleboxes to recognize the puzzles and solutions relevant to them.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| K, 1 byte     |    Reserved   |        Opaque, 6 bytes        /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Random # I, 8 bytes                                           |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Puzzle solution #J, 8 bytes                                   |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


Type              322
Length            20
K                 K is the number of verified bits
Reserved          Zero when sent, ignored when received
Opaque            Copied unmodified from the received PUZZLE
                  parameter
Random #I         Random number
Puzzle solution   Random number
```

---

## 4. Security Considerations

TOC

This HIP extension specifies how HIP-aware middleboxes interact with the handshake and mobility-signaling of the Host Identity Protocol. Its scope is restricted to the authentication of end-hosts and does not include the issue of authenticating ESP traffic on the middlebox. Providing middleboxes with a way of adding puzzles to the HIP control packets may cause both HIP peers, including the Responder, to spend CPU time on solving these puzzles. Thus, it is advised that HIP implementations for servers employ mechanisms to prevent middlebox puzzles from being used as DoS attacks. Under high CPU load, servers can e.g. prioritize packets that do not contain difficult middlebox puzzles.
If multiple middleboxes add ECHO_REQUEST_M parameters to a HIP control packet, the remaining space in the packet might not be sufficient for further parameters to be added. Moreover, as the ECHO_REQUEST_M must be

echoed within an ECHO_RESPONSE_M, the space in the subsequent packet may not be sufficient to add all ECHO_RESONSE_M parameters. Thus, middleboxes SHOULD keep the size of the nonces small.

## 5. IANA Considerations

TOC

This document specifies four new HIP parameter types. The preliminary parameter type numbers are 322, 962, 65332, and 65334.

## 6. Acknowledgments

TOC

Thanks to Shaohui Li, Miika Komu, and Janne Lindqvist for the fruitful discussions on this topic. Many thanks to Stefan Goetz and Rene Hummen commenting and helping to improve the quality of this document.

## 7. Normative References

TOC


| | |
|---|---|
| [I-D.ietf-hip-base] | Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, “Host Identity Protocol,” draft-ietf-hip-base-10 (work in progress), October 2007 (TXT). |
| [I-D.ietf-hip-mm] | Henderson, T., “End-Host Mobility and Multihoming with the Host Identity Protocol,” draft-ietf-hip-mm-05 (work in progress), March 2007 (TXT). |
| [I-D.ietf-hip-nat-traversal] | Komu, M., Henderson, T., Tschofenig, H., Melen, J., and A. Keranen, “Basic HIP Extensions for Traversal of Network Address Translators,” draft-ietf-hip-nat-traversal-09 (work in progress), October 2009 (TXT). |
| [I-D.ietf-hip-registration] | Laganier, J., “Host Identity Protocol (HIP) Registration Extension,” draft-ietf-hip-registration-02 (work in progress), June 2006 (TXT). |
| [I-D.tschofenig-hiprg-hip-natfw-traversal] | Tschofenig, H. and M. Shanmugam, “Traversing HIP-aware NATs and Firewalls: Problem Statement and Requirements,” draft-tschofenig-hiprg-hip-natfw-traversal-06 (work in progress), July 2007 (TXT). |
| [RFC2119] | Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML). |

## Author's Address

TOC


| | |
|---|---|
| | Tobias Heer |
| | Distributed Systems Group, RWTH Aachen University |
| | Ahornstrasse 55 |
| | Aachen 52062 |
| | Germany |
| Phone: | +49 241 80 214 36 |
| Email: | heer@cs.rwth-aachen.de |
| URI: | http://ds.cs.rwth-aachen.de/members/heer |


## Full Copyright Statement

TOC



## Intellectual Property

standard. Please address the information to the IETF at ietf-ipr@ietf.org.