LHIP: Lightweight Authentication for HIP
draft-heer-hip-lhip-00.txt

Tobias Heer
RWTH Aachen University,
Helsinki Institute for Information Technology

68th IETF Meeting, Prague
March 2007
Motivation

- HIP is great!
- Host authentication
- End-to-end encryption
- Mobility (MM extension)
- Multihoming (MM extension)
- ....

But: quite much PK cryptography involved
Some Numbers

- **Nokia N770**
  - CPU: ARM 220 Mhz
- **Benchmarks**
  - RSA
  - DSA
  - DH
### Some Numbers (cont'd)

<table>
<thead>
<tr>
<th>Initiator</th>
<th>Responder</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BEX</strong></td>
<td></td>
</tr>
<tr>
<td>2x Verify</td>
<td>1x Verify</td>
</tr>
<tr>
<td>1x Sign</td>
<td>1x Sign</td>
</tr>
<tr>
<td>1x DH</td>
<td>1x DH</td>
</tr>
</tbody>
</table>

- **Update**
  - 1x Verify   | 1x Verify   |
  - 1x Sign     | 1x Sign     |

- **Close**
  - 1x Verify   | 1x Verify   |
  - 1x Sign     | 1x Sign     |

---

**„Off-the-shelve“ N770 as Initiator**

- HI initiator: RSA 1024
- HI responder: DSA 1536

DH key-length: 384

- **BEX**: 797 ms
- **Update**: 469 ms
- **Close**: 469 ms
Why are These Numbers Problematic?

- Not just one HIP association!
  - UPDATEs (several open HIP associations)
  - Simultaneous BEXes
- Can't we just reduce the key length?
  - Weak keys?
  - Servers: multiple keys for multiple classes of clients?
- Won't time heal it?
  - Over-provision devices just for HIP?
  - More HIP hosts – more HIP associations
Lightweight HIP

- Idea was floating around for a while
- Master's thesis
  - Protocol proposal
  - Implementation
  - Performance evaluation

- Is this LHIP what the HIP folks want/need?
What is LHIP?

- HIP without PK
  - No host authentication
  - No encryption
- Reuse HIP namespace
  - ID locator split
  - Same name for LHIP and HIP
  - But don't break HIP!
- Support for MM
  - Authenticated UPDATEs
- Upgrade from LHIP to HIP
What LHIP can/can't do without PK

• LHIP cannot:
  – ... authenticate a host's identity (w/o PK)
  – ... encrypt payload
  – ... protect against MITM during BEX

• LHIP can:
  – ... authenticate succeeding messages
  – ... integrity protect control messages
  – ... protect against MITM after BEX
  – Middleboxes can verify LHIP control messages
Outline

- LHIP authentication
- LHIP associations (BEX)
- Closing an LHIP association
- Upgrade from LHIP to HIP
- Open questions
LHIP in the Stack

API

TCP / UDP

(L)HIP

LHIP Auth

IP

IPsec

(NULL mode)
How to Substitute RSA/DSA/DH?

- No shared keys anymore:
  - Authentication of HIP control packets?
  - e.g. UPDATE from new IP?
- Interactive Hash Chain (IHC) based signatures
- Similar to Weak Identifier Multihoming Protocol
  - 2004: draft-ylitalo-multi6-wimp-00
- Very low processing cost to sign & verify
- BUT: One additional RTT per signed packet
Hash Chains

- Cryptographic hash function $H$
- $h_0 = H(\text{rand})$
- $h_1 = H(h_0) = H(H(\text{rand}))$
- ...
- $h_n = H(h_{n-1}) = H(...H(H(\text{rand}))...)$
- $(h_n, h_{n-1}, ..., h_1, h_0, \text{rand})$
- Can be used for authentication
- $h_n$ is denoted anchor
IHC Based Signatures

Sender

$h^v_i$

A1: $h^v_{i-1}$

S1: $h^s_{i-1}$, msg, HMAC(msg, $h^s_{i-2}$)

S2: $h^s_{i-2}$

Verifier

$h^s_i$

(... $h_i$, $h_{i-1}$, ... $h_1$, $h_0$, rand)
IHC Based Signatures

**Sender**

- \( h_i^v \)
- \( S1: h_{i-1}^s, \text{msg, HMAC}(\text{msg, } h_{i-2}^s) \)
- \( A1: h_{i-1}^v \)
- \( S2: h_{i-2}^s \)

**Verifier**

- \( h_i^s \)
- \( \text{Signature} \)

- \( (... h_i, h_{i-1}, ... h_1, h_0, \text{rand}) \)
LHIP & IHC Based Signatures

- LHIP uses a variant of the IHC based signature
  - Easier to handle for middleboxes
  - Eliminated a possibility for a MITM attack
- Authenticated duplex channel
- LHIP signs the HIP HMAC parameter
  - 0..0 as HMAC key
  - HIP HMAC is used as message digest
  - Same semantics
LHIP Control Message Authentication

1. **S1:** Signature
   - **A1:** Ack
   - **S2:** Message, key
     - **A2:** Ack

Message Buffer:
- **LHIP**

HIP

Message Buffer:
- **LHIP**

HIP
LHIP Mobility Update

Initiator

S1
A1
Update message
A2
S1
A1
Update message
A2

Responder

+ 1 RTT

Parallel

Parallel
Predefined Signals

- Simple signaling with predefined output
  - e.g. CLOSE
    - Close association if sent
    - No additional information needed
    - Protection required
- Exchange $h^c_0 = H(rand)$ during BEX
- Disclose $rand$ if predefined signal is sent
  - e.g. add $rand$ to CLOSE message
- Peer and middleboxes can authenticate signal
LHIP BEX

- Similar to HIP BEX
  - 4 way
  - I1 identical for both
  - Additional parameters in R1, I2, R2
    - Hash chain anchors
  - Modified parameters
    - HIP_TRANSFORM: new LHIP suite
    - Mandatory ECHO_REQUEST
  - Unused parameters (during BEX)
    - Diffie-Hellman public keys is still present
HIT Blocking Attack

1. Attacker: Connect, HI V

2. Victim: Connect, HI V
HIT Stealing Attack

Server: \textit{HI S}

1. Attacker: Connect, \textit{HI S}

2. Victim: Connect to \textit{HI S}
RSA/DSA is Required (in some cases)

- Protect the HIP namespace
- Protect pure HIP hosts in particular
- PK authentication is required...
  - In case of collisions:
    - second LHIP host must authenticate
  - During association establishment:
    - Authenticate incoming or outgoing comm.
- Optional request for host authentication
  - Signaled in R1 and I2
LHIP Payload

- **IPsec**
  - No symmetric keys available
  - ESP NULL mode w/o AH?
  - Simpler to implement
  - Same payload handling for HIP & LHIP

- **IP**
  - No keys.... that's okay!
  - How to “catch” and process packets?
  - Harder to implement
LHIP Payload (cont'd)

- Currently unprotected
- Use cleartext key as “secret”?  
  - Insecure if attacker eavesdrops BEX  
  - Maybe secure after mobility
- Use hash chains to protect payload?  
  - Many hash chain elements needed  
  - Mixture TESLA, IHC based signatures?
- Other options?
- Would LHIP just pretend to be somewhat secure?
LHIP Upgrade

- Triggered by:
  - Application (same socket) - API
  - Request for full HIP assoc. (other socket)

Initiator

U1: ESP_INFO, [ECHO_RESP.], HMAC, [SIG], \( h^c_0 \)

Responder

U2: ESP_INFO, [ECHO_RESP.], HMAC, [SIG], \( h^c_0 \)
BEX Performance

DSA Host Identifiers

Responder key length (bit)

Initiator key length (bit)

Total processing time (ms)
HC Signature Performance

- HC signatures
  - Sign: 2.3 ms
  - Verify: 3.1 ms
  - Plus 1.5 x RTT
- RSA / DSA
  - Signature
  - Verification
  - Plus 0.5 x RTT
LHIP Summary

• HI namespace reuse

• Performance
  – Less RSA / DSA
  – No DH

• Mobility, multihoming & more

• Middleboxes can verify signatures w/o RSA/DSA

• Extension

• Just a suggestion

• Could this be useful for the WG or RG?
Appendix I
Interactive Hash Chain Based Signatures
IHC Based Signatures

**Sender**

\[ h_i^v \]

\[ S1: h_{i-1}^s, \text{ msg, HMAC}(\text{msg, } h_{i-2}^s) \]

\[ A1: h_i^v \]

\[ S2: h_i^s \]

**Verifier**

\[ h_i^s \]
IHC Based Signatures

Sender

Verifier

\[ h_i^v \]

\[ S1: h_{i-1}^s, \]

\[ A1: h_{i-1}^v \]

\[ S2: h_{i-2}^s \]

\[ \text{msg, HMAC}(\text{msg, } h_{i-2}^s) \]

Triggers
IHC Based Signatures

Sender

\[ h^v_{i} \]

\[ S1: h^s_{i-1}, \text{ msg, HMAC(msg, } h^s_{i-2}) \]

\[ H(h^v_{i-1}) == h^v_{i} \]

\[ S2: h^s_{i-2} \]

Verifier

\[ h^s_{i} \]

\[ A1: h^v_{i-1} \]

\[ H(h^s_{i-1}) == h^s_{i} \]

\[ H(h^v_{i-2}) == h^v_{i-1} \&\& \]

\[ \text{HMAC(msg, } h^s_{i-2}) == \text{Signature} \]
IHC Based Signatures

Sender

\( h_i^v \)

S1: \( h_{i-1}^s \), \( \text{msg} \), \( \text{HMAC}(\text{msg}, h_{i-2}^s) \)

A1: \( h_{i-1}^v \), Pre-Ack, Pre-Nack

S2: \( h_{i-2}^s \), \( \text{msg} \)

A2: \( h_{i-2}^v \), Ack / Nack

Verifier

\( h_i^s \)

Pre - signature
Message Queueing

1) Take control packet from HIP (msg)
2) [Queue msg]
3) Send signed message
4) [Send next msg in Queue]
What do we need PK crypto for?

• Authentication (RSA or DSA)
  – Packet authentication
  – Host authentication

• Shared secret generation (Diffie Hellman)
  – Packet authentication (HMAC)
  – Payload encryption (AES, 3DES, Blowfish)

→ Minimize the use of RSA and DSA, replace Diffie Hellman!