Outline

- Host Identity Protocol
  - Idea behind HIP
  - Setting up associations

- Mobility and multihoming
  - Host mobility
  - Host multihoming
  - Network mobility

- Additional functionality
  - Registration
  - Rendezvous mechanism
  - DNS changes
Outline (cont)

- Middleboxes
  - NATs
- Hi3
- Standardization
  - Current status
- Ericsson HIP implementation
Host Identity Protocol

Basics
draft-ietf-hip-base-02
Host Identity Protocol

- HIP basic idea
  - Identity / locator split
  - HIP features

- Setting up connections
  - The HIP base exchange
Host Identity Protocol

Why HIP?

- HIP provides a combination of useful features:
  - Identifier locator split
  - Security
  - Mobility
  - IPv4 and IPv6 interoperability
  - Multi-homing
Host Identity Protocol

Why HIP?

- They are available elsewhere but....
- IP addresses no longer work for identifying hosts
- IPsec is hard to configure
- Mobile IP is large and complex
- Mobile IPv4 and IPv6 do not work together
- No simple solutions for multiaccess
Host Identity Protocol

The basic idea

- Process
- Transport
- IP layer
- Link layer

<IP addr, port>

IP address
Host Identity Protocol

The basic idea

- A new Name Space of Host Identifiers (HI)
- HIs = Public keys!
- HIs presented as hash values
  - Host Identity Tag HIT (IPv6)
  - Local Scope Identifier LSI (IPv4)
- Sockets bound to HIs, not to IP addresses
- New layer translates IP addresses to HIs and vice versa
HIP features
Identifier locator split

- Currently
  - Hosts located using the IP address
  - Hosts identified using the IP address

- Problems
  - Mobile host → new IP address → new identity
  - Connections to locations not entities
HIP features
Identifier locator split

- HIP
  - Hosts located using the IP address
  - Hosts identified using a Host Identity (HI)
  - Mobile host new IP address, same HI
  - Connections to identities (HIs)
HIP features

Security

- The Host Identity is a public key
  - Prove the ownership using private key
  - Used for host authentication and setting up HIP association

- Traffic protected with IPsec Encapsulating Security Payload (ESP)
  - ESP SA establishment during HIP base exchange
HIP features
IPv4 and IPv6 interoperability

- New layer in the IP stack, the HIP layer
  - Above: Host Identities
  - Below: IP addresses

![Diagram showing HIP features]
Host Identity Protocol
HIP packets – draft ietf hip base 02

- I1, R1, I2, R2 – Base exchange
- UPDATE – change connection parameters
  - Rekeying (e.g. SA lifetime expires)
  - Setting up additional SAs
  - Change in locators
- CER – Send certificates
- CLOSE, CLOSE_ACK – closing a HIP association
- NOTIFY – Notification messages
Host Identity Protocol

HIP packets

- Packets consist of a HEADER and zero or more parameters
- HIP header:

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| Next Header | Payload Len | Type | VER. | RES. |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| Controls | Checksum |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| Sender's Host Identity Tag (HIT) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| Receiver's Host Identity Tag (HIT) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| / |
| / |
| / |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| HIP Parameters |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```
Host Identity Protocol

Parameters

- Parameters are coded in Type Length-Value format
- For different purposes:
  - Puzzle – solution
  - Diffie-Hellman
  - Transforms
  - Signatures
  - HMACs
  - ...

HIP –
Setting up connections

draft-ietf-hip-base-02
draft-jokela-hip-esp-00
HIP base exchange

- 4-way handshake
- Creates a HIP association
  - Authentication of hosts
- Negotiates security parameters
  - Diffie-Hellman
- Establishes ESP security associations
  - Algorithms
  - Keys
- Opportunistic mode if responder’s identity unknown
  - Use only destination IP address in initialization, learn HI
The HIP base exchange
DNS query – resolving the responder’s locator

DNS query: “Responder”
DNS response: HI, IP address
The HIP base exchange

Initialization

Initiator (IN)

I1: Initialization, "Hello, I’m here. I want to talk HIP!"

Responder (RN)

DNS

Internet
The HIP base exchange
Packet I1

- The HIP Initiator packet
  - Contains only HIP header
  - Opportunistic mode: Responder’s HIT unknown

Header:

Packet Type = 1
SRC HIT = Initiator's HIT
DST HIT = Responder's HIT, or NULL
The HIP base exchange
Response: including ESP SA initialization

Initiator (IN)

R1: Challenge: "Solve this puzzle"
\[ D \text{ H}_{\text{Responder}} \]

Responder (RN)
The HIP base exchange
Packet R1

- Header
- Parameters
  [ R1_COUNTER, ]
  PUZZLE,
  DIFFIE_HELLMAN,
  HIP_TRANSFORM,
  ESP_TRANSFORM,
  HOST_ID,
  [ ECHO_REQUEST, ]
  HIP_SIGNATURE_2
  [, ECHO_REQUEST ]

- 64 bits
- Current generation of valid puzzles
- Incremented periodically by sender
The HIP base exchange
Packet R1

- **Header**
- **Parameters**
  - [ R1_COUNTER, ] PUZZLE, DIFFIE_HELLMAN, HIP_TRANSFORM, ESP_TRANSFORM, HOST_ID,
  - [ ECHO_REQUEST, ] HIP_SIGNATURE_2 [, ECHO_REQUEST ]

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>+++++++++++++++++++</td>
<td>+++++++++++++++++++</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Type</td>
<td>Length</td>
<td></td>
</tr>
<tr>
<td>+++++++++++++++++++</td>
<td>+++++++++++++++++++</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K, 1 byte</td>
<td>Lifetime</td>
<td>Opaque, 2 bytes</td>
<td></td>
</tr>
<tr>
<td>Random #I, 8 bytes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Random #I (64 bits)
- Initiator solves #J
  - K lowest bits of the hash must be zero
The HIP base exchange
Packet R1

- **Header**
- **Parameters**

[R1_COUNTER, ]
PUZZLE,
DIFFIE_HELLMAN,
HIP_TRANSFORM,
ESP_TRANSFORM,
HOST_ID,
[ECHO_REQUEST, ]
HIP_SIGNATURE_2
[, ECHO_REQUEST ]

- **Group ID**
  - 384-bit group 1
  - OAKLEY well known group 1 2
  - 1536-bit MODP group 3
  - 3072-bit MODP group 4
  - 6144-bit MODP group 5
  - 8192-bit MODP group 6

- Public value is the Diffie Hellman public key generated by the sender
  ⇒ Initiator can calculate the shared secret
The HIP base exchange
Packet R1

- Header
- Parameters

[ R1_COUNTER, ]
PUZZLE,
DIFFIE_HELLMAN,
HIP_TRANSFORM,
ESP_TRANSFORM,
HOST_ID,
[ ECHO_REQUEST, ]
HIP_SIGNATURE_2
[ , ECHO_REQUEST ]

- Proposed HIP transforms
- Initiator selects one of them
- Defined:
  - AES-CBC with HMAC-SHA1 1
  - 3DES-CBC with HMAC-SHA1 2
  - 3DES-CBC with HMAC-MD5 3
  - BLOWFISH-CBC with HMAC-SHA1 4
  - NULL-ENCRYPT with HMAC-SHA1 5
  - NULL-ENCRYPT with HMAC-MD5 6
The HIP base exchange
Packet R1

- Header
- Parameters
  [ R1_COUNTER, ]
  PUZZLE,
  DIFFIE_HELLMAN,
  HIP_TRANSFORM,
  ESP_TRANSFORM,
  HOST_ID,
  [ ECHO_REQUEST, ]
  HIP_SIGNATURE_2
  [, ECHO_REQUEST ]

- Proposed ESP transforms
- Initiator selects one of them
- Defined:
  - AES-CBC with HMAC-SHA1
  - 3DES-CBC with HMAC-SHA1
  - 3DES-CBC with HMAC-MD5
  - BLOWFISH-CBC with HMAC-SHA1
  - NULL-ENCRYPT with HMAC-SHA1
  - NULL-ENCRYPT with HMAC-MD5
# The HIP base exchange

## Packet R1

- **Header**
- **Parameters**

<table>
<thead>
<tr>
<th>R1_COUNTER,</th>
<th>PUZZLE,</th>
<th>DIFFIE_HELLMAN,</th>
<th>HIP_TRANSFORM,</th>
<th>ESP_TRANSFORM,</th>
<th>HOST_ID,</th>
<th>ECHO_REQUEST,</th>
<th>HIP_SIGNATURE_2,</th>
<th>ECHO_REQUEST</th>
</tr>
</thead>
</table>

- Responder’s HI (public key)
- At the moment: DSA or RSA
- Domain Identifier
- FQDN or NAI (login@FQDN)
The HIP base exchange

Packet R1

- **Header**
- **Parameters**

[ R1_COUNTER, ]

PUZZLE,

DIFFIE_HELLMAN,

HIP_TRANSFORM,

ESP_TRANSFORM,

HOST_ID,

[ ECHO_REQUEST, ]

HIP_SIGNATURE_2

[, ECHO_REQUEST ]

- Opaque data: must be echoed back
- May be covered by signature
The HIP base exchange

Packet R1

- **Header**
- **Parameters**

  - [ R1_COUNTER, ]
  - PUZZLE,
  - DIFFIE_HELLMAN,
  - HIP_TRANSFORM,
  - ESP_TRANSFORM,
  - HOST_ID,
  - [ ECHO_REQUEST, ]
  - HIP_SIGNATURE_2
  - [, ECHO_REQUEST ]

- Signature calculated over the HIP packet, excluding
  - Initiator’s HIT
  - checksum field
  - PUZZLE: opaque and #1
  - Any TLV after the signature TLV
The HIP base exchange
Puzzle solved, initiator continues with I2

Initiator (IN)

- Solve puzzle
- Generate keying material
- Select ESP SPI

I2: Puzzle solution, Dp,PH,HI,SPI

DNS

Responder (RN)

Internet
The HIP base exchange
Packet I2

- **Header**
- **Parameters**

```
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            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Reserved            |         Keymat Index          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Old SPI                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            New SPI                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- Keymat Index: tells the point from where keys are drawn from the keying material (zero in base exchange)
- Old SPI (zero in base exchange)
- New SPI; Initiator’s ESP SPI

- `ESP_INFO`, [ `R1_COUNTER`, ]
- `SOLUTION`
- `DIFFIE_HELLMAN`
- `HIP_TRANSFORM`
- `ESP_TRANSFORM`
- `ENCRYPTED{HOST_ID}`, [ `ECHO_RESPONSE`, ]
- `HMAC`
- `HIP_SIGNATURE_2`
The HIP base exchange
Packet I2

- Header
- Parameters

ESP_INFO,
[ R1_COUNTER, ]
SOLUTION,
DIFFIE_HELLMAN,
HIP_TRANSFORM,
ESP_TRANSFORM,
ENCRYPTED{HOST_ID},
[ ECHO_RESPONSE, ]
HMAC,
HIP_SIGNATURE_2

- R1_COUNTER may be echoed back if it was in the received R1 packet
The HIP base exchange
Packet I2

- Header
- Parameters
  ESP_INFO,
  [ R1_COUNTER, ]
  SOLUTION,
  DIFFIE_HELLMAN,
  HIP_TRANSFORM,
  ESP_TRANSFORM,
  ENCRYPTED{HOST_ID},
  [ ECHO_RESPONSE, ]
  HMAC,
  HIP_SIGNATURE_2

- #J: Calculated solution to the puzzle
  SHA-1(I, HIT I, HIT R, J)
  - K lowest order bits must be zero
The HIP base exchange
Packet I2

- Header
- Parameters
ESP_INFO,
[ R1 COUNTER, ]
SOLUTION,
DIFFIE_HELLMAN,
HIP_TRANSFORM,
ESP_TRANSFORM,
ENCRYPTED{HOST_ID},
[ ECHO_RESPONSE, ]
HMAC,
HIP_SIGNATURE_2

- Initiator’s public D_H key value
⇒ Responder can calculate the shared secret
The HIP base exchange
Packet I2

- Header
- Parameters

ESP_INFO,
[ R1_COUNTER, ]
SOLUTION,
DIFFIE_HELLMAN,
HIP_TRANSFORM,
ESP_TRANSFORM,
ENCRYPTED{HOST_ID},
[ ECHO_RESPONSE, ]
HMAC,
HIP_SIGNATURE_2

- Initiator’s selection for HIP crypto functions
- One of the proposed in R1
The HIP base exchange
Packet I2

- Header
- Parameters

ESP_INFO,
[ R1_COUNTER, ]
SOLUTION,
DIFFIE_HELLMAN,
HIP_TRANSFORM,
ESP_TRANSFORM,
ENCRYPTED{HOST_ID},
[ ECHO_RESPONSE, ]
HMAC,
HIP_SIGNATURE_2

- Initiator’s selection for ESP crypto functions
- One of the proposed in R1
The HIP base exchange
Packet I2

- Header
- Parameters
  ESP_INFO,
  [ R1_COUNTER, ]
  SOLUTION,
  DIFFIE_HELLMAN,
  HIP_TRANSFORM,
  ESP_TRANSFORM,
  ENCRYPTED{HOST_ID},
  [ ECHO_RESPONSE, ]
  HMAC,
  HIP_SIGNATURE_2

- Encrypted Initiator’s HI
- HOST_ID TLV in ”Encrypted data” field
The HIP base exchange
Packet I2

- Header
- Parameters
  ESP_INFO,
  [ R1_COUNTER, ]
  SOLUTION,
  DIFFIE_HELLMAN,
  HIP_TRANSFORM,
  ESP_TRANSFORM,
  ENCRYPTED{HOST_ID},
  [ ECHO_RESPONSE, ]
  HMAC,
  HIP_SIGNATURE_2

- If echo_request was present in R1, it must be echoed in a response
The HIP base exchange
Packet I2

- Header
- Parameters

ESP_INFO,
[ R1_COUNTER, ]
SOLUTION,
DIFFIE_HELLMAN,
HIP_TRANSFORM,
ESP_TRANSFORM,
ENCRYPTED{HOST_ID},
[ ECHO_RESPONSE, ]
HMAC,
HIP_SIGNATURE_2

- HMAC is calculated over the HIP packet, excluding
  - checksum (zeroed)
  - TLVs following HMAC
The HIP base exchange

Packet I2

- **Header**
- **Parameters**

  ESP_INFO,  
  [ R1_COUNTER, ]  
  SOLUTION,  
  DIFFIE_HELLMAN,  
  HIP_TRANSFORM,  
  ESP_TRANSFORM,  
  ENCRYPTED{HOST_ID},  
  [ ECHO_RESPONSE, ]  
  HMAC,  
  HIP_SIGNATURE_2

- Signature calculated over the HIP packet, excluding  
  - Initiator’s HIT  
  - checksum field  
  - PUZZLE: opaque and #I  
  - Any TLV after the signature TLV
The HIP base exchange
Finalizing connection setup

Initiator (IN)

DNS

Internet

Responder (RN)

R2: $SPI_{\text{Responder}}$

- Verify puzzle
- Generate keying material
- Select ESP SPI
The HIP base exchange
Packet R2

- Header
- Parameters
  - ESP_INFO,
  - HMAC_2,
  - HIP_SIGNATURE

- Keymat index: tells the point where keys are drawn from the keying material (zero in Base exchange)
- Old SPI (zero in Base exchange)
- New SPI; Responder’s ESP SPI
The HIP base exchange
Packet R2

- Header
- Parameters
  ESP_INFO,
  HMAC_2,
  HIP_SIGNATURE

- HMAC over the HIP packet plus an additional senders HOST_ID, and excluding
  - checksum (zeroed)
  - TLVs following HMAC_2
The HIP base exchange
Packet R2

- Header
- Parameters
  ESP_INFO
  HMAC_2
  HIP_SIGNATURE

- signature over the HIP packet, excluding
  - checksum (zeroed)
  - Any TLVs following the signature
The HIP base exchange

Initiator (IN)  

Internet  

Responder (RN)  

DNS  

ESP Security Association
Keying material

Keying material generation

- Diffie-Hellman shared secret: $K_{ij}$

$$KEYMAT = K_1 | K_2 | K_3 | ...$$

where

- $K_1 = SHA-1( K_{ij} | \text{sort}(HIT-I | HIT-R) | 0x01 )$
- $K_2 = SHA-1( K_{ij} | K_1 | 0x02 )$
- $K_3 = SHA-1( K_{ij} | K_2 | 0x03 )$
- ...
- $K_{255} = SHA-1( K_{ij} | K_{254} | 0xff )$
- $K_{256} = SHA-1( K_{ij} | K_{255} | 0x00 )$
- etc.
Keying material

Usage

- **Draw HIP keys**
  - HIP encryption  Direction 1 *)
  - HIP integrity (HMAC)  Direction 1 *)
  - HIP encryption  Direction 2 *)
  - HIP integrity (HMAC)  Direction 2 *)

- **Draw ESP keys**
  - ESP encryption  Direction 1 *)
  - ESP authentication  Direction 1 *)
  - ESP encryption  Direction 2 *)
  - ESP authentication  Direction 2 *)

- Keysizes are natural sizes for used algorithms
- *) Depends on the numeric value comparison of HITs
Mobility and Multihoming

Host mobility and micromobility
draft-ietf-hip-mm-01
HIP features

Mobility

- Connections bound to constant Host Identities (HIs)
  - Mobile host ➔ new locator (IP address)
    same connection endpoint (HI)
  - Connections don’t break
    - Peer host informed of new locator (IP addr.)
  - Mobility between IPv4 and IPv6 is supported
HIP: UPDATE
For updating location information

- Header
- Parameters

SEQ, ACK, ESP_INFO, DIFFIE_HELLMAN, LOCATOR, ECHO_REQUEST/ ECHO_RESPONSE, HMAC, HIP_SIGNATURE_2

- UPDATE sequence number
- Updated by one for each new UPDATE
- Scope: only the current HIP association
HIP: UPDATE
For updating location information

- Header
- Parameters
  SEQ, ACK, ESP_INFO, DIFFIE_HELLMAN, LOCATOR, ECHO_REQUEST/ ECHO_RESPONSE, HMAC, HIP_SIGNATURE_2

- One or more ACK parameters
- Acks received Update IDs
HIP: UPDATE
For updating location information

- Header
- Parameters
  SEQ,
  ACK,
  ESP_INFO,
  DIFFIE_HELLMAN,
  LOCATOR,
  ECHO_REQUEST/
  ECHO_RESPONSE,
  HMAC,
  HIP_SIGNATURE_2

- Keymat Index: tells the point from where keys are drawn from the keying material (zero in base exchange)
- Old SPI (zero in base exchange)
- New SPI; Initiator’s ESP SPI
HIP: UPDATE
For updating location information

- **Header**

- **Parameters**
  - SEQ,
  - ACK,
  - ESP_INFO,
  - DIFFIE_HELLMAN,
  - LOCATOR,
  - ECHO_REQUEST/
    ECHO_RESPONSE,
  - HMAC,
  - HIP_SIGNATURE_2

- public Diffie-Hellman key value
  - needed if rekeying requested
HIP: UPDATE
For updating location information

- **Header**
- **Parameters**

  SEQ,
  ACK,
  ESP_INFO,
  DIFFIE_HELLMAN,
  LOCATOR,
  ECHO_REQUEST/
  ECHO_RESPONSE,
  HMAC,
  HIP_SIGNATURE_2

- Traffic type: signaling / user data
- Locator type: IPv6, IPv4 in IPv6 ...
- Locator is the new address
- Can be multiple LOCATORs
HIP: UPDATE
For updating location information

- Header
- Parameters
  SEQ,
  ACK,
  ESP_INFO,
  DIFFIE_HELLMAN,
  LOCATOR,
  [ ECHO_REQUEST/
    ECHO_RESPONSE, ]
  HMAC,
  HIP_SIGNATURE_2

- Opaque data to be echoed back
- Optional address check
HIP: UPDATE
For updating location information

- **Header**
- **Parameters**
  - SEQ,
  - ACK,
  - ESP_INFO,
  - DIFFIE_HELLMAN,
  - LOCATOR,
  - ECHO_REQUEST/
  - ECHO_RESPONSE,
  - HMAC,
  - HIP_SIGNATURE_2

- Response to the echo_request
HIP: UPDATE
For updating location information

- Header
- Parameters
  SEQ,
  ACK,
  ESP_INFO,
  DIFFIE_HELLMAN,
  LOCATOR,
  ECHO_REQUEST/
  ECHO_RESPONSE,
  HMAC,
  HIP_SIGNATURE_2

- HMAC is calculated over the HIP packet, excluding
  - checksum (zeroed)
  - TLVs following HMAC
HIP: UPDATE
For updating location information

- Header
- Parameters
  SEQ,
  ACK,
  ESP_INFO,
  DIFFIE_HELLMAN,
  LOCATOR,
  ECHO_REQUEST/
  ECHO_RESPONSE,
  HMAC,
  HIP_SIGNATURE_2

- Signature calculated over the HIP packet
Location update
Without rekeying

**MN changes location**

UPDATE (LOCATOR, ESP_INFO SEQ, HMAC, SIG)

HIP association established

Internet
Location update
Without rekeying

UPDATE (ESP_INFO, SEQ, ACK, SIG, ECHO_REQUEST)
Location update
Without rekeying

UPDATE (ACK, HMAC, SIG, ECHO_RESPONSE)

MN

Internet

CN
Prevention against attacks

- HIP prevents against
  - impersonation attacks
    - HMAC – quick and cheap verification
    - SIGNATURE
  - third party DoS attack
    - Optional address check – the host is where it is supposed to be
Multihoming

- The presented mechanism can be used to ADD addresses
- Challenges
  - source and destination address selection
  - load balancing
  - need to mesh SAs to avoid replay window problems
  - IPsec SAs are symmetrically setup, but asymmetrical groups of addresses between hosts are possible
  - updating keying material for a subset of SAs
- Future work: needed policies and procedures
Mobility and multihoming

- Work going on
  - Integration with link layer
    - detect link layer changes, signal to HIP layer
    - DNA Working Group
  - integration with transport layer
    - TCP congestion control
    - draft swami tcp lmr 05.txt
  - PFKEY specifications
    - extensions to PFKEY to support SA movement
    - MOBIKE Working Group
Mobility and Multihoming

Network mobility
Mobile Networks

Nodes in the Architecture

- Correspondent Node (CN)
- Rendezvous Server (RS)
- Access Router (AR)
- Mobile Router (MR)

Mobile network

Nested mobile network

IP address visible to CN
Scenario #1
MN communicates with CN
Scenario #1
MR makes handoff

- MN moves together with the Mobile Network to a new topological location.
- The end-to-end connection between MN and CN will break.
Scenario #1
MR informs MN about the movement
Scenario #1
MN informs CN about the movement
Scenario #1
CN verifies MN's location
Scenario #1
Signalling explosion

- MR’s hand-off results in signalling explosion in highly populated mobile networks.
Scenario #2
Tunnel packets via MR's Rendezvous Server

- Hide network mobility from MN by tunnelling packets via MR's rendezvous server
- MR's rendezvous server assigns a prefix for MNs.
- This results in suboptimal triangular routing
- Difficult to implement nested mobile networks
Scenario #3
HIP NEMO Approach

- MR hides network mobility from MN
- MN Authorizes MR to signal on behalf of it
- **Public-key based end-point identifiers (HIs)** can be used to authorize and delegate signalling rights between entities.
- Possible to use certificates, e.g., X509v3 or SPKI certificates.
Scenario #3
Step 1: MR works as a HIP NAT device

- MN registers its HI and local unicast address to MR.
- MN learns MR's HI during the registration.
- MR implements HI multiplexed IP address translation.

MR = HIP NAT
Host Identifier registration
Private address space
Internet

© Ericsson AB 2004
Scenario #3
Step 2: MN Authorizes MR

- MN authorizes MR to send IP address update messages on behalf of it to the peers.
Scenario #3
Step 3: MR Delegates Rights to Signalling Proxy

- MR may delegate the signalling rights to a trusted signalling proxy.
Scenario #3
Optimizing MR->CN Signalling

- The signalling proxy sends address updates on behalf of the MN to CNs.
Scenario #3
Challenge-Response Tests Flood the Radio Link

- The peer nodes must verify that the MN is in the location where the signalling proxy claims the MN to be.
Scenario #3
Signalling Proxy = HIP NAT

- The on-path signalling proxy may hide the location of the mobile network.

Diagram:
- RS-of-MN
- CN (2)
- Internet
- BU signalling from Sig. Proxy
- Single BU from MR to Sig. Proxy
- Signalling proxy = HIPNAT
- AR
- MR
- MN
Scenario #3
Signalling Proxy = Mobile Anchor Point

- The signalling proxy may support regional mobility and act as an on-path Mobile Anchor Point (MAP).
Scenario #3
Optimizing CN- to MR Signalling

- The micro-mobility support minimizes the amount of over-the-air signalling
Many Roles of the Mobile Router

- Access router (AR)
- HIP NAT
- Mobility Anchor Point (MAP)
- Signalling proxy
MR - conclusions

- The Ericsson's secure HIP-NEMO solution is based on the public-key based Host Identifiers (HIs).
- HIs are used to delegate signalling rights between entities in the architecture to save radio resources and to optimize routing between the end-points.
  - Optimized radio link signalling inside a mobile network, and between the mobile network and the core network.
  - Optimized routing between MN and its peers.
  - No tunnelling required – better with nested mobile networks.
Additional functionality

Service registration: draft koponen hip registration 00
Registration

Protocol that is used to register with services
- Server provides service (e.g. rendezvous, PATH)
- Client registers as a service user
Additional functionality

Rendezvous extensions: draft-ietf-hip-rvs-01
DNS: draft-ietf-hip-dns-01
DNS based initial rendezvous

- Keys and/or HITs in DNS
- IP addresses in DNS
- PKI or DNSSEC needed to secure binding from DNS names to the keys

- DNS does not support mobility well
  - caching
  - DYNDNS is not considered to be fast enough
  - Simultaneous movement problem
- ➔ A separate Rendezvous point is needed
Rendezvous server

- DNS contains the location of the *Rendezvous server* (*RVS*)
- *RVS* has the current location of the MN
- MN constantly updates the location information to the RVS
- Three modes of operation specified (at the moment):
  - I1_REWRITE
  - I1_TUNNEL
  - BIDIRECTIONAL
Rendezvous server

1. Base exchange (1st time only)
2. Registration (1st time only)
3. Location update (after movement)
Rendezvous server

1. Query MN’s address
2. Get: HI_{MN}, IP_{RVS}
Rendezvous server

2. Rewrite destination IP address, forward I1 to HIT$_{MN}$, IP$_{MN}$

1. I1 to HIT$_{MN}$, IP$_{RVS}$

If necessary, rewrite source IP and add FROM parameter
Rendezvous server

MN

RVS

DNS

MN: HI, IP_{MN}

MN: HI, IP_{RVS}

Internet

R1, I2, R2 directly between hosts

CN
DNS modifications

We need support for maintaining HIs in the DNS

Two new DNS RRs defined

- HIPHI
  - Stores HI and/or HIT
  - Related to public key field of an IPSECKEY RR

- HIPRVS
  - HIP node’s Rendezvous Servers FQDN or IP address
  - Related to gateway field of an IPSECKEY RR
HIT based rendezvous

- HIT 128 bits long *random* number
  - Decentralized lookup based on random identifier is hard
- => DHT based overlay like i3
  - based on DHT
  - Hi3
Additional functionality

NAT traversal
NAT traversal

- NATs
- Legacy NAT traversal
- Advanced HIP aware NAT traversal
NATs general

- Four types of NATs
  - RFC3489
- Often used to give Internet access to multiple hosts using single public IP address
- Rewriting the source address and port can break end-to-end connectivity
NAT problems

- Without an open pin hole, no incoming connections
- Applications may be broken
  - IPs and ports used in payloads
  - When using separate control and data plane connections
  - Applications require knowledge about the address
  - IP fragmentation
- Integrity protection may be broken if IP addresses are changed
HIP and NAT

- HIP should make NAT traversal easier
  - We don’t care much about IP addresses
- HIP base exchange and NAT
  - NATs usually don’t pass proto 99 ➔ we use HIP over UDP
  - IPv6: NATs are not a problem (at least yet)
- HIP data exchange and NAT
  - In HIP checksums are calculated using HITs
  - SPI values may collide
  - ➔ Encapsulate IPsec over UDP (RFC 3498)
Legacy NAT traversal

draft nikander  hip  path  00

HIP host A 192.168.0.1

HIP host B 128.42.86.101

NAT 24.21.100.213

192.168.0.1:2157 == 24.21.100.213:1934

HIP PATH

Base exchange and registration - HIP over UDP

Pin-hole opened in NAT

Required for incoming connections

Host A is at 24.21.100.213:1934
Legacy NAT traversal

draft nikander  hip  path  00

HIP host A 192.168.0.1

HIP host B 128.42.86.101

HIP PATH

server

Host A is at 24.21.100.213:1934

192.168.0.1:2157 == 24.21.100.213:1934

NAT 24.21.100.213

192.168.0.1:2157 == 24.21.100.213:1934

I1, via PATH server (RVS)
Legacy NAT traversal

draft nikand h path 00

HIP host A 192.168.0.1

HIP host B 128.42.86.101

R1, I2, R2 directly

New pin-hole opened when R1 was sent to Host B

Host A is at 24.21.100.213:1934

HIP PATH

server

NAT 24.21.100.213

192.168.0.1:2158 == 24.21.100.213:1935
192.168.0.1:2157 == 24.21.100.213:1934

Host A is at 24.21.100.213:1934

HIP host B 128.42.86.101

Internet
Legacy NAT traversal

draft nikander hip path 00

HIP host A 192.168.0.1

HIP host B 128.42.86.101

HIP PATH

Host A is at 24.21.100.213:1934

ESP SA

NAT
24.21.100.213

Internet

192.168.0.1:2158 == 24.21.100.213:1935
192.168.0.1:2157 == 24.21.100.213:1934
HIP aware NATs

- inspect base exchange, get HITs and SPI pairs
  - SPIs in I2, R2
  - reads UPDATE messages for information update
  - recalculates checksums

- SPINAT – SPI based NATting
  - Change addresses based on SPIs in ESP header

- draft stiemerling-hipnat-0.txt
- draft tschofenig hiprg-hipnatfw-traversal-0.txt
Hi3

Internet Indirection Infrastructure and HIP
I3 – Distributed Hash Tables

- DHT - distributed directory for flat data
- Different ways to implement (DKS, Chord, ........)
- Each i3 server maintains part of the data
  - Lookups based on an identifier
  - Query goes from server to server until the id is found

- Used to create overlay networks
Insert trigger: I want to received data with ID

ID: Address

Distributed Hash Table
Internet Indirection Infrastructure

Server

Data to ID

Sending data to ID

ID: Address

Distributed Hash Table

i3

Client

Data forwarded based on trigger
Hi3

What is it?

- Combination of HIP and i3
- i3 overlay for signalling
  - Routes only HIP packets
- User data normally using ESP SA
- During registration, server can give R1s to the Hi3
  - Hi3 can respond with R1 on behalf of Server
  - Hi3 can block false I2 messages (wrong solution)
  - At R2, the infra can control FW to open data hole for ESP
- Hi3 as a rendezvous point for hosts
Hi3
Connection setup

I1 (to HIT_s)
R1

Hi3

I2 (to HIT_s)
R2

I2 (to HIT_s, IP_s)

Client
ESP SA for user data traffic

Server
HIT_s
Hi3
Protection against attacks

- Enhanced DoS protection
  - caching R1s and replying to I1s
  - Pre-verification of puzzle solution
  - Integration of firewalls
    - Hi3 can select the used fw

- Attacker gets server’s IP only after full Base Exchange
  - if FW NATs, the attacker gets only IP of FW
  - Under attack, FW can change the used interface

- How to protect DHT:DHT based load balancing
  (multiple mappings)
Standardization
Standardization

- IETF: HIP Working Group
  - Targetted to be ready in summer 2005 (→ fall?)
  - Experimental RFCs
    - Base HIP
    - ESP usage with HIP
    - Mobility and Multihoming
    - DNS
    - Rendezvous

- IETF: HIP Research Group
  - More research oriented discussion about locator / identity split
Evolution of drafts: Early era
Evolution of drafts: Restart

- mos-hip-arch-03
  - Apr 2003
  - mos-hip-06
    - May 2003
    - mos-hip-arch-03
      - Apr 2003
    - mos-hip-06
      - May 2003

- mos-hip-arch-03
  - Sep 2003
  - mos-hip-06
    - Feb 2004

- mos-hip-arch-03
  - Jun 2004
  - mos-hip-06
    - May 2003

- mos-hip-arch-03
  - Oct 2004
  - mos-hip-06
    - May 2003

- mos-hip-arch-03
  - ietf-hip-arch-00
    - Oct 2004
  - mos-hip-06
    - ietf-hip-base-00
      - May 2003
    - ietf-hip-base-01
      - Oct 2004

- mos-hip-arch-03
  - ietf-hip-mm-00
    - Oct 2004
  - mos-hip-06
    - ietf-hip-mm-00
      - May 2003
    - ietf-hip-mm-00
      - Oct 2004

- mos-hip-arch-03
  - ietf-hip-mm-00
    - Oct 2004
  - mos-hip-06
    - ietf-hip-mm-00
      - May 2003
    - ietf-hip-mm-00
      - Oct 2004

- mos-hip-arch-03
  - ietf-hip-dns-00
    - Oct 2004
  - mos-hip-06
    - ietf-hip-dns-00
      - May 2004
    - ietf-hip-dns-00
      - Oct 2004

- mos-hip-arch-03
  - ietf-hip-rvs-00
    - Oct 2004
  - mos-hip-06
    - ietf-hip-rvs-00
      - Jul 2004
    - ietf-hip-rvs-00
      - Oct 2004
Evolution of drafts: Current status

  - Architecture

- ietf-hip-base-01 (Oct 2004)
  - ietf-hip-base-02 (Feb 2005)
  - jok-hip-esp-00 (Feb 2005)
  - Base exchange
  - Using ESP

- ietf-hip-mm-00 (Oct 2004)
  - ietf-hip-mm-01 (Feb 2005)
  - Mobility & Multihoming
  - DNS

- ietf-hip-dns-00 (Oct 2004)
  - ietf-hip-dns-01 (Feb 2005)
  - Rendezvous

- ietf-hip-rvs-00 (Oct 2004)
  - ietf-hip-rvs-01 (Feb 2005)
  - kup-hip-reg-00 (Feb 2005)
  - Registration
HIP implementation on FreeBSD
Implementation

- FreeBDS 5.3
- Source code available at: [http://hip4inter.net](http://hip4inter.net)
  - HIP base exchange
  - Mobility and multihoming
  - v4/v6 interoperability
- Implements currently
  - draft-ietf-hip-base-01
  - draft-ietf-hip-mm-00
- Under 20,000 LoC
Architecture

[Diagram showing the architecture with various components including Application, DNS server, DNS library, HIP daemon, Socket API, IPsec SPD, and IPsec SAD.]
Architecture
Architecture

Context – HIP association
- HI – IP address mappings
- Keying material
- Security Associations
- Security Policies
Architecture

States
- HIP association state machine
- Incoming HIP packet handling
  - Packet validation
  - Notifications on errors
  - State transitions
- Timers
Architecture

Protocol
- Packet generation
  - Outgoing HIP packets (PFHIP)
  - Resolver (PFres)
  - Administrative tool (PFadmin)
  - IPsec SPD/SAD (PFkey)
- Incoming packets: callbacks to STATES
- HIP TLV handling
Architecture

PFHIP
- HIP packet communication between the kernel and user space
- Incoming packets: callbacks to PROTOCOL
Architecture

PFadmin
• Administrative interface for the command-line tool
Architecture

PFres
• Interface to the resolver library
Architecture

**PFkey**
- Communication to IPsec SPD and SAD
- PF_KEY Key Management API, Version 2 (RFC 2367)
Architecture

HIPD_IO
- Abstraction for socket input/output functions
- Handles creation, reading, writing, and closing of sockets
Architecture

Cryptographic functions
- Cookie generation
- Diffie-Hellman calculations
- RSA/DSA signature and HMAC generation and verification
- Encryption/decryption of HIP TLVs
Architecture

Debug
• Debugging functions
  • Current state
  • Keying material
  • Packets from IO
  • ...

HIP daemon
**Architecture**

**HIT resolver**
- Resolves the HI based on FQDN
- Sends FQDN, HI, and IP addresses to the HIP daemon
- In case of IPv4, waits for the LSI from the HIP daemon
LSI-HIT hash table
- Maps the LSI to the corresponding HIT for BEET processing
Architecture

**BEET mode**
- Encrypts / decrypts user data
- Converts HITs to IP addresses
- Converts IP addresses to HITs based on SPIs
- Has tunnel-mode semantics but transport-mode format
Links

- HIP WG supplemental page
  - http://hip.piuha.net
- Ericsson Research: HIP for BSD project
  - http://www.hip4inter.net