Introduction to Host Identity Protocol (HIP) and its Applications

Tutorial at 2nd AsiaFI School

Andrei Gurtov

Helsinki Institute for Information Technology

Slides jointly with Ekaterina Vorobyeva
http://www.hiit.fi/~gurtov

January 2009

Outline [1]

- Introduction to HIP architecture
- Background on network security
- The HIP architecture
- Base protocol
- Main extensions
- Advanced extensions
- Performance measurements
- Lightweight HIP

[1] ©Andrei Gurtov, 2008. Figures from Host Identity Protocol (HIP): Towards the Secure Mobile Internet, Andrei Gurtov, 2008, ©John Wiley & Sons Limited. Reproduced with permission.

Outline (cont.)

- Middlebox traversal
- Name resolution
- Micromobility
- Communication privacy
- Possible HIP applications
- API
- HIP with other protocols
- Implementations

Reading material

- A. Gurtov, Host Identity Protocol (HIP): Towards the Secure Mobile Internet, ISBN 978-0-470-99790-1, Wiley and Sons, June 2008. (Hardcover, 320 p).
- Jokela P, Moskowitz R and Nikander P 2008 Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP). RFC 5202
- Kent S 2005a IP Authentication Header. RFC 4302 (Proposed Standard)
- Kent S 2005b IP Encapsulating Security Payload (ESP). RFC 4303 (Proposed Standard)
- Kent S and Seo K 2005 Security Architecture for the Internet Protocol. RFC 4301 (Proposed Standard)

Reading material (cont.)

- Krawczyk H, Bellare M and Canetti R 1997 HMAC: Keyed-Hashing for Message Authentication. RFC 2104 (Informational).
- Laganier J and Eggert L 2008 Host Identity Protocol (HIP) Rendezvous Extension. RFC 5204
- Laganier J, Koponen T and Eggert L 2008 Host Identity Protocol (HIP) Registration Extension. RFC 5203.
- Manral V 2007 Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH). RFC 4835 (Proposed Standard).
- Moskowitz R and Nikander P 2006 Host Identity Protocol Architecture. RFC 4423, IETF.

Reading material (cont.)

- Moskowitz R, Nikander P, Jokela P and Henderson T 2008 Host Identity Protocol. RFC 5201.
- Nikander P, Henderson T, Vogt C and Arkko J 2008 End-Host Mobility and Multihoming with the Host Identity Protocol. RFC 5206
- Nikander P and Laganier J 2008 Host Identity Protocol (HIP) Domain Name System (DNS) Extension. RFC 5205.
- Nikander P, Laganier J and Dupont F 2007b An IPv6 prefix for overlay routable cryptographic hash identifiers (ORCHID). RFC 4843, IETF.
- Orman H 1998 The OAKLEY key determination protocol. IETF RFC 2412
- Rivest RL 1992 The MD5 message digest algorithm. RFC 1321

Reading material (cont.)

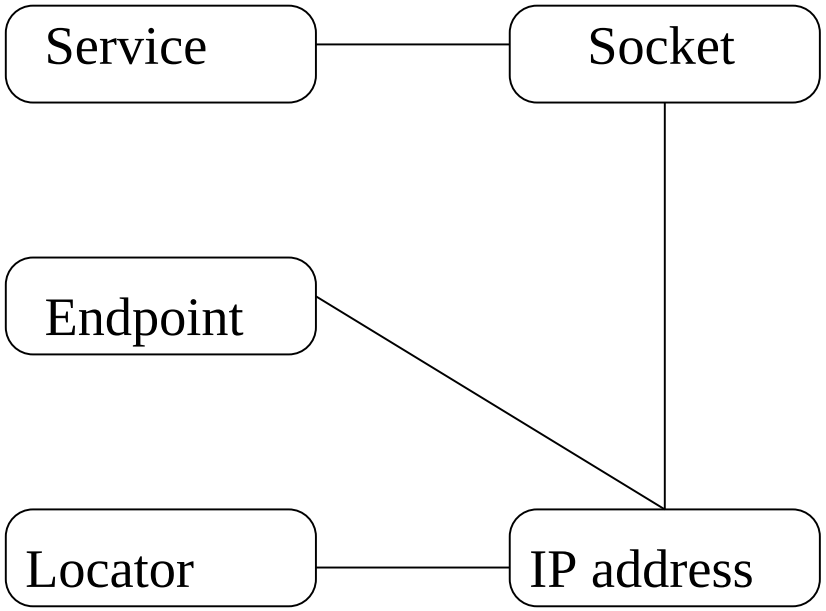
- Rosenberg J, Weinberger J, Huitema C and Mahy R 2003 STUN: Simple traversal of user datagram protocol (UDP) through network address translators (NATs). RFC 3489, IETF
- Saltzer JH 1993 On the naming and binding of network destinations in local computer networks. RFC 1498, IETF.
- Stiemerling M, Quittek J and Eggert L 2008 NAT and Firewall Traversal Issues of Host Identity Protocol (HIP) Communication. RFC 5207.
- Kivinen T and Kojo M 2003 More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE). RFC 3526
- Kaufman C 2005 Internet key exchange (IKEv2) protocol. RFC 4306, IETF.

Overview

Identifier-locator split

- Network prefixes of IP addresses
  - IP addresses are located in a close geographical area
- The role of host identifier (e.g. DNS)
- Dual role of IP addresses
  - identifying function of IP addresses
  - locating function of IP addresses

Location and identity of hosts are combined in the Internet

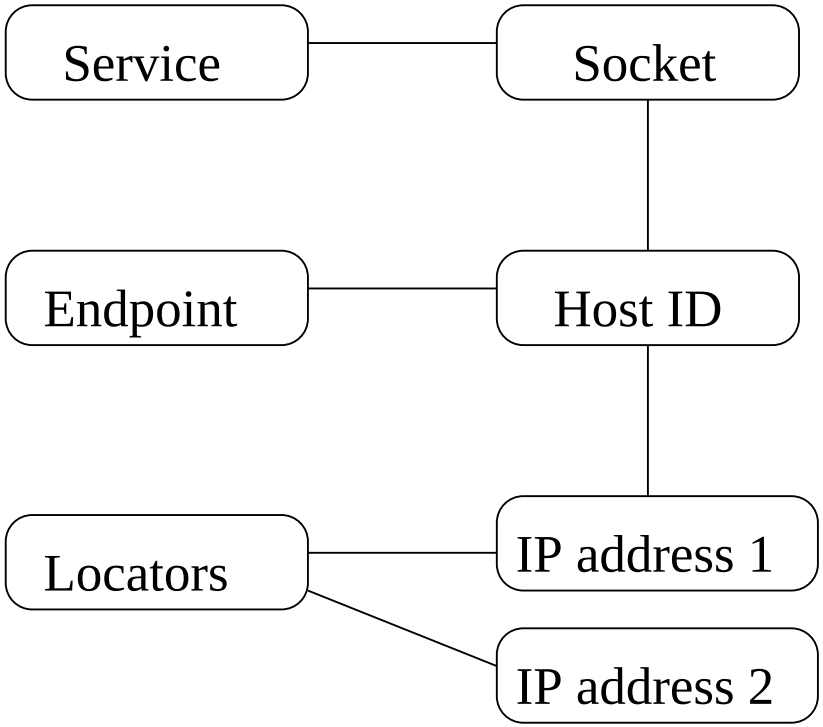


- the role of IP as identifier and locators are still mixed
- separate service uses own socket
- the endpoint identity is attached to the IP address

Identifier-locator split (cont.)

- HIP splits host identifier and locator
- A security mechanism is essential to prove the identity
  - a long randomly generating string - not sufficient in a public Internet
  - a self-generated public-private key pair as the host identity
- Host identity separates socket and network interfaces
  - several locators can be associated with one identity
  - a single host can have multiply identities
  - group host identities (in the research phase)

Separating location and identity of Internet hosts

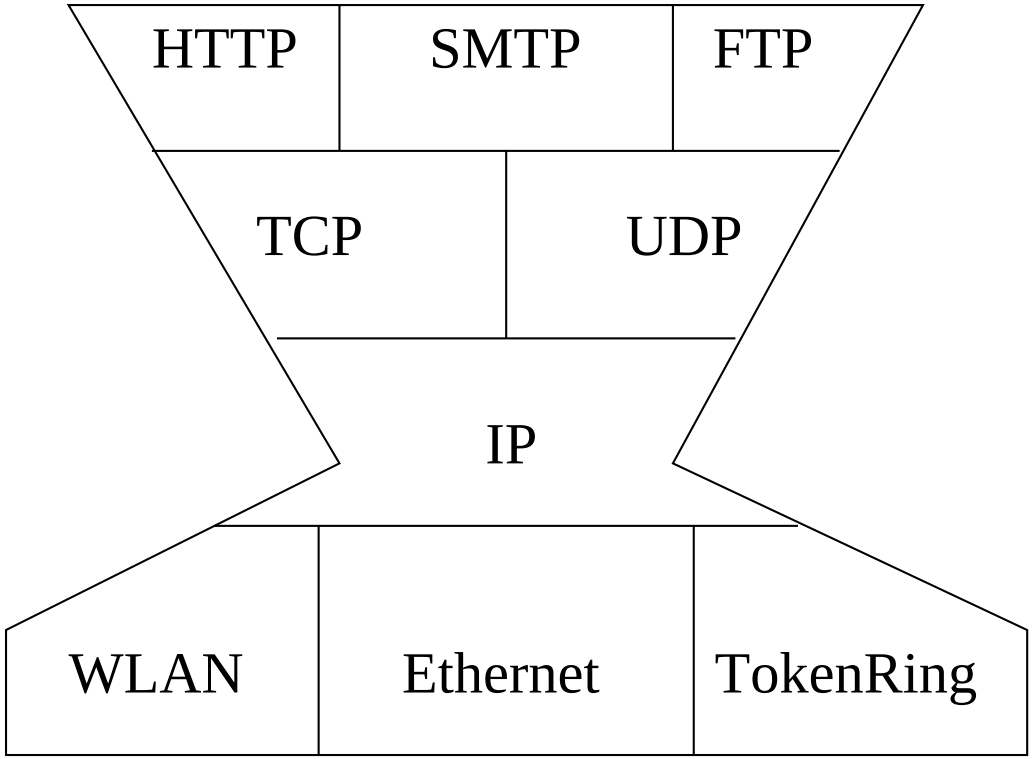


- the positioning of host identity between socket and network interfaces
- the sockets are bound to the host identity instead of a locator

HIP in the Internet Architecture

- IP protocol - the only routable network-layer protocol in use
- IP protocol is able to run over a wide range of link technologies
  - Ethernet
  - Wireless LAN
  - Token Ring
- Multiple transport protocols can run on top of IP
  - TCP and UDP
- The large number of application uses the transport protocol
  - HTTP
  - SMTP
  - FTP

IP as a waist of the Internet protocol stack

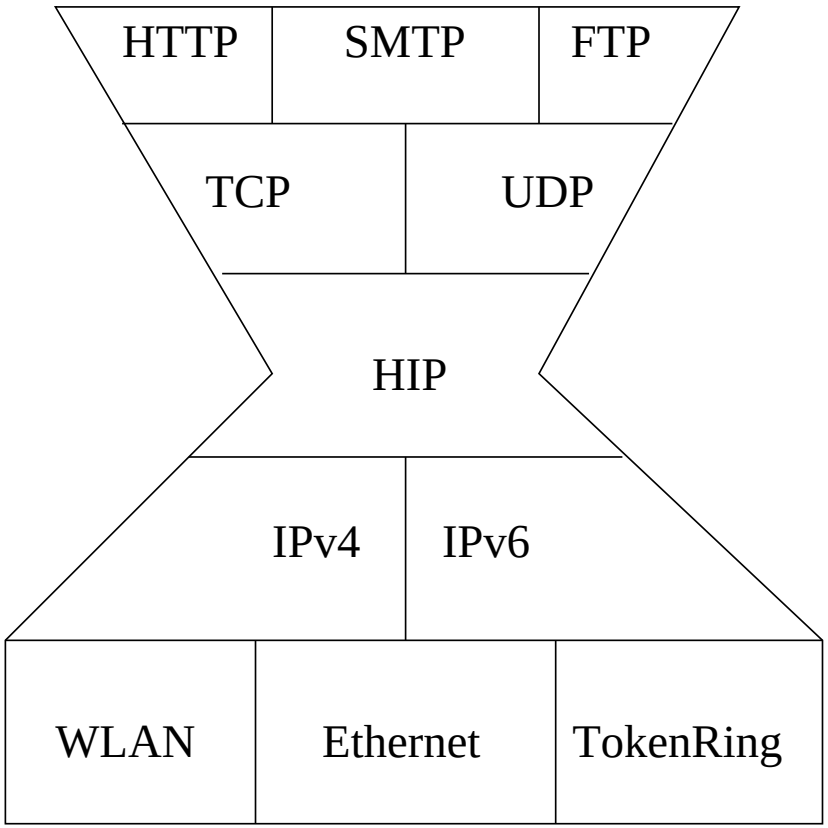


- IP - narrowest part of the stack
- waist of the Internet

HIP in the Internet Architecture (cont.)

- A major problem in the original Internet architecture
  - tight coupling between networking and transport layers (e.g., TCP checksum calculation)
  - impossible independent evolution of two layers
- Introduction of a new networking or transport protocol requires changes to other layers
- The dramatic growth of the Internet scale (introduction of IPv6)
- Unfeasible deployment of a new IP version with a flag day
- The necessity of simultaneous routing of both IP protocol versions
- HIP architecture can restore the original Internet hourglass model

HIP as a new waist of the Internet protocol stack

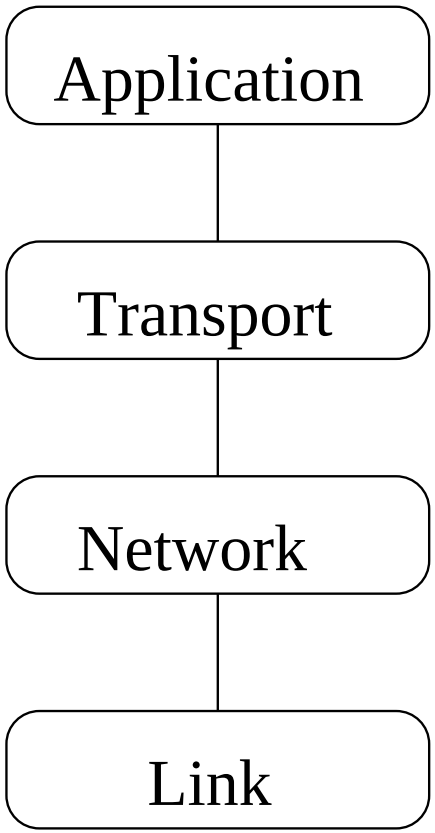


- HIP replaces IPv4 in its role
- IPv4 and IPv6 run underneath HIP
- transport protocols on top of HIP

HIP in the Internet Architecture (cont.)

- The problem of Denial-of-Service (DoS) attacks
  - server creates a significant state during establishment of a TCP connection after replying to a SYN packet
  - there is no assurance that the SYN has arrived from the genuine host
  - moderate number of host can swamp the server with SYN messages
- HIP prevents creating the state before the client is verified
- By means of cryptographic puzzles HIP prevents the client generating connection attempts at an overly fast rate
  - puzzle offers a client to reverse a hash function that requires significant computational resources. Verifying the puzzle at the server is a short operation

The IP protocol stack



IP address, port

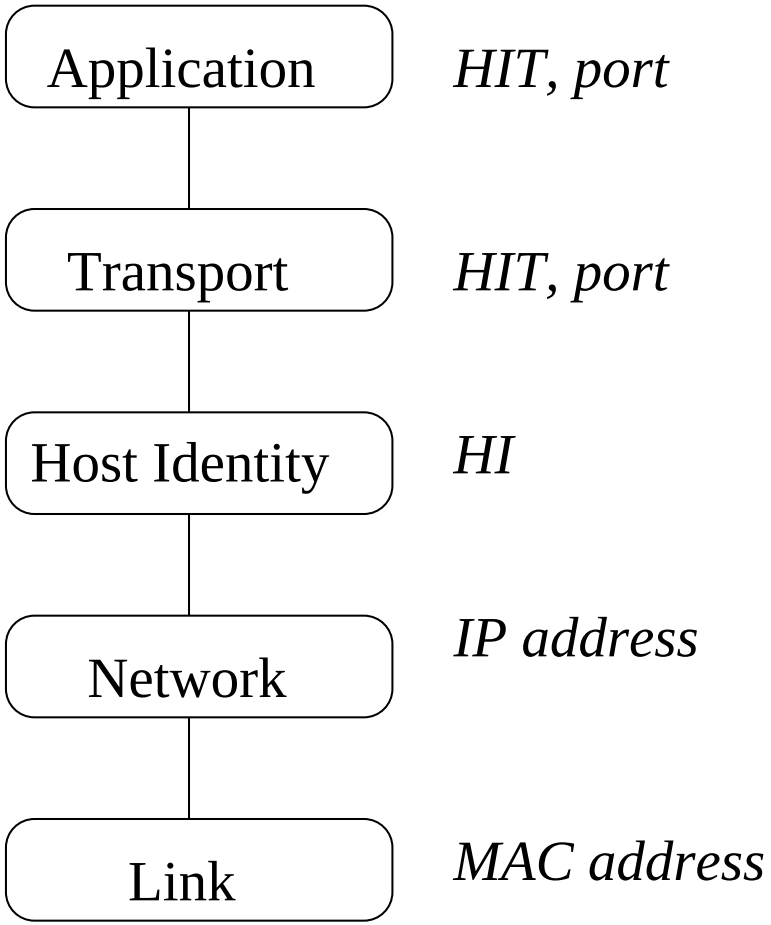
IP address, port

IP address

MAC address

- a Berkley socket binds to the IP address and transport protocol family
- the state created at a transport layer uses the IP and transport protocol port number to deliver data to a correct application
- the network layer uses the destination IP to determine a right transmission link
- the Network Interface Card (NIC) address is added on the link

The protocol stack of HIP



- HIP - a sub-layer between the network and transport layers
- the application and transport protocol use the host identity tag (HIT) in their messages
- HIP sub-layer maps HITs to the IP address before passing a packet to the networking layer
- transmission of the packet then follows the same pattern as in a plain IP stack

Brief history of HIP

- The problem of naming hosts and data in the Internet
  - RFC1498 from 1993 reprints the paper on naming from 1982
  - resource name, address, and route
  - services and users, network nodes, network attachment points, and paths
  - three bindings of a service to node, a node to attachment point, and an attachment point to a route
- Name Space Research Group (NSRG) - in IRTF from 1993 to 2003
  - other namespaces than the 32-bit IPv4 addresses
  - Robert Moskowitz from ICSA, Inc - the original inventor of HIP

Brief history of HIP (cont.)

- The draft *moskowitz-hip-00* is an individual submission in the IETF, May 1999
- From 1999 to 2002, R.Moskowitz has held informal meetings during the IETFs
- Several revisions of the HIP architecture and protocol specifications were published as individual submissions
- In 2002, Pekka Nikander became interested in HIP and took over the leading of the standardization effort from R.Moskowitz
- New packet structure, the state machine and the protocol details were developed together with Ericsson NomadicLab, Boeing, and HIIT
- The specifications were published as individual submissions until 2004

Brief history of HIP (cont.)

- In June 2004 an IETF working group on HIP was created and *draft-ietf-hip-base-00* was published
  - the HIP WG is chaired by David Ward (Cisco) and Gonzalo Camarillo (Ericsson)
  - the purpose was ”to define the minimal elements that are needed for HIP experimentation on a wide scale”
- First outcome of the group - overview of HIP architecture
  - the HIP BE and ESP encapsulation specifications
  - mobility and multihoming extensions
  - DNS and RVS, and registration extensions

Brief history of HIP (cont.)

- In late 2006, NAT traversal, the application support and native API - as WG items
- In 2004, HIP RG was chartered at the Internet Research Task Force (IRTF)
- In 2005, Andrei Gurtov (HIIT) replaced Pekka Nikander
- The task of HIP RG
  - evaluation of the impact of wider HIP deployment on the Internet
  - development of experimental protocol extensions that are not yet ready for standardization in the IETF

Introduction to network security [2]

[2] Based on work contributed by Tobias Heer, RWTH.

Goals of cryptographic protocols

- Authentication
  - determination of the message origin
- Authorization
  - only authorized network entities have an access to restricted resources, data, services
- Accountability
  - identifying the user of a service unambiguously in order to account for that service

Goals of cryptographic protocols (cont.)

- Data integrity
  - the contents of the message is not altered
- Confidentiality
  - data protection from unauthorized access
  - not necessarily restricted to the protection of data
  - meta-information about the communication entities
- Reliability
  - a host that provides services should not be vulnerable to attack

Goals of cryptographic protocols (cont.)

- Non-repudiation
  - a network entity or user should not be able to falsely deny its participation in communication
- Privacy
  - the identity of a network entity or user should not be revealed to unauthorized parties
- Consistency
  - two honest hosts establish a communication context hosts should have a consistent view of the parties involved in the communication process

Basics and terminology

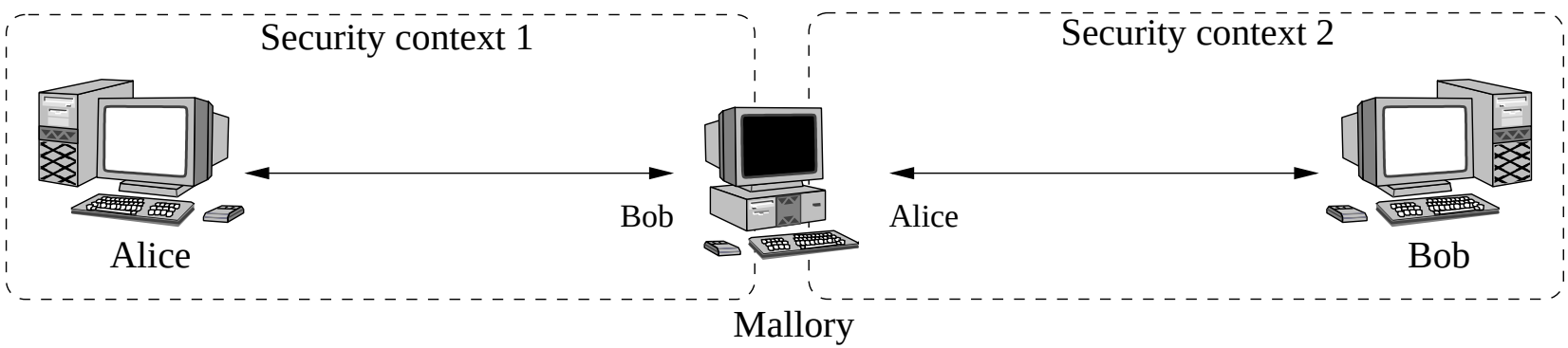
- *Peers* two or more hosts that exchange data
- *A* or *Alice* and *B* or *Bob* two communicating hosts
- *Mallory* an attacking party
- *Security context* the set of information that is required to apply security messages
- *Ciphers* algorithms that encrypt and decrypt data
- *Plaintext* the unprotected text
- *Ciphertext* the encrypted text

Attack types

- Eavesdropping
  - the process of overhearing a private communication
  - an attacker can messages that a group of peers exchange
  - simple - on unprotected wireless communication channels
  - difficult - on wired communication channels
  - compromising the confidentiality of data
  - data encryption - the most common way to deal with attacks
- Impersonation
  - an attacker aims at maliciously pretending to represent another host
  - undermining a range of security goals: authentication, authorization, non-repudiation, accountability, data integrity, confidentiality

Attack types (cont.)

- Man-In-The-Middle attacks
  - a special form of impersonation attack
  - an attacker is on the network path between two hosts
  - the attacker can delay, modify or drop packets
  - a one way to compromise public-key cryptography



Attack types (cont.)

- Delay and replay attacks
  - the attacker uses validly encrypted or integrity protected data in a fraudulent way to subvert a communication protocol
  - the goal of a replay attack tricking the receiver of the replays into triggering duplicate transactions
- Denial-of-Service attacks
  - an attack targets hosts that provide some sort of service
  - the goal of DoS consumption of the victim resources to an extent that it is not able to provide any service to legitimate host
  - aiming the security goal of reliability

Attack types (cont.)

- Exhaustive key space search - *brute force* attack
  - attacks that try to break the ciphers employed by the protocol
  - finding the secret key - the natural way of breaking the protection
  - breaking a key-based algorithm - trying different keys
  - using large key-spaces - an enormous computational cost for an exhaustive key search
- Cryptoanalysis
  - deciphering a message without knowledge of the encryption key
  - mathematical findings and shortcuts - breaking or decrease of the security of a cipher
  - the side channel attack - exploit of weaknesses in the physical implementation of a cipher

Defense mechanisms. Symmetric cryptography

- The same key for encryption and decryption
  - requirement to keep the key secret
  - all communication peers must be in possession of the same secret key
  - widely used cryptography less CPU-demanding than public-key cryptography
- Block ciphers
  - encryption of plaintext blocks of a fixed length into identical sized blocks of ciphertext
- Stream ciphers
  - encryption of a plaintext stream symbol by symbol

Defense mechanisms. Symmetric cryptography (cont.)

- The Advanced Encryption Standard: AES
  - the official successor of DES
  - Federal Information Processing Standard approved encryption algorithm (Standard 2001)
  - the operations are applied in several rounds
  - operating on a fixed block length; allowing 128-, 192-, 256-bit keys
  - the number of rounds (10, 12, 14) is determined by the key length
- Key exchange for symmetric cryptography
  - utilizing public-key cryptography to securely agree on shared secrets
  - the Diffie-Hellman key exchange can be used

Defense mechanisms. Public-key cryptography

- Encryption and decryption keys are related in a non-trivial way
  - the encryption key publicly accessible as a public key
  - the decryption key kept secret as a private key
  - encrypted data can be sent to any host without exchanging shared secret
- Easy, hard, feasible and infeasible
  - trapdoor functions
  - easy calculation
  - hard to reverse

Defense mechanisms. Public-key cryptography (cont.)

- Properties of trapdoor functions
  - must be easy to compute - the encryption is computationally feasible
  - must be hard to reverse the without the possession of additional information
  - it must be computationally hard for an attacker to calculate the secret
- Diffie-Hellman key exchange
  - the first PK cryptosystem was published by W.Diffie and M.Hellman
  - secure exchange of symmetric keys over insecure channel
  - the discrete logarithm problem

Defense mechanisms. Public-key cryptography (cont.)

- The key exchange
  - two communication parties agree on *p* and *g*
  - only few combination of *p* and *g* are in use
  - the DH groups are publicly defined in protocol specification
  - the DH consists of four steps
    - the Initiator selects a number $x \in \{1, ..., p-2\}$ and calculates $g^x mod\, p$; g^x is sent to the Responder, x is kept secret
    - the Responder selects a number $y \in \{1, ..., p-2\}$ and calculates $g^y mod\, p$ and $k = g^{x^y} mod\, p = g^{xy} mod\, p$; k is the shared secret
    - the Responder sends back g^y as its public key
    - the Initiator calculates $k = g^{y^x} mod\, p = g^{y^x} mod\, p = g^{x^y} mod\, p$

Defense mechanisms. Public-key cryptography (cont.)

- The RSA algorithm
  - PK algorithm published by Rivest, Shamir, and Adelman
  - suitable for encryption and message authentication
  - factorizing product of two large prime numbers into the two original prime factors is computationally hard
- RSA key generation
  - find two large prime numbers p and q of similar size
  - calculate $n = pq$ and $\varphi(n) = (p-1)(q-1)$; φ is Euler's φ function
  - choose a random number e, $1 < e < \varphi(n)$ with $gsd(e, \varphi(n)) = 1$
  - calculate $d = e^{-1} \bmod \varphi(n)$ by using the extended Euclidean algorithm

Defense mechanisms. Public-key cryptography (cont.)

- RSA encryption
  - looking up the public key of a peer $k_{pub} = (n, e)$
  - dividing a message into pieces and transforming each into a number $m \in Zn$
  - calculating the cipher text $c = m^e \bmod n$
- RSA decryption
  - repeats the encryption process with d used instead of e
  - $c^d \bmod n = (m^e \bmod n)^d \bmod n = m$

Defense mechanisms. Public-key cryptography (cont.)

- RSA signatures
  - signature process is similar to the encryption process
  - the signature process is performed with the private key of the sender
  - the signature verification is performed with the sender's public key
  - CPU-intensive exponentiation - transmitting short messages
- DSA signatures
  - specified to be used with the Digital Signature Standard (DSS)
  - a variety of the El Gamal signature scheme
  - the discrete logarithm problem
  - the properties, application, and performance are similar to RSA

Defense mechanisms. One-way cryptographic hash function

- A third class of cryptographic mechanism
  - second preimage resistant - for a given x it is computationally hard to find $x' \neq x$ with $H(x') = H(x)$
  - collision resistant - it is computationally hard to find any two values x and x' with $x \neq x'$ and $H(x) = H(x')$
  - preimage resistant - for a given $z = H(x)$ it is computationally hard to find a preimage x' with $H(x') = z$
- Two ways of designing cryptographic hash functions
  - the same algebraic problems as PK cryptosystems
  - block-cipher-based hash functions

Defense mechanisms. One-way cryptographic hash function (cont.)

- Message digests
  - hash functions are often used for message authentication
  - production of a short message digest of fixed length
- Hashed message authentication codes: HMAC
  - signing messages - all peers must be in possession of a shared secret
  - $MAC(m) = H(k \| m)$
  - $HMAC_k(m) = H((k \oplus opad) \| H((k \oplus ipad) \| m))$
  - generation and verification is very cheap in terms of CPU cycles

Defense mechanisms. One-way cryptographic hash function (cont.)

- Hash chains
  - in 1981, Leslie Lamport - the use of chains of hashes
  - basic idea - the iterated application of a cryptographic hash function on some random or pseudo random seed value r
  - $h_0 = r$; $h_n = H(h_n - 1); = H(H(H(...H(r)...)))$ n times
  - $(h_i, h_{i-1}, h_{i-2}, ..., h1, r)$ - the hash chain; h_i - the hash chain anchor
  - it is easy to verify that h_i belongs to the same hash chain as h_{i-1}
  - it is computationally hard to find h_{i-1} if only h_i is given
  - given h_{i+1}, it is hard to find an h' with $H(h') = H(h_i) = H_{i+1}$
  - given h_i, it is possible to verify that h_{i-n} is part of the same hash chain if $0 < n \leq i$

Defense mechanisms. One-time signatures

- Lamport - one-time signatures based on hash functions
  - $H(r)$ and $H(r')$ are published as a public key
  - a one-bit message can be signed by publishing r as signature s if the bit is 1 and r' otherwise
  - comparing the $H(s)$ with $H(r)$ and $H(r')$ - verification
  - the signature can be used to sign $l - bit$ messages by using $2l$ hashes
- An improvement of the scheme - Merkle 1988
  - reduction of the number of hashes to $l + [log_2(l)] + 1$ and the average number of hashes per signature to $(l + [log_2(l)] + 1)/2$
- Winternitz - a scheme reduces the signature size
- Zhang - two protocols for signing routing messages

Defense mechanisms. Sequence numbers

- A simple measure to avoid replaying, reordering or dropping packets
  - a monotonically increasing number is assigned to every message
  - the receiver can determine if a message has already been processed
  - messages without the next expected number - a duplicate message or a part of a replay attack
- The decision - a message is a part of an attack
  - a strict matching - all preceding packets must have been received before a packet with a higher sequence number is accepted or processed
  - a relaxed matching - hosts can use a window of legitimate sequence numbers

Defense mechanisms. Cryptographic nonces

- A nonce - a number or a string that is used only once
  - verifying the reachability of a host
  - defense against replay attacks
  - request-response mechanisms
  - proof of the secret possession
- The nonce is only used once
  - prevention of replay attacks - nonce cannot be reused by an attacker
- Nonce creation
  - can be derived from random or pseudo-random numbers
  - can include information that the host wants to embed

Defense mechanisms. Client puzzles

- A host is allowed to artificially generate peer's CPU load
  - a correct solution of the puzzle is required
  - mechanism is made less attractive for attackers
- The puzzle is easy to verify while difficult to solve
  - puzzles are based on cryptographic hash functions
  - a hash function is applied to the concatenation of a server-generated nonce and varying client-chosen value
  - the peer is required to find j for which the result $H(i|j)$ has k bits with the value 0 as lowest-order bits
  - k defines puzzle difficulty

Security protocols

- Some or all security goals can be achieved
  - security protocols employ cryptographic algorithms
  - specifying how communication parties must act
- Security context
  - parameters for the employed security mechanisms: the choice of mechanism, keys, the state of communication peers
- An establishment phase
  - all communicating peers agree on a set of algorithms and the keys
  - the life cycle ends - the closing of the security association and the deletion of the security context

Security protocols. MODP Diffie-Hellman groups

- Successful generation of the secret
  - the same modular exponential groups must be used
  - the number of known groups is specified (RFC2412, RFC3526)
  - the prime numbers p of different groups vary from 768 to 8192 bits
- A 384-bit group is defined by Moskowitz
  - devices with few CPU resources is allowed to use HIP
  - the group is insecure
- The groups have been selected based on a certain pattern
  - the high order and low order 64 bits are forced to 1
  - the middle section of group - the binary expansion of $Pi(\pi)$

Security protocols. Keying material

- The DH key exchange generates keying material
  - the length of the keying material is greater than the length of the symmetric keys
  - single bits must be selected as a symmetric key
- IKE and HIP
  - a hash function is used to select bits
  - several keys can be derived
- Binding between the session keys and additional parameters of the key exchange
  - parameters can be included into the key derivation
  - concatenation of the DH shared secret with the additional parameters

Security protocols. Transforms

- A large variety of cryptographic algorithms
  - the cryptographic algorithms are interchangeable
  - different key lengths can be used for symmetric and asymmetric ciphers
- Keeping a security protocol suitable for many application scenarios
  - the choice of algorithms and key-sizes should be negotiated
- Negotiations during the handshake
  - transform parameter contains transform suites
  - transform suite - the combination of several algorithms and key-sizes
  - suites are identified by an index

Security protocols. Transforms (cont.)

- The order of the transform suites
  - the preferences of the host in decreasing order
  - the most preferred transform suite is listed first
  - a peer can choose any of the contained suites
- Hosts use the algorithms and key-lengths, indicated by the chosen transform suite

Security protocols. IP security architecture: IPsec

- The IP itself does not offer sufficient security
  - no defenses against eavesdroppers or attackers
- IPsec security architecture - RFC2401, 1998
  - authenticating the source
  - the integrity of IP packets
  - new header - unprotected IP traffic is distinguished from IPsec traffic

Security protocols. IP security architecture: IPsec (cont.)

- Security Associations
  - endpoints must agree on a set of algorithms
  - hosts can employ a range of protocols to negotiate
  - the shared security context - SA
  - IPsec SA - simplex connections that afford security services to the traffic carried by them (RFC4301)
  - two SAs - an incoming and an outgoing - are necessary to protect a duplex channel

Security protocols. IP security architecture: IPsec (cont.)

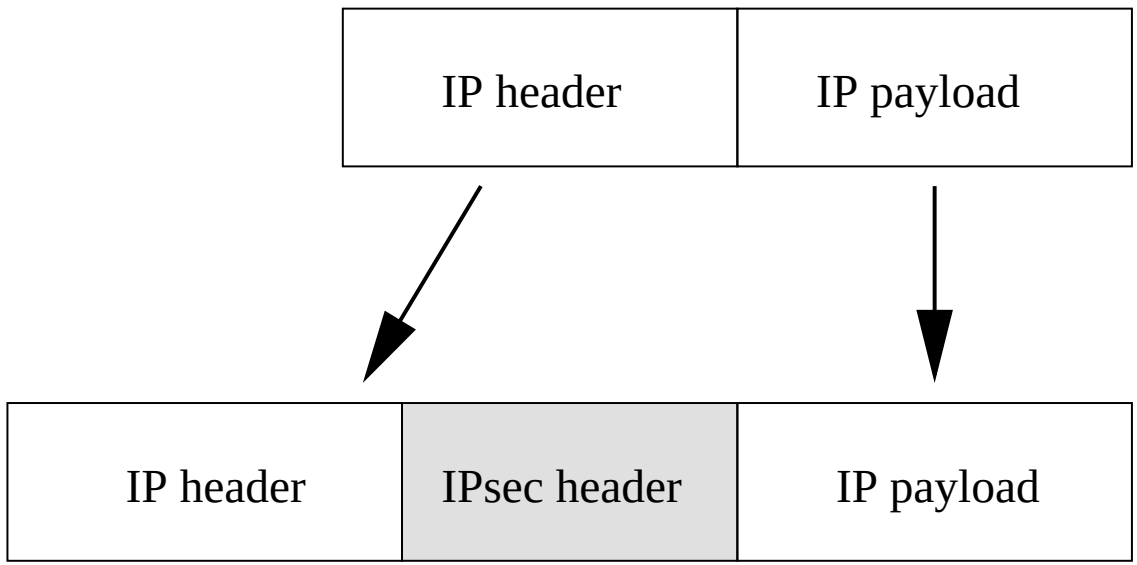
- Security Association Database (SAD)
  - information about established SA
  - two SAs for every duplex communication
  - information about the IPs of a distant host, the IPsec protocol
  - the contexts needed to process incoming and outgoing IP packets
- Security Policy Database (SPD)
  - information about the services that are offered by IP datagrams
  - kinds of security mechanisms are defined
  - traffic that stays unprotected and traffic that may not leave the host is defined

Security protocols. IP security architecture: IPsec (cont.)

- Security Parameter Index (SPI)
  - demultiplexing information in the IPsec header
  - identifying the right security context

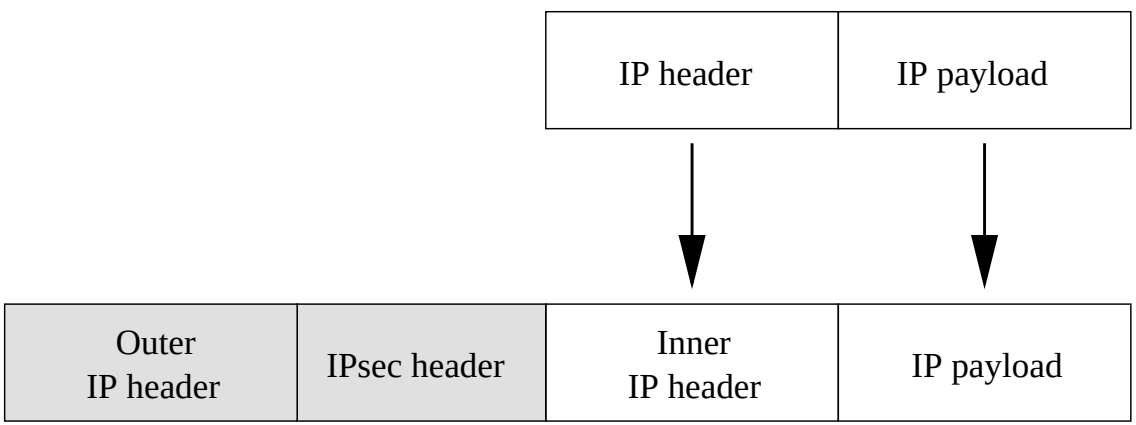
Security protocols. IPsec modes

- Transport mode
  - the mode is used when traffic between hosts has to be protected
  - the additional header between the IP and transport layer header
  - the header structure contains all information required to verify or decrypt the packet



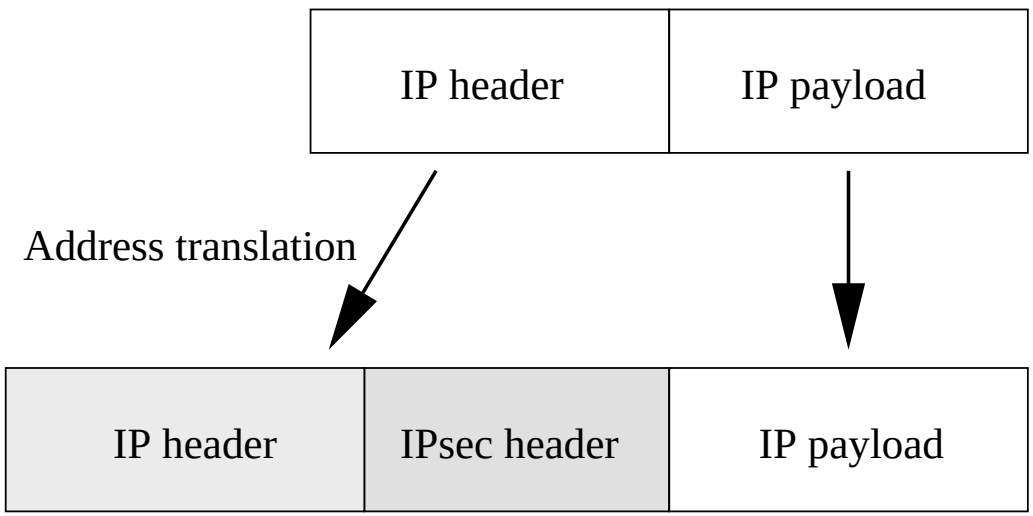
Security protocols. IPsec modes (cont.)

- Tunnel mode
  - the mode is used to securely connect networks
  - two intermediate gateways establish an IPsec association
  - the traffic is securely tunneled between the gateways
  - an additional IP header before the IPsec header



Security protocols. IPsec modes (cont.)

- Bound End-to-End Tunnel mode
  - semantics similar to the tunnel mode with a transport mode like packet format
  - IP packets are tunneled from one gateway to another
  - the outgoing gateway modifies the existing IP header
  - implementation of HIP payload encryption in a very efficient way



Security protocols. IPsec security protocols

- The Authentication Header (AH) protocol
  - ensuring of the authenticity of an IP packet and its source
  - IP packets must be integrity protected but not necessarily encrypted
  - checksums, based on cryptographic hashes, are employed
  - the IP packet payload and certain fields of IP header protection
  - unprotected fields - Time To Live, Header Checksum
- The Encapsulating Security Payload (ESP) protocol
  - protection of IP payload against eavesdroppers
  - symmetric cryptography is employed
  - the sender authentication
  - protection of a source address

Security protocols. IPsec security protocols (cont.)

- A sequence number field in the IPsec header
  - protection from replay attacks
  - a window of sequence numbers - replay protection window
  - the highest sequence number received by a host - the ”right” edge of the window
  - the range of valid numbers - the size of the window and highest number
  - the sequence number is lower than ”left” edge - packet is dropped

Security protocols. Internet Key Exchange: IKE

- The IKE protocol specifications
  - RFC2407, RFC2408, RFC2409
- IKEv2
  - simplification of the security mechanisms in IKE
  - some security loopholes and weaknesses are closed
- Two phases of the key negotiations in IKEv2
  - the first phase - a host verifies the identity of its peer, generates the keying material, establishes a secured channel: the IKE_SA
  - the second phase - new SAs (child SAs) are negotiated over the channel

Secure DNS

- The present DNS
  - insecure, susceptible to attacks
  - cache poisoning - a most common attack
- Secure DNS extensions (DNSSEC)
  - ensuring that the public key comes from a trusted DNS
  - important for HIP deployment
- Current DNSSEC specifications
  - RFC4033 - overview and requirements of the DNSSEC architecture
  - RFC4034 - new Resource Records (RR)
  - RFC4035 - the DNS protocol extensions

Secure DNS (cont.)

- The DNSSEC architecture
  - storing a DNS server's public key in a DNSKEY RR
  - the DNSSEC support is indicated in a query - the DNS signs replies with private key
  - the Delegation Signer (DS) parameter - the digest of the child DNS server public key
  - verifying the digest - a trust link from the parent DNS to the child DNS
  - authenticity of the reply - a chain of trust links up to the DNS server for which which the client has a trusted public key stored

DNSSEC Example

```
[gurtov@hippy ~]$ dig +dnssec www.se any
; <<>> DiG 9.3.2 <<>> +dnssec www.se any
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8237
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 3, ADDITIONAL: 2
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;www.se.                                IN      ANY
;; ANSWER SECTION:
www.se.                 3260    IN      A       212.247.204.135
www.se.                 6860    IN      RRSIG   NSEC 5 2 7200
20060821182406 20060816070558 54245 se.
Oex8Tj20u9Q9f7l5Idepb53acoFLbNvwX6xcNBpUezEnxKAze+0vrQvM
7i0X5QYq/EQ8EtPEhLH/SXjJS4+mmYXV3mNVdT96rYSeKX7oh+N82+VY
vCaKPxZLZy22JYhEPxs5CZxiSGEgPq3ncYw6+mhzKjxm3AA5XtpUhZ3B tW4=
www.se.                 6860    IN      NSEC    www-3.se. NS RRSIG NSEC
www.se.                 3260    IN      NS      ns.nic.se.
www.se.                 3260    IN      NS      ns2.nic.se.
```

```
www.se.                 3260    IN      NS      ns3.nic.se.
;; AUTHORITY SECTION:
www.se.                 3260    IN      NS      ns.nic.se.
www.se.                 3260    IN      NS      ns2.nic.se.
www.se.                 3260    IN      NS      ns3.nic.se.
;; ADDITIONAL SECTION:
ns3.nic.se.             17256   IN      A       212.247.3.80
;; Query time: 406 msec
;; SERVER: 193.210.18.18#53(193.210.18.18)
;; WHEN: Thu Aug 17 12:41:52 2006
;; MSG SIZE  rcvd: 358
[gurtov@hippy ~]$
```

Architectural overview

Internet namespaces

- Namespace allows uniquely identify an entity
- Two namespaces are globally deployed in the Internet:
- IP addresses
  - also serve as host locators
- DNS names
  - human-friendly host names
  - can be location independent (e.g. .*net* domain)
  - can be limited to a certain geographical area (e.g. .*fi* for Finland)

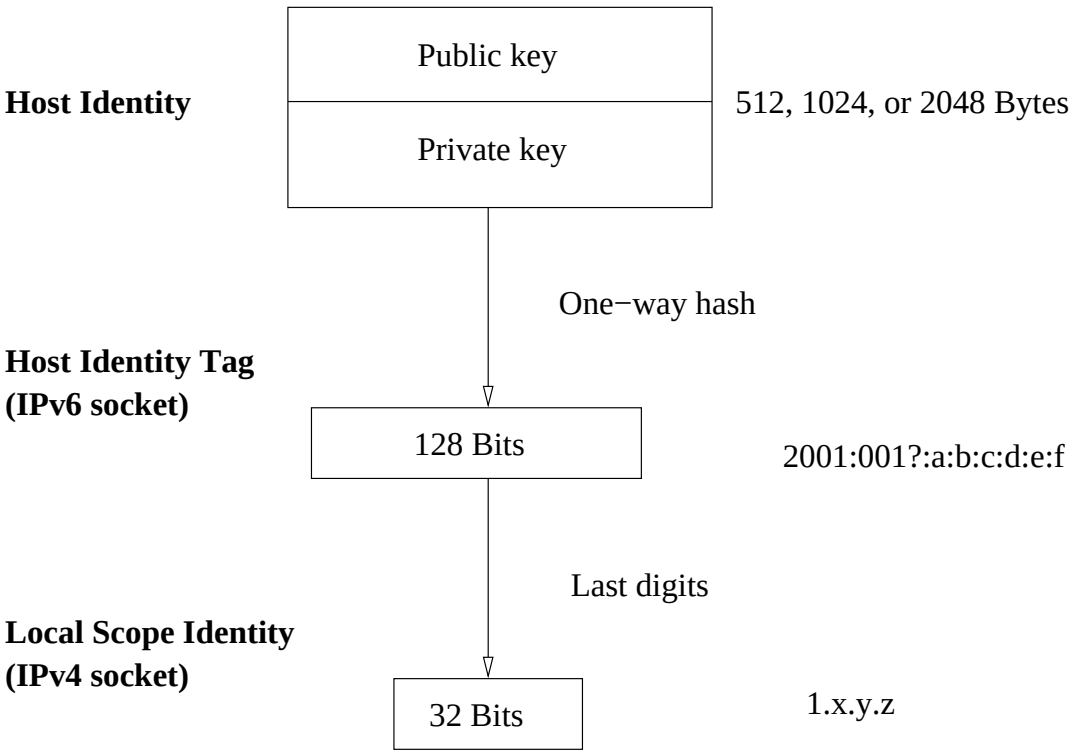
Internet namespaces (cont.)

- DNS namespace has several limitations
  - updating the IP in DNS can be too slow to support mobility
  - most hosts do not have modification access to the DNS servers
  - basic DNS service is not secure
  - many DNS names are bound to a specific organization or country
- Three shortcomings of current namespaces
  - changing the host address breaks transport layer connection
  - authentication of the host is not supported
  - privacy-preserving communication is not provided

Methods of identifying a host

- In HIP, a pair of self-generated public and private keys provides the HI
  - the length of the public key - 512, 1024 or 2048 bytes
  - RSA algorithm is used by default
  - support of the DSA algorithm
  - generation of the new keys is time-consuming operation
  - large and variable size of the public key makes it inconvenient for use
- Two additional forms of host identity: HIT and LSI

Methods of identifying a host (cont.)



- Host Identity Tag (HIT)
  - the same length as an IPv6
  - statistically unique
  - probability of collisions is negligible
  - has a prefix 2001:0010::/28
- Local Scope Identity (LSI)
  - probability of collisions is significant
  - has only local meaning
  - has a prefix 1

Overlay Routable Cryptographic Hash Identifiers

- IPv6 prefix for ORCHIDs reserves a part of the IPv6 address space to serve as identifiers in the socket API
- Internet Assigned Numbers Authority (IANA) allocated a prefix for ORCHIDs
- ORCHIDs appear as IPv6 addresses
- ORCHIDs are not routable at the IP layer, expected to be routable at the overlay layer on top of IP
- Application can transparently use ORCHIDs in place of IPv6 addresses

ORCHIDs. The purpose of an IPv6 prefix

- The goals
  - prevention of confusion with regular IPv6 addresses
  - ORCHIDs as identifiers in the legacy application APIs
  - possibility to experiment with new network architectures
  - support of several different protocols (HIP, MIP) with the same IPv6 prefix
- Properties of ORCHIDS
  - generating using the hash function - secure binding to the input parameters and statistical uniqueness
  - compatible with an IPv6 format

Generating and routing an ORCHID

- SHA1 hash over a 128-bit context ID concatenated with an input bitstring
  - bitstring must be statistically unique, can often be a public key
  - context ID - randomly generated value, defines the type of ORCHID
- The final ORCHID - concatenating an IANA allocated 28-bit prefix with a 100-bit bitstring extracted from the middle of the hash output
- Location-independent end-point identifier
- Can be routed on an overlay layer

ORCHIDs properties

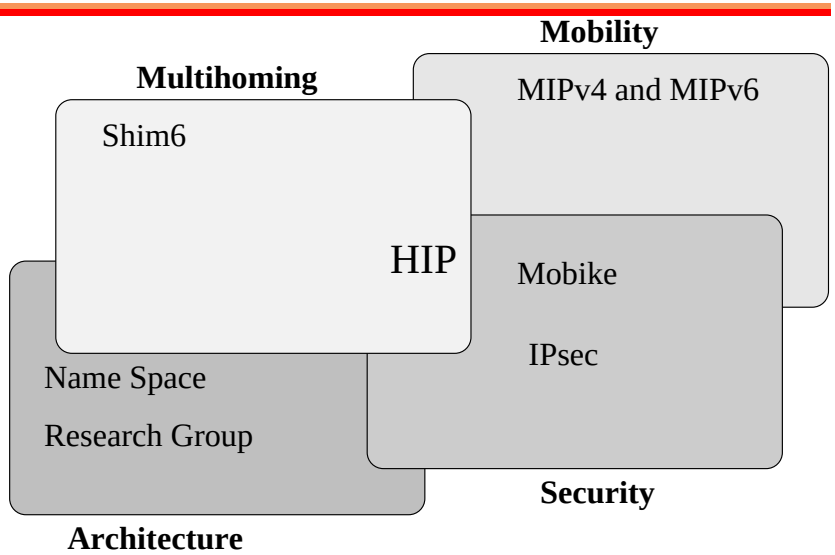
- Statistically unique
- Two types of collisions are theoretically possible
  - two host generated within the same context, but with different bitstring can collide
  - two ORCHIDs from different context can collide
- All contexts use the same hash function to generate an ORCHID

The role of IPSec

- ESP transport mode is used to carry HIP data packets
- HIP control messages
  - session key exchange - friendly with middleboxes
- Secure Parameter Index (SPI)
  - SPI in the packet identifies Security Association (SA)
  - SPI is mapped to HITs (HIT compression)
- There is no HIP-specific data packet format, but the standard IPSec mode is used
- Lightweight HIP

HIP relation to other IETF activities

- Mobility for IPv4 (mip4) and Mobility for IPv6 (mip6)
  - development of the Mobile IP protocol
  - adoption of IKEv2
- Better-Than-Nothing-Security (btns)
  - unauthenticated Security Associations
- Site Multihoming in IPv6 (multi6)
  - multihoming is currently implemented in IPv4

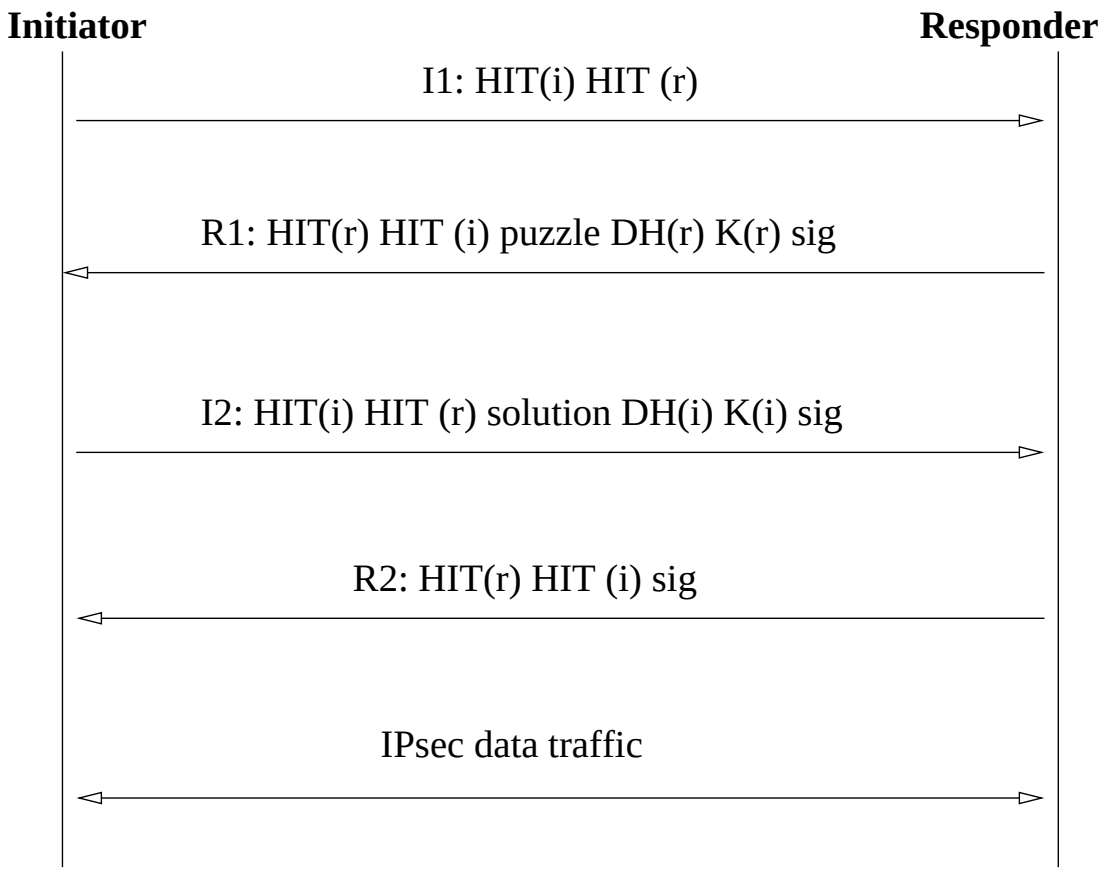


- Site Multihoming by IPv6 Intermediation (shim6)
  - shim layer between the IP and the transport layers
- IKEv2 Mobility and Multihoming Protocol (MOBIKE)
  - can be used for mobile VPN or site multihoming
- Mobility for IP: Performance, Signaling, and Handoff Optimization (mipshop)
  - Hierarchical Mobile IPv6 (HMIPv6), Fast Handovers for Mobile IPv6 (FMIPv6)

Base Protocol

The HIP base exchange protocol

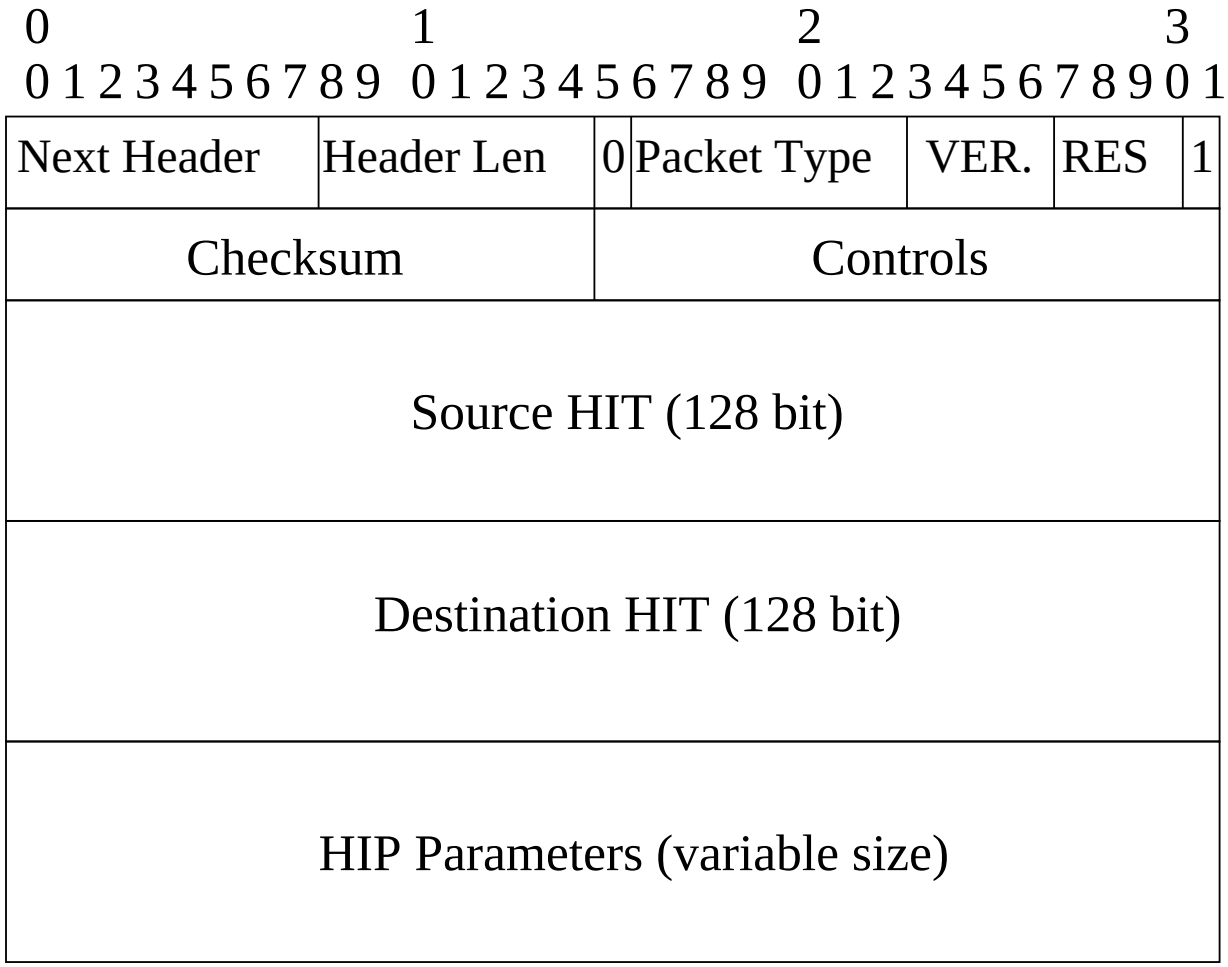
- HIP BE establishes SA between two hosts
- four-message exchange



Base Exchange

- HIP control packets
  - transmitted after a basic IPv4 or v6 header
  - protocol number assigned by IANA is 139 (early HIP - 253)
  - basic HIP header common for all HIP messages
  - HIP checksum is calculated over a pseudoheader including source and destination IP, HIP packet length, and protocol number

A general packet format of HIP messages



Base Exchange (cont.)

- I1 packet
  - starts the Base Exchange
  - contains the Initiator and the Responder HITs (can be obtained from the DNS)
  - the Responder HIT can be NULL - opportunistic mode
  - contains the basic header and no parameters
  - the only defined HIP control flag is the lowermost bit for Anonymous identifier

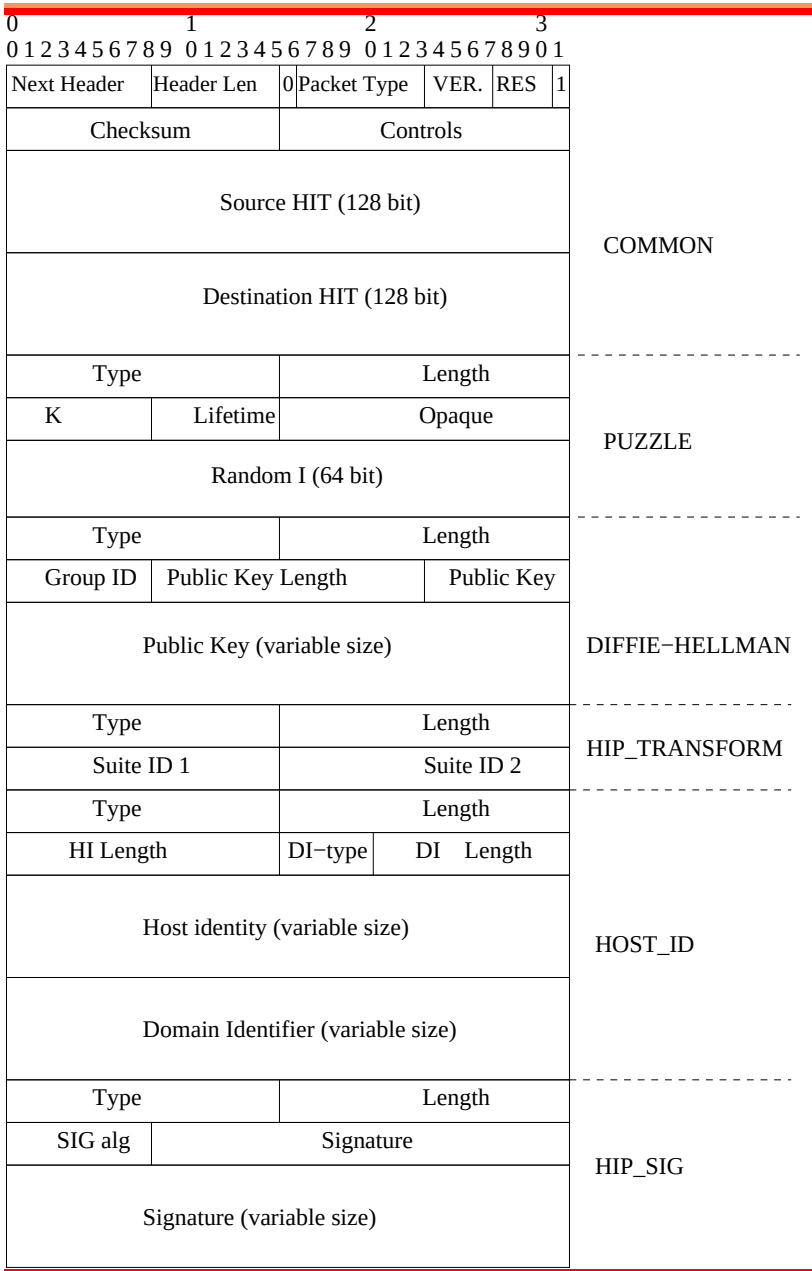
HIP I1 packet captured with Wireshark

```
Internet Protocol, Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 60
    Identification: 0x0000 (0)
    Flags: 0x04 (Don't Fragment)
    Fragment offset: 0
    Time to live: 63
    Protocol: Unknown (0xfd)
    Header checksum: 0x4087 [correct]
    Source: 193.167.187.106 (193.167.187.106)
    Destination: 193.167.187.132 (193.167.187.132)
Host Identity Protocol
    Payload Protocol: 59
    Header Length: 4
    Packet Type: 1
    Version: 1, Reserved: 1
    HIP Controls: 0x0000
        .... .... .... ..0. = Certificate (One or more CER packets follows)
        .... .... .... ...0 = Anonymous (Sender's HI is anonymous)
    Checksum: 0x6b06 (correct)
    Sender's HIT: 200100150A097C449CA1257176DD0872
    Receiver's HIT: 20010014766EFBEEF74DEC73D6C528C0
```

Base Exchange (cont.)

- R1 packet
  - upon receiving the I1 packet the Responder does not create a state
  - the Responder replies with an R1 packet
  - the packet can be pre-generated
  - contains a Diffie-Hellman key, a cryptographic puzzle, and a public key
  - signed by the Responder with its private key

Obligatory fields of HIP R1 packet



- critical bit in each parameter to indicate its importance
- if critical parameter is not recognized, the packet is not processed further - NOTIFY or ICMP message is sent to the peer host

Base Exchange (cont.)

- Cryptographic puzzle in R1
  - protects the Responder from DoS attacks
  - task to find value that produces zeros when passed through an SHA-1 has function
  - the number of zeros - puzzle difficulty
  - for the Responder the puzzle verification is fast hash computation
  - dynamical adjustment of the puzzle difficulty
  - lifetime - the number of seconds the Initiator may use for finding a solution

Base Exchange (cont.)

- Diffie-Hellman parameter in R1
  - contains the public key from a certain group
  - 384-bit and 1536-bit More Modular Exponential (MODP) groups
  - can include a maximum of two groups
  - the Initiator can select a stronger or weaker key
- HIP_TRANSFORM parameter in R1
  - a list of cryptographic algorithms supported by the Responder
  - up to six transform identifiers (Suite-IDs)

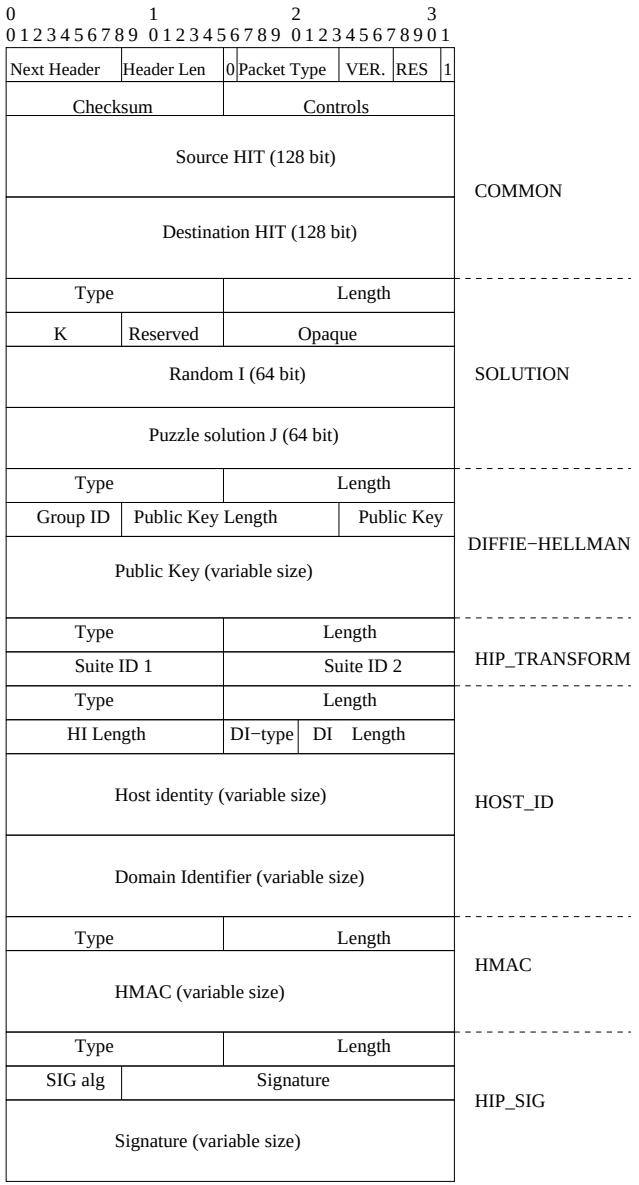
Base Exchange (cont.)

- HOST_ID parameter in R1
  - a host public key
  - a domain identifier (a Fully Qualified Domain Name or a Network Access Identifier)
- HIP_SIGNATURE parameter in R1
  - a signature
  - RSA or DSA algorithm is used to generate the signature
- ECHO_REQUEST parameter in R1
  - the Initiator should return unmodified in I2
- R1_COUNTER parameter in R1

Base Exchange (cont.)

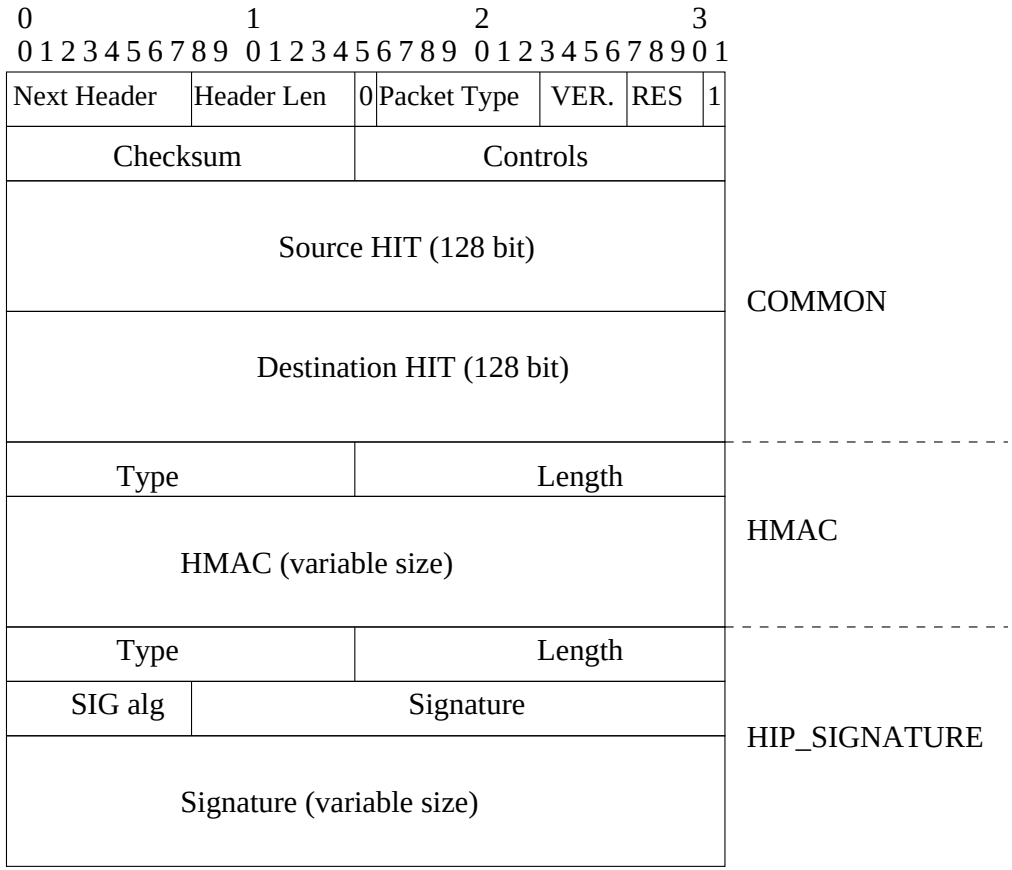
- I2 parameter
  - similar fields as R1
  - the puzzle parameter is replaced by the puzzle solution
  - an HMAC parameter
  - unmodified R1_COUNTER parameter
  - ECHO_REPLY parameter

Obligatory fields of HIP I2 packet



- puzzle solution
  - random number I and J value that should hash into K zero bits
- HMAC signature
  - calculated over the HIP packet
  - faster than a HIP signature

Base Exchange (cont.)



- Format of R2 packet
- HMAC parameter
- HIP_SIGNATURE parameter
- the data can flow on a HIP association after this packet
- possibility of transmitting upper-layer data in I2 and R2

Header of ICMP echo request packet after HIP BE

```
[Time delta from previous packet: 0.003767000 seconds]
Internet Protocol, Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0xa7 (DSCP 0x29: Unknown DSCP; ECN: 0x03)
    Total Length: 136
    Identification: 0x0000 (0)
    Flags: 0x04 (Don’t Fragment)
        0... = Reserved bit: Not set
        .1.. = Don’t fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 63
    Protocol: ESP (0x32)
    Header checksum: 0x405f [correct]
    Source: 193.167.187.106 (193.167.187.106)
    Destination: 193.167.187.132 (193.167.187.132)
Encapsulating Security Payload
    SPI: 0x940e5f06
    Sequence: 1
```

Other HIP control packets

- UPDATE packet
  - similar to an R2 format
  - SEQ parameter (incremented by one starting from zero)
  - ACK parameter (can include several sequence numbers)
  - HIP acknowledgments are not cumulative
- NOTIFY packet
  - informs the peer on protocol errors
  - the HIP host should not make any state changes
  - NOTIFY parameters, HIs, and HIP_SIGNATURE
  - errors types (INVALID_SYNTAX, CHECKSUM_FAILED, INVALID_HIT etc.)

Other HIP control packets (cont.)

- CLOSE packet
  - terminates HIP associations
  - ECHO_REQUEST, HMAC, HIP_SIGNATURE parameters
  - acknowledged by a CLOSE_ACK packet
- HIP-specific ICMP messages
  - indicates unsupported protocol version, invalid puzzle solution, non-existing HIP association, or malformed parameter
  - limited rate for transmitting
  - misuse by attackers
  - caution with incoming ICMP messages

IPsec encapsulation

- Security Associations establishment, one in each direction
- Data packets are authenticated and encrypted by ESP
- Symmetric keys agreement during the HIP BE

ESP transforms

- HIP ESP packets look the same as IPsec ESP packets
  - HIs are not transported in the HIP ESP packets
  - SPIs locate a correct HIP association
- A set of supported IPsec algorithms in the ESP_TRANSFORM in I2
- SPI value in the ESP_INFO in I2 and R2
- All HIP implementations support AES, HMAC-SHA1-1-96, ESP NULL with SHA-1 or MD5
- Two ways of ESP support implementation

Standard IPsec ESP transport mode

- Uses IP addresses in setting Security Parameter Database (SPD) and Security Association Database (SAD)
- Upper protocol layers and transmitted packets use the IP addresses
- HIP layer calculates and verifies upper-layer checksums using HITs
- SPI value
  - the Initiator determines the SPI to be used by the Responder in its outgoing SA
  - the Responder determines the SPI to be used by the Initiator in its outgoing SA
  - the SPI value is assigned randomly and changed for each new SA

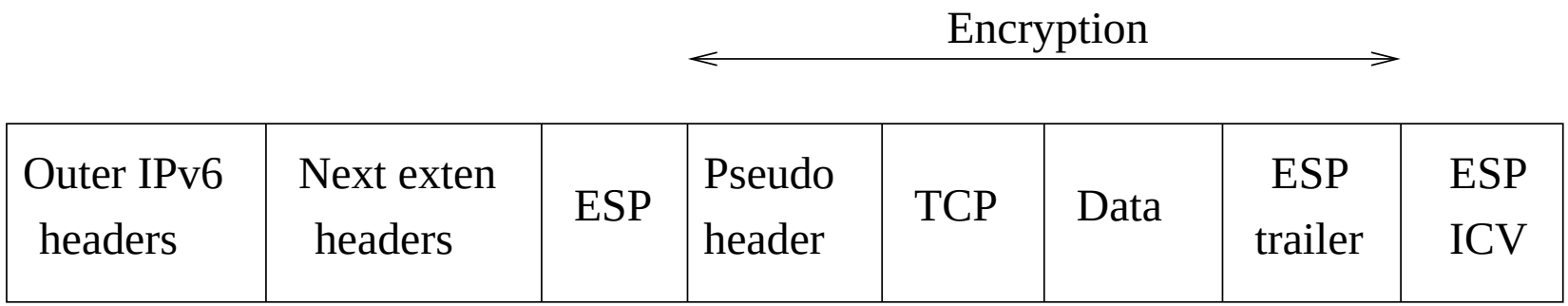
Standard IPsec ESP transport mode (cont.)

- Host cannot establish an SA
  - HIP_NOTIFY parameter informs the peer (e.g. NO_ESP_PROPOSAL_CHOSEN - no acceptable cryptographic algorithm to choose)
- Rekeying
  - creating a new pair of SAs and removing the old SAs
  - new SA is sent in the ESP_INFO parameter in an HIP UPDATE message

ESP Bound End-to-End Tunnel

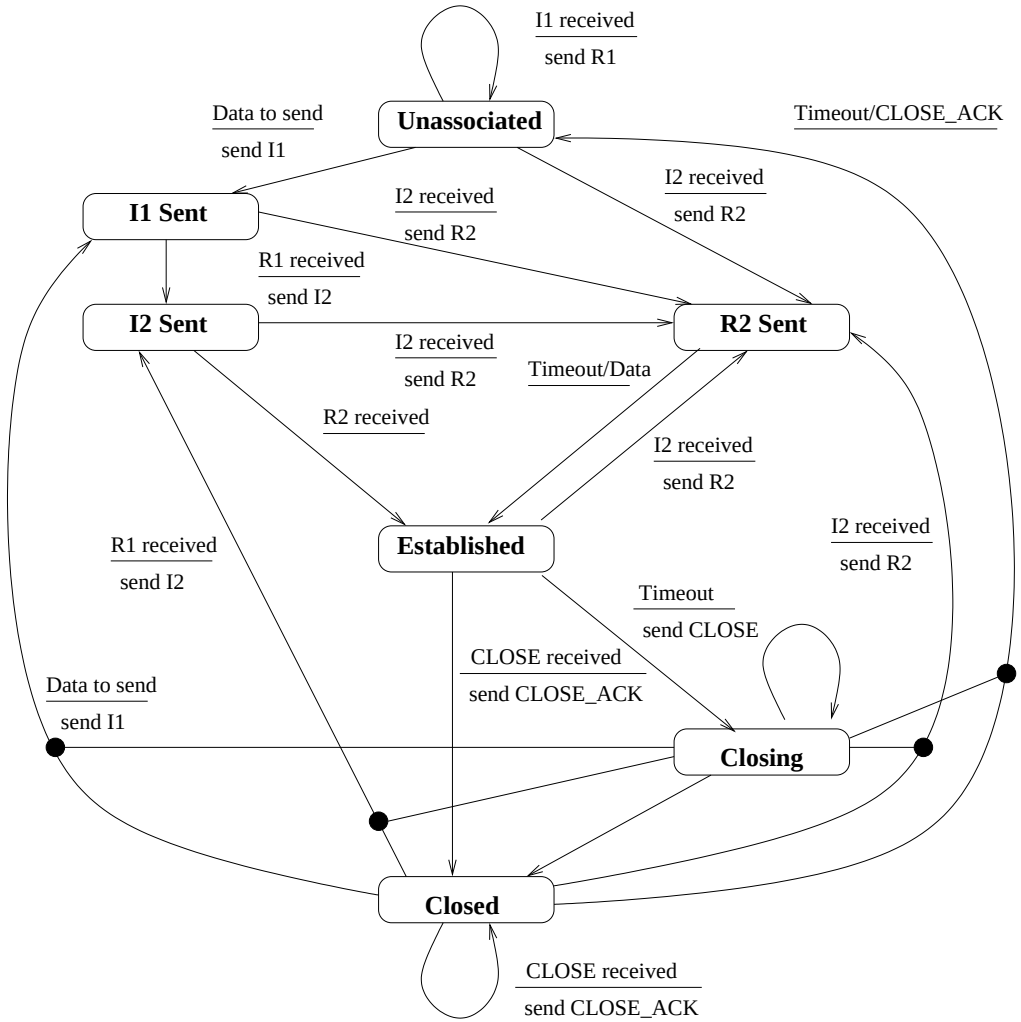
- All IPsec state information is kept in one place
- The BEET mode
  - a combination of transport and tunnel IPsec modes
  - a tunnel between two end hosts
- A pair of IP addresses is transmitted in packet headers
  - outer addresses - used on the wire (HITs that are fixed, can be left out of the headers)
  - inner address - applications see (can change dynamically during the SA lifetime)
- The ability to traverse middleboxes
- Inner and outer addresses can be from different address families

BEET headers with inner IPv4 addresses with IP options and outer IPv6 address



- the outer IPv6 is the one transmitted on the wire
- the body includes transport protocol segment encapsulated into ESP
- pseudoheader contains some IP options of inner IPv4 header - header type and length fields
- normally, no inner header is presented
- the parameter values from inner IP header are copied to the outer header upon transmission and back upon reception

A simplified state machine of a HIP association



Main extensions

Mobility and multihoming architecture

- The locator is a more general concept than a network address
- Defined locator format: a combination of a SPI prefix and an IP address
- The mobility and multihoming extensions modify a set of locators using an UPDATE message
- The requirement of peer's reachability after a mobility event

Mobility and multihoming architecture (cont.)

- Mobility with a single SA pair without rekeying - the simplest scenario
- First UPDATE packet
  - when one of the hosts moves to another subnet it needs to change its IP address
  - the host obtains the new IP address
  - UPDATE message is sent to the peer
  - new address in the LOCATOR parameter
  - the Old SPI and New SPI values in the ESP_INFO are set to the current outgoing SPI
  - contains a sequence number

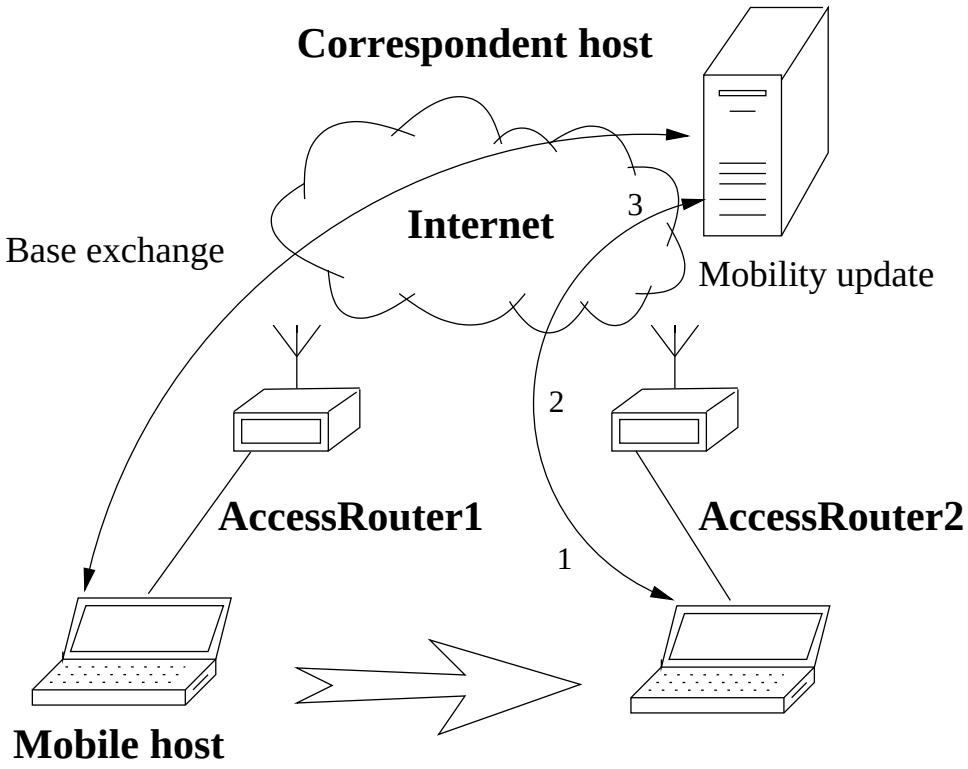
Mobility and multihoming architecture (cont.)

- Second UPDATE packet
  - allows peer to verify host reachability at the claimed new address
  - the Old SPI and New SPI in the ESP_INFO are set to the current outgoing SPI
  - includes acknowledgment for the first UPDATE
  - contains sequence number
  - an ECHO_REQUEST - verifying the new address

Mobility and multihoming architecture (cont.)

- Third UPDATE packet
  - an ECHO_RESPONSE proves that the host receives the data on the new address
  - acknowledges the second UPDATE message
  - not acknowledged
  - if the packet was lost peer retransmits the second UPDATE packet
- Mobility with a single SA pair with rekeying
  - a change of SAs

Mobility update signaling



1)UPDATE(ESP_INFO, LOCATOR, SEQ)

2)UPDATE(ESP_INFO, SEQ, ACK, ECHO_REQUEST)

3)UPDATE(ACK, ECHO_RESPONSE)

- new SPI - new value to be used for incoming SPI by mobile host
- an index to generate a new ESP session key
- new SPI for peer
- DIFFIE_HELLMAN parameters may be included to generate a new keying material

Multihoming as extension of mobility

- Basic multihoming scenario corresponds to mobility without rekeying
  - IP addresses belong to a single subnet - multihoming with a single pair of SA is feasible
  - address corresponds to different topological places - general case
  - requirement of separate SAs for a new address
  - to inform the peer of a new SA - the Old SPI is set to zero, the New SPI - new value of incoming SPI
  - peer uses the source and destination address of UPDATE to create a new SA
- Other cases correspond to mobility with rekeying

Multihoming as extension of mobility (cont.)

- Advanced scenario may combine mobility and multihoming
  - a mobile host can enable or disable network interfaces based on its location
  - the host is reachable by any address in its current active locator set
  - a mobile phone having UMTS, GPRS, WLAN - practical example
  - insufficient number of experiments
- Host can send LOCATOR in R1 and I2 during the BE
  - notifying about other locally available IP addresses
  - verifying of host reachability is necessary before sending significant data

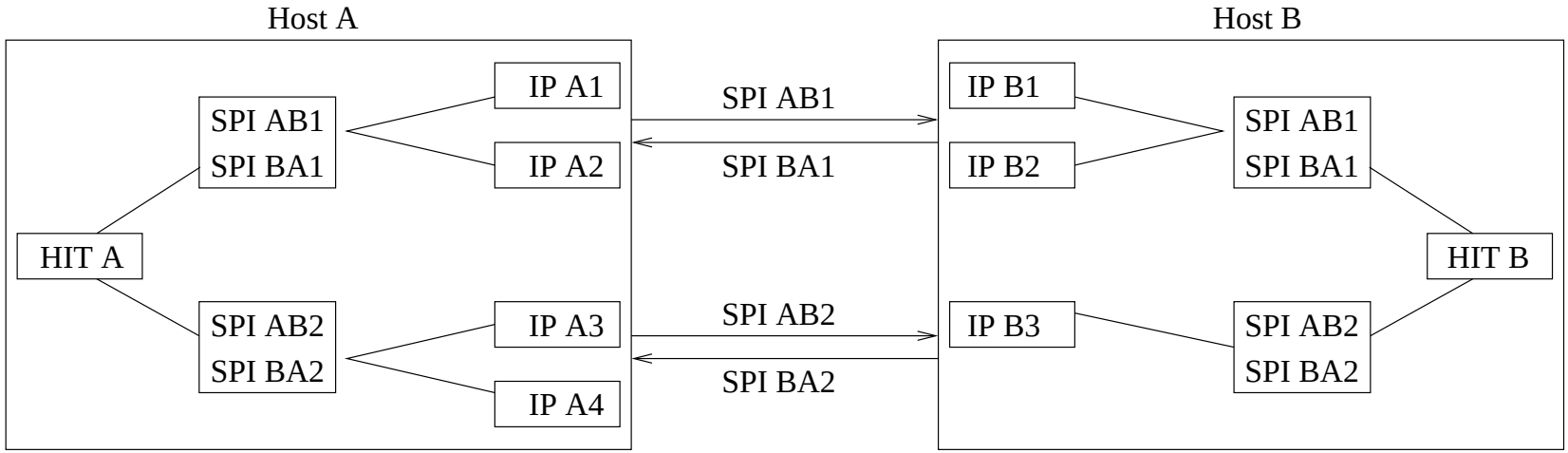
Effect of ESP anti-replay window

- One of the available IP addresses - a preferred locator
  - the default preferred locator - address used in the BE
  - data is sent to a preferred locator
  - source address - host's own preferred locator
- HIP associations - two uni-directional ESP SAs
  - several IP addresses can be added to an SA
  - transmitting and receiving HIP data packet through any of the addresses
  - SPI is used to lookup a proper SA
  - any number of SAs can be established between two HIP hosts

Effect of ESP anti-replay window (cont.)

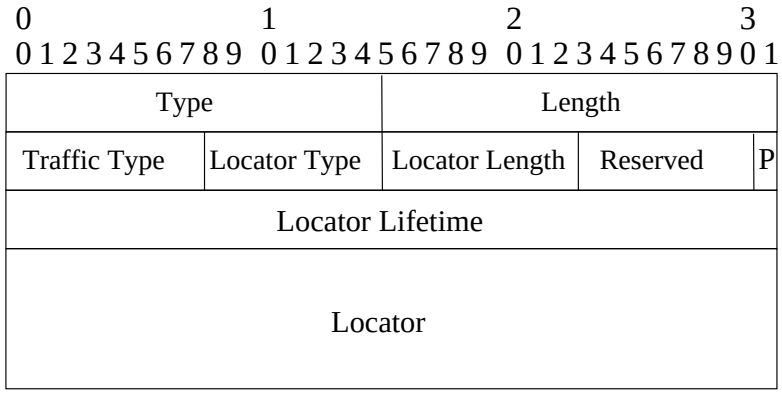
- The problem of single SA for multiple IP addresses
  - separate interfaces - different paths in the network
  - the anti-replay window limits the sequence number acceptable within SA
  - prevention of capturing and retransmission of packet duplicates
  - anti-replay window can be set large to permit the use of a single SA
- Each physical interface has a separate SA - specification recommendation
- One or more addresses can be assigned to an SA
- The addresses in the same group should be related (follow the same network path)
- Fate sharing

Relationship between HITs, SPIs, and IP addresses



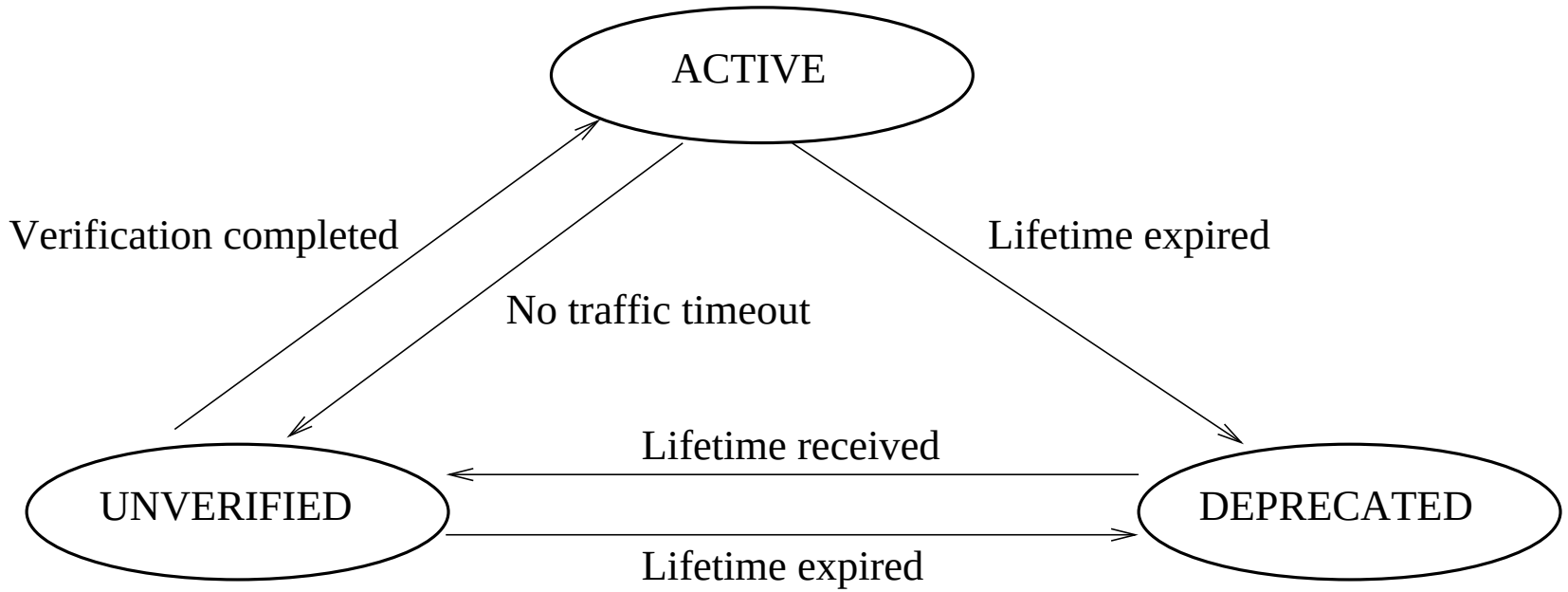
- two pairs of SA are created between A and B
- each SA has two destinations IP (except SPI AB2)
- a pair of SAs between groups of addresses
- asymmetric SAs are also possible (not all HIP implementation may support)
- a single address can be assigned to more than one SA (current specification does not support)

Format of LOCATOR parameter



- LOCATOR - a HIP critical parameter
- implementation does not understand parameter - the packet is not processed
- Zero locator fields - effect of deprecating all host addresses
- Traffic Type - locator is for HIP data (type 2) or control packet (1) or both (0)
- Locator Type - format of the locator fields (type 0 - IPv6 or IPv4-in-IPv6, non-ESP use; type 1 - first 32 bits of an SPI || IPv6 or IPv4-in-IPv6, ESP encapsulation)
- Locator Length - the length of the locator field in four bytes (1020 bytes - max)
- Locator Lifetime - how long the locator remains active (in seconds)

Three states of a locator



- ACTIVE - address is tested to be reachable and has not expired
- UNVERIFIED - address reachability has not yet been tested
- DEPRECATED - expired locator
- direct transmission from DEPRECATED to ACTIVE is not possible

Credit-based authentication

- A malicious host can re-direct traffic to a third party to cause DoS attack
- CBA must be used when sending data to an IP in UNVERIFIED state
- CBA approach is not HIP-specific (MIP also uses it)
- The attacker can forge the source IP to make them appear as arriving from the third party
- CBA idea - limit the rate of data transmission from a HIP host to the third party
- Host maintains a credit for each peer
  - the credit - the number of bytes received from IP of the peer
- The CBA turns on if there are no ACTIVE addresses, but UNVERIFIED address is available

Credit-based authentication (cont.)

- Credits
  - credit $>$ packet size - transmitting of a packet is possible
  - credit is increased with each arriving packet
  - credit is decreased with each transmitted packet
  - credit is too low - the host can drop it or queue it until the credit is sufficient or ACTIVE address
- Credit aging mechanism is intend to prevent an attacker building up a large credit
  - credit aging decreases the credit over time (5 sec. - aging period, 0.875 - aging factor)

Interaction with transport protocols

- Multihoming support for purpose of failover
- Multihoming for load balancing - simultaneous transmitting data from several locators
- Several issues at the transport layer
- Packet reordering
  - different times to reach destination - different order
  - case of out-of-order packet - TCP sends a duplicate acknowledgment
  - acknowledgment contains the highest seq number received so far
  - sender does not know if acknowledgments result from packet losses or reordering
  - sender waits for three duplicate acknowledgments to conclude that a packet was lost

Interaction with transport protocols (cont.)

- Spurious fast retransmit in TCP
  - out-of-order packet is mistakenly retransmitted by the sender
  - wasting network bandwidth
  - slow down the transmission (TCP halves the transmission rate in the case of packet loss)
- Solutions to overcome packet reordering
  - Eifel algorithm, based on TCP timestamp option
  - Duplicate Selective Acknowledgments (D-SACK)
  - reordering buffer below the transport layer

Interaction with transport protocols (cont.)

- Estimate of a retransmission timer
  - average of the path's RTT
  - timeout expires, the sender retransmits all outstanding packets in go-back-N fashion
  - retransmission timer has not actual value
  - variation of different RTT - retransmission timeout value is unnecessary high
  - the sender waits long before retransmitting, when packets are lost
  - Selective Acknowledgments (SACK) and Limited Transmit
  - using of a low-RTT path to let the timeout become low
  - spurious retransmission timeout

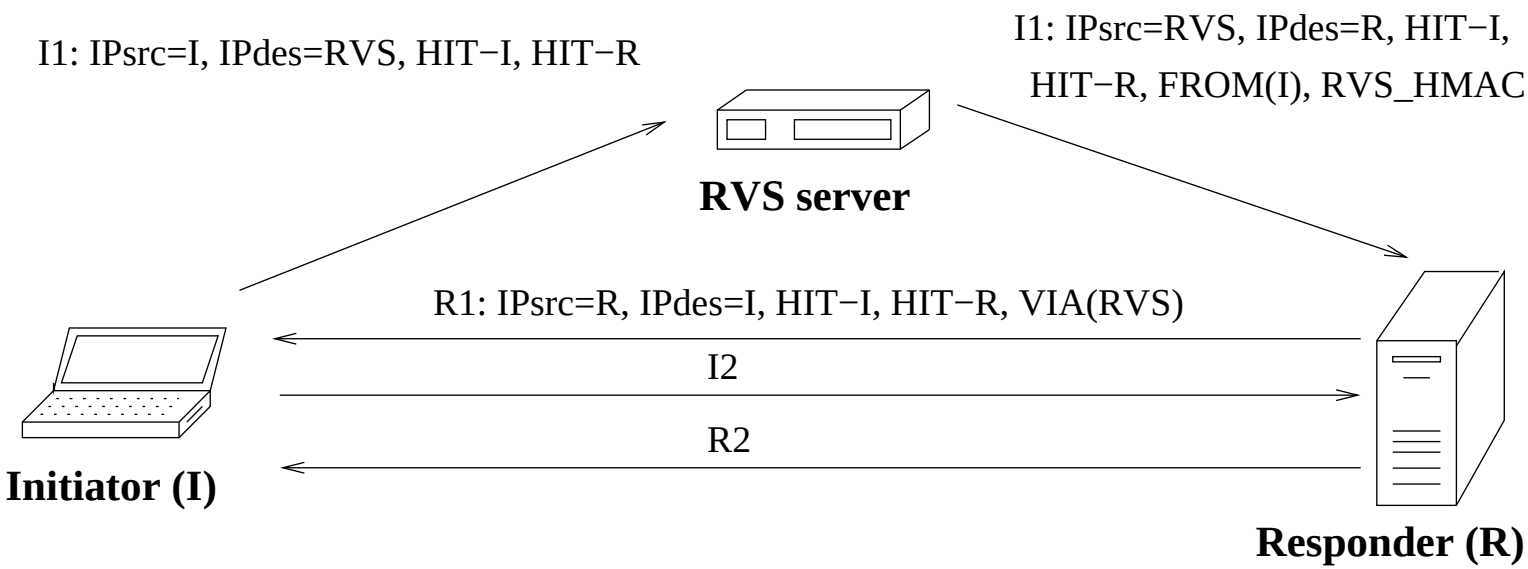
Interaction with transport protocols (cont.)

- Estimate of paths capacity
  - load balancing
  - the HIP layer estimates delay, bandwidth, and loss rate of each path
  - a HIP scheduler distributes packet
  - the design of the scheduler - current research work
- Different Maximum Transmission Unit (MTU) size
  - MTU is defined in the beginning of a connection
  - HIP mobility is hidden from the transport layer
  - possibility of packet fragmentation on the new path without knowledge of the transport protocol
  - the HIP layer could inform the transport layer of mobility events - transport triggers (under research)

Rendezvous server

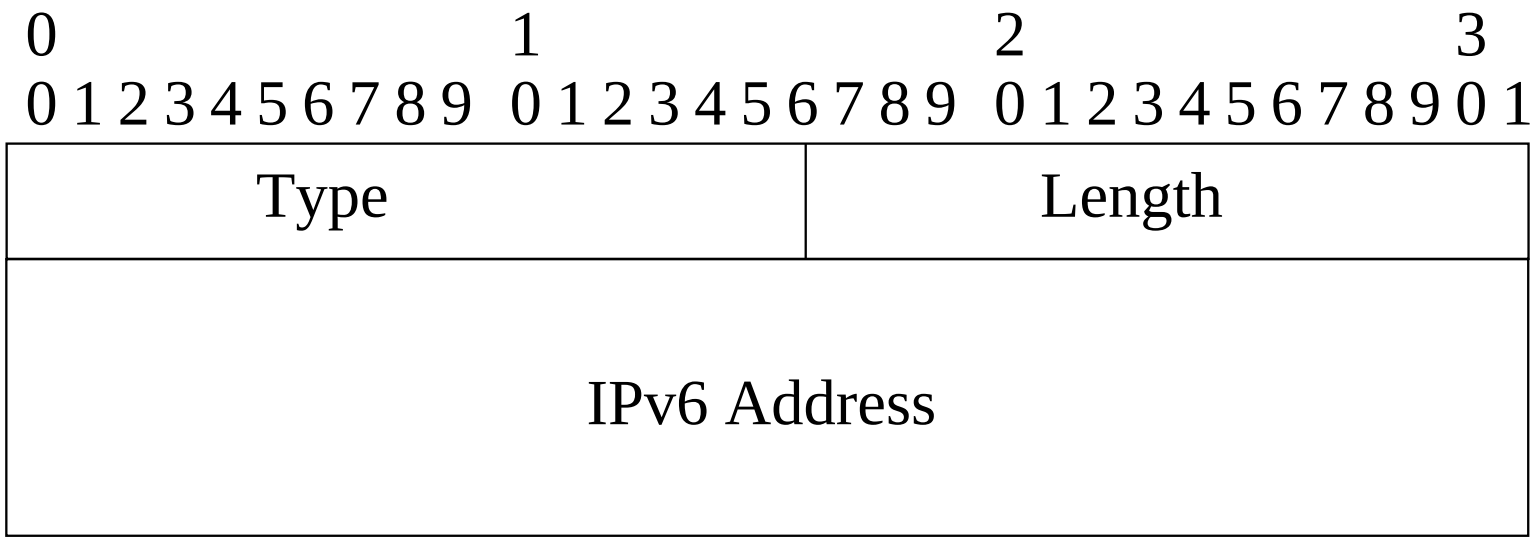
- A HIP mobile host can change its IP address often
  - other hosts cannot initiate a HIP association to a mobile host if its current IP is unknown
- The RVS provides host with stable IP address
  - the Initiator obtains the RVS address from DNS and sends an I1 to RVS
  - the RVS relays the I1 during the HIP BE
  - the mobile host registers with RVS by means of the registration protocol
  - the host updates its registration with RVS during mobility
  - the host can directly notify peers of IP address changes

HIP base exchange using a RVS server



- the I1 packet travels through a RVS, following control messages flow directly between hosts
- to distinguish a relay request the RVS compares the Responder's HIT in the I1 with its own
- the RVS checks active registrations for such HIT
- rewriting of the IP header - the address of the HIP owner in place of the destination address
- replacing the source IP in I1 with RVS own to bypass egress filtering
- recomputing of the IP checksum

Rendezvous parameters, FROM and VIA



- the Responder needs to know the IP of the Initiator
- FROM parameter contains the Initiator's source address
- RVS_HMAC parameter
- the Responder replies with R1 directly to the Initiator including VIA parameter
- VIA parameter contains the IP of RVS and is meant for diagnostics of HIP operations

HIP requirements to DNS

- Legacy DNS can store both HITs and IPv6 addresses
- DNS servers need modification to store HIP specific Resource Records (RR)
  - HIP RR - host public key, HIT, domain name of a RVS
- FQDN to IP address lookup
- The resolver also performs a FQDN to HI lookup
  - a HIT is returned - request family is INET6 or UNSPEC
  - an LSI is returned - request family is INET
- HIP layer stores IP addresses of a given HI after removing other records

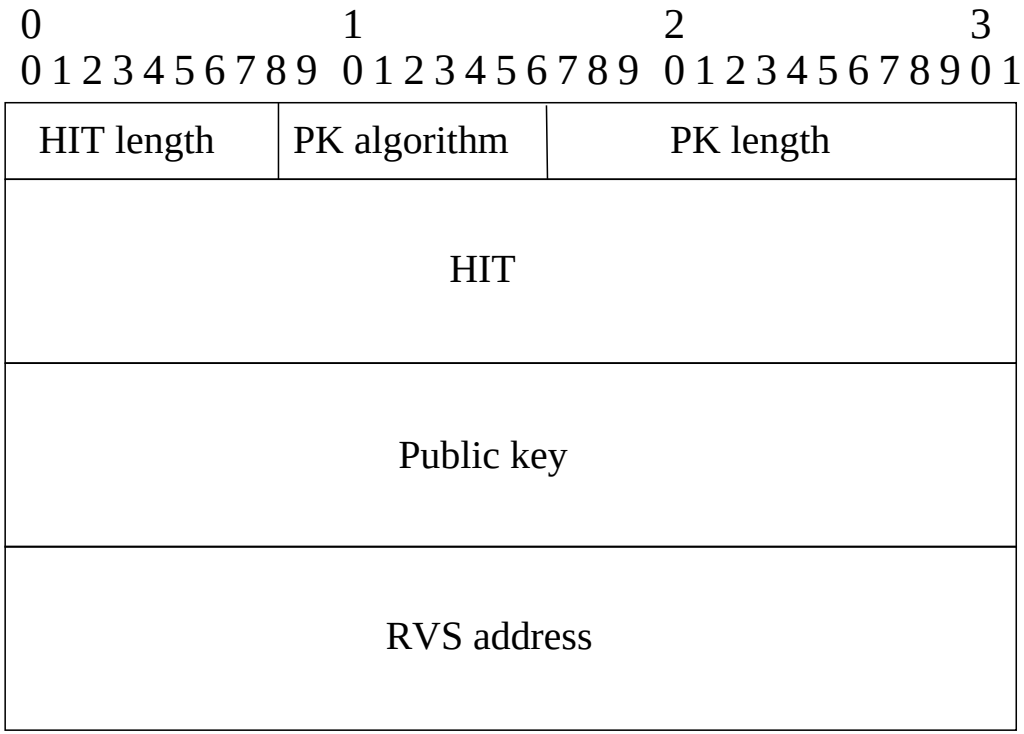
HIP requirements to DNS (cont.)

- A HIP application can directly request the HIP RR
- A benefit of storing HIs to DNS - prevention MITM attacks
  - by means of DNS information the Initiator can verify the Responder's identity
- DNSSEC - prevention of spoofing attacks

Storing a RVS address

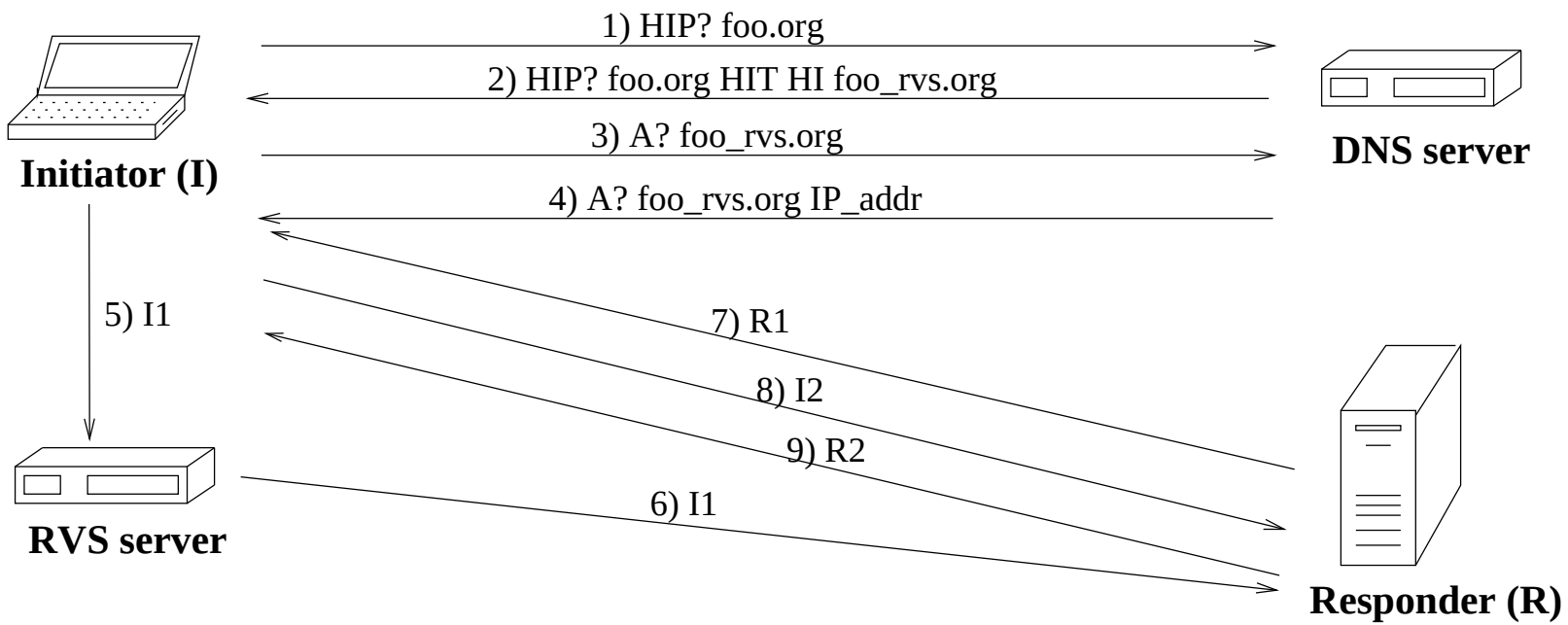
- Frequent changes of IP address
  - Hard to update DNS information fast enough
- The host can store the DNS name of its RVS
- The host can store the IP address of the RVS in the DNS instead of own IP addresses
- The Initiator sends I1 directly to RVS, RVS forwards it to a current IP address of the Responder

DNS Resource Record (RR) for HIP data



- two public key algorithms: DSA and RSA
- public key is encoded as defined for IPSECKEY RDATA RR (RFC 4025)
- RVS names are encoded (RFC 1035)
- length information for RVS names is not needed (evident from the encoding)

DNS query for a HIP mobile host



- a HIP host first queries a HIP RR from DNS, the IP address of the Responder or its RVS
- the Responder's host is likely to be mobile
- updating the IP address directly to the RVS
- the Initiator forwards the I1 through the RVS

DNS security

- DNSSEC provides a secure channel from the DNS to the host
- The absence of the DNSSEC can lead to a forgery of DNS replies by an attacker
  - the attacker can redirect HIP packets to itself or to a third party
- Increasing number of possible attacks on the one-way property of hash function such as SHA-1
  - the attacks decrease efforts needed to find a collision on a hash output
  - two separate hash inputs map to the same hash output
  - HIT is hash of the public key - the Initiator should use the full public key to identify the Responder

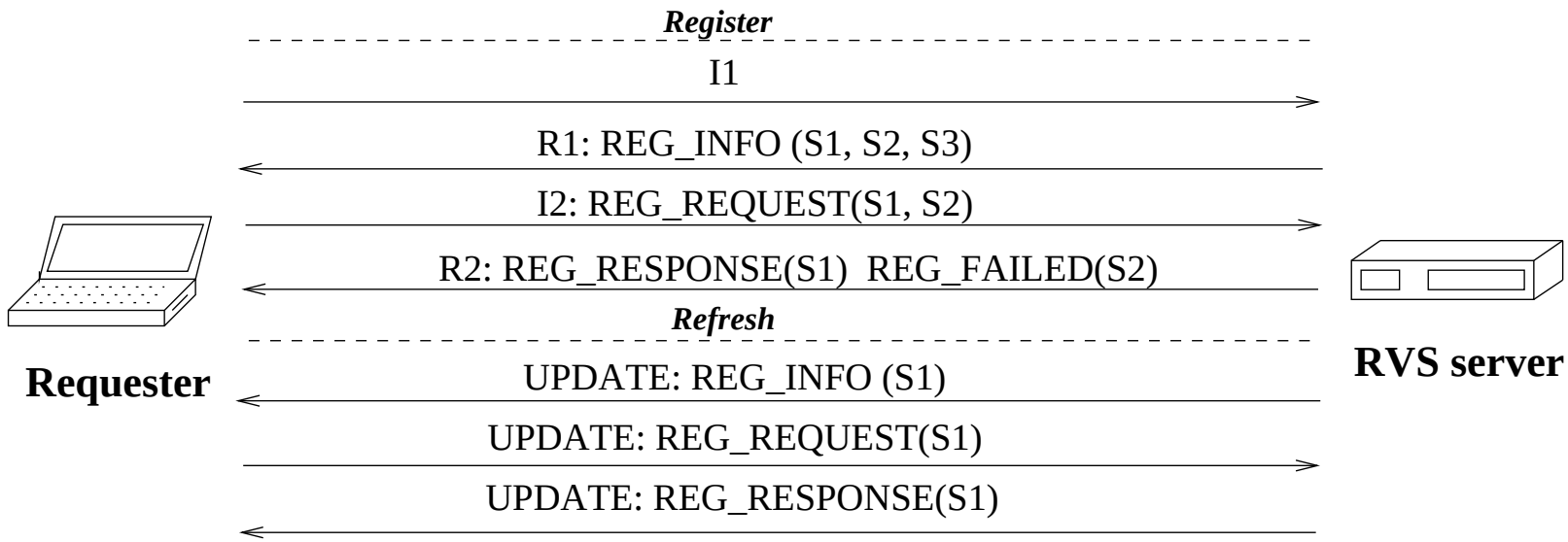
Registration protocol

- HIP registration protocol - extension to the base protocol
- Registration to a service, such as RVS or HIP-aware firewall
- Implemented as a HIP BE with additional parameters
  - the registrar announces a list of offered services, REG_INFO in R1 or UPDATE
  - the requester includes the REG_REQUEST to I2 or UPDATE
- Registration can be mandatory - registrar can terminate HIP BE with a NOTIFY
  - NOTIFY includes REG_REQUIRED parameter

The process of registration

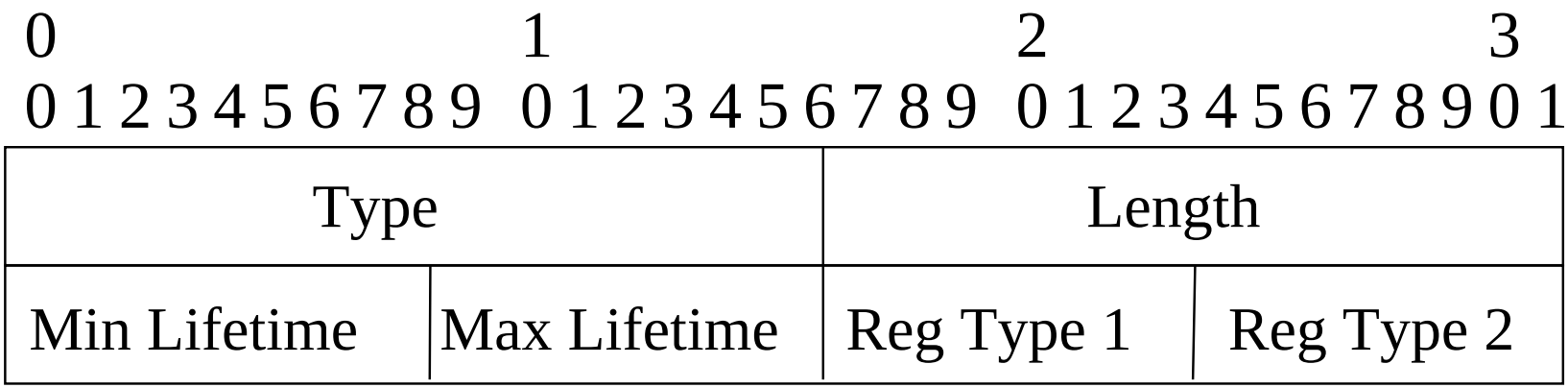
- The registrar receives I2 with REG_REQUEST
- Authentication and authorization of the requester based on its public key
- The details of the authorization depend of the service offered
  - firewall can check if requester's HIT is included into appropriate Access Control List
- Authorization successful - the registrar creates state
- Otherwise - REG_FAILED parameter indicates failure type
- Registration for several services - packet can contain both REG_RESPONSE and REG_FAILED

Registration with a RVS server and its refresh



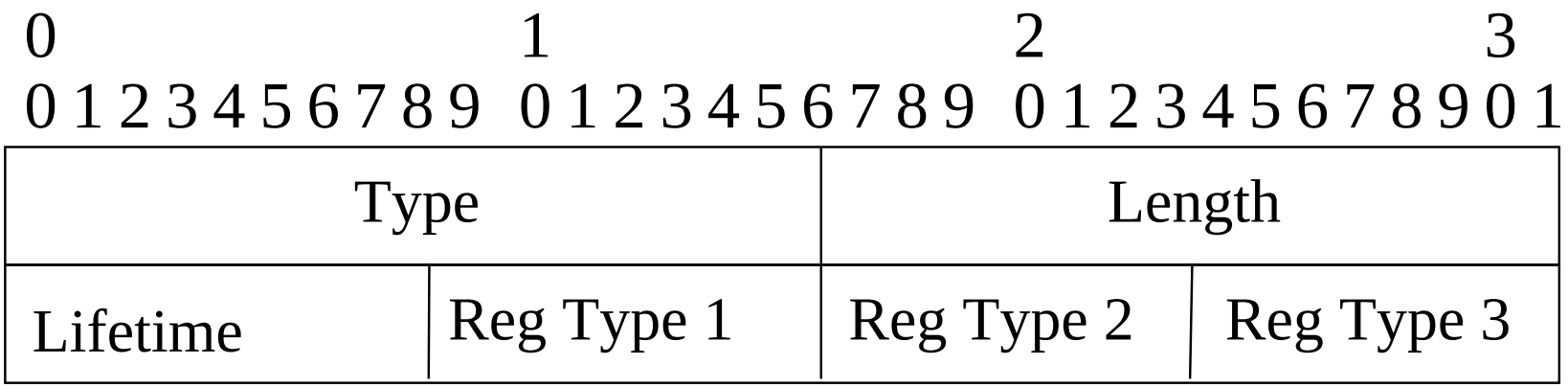
- registration during the BE
- S1, S2, S3 - existed services
- attempts to register to S1 and S2
- refresh process is initiated by the registrar with an UPDATE

Packet formats, REG_INFO parameter



- lifetime field is encoded in an exponential form (range from 4 sec up to 175 days)
- all services should support a minimum lifetime of 10 sec and the maximum of 120 sec
- one or more Registration Types fields that are offered by the registrar

Packet formats, REG_REQUEST and REG_RESPONSE



- Registration Types in the order of preference
- best interoperability - sender should include more than one parameter
- REG_REQUEST and REG_RESPONSE with a zero lifetime cancel registration
- REG_FAILED parameter has a similar format to REG_REQUEST (lifetime is replaced by a failure type: service is not supported or additional credentials are required)

Advanced extensions

Opportunistic mode

- HIP association establishment without prior knowledge of the Responder's HIT
- Initiating opportunistic BE
  - BE starts from the I1 with NULL destination HIT
  - the Responder can choose any of its HI
  - NOTIFY packet with BLOCKED_BY_POLICY error means that the Responder does not allow opportunistic mode
  - the Initiator locates the I1 using the source IP of R1
- Security vulnerabilities - should be used in a trusted environment
- Susceptible to MITM attacks

Opportunistic mode (cont.)

- A leap of faith
  - SSH users have to place certain trust that key is authentic
- It is best to perform the leap of faith in a trusted environment
- High probability that the exchange HIs are authentic
- After, HIP associations can be established in the public Internet

Implementation using a TCP option

- Special TCP option to carry the Initiator's HIT in a TCP SYN message
- Establish a regular TCP connection if the Responder does not support HIP
- If the Responder supports HIP
  - treatment of TCP SYN as an opportunistic I1
  - the Responder replies with R1
  - the Responder can discard the SYN packet - the TCP state is not created, avoiding DoS
  - the Initiator will retransmit SYN after HIP BE

Implementation using a TCP option (cont.)

- The drawback of using TCP - limits the operation to only TCP applications
- Design of solution that would work for all transport protocols
- IP header option to carry the HIT
  - packet with unknown IP options are often dropped by routers

Piggybacking transport headers to BE

- Carrying any transport protocol headers in HIP packets appears useful
  - TCP application - 3.5 RTTs delay before data transmission
  - combination of BE and TCP handshake reduces delay to 2 RTTs

Piggybacking transport headers to BE (cont.)

- Piggybacking TCP SYN packet to HIP I2
  - the Responder checks the puzzle solution in I2
  - processing of the SYN
  - creating state and transmitting TCP SYN-ACK in R2
  - DoS resistance capabilities of HIP are preserved
  - Next Header field in I2 is set to 6 to indicate a TCP header
  - Next Header in R2 is set to 6, TCP SYN-ACK packet after all I2 fields
  - no support of current specifications
  - ”piggybacking compatible” flag in I1 and R1

Security concerns

- Further reduction of delay - TCP SYN in I1
  - additional data at the Responder before puzzle verification
  - disruption of DoS resistance of HIP
- DoS attacks are experienced occasionally - Responder may allow data in I2
- TCP ACK message can already contain user data
- The user data needs to be encrypted - not possible with the current BE specification
- Piggybacking TCP SYN and SYN-ACK to I2 and R2 exposes TCP headers

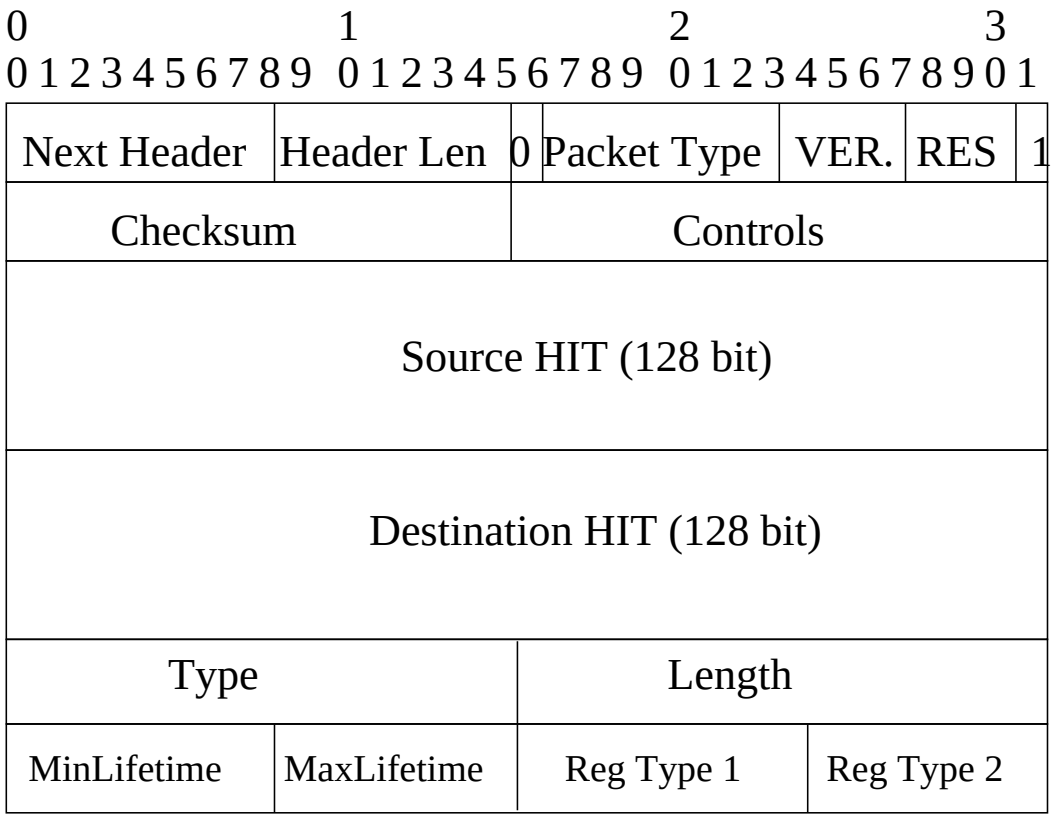
HIP service discovery

- Finding available HIP services
  - services currently specified for HIP - RVS
  - can be defined in the future - HIP-aware firewall or NAT
- Registration process to a discovered service - registration protocol
- Bootstrap (BOS) packet
  - discovering HIP hosts in the same LAN
  - BOS packet includes the HIT of the host to be broadcast
  - the BOS - rudimentary service discovery mechanism

Overview of service discovery

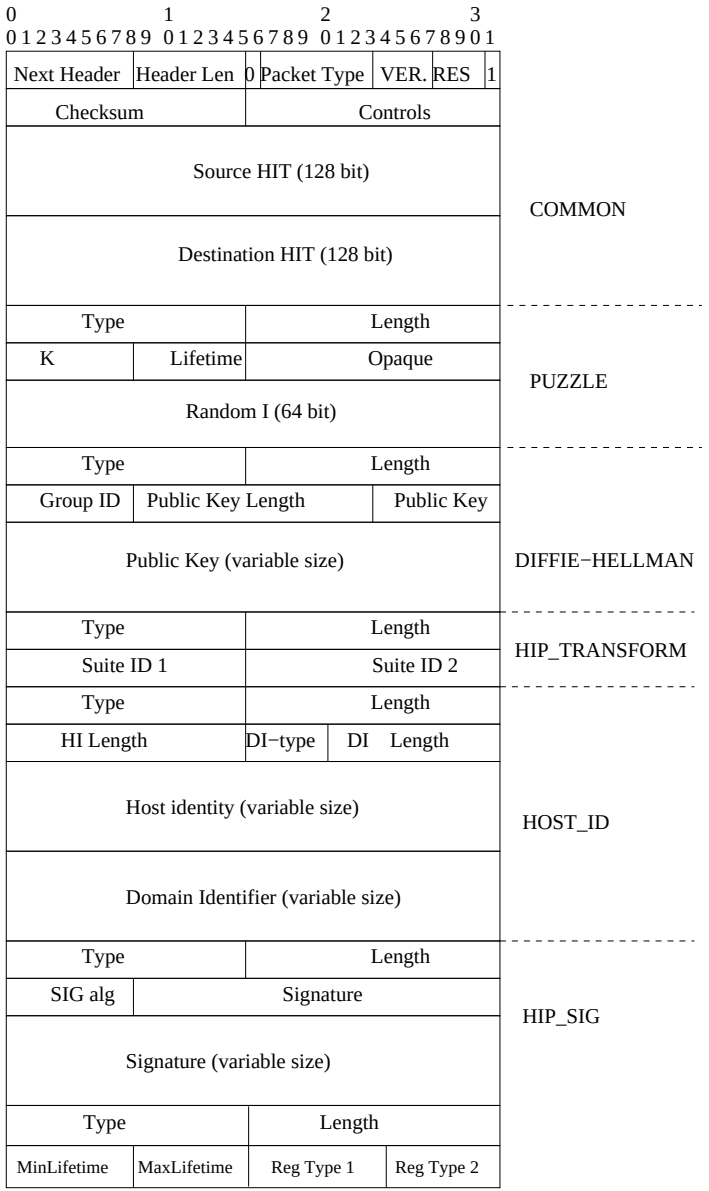
- Two modes of operation: on-the-path and local network
- On-the-path
  - service provider middlebox is located on the path to the destination
  - a HIP host sends an I1, UPDATE or Service Discovery packet towards the destination
  - packet arrives at the Service Provider - Service Announcement packet to the HIP host
- Local network discovery
  - HIP host broadcast a Service Discovery packet within the local area network or to IPv6 site-local addresses
  - enables location of services that are outside of the path to know HIP peer hosts

HIP service discovery packet format



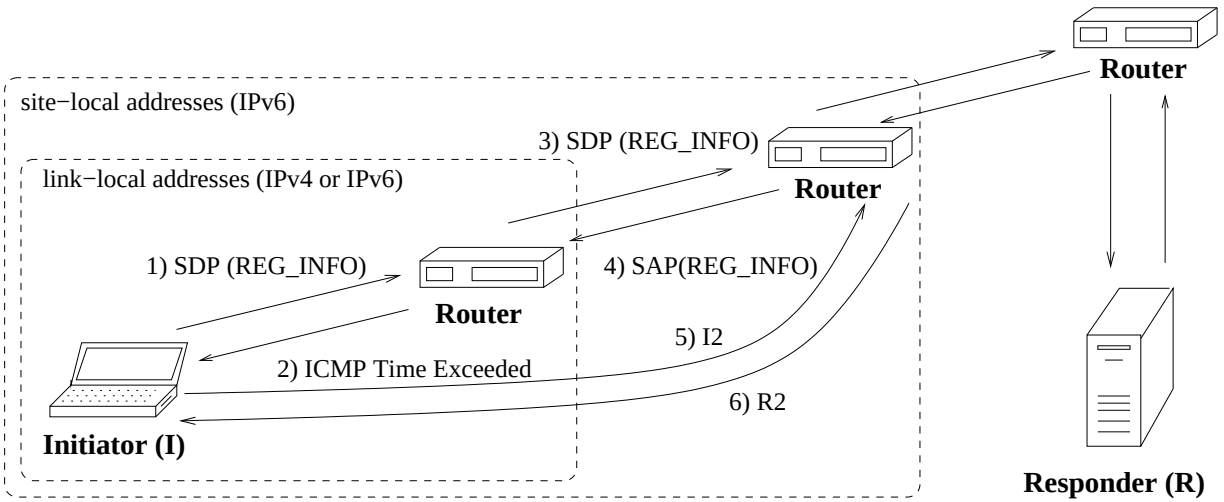
- the Service Discovery and Service Announcement can include the same fields as in I1 and R1
- a HIP host that initiated the discovery replies with I2 to the Service Provider
- a BE can be completed with R2 from the Service Provider avoiding additional delays

HIP service announcement packet format



- has a similar format to R1
- REG_INFO describes the available services
- active discovery - SAP is sent in response to the Service Discovery
- passive discovery - SAP can be sent if an I1 or UPDATE passes through a middlebox offering a service

On-the-path Service Discovery



- a HIP host attempts to discover a suitable service
- the HIP host sends SDP with REG_INFO, where the desired service is specified (HIP-aware firewall)
- the source HIT - HIT of the Initiator, the destination HIT - zero
- the Initiator would like to discover service from known host - HIT in place of zero

On-the-path Service Discovery (cont.)

- the Initiator sets the Time-to-Live (TTL) to one and transmits the SD to the network
- the first router - ICMP Time-Exceeded, does not implement any services (before replying router checks if the source address of SD is a unicast IP)
- the HIP host increases the TTL to two and transmits a new SD
- during the discovery process the Initiator is in the SD_SENT state
- the Initiator can store several SAP, before deciding with which Service Provider it wants to register
- the process continues until a pre-defined TTL is exceeded, the destination host is reached, or a SAP is received
- Service Providers send the SA if they provide a service listed in the REG_INFO in SDP
- the second router implements a HIP firewall service, replies with SAP
- I2 and R2 complete the BE and registration between the Initiator and firewall service
- the Initiator can make a separate BE to the Responder host

Passive Service Discovery

- Implemented by a Service Provider replying to the Initiator host after forwarding its I1 or UPDATE
- The Service Provider should verify if source HIT is in the list of HITs currently served
- The Service Provider remains stateless after transmission of the SAP to avoid DoS
- The Initiator is interested in the offered service - I2 to complete the registration process
- The HIP implementation on the Initiator must allow arriving SAP in the ”I1 sent” or ”UPDATE sent” states

Regional Service Discovery

- Sending a SDP to a local multicast or broadcast address
- Destination HIT - zero, destination IP - special-purpose address
- For IPv4 host can use only the broadcast address 255.255.255.255
- For IPv6 host can use one of the following:
  - link-local all multicast address, FF02:0:0:0:0:0:0:1
  - link-local routers multicast address, FF02:0:0:0:0:0:0:2
  - site-local routers multicast address, FF05:0:0:0:0:0:0:2
- SAP is sent after a small random delay to avoid congestion with many SAP
- The transmission rate of SAP is limited to avoid DoS to a spoofed IP address

HIP service discovery (cont.)

- Presented mechanisms are insecure
  - SDP is not protected by a signature
  - SDP could be modified in transit or sent from a spoofed IP address
- The Initiator should take SAP only as a hint and use external security mechanism to verify authenticity of the service provider
- Depending on security settings a HIP host can implement the active discovery mode

Simultaneous multi-access

- HIP mobility and multihoming extensions permit the use of multiple interfaces
  - for communicating with different peer hosts
  - for failover when primary path to the peer fails
- Simultaneous use of multiple interfaces towards the same host is not supported

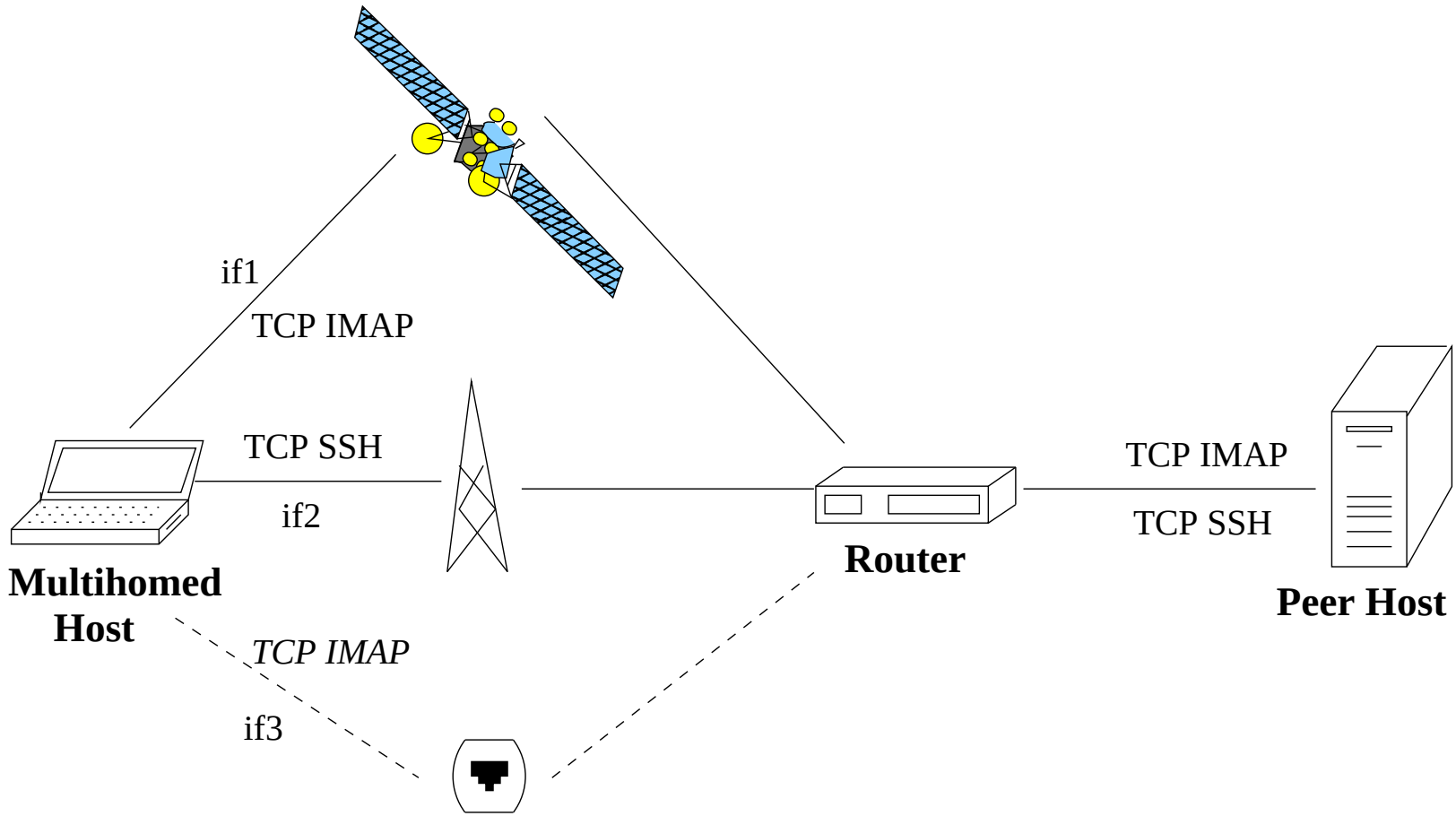
Flow binding extensions

- The experimental HIP extensions for SIMA - multiple interfaces towards the same peer
- Separating transport-layer connections (TCP and UDP)
- Transmission over multiple interfaces by a single transport connection - not supported
- SIMA_FLOW_BINDING parameter in UPDATE
- Peer host can select a SA and destination LOCATOR
- SIMA_ACK parameter in reply UPDATE confirms SIMA support
- The location update before the Flow Binding

Flow binding extensions (cont.)

- Flow identification for only TCP and UDP
- Use of other protocols requires further extensions
- A flow binding database contains flow preferences
- For each flow - list of interfaces in the preference order
- By selecting the SA, flow selects the interface to use (SA has a source and destination address)
- SIMA_FLOW_BINDING includes rules for allocating flows to interfaces
- The peer follows the rules for selecting SPI and destination address
- To change the flow preferences - re-send an UPDATE with a new SIMA_FLOW_BINDING

Example of simultaneous multi-access



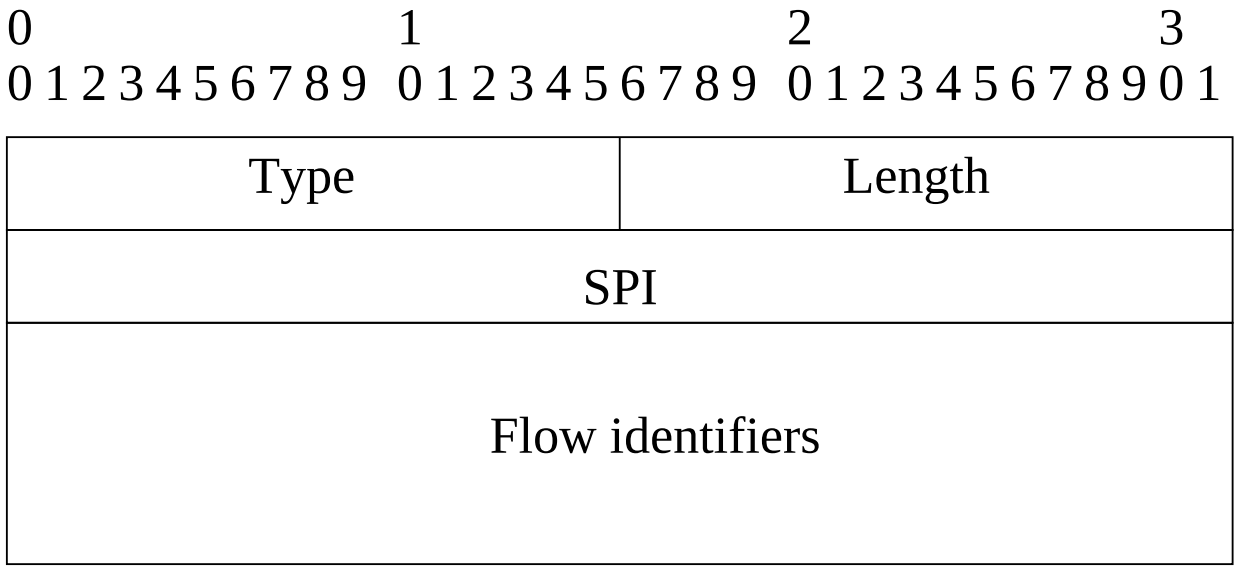
Example of simultaneous multiaccess (cont.)

- TCP connections to a port 143 (IMAP email), interfaces if3, if1, if2
- TCP connections to a port 22 (SSH remote shell), interfaces if2, if1, if3
- if3 is inactive in the beginning
  - multihomed host informs the peer to use if1 and if2
  - SA is set to the peer from both if1 and if2 by sending an UPDATE with two SIMA_FLOW_BINDING

Example of simultaneous multiaccess (cont.)

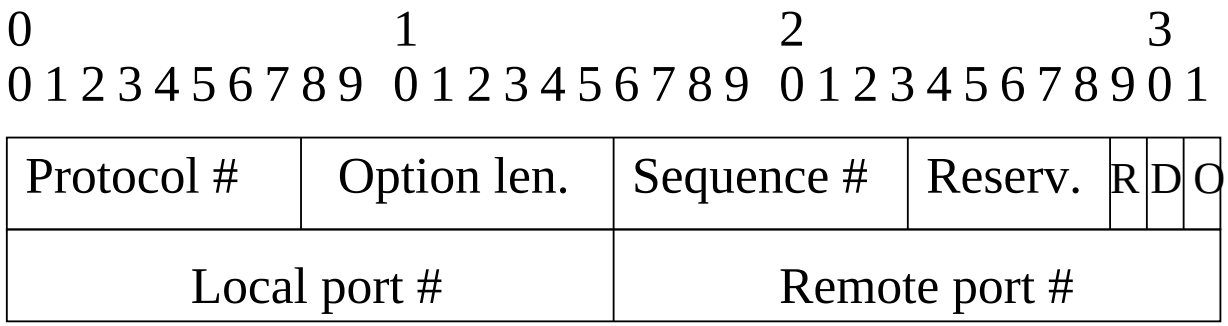
- if3 becomes available
  - normal UPDATE with LOCATOR
  - SAs are created between if3 and peer
  - UPDATE with the SIMA_FLOW_BINDING with a SPI value of if3 for TCP port 143
  - IMAP flow binding uses the new interface if3

Packet formats, the SIMA_FLOW_BINDING parameter



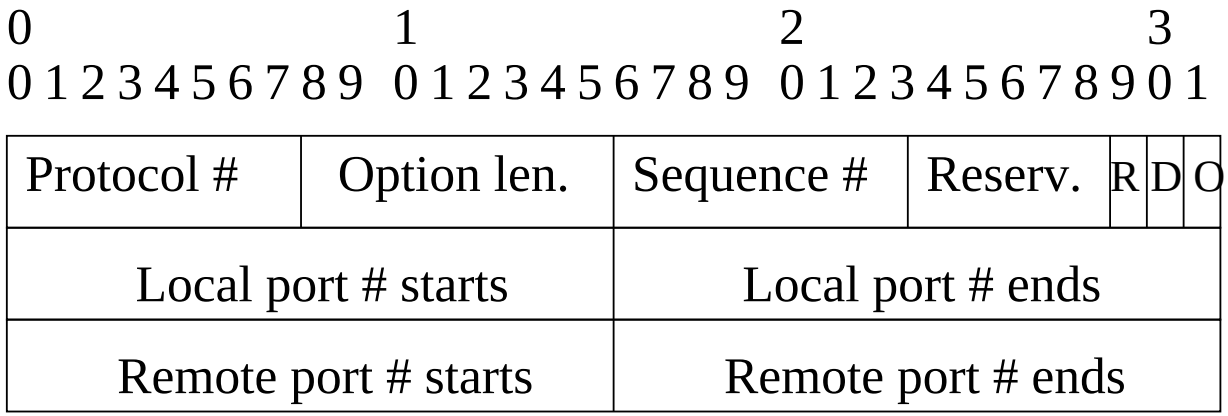
- the type value - 64
- one or more flow identifiers to a single SPI
- one UPDATE can contain multiple SIMA_FLOW_BINDING parameters
- the receiver ignores SIMA_FLOW_BINDING if it does not support SIMA

Packet formats, TCP or UDP flow identifier



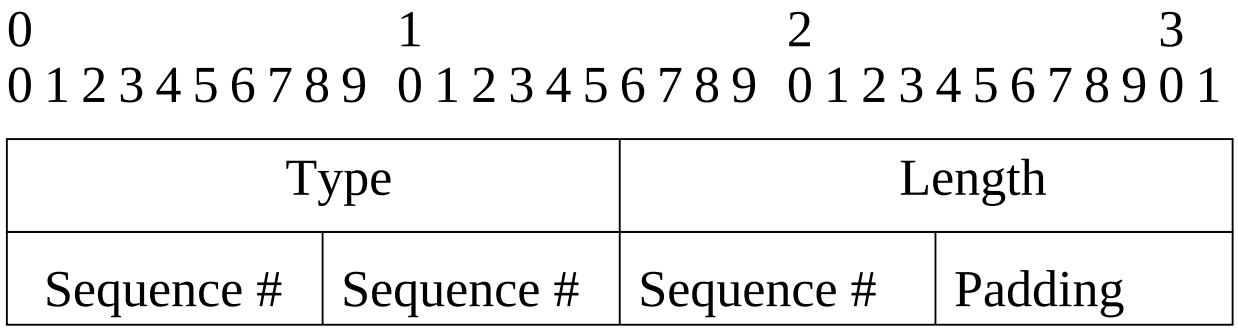
- optional length - the length of identifier without the basic header
- reserved field is set to zero when sent
- R flag indicates the use of port ranges
- D flag requires the peer to discard all existing flow bindings
- O flag is set to add a flow and cleared to remove the flow

TCP or UDP flow identifier with port ranges



- can bind multiple flows to use a given interface by specifying a range of local ports and remote ports for TCP or UDP

Packet formats, the SIMA_ACK and SIMA_NACK parameters



- SIMA_ACK confirms support of SIMA by a HIP peer
- SIMA_ACK includes a list of sequence numbers from SIMA_FLOW_BINDING received successfully
- SIMA_NACK informs that the flow binding updates with given sequence numbers were rejected

Packet formats

- UPDATE with one or more SIMA_FLOW_BINDING parameters is sent
  - set of active interfaces changes (e.g., mobility)
  - flow binding rules change (e.g., a user alerts configuration settings)
- Each SIMA_FLOW_BINDING has a sequence number - incremented by one each time
- A peer host updates its flow binding table according to SIMA_FLOW_BINDING
- Outgoing IPsec packet - peer performs a table lookup
- Incoming ESP packets - handled regularly by SIMA host

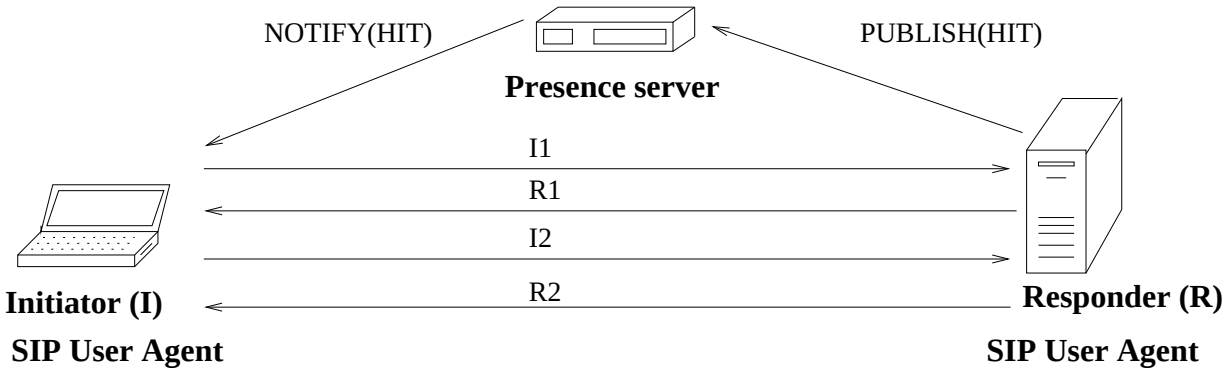
Disseminating HITs with a presence service

- Establishment of a HIP association - knowledge of a peer HIT or the use of opportunistic mode
  - peer's HIT can be manually configured, distributed using DNS or DHT lookup
  - Opportunistic mode is convenient, but MITM attacks are possible
- The mechanism to exchange HITs is necessary, especially when HIP is used with SIP

HITs in the Presence Information Data Format

- New information element in the PIDF
  - the element is developed by the SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE) working group at IETF
- Dissemination of HIs or HITs using the presence data model in SIP
  - the element sets up a mapping between HI and SIP URI
- Logically separate service (User Agent) and device parts of the architecture
- A SIP service can run on multiple devices and be identified by a Globally Routable Unique UA URI (GRUU)
- The devices can be identified using a HIP HI

HIT dissemination using a SIMPLE extension



- SIMPLE PUBLISH message uploads Responder's HIT to the presence server
- SIMPLE NOTIFY message distributes the HIT to HIP Initiator
- the Initiator can start a SIP session
- the Initiator triggers the HIP BE and establishment of ESP SA
- the SIP handshake completes over over the HIP association

HIT dissemination using a SIMPLE extension (cont.)

- the mechanism also supports the use of HIP RVS
- the Responder can supply an IP of its RVS in the PUBLISH message
- the RVS relays the I1 message to the Responder
- the RVS is not registered to the presence server - the Initiator may need to perform a separate HIT to IP mapping to locate a mobile Responder host

Multicast

- The HIP architecture is for host-to-host communication, unicast
- Internet TV - data transmission from one source to multiple destinations
- Video-conferencing - transmission from several sources to several destinations
- Multicast data transmission
- Multicast implementation
  - on the networking layer as native IP multicast (ASM, SSM)
  - on the overlay layer as an application service (i3)
- HIP multicast - a public key identifies a group of hosts
  - group must generate and share a private key between members

Challenges for IP multicast

- Mobility of host participating in multicast is a largely unsolved problem
- Multicast source changes the IP - reconstruction of multicast tree is necessary
- Current multicast solutions do not allow construction of native dual-stack IPv4/v6 multicast trees and do not support multihoming
- Two common approaches for multicast receiver mobility
  - Bidirectional Tunneling
  - Remote Subscription

Challenges for IP multicast (cont.)

- Bidirectional Tunneling
  - the receiver subscribes to the multicast stream
  - subscription - via receiver's home agent located in the user's home network
  - the user moves to a foreign network - creates a tunnel to the home agent
  - the home agent relays the user's multicast signaling and stream data
  - the user can move between networks without affecting the multicast tree

Challenges for IP multicast (cont.)

- Remote Subscription
  - the asks the local multicast router in a visiting network to join the multicast tree
  - the old branch of the multicast tree from the previous user's network location eventually times out
  - the data starts arriving to the new user's location

Challenges for IP multicast (cont.)

- In SSM - the entire multicast tree uses the IP of the source as the root
  - the sender moves - the entire tree needs to be rebuilt using the new IP as the root
  - rebuilding process - multicast stream is interrupted and listeners do not receive any data
  - bi-directional tunneling can be applied to avoid the tree reconstruction for the mobile multicast sender
- Solutions to provide authentication to multicast receiver
  - Multicast Control Protocol (MCOP) - only able to authenticate the subnetwork

Challenges for IP multicast (cont.)

- Difficulty of constructing a multicast tree combining IPv4 and IPv6 hosts
- Solutions for IP version interoperability
  - aimed at unicast communication
  - do not yet support construction of dual-stack multicast trees

Host Identity Specific multicast

- ASM - hosts are identified with an IP of the multicast group (G)
- SSM - unicast IP of the multicast source is necessary to construct the multicast tree (S,G) in addition to the multicast address
- HISM - HIT_S for multicast source and HIT_R for multicast receivers
  - the use of HITs allows changing of IP without the need to reconstruct the multicast tree

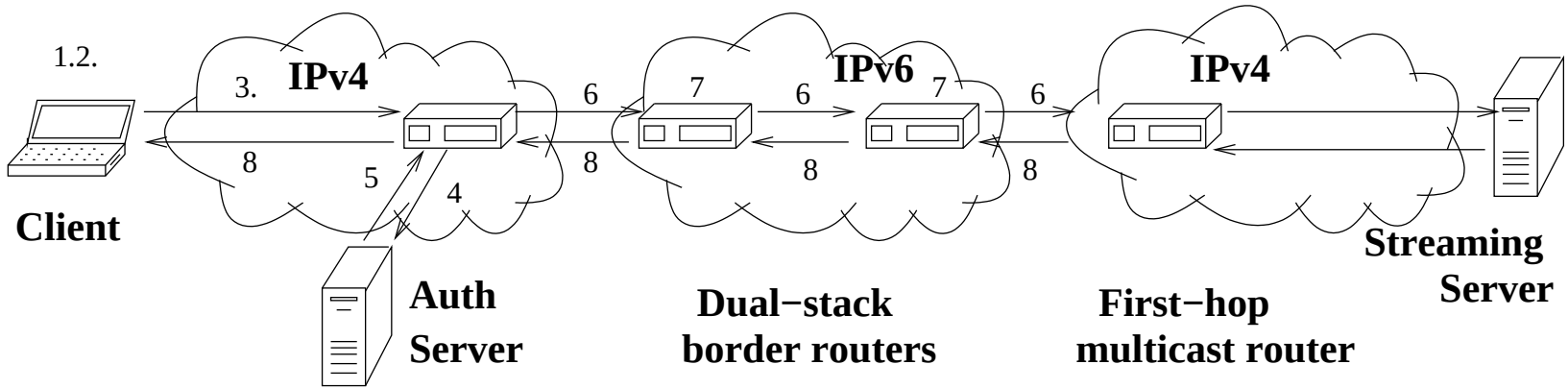
Host Identity Specific multicast (cont.)

- The Session ID (SID) - version-independent identifier
  - has a length of 26 bits
  - applications can create IPv4 or IPv6 multicast addresses from SID
  - for IPv4 - appending the prefix ”111011” before 26-bit SID
  - for IPv6 - appending the prefix ”FFFF::” before 26-bit SID
  - the HISM channel is identified by a pair (HIT_S, SID)
  - benefits of source specification of SSM are preserved
  - dependency of fixed IP addresses is removed

Host Identity Specific multicast (cont.)

- In traditional multicast, a host can join the multicast channel by following the subscription procedure
  - the host application triggers subscription by giving the group multicast address G and IP address of the multicast source (for SSM only) to the host operating system
  - the host sends out join messages to the multicast group using Internet Group Management Protocol (IGMP) for IPv4 and Multicast Listener Discovery (MLD) reports for IPv6
  - the join message reaches the first multicast router (designated router) - Protocol Independent Multicast (PIM) join message to attach to the multicast tree
  - PIM join message reaches a multicast router - multicast stream data starts to flow to the receiver host

Architecture of Host Identity Specific Multicast



(1) - the application specifies the pair (HIT_S, SID) that the user wants to listen

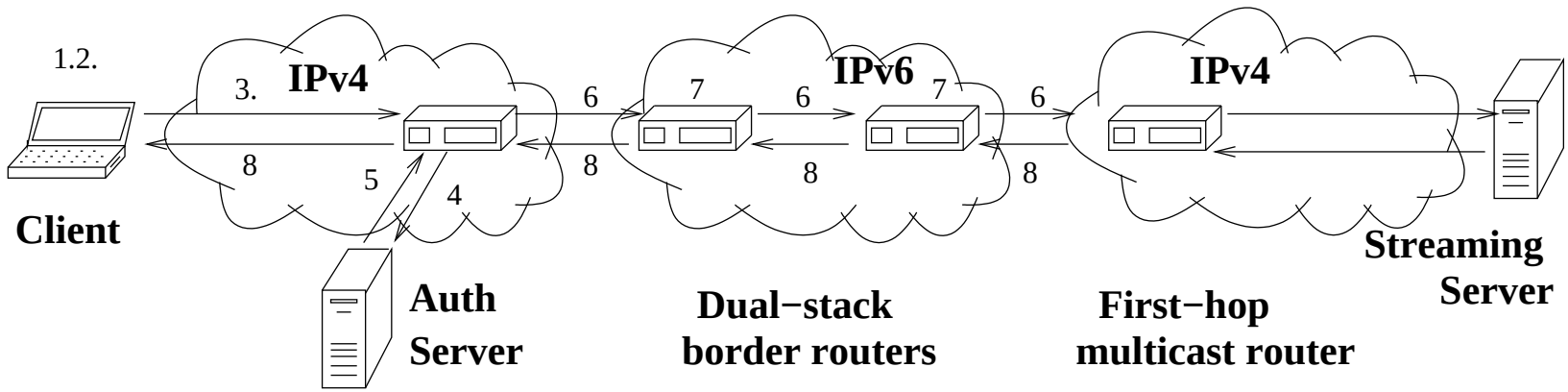
(2) - the operating system constructs IPv4 or IPv6 multicast from SID

(3) - the group management join message is sent, contains the HIT_R for user authentication and address control

(4) - the DR starts authenticating the subscription request based on (HIT_S, SID) and HIT_R

(5) - the authentication server evaluates authorization of the host to view the multicast stream based on HIT_R and resolves the current IP of the source based on HIT_S

Architecture of Host Identity Specific Multicast (cont.)



(6) - the server maintains a list of dual-stack border routers for IPv4/IPv6 interoperation and gives the address of the dual-stack router closest to the source. Authentication is successful - the DR joins the multicast tree

(7) - the normal PIM is sufficient - source and receiver use the same IP. Otherwise, the join request is constructed using the dual-stack router as multicast source

(8) - the dual-stack router performs address conversion and continues construction of the tree. The join procedure completes when the join request reaches the DR router of the source or another multicast router that is already a part of the tree

Host Identity Specific multicast (cont.)

- HISM - reconstruction of the multicast tree when the source changes its IP is needless
- It is sufficient that the resolution mechanism returns a correct IP based on the HIT_S
- The deployment of HISM generates a software update in the multicast routers but does not require support of HIP in the routers
- The use of public-key cryptography in the routers would significantly reduce their routing speed

Host Identity Specific multicast (cont.)

- To support HISM, applications should give the pair (HIT_S, SID) of the multicast stream the user is interested in
  - the legacy IPv6 applications can easily pass HIT_S instead of source IP address in SSM
  - the HIP layer converts the multicast group address to SID for legacy applications
  - new applications should be written to use the HIT_S and SID explicitly
- To join a multicast group, hosts use a group management protocol, such as IGMPv3 for IPv4 and MLDv2 for IPv6

Host Identity Specific multicast (cont.)

- A new Version Independent Group Management Protocol (VIGMP)
  - for HISM hosts that carries host and session identifiers but is backward compatible with IPv4/v6 ASM and SSM multicasting
  - the advantage of version-independent approach - all hosts need one multicast implementation, independent of the current type of network connectivity
  - VIGMP messages include the HIT_R that is used for authentication

Host Identity Specific multicast (cont.)

- A new Version Independent Group Management Protocol (VIGMP)
  - for backward compatibility, receivers can join the multicast tree by specifying IP addresses of the group and the source
  - the authentication server then maps addresses to source and session identifiers
  - the router broadcasts the multicast query to locate potential listeners
  - interested and subscribed hosts reply with a report containing identifiers of the session, own host, and multicast source

Host Identity Specific multicast (cont.)

- An VIGMP-capable router can inform receivers of changes
  - the multicast parameters in a router session report message
  - multicast source moves - new unicast IP address is announced to the listeners through report message
- The HISM authentication mechanism
  - an extension of Multicast Control Protocol (MCOP)
  - authentication of host based on HIT_R, not just network subnets
  - the receiver host can perform the HIP BE to the multicast source if mutual authentication is required
  - for larger multicast groups, the overloading of the sender is possible - additional authentication infrastructure is necessary

Host Identity Specific multicast (cont.)

- The first-hop multicast router (Designated Router, DR) receives a join request from the user host
- Forwarding of the message to the authentication server
- The policy database at the server contains active multicast session identifiers with source HIT and IP, and a list of authorized receiver HITs
- Multicast sources register their host and stream identifiers with the multicast server - creating a new multicast stream
- The mapping between the source HIT and its IP is maintained in DNS
- The HIP-specific DNS resource record includes the name of the RVS and can be extended to include HISM-specific fields
- The use of DHT for storing the mapping information - for quickly moving source

Host Identity Specific multicast (cont.)

- The authentication server maintains a list of dual-stack border routers
  - the routers are able to convert between IPv4 and IPv6 multicast packets
  - the authentication server adds dual-stack routers to an anycast group identified by an anycast IP address
  - sending message to the group - only the closest router will receive it
  - after successful authorization - the authentication server sends the IP address of the source and the address of the dual-stack router to the user's designated router
  - designated router forwards the message to the user's host

Host Identity Specific multicast (cont.)

- Active session - the authentication server periodically checks if the IP address of the multicast has changed
  - change is detected - the server informs the Designated Router
  - the DR forwards the new source IP to all receivers using an VIGMP message
  - the authentication state at the DR and the server eventually times out
  - requirement to the multicast receiver to re-authenticate periodically
  - soft-state approach accommodates the presence of mobile hosts that can leave the network without removing the multicast subscription
- HISM routing is based on Protocol Independent Multicast, Sparse Mode (PIM-SM)

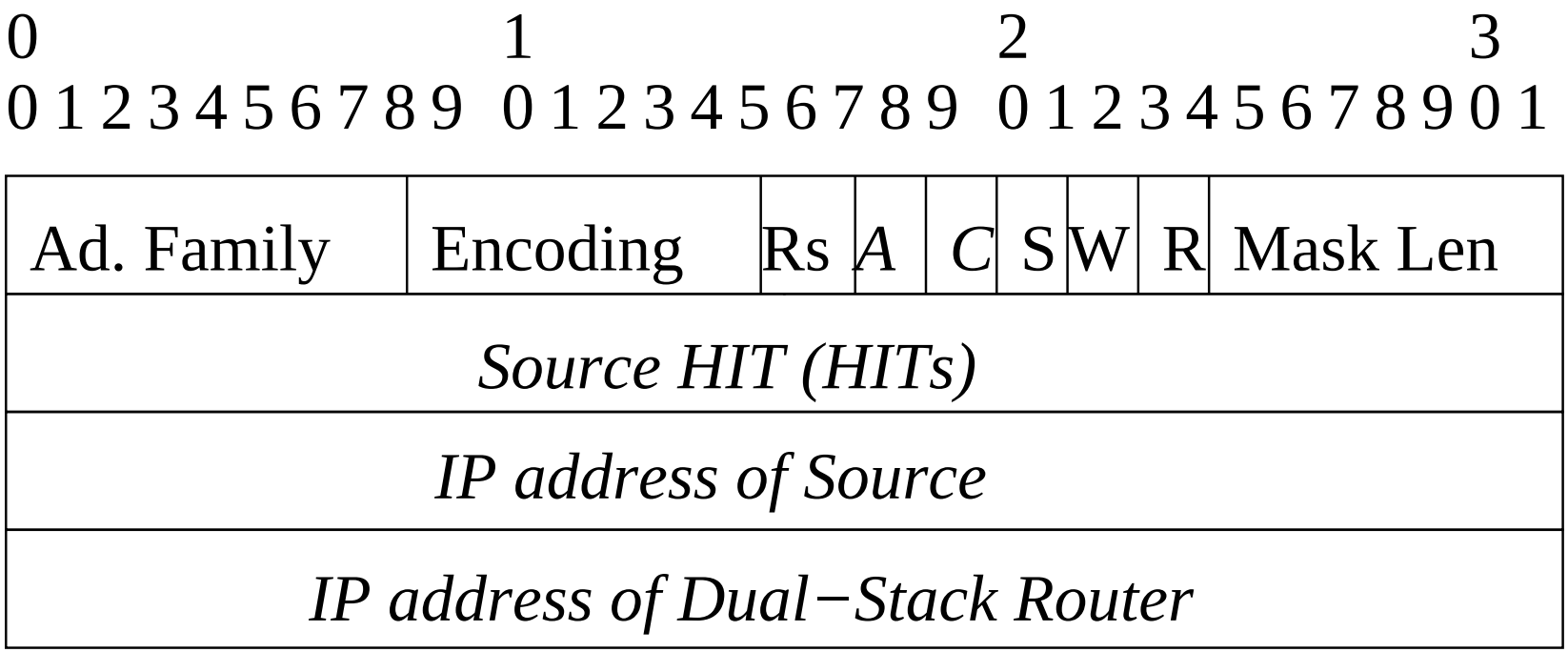
Host Identity Specific multicast (cont.)

- The DR plays a significant role in construction of the tree by authenticating the user
  - several border multicast routers are present - only one is selected as DR
  - avoiding of multiple versions construction of the multicast tree
  - the DR receives a join message form the user host and authenticate it
  - the DR sends the PIM join message upstream toward the source
  - the message includes two new flags:
    ”C” for requiring IP version conversion
    ”M” for indicating a change of the source IP for mobility

Host Identity Specific multicast (cont.)

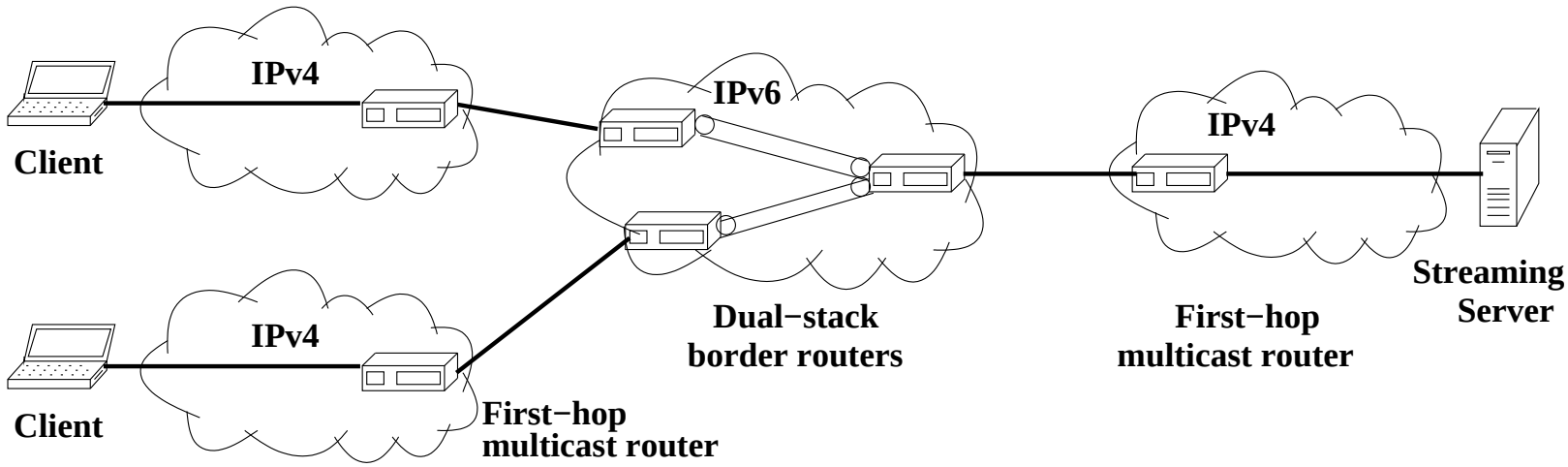
- The core multicast routers store the forwarding state based on source HIT, not IP address
- The routing record has the form
  $HIT_S|SID|Input\ interface|Output\ interface$
- The routers do not have the capability to resolve HITs to IP addresses
- Using HITs prevents tree reconstruction in the case of source mobility and enables construction of dual-version multicast trees
- The IP addresses are only used during the tree construction phase

Encoded source format for HISM



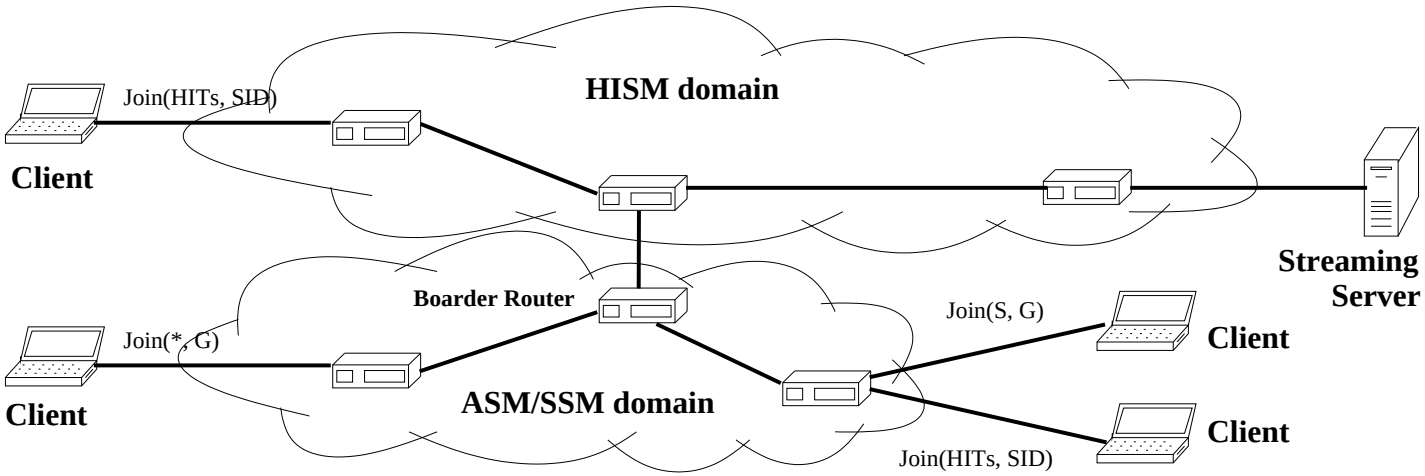
- the content of encoded source format
- includes HIT_S, IP address of the source, IP address of dual-stack router
- the new HISM-specific fields are in *italics*

Tunneling solution for traversal of IPv6 domains for IPv4 multicast



- the transit network has a different IP version than the source and receivers - unicast tunnels can be constructed to pass through the transit network
- the possibility of multiple packet copies going over the same link
- prevention of receivers within the transit network from joining the multicast tree
- HISM enables native multicast trees without tunneling

Mixed multicast tree with HISM and legacy SSM/ASM parts



- legacy ASM and SSM hosts can possibly join the HISM tree
- a flag or an option in Hello message indicates the HISM capability of border routers
- the border routers convert between legacy ASM/SSM join messages and HISM messages by inserting or removing HIT_S and other fields
- tunnel through the legacy network to the next HISM-capable border router
- the use of tunnel is a must if dual-stack border routers perform IP version conversion

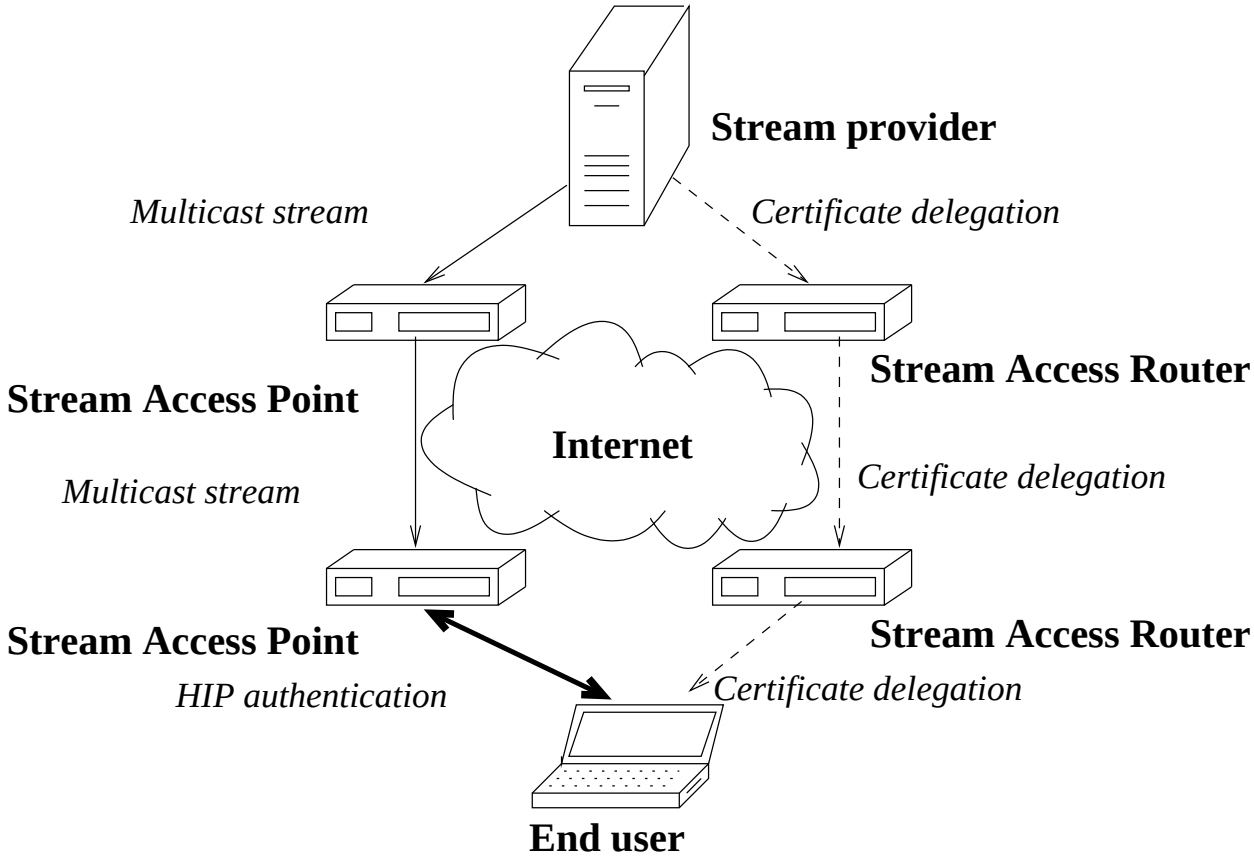
Authenticating multicast receivers

- IP multicast - the source host does not know how many receivers it currently has
- It is not possible for source host easily offer commercial services requiring accounting and payment
- Deploying IP multicast is a challenge for Internet Service Providers due to business models
- A provider that forward more traffic outwards than inwards has to compensate the difference with payments to its peer providers
- The IPv6 multicast is still under development with trends towards SSM
- IPv6 multicast requires deployment of IPv6 that is not global yet

Authenticating multicast receivers

- An architecture proposed by Jokela
- The multicast source encrypts the data and retailers authenticate multicast receivers with HIP
- The architecture uses Simple Public Key Infrastructure (SPKI) certificates
  - the *issuer* - the public key of the stream provider who creates the certificate
  - the *subject* - the public key of the certificate user
  - *Delegation* - a Boolean flag, allows or disallows further use of the certificate beyond the subject
  - *Authorization* - the actions that the subject is entitled to perform
  - *Validity* sets the earliest and latest dates when the certificate is valid
  - *Signature* is created with the private key of the issuer

Authenticating multicast receivers with HIP



Authenticating multicast receivers with HIP (cont.)

- *Stream Provider* transmits an encrypted multicast stream and creates certificates authorizing its access
- *Stream Access Retailer* assists the Stream Provider in creating certificates for receivers
- *Stream Access Point* is a multicast router in the network that verifies the certificate of the receiver and adds it to the multicast distribution tree
- *End user* obtains a certificate from the Stream Access Retailer after a payment and presents it to the Stream Access Point for receiving the multicast transmission
- the architecture assumes that the Stream Provider trusts Stream Access Retailers and Access Points to handle the authorization and accounting of users correctly

Authenticating multicast receivers with HIP (cont.)

- after obtaining the certificate, the end user executes the HIP BE to the local Stream Access Point
- the end user supplies the certificate using HIP CER packet
- the Stream Access Point checks that the user's public key is the same as in the certificate
- the certificate also includes the stream group ID (the stream that the user is subscribing to)
- after authenticating the user, the Stream Access Point can join the multicast tree for the stream and starts delivering it to the user
- to prevent the user from sharing its identity with other users, the Stream Access Point contacts the Stream Access Retailer to check if the certificate is still unused and mark it as used afterward
- data is sent encrypted from the Stream Provider, which periodically changes the decryption key

Performance measurements

Performance measurements

- Current trend of moving mobile telecommunications systems to IP technology
- The security aspect of using IP protocol stack on lightweight devices is not sufficiently explored
- Encryption and public key signatures are computationally expensive operations
- The possibility of stressing CPU and battery resources of mobile devices
- The data throughput and latency can be negatively affected

HIP on Nokia Internet Tablet

- Nokia 770 Internet Tablet - Linux-based PDA
- Measurements of port of HIP for Linux (HIPL) implementation to the Nokia 770
  - previous evaluation of HIP on standard Internet hosts
  - a HIP assessment on mobile devices with restricted resources
- HIP measurements over WLAN
  - Nokia Tablet as a mobile client, mainly as Initiator of HIP association
  - data throughput, latency and power consumption on the HIP BE and MU
  - analysis of results and suggestions

HIP on Nokia Internet Tablet (cont.)

- The choice of Nokia 770 as a target device
  - PDA with a set of limited resources - good example of lightweight hardware
  - mobile client - HIP mobility issues
  - attracts more and more users as well as developers due to number of applications (VoIP, Audio, Video an Demand, etc.)
  - applications might utilize the benefits of HIP
  - Linux-based PDA - any software porting is much easier on the open source platforms

HIP on Nokia Internet Tablet (cont.)

- Nokia Internet Tablet
  - high-resolution touch screen display
  - built-in WLAN, Bluetooth support
  - designed for easy web browsing
  - convenient for Internet telephony and instant messaging, reading emails and documents, playing media content
  - Texas Instruments (TI) OMAP 1710 CPU, 220 MHz, 64 MB DDR RAM
  - 1500 mAh Li-Polymer battery
  - Internet Tablet OS 2006 edition, 2.6.16 Linux kernel
  - GNOME-based graphical user interface

HIP on Nokia Internet Tablet (cont.)

- Porting HIP to the Nokia 770 Internet Tablet
  - HIPL implementation was used (HIIT)
  - HIP daemon and the other utility programs of HIPL are userspace applications
  - a few modifications to the Linux kernel are necessary in order to support HIP
  - three patches have to be applied to the Nokia kernel
  - Nokia kernel must have IPv6, IPsec, AES, 3DES, and SHA-1 support

HIP on Nokia Internet Tablet (cont.)

- Porting HIP to the Nokia 770 Internet Tablet
  - low computational power makes it impossible to compile big software projects as well as Linux kernel directly on the PDA
  - Scratchbox cross-compilation environment, located on a PC
  - flashing custom kernel image
  - userspace applications - Debian binary file (*.deb)

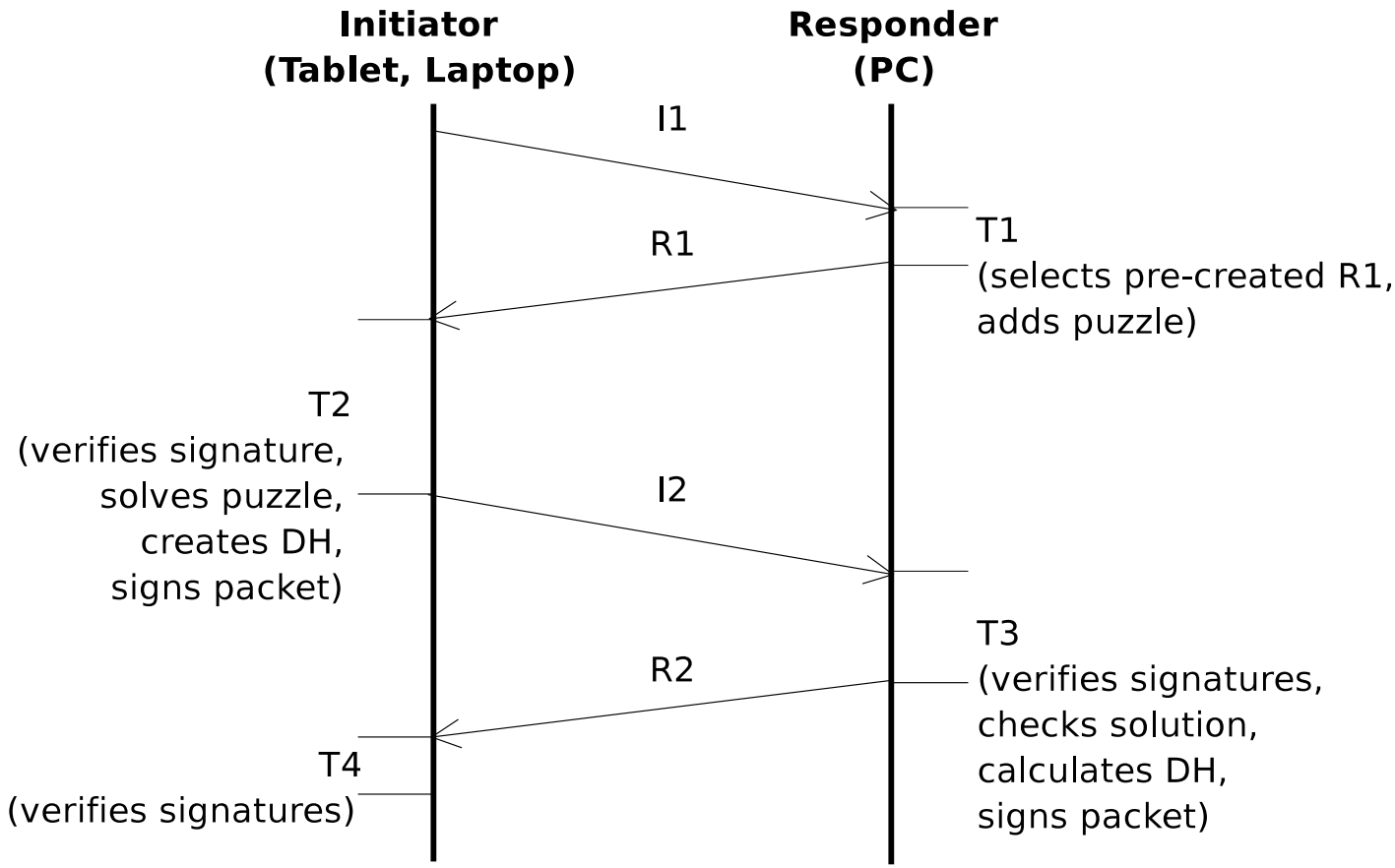
Test environment

- Nokia 770 Internet Tablet (Tablet)
- Intel Pentium 4 CPU 3.00 GHz machine with 1GB of RAM (PC)
- Switch and WLAN access point (AP)
- Network provided both IPv4 and IPv6 addresses
- The wireless AP supported IEEE 802.11g standard and WPA (Wi-Fi Protected Access) encryption
- All communications parties - the same implementation of HIP
- Repeated measurements on 1.6 GHz, IBM laptop (Laptop) - to better indicate the Tablet's performance level

Basic HIP characteristics, duration of HIP BE

- Determination of the duration of various BE stages
  - generating and processing of HIP messages by the Tablet
- A script established a HIP association 50 times
- A number of varying scenarios
- Significant differences between IPv4 and IPv6 performance were not found
- Only results with the RSA HITs mapped to IPv6 addresses were presented

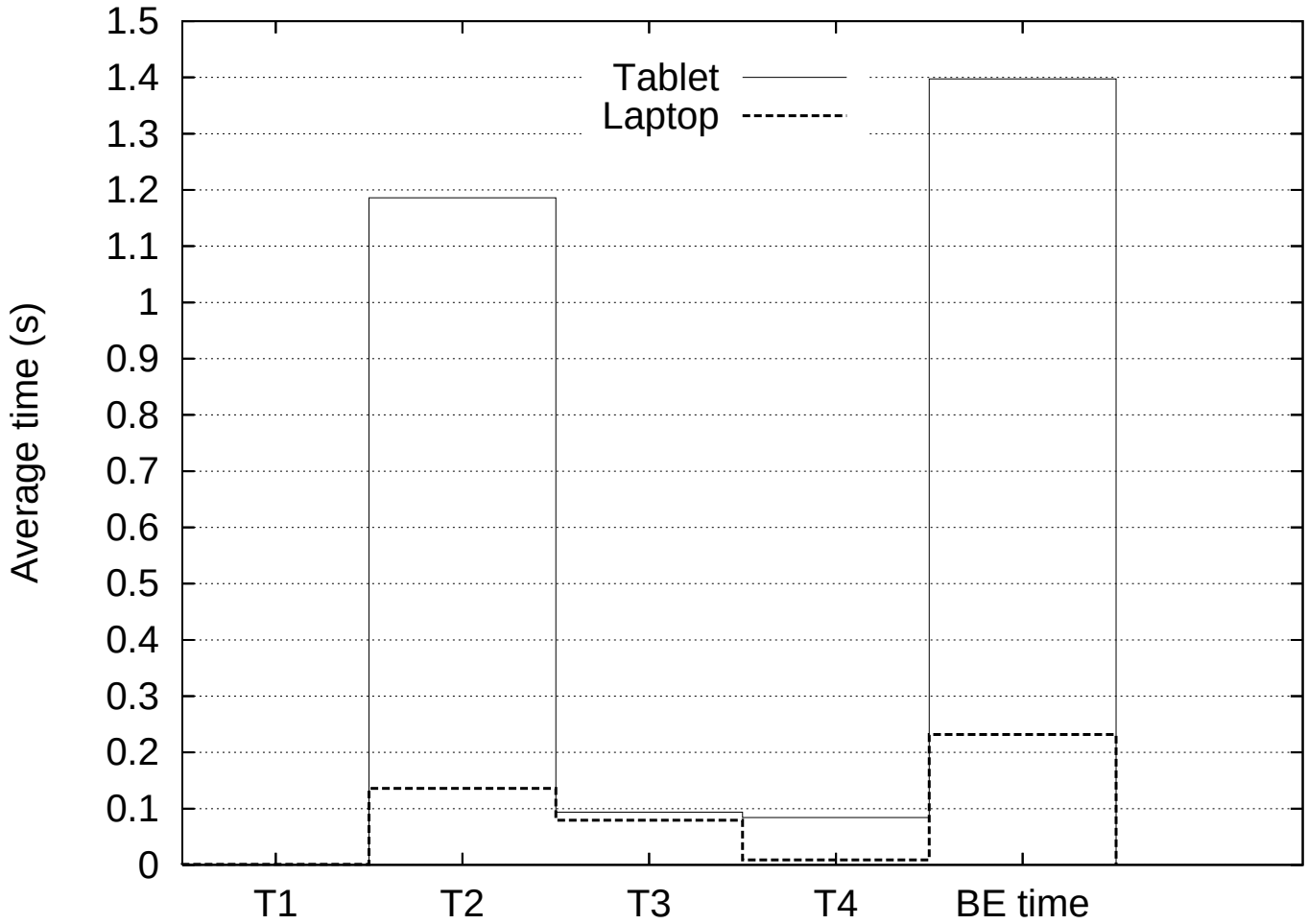
The times measured on the Initiator and the Responder



The times measured on the Initiator and the Responder (cont.)

- the I1 packet generation was not considered due to it insignificance
- T1 - the time for the Responder to process an I1 and generate an R1
- T2 - the time for the Initiator to process the R1 and generate an I2, contains a number of CPU-intensive cryptographic operations
- T3 - the time needed by the Responder to process the I2 and generate an R2
- T4 - the Initiator processes the R2 and completes the BE

Duration of HIP Base Exchange stages for Tablet and Laptop



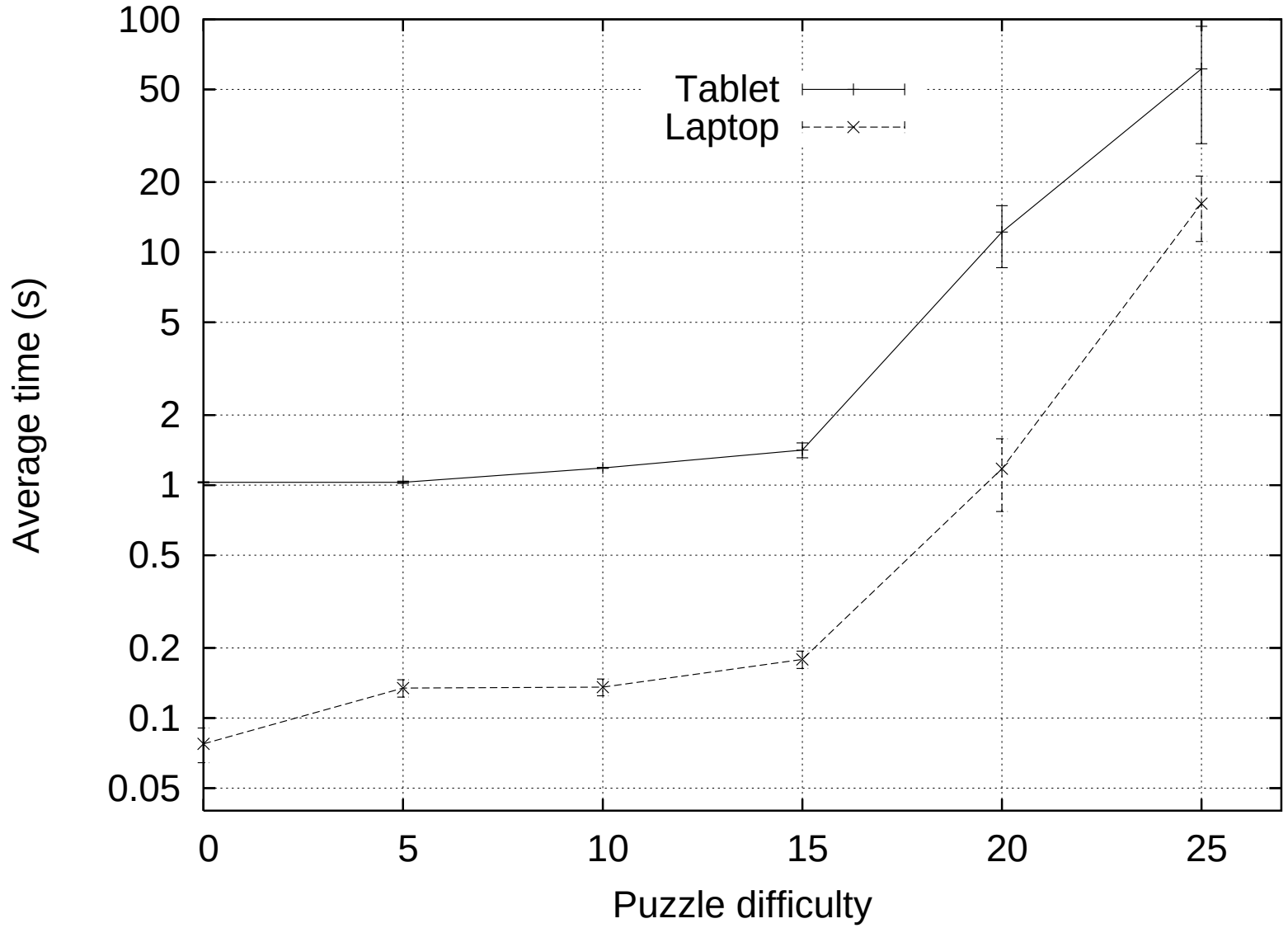
Duration of HIP Base Exchange stages for Tablet and Laptop (cont.)

- two HIP associations: the Initiators are Tablet and Laptop, the Responder is PC
- T1 and T3 - times are measured for the PC
- T2 and T4 - correspond to both Tablet and Laptop
- the Laptop greatly outperforms the Tablet for all operations; T2 for the Tablet is nearly 1.2 sec., for the Laptop is 0.14 sec
- BE time is independent of whether PC or Tablet initiates the handshake
- BE for a Tablet-to-Tablet scenario is over 2.6 sec.
- Tablet spends a similar amount of time for T2 and T3 phases

Puzzle difficulty

- The Initiator solves a cookie puzzle upon receiving an R1
- Puzzle protects the Responder against DoS attacks by compelling the Initiator to spend a certain amount of CPU cycles
- The Responder's opportunity to adjust the puzzle difficulty
- The difficulty (K) - a number of bits that must match in a hash output send back to the Responder

Processing time vs. puzzle difficulty



Processing time vs. puzzle difficulty (cont.)

- the puzzle difficulty and BE dependency for the Tablet and the Laptop
- the time needed to solve puzzle grows exponentially with increasing its difficulty
- setting a high value of K for the Tablet would not be possible - Tablet's CPU will take a long time to solve such puzzle
- puzzle difficulty of 20 keeps Tablet's CPU busy for over 10 sec.
- puzzle difficulty of 20 keeps Laptop's CPU busy for 1.3 sec.
- balancing between the puzzle difficulty and the time limit

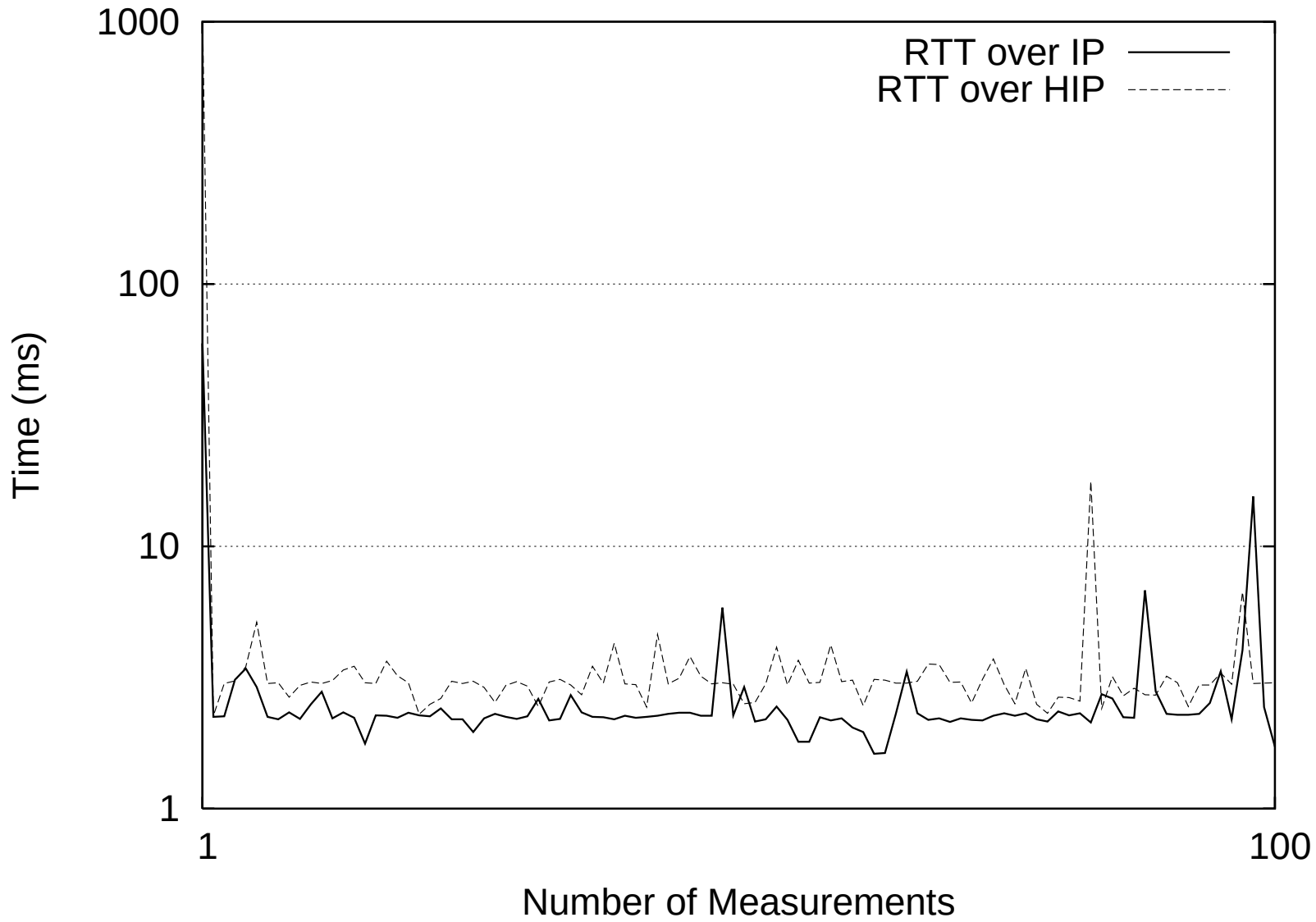
RTT

- RTT (Round Trip Time) equals the time for a packet to travel from a node across a network to another node and back
- Evaluation of the effect of using HIP on RTT for applications
- Ping6 tool - 100 ICMP messages over HIP and over plain IP
- Scenarios include Tablet, Laptop and PC as HIP communications entities

Average RTT for Tablet and Laptop

RTT	Mean (ms)			Standard deviation (ms)		
	IPv6 (64B)	IPv6 (116B)	IPv6/HIP (ESP)	IPv6 (64B)	IPv6 (116B)	IPv6/HIP (ESP)
PC→Tablet	2.223	2.358	2.936	0.470	0.425	0.931
Tablet→PC	1.901	1.900	2.748	0.332	1.235	1.347
PC→Laptop	1.026	1.049	1.177	0.340	0.312	0.243
Laptop→PC	1.065	1.070	1.207	0.338	0.427	0.502

Round Trip Time, PC as the Initiator



Round Trip Time, PC as the Initiator (cont.)

- first RTT appears to be high because of the BE and ARP query
- on average, HIP raises the latency by $34 - 45\%$
- the default size for ICMP message is 64 bytes
- with HIP size is augmented by ESP headers and amounts to 116 bytes
- the RTT times for a plain ICMP of the size 64 and 116 were measured
- the RTT time for an ESP encapsulated ICMP packet was measured
- increase of the ICMP packet size affects the latency by a small factor (6%)
- HIP increases the RTT value
  for PC-to-Tablet connection on $\approx 37\%$

  for PC-to-Laptop connection on $\approx 15\%$

Throughput

- IPsec ESP data encryption can reduce the maximum achievable data rate
- An iperf tool - TCP packets generating
- WLAN Access Point - data encryption by means of WPA protocol
- Evaluation of ESP and WPA overhead

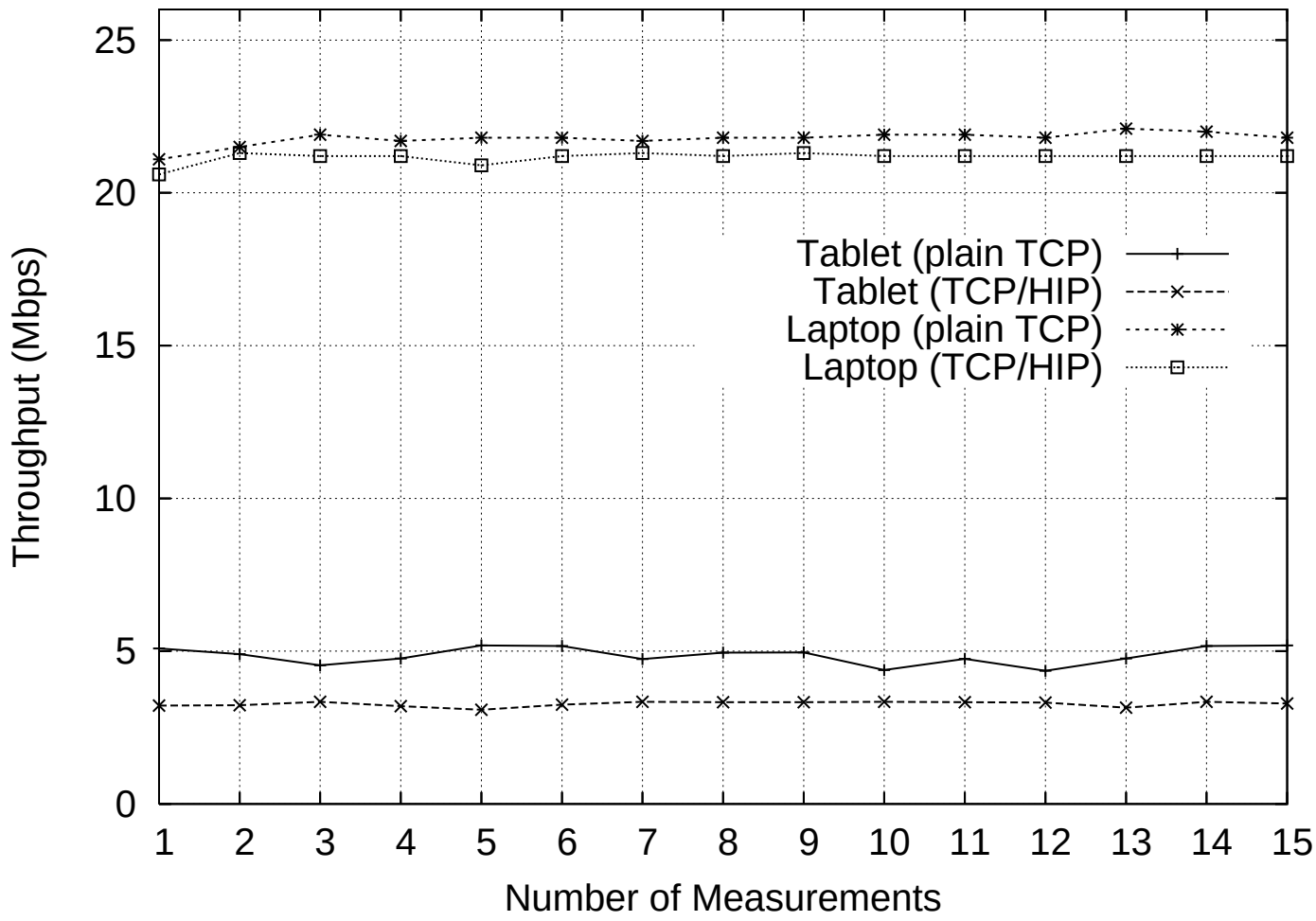
TCP throughput in different scenarios

Throughput	**Mean/Standard deviation (Mbit/s)**				
	TCP	TCP/HIP	TCP+WPA	TCP/HIP	+WPA
Tablet→PC	4.86/0.28	3.27/0.08	4.84/0.05	3.14/0.03	
Laptop→PC	21.77/0.23	21.16/0.18	-	-	

TCP throughput in different scenarios (cont.)

- 4.86 Mbit/s represents the upper bound of the throughput achievable by the Tablet (plain TCP/IP, no encryption)
- the Tablet's specification claims supporting IEEE 802.11 b/g - maximum data rate of 54 Mbit/s
- WPA encryption makes a minor impact on the throughput - reduces the data rate only by 0.4 %
- the ESP influence is much stronger - reduces the data rate by 32 %
- mutual impact of WPA and ESP is bigger
- the Laptop achieves 21.77 Mbit/s over the plain TCP/IP with open wireless link
- the impact of ESP encryption equals 3 % of the throughput decrease

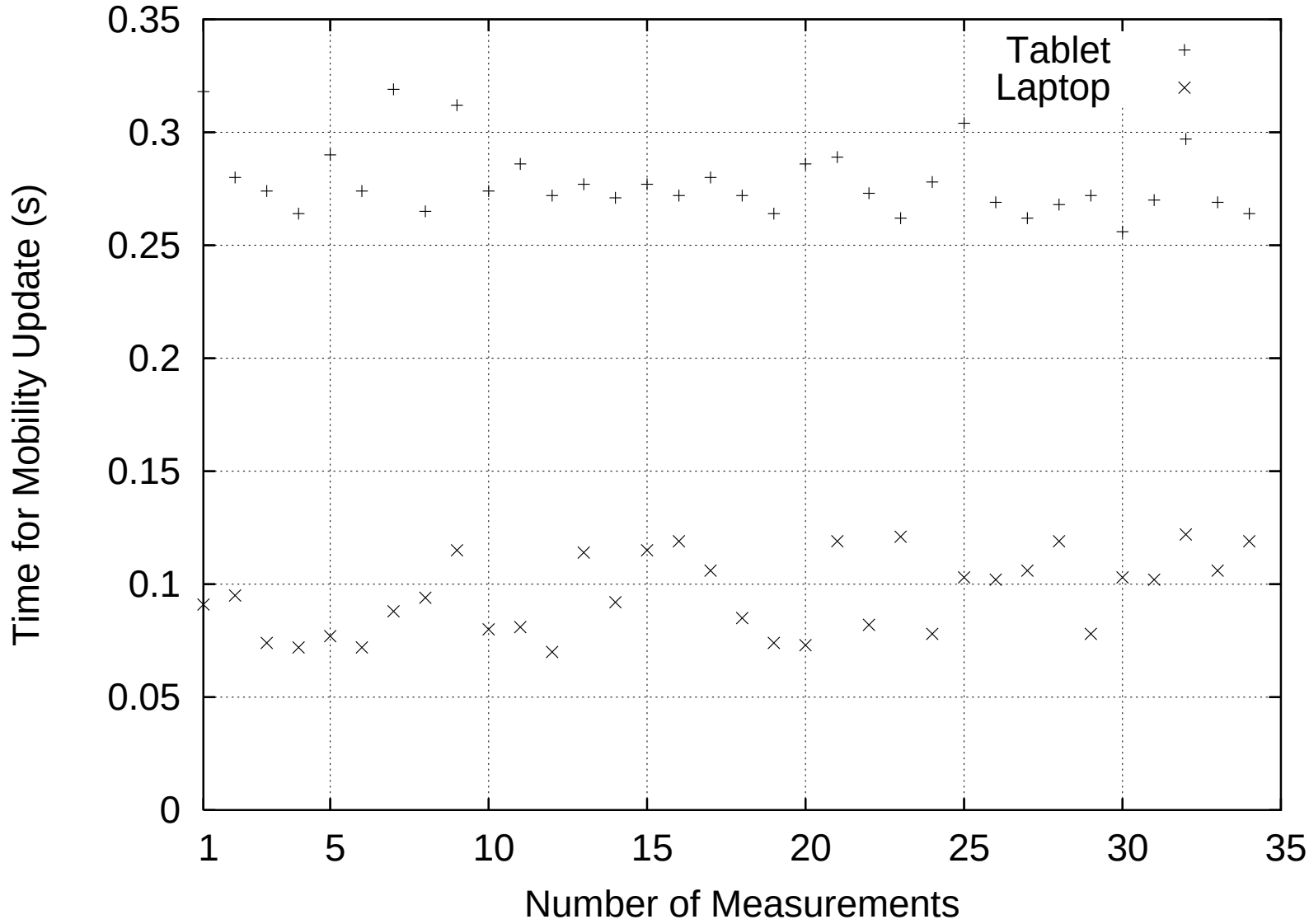
TCP throughput in an open wireless network



Duration of Mobility Update

- The time to exchange MU packets was measured
- Manual changing of the IP address
- The tests were repeated 50 times

Duration of Mobility Update (cont.)



Duration of Mobility Update (cont.)

- the average duration of MU between the Tablet and the PC is 287 ms
- in reality, the delay can be lower for application
- there is a possibility to send data packet after first UPDATE
- Credit-Based Authorization (CBA)
- CBA limits the transmission rate to a new IP - prevention of hijacking of arbitrary IP address
- the average time for generating, sending and processing the first UPDATE packet is around 20 ms
- the Laptop completes the MU in 100 ms

Battery lifetime

- Power consumption - a crucial issue for any portable device
- The capacity of the Tablet's battery keeps the device in standby mode for a few days
- Battery resources exhausted quickly by applications requiring data transmission over WLAN
- Assessment of how expensive the HIP operations might be in terms of power consumption
- External multimeter
- Measurements of the battery's current
- Computation of a theoretical time of battery depletion
- Empirical assumption about the lifetime

Power consumption by applications

Application/Mode	Current (A)
HIP Base Exchange	0.36
ESP traffic (iperf with HIP)	0.38
Plain TCP (iperf without HIP)	0.38
Video stream from a server	$>$ 0.50
Local video	0.27
Audio stream from a server	0.40 - 0.50
Local audio	0.20
Browsing (active WLAN)	0.35 - 0.50
Passive WLAN	0.12
Activating screen	0.12 - 0.14
Standby mode	$<$ 0.01

Battery lifetime

- with HIP, the average current is 0.38 A
- a fully charged 1500-mAh battery kept the Tablet working for about 3.5 hours
- almost no difference in power consumption between the HIP-enabled and non-HIP applications
- establishing a HIP association, MU, and ESP encrypted traffic consume a similar amount of the current (0.36-0.38 A) equivalent to a plain TCP/IP data connection
- low computational power of the Tablet's CPU - is kept busy all the time upon transmitting data over WLAN
- HIP consumes more power than the non-HIP applications if the data throughput is compared
- in terms of power consumption the use of HIP involves longer CPU utilization and consequently more energy consumed for a task

Summary

- The unmodified HIP may be used in scenarios where a lightweight device communicates through a single proxy server in the Internet
  - a HIP associations establishment is 1.4 sec and MU is 287 ms
- Scenarios with two mobile hosts or multiple parallel associations - unmodified HIP is too heavy for lightweight devices
  - HIP associations establishment for two Tablets is 2.6 sec
- The use of WPA encryption has negligible effect on throughput, while ESP encryption with HIP reduces the throughput
  - the Tablet achieves 4.86 Mbps in a WLAN without HIP
  - with ESP encryption the throughput equals 3.27 Mbps

Summary (cont.)

- The RTT over WLAN is only several milliseconds. HIP increases the RTT by a few milliseconds
- The use of ESP encryption with HIP does not affect the battery consumption in the Tablet. The energy cost per byte is higher with HIP
  - the Tablet CPU is always fully utilized when an application transmits data over WLAN that depletes the battery in 3-4 hours
- Application of the measurement results to a wide range of mobility and security protocols
- Motivation for proposing Lightweight HIP
- Future work - comparison of LHIP and HIP on newer Nokia N800, HIP implementation on the Symbian OS platform

Lightweight HIP[3]

[3]Based on work contributed by Tobias Heer, RWTH.

Security functionality of HIP

- HIP uses several cryptographic algorithms and protocols
- Secure communication and tamper-proof protocol functionality are provided by HIP
- CPU-intensive calculations
- Performance analysis of cryptographic functions in HIP
- The need of lightweight version of HIP for CPU-poor devices

Performance limitations of HIP

- Cryptographic components of HIP significantly slow down the BE and MU
- The duration of HIP processes depend on the public key
  - the Initiator and the Responder have to calculate one signature with their own private key and one signature verification with the public key of the peer during the BE
- The computation of the Diffie-Hellman shared secret
- No payload traffic between the two hosts before the BE is completed
- CPU is busy during the RSA, DSA, and Diffie-Hellman computation
- The interruptions of real-time applications are possible during computations

Performance limitations of HIP (cont.)

- RSA and DSA signatures are also used during the update process
  - the initiator must sign two packets with RSA or DSA
  - the Responder must sign one message
- The use of HMAC signatures is also possible during update process
- The updating of all host's HIP associations after changing its set of IP is necessary
- Signing several packets increases the delay
- Computational scenario can be appropriate or not
  - applications exchange sensible/valuable data - protection is required
  - HIP slows down applications that do not require such protection

Problem statement

- HIP aims at being a general purpose solution
  - the new namespace and the separation of the IP and the host identity - convenient and efficient way to address hosts regardless of location
- Using HIP is only possible if both hosts support the protocol
- Achievement of widespread deployment - HIP should be applicable to many applications, environments, and platforms
- The performance penalties for all communication flows
- Performance issues limit the applicability of HIP
  - web browsing on mobile devices, with a delay of seconds, can discourage users from using HIP
  - web pages with contents from several servers - several HIP associations at the same time to display the page

Problem statement (cont.)

- Mobile hosts that establish several HIP associations with different hosts can also experience delays of several seconds
- Only few users will find such delays acceptable when it comes to audio streaming or voice telecommunications
- CPU load peaks after mobility events
  - the peaks cause the freezing of all applications for several seconds
- The question arises if the gain of usability due to support of mobility really outweighs the reduces usability due to the HIP performance issues
- Using HIP in P2P networks
  - extremely long delays for location updates
  - applications such as Skype benefit from the HIP mobility support

Problem statement (cont.)

- Many application cases require neither encryption nor authentication on the network layer
  - many web sites do not provide data that requires protection
  - data that requires protection is typically encrypted on higher network layers
  - many applications use their own authentication mechanism
- Many applications do not require HIP-based authentication and encryption services
- Applications can benefit from HIP-based mobility and multihoming support

Problem statement (cont.)

- Sufficient CPU resources of mobile devices to run HIP without delays in the future - solution?
  - increasing CPU frequency - using more energy
  - devices that do not require faster CPUs to perform their basic duties are unlikely to be equipped with a faster CPU just to support HIP
  - faster CPU requires more expensive hardware and more battery resources
  - additional CPU resources are only required during the load peaks caused by the HIP BE and MU
- Long delays might influence the users to favor HIP-less solutions

Problem statement (cont.)

- Limiting the scope of HIP to applications that require both security plus mobility and multihoming will hinder the deployment for small mobile devices
- Hindrance of the deployment of HIP in general - large group of devices that require the features of HIP will not be able to use it without compromises
- Wide support from developers and manufacturers - elevation of HIP from isolated solution for special problems to an omnipresent general purpose solution
- Lightweight version, which provides HIP-like mobility support without CPU-intensive operations - HIP is attractive for users of weak mobile devices

Scope of LHIP

- Performance tests - the public key cryptographic operations consume a considerable amount of time
- First goal for LHIP - replacement of the public-key algorithms in HIP
- Without public key cryptography LHIP cannot provide the same security nor the same functionality as HIP
- The definition of the LHIP scope is important
- The primary goal of LHIP - increase of the HIP performance on CPU-poor devices
  - mobile devices with CPU frequencies below 200 MHz should be able to use LHIP without serious limitation of usability

Scope of LHIP (cont.)

- Without public-key cryptography, authentication is out of scope for LHIP
  - sacrifice of the host authentication for the sake of performance
  - assumption of applications' own authentication mechanisms
- Applications are free to use HIP instead LHIP if authentication is required
  - the attacker can use LHIP to impersonate a host, but it cannot cause harm because of data properties or higher level authentication mechanism
  - it must be possible neither for an attacker to steal an established HIP or LHIP association, nor to impersonate one of the peers

Scope of LHIP (cont.)

- Protection against MITM attackers during the BE - verifying the cryptographic properties of the HIs
- A basic assumption of LHIP - no such attacker is present during the BE
  - one can assume that applications implement measures to identify these attacks if such protection is necessary
- Payload encryption and authentication is also out of scope for LHIP
  - LHIP trusts that the transport layer and the upper layers provide adequate measures to authenticate and encrypt data
- LHIP should provide protection against MITM that take place after the BE
  - assumption that a MITM can be on the communication path after location changes or after multihoming events
  - frequent subnetwork changes should not endanger the security

Scope of LHIP (cont.)

- The concept of decentralized end-host mobility should not be altered
- The way HIP provides its services should not be changed
- LHIP - the same functionality but without authentication and encryption
- Replacement of certain mechanism in HIP
- Introduction of new techniques to gain performance
- LHIP should use the same namespace and the same name resolution infrastructures
- Support of middleboxes and network infrastructure elements in a similar way as HIP
- LHIP must provide basic protection and security

Threat model

- Definition of a threat model is necessary
- Identification of attackers and attacks types that have to be expected for a lightweight version of HIP
- Typical attacker for LHIP:
  - an attacker can receive all packets, including the BE and MU packets
  - the attacker can modify, drop, resend, duplicate, and delay packets
  - the attacker has more CPU resources than the victim. The attacker cannot compute calculations that are considered to be computationally infeasible. The attacker cannot reverse hash functions nor can it forge RSA or DSA signatures
- The LHIP design copes with such an attacker and provides adequate measures against it

HIP high-level goals

- HIP uses a wide range of different cryptographic mechanisms to achieve security, confidentiality, and authentication
- HIP achieves three high-level goals in terms of security
- H1: Payload security
  - the access to payload is restricted to the peers that share the same HIP context
  - two hosts can send data over insecure channels in a secure way
  - it is computationally infeasible for an attacker to decrypt the transferred payload
  - an attacker cannot manipulate the payload unnoticeable

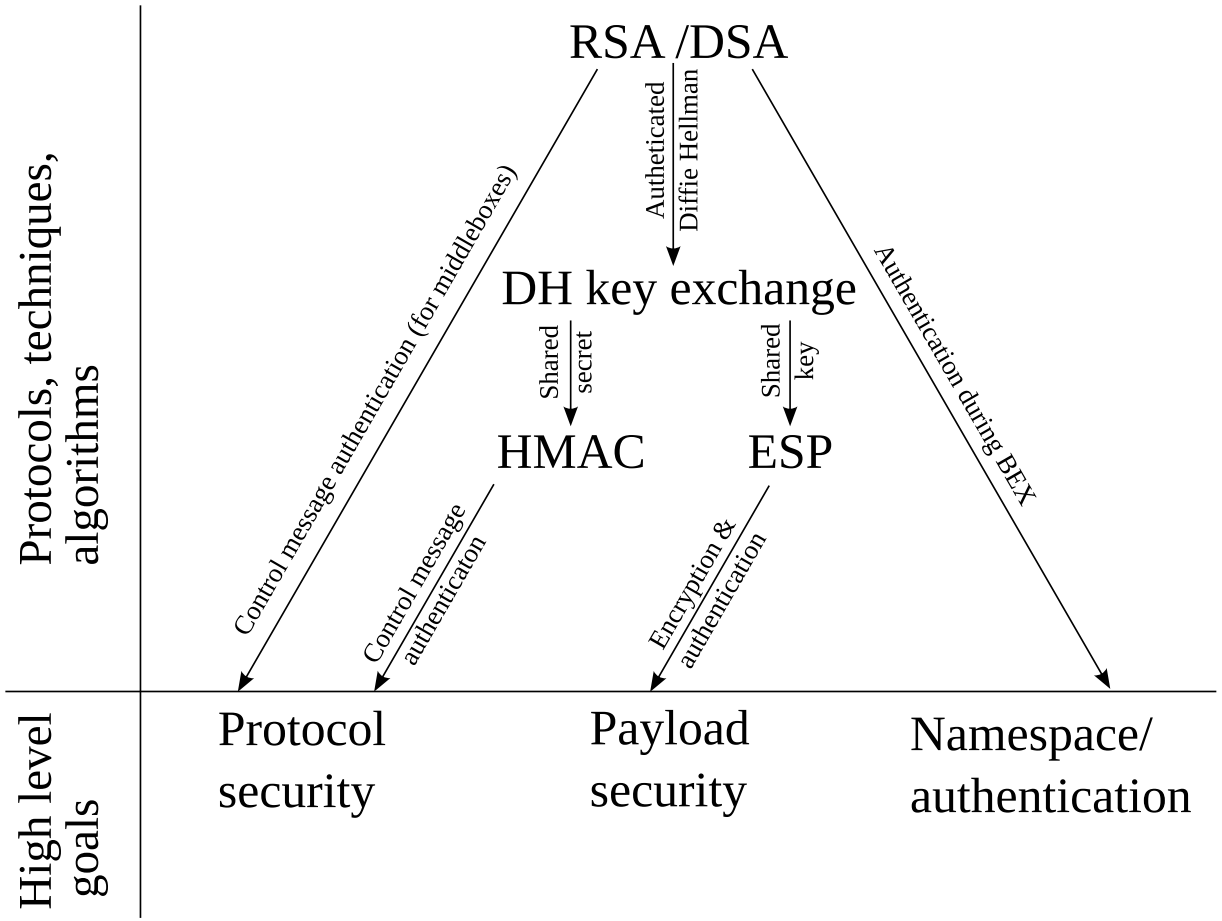
HIP high-level goals (cont.)

- H2: Protocol security
  - communication protocol is not susceptible to malicious interactions with attackers or misbehaving peers and network components
  - an attacker cannot influence the protocol behavior in a way that causes harm to one of the communicating peers or any other third party
  - must be able to prevent and discover accidental or intended misuse that leads to limited functionality
  - protocol security is not restricted to the communication peers only
  - protocol security also takes the communication channel, intermediate nodes

HIP high-level goals (cont.)

- H3: Namespace security
  - trusted communication between hosts
  - provides a way to securely address hosts, using HIP
  - must protect the namespace from intended or accidental misuse
  - HIP must provide protection against identity theft and impersonation

HIP high-level goals (cont.)



- HIP high-level goals and dependencies between techniques, protocols and algorithms that are employed to achieve the goals

HIP high-level goals (cont.)

- For achieving security goals, HIP strongly relies on public-key cryptography
- Enforcement of protocol security by utilizing RSA and HMAC signatures
- Signatures ensures that packet have been sent by the legitimate peer
- The HMAC algorithm requires a shared secret
- The Diffie-Hellman key exchange provides shared secret
- The key exchange is protected with RSA signatures
- RSA signatures are also used in UPDATE packets (middleboxes cannot use the HMAC signatures)
- The goal of *protocol security* is reached by utilizing RSA, HMAC signatures, and Diffie-Hellman key exchange

HIP high-level goals (cont.)

- Payload security for HIP is provided using ESP encapsulation, utilizing symmetric algorithms
- Encryption algorithms require both hosts to share secret symmetric keys
- Keys generated during the authenticated Diffie-Hellman key exchange
- *Payload security* depends on the ESP protocol, the Diffie-Hellman key exchange, and RSA signatures
- *Namespace security* and host authentication depend on RSA signatures
- Signing R1, I2, and R2 with the private key belonging to a host identity guarantees that the host is using its own identifier
- The RSA signature reveals spoofed identifiers

HIP high-level goals (cont.)

- Removing a single public-key component from HIP directly affects the high-level goals
  - HMAC algorithm can only provide protocol security if a shared secret was provided with the Diffie-Hellman key generation
  - statement about the host identity is only possible if the Diffie-Hellman key exchange was authenticated
- Removing the public-key algorithms for the sake of performance leaves HIP without appropriate measures to provide protocol, payload, and namespace security

LHIP high-level goals

- The dedicated goal of LHIP is to reduce the computational cost of HIP
- Retaining the high-level goals H2 - *protocol* and H3 - *namespace security*
- LHIP relaxes the goal H3
  - it is only required that the namespace cannot be used for attacks worse than attacks of plain IP
- Host authentication is not required as long as identity theft during the lifetime of an HIP association is not possible
- High-level goal H1 - *payload security* is sacrificed for the sake of performance
- The level of payload security should still be comparable to plain IP

LHIP high-level goals (cont.)

- Further goals - compatibility with the HIP and upgradability from LHIP to HIP
- Upgradability - ability of hosts to transform an LHIP association into a HIP association
- Transform should be without tearing down and re-opening the association
- Additional goals - new high-level goal *compatibility*

LHIP high-level goals (cont.)

- LH1: Increased performance
  - the performance penalties of public-key cryptography should be reduced to amounts acceptable for use on mobile devices with little CPU power
  - the delays during the BE and the connection disruption during updates are noticed by the users
- LH2: Protocol security
  - must be provided in LHIP
  - attacks that worse than today's attacks on TCP/IP should not be allowed
  - this includes attacks on the communicating peers, third parties, and infrastructure elements

LHIP high-level goals (cont.)

- LH3: Namespace security
  - should be provided to a certain extent
  - attacks must not be able to impersonate a peer during the lifetime of an LHIP association
  - LHIP must be able to deal with namespace conflicts
  - must not yield new ways of attacking HIP or LHIP hosts

LHIP high-level goals (cont.)

- LH4: Compatibility
  - should be provided to a certain extent
  - infrastructure elements should be used by HIP and LHIP in the same way
  - LHIP should resemble HIP and provide the same functionality regarding mobility and multihoming support
  - the message format and the packet contents should no differ much
  - middleboxes should be able to learn about new IP of hosts and new SPI in a secure way
  - LHIP associations should be upgradable to HIP associations during their lifetime
  - compatibility with legacy applications - the same way as in HIP

LHIP high-level goals (cont.)

- Modifying HIP to achieve secure mobility and multihoming for TCP/IP without payload encryption and host authentication
- The HIP dependency - ESP encryption and high-level goal H1 - *payload security* can be removed from HIP without affecting the other high-level goals
- The vector for performance improvements reached by removing payload encryption is limited
- ESP protocol does not influence the long connection establishment delay on weak devices
- A much greater gain in performance can be achieved by reducing the use of public-key cryptography during the BE and update processes

Possible approaches

- Tree simple approaches to reduce the number of public-key operations in HIP
- The approaches show that it is not trivial to speed up HIP
- Violation of one of the desired high-level goals
- Achievement of insufficient performance improvements

Alternative 1: remove Diffie-Hellman

- The first trivial solution - remove the Diffie-Hellman key exchange
- Reduces of the cost of establishing a new HIP association
  - CPU-intensive Diffie-Hellman shared key computations are omitted
- Affects the symmetric cryptography elements and the hashed message authentication codes
  - no shared keys are provided
- A shared key for ESP is not needed in LHIP
  - payload encryption is not necessary
  - ESP can operate in *NULL mode*, which does not require a shared key
- Shared secret is required for calculating the HMAC signatures

Alternative 1: remove Diffie-Hellman (cont.)

- One can also remove the HMAC signatures - they are useless without a shared secret
- HIP uses HMACs and RSA to authenticate HIP control messages
- The security properties of RSA are sufficient to provide adequate authentication
- RSA signatures have to be used - to enable middleboxes to authenticate UPDATE and CLOSE messages
- This approach seems to achieve the desired high-level goals

Alternative 1: remove Diffie-Hellman (cont.)

- RSA signature verification is much more CPU-intensive than HMAC verification
- The absence of the HMAC - the peers cannot check the authenticity of UPDATE messages in a cost-efficient way before calculating expensive public-key operations
- New weakness - DoS attacks
  - both LHIP peers, the weak client as well as its peer, possible a commercial server, become susceptible to attacks aiming at CPU-cycle consumption
  - RSA signatures would always have to be checked without possibility to filter out attacker messages

Alternative 1: remove Diffie-Hellman (cont.)

- An attacker can send invalid UPDATE messages with spoofed HI
  - the attacker can use the HIs of the victim's peers
  - the victim can only filter out forged messages by verifying the RSA or DSA signatures
  - the attacker can send random bits as forged RSA and DSA signatures
- The attacker has an advantage over the victim
- An efficient way to consume large amounts of CPU cycles on the target host
- The first approach cannot achieve the second high-level goal LH2 - *protocol security*

Alternative 2: remove DSA/RSA

- The first approach - an attacker can use unprotected RSA signatures
- RSA signatures cause an unacceptable amount of computational overhead
- Host authentication is not a necessary requirement for LHIP
- Removing RSA from HIP could be considered an option
- A Diffie-Hellman shared secret can be used for host authentication during the lifetime of a HIP association
- The HMAC in the UPDATE messages provides authentication for the communication peers
- Providing weak authentication with temporal separation - an attacker cannot impersonate one of the communication peers

Alternative 2: remove DSA/RSA (cont.)

- Computationally expensive DH is required for creating the shared secret
- Middleboxes have no means of verifying the authenticity of UPDATE
  - no secure option to learn of updated IP addresses or new SPIs
- Limitation of the LHIP usability with packet inspecting middleboxes
  - middleboxes will not support HIP mobility without authenticated updates
- New attacks, targeted at middleboxes
- Completely removing RSA enables attackers to use spoofed HIs
- No means of proving host identity to a remote host
- The high-level goal LH3 - *namespace security* cannot be achieved

Alternative 3: reduction of the key size

- RSA and DH key exchange can use keys of different length
- Short RSA keys - generation of weaker identifiers
- Short DH keys - generation of weak session keys
  - do not offer sufficient security for long-term data security
  - provide short-term session keys for the HMAC authentication
- The potential for improvements is limited to medium-sized keys
- Very short RSA and DH keys are easy to compromise and should not be used
- To lower processing cost of generation signatures and calculating a shared secret medium-sized keys can be chosen

Alternative 3: reduction of the key size (cont.)

- The verification of HIs is still necessary
- The weak device has no influence on the key length selection of its peer
- Server wants to support weak clients and clients with sufficient CPU power
  - has to lower its provided level of security for all hosts
  - must use different identifiers for weak hosts
  - two identifiers - practical problem when identifiers are passed from one host to another
  - complicating the mapping between the host and its identifiers
  - legacy applications have no means of expressing which keys should be used
- Conflict with goal four, LH4 - *compatibility*

Alternative 3: reduction of the key size (cont.)

- One solution is combining several keys into one identifier
- The HIT will be a hash over all these keys
- Key revocation is difficult
- Changing of identifier components is impossible without changing the complete identifier and the HITs generated from them
- Compound identifiers are short lived
- Serious reduction of the host identity namespace usability
- Decreasing the key length would violate goal LH3 - *namespace security* or LH4 - *compatibility*

LHIP approach

- Non-trivial solution is necessary to provide efficient mobility and multihoming in a secure way
- Introduction of the LHIP approach
- Advantages and disadvantages, effectiveness and practical usefulness

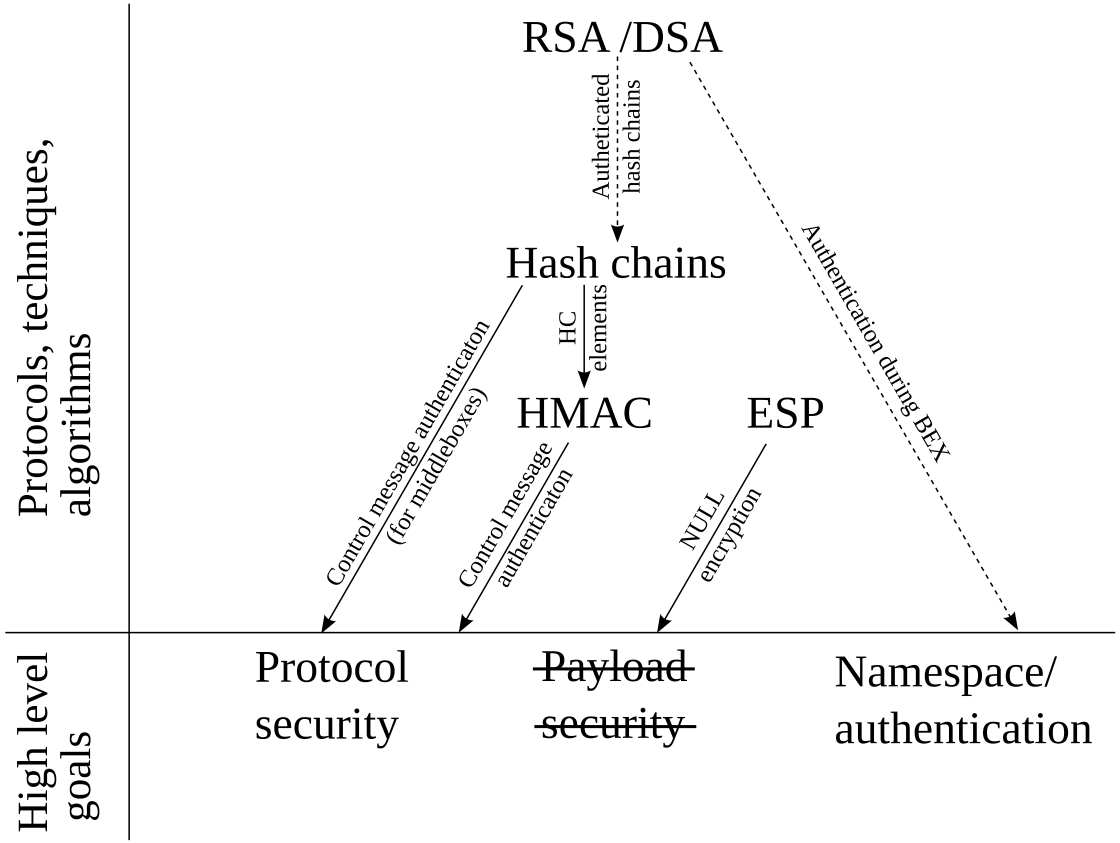
LHIP design

- The design and rationale of LHIP
- High-level goals for a lightweight version of HIP cannot be reached by removing the public-key operation alone
- Alternative authentication mechanism is required for weak devices
- There are no computationally inexpensive algorithms with identical properties
- Hash chains offer an efficient way to authenticate messages without a shared secret
- Hash chains require different application and offer different security properties
- Chains are alternative to DH and RSA in the context of HIP

LHIP design (cont.)

- Thread model - assumption that no active MITM attack is performed during the BE
- This assumption and the relaxed high-level goal LH3 allow to use RSA signatures only in cases of conflicts and namespace violations
- Hash chains signatures provide authentication for LHIP control messages
- Middleboxes can verify the hash chains signatures - no need for RSA signatures
- LHIP can operate completely without computing public-key operations
- LHIP relies on RSA and DSA signatures in the case of namespace conflicts

LHIP high-level goals



- LHIP high-level goals and dependencies between techniques; protocols and algorithms being employed to achieve them. The abstract high-level goals performance and compatibility are not depicted. Optional processes are indicated by dashed lines

Hash chains for HIP authentication

- The basic idea behind hash chain based packet authentication is to let the sender of a message sign the message with an HMAC created with a secret value as the key
- The absence of an encrypted channel to the receiver - the sender must transmit the message, the signature, and the key in cleartext to the receiver to allow signature verification
- To secure signature process, a host only accepts messages as authentic if the signature has been created before the key was disclosed
- This ensures that only the host that was in possession of the secret before it had been disclosed was able to sign the message
- Using hash chains as a source for the keys allows to relate several consequent signatures to the owner of a hash chain

Time-based signatures

- The concept of time based signatures as the concept of interactive hash chains signatures stems
- The *Timed Efficient Stream Loss-tolerant Authentication* (TESLA) protocol uses a time-based approach to sign messages
- The approach requires loose time synchronization between the peers
- The sender sends a hash anchor to a group of receivers
- Each message is protected by an HMAC, which is created with an unrevealed element of the hash chain
- The sender uses each unrevealed element of the hash chain to sign messages for a certain time span
- The element is disclosed after this time span

Time-based signatures (cont.)

- Receivers can authenticate the message as soon as the corresponding element is disclosed
- Messages signed with an element that has already been disclosed are dropped by the receiver to prevent attacks
- The time synchronization ensures that receivers only accept packets while the sender has not disclosed the corresponding hash chain element
- This guaranties that only the sender, who is in possession of the hash chain, can sign the messages
- An attacker can only sign packets after the disclosure of the element
- Other hosts would not accept any more packets signed with the hash chain element

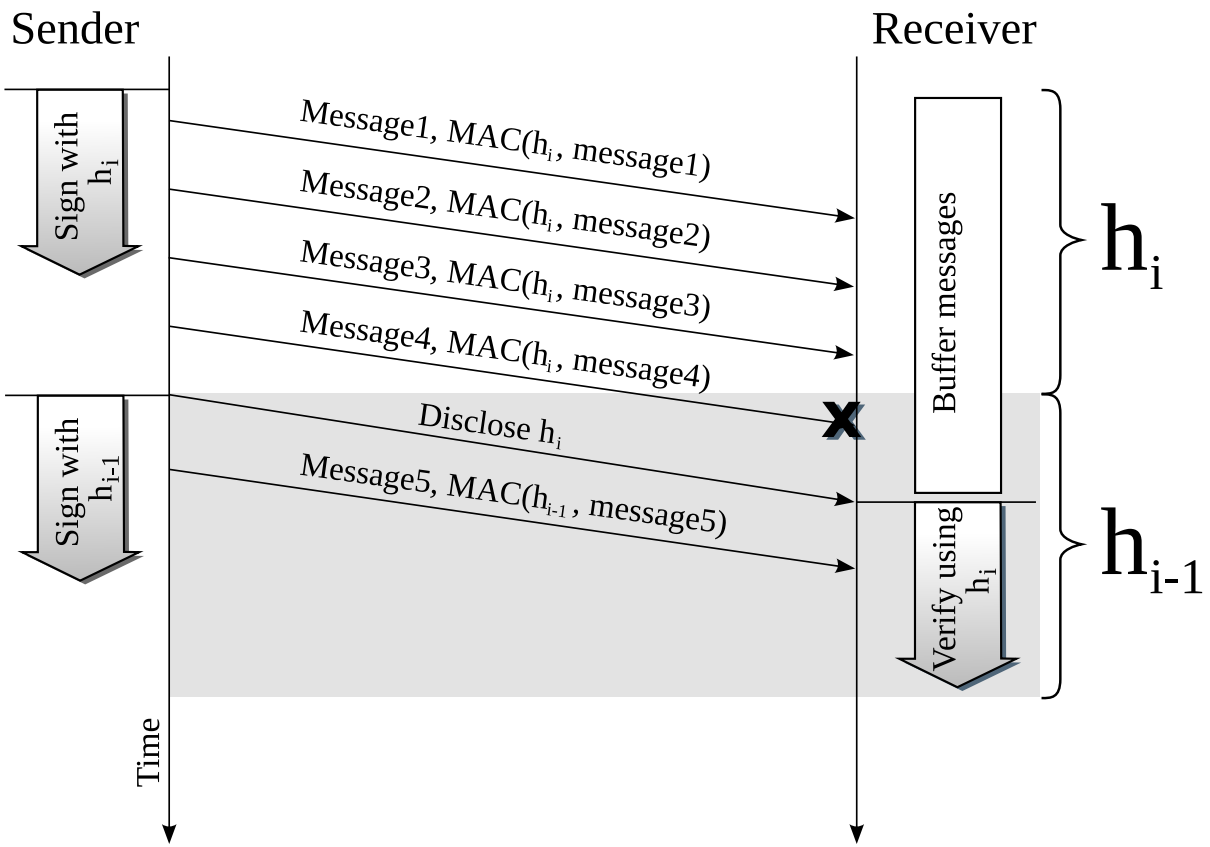
Time-based signatures (cont.)

- When disclosing an element, the next unrevealed element of the hash chain becomes the element with which future packets are signed
- The next element is valid for a fixed interval before it is disclosed
- Time-based method enables TESLA to deliver authenticated messages with little additional overhead
- Signing messages requires
  - the use of one hash chain element per time interval
  - the usual cost for generating and verifying an HMAC

Time-based signatures (cont.)

- TESLA introduces an additional delay
  - a receiver cannot verify the message before the sender disclose the corresponding element
  - the receiver has to buffer the message until then
- Hash elements are revealed even when no signed payload was sent

Signatures with the TESLA protocol

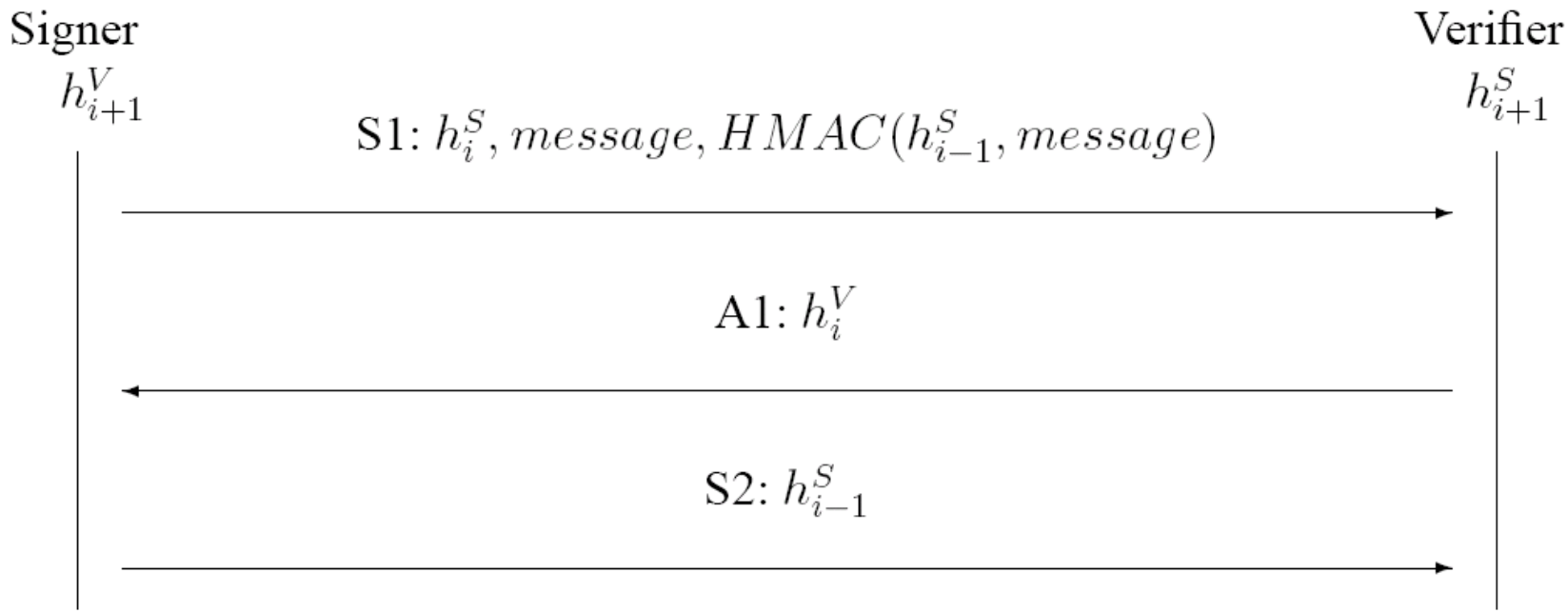


- the sender signs and sends multiple messages
- the receiver verifies the messages after the sender has disclosed h_i.

Interactive signatures based on hash chains

- Torvinen and Ylitalo use an approach that is similar to TESLA to sign messages with hash chains and hash functions
- Undisclosed hash chain elements are used to generate message authentication codes
- A mechanism is required to separate the time span when a hash chain element is used for signing messages from the time for message verification
- The interactive approach relies on a strictly sequential message exchange between the communicating peers

Interactive hash chain signature



- a sender sends a message and an HMAC (S1)
- the verifier sends an acknowledgment packet (A1) and buffers the message
- the signer discloses the corresponding hash chain element in a second signature packet (S2)
- the receiver can verify the buffered message as soon as it receivers the disclosed hash chain element

Interactive hash chain signature (cont.)

- The protocol must fulfill three basic requirements to be secure
  - the verifier must receive S1, signed with a hash element h_i, before the signer discloses h_i
  - the verifier must not accept any messages signed with h_i after the signer has disclosed h_i
  - the signer must not disclose h_i before it receives the A1 from the verifier

Interactive hash chain signature (cont.)

- First requirement
  - only the host, which posses a hash chain, can sign messages
  - an attacker cannot sign messages with the signer's hash chain as it is not in possession of the required undisclosed hash chain element
  - the signer does not disclose its next hash chain element before it has received the signed packet
  - the signer waits for the acknowledgment form the verifier before it sends the next hash chain element
  - the signer can distinguish valid acknowledgments from acknowledgments an attacker could have forged

Interactive hash chain signature (cont.)

- Second requirement
  - an attacker cannot sign any messages with a disclosed element of a hash chain
  - the verifier would not accept any packets after it has received the signature from the signer
  - the verifier must be able to distinguish signature packets from bogus packets that an attacker could send
  - an attacker could constantly send forged packets to force the verifier to deny valid signature packets
  - the interactive approach utilizes hash chains to meet these requirements

Interactive hash chain signature (cont.)

- Second requirement
  - not only the signer but also the verifier uses a hash chain
  - every packet contains a hash chain element as clear text
  - the receiver of an S1 message verifies that the message was sent by legitimate signer by hashing the contained cleartext hash value and comparing it with the last hash value revealed by the peer
  - the signer has sent some message but it cannot verify the message content
  - receiving the signer's hash chain element - the verifier cannot accept any more packets containing this element

Interactive hash chain signature (cont.)

- Third requirement
  - the verifier adds an element of its own hash chain to the A1
  - providing the signer with the knowledge that the verifier has received the signed message
  - it is safe to disclose the next hash chain element in the S2
  - this enables the verifier to verify the message

Interactive hash chain signature (cont.)

- An attacker cannot forge any packets without interaction with the legitimate host
  - the attacker cannot calculate undisclosed elements of the hosts hash chains
  - a MITM attack can be prevented - an attacker cannot sign messages before the verifier has acknowledged the receipt of the signed packet
  - the acknowledgment cannot be forged by an attacker - it contains an undisclosed hash chain element

Interactive hash chain signature (cont.)

- Two hash chains are an unforgeable chain of triggers with each element of the one chain triggering an element of the other chain to be published
- The cleartext hash chain elements in the packets - *trigger values*
- Interactive hash chain signatures do not require any form of time synchronization
- A delay of one additional RTT is induced for every signed packet
- The approach has more overhead than TESLA due to the additional interaction
- The packets containing only trigger values are very small and require a negligible amount of network bandwidth

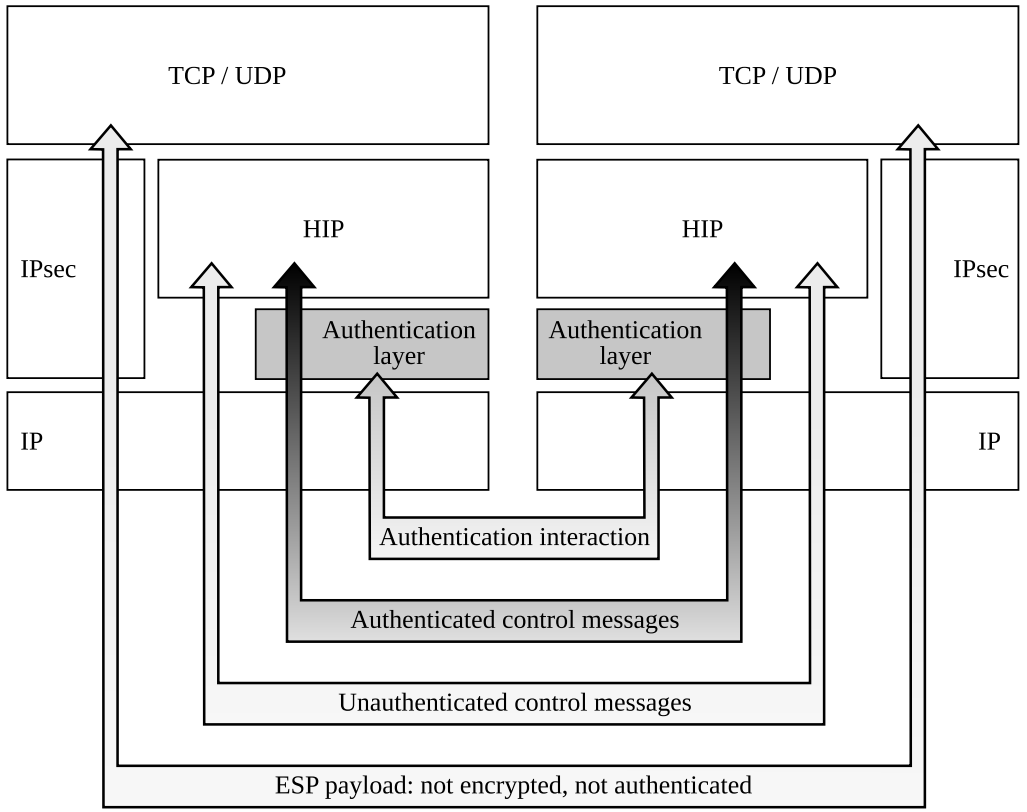
LHIP authentication layer

- Hash chains cannot provide the same level of security
- Hash chains cannot be used in the same way as shared secrets, they can only provide temporary secrets
- Interactive Hash Chain (IHC) signatures differ from regular HMAC
  - message cannot be verified instantly - the receiver must buffer each message until it receives the hash chain element
  - two hash chain elements of an identical hash chain must never be on the wire at the same time - sending signed messages one by one
  - the interaction necessary to delivering a signed message is more complex than for secret shared secret HMACs
- Packet loss, retransmission, duplicate packets, invalid messages must be handled with respect to the security of hash chains

LHIP authentication layer (cont.)

- Integrating IHC signatures tightly into HIP - complicating of the protocol and requirement of complex modifications to HIP
- LHIP creates a new authentication layer, which extends HIP
- The authentication layer is transparent for most of the functionality of HIP
- Reuse of nearly all of the mechanisms that HIP utilizes to provide secure mobility and multihoming
- The isolated authentication layer ensures message authenticity with hash chains
- HIP can create and process packets as if a shared secret was present

The LHIP authentication layer (cont.)



- the authentication layer accepts unsigned control messages from HIP
- the layer determines whether it must apply a signature or not by the HIP header type and the message contents

The LHIP authentication layer (cont.)

- the authentication layer enqueues all messages that require protection and sends them one by one
- the authentication layer takes care of the necessary interaction messages required for IHC signatures
- packets are not passed to the HIP layer but are handled by the authentication layer
- the authentication layer verifies that all signed messages are valid
- the layer handles tasks such as packet retransmission and duplicate packets
- providing of support for undeliverable packets
- payload is sent encapsulated into unencrypted ESP packets

Improved IHC signatures

- Signer (S) initiates the signature process, verifier (V) - its peer
- The IHC approach leaves some space for improvements that provide
  - better resilience against MITM attacks
  - simpler packet handling
  - increased efficiency for acknowledged messages
- Three improvements: dedicated hash chains, pre-signatures, pre-acknowledgments
- The improvements are based on the second preimage resistance and collision resistance of cryptographic hash functions
- Simplifying of the state machine of the authentication layer and avoiding of some security issues

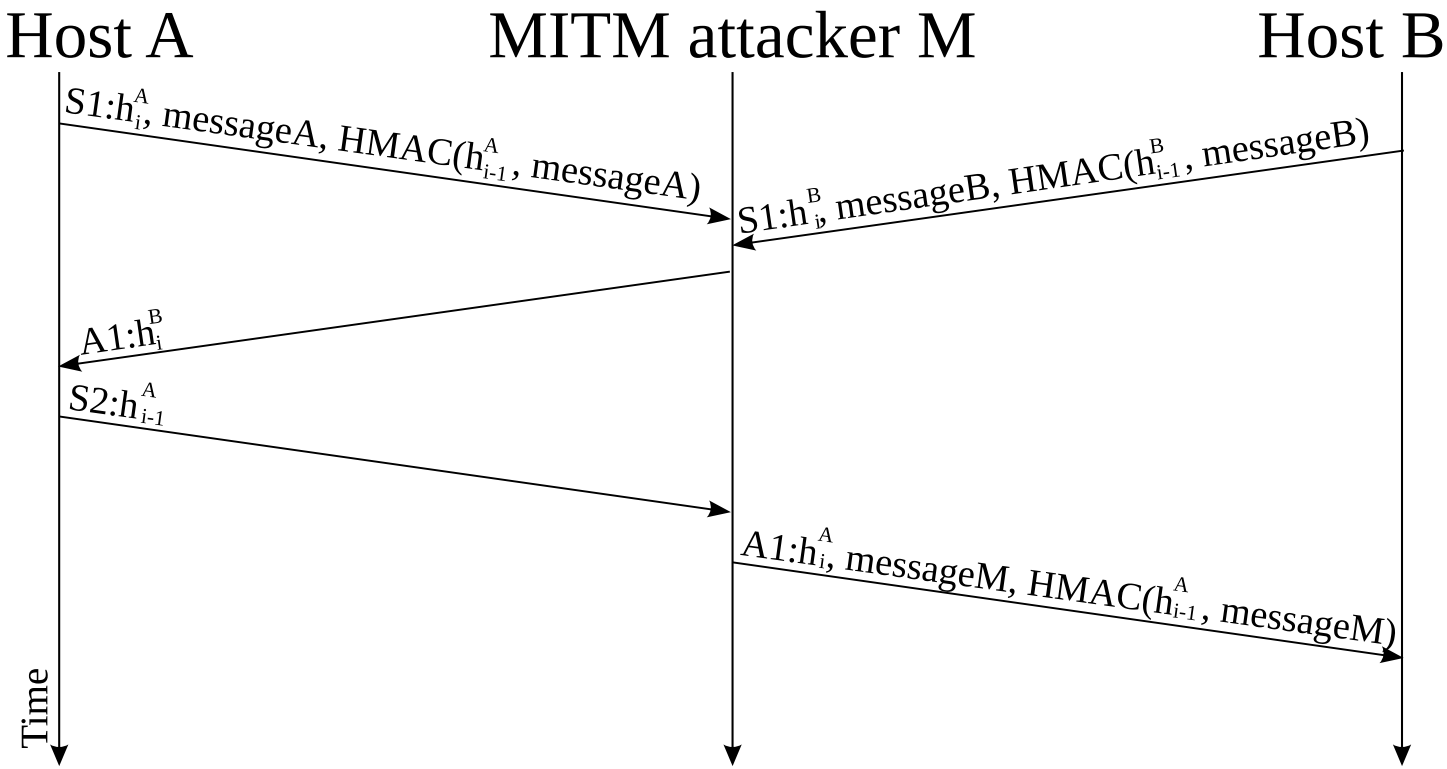
Improved IHC signatures (cont.)

- The difference between the original IHC-based signatures and the improved IHC-based signatures
- Assumption that both hosts use the same cryptographic hash function H
- The output length of H is l bits
- A 160-bit output length is used for practical examples

Dedicated hash chains

- The IHC approach - one hash chain on the signer and one hash chain on the verifier side
- Both hosts can exchange authenticated messages over insecure channels
- The scheme is not secure for mobility updates as they are performed by HIP
- Hosts may change their location at any time
- Hosts may also change their location concurrently
- Each host must be able to send signed messages at any time
- An MITM attacker can mount an attack
  - both hosts send their next hash chain element at the same time due to simultaneous location updates

MITM attack on an IHC signature scheme



- A and B have established a communication context and have exchanged their hash chains anchors h^A_{i+1} and h^B_{i+1} earlier
- both hosts decide to send a signed message at the same time

MITM attack on an IHC signature scheme (cont.)

- a MITM attacker M can delay both S1 packets
- the attacker can use the hash chain element h_i^B to acknowledge A's message before B has received it
- the attacker sends h_i^B to host A
- h_i^B is interpreted as a valid acknowledge that host B has received the first message from A
- A would react to the acknowledgment by disclosing h_{i-1}^A
- the attacker now is in possession of h_i^A and h_{i-1}^A
- B has never acknowledged the receipt of h_i^A
- B would accept message signed with h_{i-1}^A
- the attacker could now generate a valid signature for an arbitrary message, which would appear to be signed by A

Dedicated hash chains (cont.)

- The main problem of IHC signature scheme - A cannot distinguish hash values of B that are meant for acknowledgments from hash values that are meant for signatures
- LHIP uses two dedicated hash chains per host to avoid this situation
- Every host generates a *signature chain* and a *trigger chain*
- Signatures are created with elements from the signature chain
- Acknowledgments use hash chain element from the trigger chain
- The trigger chain elements trigger the release of the next hash chain element of the signer's signature chain

Dedicated hash chains (cont.)

- Both host exchange the two host anchor during the BE
- After the exchange a host can send updates whenever it is necessary
- Simultaneous updates are no threat to this construction
  - an attacker cannot sign messages with hash chain elements from a trigger chain
  - the attacker cannot acknowledge message with elements from a signature chain
- Using two hash chains per host does not increase the overhead
  - IHC signatures, with and without dedicated hash chains, use one hash chain element in each side
- The additional effort - exchanging two hash chain anchors per host

Dedicated hash chains (cont.)

- Both hosts can initiate the signature process
- It is not possible to send any further signed messages before the process is completed
- Using one pair of hash chains per host creates one duplex channel
- Multiple channels - multiple pairs of hash chains
- Several channels enable parallel signature interaction handling and reduce delay
- The sequential way in which HIP handles UPDATE message does not require such parallel channels

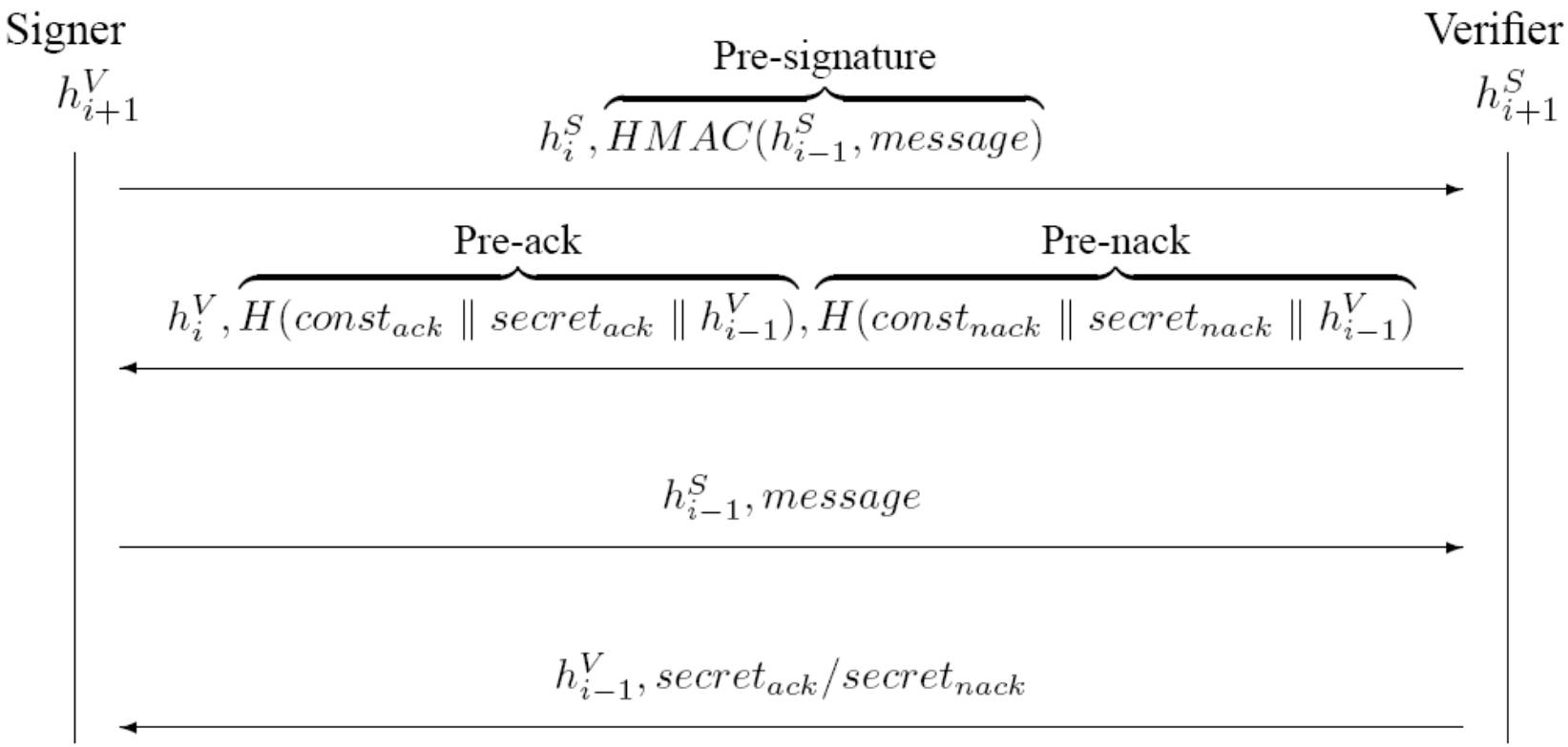
Pre-signatures

- The original IHC signature approach - three packets for each message
- The S1 packet - a hash chain element h_i, the message m, the HMAC
- The HMAC is generated from m and the next undisclosed element h_{i-1}
- The packet must be buffered by the verifier and cannot be verified before h_{i-1} has been disclosed by the signer
- This scheme bears some problems
  - requirement of strict serialization
  - multiple messages cannot be sent at once
  - packing several signed messages into one packet is not possible
  - the number of bytes per packet is limited by the max packet size
  - sending several messages signed with one hash chain element collides with the interactive scheme

Pre-signatures (cont.)

- Proposal - not to send the actual message in the S1 but to send it in the S2 instead
- The first packet - only the signer's next hash value h_i^S and the signature HMAC(h_{i-1}^S, m)
- The verifier acknowledges this packet with a hash chain element of its trigger chain
- The signer discloses h_{i-1}^S and the message m in a second packet
- The signature without the message - *pre-signature* as it signs a message that is sent later

Interactive hash chain signatures with pre-signatures and pre-acknowledgments



- the message is sent in the third packet instead of in the first packet

Pre-signatures advantages. Small size

- The Responder must buffer the first message of the IHC signatures to verify it
- Verification - by using the hash chain element from the second message
- Pre-signatures - reduction of the amount of data that needs to be buffered
- The benefit for end-hosts is marginal
  - the end-hosts typically have enough memory to buffer the signatures plus messages
- Memory is a scarce resource in middleboxes, such as firewalls and NAT boxes
- Middleboxes need fast access to the memory to achieve high throughput

Pre-signatures advantages. Small size (cont.)

- Middleboxes must buffer the first of the IHC signature messages to verify the signatures
- Pre-signatures drastically reduce the amount of memory required for message authentication in middleboxes
- Only the small pre-signature must be buffered
- The middleboxes buffer the small pre-signatures instead of the larger cleartext message
- Pre-signatures reduce the amount of data written and read from the memory of the middlebox

Pre-signatures advantages. Cumulative transmission

- Many pre-signatures fit into one HIP control packet
- Many pre-signatures can be placed at the verified side with one message
- All of these pre-signatures are generated with the same undisclosed hash chain element
- The verifier acknowledges the receipt of a cumulative pre-signature just as it would acknowledge a single pre-signature
- The signer can now send all pre-signed messages in parallel
- The performance of system is increased

Pre-signatures advantages. Efficient retransmission

- A signature can only be verified if both packets form the signer to the verifier are transmitted without any errors
- The first packet is erroneous - first packet must be retransmitted
- The second packet is erroneous - both packets must be retransmitted
- The first packet requires to be retransmitted in every error case
- The second packet needs to be retransmitted when an error occurs in the second packet
- The actual message m in the first packet - the large message needs to be retransmitted in both error cases
- Placing m in the second packet reduces the probability of retransmissions of the potentially large message m

Pre-signatures advantages. Confidentiality

- The original IHC signature scheme - the cleartext message in the first packet
- The verifier cannot trust this message until it is verified by using the hash chain element disclosed in the second packet
- An attacker can read and use the message from packet under the assumption that it is authentic
- The assumption gives the attacker an advantage
  - the attacker gains knowledge of the message and can react to the message before the verifier can use the message
  - the attacker can use this knowledge to foresee and disturb the actions of the verifier

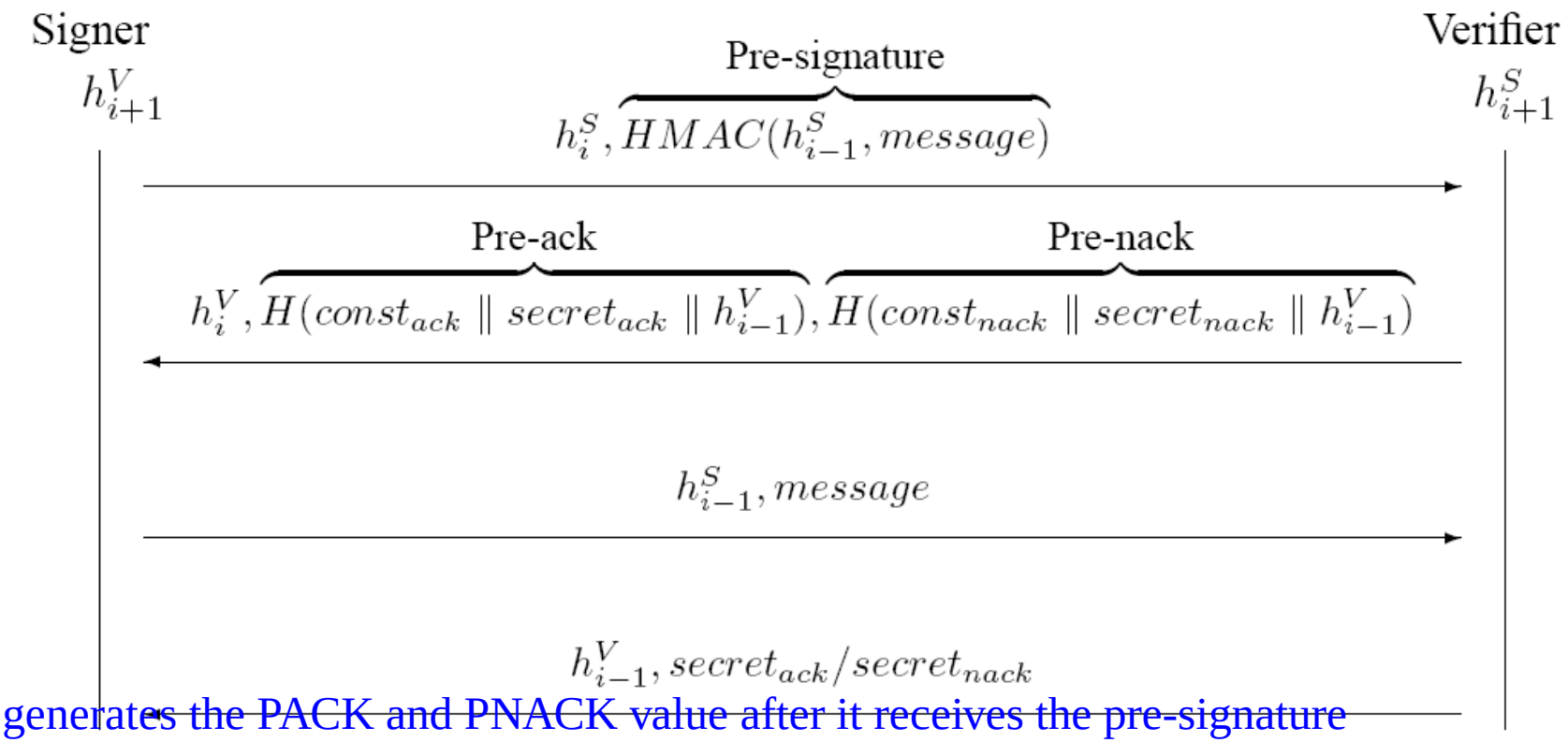
Pre-signatures advantages. Confidentiality (cont.)

- Pre-signatures do not reveal the contents of the message until it can be verified
- An attacker cannot use the interval before the message is verifiable to react to it
  - this is important whenever two or more parties compete for a limited resource
- Prior knowledge of the message gives an attacker an unfair advantage
- LHIP does not necessarily require confidentiality
- It might turn out beneficial for interaction with certain kinds of LHIP-aware middleboxes

Pre-signatures advantages. Pre-acknowledgments

- Each signed message from S to V - tree-way IHC signature process
- Signed acknowledgment for a signed message - another set of three packets
- Reduction of the overhead - introduction of *pre-acknowledgments* (PACK) and *pre-negative acknowledgments* (PNACK)
- Pre-acknowledgments
  - contain pre-signed acknowledgments likewise the pre-signatures contain pre-signed message
  - contain hashed information, enabling S to verify whether V has sent an acknowledgment or a negative acknowledgment

Interactive hash chain signatures with pre-signatures and pre-acknowledgments



- V generates the PACK and PNACK value after it receives the pre-signature
- at this point it is unclear whether the signature process will be successful or not

Interactive hash chain signatures with pre-signatures and pre-acknowledgments (cont.)

- V creates pre-signatures of acknowledgments for both cases: PACK and PNACKs
- constant number $const_{ack}$ and $const_{nack}$ as messages
- pre-signatures are created with the next undisclosed hash chain element h_{i-1}^{V}
- using ordinary pre-signatures - the acknowledgments as well as the negative acknowledgments are signed
- S could verify the PACK as well as the PNACK value
- to avoid this - V includes a secret number $secret_{ack}$ in the PACK value
- V includes a different secret number $secret_{nack}$ in the PNACK value
- an attacker cannot forge these values as they contain an undisclosed element of the hash chain of S

Interactive hash chain signatures with pre-signatures and pre-acknowledgments (cont.)

- $PACK = H(const_{ack} \| secret_{ack} \| h_{i-1}^{V})$
- $PNACK = H(const_{nack} \| secret_{nack} \| h_{i-1}^{V})$
- V sends back both values in the second packet of the IHC signature process
- S stores both values and waits for V to disclose its next chain element and the $secret_{ack}$ or the $secret_{nack}$ value
- after receiving and verifying the message - V sends back either an acknowledgment or an negative acknowledgment
- a packet containing h_{i-1}^{V} as the trigger value and $secret_{ack}$ or $secret_{nack}$, respectively
- S verifies the acknowledgment by using formula for PACK with the disclosed values
- $const_{ack}$ and $const_{nack}$ values are publicly known

Interactive hash chain signatures with pre-signatures and pre-acknowledgments (cont.)

- the acknowledgment is valid - the result matches the PACK in the second packet
- the corresponding check is done for negative acknowledgments in case $secret_{nack}$ is contained in the fourth packet
- the contents of the PACK and PNACK values have a fixed length - avoiding length extensions attacks on the hash function
- V must choose $newsecret_{ack}$ and $secret_{nack}$ for every acknowledgment
- unique secrets are essential for the protocol to function properly
- PACK and PNACK - acknowledgments for signed messages with only one additional packet
- PACK and PNACK values are efficient to compute and verify as only few hash operations are necessary

Pre-signature advantages. Retransmissions

- Packet retransmissions are necessary for two reasons
  - IP packet can get lost or arrive corrupted
  - syntactically correct packets with valid checksum field can carry invalid signatures
- The LHIP authentication layer must react differently to both situations
- Simple packet loss
  - the last packet retransmission without using a new hash chain element
  - retransmissions are always initiated by the signer
  - do not require a new element as no further hash chain elements have been disclosed after the packet was lost

Pre-signature advantages. Retransmissions (cont.)

- Invalid signatures
  - restarting the signature process with a new PSIG packet that uses new hash chain elements
  - the invalidity of the signature is detected when the A2 is received
  - the S1 or the S2 is malicious - both packets must be retransmitted
  - a new undisclosed hash chain element to re-start the signature process
  - each failed signature - the use of two new hash chain elements on the signer and verifier side
  - the number of retries for each communication interface must be limited
  - a sender should use another interface for resending the S1 packet if the limit is reached

LHIP integration

- The LHIP authentication layer is located below HIP
- All HIP control packets are passed through the authentication layer
- The layer determines whether a packet requires protection or not
  - examine the parameters in the packet
  - update packets containing new location information, new IP addresses, and new hash chain anchors need to be protected
  - protection allows avoiding impersonation and connection disruption
- Packets that require authentication are buffered and sent with the IHC signature scheme

LHIP integration (cont.)

- The authentication layer on the remote peer verifies the hash chain signature and delivers the packet to the HIP instance
- Unprotected packets are sent directly - not processed by the authentication layer
- The authentication layer reuses the HIP packet format and parameter format to attach control information to HIP packets

Pre-signature creation

- The LHIP layer mimics the properties of the HMAC signature in the HIP packets
  - provides a similar kind of authentication that protects the same part of the HIP packet
- The HIP layer signs messages with the HMAC algorithm as it would without hash chain signatures
- The LHIP layer signs the contents of the HMAC parameter with IHC signatures
- The absence of a shared key - the HIP uses a fixed zero key for the HMAC creation
  - the HMAC is not more than a message digest

Pre-signature creation (cont.)

- Using the message digest as basis for the LHIP authentication has an advantage
  - the HIP layer can still use all mechanisms that rely on the presence of the HMAC signatures
  - the message digest instead of the message - no reduction of the IHC signatures security
- The authentication layer of the signer generates the pre-signature
  - from the contents of the HMAC parameter and the next undisclosed element of a chain

$$psig = HMAC(h_{i-1}^{S}, HMAC(0...0, m))$$

Pre-signature creation (cont.)

- The LHIP layer on the receiving host verifies the IHC signature and passes the verified message to HIP
  - HIP can verify that the right portion of the message is covered by the HMAC signature and the message digest is valid
  - HIP can verify the HMAC with the zero key

Parameters

- The LHIP layer must exchange authentication data to provide its services
- Data is carried by HIP packets and parameters
- Several new parameters for use with the authentication layer are defined
- The parameters carry the trigger values, the hash chain anchors during the BE, the pre-signatures, and the acknowledgments
- The parameters use the TLV structure
- The selection of the parameter type numbers reflects the position of the parameter in the HIP packet - all parameters in the packet must be in ascending order

Parameters. Hash Chain Anchor Parameter

- The HCAP is a critical parameter
- Carrying of the hash chain anchor
- The anchors are exchanged during the BE
- A message may carry HCAPs
- Besides the HIP parameter header, the HCAP contains
  - an 8-bit type field
  - a variable bit field for the anchor value
  - the 8-bit type field determines whether the variable bit field is an anchor for the signature chain, trigger chain or some other hash chain
- The HCAP is located in the parameter type range and is signed with RSA or HMAC signature

Parameters. Hash Chain Value Parameter

- The HCVP is also critical parameter
- The HCVP carries the trigger values necessary for the IHC signature interactions
- The HCVP is appended to all packets of the authentication layer
- The parameter type number must be higher than any parameter type number used by the HIP peers - it is added after the HIP processing of the outgoing control messages

Parameters. Pre-Signature Parameter

- The PSP is used in the PSIG packet
- The PSP contains the pre-signature of the message
- The content of the PSP
  - the HIP parameter header
  - a variable bit field for the hash function output
- The PSP is appended behind the HCVP
- The parameter is not critical - it may be contained within R1 and I1 messages

Parameters. PACK, PNACK, HACK and HNACK

- The PACK and PNACK carry the PACK and PNACK values
- The Hash Chain ACK (HACK) and Hash Chain NACK (HNACK) carry the positive and negative acknowledgments in the ACK packet
- The content of the parameter
  - a variable length field that holds the $secret_{ack}$ or $secret_{nack}$ hash value
- The parameters are critical
- LHIP introduces some more parameters that are not directly used by the authentication layer

Packet types

- The authentication layer uses three new kinds of packets: the PSIG, TRIG and ACK
- The MSG packet is the actual message with IHC signature parameters appended - no special packet number is assigned
- The packet numbers are strictly preliminary
- PSIG packet
  - contains the HIP header, the hash chain trigger encoded as HCVP, and the pre-signature encoded as PSP
- TRIG packet
  - contains the HIP header, a hash chain trigger in an HCVP, and the PACK and PNACK parameters

Packet types (cont.)

- MSG packet
  - the HIP update packet requiring protection
  - the authentication layer adds a trigger encoded as HCVP to it
  - the trigger indicates - the packet is part of the signature process
- ACK packet
  - contains the HIP header and a hash chain trigger as HCVP
  - contains either the HACK or HNACK
  - the presence of a valid HACK - the signature process was successful
  - the presence of a valid HNACK - the signature was invalid

LHIP associations

- LHIP layer provides protocol security for HIP
- The establishment of an LHIP communication context
- LHIP ensuring of secure mobility updates
- Closing an LHIP association

LHIP base exchange

- HIP uses the BE to create the HIP communication context
- LHIP creates a communication context at the beginning of a communication
- LHIP extends the HIP BE to establish the LHIP communication context
  - LHIP hosts generate the hash chains
  - exchanging of the hash chain anchors with the remote peer during the BE
- No public-key operations are performed during the LHIP BE
- The LHIP BE is kept similar to the HIP BE
  - maintaining of compatibility with HIP
  - enabling of hybrid LHIP and HIP implementations

LHIP base exchange (cont.)

- The LHIP BE consists of the same four way handshake as the HIP BE
- All HIP specific fields except the RSA and HMAC signatures are present in the LHIP BE
- LHIP does not change the semantic nor the functionality of HIP parameters such as the puzzle, or the R1 generation counter
- LHIP extends the functionality of the transform parameter
- LHIP uses the new LHIP parameters during the BE

HIP transforms

- The fourth LHIP high-level goal is compatibility with HIP
- Two LHIP-enabled HIP hosts need a way to agree on a HIP or LHIP
  - LHIP-enabled HIP hosts communicate with pure HIP hosts by HIP
  - not LHIP-enabled host should establish a HIP association with LHIP hosts
- The choice of whether to use LHIP or HIP - negotiation during the HIP BE, the LHIP parameters must be exchanged during this stage
- The I1 packet is identical for both HIP and LHIP
  - indicates that the Initiator requests to establish either a HIP or an LHIP association
  - the type of association is not determined yet

HIP transforms (cont.)

- LHIP defines an LHIP transform suite ID
- An LHIP-enabled Responder can express its preference by adding the LHIP transform suite ID to the HIP transform parameter in the R1
  - the LHIP transform suite ID indicates that the Responder supports LHIP for message authentication
  - the order of HIP transforms - the Responder prefers LHIP or HIP
- The absence of the LHIP transform suite
  - the host does not support LHIP
  - the host is not willing to establish an LHIP association for selected HI

HIP transforms (cont.)

- The Initiator selects a HIP transform suite from the given suites and sends back to the Responder
- LHIP will be used - the Initiator selects the LHIP transform suite
- The Initiator does not support LHIP
  - will not select the LHIP transform suite and ignore it
  - will always use plain HIP associations
- The way of agreeing on a HIP or LHIP selection maintains full compatibility between LHIP-enabled and normal HIP hosts
- No additional packets or parameters are needed to agree on an LHIP association

Hash chain anchors

- LHIP exchanges all hash chain anchors during the BE
- The anchors are added to the I2 and the R2
  - the possibility for Responder to stay stateless until it receives the I2
  - the possibility for Responder to use pre-created R1
- Adding the Initiator's hash chain anchors to the I1
  - requirement for Responder to establish state for storing the anchors after receiving the I1
  - the HIP I1-R1 mechanism enables the Responder to verify the routability of the Initiator's IP address before establishing state
  - stateless locator verification is impossible
- Decision - adding the hash chain anchors to the I2

Hash chain anchors (cont.)

- Adding the hash chain anchors of the Responder to the R1
  - the Responder cannot pre-create the R1
  - requirement to generate hash chains for every R1
- Using one pre-created message with only one set of hash chains is impossible
  - a hash chain must only be used for one LHIP association
  - reusing hash chain - IHC signature system insecure
- Computing new hash chains for every message - enabling of attacks with storm of I1
- Decision - adding the hash chain anchor to the R2

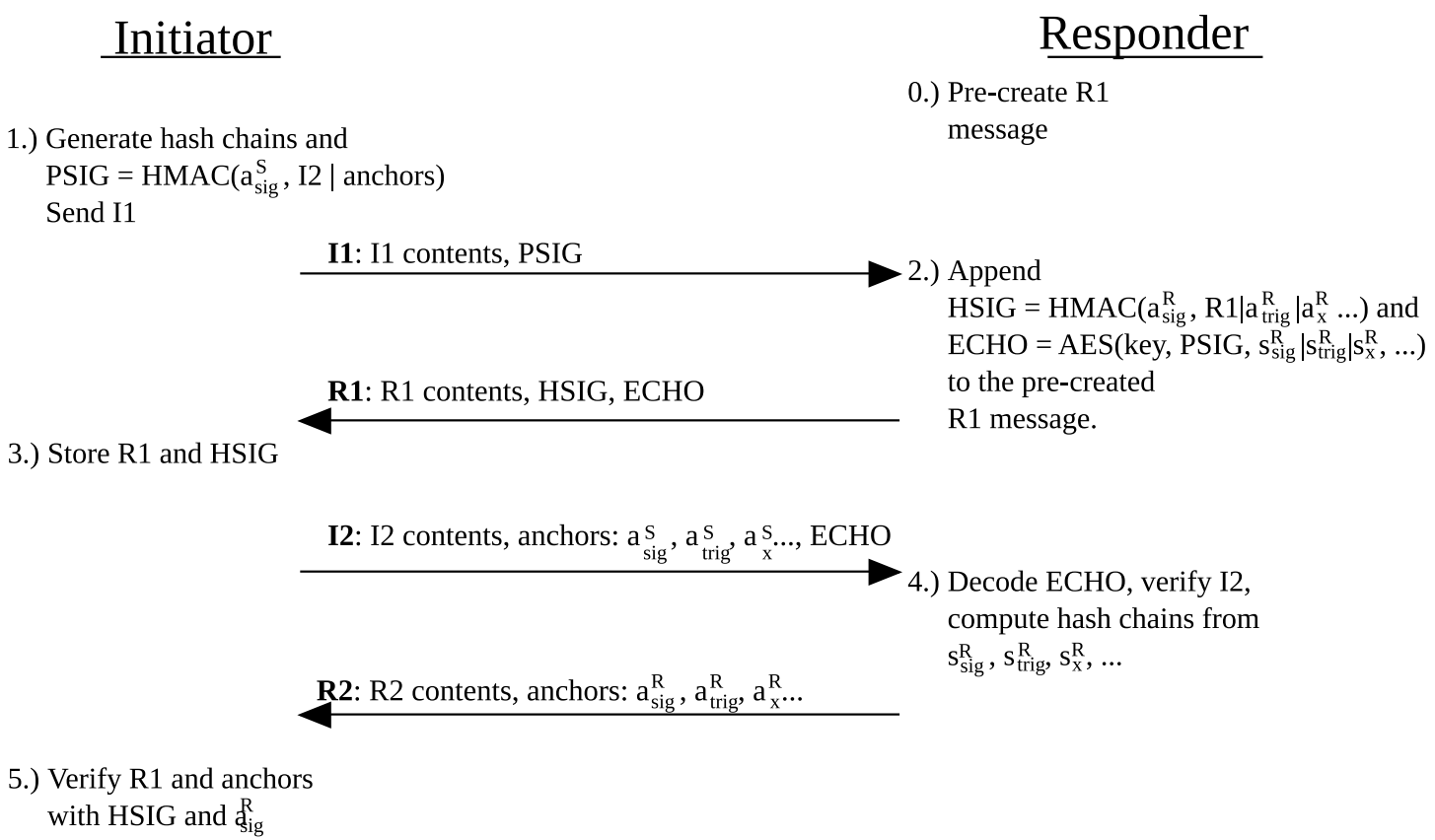
RSA and DSA signatures

- The R1 is the only message of the LHIP BE that must be signed with RSA or DSA - compatibility reasons with HIP
- All other messages are not necessarily signed with public-key signatures
- Hosts cannot authenticate their peers
- Weak hosts do not need to calculate public-key signatures inmost cases
- Public-key signatures are necessary in case of namespace conflicts

Hash chain signed base exchange

- The four-way BE cannot be protected by the authentication layer
  - the hash anchors are exchanged in the last Initiator and in the last Responder messages
- Using the authentication layer to sign this messages
  - requirement of additional packets
  - not significant increase of the security - an MITM attacks could still replace the hash anchors transmitted during the BE
- A way to gain increased security - using chained bootstrapping in a similar way to how WIMP does

Chained bootstrapping for LHIP



- the LHIP base exchange is depicted in a simplified form
- both peers sign their base exchange messages with IHC signatures

Diffie-Hellman and public-key signatures

- The Diffie-Hellman and the RSA or DSA HIs are exchanged during the LHIP BE
- The Diffie-Hellman is not calculated
- The RSA and DSA identifier of the peer is not verified
- The values are stored for later use during the upgrade process from LHIP to HIP
- Hybrid LHIP- and HIP-enabled Responders must sign the pre-created R1 message to maintain full HIP compatibility
- The overhead is acceptable - only the one signature is required for potentially many associations

Pre-created hash chains

- Host can speed up the connection establishment - pre-created hash chains
- There is no need to calculate the hash chains during the BE
- A host can pre-calculate a pool of hash chains to reduce the computational effort during the BE
- The host takes hash chains from the pool whenever it establishes a new association
- The hosts refill their hash chains pools whenever they have CPU cycles to spare

HIP payload

- LHIP requires to utilize a tunneling protocol to transfer its payload
- IPsec ESP is the only transfer protocol being specified
- LHIP uses ESP to sustain the compatibility with HIP and HIP-aware middleboxes
- The ESP traffic cannot be encrypted securely - missing shared secret
- LHIP uses ESP NULL mode for packet encapsulation
- LHIP uses a fixed zero-string for packet authentication
- This way of operating IPsec
  - neither encrypts nor authenticates the ESP packets
  - performs all operations required to tunnel the payload to the destination

Concluding the LHIP base exchange

- At the end of the LHIP BE both hosts
  - have agreed to use LHIP
  - have exchanged their hash chain anchors
- An encrypted ESP BEET tunnel is established
  - the hosts can exchange payloads via the tunnel
- The hosts are able to send authenticated UPDATE message
  - an attacker cannot unnoticeable modify the UPDATE message
- The hosts have stored the DH keys and HIs of their peer for later use

Concluding the LHIP base exchange (cont.)

- Neither of the hosts
  - has computed RSA signature
  - has done RSA signature verification
  - has done the DH shared secret computation
- Both hosts save
  - one RSA signature
  - one RSA signature verification
  - one DH shared key computation

Mobility and multihoming signaling

- New authentication layer protects mobility and multihoming updates
- LHIP does not modify the basic update process
- LHIP does not replace the update messages with other packets
- Compatibility with HIP is maintained
- Reuse of the HIP update functionality, including present and future extensions without modifications is allowed
- The LHIP layer protects the first pair of HIP UPDATE messages
- The U1 and U2 messages need protection
  - contain the new location information
  - contain the SPI values of the peers

Mobility and multihoming signaling (cont.)

- The U1 and U2 messages need protection
  - contain the new location information
  - contain the SPI values of the peers
- An attacker can disrupt an LHIP association or reroute traffic in the case of unprotected message
  - an attacker does not necessarily have to be located on the communication path
  - guessing the HIs, SPIs, and IP addresses of the peers is sufficient to perform an attack

Unprotected third update message

- The third update message - a response to the return routability test
- Contains the ack parameter and the $echo\ response$ parameter
- Protection is not needed
- Decision does not influence the security of LHIP
  - assumption of two kinds of attackers
  - eavesdroppers who are not on the communication path
  - MITM attackers on the communication path

Unprotected third update message (cont.)

- The case of an MITM attack
  - an attacker can either drop or modify the message
  - the attacker cannot modify the message without invalidating it - it only contains an acknowledgment and the echo response
  - the receiver can easily discover modifications of these parameters
  - modifications and destruction lead to the same result: the path cannot be verified
  - a signature cannot protect against this attack - it can only identify the manipulation, not prevent it

Unprotected third update message (cont.)

- The case of an eavesdropper
  - an eavesdropper can send a forged packet with invalid content
  - an eavesdropper cannot drop the actual UPDATE packet
  - an eavesdropper can possibly deliver a forged UPDATE before the legitimate UPDATE reaches the host
  - the receiver can ignore invalid messages and wait for a packet with a valid echo response
  - timeouts to mark a locator as unroutable - no valid response within a certain time span

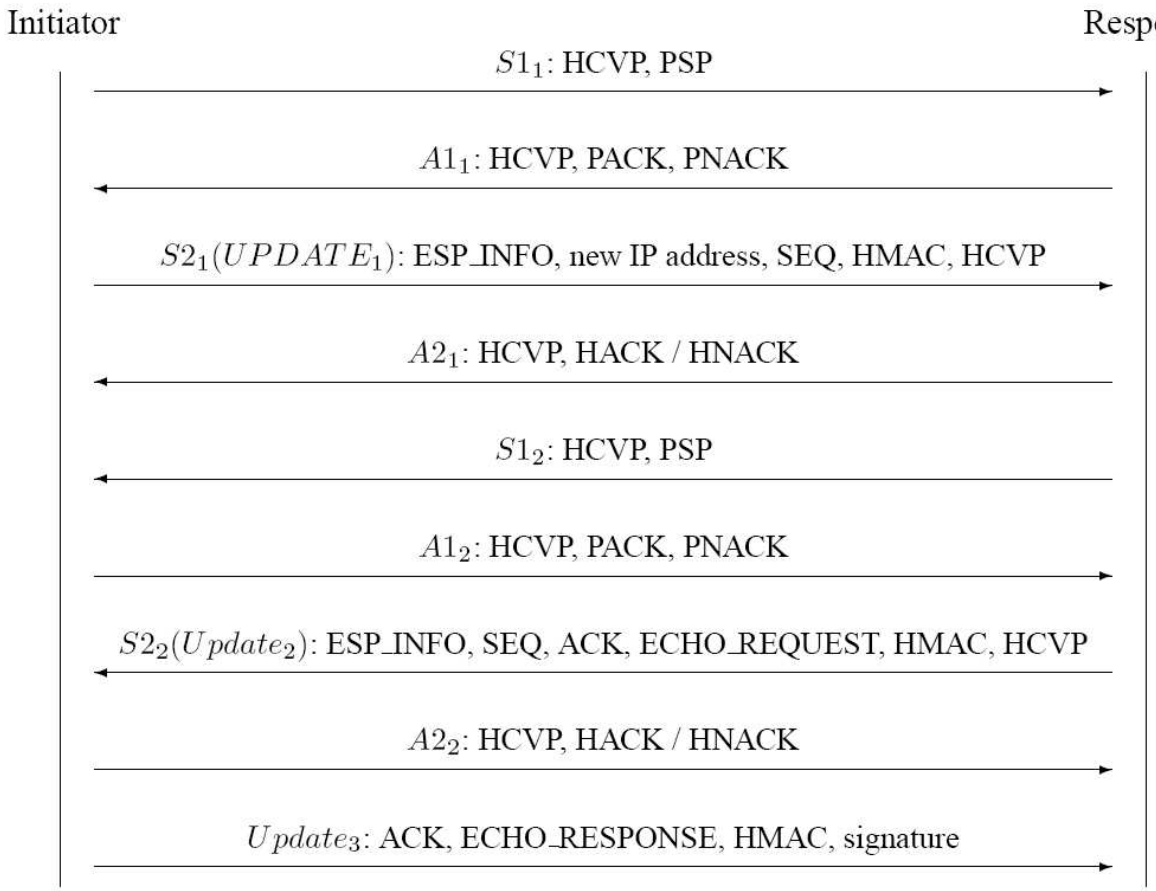
Unprotected third update message (cont.)

- Decision not to protect the third message - a host will notice manipulations even without signatures
- Saving of three IHC signature packets
- Further HIP extensions might need to send protected data in the third UPDATE packet
- Such extensions would also need to redefine the LHIP behavior

LHIP update process

- The three-way HIP update process is extended by the packets for IHC signature authentication
- Two full IHC signature authentication cycles are added for the packet carrying new locator information and ESP
- The update process - a nine-way update process
- The ACK packet and the following packet are sent in parallel
- Reduction the transmission time required for the update process to 3.5 RTTs

The LHIP update process



- the update process is protected by IHC signatures
- the original update packets are marked with U1, U2, and U3
- the U1 and U2 packets are the MSG packets of our IHC signature scheme
- neither RSA nor DSA signatures need to be calculated - the middleboxes can use the IHC signatures to authenticate packets
- LHIP update process reduces the computational overhead of HIP updates

Updating hash chain anchors

- Hash chains are finite sequences
  - a host might run out of hash chains elements after some time
  - a new hash chain anchor must be retransmitted to its peer
  - the hash chain anchor must be signed to avoid identity theft during the lifetime of an LHIP association
- LHIP uses UPDATE messages to exchange new hash chain anchors
  - the anchor can be piggy-backed on UPDATE packets
  - the anchor can be sent in dedicated UPDATE packets
- The IHC signature scheme is used to authenticate the packets
- A host receives a new anchor and stores it
- The host waits until its peer decides to use the new hash chain

Updating hash chain anchors (cont.)

- Beginning of new hash chain use is not explicitly announced by a host
- First undisclosed element of the new chain in a new cycle is used
- A peer notices - the HCVP parameter in the packet is invalid
- Verification of the HCVP with the last known hash value of the peer's old hash chain
- Attempt to verify the HCVP with the newly announced anchor value
- The use of new hash chain values in the case of successful verification
- The implicit hash chain activation - no additional packets or parameters
- The HCVP verification fails - calculation of two hashes instead of one
- Sending hash chain anchors is possible at any time

LHIP interaction with middleboxes

- HIP UPDATE messages are processed by middleboxes on the communication path
- Middleboxes learn about new IP and SPI by inspecting the UPDATE message
- HIP uses RSA signatures to sign the UPDATE messages
- The middleboxes are able to filter out forged messages that aim at disrupting a HIP association by modifying the state of middleboxes
- The middleboxes cannot use the HMAC signatures for message verification - no possession of the shared secret
- IHC do not rely on a shared secret - the middleboxes can use IHC signatures to verify HIP UPDATE messages without public-key operations

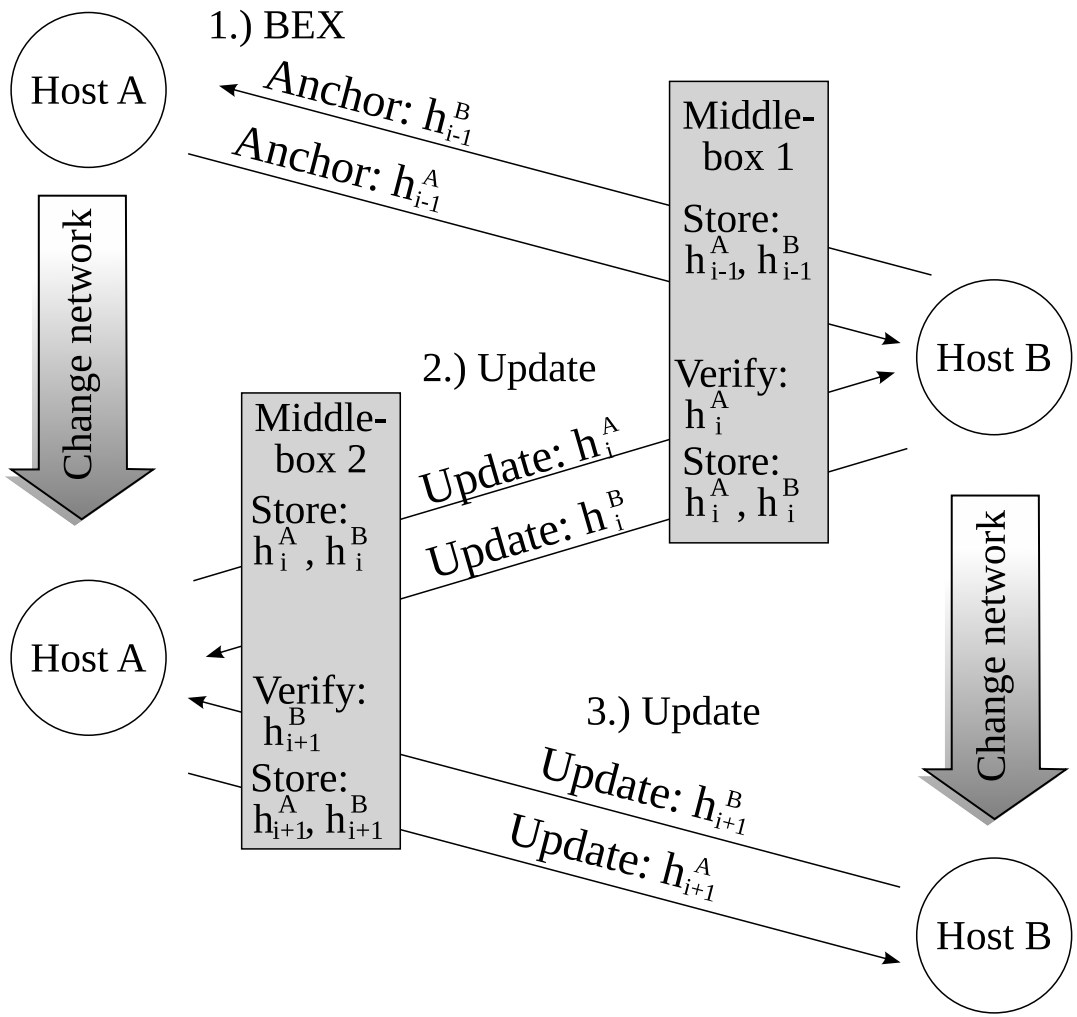
Update verification

- The trigger value - a middlebox learns about the update packet
- Assumption: host A and host B - an LHIP association
- A middlebox M learns the hosts' hash chain anchors during the BE or the update process
  - the anchor value h_{i+1}^{A} of the signature chain of host A
  - the anchor value h_{i+1}^{B} of the trigger chain of host B
- Host A changes position - an IHC signature protected UPDATE packet
- M has to learn about the new IP of A during the update process
- M observes the first PSIG packet from A to B
  - the next undisclosed element h_{i}^{A} in the packet

Update verification (cont.)

- Verification of PSIG packet - comparison of $H(h_i^A)$ with $H(h_{i+1}^A)$
- M buffers the pre-signature and waits for the TRIG packet from B
- The TRIG from B to A - B has received the pre-signature
- M can verify the origin of the TRIG packet with the HCVP parameter contained in it
- Host A sends MSG packet
  - the new locator of the host A and h_{i-1}^A
- M can verify the MSG packet and the authenticity of the new IP and update its mapping between HITs and IP addresses accordingly

Middlebox state update



- middleboxes learn the hash chain elements during the base exchange and during updates

Middlebox state update (cont.)

- host A and host B - an HIP association is established and anchors values are exchanged
- middlebox 1 stores values to verify packet later on
- host A changes location - an update is sent from a new IP
- middlebox 1 can verify the hash chain value in the PSIG packet
- host A moves behind middlebox 1
- middlebox 1 allows outgoing traffic from A and stores the hash chain values during the update process
- middlebox 1 can verify PSIG messages from whenever B moves to another location

Middlebox state update (cont.)

- Middleboxes must buffer the first message of the IHC signature
- Buffering until the message with the corresponding hash chain element arrives
- The pre-signature keeps the first message small
- The middlebox - PSIG and TRIG packets passing without verification of the source IP
- Checking that only limited number of PSIG and TRIG are sent to the same IP - avoiding DoS attacks
- Checking that PSIG and TRIG packets contain no payload besides the HCVP, the PSP, the PACK, and the PNACK
- Dropping packets that contain further parameters

Closing an LHIP association

- HIP CLOSE message - a HIP association should be closed
- Both hosts delete the state and tear down the BEET tunnel
- HIP CLOSE messages are protected by HMAC and public-key signatures
- LHIP must naturally protect the message
  - prevention attackers from closing an LHIP association with forged CLOSE message
- Naive approach - protection the closing procedure with IHC signatures
  - two-way close process - eight-way IHC signed process
- CLOSE message - one piece of binary information

Closing an LHIP association (cont.)

- An approach similar to Lamport's one-time signatures
  - exchange of a short hash chain
  - hash chain consists of the seed value r and the anchor value $h1$
  - the hash chain - a close chain
  - the hash chain only consists of two elements binary hash chains
- The anchor of a binary chain is published during LHIP BE
- The host decides to close LHIP connection - disclosing the seed value r
- Peer verification of CLOSE message - comparison $H(r)$ with the anchor of the close chain $h1$
- Values match - a host can be sure that peer requests to close HIP association

Closing an LHIP association (cont.)

- CLOSE message with an ack parameter and the seed of it own close chain are sent back
- Both hosts can verify that their peer is closing the LHIP association
- Additional packets or the use of public-key signatures are not required
- Middleboxes are able to authenticate the close messages
- Middleboxes that have not observed BE
  - cannot authenticate CLOSE messages
  - they have not learned the anchor of the close chain

Closing an LHIP association (cont.)

- Middleboxes must use conventional approaches to determine when to delete the state allocated for the communication peers
- An alternative way to inform middleboxes about the anchor - anchor in the signed part of every UPDATE message

Security considerations

- The third high-level goal of LHIP is namespace security
- It is not necessary for LHIP to authenticate hosts
- The namespace must be protected from malicious use
- HIP verifies the authenticity of the HIs and the public-key signatures during the BE
- LHIP reduces the computational overhead by omitting verifications
- Illustration of possible attacks on the host identity namespace

Attacks without namespace protection

- The attack vector opened by the LHIP design - impersonation and DoS
- The attacks build on the fact - HIs of LHIP host are not verified during the BE
- The attacker
  - needs *a priory knowledge* of HIP associations before they are established
  - must learn victim's peer host before the BE takes place
- The knowledge can be gained by communication pattern analysis of victim hosts
- E-mail communication, automated update services, instant messaging, shared folders and printers - examples of frequent communication between the same hosts, can be subject to the attack

HI blocking attack

- The identity blocking attack - HIP and LHIP use a pair of HIs to identify a HIP context
- Only one HIP or LHIP association can exist for every pair of HIs
- No problems for plain HIP
  - only a hosts that is in possession of the private keys is able to establish a HIP association
  - the HI is statistically unique and collisions are expected to be extremely unlikely
- LHIP does not use RSA signatures to authenticate the identities
  - every host can impersonate any other LHIP host

HI blocking attack

- Host authentication is not a goal of LHIP
- An attacker exploits the fact - only one LHIP association can be assigned to a pair of HI
- The attacker blocks connection attempts of a victim
  - establishment of a HIP association with the victim's HI before the legitimate host does
- The attacker can misuse the unprotected host identity namespace for attacks
  - the attacks are worse than today's attacks on the IP or TCP protocol and must be dealt with

HI blocking attack (cont.)

- The use of public-key signatures in case of conflict prevents the attack
  - for every I1 the Responder checks whether a HIP association exists that uses the HIs in I1
  - in case of existing the Responder requires the Initiator to sign I2 with its HI
  - the signature is valid - the Responder discards the existing LHIP or HIP association and establishes a new one
- Legitimate hosts are protected from namespace attacks

HI blocking attack (cont.)

- Legitimate hosts can prove their identity to servers that have already established an association with an attacker
- The policy does not introduce a performance overhead
- The use of public-key signatures is only necessary in the case of conflict
- The presented approach opens the attack vector
  - attackers can try to provoke HI collisions
  - force a server to verify the HIs of spoofed I2 packets
- A server suspects that it is under attack
  - the server is able to force the client to solve a cryptographic puzzle by utilizing the HIP puzzle mechanism

HI stealing attack

- The attacker exploits the following fact
  - a host that has already established a HIP association with another host would reuse this association for all further communication
- An attacker A can impersonate a server S by using the server's HIs
- Impersonation of the server in a global scope influences the name resolution infrastructure

HI stealing attack (cont.)

- Spoofing the HIT of the server and establishment of an LHIP association with a victim V
  - process on the victim tries to open an LHIP association with S
  - the LHIP layer reuses the LHIP association with HIs and sends the packet to the attacker
  - circumstance of the name resolution infrastructure and impersonation

HI stealing attack (cont.)

- Problem when the Initiator assumes that the name resolution infrastructure is trustworthy
- The attack is possible
  - no verification for incoming or for outgoing traffic
  - one HIP association is used for all traffic between two hosts
- Opening an LHIP association without attacker's identity proving
- The host identity resolution infrastructure is circumvented

HI stealing attack (cont.)

- Authenticating the HIs of peers prevents this attacks
  - a client should authenticate the HIs of its peer whenever it acts as a Responder
  - a server should authenticate the HIs of its peer whenever it acts as an Initiator
  - increase of the LHIP computational overhead
- In most cases clients do not act as Responders, servers do not act as Initiator
  - Clients - unauthenticated LHIP for outgoing traffic, servers - for incoming traffic

Defense mechanism

- HIs are used without verification - host identity namespace is vulnerable to attacks
- Conditional use of RSA for host identity verification - prevention
- LHIP - discovering a way for hosts that discover namespace conflicts to inform their peers that they must authenticate
- Mechanism of conditional authentication and security implications

LHIP flags

- LHIP flags parameter (LFP) - information about an LHIP association
  - indication of host requirement for peer's authentication and willing of host to authenticate itself
- During the BE the Responder adds the LFP to the R1
  - the *authentication required* flag - the Initiator must authenticate
  - the *authentication granted* flag - willing to authenticate to the host
- The Initiator uses RSA to sign the I2 if requested

LHIP flags (cont.)

- The Initiator adds an LFP to the I2
  - the Responder should authenticate or not
- The Responder uses RSA to sign the R2 if requested
- The host is not willing to authenticate - the BE fails
- Opening an LHIP association is not possible

LHIP flags (cont.)

- The Initiator can determine the situation on receipt of the R1 and proceed with a regular HIP BE
- The attacker can exploit the conditional signatures to mount DoS attacks
- A host may deny to use RSA according to its policies
  - setting the according bit in the LHIP flags parameter to 0
- The LHIP parameters - after all other parameters in the R1
  - parameter is appended to the pre-created R1
  - 63404 - the parameter type number
  - the parameter is not critical - ignored by non LHIP implementations

Prevention of replay attacks

- RSA signatures are susceptible to replay attacks
  - the signed part of the message does not contain data unique for every message
- The Diffie-Hellman key exchange - prevention of replay attacks for HIP
- The Diffie-Hellman shared key is not generated by LHIP hosts
- The Responder cannot distinguish replayed LHIP I2 and R2 from legitimate message - no unique data
- Discarding of the old HIP association and establishment of the new HIP association with the attacker

Prevention of replay attacks (cont.)

- Unique echo request parameter in the R1 and I2 prevents the attack
- The signed echo response
  - checking that no other LHIP association has been established with the same echo response

Authenticated hash chains

- The RSA signatures cover the hash chain anchors
- The hash chain is tied to the identity of a host
- The hash chain can be used to authenticate the host later on
- The peer and the middleboxes on the path can verify legitimacy of HI owner that sent updates and close messages
- The authenticated LHIP mode has the same namespace properties as HIP
- The identity of the host can be authenticated unambiguously
- Impersonation of the host during the BE or LHIP association lifetime is not possible

Association upgrades: from LHIP to HIP

- One design goal of LHIP - upgradability to HIP
- An LHIP association should be transformable to a HIP association
- Transformable without the need to close or reestablish the association
- Upgrade from LHIP to HIP
  - calculation the shared secret
  - generating the necessary keys from secret
  - verifying the HIs
  - replacement the NULL mode encrypted BEET tunnel with an encrypted BEET tunnel
- Nearly all needed information is already exchanged during the BE
- Necessary modification to the BE

Second HIP transform

- The HIP transforms indicate algorithms used during a HIP association
- The selected HIP transforms defines
  - the hash function used for the HMAC algorithms
  - the symmetric key algorithms used for ESP encapsulation
- The LHIP transform - a host will use lightweight authentication layer
- Upgrade of an LHIP association - agreement on a transform suite to be used after the upgrade

Second HIP transform (cont.)

- LHIP hosts use the transform parameter to agree on two choices
- An LHIP-enabled host wants to use upgradable LHIP
  - the LHIP transform as the first choice
  - a HIP transform as the second choice
- Storing the selected HIP transform and using it during the upgrade

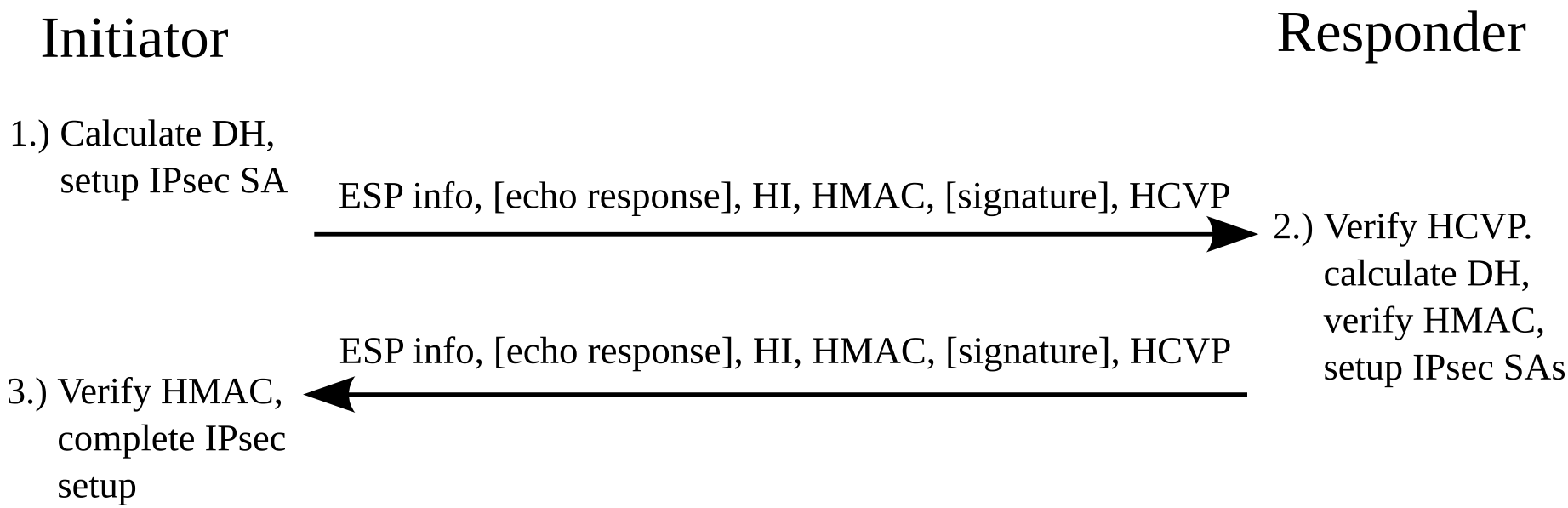
Upgradable associations

- An LHIP associations are upgradable
  - DH parameters are exchanged
  - the second HIP transform is exchanged
  - an echo request parameter is exchanged
- Hosts must be willing to upgrade association depending on local policies
  - the *upgradable flag* in the LHIP flags parameter
- All necessary parameters have been exchanged and both hosts express their willing - the connection is upgradable

Upgrade process

- Upgrades are necessary
  - an application opens a socket and requires authentication and encryption for this socket
  - local policies or a change of location might cause an application to require increased security
- The upgrade process - a two-way message exchange
- HIP UPDATE messages carry the upgrade information
- UPDATE messages as UPGRADE messages
  - no need to define a new message type
- *Initiator* initiates the upgrade process, its peer - *Responder*

The LHIP upgrade process



- a host has to add the echo response parameter and the RSA signatures if it has not already authenticated itself during the LHIP BE

Upgrade process (cont.)

- LHIP uses RSA and DSA signatures during the UPGRADE process
- Process is not protected by client puzzles
- Attacker can open numerous LHIP associations and upgrade them at the same time to put a heavy load on the Responder - open question
- Exchanging a fresh puzzle before the upgrade - two additional messages
- The puzzle from the BE - enough time for the attacker to compute the puzzle before attack
- Hosts can use the LHIP flag - no support of the LHIP upgrade process
  - the upgrade process - establishing a new HIP association through a standard HIP BE

First upgrade message

- The Initiator calculates the DH shared secret before initiating an upgrade
- The secret based solely on the DH keys exchanged during the BE
- Generating the necessary symmetric keys and the keys for the HMAC
- New IPsec security policies
  - selected in the second HIP transform algorithms
- New outgoing IPsec security association

First upgrade message (cont.)

- The Initiator prepares the UPGRADE message
  - contains the SPI value of a new IPsec SA
  - HMAC is added to protect the SPI
- Host uses private RSA key to prove its identity
  - the echo response from the BE before the HMAC and the RSA signatures are applied

Role of the echo response

- Protection of the UPGRADE progress from replay attacks
- Ensuring that the RSA signature was created for the current LHIP association
- Ensuring that the UPGRADE message is not a replay of a former LHIP UPGRADE exchange
- The Responder can verify that the Initiator is in possession of the secret corresponding to the host identity

Role of the HMAC, RSA signature ordering

- Assumption - HMAC is added after the RSA
- An MITM attacker replaces the DH parameter during the BE
  - the DH is not protected by signatures - the modification is not noticeable
- LHIP upgrade - the Initiator signs the upgrade message and appends the HMAC
- The MITM attacker replaces the HMAC signature with the HMAC signature generated with its own DH key
- The Responder uses the attacker's DH key to calculate the shared secret
- The Responder accepts the packets of the attacker instead of the Initiator

Role of the HMAC, RSA signature ordering (cont.)

- The order of the HMAC and the RSA signatures prevents identity theft
- The HMAC signature is included in the RSA signature and cannot be replaced by an attacker
- Tie of the HMAC key and the DH public key of Initiator to its HI

Protection of the first upgrade message

- The UPGRADE triggers the DH shared key calculation at the Responder
  - CPU-intensive operation
  - provocation of the DH calculations with a forged message
- The UPGRADE only triggers the update
  - does not contain any explicit information not being protected by the HMAC
- Peers exchange the hash chain anchor of a binary hash chain, the upgrade chain, during the BE
- The disclosure of the next hash chain element - a host has initiated the LHIP upgrade process

Protection of the first upgrade message (cont.)

- The disclosed upgrade chain element is attached as HCVP
- The Responder first checks the HCVP parameter
  - hashing and comparison with the anchor value of the upgrade chain
  - the comparison is inexpensive
  - the comparison proves that the legitimate peer initiated the upgrade process
- The hash chain verification is successful - calculation the DH shared secret
- Generating the symmetric keys and the key for the HMAC from the shared secrets

Protection of the first upgrade message (cont.)

- The Responder can verify the HMAC in the UPGRADE by freshly calculated key
- The Responder modifies the IPsec policies to use the encryption algorithms selected in the second HIP transform
- Setting up the outgoing and incoming IPsec SAs
- Including the HIs of the peers in the UPGRADE is necessary to support HIP-aware middleboxes
  - the HIs enable the middleboxes to learn the public keys of the hosts, in case the middlebox has not observed the BE

Concluding the upgrade

- The Responder sends back an HMAC-protected UPDATE message
  - the message contains the Responder's SPI for incoming IPsec SA
  - the Responder must authenticate if it has not authenticated during the LHIP BE
  - the echo response from LHIP BE must be included to avoid replay attacks
- The Initiator can use the HMAC to verify the UPDATE
- The Initiator sets up its outgoing SA with the SPI given by the peer
- Both peers have upgraded to HIP and established a BEET tunnel
- No use of LHIP authentication layer any more

Concluding the upgrade (cont.)

- The LHIP upgrade only requires two messages
- The DH shared secret and the RSA signatures can be pre-created to speed up the process
  - the upgrade is inexpensive and can be executed with little computational overhead
- The first and the second UPGRADE packets are symmetric
  - prevention problems when both hosts decide to send an UPGRADE at the same time
  - each host will take the UPGRADE packet of its peer as the reply to its own packet
- No additional measures are necessary to cope with simultaneously sent UPGRADE packets

HIP downgrade

- A downgrade from HIP to LHIP is not desirable
  - both hosts are already in possession of the shared secret
  - efficient message authentication and symmetric encryption are enabled
- The computational cost of an established HIP association could be lowered by using NULL encryption
- The runtime modification of a HIP association is out of scope for LHIP
- The hash-chain-based update process might be of interest for middleboxes and weak hosts
- A mixed HIP and LHIP mode is future work

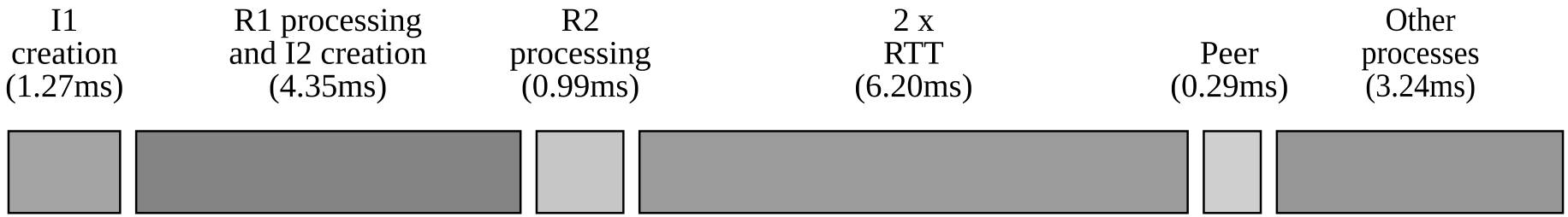
LHIP performance

- HIP for Linux implementation and LHIP-enabled implementation were used to compare the HIP and LHIP performance
- The time consumed by relevant processes in the *hipd* was measured
- The measurement show how much time is spent on processing for the BE and the update process
- The Nokia 770 acts as the Initiator in tests
- The 3.2 GHz Xeon acts as the Responder in tests
- 802.11/b Wi-Fi network

LHIP base exchange

- Three different categories of processes were measured
  - cryptographic computations performed on the hosts to generate keys or to sign and verify messages
  - protocol processing that is performed to process packet parameters, send packets, and to modify the state of a HIP or LHIP associations
  - packet processing - all operations required to process a packet including cryptographic computations and other protocol processing
  - total processing time - packet processing, packet transmission time and other delays caused by external influences

The LHIP base exchange between the N770 and a 3.0 GHz Xeon server



- the total duration of the BE is 14.56 ms
- the low delay is achieved by removing all asymmetric cryptography from the BE
- the hash chains for the IHC signatures have been pre-calculated before the BE
- on-line computation of the hash chains delays the LHIP BE by 1.17 ms (8.0%) on the Initiator and by 0.07 ms (0.5%) on the Responder
- the impact of LHIP processing on the initial delay of a connection is low for typical Internet RTTs, with RTTs of 50 ms the duration for computations on the hosts is 6% (7 ms) of the total BE time for LHIP (109 ms)
- LHIP reduces the packet processing time during the BE to 6.1 ms on the N770

LHIP base exchange (cont.)

- A main design goal of LHIP was to reduce the overhead of the public-key operations during the BE
- Presented LHIP design - all public-key operations from the BE are removed, other CPU-intensive processing steps are not introduced
- The BE performance improvement in LHIP is also important
  - a HIP association is established with every peer
  - introducing a long delay for every association is problematic
  - the demand for the HIP services is not clear before the association
  - developing reliable heuristics to determine the need for the services is difficult
  - requiring user interaction for each association reduces usability
  - solutions - use HIP for all associations or for one

LHIP base exchange (cont.)

- LHIP BE allows letting the applications determine whether encryption and authentication is required
- LHIP provides mobility and multihoming support with low computational overhead
- LHIP provides less security than HIP
- None of the hosts has to authenticate during the LHIP handshake
- Host authentication delays the I2 or R2 generation by the time that is required to compute the RSA signatures
- The I2 and R2 processing are delayed by the time to verify the RSA signatures

LHIP update

- The LHIP layer adds preceding and trailing IHC messages to every HIP UPDATE to authenticate them
- More packets must be processed and transmitted
- Determination of the overhead of the improved IHC scheme
- Each HIP UPDATE is accompanied by three IHC packets: S1, A1, A2
- The IHC signatures replace the RSA and DSA in the UPDATEs
- Comparison of IHC and RSA signatures
  - the processing time of IHC signature packet on the Nokia 770 and 3.2 GHz server
  - Internet Table OS 2005 on the N770

Performance of IHC signatures compared to RSA and DSA

	N770	Xeon 3.2 GHz
IHC signatures		
Send PSIG	0.33 ms	0.03 ms
Process PSIG, send TRIG	1.47 ms	0.05 ms
Process TRIG, send MSG	1.52 ms	0.05 ms
Verify MSG, send ACK	1.60 ms	0.05 ms
Process ACK	0.49 ms	0.05 ms
Sender(total)	2.34 ms	0.13 ms
Receiver (total)	3.07 ms	0.10 ms
RSA and DSA signatures		
RSA 1024 sign	181.32 ms	9.09 ms
RSA 1536 sign	578.70 ms	27.03 ms
DSA 1024 sign	96.71 ms	1.34 ms
DSA 1536 sign	229.27 ms	2.18 ms
RSA 1024 verify	10.53 ms	0.15 ms
RSA 1536 verify	23.47 ms	0.23 ms
DSA 1024 verify	118.73 ms	1.61 ms
DSA 1536 verify	287.76 ms	2.81 ms

Performance of IHC signatures compared to RSA and DSA (cont.)

- the IHC signatures consume 1.2% of a processing time of a 1024-bit RSA signature and 2.5% of the processing time of a 1024-bit DSA signature on the tablet
- hosts may open HIP or LHIP associations to several peers
  - assumption - N770 uses a 1024-bit RSA HI and has 10 open associations
  - N770 must compute 20 RSA signatures to send the first and the second UPDATE to the peers
  - 3626.4 ms for processing the RSA signatures for 10 updates
  - IHC signed updates - 55 ms for packet processing on the N770 and 2.3 ms on the Xeon
- LHIP achieves the high level goal LH1 - *performance* for the update process
- IHC signatures delay the message delivery by one additional RTT
  - the PSIG and TRIG packets need to be exchanged before the actual message is sent

Performance of IHC signatures compared to RSA and DSA (cont.)

- the IHC process is faster than the RSA and DSA protected UPDATEs if the time for processing the IHC signature on both peers plus RTT is lower than the time for the RSA or DSA signature creation
- the PSIG creation and the TRIG handling on the N770 consumes 1.85 ms processing time
- the N770 calculates 1024-bit RSA signatures in 181.32 ms
- IHC signatures require less time of the RTT is smaller than the time for the public-key signature generation

Discussion. LH1 performance

- Measurements - LHIP can be used on hosts with low processing power
- No long delays for establishing and maintaining an LHIP association
- IHC signatures reduce the computational overhead of UPDATE messages
  - devices with little CPU resources are capable of performing multiple location updates with different peers at the same time
- The delays caused by long RTTs affect the LHIP update more strongly than does the HIP update process
- LHIP CLOSE and UPGRADE messages are protected by single hash chain elements
  - the generation and verification of the elements is cheap

Discussion. LH1 - performance (cont.)

- LHIP speeds up the closing procedure - LHIP makes RSA and DSA signatures in the close message unnecessary
- The LHIP to HIP upgrade process requires both hosts to compute the shared secret and the RSA signatures
  - the computational overhead is similar to the overhead of the base exchange
- LHIP reduces the computational cost of HIP during the BE and the update process
  - reduction is less than 2.5% of the cost of HIP with 1024-bit RSA or DSA HIs
- LHIP achieves its first high-level goal LH1 - *performance*

Discussion. LH2 - protocol security

- LHIP provides less security than HIP
  - no authentication of the HIs
  - no payload encryption
- LHIP decreases the computational overhead of HIP in scenarios that does not require these security properties
- LHIP signs UPDATE messages with IHC signatures
  - attackers cannot forge important HIP control messages

Discussion. LH2 - protocol security (cont.)

- The improved IHC signatures are no longer susceptible to MITM attacks
  - the case of simultaneous signature packets from both peers
- LHIP protects the CLOSE and the UPGRADE messages
  - prevents attacks aiming at state manipulation and connection disruption
- The first contact between two hosts must be authentic

Discussion. LH2 - protocol security (cont.)

- LHIP cannot protect against MITM attacks without authenticated I2 and R2 messages
  - LHIP allows to use RSA and DSA signatures for these packets
- Host can request strong authentication from their peer if required
- The basic functionality of LHIP does not require public-key authentication
- LHIP provides secure mobility updates
- LHIP protects all important protocol processes with hash chains and IHC signatures
- LHIP achieves the second high-level goal LH2 - *protocol security*

Discussion. LH3 - namespace security

- LHIP uses RSA and DSA signatures in case of name space conflicts and to prevent harmful impersonation attacks
- The optional authentication of the I2 and R2 and the authentication of UPGRADEs - a host cannot block HIP or LHIP nodes
- The RSA and DSA signatures prevent attacks aiming at impersonating a HIP node by first establishing an LHIP association and then upgrading to HIP
- No aware of further attacks on the HIP namespace not prevented by the messages
- LHIP achieves its high-level goal LH3 - *namespace security*

Discussion. LH4 - compatibility

- LHIP was designed to interoperate with pure HIP implementations
- The decision whether an LHIP association is possible or not is taken during the BE
- LHIP hosts always establish HIP association when they communicate with non-LHIP-aware hosts
- LHIP supports packet inspecting middleboxes
  - the middleboxes can learn the SPIs, HIs, and anchor values from the BE and UPDATEs
- Middleboxes can verify the LHIP UPDATE messages efficiently
  - the way of authentication differs from the way HIP authenticates UPDATE packets

Discussion. LH4 - compatibility (cont.)

- LHIP provides the same features for middleboxes but it requires modifying HIP-aware middleboxes
- LHIP does not modify the way HIP works in general
- The LHIP authentication layer can be adapted to support existing and future HIP extensions
  - LHIP supports the mobility and multihoming extensions
  - other extensions, such as rendezvous servers and opportunistic HIP, are not supported yet
- LHIP can interact with unmodified HIP implementations and supports middleboxes

Discussion. LH4 - compatibility (cont.)

- LHIP reuses the HIP name resolution infrastructure and the host identity namespace
- LHIP achieves the high-level goal LH4 - *compatibility*

Middlebox traversal

Requirements for traversing legacy middleboxes

- Traditional IP architecture
  - intelligence is located at end hosts
  - the network is ”dumb”, just attempts to deliver data packets to a destination host with a best effort
  - packets are not modified in transit
  - routers do not inspect the packet content beyond the header
- The Internet gradually became full of middleboxes
  - compensation for lack of IPv4 addresses
  - prevention of attacks
  - performance of data caching

Requirements for traversing legacy middleboxes (cont.)

- Middlebox - a device located on the data path between hosts, processes packets differently than standard IP routers
- A strong advantage of HIP architecture
  - ability to function without changes to existing IP routers
- Some middleboxes can affect HIP packet or prevent a successful association establishment
- Middleboxes are seen as a harmful part of the Internet by the IETF
  - designed to support only a very limited set of protocols (often TCP and UDP over IPv4)
  - prevent deployment of new protocols in the Internet

Requirements for traversing legacy middleboxes (cont.)

- Middleboxes are mostly used only at the edges of the network
- Middleboxes inspecting application payload traffic would not be able to operate with the ESP encapsulation for HIP
  - HIP encrypts the entire packet content except for the IP header
- Null encryption is used
  - a packet modification would fail the MAC and checksum verification

Requirements for traversing legacy middleboxes (cont.)

- Best case - the middleboxes forward HIP packets without special handling
  - HIP connectivity would no be affected
  - the performance may be worse than for plain TCP/IP packets
- Worst case - a middlebox drops HIP packets
  - prevention of a direct HIP association establishment
- Traversing a middlebox
  - using some well-known protocol understood by the middlebox to contain HIP packets as a payload

Requirements for traversing legacy middleboxes (cont.)

- The problem statement of HIP interactions with middleboxes focuses of NATs and firewalls as the most commonly used middleboxes
- A product of discussion in the HIP RG at the IETF
- Middleboxes can cause problems
  - to the HIP control packets of protocol number 139
  - to the data packets, encapsulated using IPsec ESP
- The problem with ESP middlebox traversal is common to other protocols using IPsec
- HIP utilizes a new IPsec mode BEET

NAT traversal. HIP control packets

- The base exchange packets are implemented differently for IPv4 and IPv6
  - for IPv6, a special extension header is used
  - for IPv4, the HIP header follows the IP header as payload
- IPv4 NATs are widely deployed at the time of writing
- Reaching a receiver located behind NATs is a common problem
  - there is no possibility for the NAT to forward HIP control packets to the right destination
  - some NATs can be configured to forward a particular protocol to one of hosts

NAT traversal. HIP control packets (cont.)

- HIP BE packets for IPv4 do not include port numbers or SPI values
  - the BE cannot traverse NATs that perform port translation
- The basic NATs that translate only the IP addresses should not prevent the HIP BE
- The behavior of IPv6 NATs is not well-known yet
  - it is likely that most NAT implementation would just support a few most common protocols thus preventing the HIP BE

NAT traversal. HIP data packets

- The default mode of data encapsulation for HIP is IPsec ESP
  - the problem of NAT traversal of HIP data packets is similar to other IPsec applications
  - HIP utilizes a new BEET IPsec mode
- NAT cannot observe the transport-layer headers in packets
- NAT can only modify fields in the IP header
  - changes can invalidate upper-layer checksums
- Traversal of IPsec packets without encapsulation is possible but complicated

NAT traversal. HIP data packets (cont.)

- The SPI value contained in ESP packets has only one-way significance
  - in one direction, the NAT can demultiplex flows using SPI
- Some NATs offer a VPN pass-through feature
  - the NAT attempts to learn of SPI exchange in both direction
  - NAT sets up a corresponding mapping
  - the NAT can change the SPI in the packet in the case of a collision
- ESP traversal over NATs trends to work robustly only for a small number of ESP hosts behind a NAT

Firewall traversal

- A firewall - a middlebox between the internal network of an organization and the rest of the Internet
- A firewall can either forward or drop a packet depending on its content
- Firewalls can be configured to permit HIP control and data traffic
- Conservative configurations
  - allow a few known protocols, such as TCP or UDP
  - certain well-known ports, such as 80 for HTTP
  - prevent HIP control and data packets
- IPv4 firewalls block packets containing an IP option
- IPv6 firewalls can block all IPv6 extension headers

Strategies for legacy middlebox traversal

- First solution - encapsulation format for HIP control and data packets
  - middleboxes are able to understand and demultiplex traffic
- UDP encapsulation
  - support of the majority of middleboxes
  - does not produce similar latency issues to TCP
- The drawback of UDP encapsulation
  - header overhead
  - possible exceeding of the path MTU value

Strategies for legacy middlebox traversal (cont.)

- A UDP header can result into packet fragmentation
  - one of multiple fragment is more likely to be lost than a single packet
  - a single lost fragment - the whole packet is discarded at the receiver
  - known mechanism to attack the destination host - network buffers
- Second solution - reachability of a Responder located behind the middlebox
  - the Responder should register to a rendezvous server
  - the rendezvous server should be located outside of the middlebox in the public Internet

Strategies for legacy middlebox traversal (cont.)

- Registration punches ”holes” in the middlebox
  - RVS serves as a contact point for an Initiator
  - RVS forwards packets through the middlebox to the Responder
  - holes expire after a short timeout
  - the Responder should send refresh messages to keep the holes open
  - after the BE the Initiator might be able to send packets directly to the Responder through the same hole
- Middlebox traversal of mobility and multihoming HIP extensions
  - the HIP should punch a hole through new middlebox
  - the host informs its peer of a new locator
  - the host can inform of a new RVS that has a hole open

Legacy NAT traversal

- Encapsulation extensions to HIP
  - traversal of legacy HIP-unaware NATs according to IETF specification
  - traversal only of NATs supporting endpoint-independent mapping
- Alternative approaches to NAT traversal are work in progress
- The Solution based on the Interactive Connectivity Establishment (ICE)
  - the latest trend within the NAT design team in the IETF
- The presented NAT traversal extensions
  - UDP tunneling of HIP control and data packets
  - the Initiator, the Responder, or both hosts are behind a NAT

Legacy NAT traversal (cont.)

- The traversal extension
  - are needed when both hosts are behind different NATs
  - only the ESP data encapsulation is currently supported
- The Responder is behind a NAT, the Initiator has a globally routable IP
  - the Responder must use a RVS supporting the NAT traversal extensions
  - the RVS must have a public IP address

Legacy NAT traversal (cont.)

- A HIP host moves from the public Internet to a private address space behind a NAT
  - the host can start using extensions for executing the mobility update
- The HIP host moves to the public Internet
  - the host can stop using the NAT traversal extensions

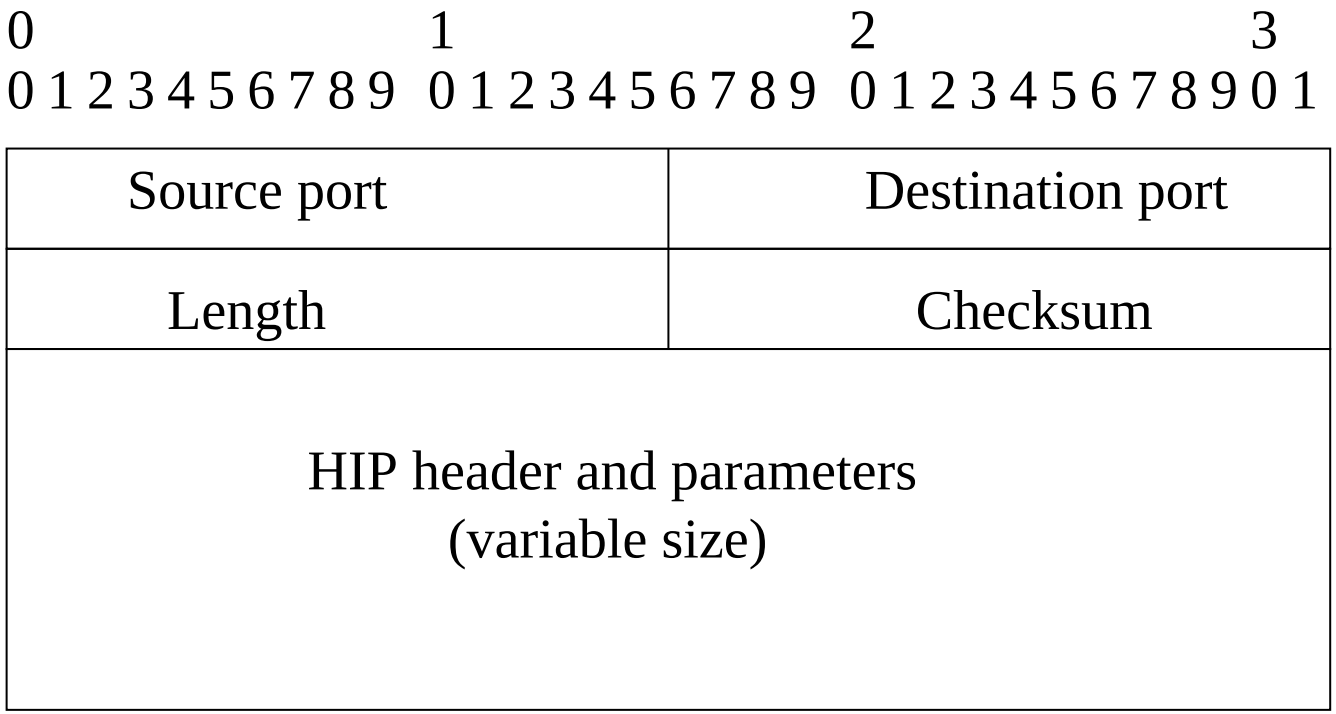
NAT detection

- A NAT present determination by the Initiator before the BE
  - a NAT is present - the Initiator employs the traversal extensions
  - a NAT is not present - the standard BE is used
- Both the Initiator and the Responder are behind the same NAT - special attention
- The NAT presence is not properly detected
  - both HIP hosts would use the traversal extensions
  - unnecessary sending of packets via a RVS located in the public Internet
  - *hairpin translation*

NAT detection (cont.)

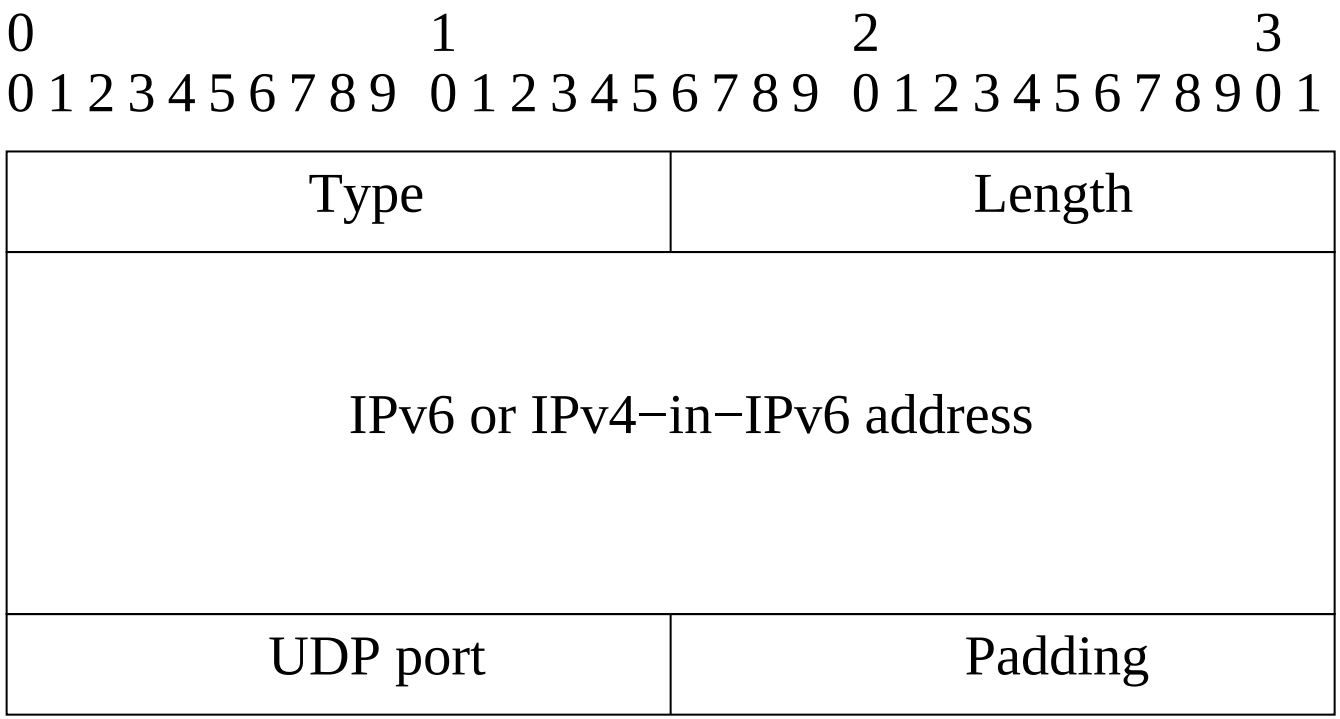
- Avoiding a hairpin translation
  - the Initiator may attempt sending an I1 message to the Responder's IP
  - R1 arrives within a certain time interval - the normal HIP BE proceeds
  - R1 does not arrive within timeout - the use of NAT traversal extensions
- The Simple Traversal of UDP through NATs (STUN) define a generic mechanism to detect the NAT presence

Header format. UDP encapsulation of HIP packets



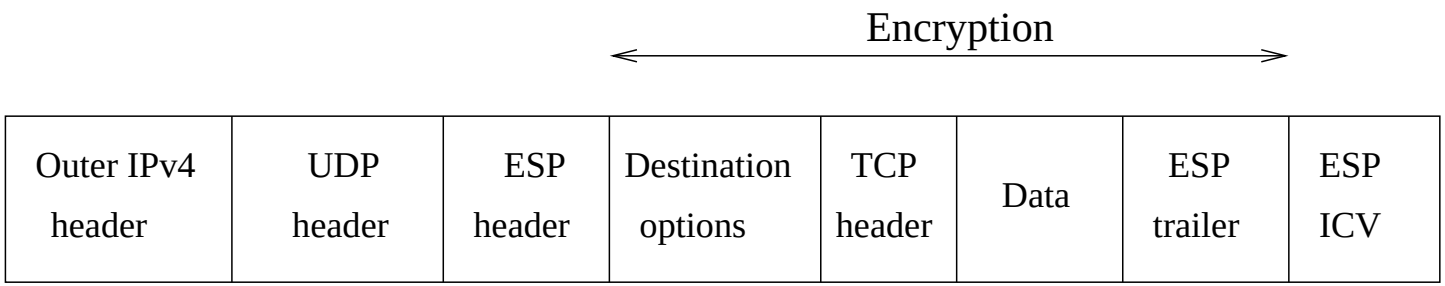
- for control packets, the UDP header is followed by a HIP header with parameters
- for data packets, the UDP header is followed by an ESP content in the BEET mode
- the UDP length and checksum fields are computed as usual, following RFC768
- the HIP checksum is not used and is set to zero

Header format. FROM_NAT and VIA_RVS_NAT



- FROM_NAT parameter contains an IPv6 or IPv4-in-IPv6 address of the NAT
- VIA_RVS_NAT parameter has the same structure as FROM_NAT
- VIA_RVS_NAT parameter serves a similar role as the VIA_RVS parameter, mainly debugging and diagnostic

Header format. ESP BEET-mode packet encapsulated in a UDP datagram

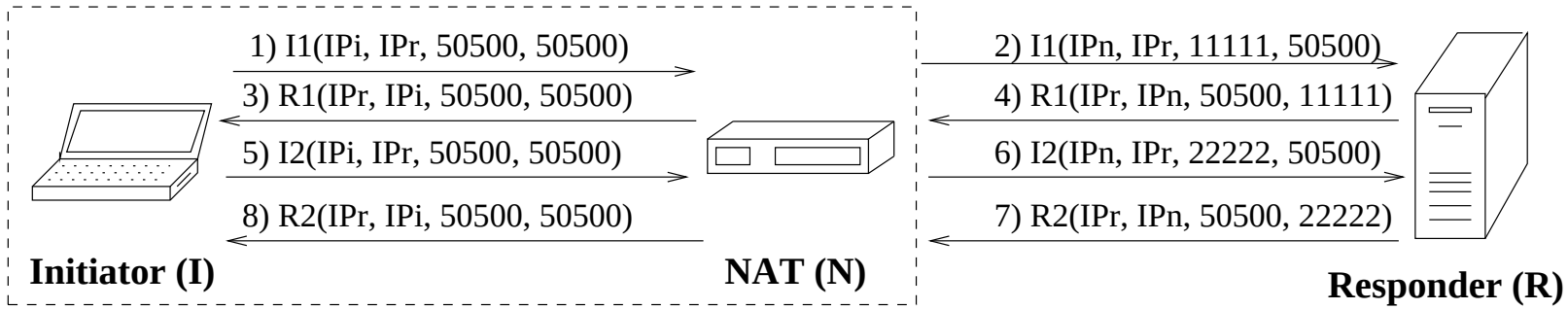


- traversal of ESP tunnel and transport mode packets over NAT using UDP encapsulation - RFC3948
- HIP uses a new ESP mode BEET - extensions to the existing ESP traversal technique are necessary
- with BEET transport layer checksums are based on HITs
- the UDP header is inserted between the IP and ESP header
- the UDP checksum is calculated using the IP addresses

Header format. ESP BEET-mode packet encapsulated in a UDP datagram (cont.)

- destination IP options are placed after the UDP and ESP headers
- the HIP checksum is set to zero when computing the UDP checksum
- the receiver checks the UDP checksum and drops it if the checksum fails
- the receiver performs BEET verification and decryption
- the UDP checksum already protects the entire packet
- the HIP checksum is not verified by the receiver

Initiator behind a NAT



- the Initiator is located behind the NAT, the Responder in a public Internet
- IP_i - an IP address, private or public, of the Initiator
- IP_r - an IP address of the Responder
- IP_n and IP_s - public IP address of NAT and of rendezvous server
- the UDP port 50500 is used for HIP NAT traversal
- ephemeral UDP ports are shown as 11111 or 22222

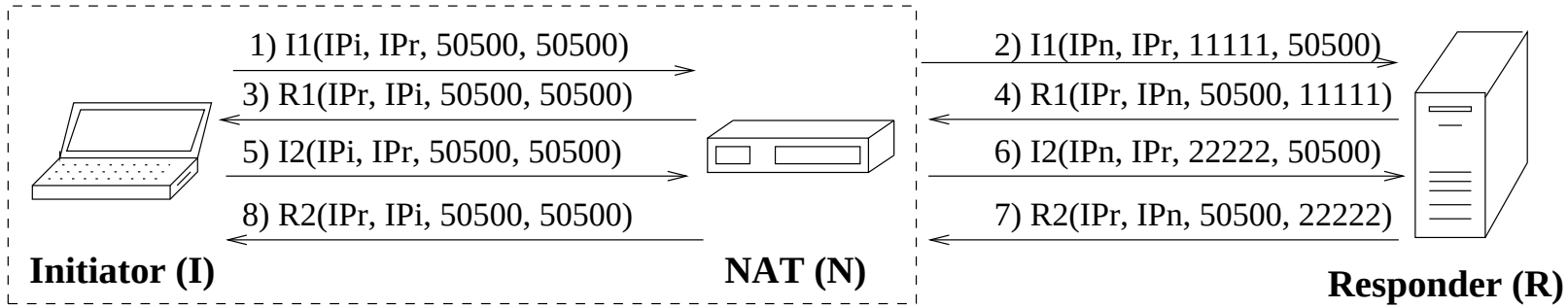
Initiator behind a NAT (cont.)

- The Initiator starts the BE with an I1 message
  - the destination UDP port is set to 50500
  - the source port is recommended to set to 50500
  - the source port is can be set to randomly chosen unused port - the advantage of supporting multiple HIP clients behind the same NAT
- The HIP Responder processes a UDP-encapsulated I1 packets
  - the I1 packet processing as for the normal BE after decapsulation
  - the source port is set to 50500
  - the source IP address is set to the address at which the message was received
  - UDP encapsulation must be used

Initiator behind a NAT (cont.)

- Often the NAT has sufficiently long state timeout
  - the same port can be used for I1-R1 and I2-R2 messages
- Implementations must support other scenario
  - the NAT state expires after the R1
  - a new port can be assigned to I2 and R2

Initiator behind a NAT (cont.)

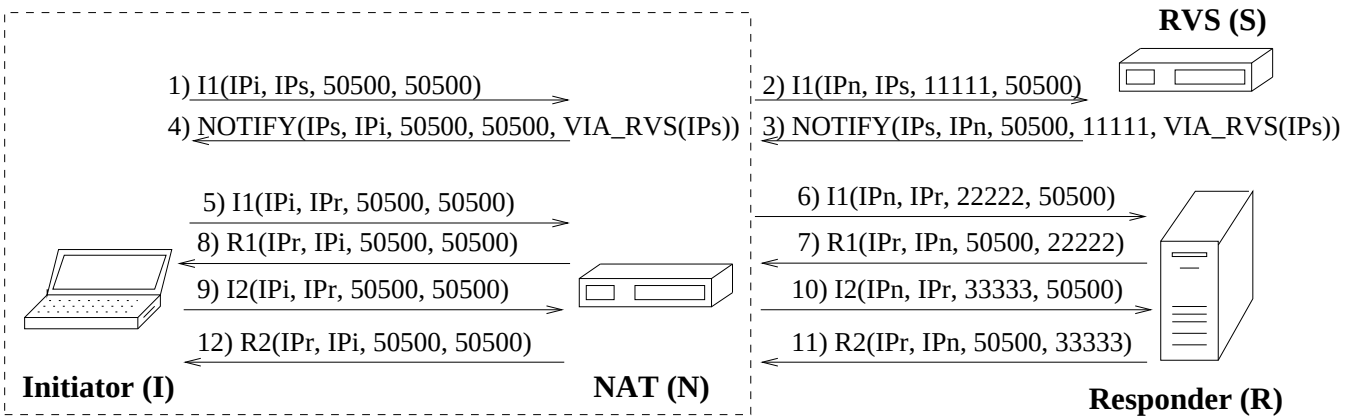


- the Initiator sends an I1 encapsulated into UDP to the Responder's public IP; 50500 - source and destination UDP ports
- the NAT translates the source UDP to 11111 and replaces the source IP with its own public IP
- the Responder processes the I1 and replies with R1, the destination IP and UDP are taken from I1
- the NAT replaces the destination address and port with the private Initiator's IP and port 50500
- the Initiator processes the R1 and sends I2 with the same port and address as I1
- the mapping state was expired - new port is assigned by the NAT (22222)
- the Responder replies with an R2 to the new port taken from the I2

Initiator behind a NAT (cont.)

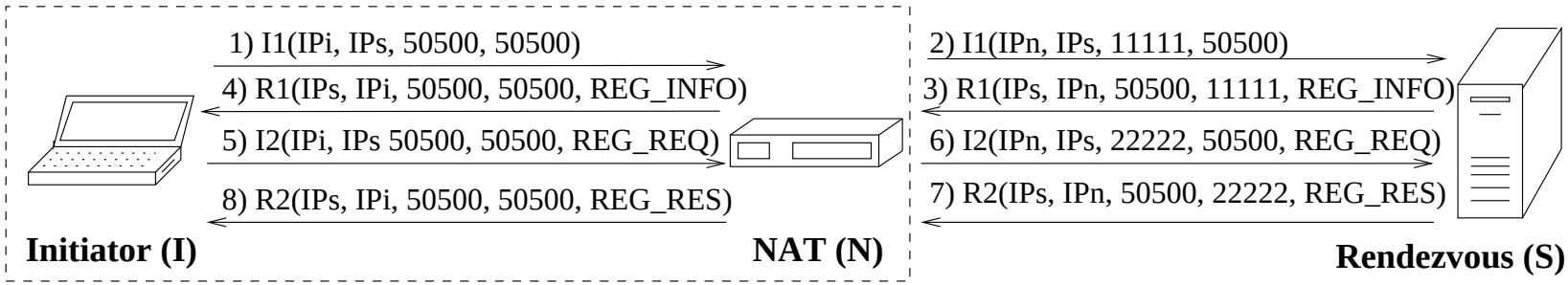
- The HIP BE over UDP completes
  - the data is carried using the ESP BEET mode encapsulated to UDP
  - ESP SA with SPIs are set up normally in each direction
- Addresses at the Initiator
  - the inner source and destination address are local and peer's HIT
  - the outer source and destination are local private IP and the public IP of the peer
- Addresses at the Responder
  - the outer and destination addresses are the local public IP and the public IP of the NAT
- The Initiator and Responder use the same IP for ESP packets as for the BE

Initiator behind a NAT using a rendezvous server



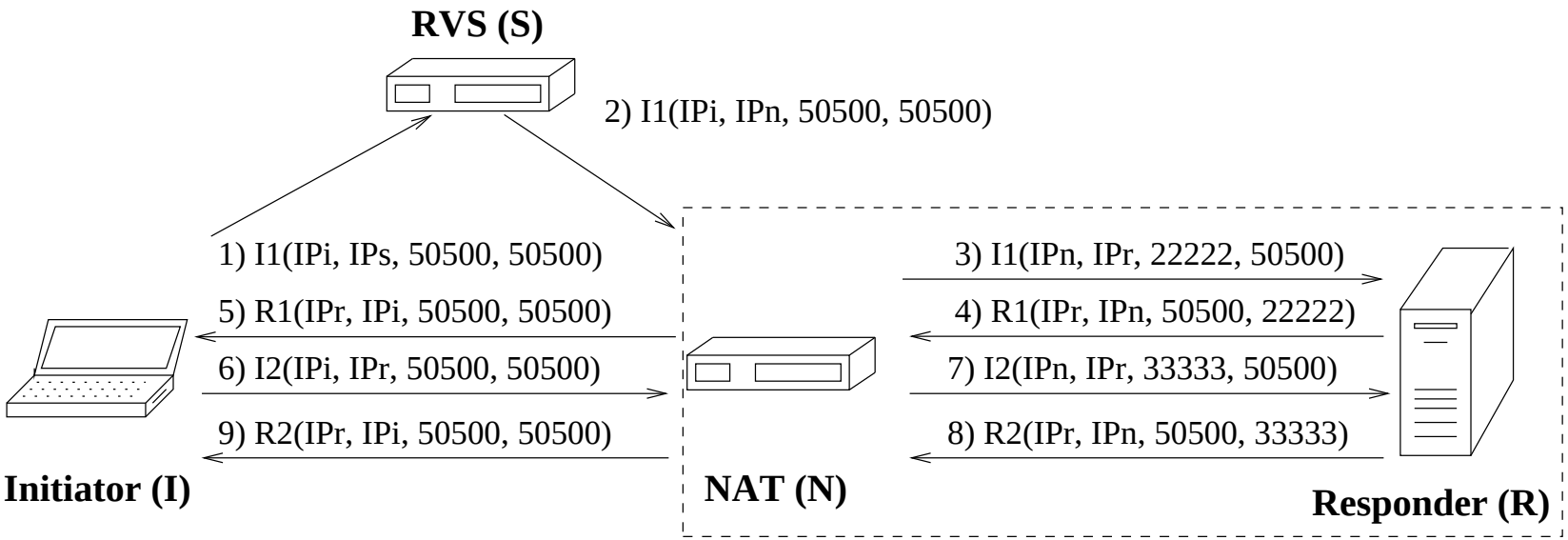
- the Initiator is behind the NAT, the Responder is in a public network
- the RVS listens on UDP port 50500 for incoming I1 from the Initiator
- only some NATs allow any host to send UDP packets through a pinhole open to the RVS
- the RVS replies to an I1 with a NOTIFY
- the Initiator receives the Responder's address and re-sends I1 to the Responder
- the retransmissions are made to the RVS - NOTIFY messages are unprotected by signatures
- the Responder replies with R1 to the Initiator through a pinhole created in a NAT

Registration to a RVS from behind a NAT



- the process is similar to the usual registration, the messages are encapsulated to UDP
- the Initiator sends I1 to the UDP port 50500 at the RVS
- RVS replies with a UDP-encapsulated R1 containing the REG_INFO
- the Initiator sends REG_REQ in I2 and receives REG_RESP from the RVS in R2
- successful registration - the Responder sends keep-alive messages with the same UDP port as in I2 to the RVS to maintain a pinhole open in the NAT

Responder behind a NAT



- the BE towards the Responder behind a NAT
- the host tries to initiate a BE - an I1 packet to the RVS without encapsulation
- the RVS places I1 to a UDP, zeros the HIP header checksum, sends the datagram to the NAT
- the destination address and UDP port of the datagram are set to the NAT's public IP and to the value stored from I2
- the source IP of the datagram is the public IP of the RVS and the source UDP port is 50500

Responder behind a NAT (cont.)

- the RVS adds a FROM parameter (Initiator's public IP protected by the RVS_HMAC) to I1
- the IP header checksum is recalculated by the RVS
- the NAT has the pinhole still open (keep-alive messages) and replays the datagram to the Responder
- the Responder processes I1 and sends UDP-encapsulated R1 to the Initiator's public IP
- the Responder may add VIA_RVS_NAT to R1 containing the IP and UDP port that RVS used when forwarding I1
- the R1 is sent to UDP port 50500 and the Initiator's public IP
- the Initiator implementing NAT-traversal extensions listens to UDP port 50500
- the Initiator determines that the Responder is behind a NAT
- UDP-encapsulated I2 and R2 are forwarded by NAT

Responder behind a NAT (cont.)

- A BE completes
  - Initiator and Responder establish a SA in using UDP-encapsulated BEET mode
- The inner addresses in SAs are HITs
- Outbound SA at the Initiator
  - the outer source address is set to Initiator's public IP
  - the outer destination address is set to the NAT's public IP
  - the UDP source port is 50500
  - the destination UDP port is derived from the R2

Responder behind a NAT (cont.)

- Inbound SA at the Initiator
  - the outer source IP is taken from the R2
  - the outer destination address is the public IP
  - the UDP source port is taken from the R2
  - the UDP destination port is 50500
- Outbound SA at the Responder
  - the outer source address is Responder's private IP
  - the outer destination address is the Initiator's public address
  - the source UDP port is the same as the source port in R2
  - the destination UDP is 50500

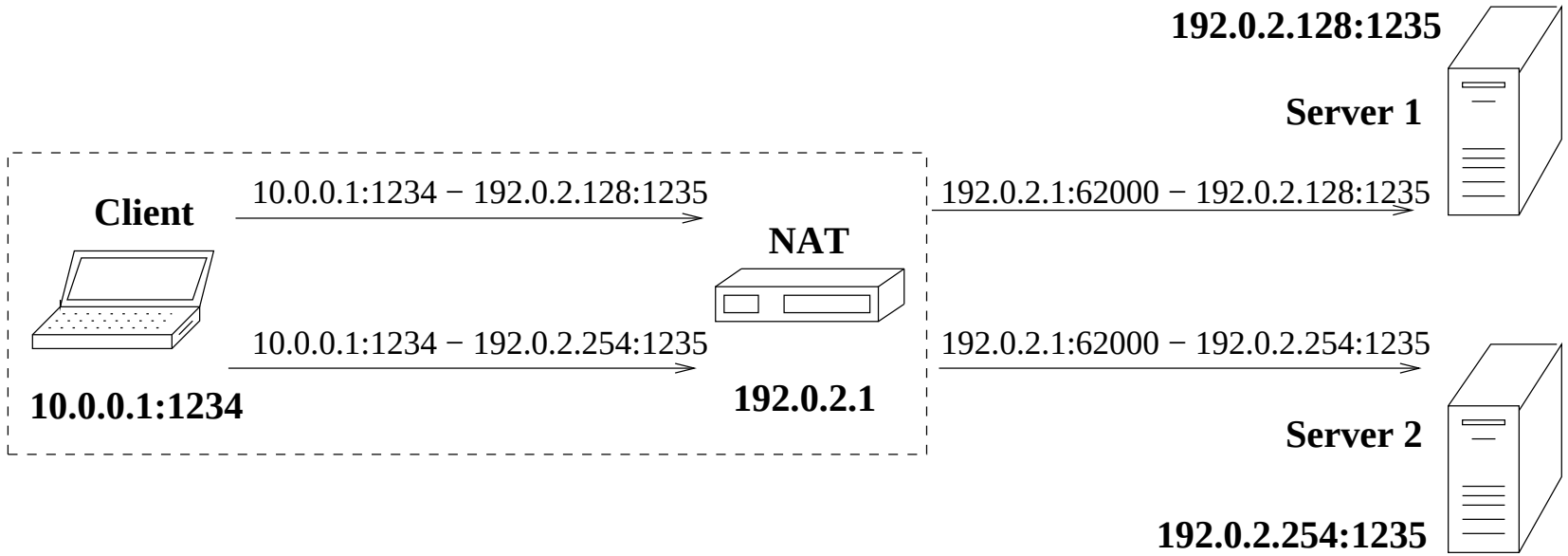
Responder behind a NAT (cont.)

- Inbound SA at the Responder
  - the outer source address is set to the Initiator's public IP
  - the outer destination is the private address
  - the UDP source port is 50500
  - the destination port is same as the source port of R2
- The RVS mechanism
  - NOTIFY - VIA_RVS with public address and port information of the Responder's NAT
  - the Initiator can use the NAT information to launch a DoS attack
  - more secure mechanism - relay I1 and R1 through the RVS to avoid exposing NAT information to a possibly malicious Initiator

Initiator and Responder behind a NAT

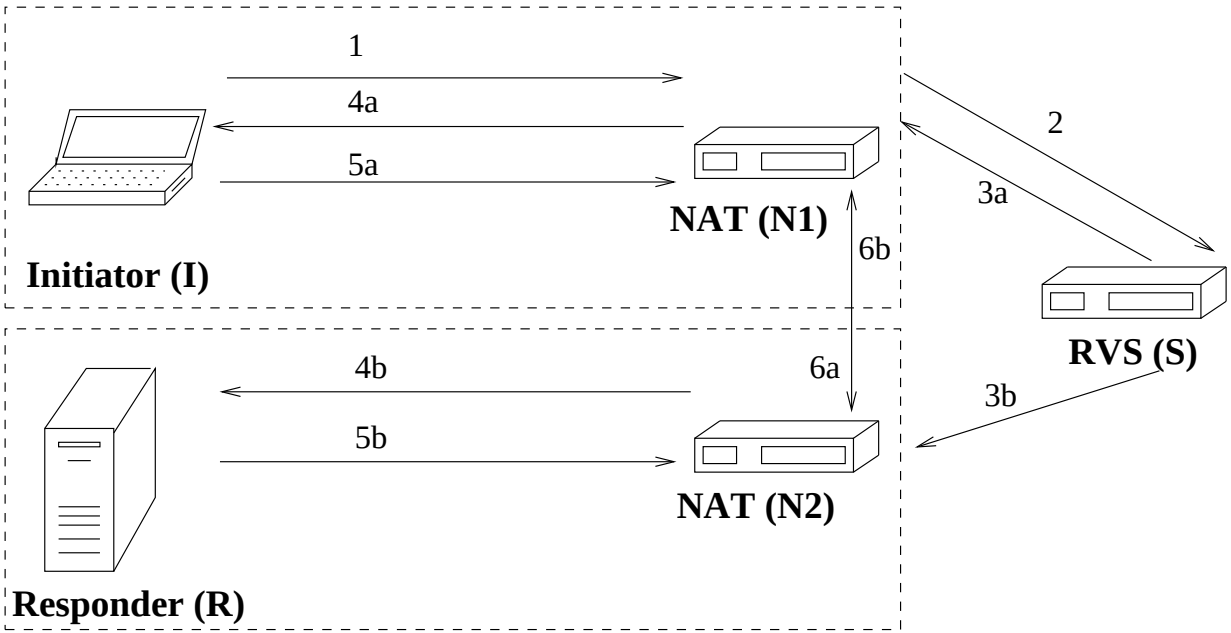
- The HIP BE and IPsec data traffic in the scenario when both Initiator and Responder are located behind two different NATs
- Both NATs should support endpoint-independent mapping
  - the NAT translates a packet from a given internal address and port to any external address and port in the same way

Address and port translation by a NAT performing endpoint-independent mapping



- the client initiates two parallel sessions to two separate servers simultaneously
- the NAT uses the same public address and port 192.0.2.1:62000 for mapping for both servers
- a client can be identified by the address and port assigned by the NAT

Initiator and Responder behind NATs



1) I1(IPi, IPs, 50500, 50500) 2) I1(IPn1, IPs, 11111, 50500)

3a) NOTIFY(IPs, IPn1, 50500, 11111, VIA_RVS_NAT(IPn2, 44444))
3b) I1(IPs, IPn2, 50500, 44444, FROM_NAT(IPn1,11111), RVS_HMAC)
4a) NOTIFY(IPs, IPi, 50500, 11111, VIA_RVS_NAT(IPn2, 44444))
4b) I1(IPs, IPr, 50500, 44444, FROM_NAT(IPn1,11111), RVS_HMAC)
5a) I1(IPi, IPn2, 50500, 44444)
5b) R1(IPr, IPn1, 50500, 11111, VIA_RVS_NAT(IPs,50500))
6a) I1(IPn1 IPn2, 11111, 44444)
6b) R1(IPn2, IPn1, 44444 11111, VIA_RVS_NAT(IPs,50500))

- the Responder registers with its RVS
- the Initiator sends a UDP-encapsulated I1 to the RVS
- the RVS listens on UDP port 50500 for incoming packets
- the RVS forwards UDP-encapsulated I1 to the UDP port that the Responder stored during registration
- the RVS adds FROM_NAT, source UDP port, RVS_HMAC to the I1
- the RVS sends a NOTIFY packet to the Initiator

Initiator and Responder behind NATs (cont.)

- the Initiator sends own NOTIFY to the Responder through its NAT - a pinhole in the Initiator's NAT
- R1 creates a pinhole in the Responder's NAT
- I2 and R2 packets flow directly between Initiator and Responder through pinholes
- prevention of the translation state expiring in NAT - NOTIFY keep-alive packets

Initiator and Responder behind NATs (cont.)

- The BE completes
  - Initiator and Responder transmit data packets in UDP-encapsulated BEET mode
  - IP addresses and UDP ports for SA are the same as in the BE
- Temporarily no data is sent over a HIP association
  - keep-alives are necessary to prevent premature state expiration in NATs
  - only outgoing packets refresh the NAT state - both hosts behind NAT should transmit keep-alive packets to each other

Multihoming and mobility with NATs

- Multihoming extensions - SAs between two hosts over multiple paths
  - some paths can pass through NATs
  - some paths can have direct IP connectivity
- Handle multihoming for two ways
  - some available paths use UDP encapsulation to traverse NAT
  - some paths use plain HIP packets
  - a host using NAT traversal extensions can change its IP using HIP mobility mechanism
- A host using NAT traversal extensions can change its IP using HIP mobility mechanism
- Simultaneous hosts mobility or ”double-jump” is not supported

Multihoming and mobility with NATs (cont.)

- Possible host mobility scenarios
  - moving between a private (behind a NAT) address space and the public Internet
  - moving between private address space of two different NATs
  - changing the IP address behind the same NAT
- The correspondent node location
  - a public or private network
  - with or without a RVS
- Some combination may not work depending on the type of NAT

Multihoming and mobility with NATs (cont.)

- A mobile host changes its location
  - the host attempts to detect the presence of a NAT
  - the NAT is not detected - an UPDATE is sent as a plain HIP packet
  - the NAT is detected - an UDP-encapsulated UPDATE is sent
- The mobile host sends an UPDATE to all its peers through their RVS and directly to peers
- The peers reply with a plain UPDATE or UDP-encapsulated UPDATE
- The RVS relays an UPDATE packet
  - a FROM parameter for a plain HIP packet
  - a FROM_NAT parameter for UDP-encapsulated packet
  - HMAC protects both parameters

Multihoming and mobility with NATs (cont.)

- The peer host replies to the UPDATE
  - VIA_RVS or VIA_RVS_NAT are added
- Sending of private IP addresses in the LOCATOR is disallowed in current specifications for NAT traversal
  - the host has no public IP - the LOCATOR contains only the type and length fields
  - assumption - during encapsulation the UDP header contains the necessary information to access the private locator
  - restrictions may be harmful in the hairpin translation - hosts within the same private address communicate without involving the NAT
  - private IP in a LOCATOR - useful to support mobility within a single address space

Traversing firewalls

- Many existing firewalls support only TCP and UDP packets
- Packets, such as HIP or IPsec packets are dropped by firewalls
  - a user has control - enabling HIP traffic traversal without disabling the firewall altogether is not possible
  - the user cannot modify firewall rules - the firewall allows incoming UDP packets only if a packet from the same port and address has been recently transmitted in outbound direction from the firewall
- The firewall allows outgoing UDP packets to port 50500
  -

Traversing firewalls (cont.)

- The Initiator is behind a stateless firewall
  - the firewall should be configured to allow all incoming UDP packets from UDP port 50500 to any internal IP address and UDP port
- The Responder is behind firewall
  - the firewall should allow incoming UDP packets to port UDP 50500 from any IP address or port

Requirements for HIP-aware middleboxes

- Middlebox cannot inspect the protocol headers inside the IPsec ESP payload of HIP data packets
- The HIP control protocol is designed to reveal sufficient information in signaling packets
  - HIP-aware middleboxes can separate and follow subsequent flows
  - middleboxes can potentially filter out unwanted traffic
  - HIP more suitable to traverse middleboxes than other protocols that set up ESP SA
- Two approaches are possible to traverse new ”architectured” middleboxes
  - HIP-specific modifications monitoring the HIP BE and learn SPI
  - generic signaling mechanism to middleboxes

Requirements for HIP-aware middleboxes (cont.)

- SPINAT combines traversal of HIP control and data packets in a single solution
  - the middlebox identifies the ESP packets by destination IP and SPI
  - verification before creating internal state that HIP control messages are authentic
  - the middlebox learns the necessary flow identifiers from ESP_INFO in I2 and R2 or from LOCATOR in UPDATE
- HIP hosts located behind middleboxes and willing to be reachable from outside private domain
  - it is necessary to perform an explicit registration procedure to the middlebox
  - maintaining the registration state by keep-alive messages

Requirements for HIP-aware middleboxes (cont.)

- Signaling of SPI values from host to middleboxes
  - sufficient to enable traversal of HIP data traffic through middleboxes
- Extensions to the signaling mechanism includes HIP control traffic
  - enabling traversal of HIP control traffic
- The NSIS NAT/FW traversal protocol
  - setting up the traversal state in all middleboxes along the path

Requirements for HIP-aware middleboxes (cont.)

- Implementation of HIP-aware middlebox
  - must not introduce new possibilities of traffic attack
  - must be able to intercept and authenticate HIP signaling messages
  - should drop any HIP traffic that does not have a properly set up state in the middlebox or fail authentication
- Ideally, a HIP-aware middleboxes
  - function in a scenario where packets can go through different routers
  - packets can enter the local network through one firewall and leave through another
- State synchronization protocol - agreement on common flow identification parameters

HIP-aware firewall

- A firewall separates an organization's network from the rest of the Internet
- Access Control List (ACL) - packets forwarding policies in the firewall
- Current firewalls can filter out packets based on
  - IP addresses
  - transport protocol
  - port values
- Port values are often unprotected in data packets
  - an attacker can often penetrate the firewall defenses
- Legacy firewalls often disallow IPsec traffic and drop HIP control packets

Flow identification

- Designing of firewall that understands HIP messages and authenticates hosts based on HI
- Firewalls can be stateless
  - packet filtering based only on the ACL
  - simple to implement
  - only coarse-grain protection
  - performance can be high - packet processing requires little memory or CPU resources
- Firewalls can be stateful
  - can follow and remember packet flows
  - determines if a packet belongs to an existing flow or starts a new flow

Flow identification (cont.)

- Stateful firewall
  - protocol header information - a flow identifier classifies packets
  - the flow is removed when the flow terminates or after a timeout
- A firewall can drop suspicious packets
  - checksum is failed
  - sequence number is outside of the current sliding window
- A transparent firewall
  - no requirement to register or even know of the firewall existence
- An explicit firewall
  - registration and authentication from the host are required

Flow identification (cont.)

- Flow identification by HIP-aware firewall
  - HITs, SPI values and IP addresses of communication hosts
  - linking together the HIP BE and consequent IPsec ESP data packets
  - the firewall only allows ESP packets with a known SPI value and arriving from the same IP as during the BE
- Host changes its location
  - the firewall learns about the changes by following the mobility update packet
- Registration to the firewall using the usual procedure
  - the HIP host and the firewall authenticate each other

Advanced extensions. Opportunistic mode

- The HIP opportunistic mode - a challenge to the firewall authentication
- The firewall allows packets based on the source and destination IP rules
  - an initial state creating based on I1 addresses
  - updating the state by including HITs after receiving the R1
  - I1 packet is disallowed by the firewall - NOTIFY packet
- The HIP architecture advantage
  - enabling the firewall to authenticate packets using the signatures in addition to HIT-based ACL
  - the Responder's identity in plain text in the R2 - firewall can verify the Responder's signature
  - the Initiator's identity is encrypted in the I2

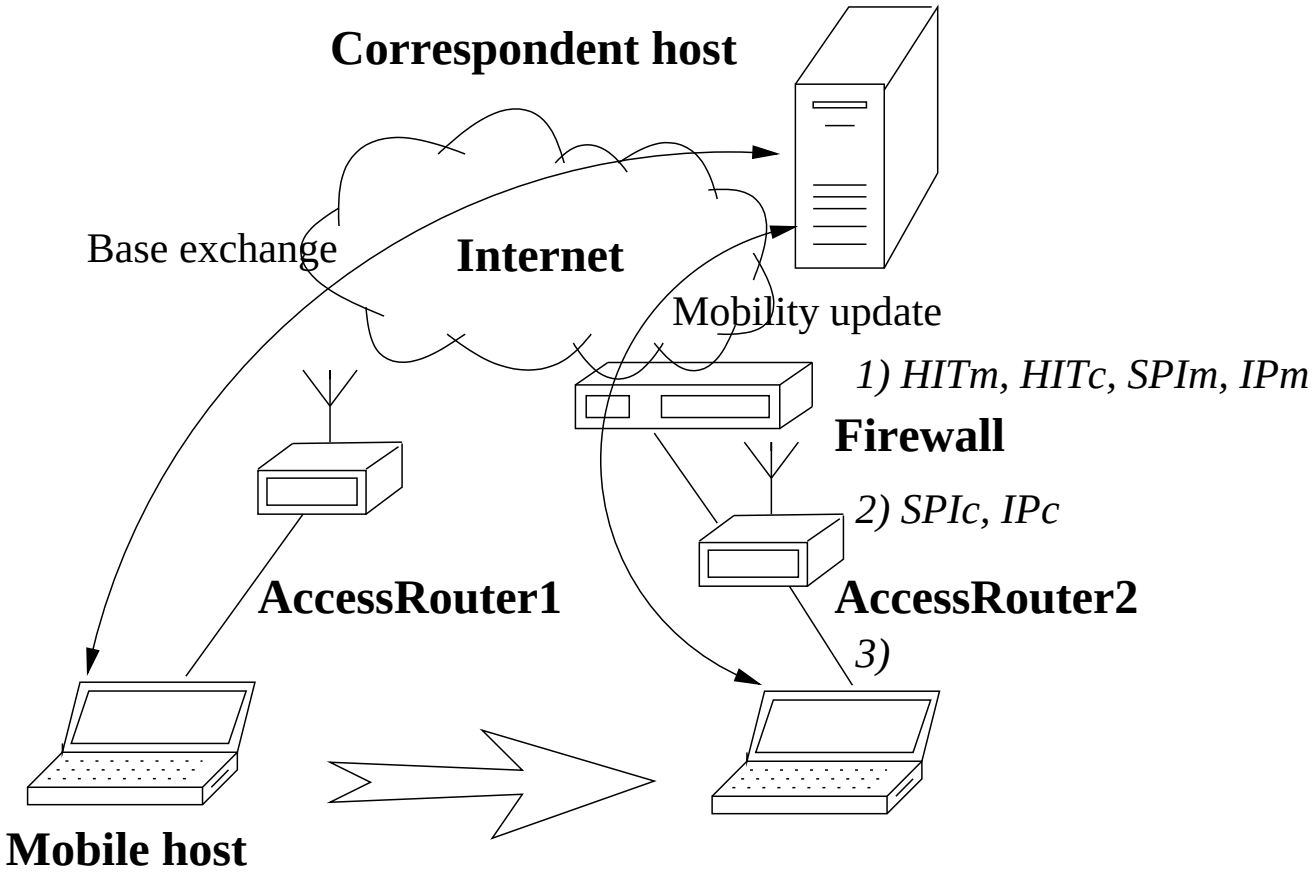
Advanced extensions. Rendezvous server

- The presence of a firewall complicates the use of a RVS
  - the RVS changes the IP in the I1 before forwarding it to the Responder
  - I1 and R1 can have different addresses arriving to the firewall
  - the firewall can look inside I1 and take the Responder's address in the FROM
- Final state configuration
  - the firewall takes IP and SPI from I2 and R2 that do not travel through the RVS

Advanced extensions. Mobility and multihoming

- The host can add and remove IP to the active address set
- The firewall needs to follow control messages
  - prevention of address hijacking and redirection attacks
- By default, mobility with rekeying and address verification
  - three UPDATE - sufficient for a firewall to learn new SPI value and IP addresses
- Mobility without rekeying
  - the firewall is not able to verify the update
  - the firewall is likely to drop subsequent packets coming from a different address

A mobile host moves behind a firewall



1)UPDATE(ESP_INFO, LOCATOR, SEQ)

2)UPDATE(ESP_INFO, SEQ, ACK, ECHO_REQUEST)

3)UPDATE(ACK, ECHO_RESPONSE)

- the host sends the first UPDATE after moving to the new network
- the firewall extracts the information from the first and second UPDATEs

Advanced extensions. Certificates

- Possibility for sending certificates in a special parameter in BE messages
  - the Simple Public Key Infrastructure (SPKI) format
  - using during registration process for authentication
- Without a certificates the firewall ACL includes public key of each host
- Certificates simplify the authentication process for HIP-aware firewall
  - storing only the public key that issued the certificate is sufficient
  - during the BE - extracting the host public key from the certificate and its verification
  - connection is closed - the host public key can be purged from the memory

Advanced extensions. Certificates (cont.)

- The limited lifetime of a certificate would control when the firewall approves the authorization
- Short lifetime
  - the host should apply fore new certificates more frequently
  - reduction of the need for the firewall to check Certificate Revocation Lists (CRL) for compromised certificates

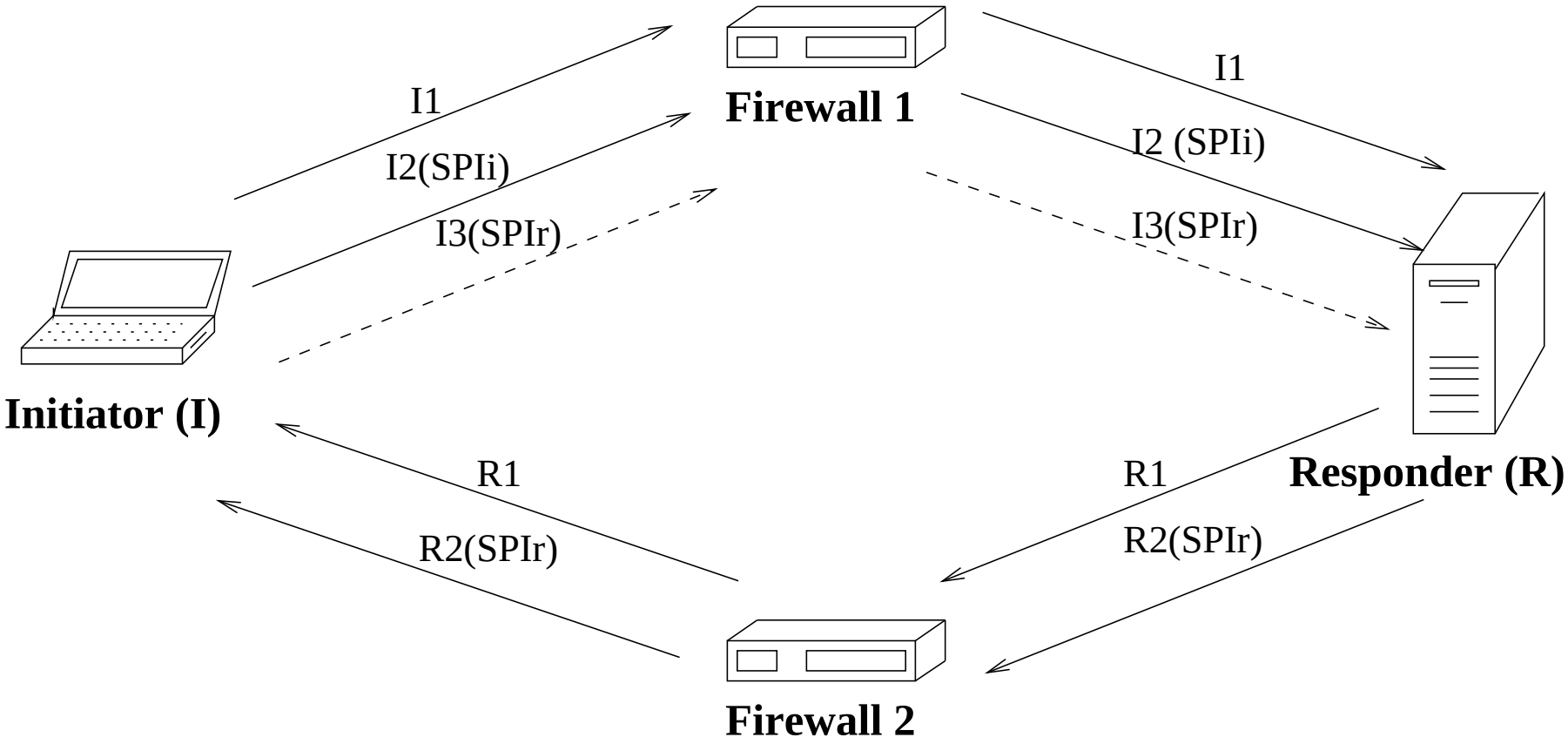
Asymmetric routing

- Asymmetric Internet routes
  - packets do not pass the same routers and middleboxes in two directions
- Problems with creating a proper state in middleboxes
- A case where outgoing and incoming HIP packets go through separate firewalls
  - a firewall 1 sees only packets from Initiator to Responder
  - the firewall 1 does not know the SPI sent by the Responder
  - the firewall 1 cannot pass through subsequent ESP packets with this SPI

Asymmetric routing (cont.)

- One solution for asymmetric firewall
  - extension of the HIP BE with a fifth packet, I3
  - I3 incurs additional signaling cost for all HIP associations
- An alternative solution
  - an SPISIG packet from a HIP host to the firewall
  - the SPISIG signals the SPI value from the opposite direction
  - the SPISIG assumes that both hosts are aware of the firewall presence
  - the SPISIG requires that hosts register to the firewall

Asymmetric routing with firewalls



- I3 packet is introduced in the HIP base exchange to deliver responder SPI to the firewall 1

Security risks

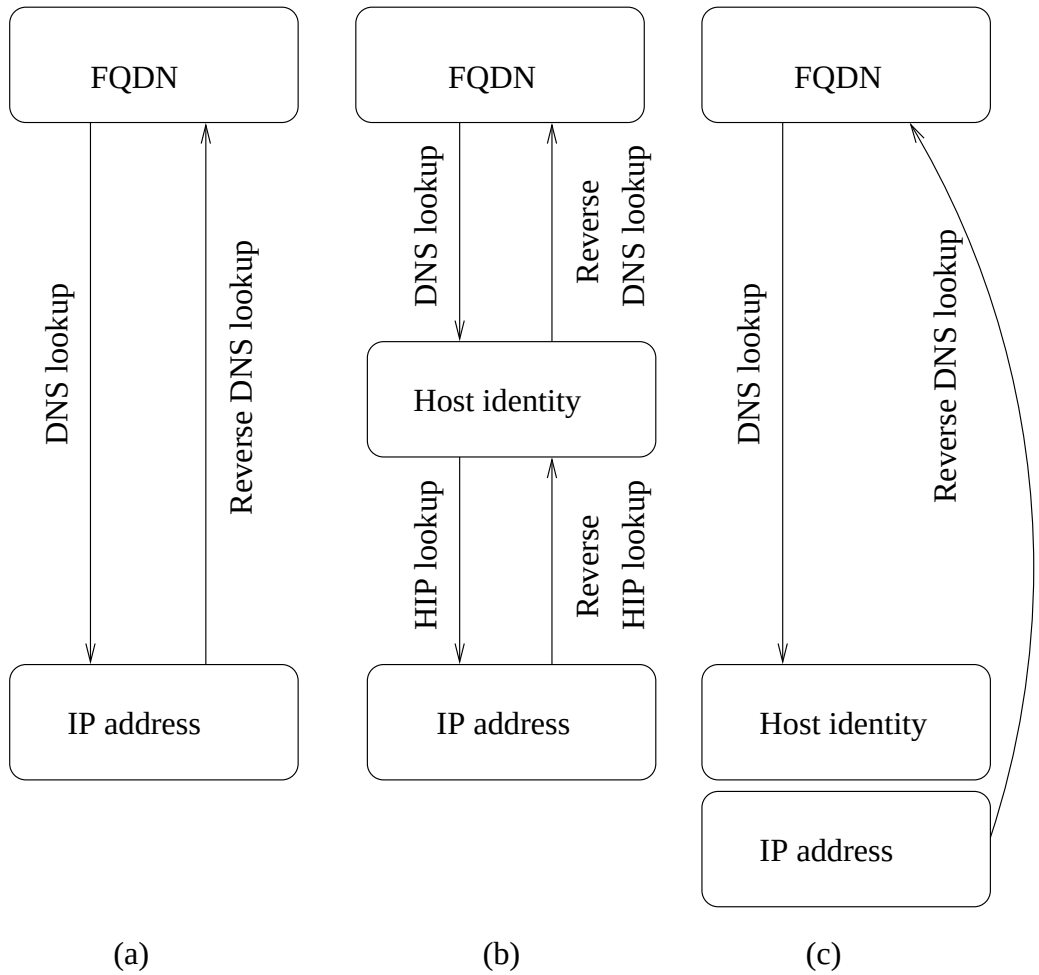
- The Initiator creates a state at the firewall
- The Initiator could mount a DoS attack
- The typical deployment scenario
  - a firewall allows connection from the local network to the Internet
  - the firewall drops all connection attempts from the Internet to the local hosts
  - incoming traffic related to the existing state initiated by a local host is permitted
  - hosts in the local network are trusted and their authentication may be unnecessary
  - the load and the risk of DoS for the firewall are reduced

Security risks (cont.)

- The Responders in the scenario
  - the responders are untrusted
  - packets need authentication in the firewall
  - the HIP BE contains the Responder's public key in plain text
  - the firewall can check signature; no need to store the Responder's HI locally
- Prototype of HIP-aware transparent firewall is implemented for HIPL

Name resolution

Name resolution mechanisms



- (a) DNS resolution in the current Internet, (b) logical resolution for HIP, (c) resolution proposed in HIP DNS extension

Problem statement of naming

- DNS resolution in the current Internet
  - a host resolves a FQDN to a set of IP addresses using a DNS lookup
  - an IP of a local DNS is pre-configured or obtained from a DHCP
  - the reverse DNS lookup maps the IP to one of the host FQDN names
- Logical resolution for HIP
  - HIP splits the resolution process in two logical parts
  - the host FQDN name is resolved to a set of HIs
  - each HI can be resolved to a set of IPs
  - HI resolution can be implemented using a DNS lookup
  - reverse DNS lookup maps a HI to one of the host FQDN names
  - reverse HIP lookup maps an IP to one of host HIs

Problem statement of naming (cont.)

- Resolution proposed in HIP DNS extension
  - two logical steps are combined into one DNS lookup
  - the DNS lookup returns both the HI and a set of IP
  - the HI can be stored in a new DNS Resource Record in a HIP-aware DNS
  - the HI also can be stored in a form of HIT in AAAA field in legacy DNS
  - benefits of simplicity and reduced latency
- HIP DNS extension approach raises several issues

Problem statement of naming (cont.)

- Possibility of direct communication
  - current Internet - DNS as an optional component
  - direct communication is always possible
  - in practice, DNS is necessary for most users
  - only the HI of destination is known - HIP hosts experience problem
  - IP resolution - a critical component to enable direct communication
- Dependency on DNS upgrade
  - HIP hosts can revert to plain IP communication
  - HIP hosts can attempt to association establishment in an OPP mode
  - full benefit of HIP - uploading of HI to DNS and DNSSEC use
  - not all hosts have sufficient authority to change DNS information

Problem statement of naming (cont.)

- Reverse host identity lookup
  - simple HIP resolution model does not provide reverse HIP lookup from a HI to a FQDN of from an IP to a HI
  - emulating lookup - from an IP to a FQDN, from a FQDN to the HI
  - a FQDN storing the HI and returned by reverse lookup can differ
  - IP structure enables reverse IP lookups from a DNS root in.arpa
  - HIT is random-looking flat data - difficult to reproduce the reverse IP mapping mechanism for HITs

Problem statement of naming (cont.)

- DNS lookups over HIP
  - DNS lookups - short operations, connectionless UDP
  - connection establishment - additional latency to DNS resolution
  - HIP association establishment is more heavyweight than TCP
  - access to DNS over HIP is unlikely to be popular
  - a local DNS performs recursive lookups on behalf of a client - DNS lookups over HIP may be an attractive option
  - the HI of the DNS is provided with its IP by manual configuration or a DHCP extension - requirements

Problem statement of naming (cont.)

- Middlebox traversal for lookups
  - middleboxes can prevent lookups from a local network
  - DNS requests traverse middleboxes well
  - other resolution mechanisms can be blocked
  - host cannot resolve HIs to IPs - direct communication is impossible
  - proposals for name resolutions should traverse through middleboxes
- Privacy of lookups
  - host's intention to communicate with a host - privacy-violating issue
  - HIP lookups aggravate the problem - HIs are long-lived names, can be cryptographically proved to belong to a host storing a private key
  - encryption of lookups to prevent eavesdropping; blur of lookup history

Problem statement of naming (cont.)

- Efficient mobility
  - HIP enables mobility of one association end-point
  - infrastructure support requirement for initial RVS with a mobile hosts or simultaneous hosts mobility
  - the simple HIP resolution model cannot provide fast updates to the IP of the mobile host
  - the two step resolution process - faster updates than DNS, e.g. a DHT
  - a HIP host can substitute its with IP of RVS that relays packets to the mobile host
  - the RVS can relay packet a single packet to a mobile host
  - mobile host sends packet directly to a peer
  - relaying all packets - higher delivery rate

Problem statement of naming (cont.)

- Interoperation with legacy hosts
  - communication between HIP-unaware and HIP hosts is possible
  - the HIP RVS extension stores an IP of the RVS in a DNS using a RVS RR
  - the only record stored for a HIP host
  - legacy host cannot obtain an IP of a HIP host
  - HITs are stored in AAAA fields of a legacy DNS - a legacy host can misinterpret a HIT for an IP and try to send packets to a HIT
  - HITs are not routable - the legacy host is not able to contact the HIP host, unless the application can try other available AAAA fields

Overview of Distributed Hash Tables

- A traditional hash table
  - a data structure associating keys with values with a hash function
  - a hash function maps a key to a bucket with a simple operation
- Collisions
  - multiple values are hashed to the same bucket and can be resolved by creating a list of values in a bucket
- A good hash function
  - computationally inexpensive
  - keys to values mapping is uniform, without bias to some key space portion

Overview of Distributed Hash Tables (cont.)

- The number of potential collisions is large
  - a useful property of reverse mapping of a hash function
  - a bucket can only be associated with a set of possible keys, not with the exact key that produces the mapping
- A Distributed Hash Table is an extension of traditional tables
  - distributed application running on multiple Internet hosts
  - a host - as a node in the DHT
  - the host is responsible for storing values of certain keys range
  - an Internet host can contact any DHT node to lookup a key in a DHT
  - the DHT nodes route the query towards a node responsible for storing the value corresponding to the key

Overview of Distributed Hash Tables (cont.)

- CAN, Chord, Pastry, and Tapestry - the first DHTs introduced in 2001
- Kademlia and Bamboo optimize certain characteristics of the first DHTs
- Many distributed application utilize DHTs
  - file sharing, cooperative web caching, instant messaging
- DHTs are designed to be scalable
  - support of thousands and more nodes
  - nodes can dynamically join and leave the DHT
  - join and leave disrupt operations DHT structure and trigger recovery
  - *churn* - a frequent change of node membership in a DHT
  - successfully completing lookups under churn - challenging design requirement for DHTs

OpenDHT interface

- OpenDHT - publicly available DHT service running on PlanetLab
  - PlanetLab - a world-wide testbed of hundreds servers
  - the user does not have to run a local DHT node to access
  - registration and accounts are not required to store and lookup data
  - available storage is shared fairly among all users
  - Sun RPC and XML RPC interfaces to access
- OpenDHT is based on Bamboo DHT servers
  - code is publicly available to set up own set of DHT servers
  - relying on a publicly maintained set of servers is more convenient
  - advanced HIP extensions require changes to OpenDHT to operate
  - private set of modified OpenDHT servers is the only alternative until the changes have been incorporate to the official OpenDHT release

OpenDHT interface (cont.)

- OpenDHT stores values using a key
  - a key can be up to 20 bytes long
  - a key - a fixed-length hash of an actual identifier
  - values have a limited Time-to-Live of a maximum of 604800 sec
- *Put* and *get* operations - basic interface
- An XML-RPC interface was used in experiments
  - XML tags corresponding to field names are used to encapsulate data values in the interface messages

General content of a put operation to OpenDHT

Field	Type
application	string
client_library	string
key	max 20-byte array
value	max 1024-byte array
ttl_sec	four-byte integer (max 604800)

- application and client_library fields for the logging of requests
- client_library refers to the name and version of the XML-RPC library used in the query
- the key selects an OpenDHT server to store the value
- tree values of reply:

”zero” - successful put

”one” - failure because of the capacity of the server is exceeded

”two” - a temporal failure, suggestion to re-try the put operation

Content of a get operation to OpenDHT

Field	Type
application	string
client_library	string
key	max 20-byte array
maxvals	four-byte signed integer (max $2^{31} - 1$)
placemark	byte array (100 bytes max)

- the placemark can be used in a subsequent query to obtain additional stored for the key

OpenDHT interface (cont.)

- Overlay Anycast Service InfraStructure (OASIS)
  - selecting the closest available server to the client
  - DNS lookup on the host name ”opendht.nyuld.net”
  - the OpenDHT client obtains an IP of the server with lowest RTT
  - the client can retrieve a list of available OpenDHT servers and itself select the server
- OpenDHT is a public open data repository
  - anyone can place data under any key
  - drawing attack - an adversary places a large number of garbage values under the same key
- Two mechanisms of attack prevention

OpenDHT interface (cont.)

- *Immutable puts*
  - from $k = H(\nu)$ (the key is a hash of the stored value)
  - cannot be removed until their lifetime expires
  - not suitable to secure HIP name resolution - the key value is fixed
- *Signed puts*
  - value and nonce signed by the private key of the client
  - another client retrieves a value stored under the given key - providing of a hash of the client public key
  - the client only obtains value stored with the private key matching the supplied public key
  - suitable to secure HIT to IP mapping

HIP interface to open DHT

- Instantiation of the OpenDHT interface to store HIP data
- A mobile host can re-publish its current IP to the DHT after moving
  - updating DHT is a fast operation compared with DNS
- The Initiator can use the lookup to obtain the Responder's IP
  - first connection attempt is made
  - the peer location information is lost

General content of a put operation to OpenDHT for HIP

Field	Value	Data type
application	hip-addr	string
client_library	implementation-dependent	string
key	128-bit HIT	base64 encoding
value	struct sockaddr	base64 encoding
ttl_sec	lifetime of address	numeric string

- base64 encoding - use of printable ASCII characters (A-Z, a-z, digits 0-9, +, /)
- binary data to base64 encoding - the data is concatenated together and six bits at a time are used as an index to a string ”ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/” to determine the substitution character
- the string is convert back to the binary form at the receiver

Content of a get operation to OpenDHT for HIP

Field	Value	Data type
application	hip-addr	string
client_library	implementation-dependent	string
key	128-bit HIT	base64 encoding
maxvals	implementation-dependent	numeric string
placemark	NULL or copied from server reply	base64 encoding

HIP interface to OpenDHT (cont.)

- The following HIP-related additional operations are under discussion for Open DHT
- Lookup of a FQDN to HIT supplements DNS
  - many users do not have access to the organization's DNS
  - many users do not have the expertise necessary to change the DNS configuration
  - mapping symbolic names instead of FQDN - the possibility to give human-friendly names to devices
- Reverse lookup from IP to HIT
  - security and debugging support in a similar to DNS way

HIP interface to OpenDHT (cont.)

- An Initiator has only the IP of a peer
  - the Initiator can attempt to use HIP OPP mode - possibility of MITM
  - an Initiator can lookup the HIT in the DHT using the IP as a key
- Reduction of the amount of configuration state
  - store the LSI to HIP mapping in a DHT
- LSIs have a local meaning only
  - managing the LSI to HIT mapping and additional delay - significant arguments against such a design
  - LSI to HIT mapping was removed from the OpenDHT HIP specification

HIP interface to OpenDHT (cont.)

- A host publishes a HIT
  - it is better to verify that the host owns the HIT
- The DHT server verification
  - requirement to host to execute the HIP BE before its HIT is published
  - update requirement of the operating system software on DHT server
  - HIP support enabling on the server
  - a few checks insertion to the DHT implementation
- HIP daemon can query IPs of all known HIP upon startup
  - querying each address from OpenDHT - noticeable delay to the user

HIP interface to OpenDHT (cont.)

- The IPs can change often for a mobile host
  - the Initiator can send I1 to the latest known IP and launch a DHT lookup in parallel
  - the I1 timer expires - I1 can be retransmitted to a current address returned from the DHT lookup
- OpenDHT did not support the remove operation
  - the inserted value could bot be removed until Time-to-Live expired
  - an IP changed - the old value remained in the DHT
  - the new value was added to the end of the list of values
  - the last value - the currently reachable address

Overview of overlay networks

- The term *overlay networks*
  - a set of virtual links between hosts on top of existing IP connectivity
  - each virtual link can span several IP hops or physical links
  - a popular solution for P2P systems, multicast and QoS services
- The advantage of overlays
  - easier deployment
  - no requirement of changing of existing routers
- The disadvantage of overlays
  - worse performance
  - indirect routing and higher protocol overhead

Overview of overlay networks (cont.)

- The *stretch*
  - the ratio of the number of IP hops
  - the latency in the overlay route versus in the direct IP route
- The use of overlays - an ”unclean” solution
  - Internet architecture viewpoint
  - the protocol stack is not layered in a classical way - some functions can be duplicated on even in a conflict at different layers
- Ease of the deployment services - an Internet Indirection Infrastructure
  - a rendezvous-based communication abstraction
  - each packet is associated with a destination identifier
  - the identifier is used to deliver the packet (e.g., trigger (id, R))

Overview of overlay networks (cont.)

- i3 provides natural support for mobility
  - a host changes address - the host needs only to update its trigger
  - address changes $R1$ $R2$ - trigger update $(id, R1)$ to $(id, R2)$
  - all packets with the identifier id are forwarded to the new address
  - change is completely transparent to the sender
- The primary aim of the Secure-i3 proposal
  - network architecture that is more robust against DoS attacks
  - the basic idea - hiding the IPs
- The indirection approach
  - straightforward implementation for multicast, mobility, and multi-address multihoming

Overview of overlay networks (cont.)

- Two types of triggers in Secure-i3
  - public - announcement of service existence, well known
  - private - actual communication between sender and the receiver(s) (ones that know the private triggers)
- Three advanced capabilities of Secure-i3
- Public trigger cannot point to the end-host
  - can point only to a private trigger
  - prevent cycles in the infrastructure and malicious misuse of triggers
  - a trigger chain of two right-constructed triggers is used to insert a given identifier into the infrastructure

Overview of overlay networks (cont.)

- To run legacy applications over i3 - a proxy located on the client and the server must be used
  - the proxy transparently intercepts DNS requests
  - the proxy forwards data packets to the i3 infrastructure
- Capability to send data directly
  - recently has been added to i3
  - *shortcuts* allows efficient data transfer between hosts
  - shortcuts do not offer any cryptographic data protection
- Hi3 relies on features from the basic i3 and the Secure-i3

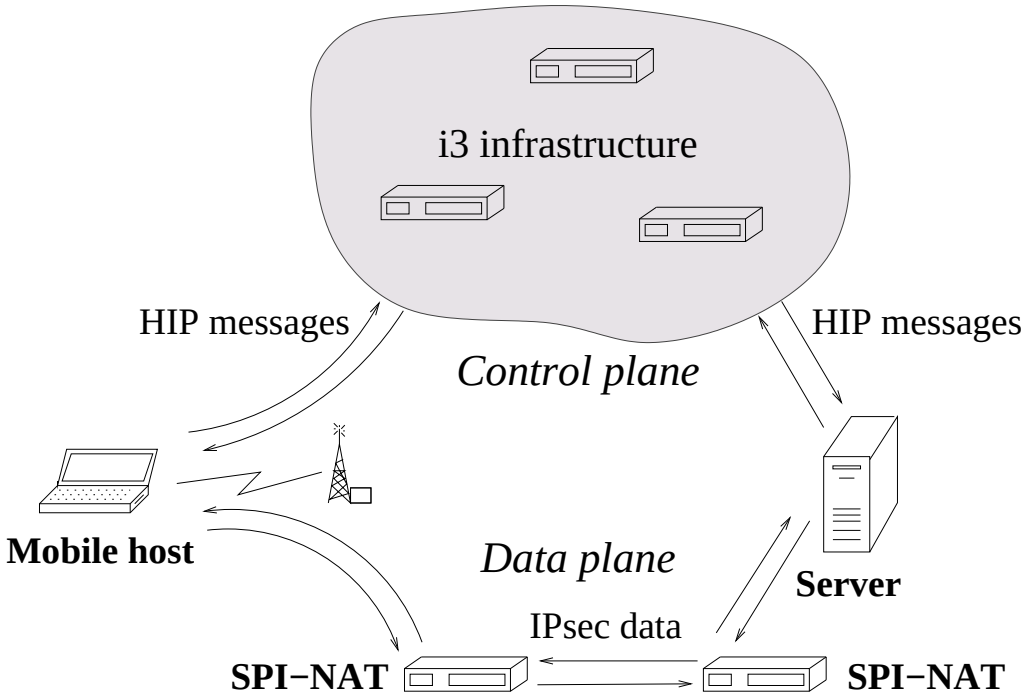
Hi^3

- Hi3 architecture in detail
  - separating session control, actual data delivery, service naming
  - problems of separating and solutions
  - the key advantages
  - perspective of the design
- The original concept of Hi3
  - a HIP RVS and a single i3 server are functionally close
  - direct, IP based end-to-end traffic while using an indirection infrastructure to route the HIP control packets

Outline

- Overview of Hi^3
  - Host Identity Protocol (HIP),
  - Internet Indirection Infrastructure (i^3), Hi^3 architecture
- Hi^3 analysis
  - goals and assumption
- Basic Hi^3 scenarios
  - traversal of HIP messages through the infrastructure
  - Hi^3 parameters and latency for HIP messages
- Scalability analysis
  - workload model, infrastructure size, resilience to DoS attacks

Separating control, data, and naming



- the concept of HIP RVS can be enhanced to an overlay rendezvous infrastructure
- the *Hi3 control plane* - HIP signaling messages
- the *Hi3 data plane* - data traffic flows directly between end-hosts, using plain IP
- HIP and i3 - secure communication for the control plane
- IPsec and IPsec-aware NAT - basic protection to the data plane

Separating control, data, and naming (cont.)

- Hi3 uses separate identifiers for location and endpoint
  - HIT - identifier for a public i3 trigger
- The pair of triggers for hosts A $[HIT_A|ID_A]$ and $[ID_A|IP_A]$
  - HIT_A and ID_A - public and private i3 identifiers of A
  - ID_A is constructed by A according to the constrained IDs technique
- Support of a higher naming layer
  - public triggers - association setup between a client and a server
  - the server creates a private identifier for the client
  - the private trigger is used to relay control messages
  - public identifiers - service identifiers in the layered architecture
  - private identifiers - ”lower” naming layer

The data plane. Protecting end-to-end traffic

- HIP for basic data protection
  - encapsulating all data traffic in ESP
  - protecting integrity, authenticity, and confidentiality
- HIP alone does not protect against distributed DoS attacks
  - the hosts reveal real IPs to the potential peers
- SPINATs protect against distributed DoS
  - placed on or close to the possible data paths
  - fast-path barrier
  - hiding the actual IP of the servers

The data plane. Protecting end-to-end traffic (cont.)

- Employing SPINATs
  - the client and server tell the address of SPINATs
  - the client and server do not tell their real IP
  - the use of SPINAT is controlled by the involved host
  - the SPINAT can act by inspecting HIP BE and HIP MU packets
- ESP envelopes and SPIs - DoS protection
  - middle box forwards and filters traffic based on (dst,SPI) pairs
  - the filtering can be extended to include source address
  - middleboxes can securely learn the appropriate mappings
  - the control and data packets take different paths - explicit signaling between policy points at the control and data path

The data plane. Protecting end-to-end traffic (cont.)

- The SPINAT knows the allocated SPI mappings
  - can filter out most unwanted traffic
- An attacker can only learn the IP of the SPINAT
  - getting packet through the SPINAT requires to the attacker to know a valid SPI
  - random packets are effectively filtered
- An attacker establishes OPP HIP association
  - the attacker learn a valid SPI
  - communication with a large number of zombies
  - spoofed traffic from zombies - a potential problem

The data plane. Protecting end-to-end traffic (cont.)

- Heuristics based on ESP sequence numbers
  - making coordinated attacks harder but not impossible
  - the zombies can increase the sequence number in rough synchrony
  - unwanted high-volume traffic where the sequence numbers mostly fall within the replay window
- Source address filtering everywhere in the network
  - preventing zombies to send valid-looking packets
  - the packet's source would necessarily be different
  - packet drop at the first SPINAT on the path

The data plain. Protecting end-to-end traffic (cont.)

- The IP source field is not needed in Hi3
  - the control packets are routed by the identifiers
  - the data packet destination is based on the local IP layer state
  - the data packet source is always ignored
  - the source address field - record the actual path taken by the packet
- A server under the attack can move the traffic to other SPINATs
- A server can tell the SPINAT to drop forwarding data on the attacked SPI
- The SPINATs are still vulnerable to DoS
  - a well designed SPINAT can handle the packets at the fast path
  - SPINATs can be placed at high capacity links

The data plane. Protecting end-to-end traffic (cont.)

- Protection works independently of i3
  - different paths can use different protection mechanisms
- The hosts do not need to learn the location of i3 servers
  - the selection of a suitable forwarder become easier
- The resulting i3 infrastructure is easier to protect than the original one
  - it need not be designed or provisioned to carry all traffic

The data plane. Supporting multiple IP realms

- Two requirements for system to work properly
  - all hosts must be reachable through the i3
  - the host must know at least one public IP of a SPINAT
- The requirements of knowing a public IP of a SPINAT
  - an anycast-based mechanism - learning suitable SPINATs
  - corporate environment - SPINAT related information could be naturally distributed
  - on-path, passive plain NATs could be detected directly (STUN or ICE method to create state)

The data plane. Supporting multiple IP realms (cont.)

- Making hosts reachable by the i3
  - locating the infrastructure in the public Internet
  - the hosts in other IP realms maintain active connectivity with the i3 server holding their private triggers
  - packet sent to the private triggers can be passed to the host over active connection
  - a host is able to create semi-permanent state with a public IP - a host can list the SPINAT's IP at the private trigger
  - the i3 server does not use an existing connection but send the packet to the SPINAT

The data plane. Supporting multiple IP realms (cont.)

- Multiple realm support requires reachability state
  - state should be created at the SPINAT between the realms
  - the state can be created explicitly
- Resulting infrastructure
  - resembles proactive hop-by-hop host routing
  - takes place on a layer above the current IP routing layer
- Assuming the existence of a single most preferred realm
  - SPINATs at the realm boundaries can learn the identifiers of the host behind them

Handling alternative encapsulation formats

- Support other encapsulation mechanisms in addition to ESP
  - the HIP architecture is planned to be extended
- HIP requirement
  - enough of information in the data packets
  - information can be successfully de-multiplexed and tagged with source and destination HITs
  - de-multiplexing must work independently of the source address
- SPINAT-based protection - future Internet HIP encapsulation method
  - the information for de-multiplexing is hard to predict
  - the information is easily verifiable with a state

Delegating part of the processing to the infrastructure

- Pointing to an infrastructure node
  - creating a trigger for the service identifier
  - a trigger does not point to a server that directly implements the service
  - an infrastructure node handles the I1 directly
  - the hosts handle I1s by replaying with pre-computed R1s
- Implementation of service/host separation
  - the R1 use a HI different from the service identity
  - explicit denoting of delegation from the service identifier to the host identifier

Delegating part of the processing to the infrastructure

- A separate *service distribution function, SDF*
  - can be integrated with the actual service nodes
  - handling I1 sent to the service
  - handling by replaying with suitable pre-computed R1
  - the R1s are not signed by a host key but by the service key
- Carrying two keys
  - the service key corresponds the HIT
  - the host key is stored in a HIP parameter - denoting delegation
  - the host key identifiers the host that provides the service to the client

Delegating part of the processing to the infrastructure (cont.)

- I1 is completely processed by infrastructure
  - the SDF allowing
  - the service need not see the I1 in the first place
- Separating function of forming R1 and replaying to I1
  - the R1 can be created and signed by a well protected node
  - the R1 distribution can take place directly by the i3 server
  - requirement - supplying the i3 server with pre-computed R1
- R1 *formation* delegating
  - the expected semantics modification of the HIP puzzle

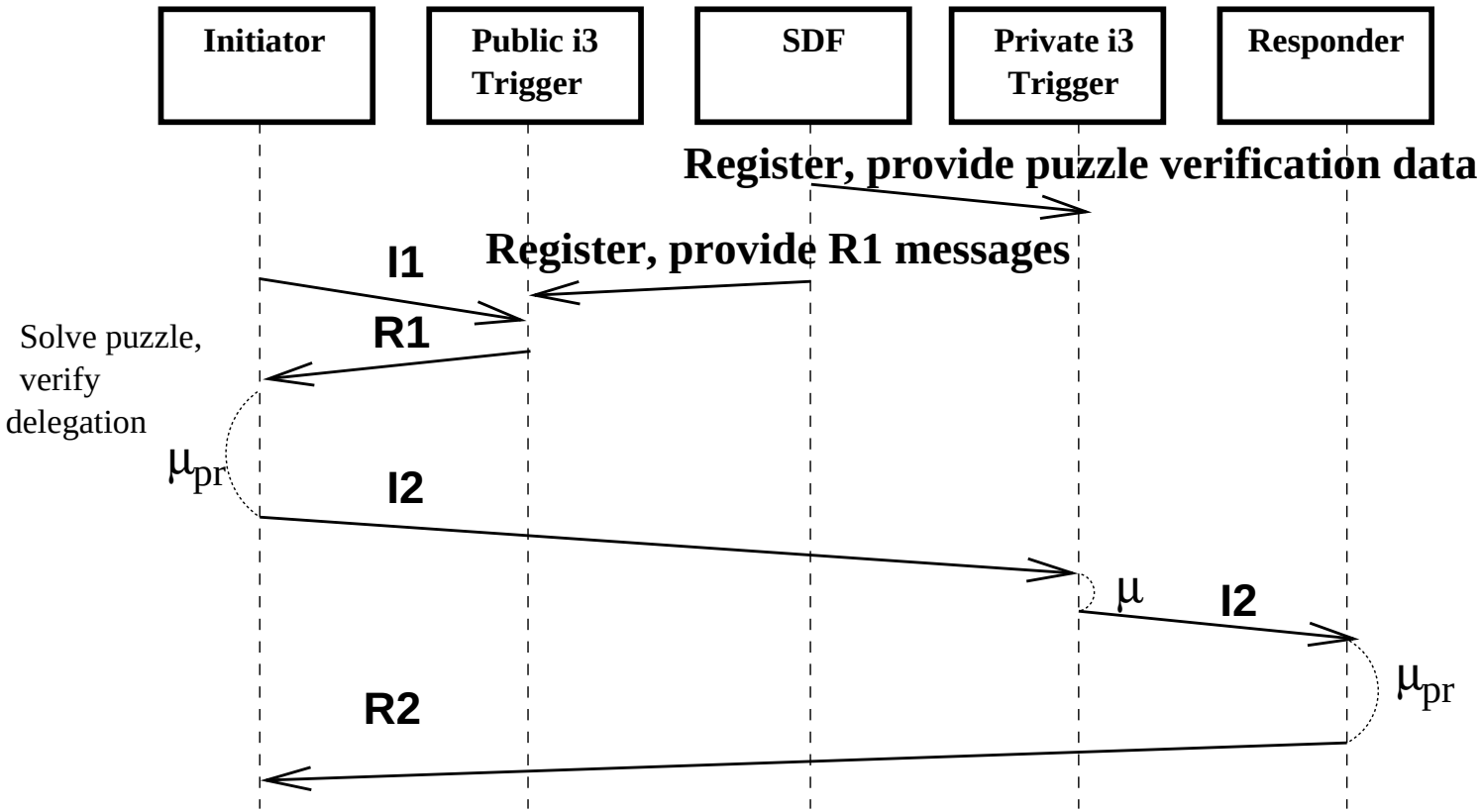
Delegating part of the processing to the infrastructure (cont.)

- A way of creating a puzzle in HIP
  - the server can verify that a given puzzle has indeed been created by it
  - the Responder rejects correctly solved puzzles not creating by it
  - protection from pre-computation attacks
- Delegating puzzle creation to another function
  - the ability to verify the puzzle origin is lost
- Enhanced processing model
  - the infrastructure must verify the puzzle origin
  - the node generating the R1 can provide the i3 server with the necessary information for verifying the puzzle's origin and freshness

Delegating part of the processing to the infrastructure (cont.)

- The server assurance of puzzle verification - different ways
- The simplest way
  - the server assumption - any I2 coming from the i3 server have passed puzzle verification
  - the server fully trusts the infrastructure and there is a secure channel - the way can be deemed secure
- Second solution
  - the SDF and the actual servers share data about puzzle creation
  - the server is allowed to (re)verify the puzzle

Hi3 base exchange with delegation



Mobility

- Basic mobility between already communicating hosts
  - can be provided directly at the HIP and SPINAT level
  - the i3 infrastructure is not involved
- The hosts lose direct reachability
  - the hosts need to revert back to the infrastructure
  - use of private triggers
  - the host must keep the infrastructure updated with their location
- Combining the end-to-end mobility and the indirect mobility
  - highly efficient (no triangle routing) and robust mechanism
- HIP for network mobility

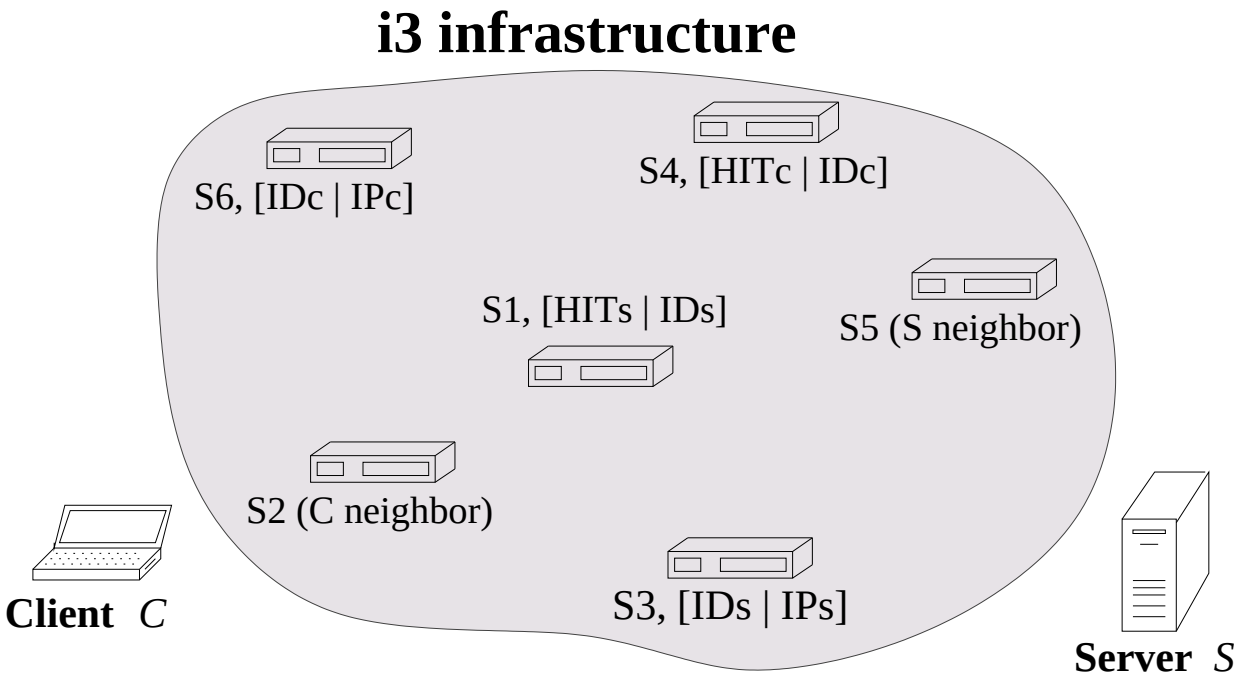
Signaling multicast and anycast

- Hi3 allows the HIP control messages to be replicated and distributed to multiple receivers
  - any immediate benefit can not be imagined
  - a HIP-based group key distribution protocol - most probably benefit from a control plane multicast facility
- The SDF forms a rudimentary control-layer anycast service
  - a service is allowed to be identified by a single identifier
  - the actual service is provided by multiple nodes
  - the i3 node is provided with IP-layer topology information
- The i3 node is provided with IP-layer topology information
  - creation of a service that is provided by a node close to the client

The control plane

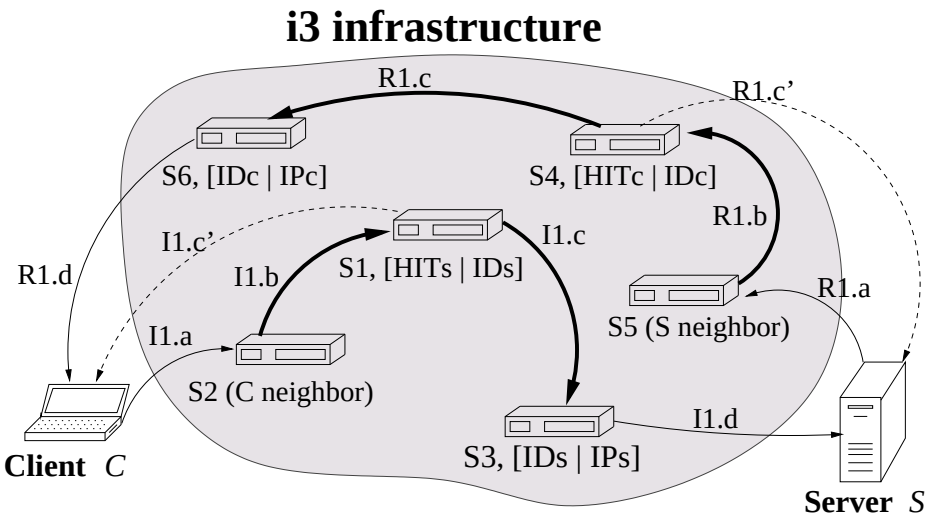
- Relaying HIP messages in two cases
- Two end-hosts establish a HIP association
  - before direct end-to-end communication
  - benefit of using the control plane - DoS protection
  - the IPs are not revealed until authentication is completed
- The hosts lose the direct end-to-end connectivity
  - the case is important for end-host mobility
  - e.g., the connectivity is lost after a simultaneous movement of hosts
- The control plane - a trusted third-party
  - establishing and keeping the data plane connectivity between peers

The control plane (cont.)



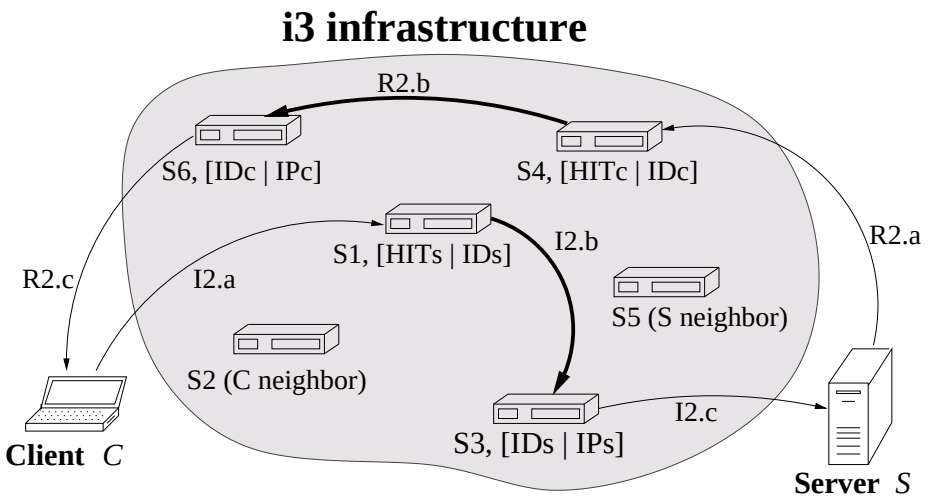
- C and S - communication end-hosts
- C is a HIP Initiator, S is a HIP Responder
- a distribution of HIT-based public and private trigger in i3
- first contacts - the peers use neighbor i3 nodes that they happen to know

Pure HIP association setup

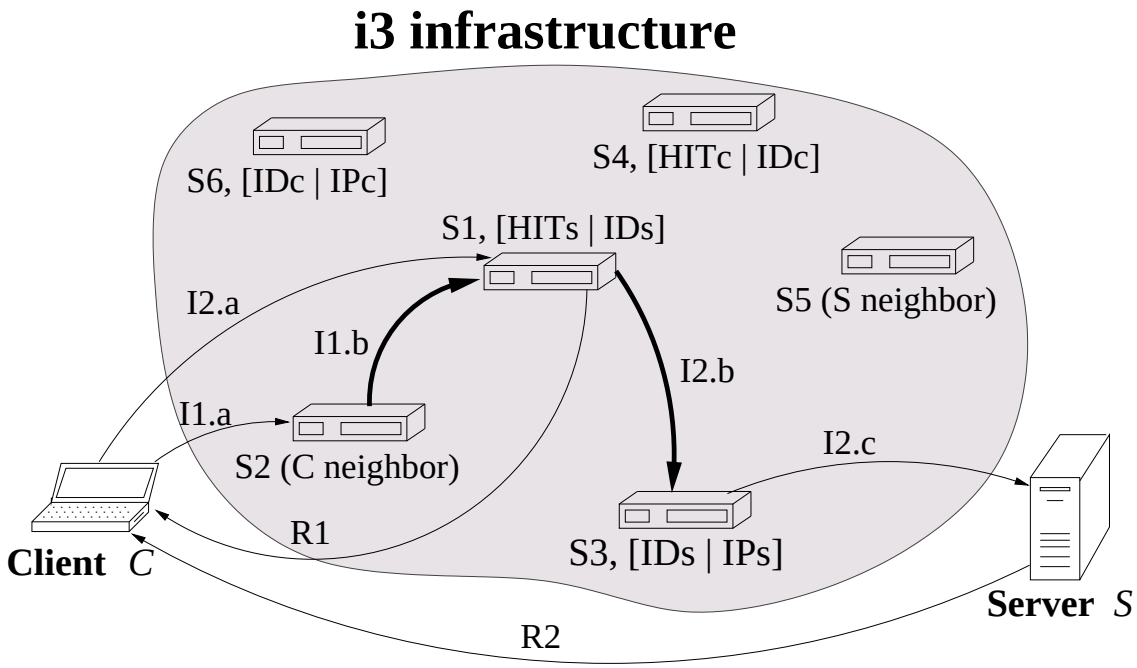


- pure association setup: I1 and R1 packets
- HITs is stored in S1
- the private trigger of S is on S3
- HITc is stored in the S4
- the private trigger of C is on S6

- pure association setup: I2 and R2 packets
- the packet sent straight to the i3 nodes keeping the public triggers

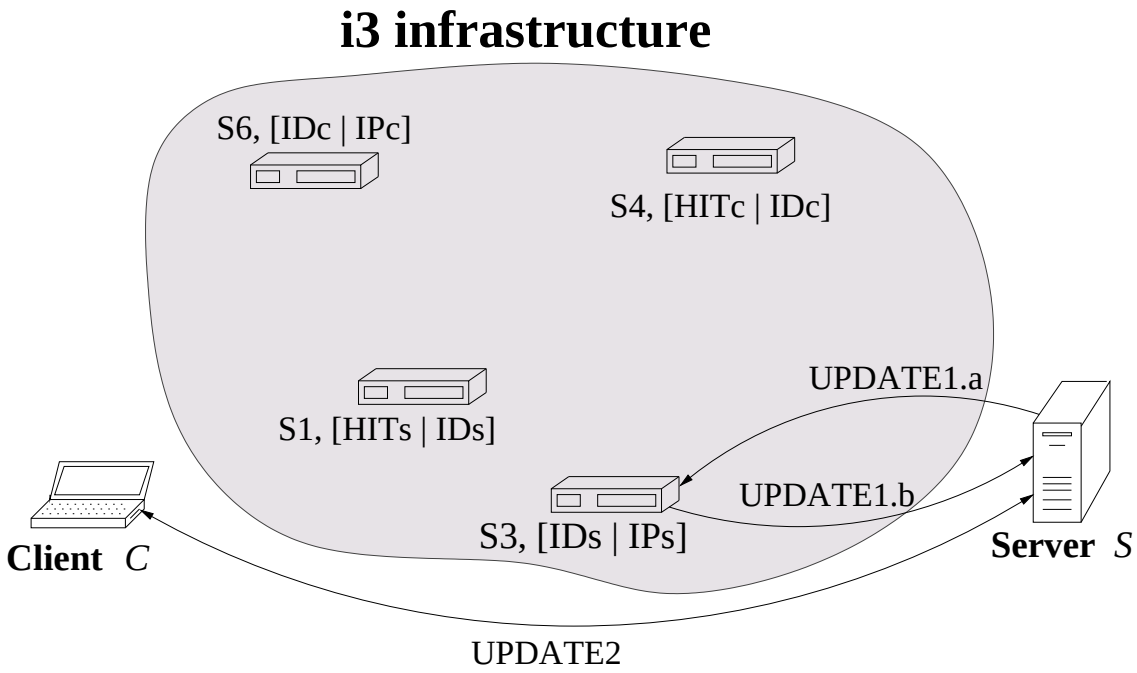


Optimized HIP association setup



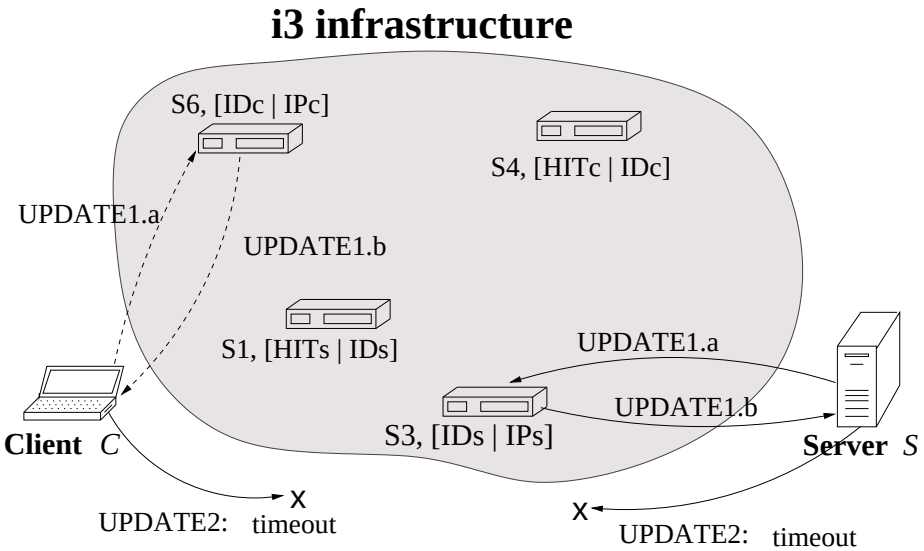
- an initial part of the setup is delegated from S to i3
- the S1 cashes pre-computed R1
- the control plane is not involved in R2 delivery
- reduction of the load of S

Location update



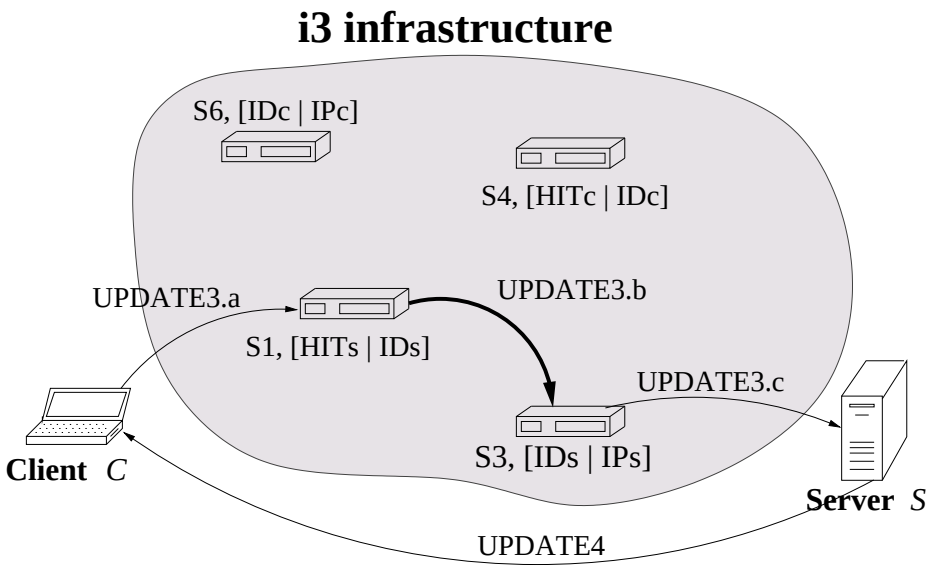
- both C and S can change their location
- the change is due to the S - S updates its private trigger in i3
- HIP update exchange runs over data plane
- UPDATE1 and UPDATE2 can be sent in parallel
- no significant overhead is introduced

Simultaneous host movement



- C and S change their addresses simultaneously
- each host uses out-of-date IP to contact the peer
- the failure can be discovered by a timeout

- recovering form the double-jump
- the C is responsible for starting the recovery



HIT insertion

- HIT_A - a HIT of A, IP_A - recent IP of A
- Two reasons for A to insert HIT_A into i3
  - A - a server, HIT_A has to be in i3 permanently
  - A - an Initiator of pure association setup, HIT_A is used by the Responder
- Inserting HIT_A
  - A constructs a private i3 identifier ID_A
  - A sends two requests to i3: public trigger $(HIT_A|ID_A)$ and private trigger $(ID_A|IP_A)$ are inserted
  - each request requires a Chord lookup to assign an i3 node
  - the acknowledgment with a recent node's IP

HIT refreshment and HIT re-insertion

- At trigger's Time To Live is limited
- Keeping HIT_A alive in i3
  - A should refresh regularly public and private triggers
  - the refreshment are sent directly to the corresponding i3 nodes
- The i3 crashes - the trigger is lost
- Re-inserting the trigger
  - insertion or refreshment request to any i3 node
- The acknowledgment is received in a successful case

HIT removal

- HIT removal happens automatically
  - private or public trigger expires
  - two requests are explicitly sent
- It can be important for a host to remove HIT as soon as possible when an attack via the triggers has been discovered

The case with separate naming for hosts and service

- The requests were designed without taking separating naming into account
- The corresponding modification is clear
  - a public identifier (HIT) is used only for a first contact with the server
  - a client is trusted - it uses a given server's private trigger

Discussion of the Hi3 design

- The gap between HIP and application layer can be filled
  - the i3 allows using two name layers
  - private triggers - a host namespace like in HIP
  - public triggers - a service namespace
- The control plane performance increases
  - clients talk directly via the node with private trigger
- The security can suffer
  - the rule of compulsory usage of a public/private trigger pair is violated
  - extra security mechanisms should be applied

Discussion of the Hi3 design (cont.)

- Any successful request - acknowledgment to the end-host
- Some request can fail
  - packet losses in a network
  - i3 inconsistency and unavailability
  - the problem is not very significant - i3 is based on top of Chord DHT, which is resilient to massive failure of nodes
- The advantages of i3
  - better DoS protection
  - support for simultaneous mobility
  - higher fault-tolerance when using a DHT with data protection

Discussion of the Hi3 design (cont.)

- The disadvantages of i3
  - reliance on an extensive infrastructure
  - server scalability
  - use of UDP
  - lack of traffic encryption
  - complexity of i3 as an overlay network
- Limited experience with widespread i3 deployment
  - difficult to assess the scalability of servers

Discussion of the Hi3 design (cont.)

- Relaying all control and data traffic through the i3 would prove burdensome
  - agreement of using i3 only for initial contact
- The combination approach of Hi3 helps to address shortcomings of HIP and i3
- Using i3 as a control plane for HIP
  - protection from DoS attacks
  - solving the double-jump problem
  - providing an initial rendezvous service

Discussion of the Hi3 design (cont.)

- Hi3 provides additional protection against DoS attacks
  - hiding parties' IP until the HIP handshake partially authenticates them
- Simultaneous mobility of both hosts in i3
  - sending update control packets via i3 when end-to-end connectivity is lost
- Hi3 inherits the challenges of the extensive i3
  - trust
  - accountability
  - cost issues

Micromobility

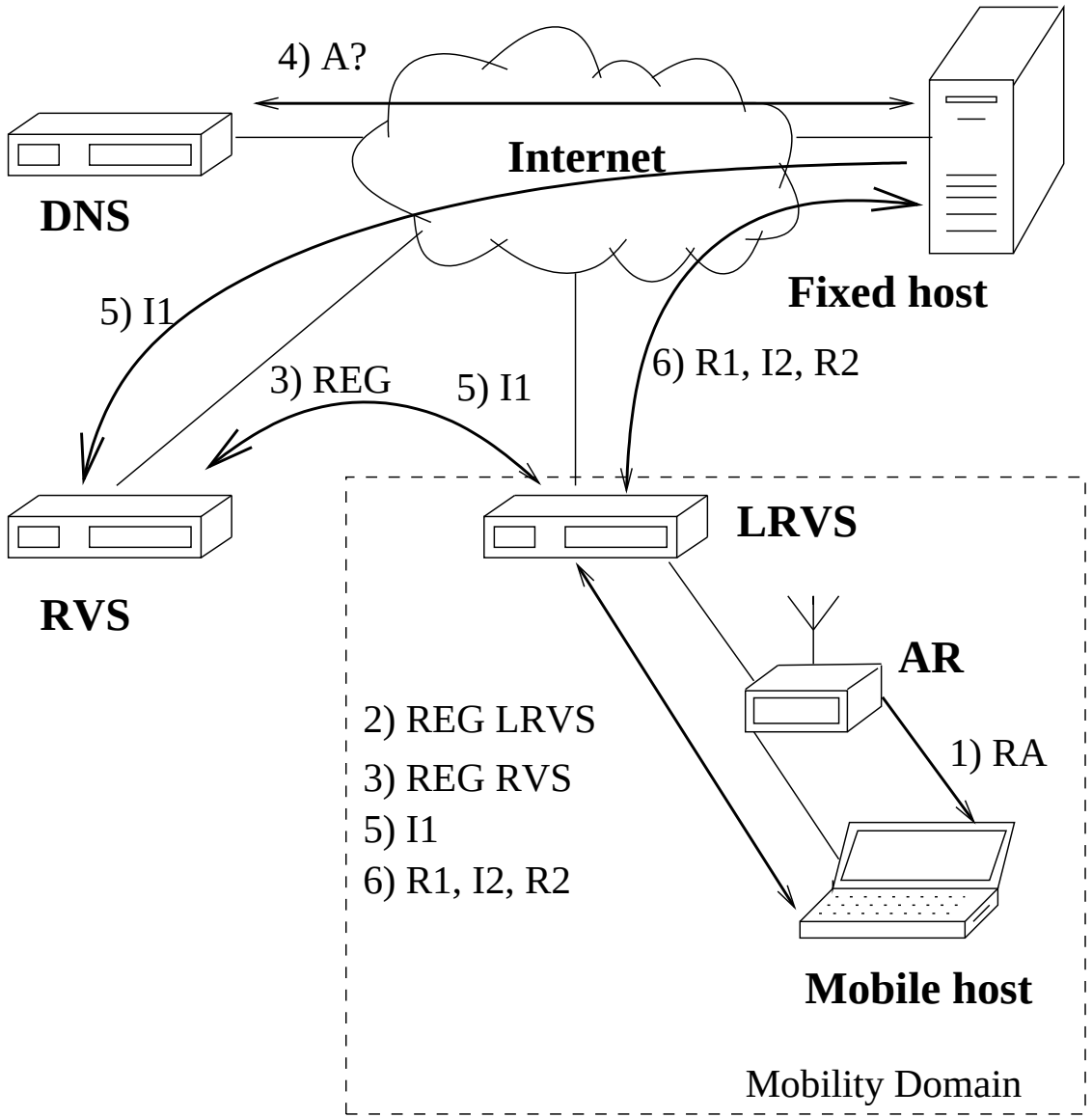
Micromobility

- Handling host mobility with a certain network area locally
- Implementation
  - a stable anchor point represents the network area
  - a host moves within the area - only the local access point need to be updated
  - a host moves out of the area - the new anchor point is selected
- Micromobility proposals include Hierarchical Mobile IP

Local rendezvous servers

- Micromobility architecture for HIP based on local RVS (LRVS)
- Each LRVS is responsible
  - for host mobility in its own micromobility domain
- Hosts in a domain
  - use private IP addresses
  - can access other host outside of its domain
- The LRVS acts as a NAT by translating local private address to its external address

HIP micromobility architecture



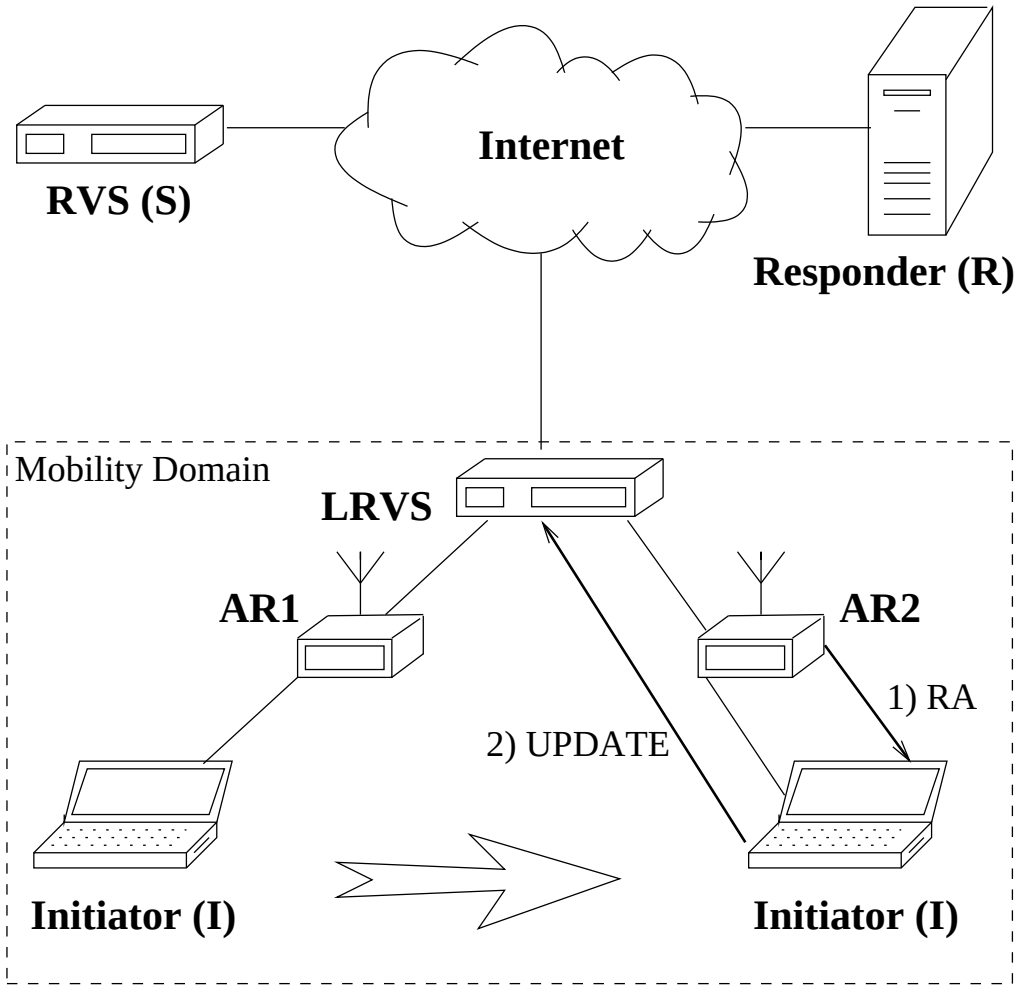
Intra-domain mobility

- A host obtains a private IP after joining the domain
  - DHCP configuration (IPv4)
  - address autoconfiguration (IPv6)
- The IP and HIT of the domain LRVS are obtained from an AR
  - information is broadcast by the AR as ICMPv6 Routing Advertising
  - the host can accelerate the configuration process by ICMPv6 Router Solicitation
- The host registers to the LRVS
  - using the normal HIP registration protocol
  - LRVS creates an entry in its mapping database (from the host local IP to the RVS external public IP)

Intra-domain mobility (cont.)

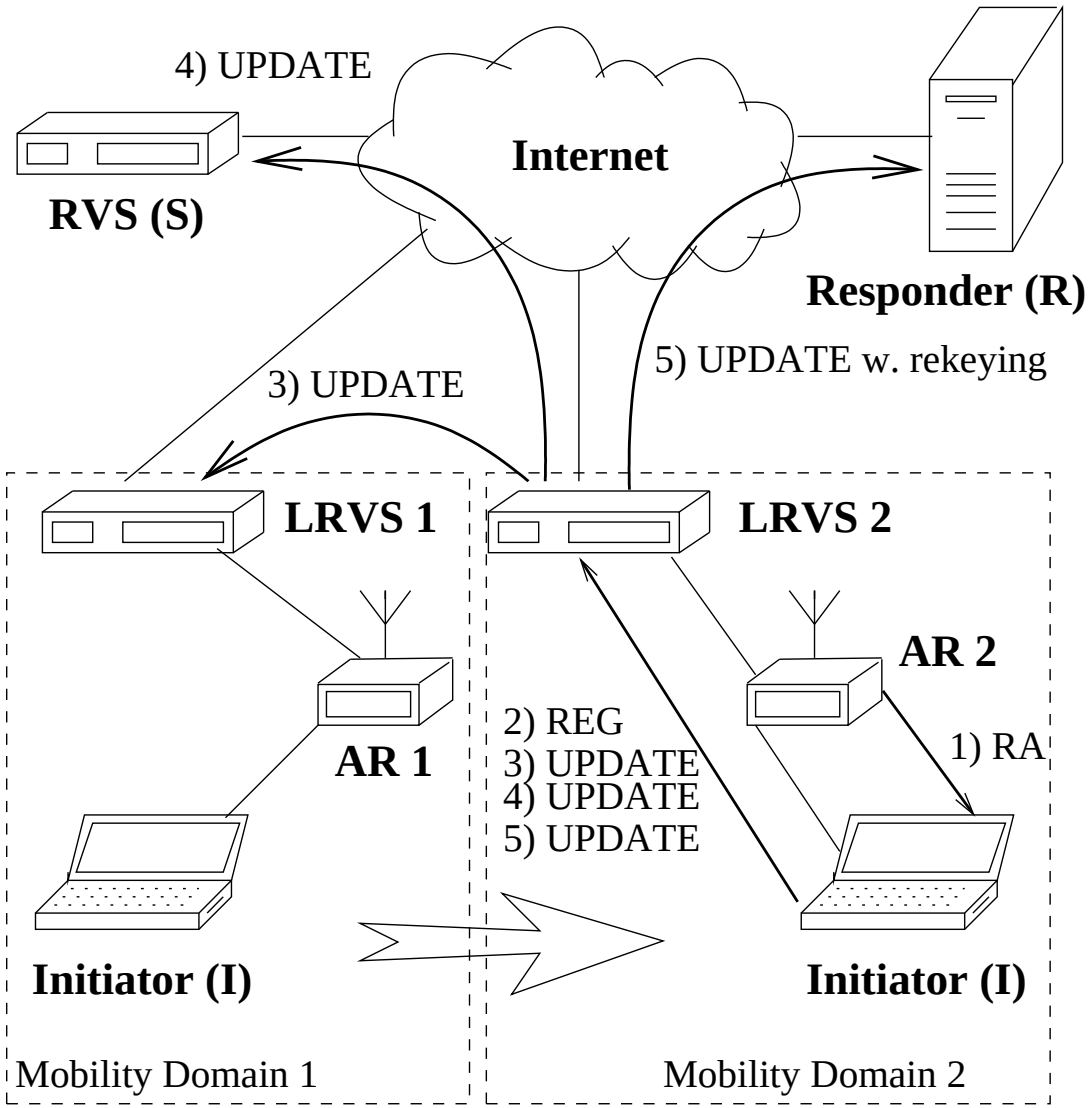
- The host registers to the RVS
  - ensuring that it is reachable by other hosts
  - the host sends its HIT and the public IP of the LRVS to the RVS through the LRVS
  - the source address is changed from the private host address to the public address
  - the mobile node is registered to LRVS and RVS
- A fixed host would like to communicate to the mobile node
  - a DNS query - finding the IP of the RVS of the mobile node
  - a DNS has replied - the fixed host sends an I1 to the RVS

Intra-domain mobility update



- the host performs an intra-domain handover to a new Access Router
- moving to a new location - the host receives advertisement messages from new AR
- the host learns that the handover is intra-domain
- the host sends its new IP to the LRVS
- mobility within the domain is completely handled by LRVS

Inter-domain mobility update



Inter-domain mobility

- The inter-domain handover is more complex
  - the host knows the domain changing by receiving a Router Advertisement
  - the host registers to the new LRVS2
  - the host informs its old LRVS1 of its new location
  - the old LRVS1 forwards packets to the mobile host through the new LRVS2
  - the hots updates its registration at RVS
  - correspondent hosts need to be updated
  - the host performs a rekeying update when moving to a new network
  - all updates are completed - the mobile host can close its registrations with the LRVS1

Secure micromobility

- A mobile host registers to an LRVS to authenticate itself
- The registration procedure is heavyweight
  - challenge for mobile hosts changing domain often
- Presenting an approach for securing micromobility
  - more lightweight
  - better suited for hierarchical mobility domains
  - no requirement of registration to the mobility anchor points

Hash chain authentication

- The trust between the anchor point and the mobile host
  - the trust between the mobile host and its correspondent host during macromobility exchange
  - the anchor point verifies that the IP belongs to the mobile host - prevention from hijacking of address and DoS attacks
- The solution is based on
  - Lamport one-way hash chains
  - secret splitting techniques

Hash chain authentication (cont.)

- A mobile host changes location - a next value of the hash chain to the peer
  - messages are protected by HMAC codes
- One or more of mobile anchor points are located on the path
  - observe the mobility exchange between two hosts
- The anchor point can authenticate the message sent by the mobile host

Hash chain authentication (cont.)

- The anchor point does not care about the ownership of the key chain
  - hash chain values remain correct within one communication context
- Using host identifiers as authenticator with hash chains
  - hijacking the identifier
- Avoiding of an identifier spoofing
  - the identifier is concatenated with a random number

Hash chain authentication (cont.)

- The protocol implementing delayed authentication consist of three messages
- The first message
  - from mobile host to the peer through the anchor points
  - $ID_m, Enc(H_{i+2}), H_i, HMAC(H_{i+1}, ID_m || Enc(H_{i+2}))$
- The correspondent host replies with a message
  - $ID_c, H_{i+1}, HMAC(H_{i+2}, ID_c)$
  - the mobile host should wait for this message before transmitting the next hash chain value

Hash chain authentication (cont.)

- The final message
  - the mobile host identifier, the hash chain value of two steps ahead in a plain text
  - ID_m, H_{i+2}
- Bootstrapping the secure communication between two hosts
  - a HIP host creates a hash chain containing n hashes
  - each micromobility exchange consumes three hash values
- Almost all values are used
  - the host must create the hash chain
  - linking the old and new hash chain

Hash chain authentication (cont.)

- Linking the old and new hash chain
  - the old starting value is replaced with the new one
  - the HMAC securely binds the new hash starting value to the old one
  - the next mobility update - the second message:
    $H_0^{new}, H_i^{old}, HMAC(H_{i+1}^{old}, H_0^{new})H_{i+1}^{old}$
- The middlebox verify that the new hash value is correctly linked to the existing hash chain

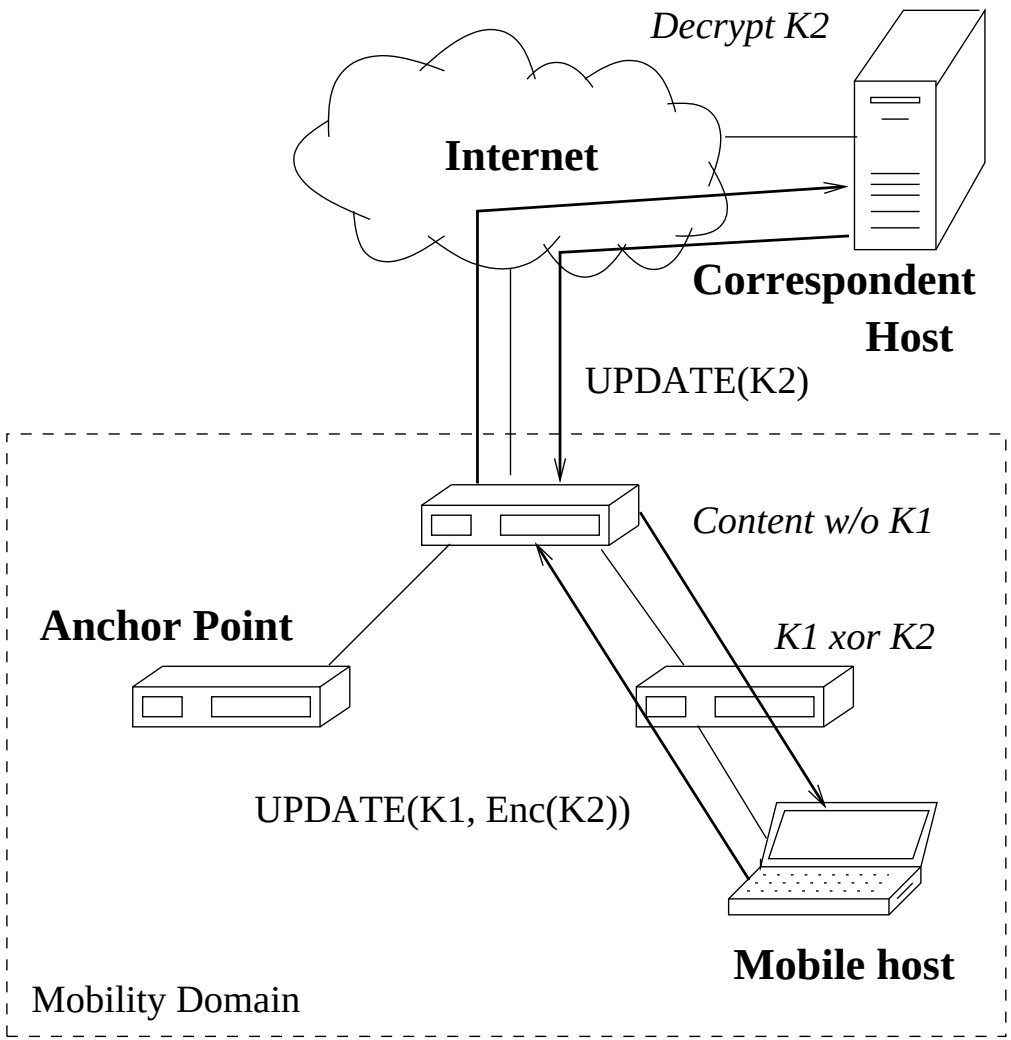
Secure network attachment

- The micromobility management protocol is an extension of HIP macromobility messages
- The UPDATE is augmented to include the hash values and parts of the session key
- Attaching at a new network
  - macromobility exchange with peers
- Middleboxes on the path learn the hash values
  - can later authenticate messages belonging to the same HIP host
  - the host can move within the local mobility area transparently to the peer
  - middleboxes handle the necessary signaling locally

Secure network attachment (cont.)

- The hash chain is used by middleboxes on the path
  - determination that update messages belong to the same host
- A key splitting technique
  - sharing a secret between a HIP host and the middlebox nearest to it
  - the secret key is split into two parts, K1 and K2
  - a mobile host sends the K1 in plain text in the first UPDATE
  - the K2 is sent in the same packet but encrypted with a session key

Secure attachment to a mobility domain

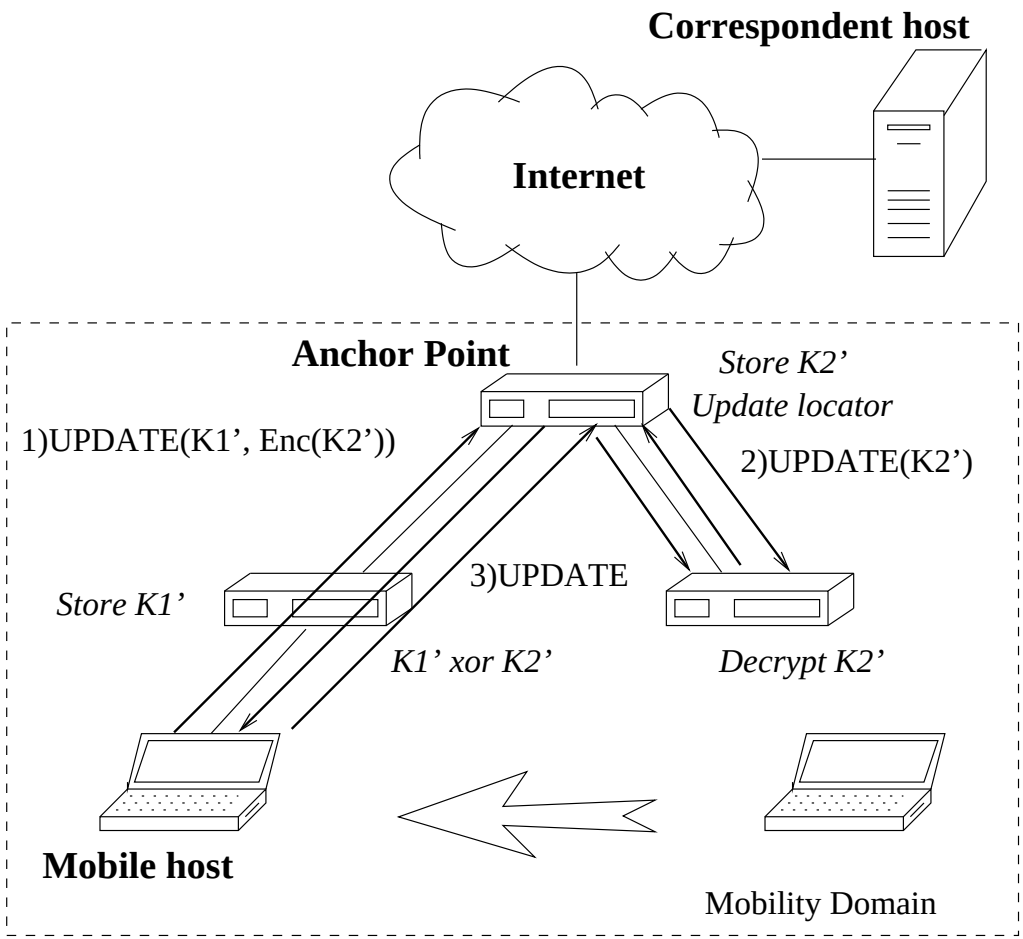


- the process of key splitting
- the closest middlebox receives the UPDATE - stores the K1, zeros the K1 in the packet, forwards the packet further on
- the correspondent host decrypts the K2 and sends back in reply UPDATE
- the middlebox obtains the K2 and verifies the whole key (whole key - XOR operation between K1 and K2)

Secure network attachment(cont.)

- Attacking the security model
  - an adversary can position itself between the mobile host and the adjacent middlebox
- Prevention the attack
  - adjacent middlebox checks if the UPDATE from the mobile host contains a zero K1
  - in the case of an attacker the middlebox zeros the K2 and forwards the UPDATE to the mobile host
  - the attacker is not able to establish a secret key with the mobile host
  - the mobility update falls back to macromobility

Micromobility handover



- secure intra-domain handover
- middleboxes on the path learn the hash chain
- the middlebox adjacent to the mobile host shares a secret key with it
- the mobile host moves to a new anchor point - a new split key, K1 is sent
- the adjacent middlebox stores K1 and forwards the UPDATE
- the next middlebox can verify the continuity of the hash value
- the root middlebox forwards the packet further to the previous anchor point and replaces the source IP with its own
- the second and third UPDATE travel through the root middlebox

Micromobility handover (cont.)

- The old adjacent middlebox
  - can decrypt the new K2 with a secret key
- K2 is sent in a plain text back to the mobile host
  - sending via the new adjacent middlebox
- The update process is completed
  - the old adjacent middlebox deletes the state associated with the mobile host
  - the new adjacent middlebox - the only entity sharing a new secret key with the mobile host

Micromobility handover (cont.)

- The prototype of the presented micromobility solution was implemented with FreeBSD HIP
  - the evaluation of the performance by simulating network latencies to distant web servers
  - the experiment with a distance equivalent to a server located in the USA
  - the micromobility update is 2.6 times shorter than the micromobility handover

Network mobility

- Extensions of HIP for network mobility
- WLAN located within a moving train - a typical example of a mobile network
  - the users are attached to the same router within the train
  - the router can change its attachment point to the Internet as the train moves between stations

Delegation of signaling

- Hosts can delegate signaling rights
  - reduction of the cost of signaling over an expensive last-hop link without weakening security
- Authorizing a mobile router to send updates
  - sending within a certain network area
  - a certificate is created $\{K^{+}_{MN}, K^{+}_{MR}, region = 10.0.0.0./20\}_{K^{-}_{MN}}$
  - the certificate is signed with the private key of mobile host
- The certificate in case of a multihomed mobile host
  - a specific link where the updates can be sent is included
  - prevention of a mobile router from claiming that the mobile host is unreachable

Delegation of signaling (cont.)

- A specific application of signaling delegation was proposed to enhance security in network mobility
  - a certificate of the form $\langle Issuer, Subject, Delegation, Tag, Lifetime \rangle$ is created
  - authorizing the mobile router to send mobility updates on its behalf
  - further delegation of rights from the mobile router to the next router
- Handover is completed
  - the mobile router can send a single message to a correspondent host
  - updating all security associations
- Reduction of the amount of signaling
  - multiple mobile hosts can be connected to the same correspondent host

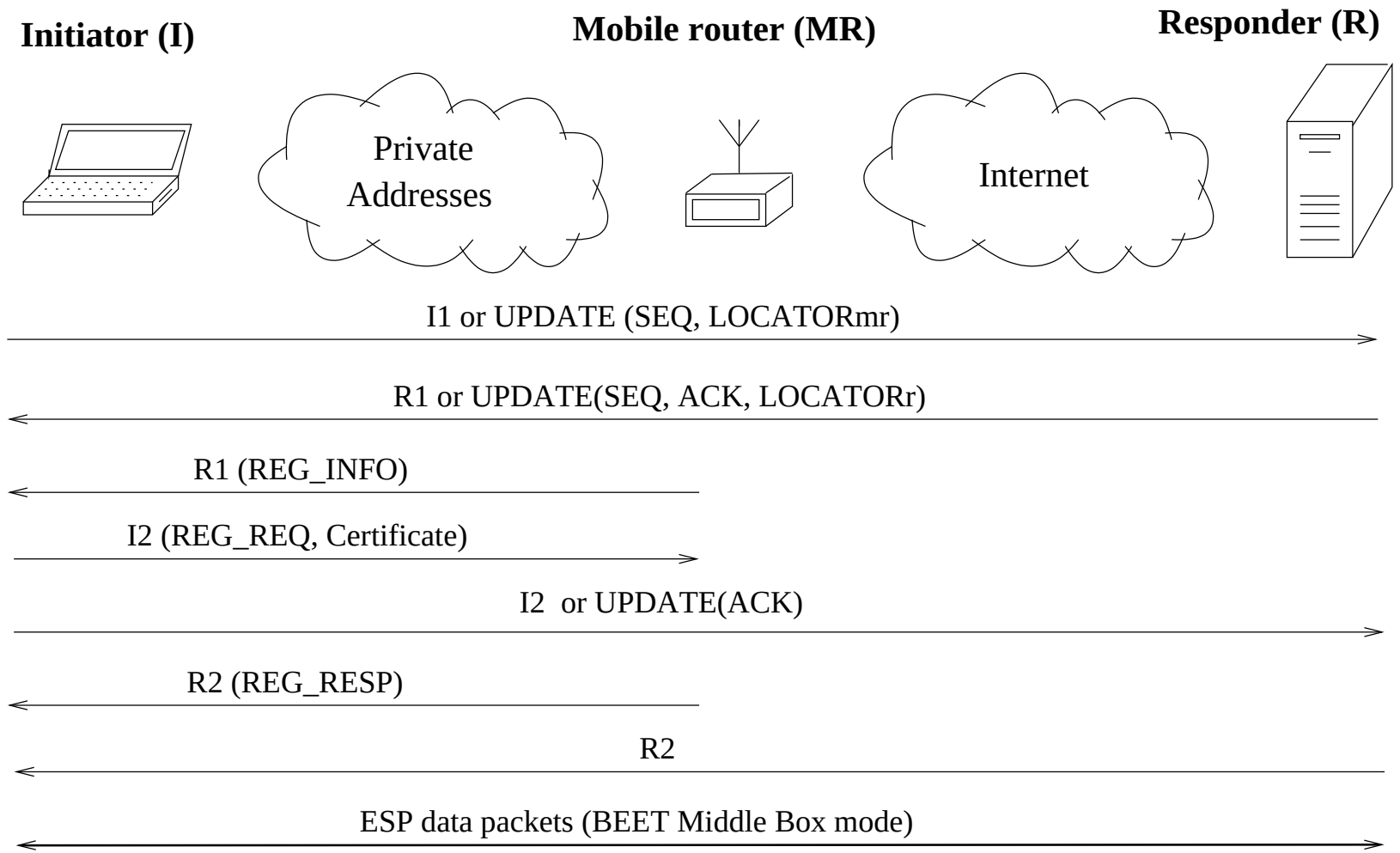
Mobile router

- An Ericsson implementation of the HIP mobile router
  - supporting nested mobile networks, where a separate subnetwork can change the location within a parent mobile network
  - integration with service discovery
  - integration with simultaneous multiaccess extensions for HIP
- Mobile hosts are located in the network with a private address space
- Delegating the signaling rights to the mobile router
  - the certificate contains mobile host's HIT and mobile router's HIT

Mobile router (cont.)

- Mobile router re-attaches to a different location
  - location updates on behalf of mobile nodes
  - the mobile nodes move together with the mobile router
  - the mobile nodes use the same private address independently of their location
- The network mobility process is transparent to mobile hosts
  - the mobile nodes use the same private address independently of their location

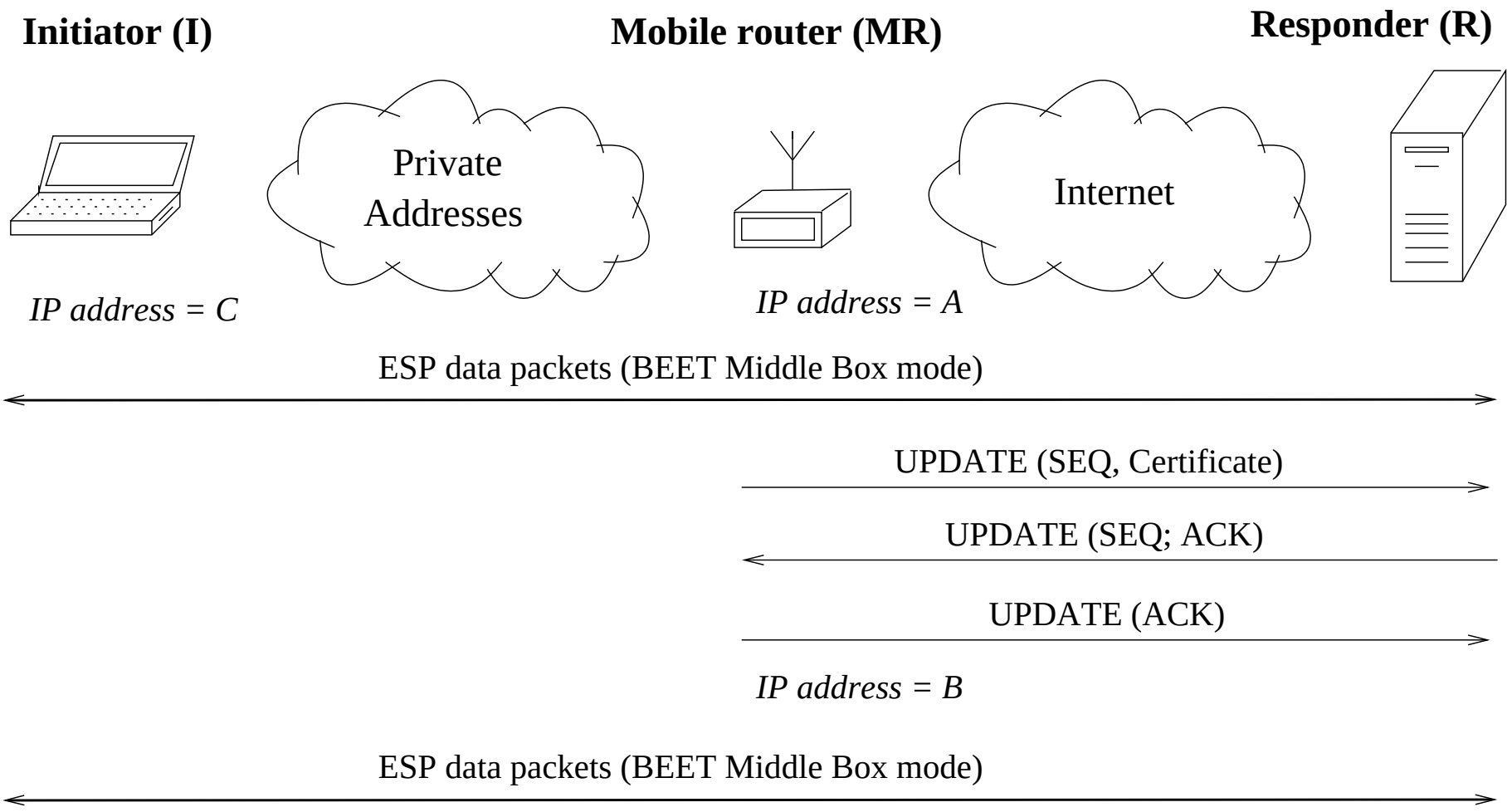
Base exchange and registration with HIP mobile router



Mobile router (cont.)

- New mobile node attaches to the mobile router
  - the registration procedure
  - updating the correspondent host of a new location
  - the I2 contains a certificate
- A BE via the mobile router
  - UPDATE message with the public IP address of the mobile router
  - address translation and packet forwarding by the mobile router
  - the BE or MU are completed - ESP Bound End-to-End Tunnel
  - standard BEET ESP mode is modified - BEET middlebox mode

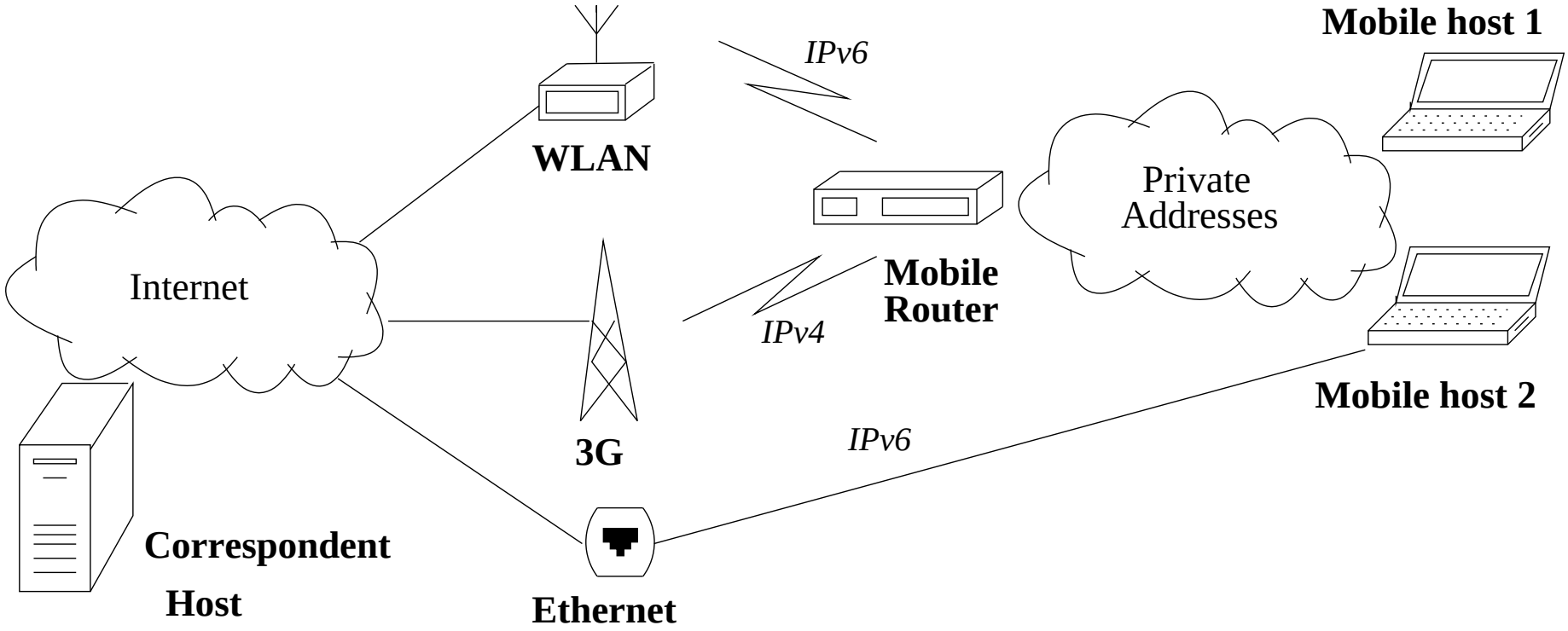
Mobile router performs an update on behalf of a HIP host



Mobile router performs an update on behalf of a HIP host (cont.)

- The mobile router attaches to a new place
  - UPDATE messages to all correspondent hosts
  - the certificate signed by the mobile host proves authorization of the router
  - the update is completed - all data packets from the correspondent host are sent to the new address of the mobile router
  - private IPs of the mobile hosts do not change

A demonstration of the mobile router and SIMA



A demonstration of the mobile router and SIMA (cont.)

- The demonstration is set up by Ericsson NomadicLab
  - the router connects to the Internet via 3G (IPv4) or WLAN (IPv6)
  - links can be active one at a time or simultaneously
  - mobile host 2 in addition has a wireline Ethernet
- The hosts support simultaneous multiaccess extensions
  - parallel data transmission over two or more available links
- A video clip is streamed from the correspondent to mobile hosts
  - the 3G or WLAN is disconnected from the router - the traffic is automatically redirected to use the other available link

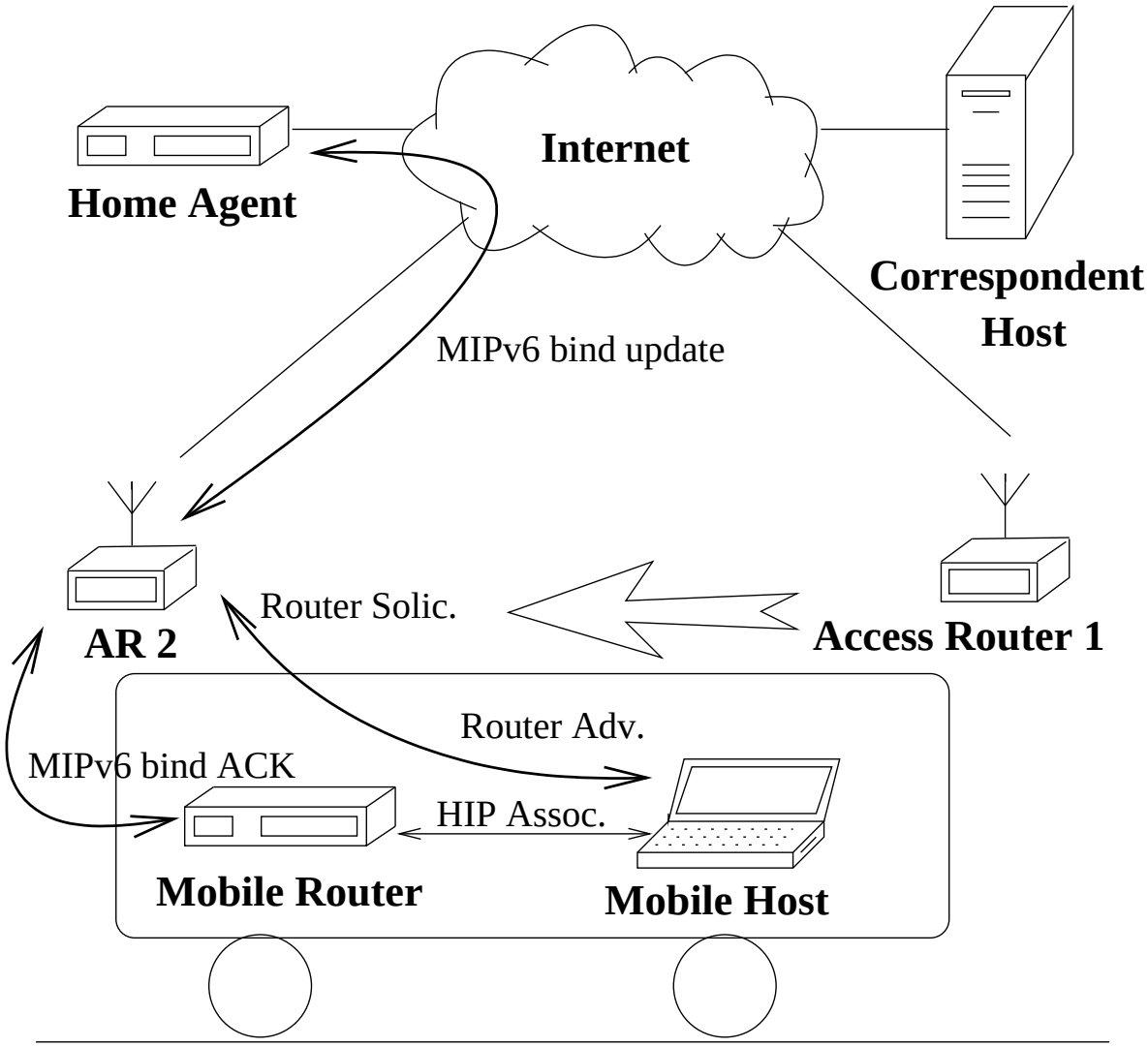
HarMoNy

- An alternative proposal for HIP network mobility - HIP and NEtwork MObility
  - focusing on a hybrid mobility solution
  - combination of Mobile IPv6 routers implementing Network Mobility (NEMO) extensions and HIP mobility
- Context-aware handover between a NEMO router and public Internet connectivity
  - the NEMO router provides a private IPv6
  - the public connectivity provides a topologically correct IP
  - the mobile host can use some clues to select between NEMO or direct Internet connectivity

HarMoNy (cont.)

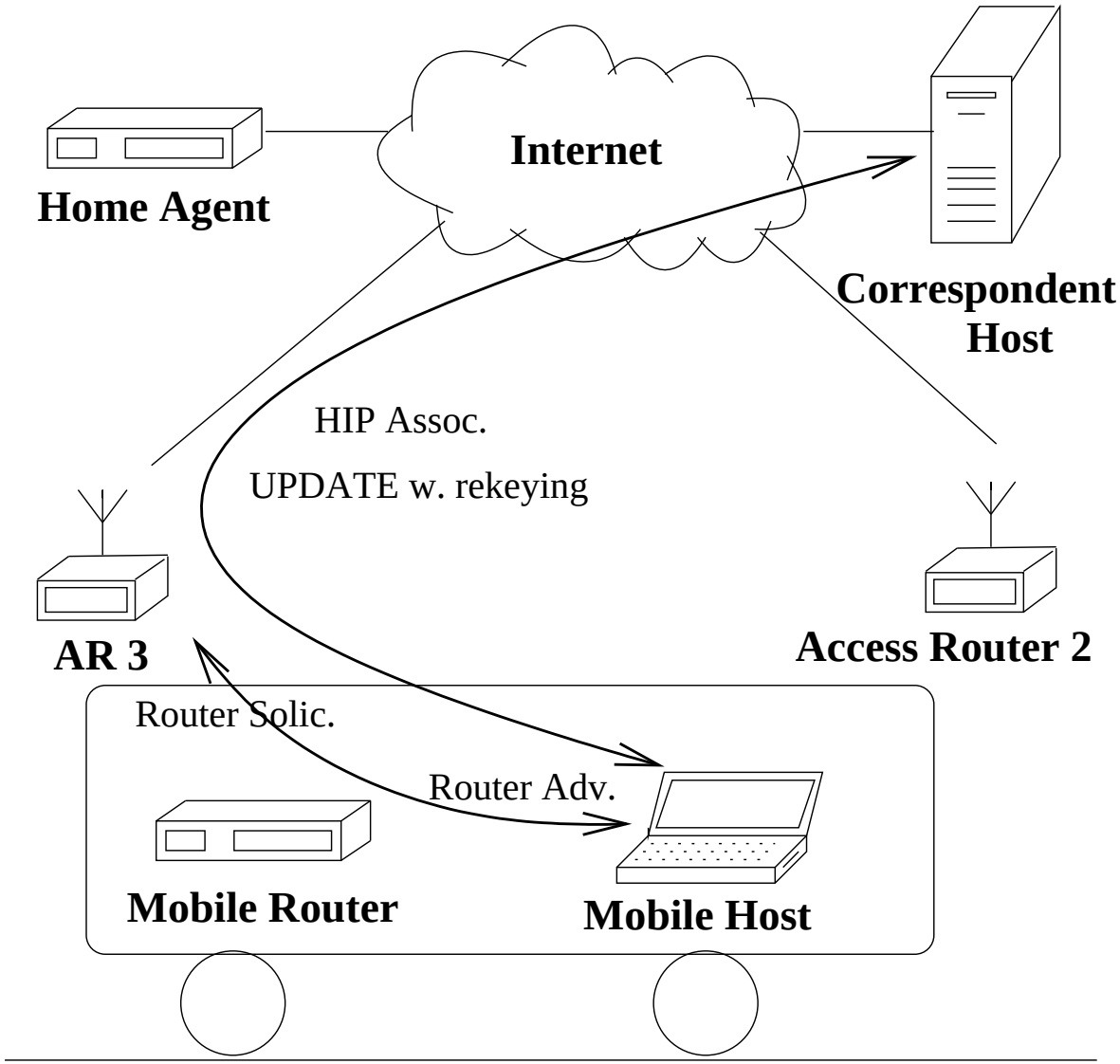
- The advantage of higher-level mobility solutions such as HIP
  - function without special infrastructure
  - function without integration with IPsec
  - function without multihoming support
- Lower-level mobility with NEMO
  - infrastructure components such as home agent are required
  - support from the local router is required
  - more efficient and require less involvement from end hosts
- HarMoNy enables simultaneous use of several mobility solutions

A HarMoNy mobile network when moving



- design of HarMoNy is suitable for a moving vehicular scenario
- the vehicle is moving fast and the rate of handovers is high - a mobile host allows the mobile router to handle handovers using NEMO with MIPv6
- the vehicle is slowing down - in addition to mobile router connectivity the mobile hosts obtains a care-of address

A HarMoNy mobile network when stationary



- the vehicle is stationary - the mobile host establishes a HIP association directly to the access router

HarMoNy (cont.)

- The number of mobile hosts in the vehicle should be taken into account
- A prototype implementation of HarMoNy
  - combines an OptiNets mobile router with HIPL
  - advertisement messages - the presence of mobile and access routers
- Selection of the address to use by the mobile host
  - dependence of the vehicle speed
- For NEMO, a MIPv6 rebind packet must reach the home agent and be acknowledgment

HarMoNy (cont.)

- The handover delay
  - the RTT between the mobile router and home agent
  - the RTT between the mobile router and correspondent host
- Handover latency of HIP was compared with NEMO MIPv6
  - without rekeying - HIP latency is half that of NEMO
  - with rekeying - HIP latency is double that of NEMO
- Packet losses of $0 - 10\%$ during handovers
- The conclusion of study
  - the use of NEMO is preferable with rekeying
  - the security properties are weaker

Communication privacy

SPINAT

- IPsec packets cannot directly traverse legacy NAT devices
  - the Authentication Header or ESP prevents packet modification
  - ESP - no visible port information for the NAT
- Encapsulation into UDP datagrams
  - traversing legacy NATs
- Security Parameter Index multiplex NAT (SPINAT)
  - specially designed device to support IPsec translation
- The HIP data packet carry no endpoint identifier
  - the SPI acts as an index to the identities in middleboxes

SPINAT (cont.)

- Packet translation by a SPINAT
  - translation between address domains and the SPI value
- The mapping state establishment by the middlebox
  - listening to the HIP BE
  - storing SPI values in both directions
- A separate signaling protocol for state establishment
  - the SPINAT is located off the BE path of HIP hosts
- A mobile host changes its network attachment point
  - different SPINAT device appears on the path
  - the HIP UPDATE exchange through the new SPINAT establishes the mapping state

SPINAT (cont.)

- The delay in packet forwarding
  - low throughput
  - packet loss
  - prevention of the use of Credit-Based Authorization (CBA)
- A large number of hosts are behind a single SPINAT
  - collisions of SPI values are quite likely
  - SPINAT cannot demultiplex IPsec packets with the same SPI
  - SPINAT drops packet or translates the SPI value on fly
- SPI value is not encrypted - SPINAT requirement
  - requirement is met with the HIP BE, but not with IKE and IKE2

SPINAT (cont.)

- SPI collisions resolve
  - SPI translation
  - enabling SPI modification by the SPINAT
- A packet is dropped in the case of a collision
  - a new SPI value is required
  - packet retransmission is necessary
- Retransmissions can be required in a busy network before new value is found
  - SPINAT can inform the source host of an SPI collision with an ICMP
  - the source host should not trust the ICMP but treat it only as a hint

SPINAT (cont.)

- On-the-fly SPI translation
  - the SPINAT replaces the colliding SPI value in packets
  - similarity to NAPT (Network Address Port Translation)
  - SPI translation is less intrusive
- Enabling on-the-fly translation
  - a modification of the BEET model is needed
  - permit of changing the SPI value
- The Stripped End-to-End Tunnel (SEET)
  - the SPI is not included into ESP header integrity protection
  - changing the details of IPsec processing not the packet headers

BLIND

- A framework for identity protection of hosts that are identified with public keys
  - the HIP BE is extended to hide identity
  - the real public keys of the host are not revealed
- The protection is robust against passive and active MITM
- Location privacy is achieved with forwarding agents

Location and identity privacy

- Public keys for identifying hosts
  - a privacy problem
  - third parties can determine the source host
- Various transactions could be linked together
  - the host uses the same public key
  - using a static IP address
- Reduction of the problem of user tracking
  - multiplexing multiple hosts behind a single NAT
  - using short address leases from DHCP
- IPv6 addresses could eliminate NAT translation and cause additional security issues (MAC in IPv6 autoconfiguration)

Location and identity privacy (cont.)

- DNS record information
  - host identity (public key) and location information (IP address)
- Identity and location are related and should be treated in an integrated approach
- The goal of the BLIND
  - framework for identity and location privacy
  - the identity protection - hiding the actual public keys from third parties
  - location privacy - integrating traffic forwarding with NAT translation and decoupling identities from locators

Location and identity privacy (cont.)

- Variations of DH key exchange are vulnerable to identity theft
  - the Responder must not generate the shared key before two messages are received - avoiding DoS
  - the Responder sends its public key in the first reply - the identity will be revealed
  - public key is sent encrypted in second packet of the BE
  - the Initiator cannot determine the Responder's identity after receiving the last message
  - an attacker can pretend to be a trusted correspondent host

Location and identity privacy (cont.)

- Prevent of identity revealing
  - the host public key and (HIT) can be encrypted with a secret key
  - cannot be easily implemented
- The BLIND framework protection
  - active and passive attackers
  - two-round-trip DH key exchange protocol
  - the host avoids storing their public keys in the reverse DNS or DHT repository - the framework achieves full location and identity privacy

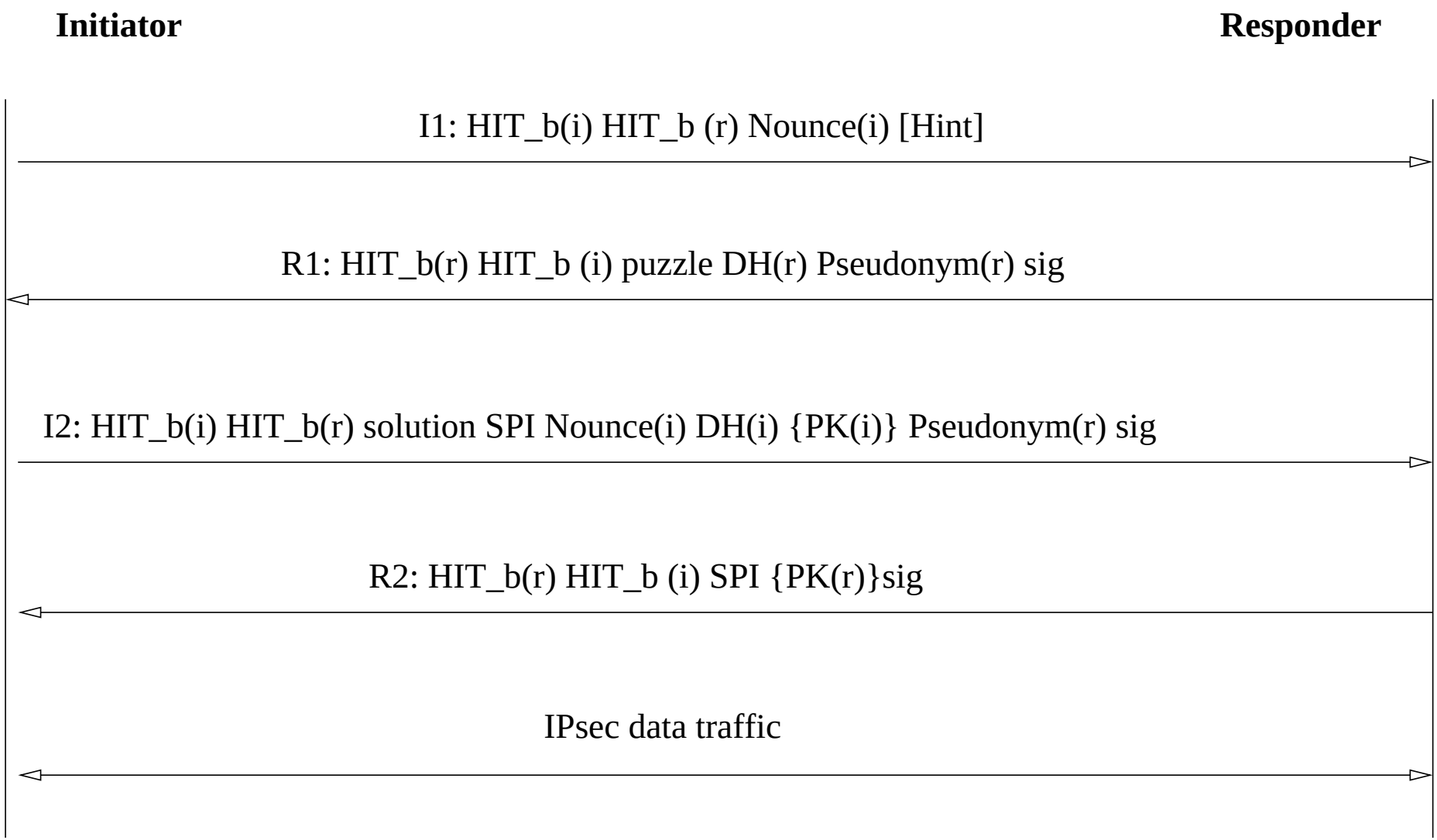
Protecting host identity

- Responder's HIT is known to the Initiator beforehand
  - the BLIND requirement
  - the Initiator can find out the Responder's HIT from a directory service
  - the Responder's can be stored from the previous session
- The identity protection
  - scrambling HITs during the DH key exchange
  - each scrambled version is used only once
- The blinded versions of HITs
  - $HIT_{blind}(i) = SHA1(nonce(i) \| HIT_{plain}(i))$
  - $HIT_{blind}(r) = SHA1(nonce(i) \| HIT_{plain}(r))$

Protecting host identity (cont.)

- The identity protection idea
  - the host knows a list of potential HITs that peers use
  - the host can test HITs one-by-one together with a known nonce
  - verifying the matching to the blinded HIT
  - only the blinded HIT and the nonce are known - computing of the plain HIT is infeasible

Modified base exchange with BLIND



Modified base exchange with BLIND (cont.)

- A random number as a pseudonym for the public key
  - the public key is not sent in R1
  - the public key is bound to the pseudonym by a signature
  - helps the Responder to locate its own public key when the I2 arrives
- The Initiator computes the keying material after receiving the R1
  - $KEY_1 = SHA1(KEY_{DH} \| HIT_b(i) \| HIT(r) \| 1)$
  - $KEY_n = SHA1(KEY_{DH} \| KEY_n - 1 \| n)$
  - $KEY_{MATERIAL} = KEY_1 \| ... \| KEY_n$

Modified base exchange with BLIND (cont.)

- The Initiator cannot verify the identity of the Responder
  - the Responder's public key is not presented in the R1
  - only the Responder's HIT is known to the Initiator
- The signature in R1 was invalid
  - the Initiator can terminate the BE
  - the Initiator's identity is no revealed
  - the third party does not know the actual Responder's HIT - the secret session key cannot be computed
- I2 is received - the Responder computes the keying material
- I2 includes the same nonce from I1 and pseudonym from R1

Protecting location privacy

- A forwarding agent
  - similar in function to a SPINAT
  - can be located anywhere in the Internet
- Virtual present in the Internet location
  - lease of a virtual interface from the forwarding agent
  - the forwarding agent replaces the IP addresses
- IP addresses are assigned from a pool
  - pool is common for all clients
  - IPv4 - the forwarding agent is likely to overload IP
  - IPv6 - the address mapping are likely to be one-by-one
  - $HIT^b_{dst}, IP_{dst}, HIT^b_{src}, IP_{src}, lifetime$

Protecting location privacy (cont.)

- Both the Initiator and Responder can use agents
  - two forwarding agents can participate in delivering packets
- Trustworthy forwarding agents
  - authenticate host before leasing the interface
  - a HIP host relies on agent not to reveal its identity
  - agent trusts the host not to establish many leases through it
  - can know the real host's identity
- Untrustworthy forwarding agents
  - allow anonymous leases
  - can reveal only the location of the host with a blinded identifier

Protecting location privacy (cont.)

- The Initiator and Responder use BLIND with agents
  - assumptions to achieve a complete identity and location privacy
  - the hosts should not keep public mappings from IP to public keys
  - the Initiator knows the HIT of the Responder beforehand
- An alternative proposal to BLIND
  - the use of rendezvous agents (RVA)
  - an RVA acts as a NAT
  - an RVA translates address between own private space and public Internet
  - hosts outside of the RVA are not able to learn the location of a host
- The privacy level is the same as that provided by the micromobility

Anonymous identifiers

- The privacy of the user
  - location and identity cannot be tracked
- Use of HIP - new security risks
  - pseudonym identifiers on the link layer are not sufficient for preserving privacy
  - the user can be tracked by identities on several protocol layers

Anonymous identifiers (cont.)

- Properties for privacy-preserving networking
  - anonymity - neither the communicating peer nor any third party is able to find the real identity of the user
  - pseudonymity - a weaker form of anonymity; persistent identifier is not connected to the real identity, individual actions of the user can be linked together
  - unlinkability - lack of possible connections between separate user actions
  - unobservability - a stronger form of unlinkability; separate actions cannot be detected or separated from the background actions of other users

Identifiers on protocol layers

Application	*URL, email, cookie, nickname*
Transport	*Port and sequence numbers*
HIP	*Host Identity, HIT*
IPsec	*SPI*
Network	*IPv4−v6 address*
Link	*MAC address*

Identifiers on protocol layers (cont.)

- Temporal identifiers can be used to track the user
  - IPsec SPI - temporal identifier
  - the IP is changed - SPI remains the same, visible to adversary outside of the local area network
  - the same SPI value - a small chance that the user is different one
  - deficiencies of the IKE for negotiating new SPI values
  - transmitting old and new SPI together - the attacker can correlate
- Public keys on the HIP layer
  - strongly identifies the user
  - tracking the user by examining the HIT, public keys and signatures
  - the LHIP is susceptible to privacy-targeted attacks - hash chains are easily linkable

Identifiers on protocol layers (cont.)

- Sequence numbers at IPsec and transport layers
  - can be used to track the user
  - sequence number is a little over the number seen at another location - the packets probably belong to the same user
  - the transmission rate of the links defines the probability
- The port number at the transport layer
  - can reveal the user
  - well-known ports for communication
  - ephemeral source ports are used sequentially
  - the attacker can track the user by noticing the same or close port
- The most serious privacy treats are limited to MAC, IP, SPI and HIT

Changing identifiers

- A natural approach to reducing privacy treats
  - short-lived identifiers
  - regularly changing prevents user tracking
  - simultaneously changing at all layers is necessary
- HIP privacy architecture that changes identifiers
  - was developed in TKK
  - simultaneous changing of identifiers on MAC, IP and HIP/IPsec
  - the default frequency of changes is every 6-10 min
  - each change - a delay of 3 sec, loss of packet is possible

Changing identifiers (cont.)

- Changing identifiers without explicit signaling
  - shared pseudo-random sequences
  - agreement on sequence of pseudo-random values
  - the time expires - the host starts to use the next value
  - SHA-256 hash function can be applied to create values
  - i-th identifier - $SHA(k||i||ID_{type})$, k - the shared secret
  - packet loss and reordering can cause gaps in the values
  - the host must be prepared to accept any value from a window
  - the only non-constant identifier that requires changing is ESP sequence number

Changing identifiers (cont.)

- Frequent change of public keys
  - computationally expensive
  - a hybrid with symmetric cryptography
  - public keys can be used only to agree on a secret key
  - secret key is later used to authenticate DH key exchange
- Hash chains on LHIP
  - easily verifiable by a third party in the forward direction
  - a keyed version of a hash chain $h_i = H(h(i-1)\|k)$ - verifiable only by parties sharing the secret k
- Periodic change of identifiers
  - middleboxes can try to follow the data flow

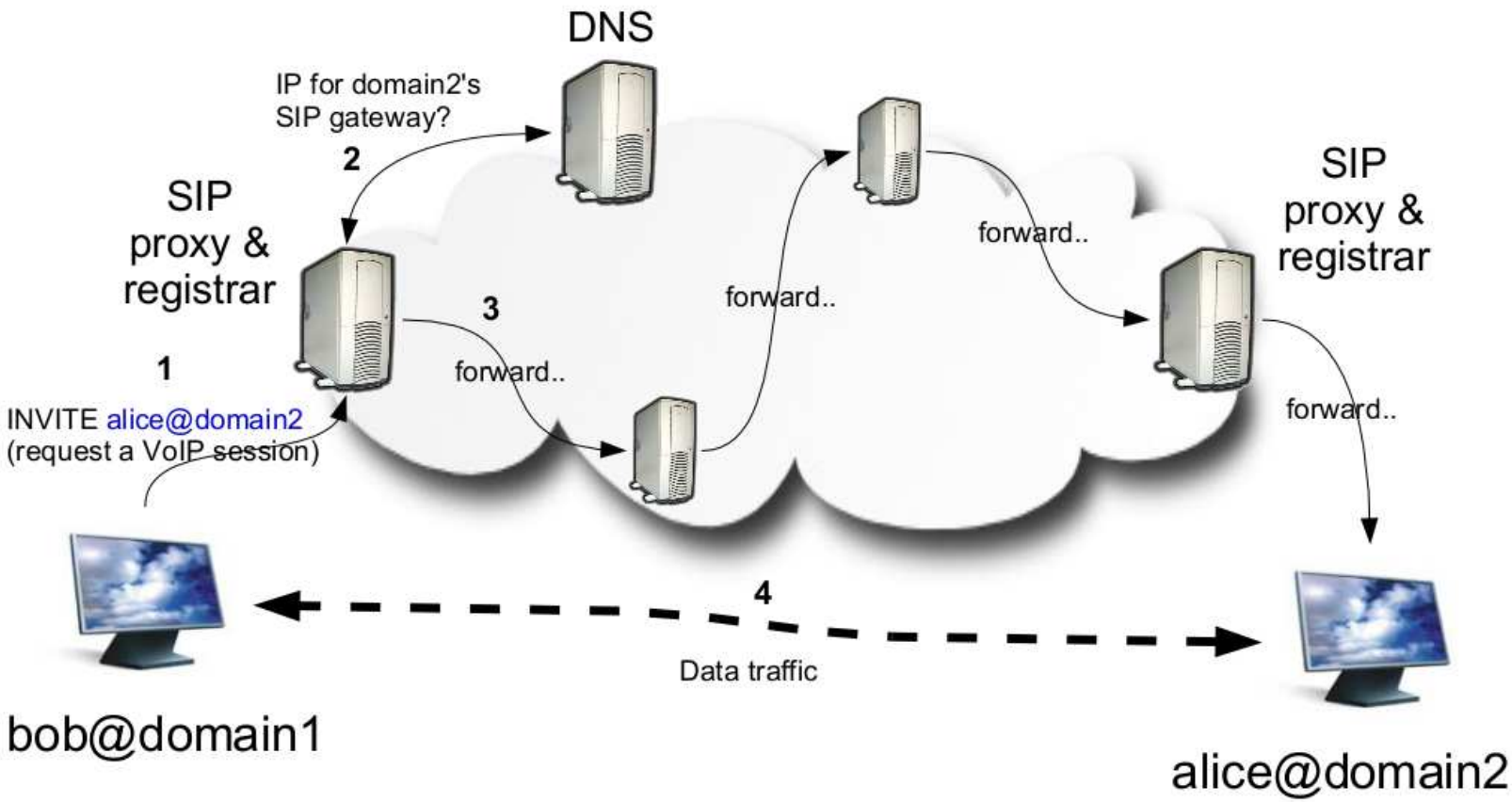
Possible HIP applications

P2PSIP-HIP Prototype[4]. Background: CS-SIP

- The Session Initiation Protocol
  - IETF standardized HTTP / MIME like text-based protocol
  - successful, industry accepted
  - used to negotiate how connections are established
- Most commonly used with registrar servers to locate peers
  - the name client-server (CS) SIP
  - the protocol syntax is not tied to any particular architecture

[4] Based on slides by Joakim Koskela

CS-SIP scenario



P2PSIP

- Remove centralized components
  - scalability, sustainability, single point-of-failure
  - MANETs (mobile and ad-hoc), Intranets, private networks
  - deployment in developing worlds
  - serverless, small-scale IP-PBX (no administration, zero-configuration)
  - security: privacy and IT policies, anonymous access
  - ISP blocking well-known gateways/ports
  - cost savings: infrastructure investments and operation

P2PSIP (cont.)

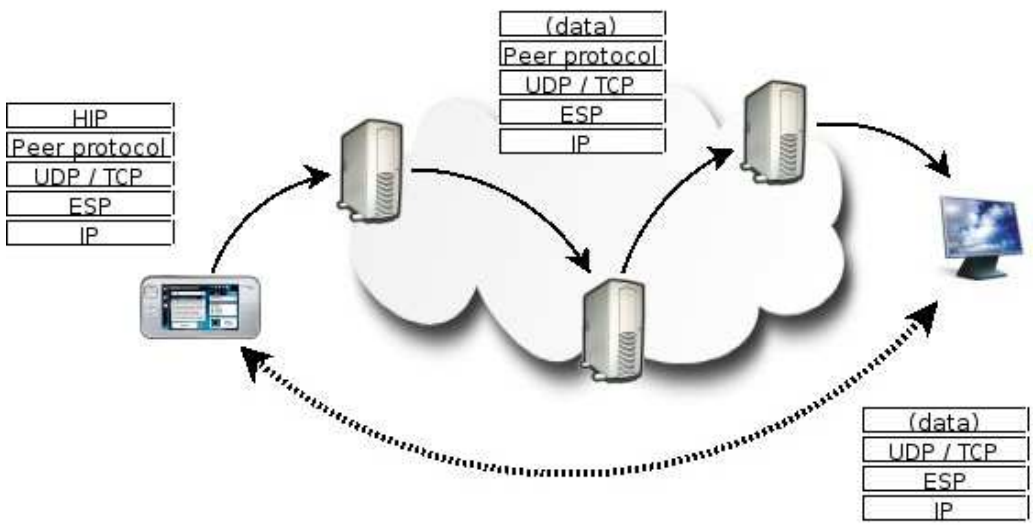
- IETF P2PSIP WG
  - replace centralized servers with a DHT for lookup and routing
  - many open issues
  - different viewpoints complicates standardization (Intranet VoIP vs. Skype replacement)

P2PSIP with HIP - the design

- Using HIP for P2PSIP may be beneficial in many ways
  - strong authentication by means of His
  - E2E encryption
  - mobility (difficult in SIP without centralized server)
  - NAT traversal (most connections go now to peers behind NATs)
- Design - draft-hautakorpi-with-hip-01
- Topology and protocol specifics not discussed
  - use HIP for all connections
- Use the overlay for relaying the HIP BEX
- Infrastructure services located using the overlay

The design

- Use existing overlay algorithms over HIP
- Use overlay to perform BEX to establish data connections
- Data connections P2P
  - use RVS/relays as defined in HIP NAT traversal

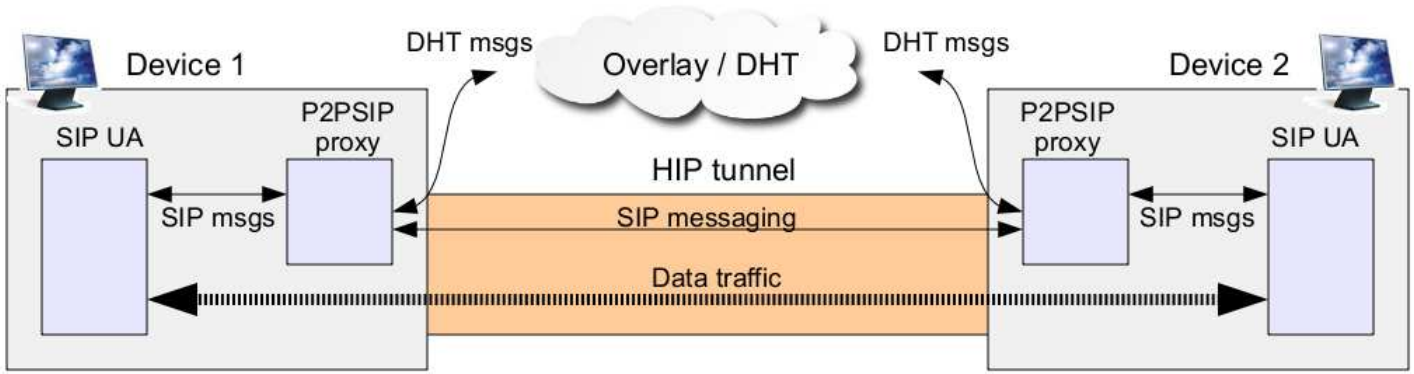


Prototype

- A local SIP proxy supporting legacy SIP applications (UAs)
- HIP used for all peer-to-peer connections
- Storage for registrations and resource lookup modular
  - currently OpenDHT and LAN broadcast used
  - own DHT possible, with or without HIP

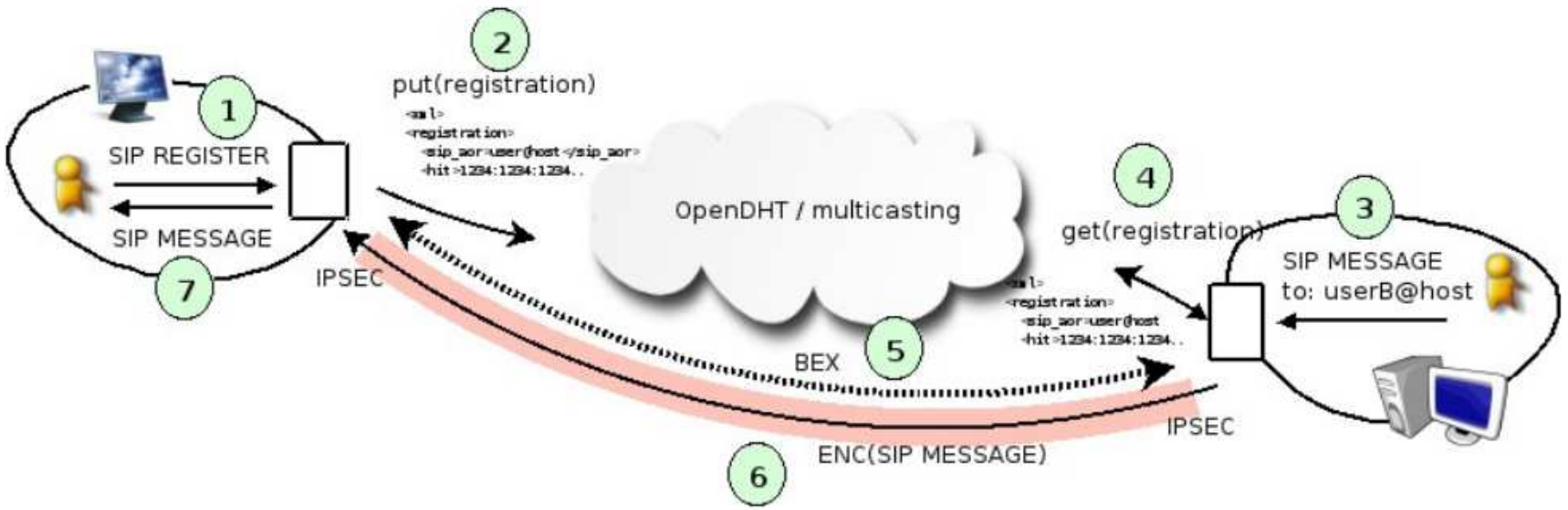
Prototype (cont.)

- SIP identities tied to HITs (and locators) using certificates
  - signed XML packages
  - identities are requested from 3rd party identity authorities



Prototype (cont.)

- Currently the overlay is used only for storage (no routing)
- All P2P traffic (even simple SIP MESSAGEs) require a HIP connection



Prototype (cont.)

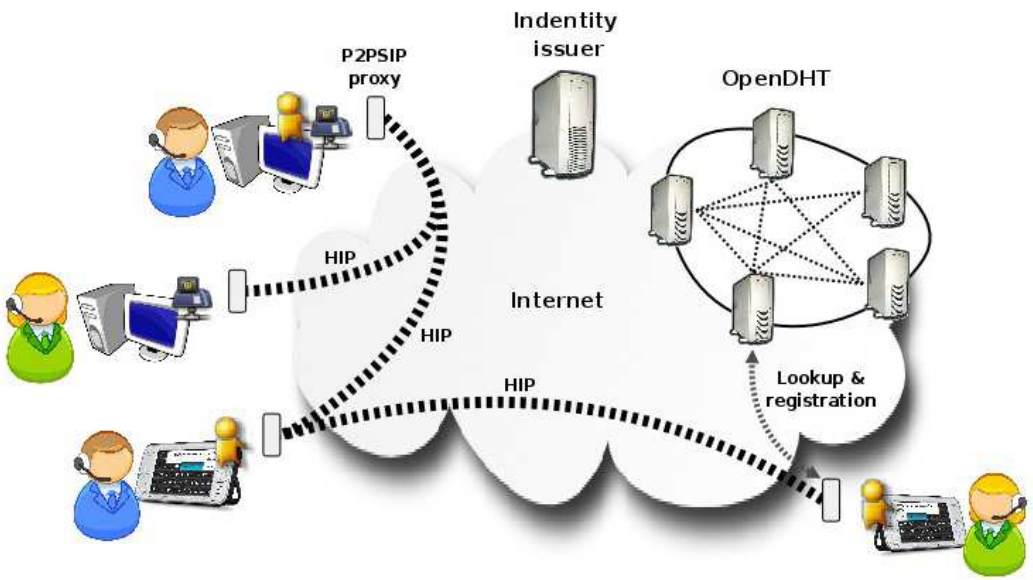
- Identities (SIP AORs) are acquired from identity authorities (CAs)
  - peers verify each others identities by checking signature
- Multiple identity issuers possible
  - for peers to be able to communicate, their identity need to be issued by a CA trusted by the other peer
  - clients may have multiple root CA certificates installed
- Peers are found by looking up registration packages from overlay
  - found by SIP AOR, contains HIT, locators (IP), RVS, validity

Prototype (cont.)

- Overlay currently used as storage for registration packages only
  - distributed storage
- Overlay handling modular
  - currently OpenDHT and multicast
  - simple get / put / rm interface for additional modules
- Own DHT and overlay routing on the roadmap

P2PSIP prototype

- Multiple UAs per proxy possible
  - gateway-like architectures possible
- SIP UA inter-compatibility tested
  - Pidgin (GAIM), Ekiga (GnomeMeeting), MiniSIP, WengoPhone, Nokia Series60, Maemo SIP communicator (Nokia N8x0)...



Virtual Private Networking

- Traditional Virtual Private Network (VPN) solutions
  - various forms of tunneling (L2TP)
  - significant amount of header overhead is added
  - difficult for middleboxes to process
  - VPN password - each establishment of tunnel
  - most VPN solutions do not support mobility or multihoming
- An ideal VPN solution
  - enabling Road Warrior
  - no requirement of manual authentication
  - secure and fault-tolerant connectivity to the intranet
  - relying on locally cached data whenever possible

Virtual Private Networking (cont.)

- Road Warriors face obstacles when trying to access intranets with VPN
  - widespread presence of NATs
  - misconfigured firewalls
  - disruptive authentication methods for network access
  - legacy intranet applications
- Many existing VPN products do not work
  - the user is located behind middleboxes
  - VPN are forced to use UDP encapsulation to traverse the NAT and firewalls

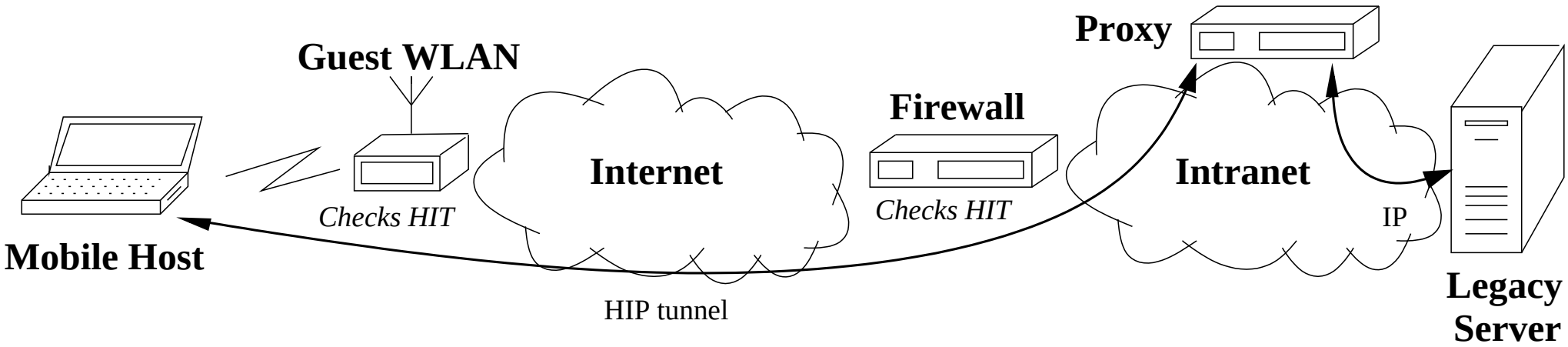
Virtual Private Networking (cont.)

- Gaining the Internet connectivity
  - redirection to a captive web page
  - registration and enter of user name and password are required
  - time consumption
  - problems for network protocols and applications
- Upgrading the corporate application servers to support new VPN software can take years

Virtual Private Networking (cont.)

- HIP as a VPN solution
  - automatic authentication
  - Single Sign-On (SSO) functionality to visiting networks
  - obtaining a list of HITs authorized to use the network
  - manual configuration of the list of authorized HITs for small companies
  - an automated solution involving PKI with ACL for larger companies
- A HIP VPN solution is likely to integrate
  - the HIP aware firewall
  - UDP encapsulation for legacy NAT traversal
  - a HIP proxy located in the intranet

Virtual Private Networking with HIP



- proxy servers - an endpoint of the HIP UDP tunnel, decapsulates payload from IPsec packets and forwards them to the corporate servers
- the proxy is likely to be necessary (corporate servers cannot be updated to support HIP directly)
- the user gets benefits accessing an intranet with HIP - mobility and multihoming support

Virtual Private Networking (cont.)

- Configuration of a simple ACL for accessing a home Linux server over HIP
  - ACLs are based on IPs and ports (/etc/hosts.allow and /etc/hosts.deny)
  - adding IP to the files - control of access to SSH, HTTP, or FTP
  - ”ALL” - the default handling for allowing or denying connections
  - access control with IP is inconvenient (users move frequently)
  - adding HITs to files instead of IPs
  - HIP BE successfully completes, the host’s HIT matches one in host.allow - the host can use service

P2P Internet Sharing Architecture

- Open WLAN networks
  - desire to enable other users to benefit Internet connectivity
  - reason of low technical competence
  - using without prior permission can have legal consequences
- Risks of open WLAN
  - traffic intercept
  - exploration of password or other sensitive information
- The users of WLAN need to be authenticated
- The user traffic needs to be encrypted

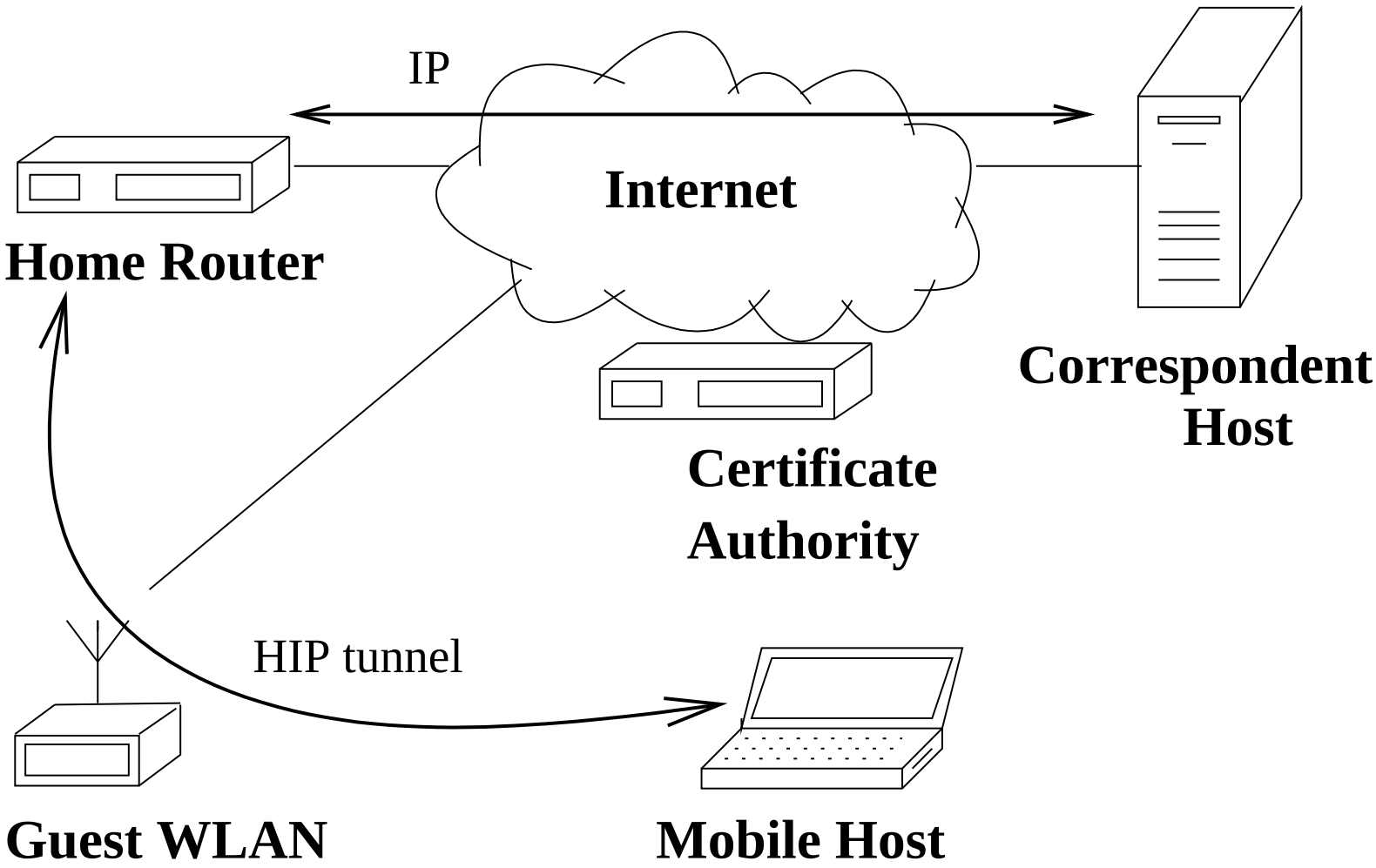
P2P Internet Sharing Architecture (cont.)

- FON and Wippies companies
  - free access points in exchange to offering other users network connectivity
  - authentication model is weak and disruptive
  - the traffic is not encrypted
  - the traffic could not be separated from other users' traffic
  - a malicious user could eavesdrop MAC and IP of authenticated user
  - the service is very popular - hundreds of thousands of installed access point worldwide

P2P Wi-Fi Internet Sharing Architecture

- PISA
  - solving of traffic security problem and access control by HIP
  - WLAN AP only allow authenticating HIP traffic to the home router that has been certified by the community
  - legal responsibility from the owner of the AP
  - the user is protected from possible attacks on the traffic

P2P Wi-Fi Internet Sharing Architecture (cont.)



P2P Wi-Fi Internet Sharing Architecture (cont.)

- Host authentication
  - initiating BE from host to the home router
  - the WLAN AP knows the list of authorized routers and forwards packets
  - the AP - a middlebox that verifies signature in HIP BE
  - the AP authenticates the host
  - a nonce insert - prevention of relying previously recorded BE
- Verifying that the user belongs to the Wi-Fi sharing community
  - a certificate in a CERT is required
  - a certificate includes HI of the home router signed by a CA

P2P Wi-Fi Internet Sharing Architecture (cont.)

- Authentication succeeds - IPsec tunnel to the home router
  - data packets are sent inside the tunnel
  - the home router extracts payload and forwards it
- Data compression
  - compensation for asymmetric speeds of many home ADSL lines
- Seamless user mobility between WLAN APs
  - authentication by the new AP - the UPDATE exchange
- The drawback of PISA
  - potential routing inefficiency (the correspondent host is close to the user, the home router is far away)

P2P Wi-Fi Internet Sharing Architecture (cont.)

- The PISA prototype and demo
  - a modified version of WLAN APs running OpenWRT Linux distribution
  - HIPL was extended with middlebox authentication functionality
  - the user is able to browse the web by transparently authenticating through a local AP
  - the traffic is tunneled to a remote home router
  - the attacker is only able to observe HIP control and encrypted IPsec packets by snooping on the local WLAN

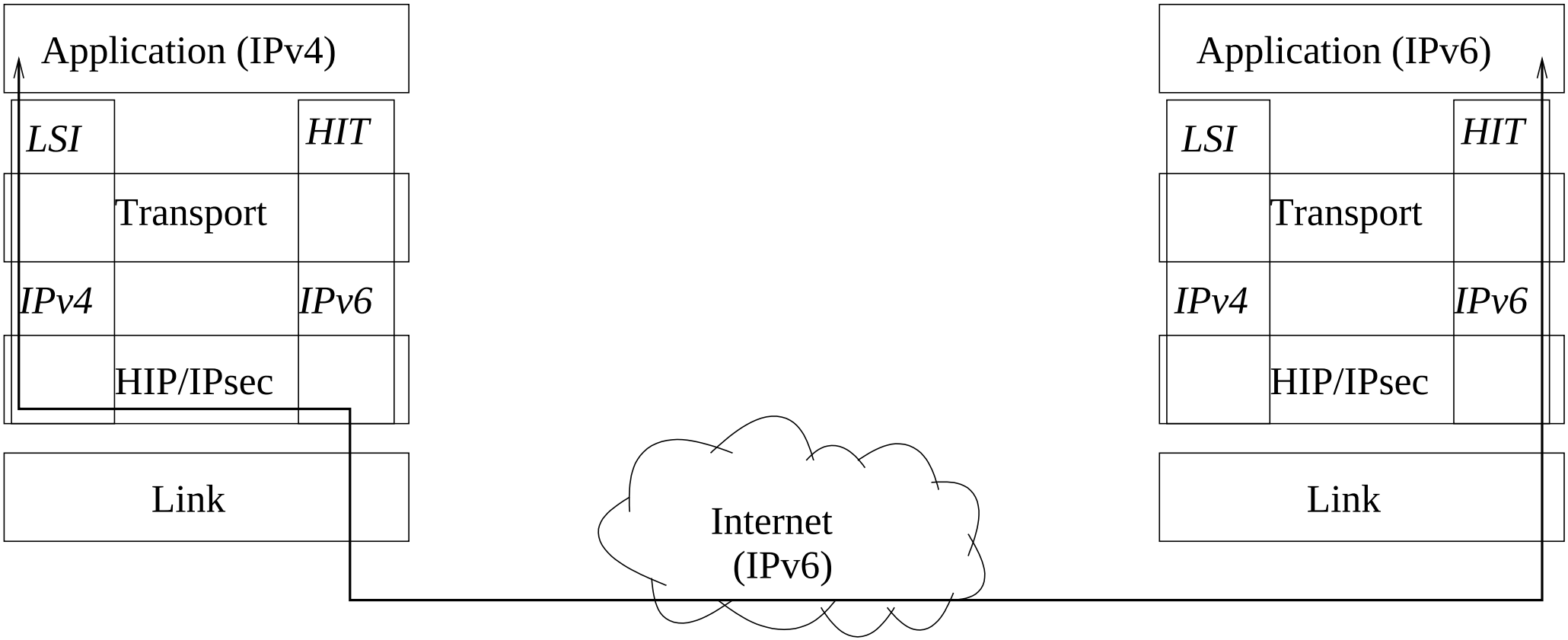
Interoperating IPv4 and IPv6

- IP-versions co-existing
  - the transition process is likely to take years
  - raising the issue of interoperation
  - a significant number of legacy IPv4-only application
- The classic TCP/IP architecture
  - the transport and networking layers are connected
  - interdependency prevents independent evolution
- The use of HIP
  - the Internet stack is closer to the OSI model
  - the multihoming - use of IPv4 and IPv6 simultaneously bound to a single or several network interfaces
  - the mobility - cross-family handovers between IPv4 and IPv6

Interoperating IPv4 and IPv6 (cont.)

- Cross-family communication
  - the transport layer checksums need to be computed
  - identical computation at both communication hosts
  - computation over 128-bit source and destination HITs
  - IPv6-format pseudoheader is used both for IPv4 and IPv6
  - IPv4 and IPv6 applications can communicate
- The Ericsson implementation of HIP
  - data packets always pass through the IPv6 version of TCP and UDP, even for IPv4 sockets

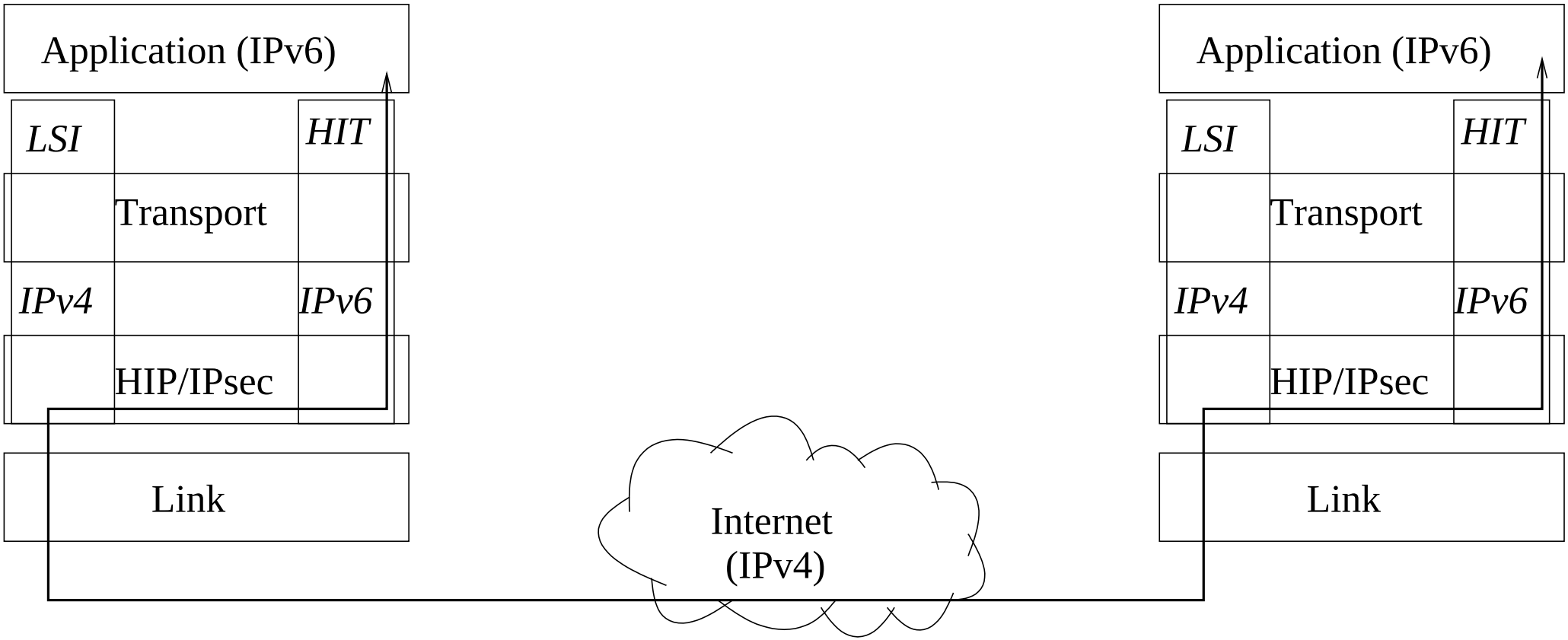
IPv4-only application talking to IPv6 application via HIP



Interoperating IPv4 and IPv6 (cont.)

- The applications explicitly using IP
  - using for network diagnostics
  - problems with cross-family communication using HIP
  - operating on the IP layer without involving HIP
- The applications passing IP in messages
  - the IP family is the same at both endpoints - normal operating
  - IPv4 applications - LSI is sent
  - IPv6 applications - HIT is passed
  - LSI to the IPv6 application - the communication is successful (most IPv6 applications understand IPv4-format)
  - HIT to the IPv4 application - the communication can fail (legacy IPv4 do not understand IPv6-format)

Two IPv6 applications talking over IPv4 network via HIP



Secure Mobile Architecture

- SMA - an integrated network architecture
  - standardized by the OpenGroup (The Open Group 2004)
- The Open Group
  - a vendor- and technology-neutral organization
  - providing integrated solutions for information flow within and between organizations
  - members - Capgemini, EDS, HP, IBM, and NEC
  - companies, government bodies, academia are part of the consortium

Components of SMA

- SMA Vision and Architecture in 2004
  - joint work of Security and Mobile Management Forums
  - combination of PKI, HIP, NDS, LENS
  - Boeing - the primary contributor to including HIP in the architecture
- The SMA name meaning
  - ”Secure” - having user and host authentication through enterprise directory as well as cryptographic identifiers in each data packet
  - ”Mobile” - the ability to change network location while being backwards compatible with legacy applications
  - ”Architecture” - an integrated approach to augment the enterprise network with end-to-end encryption, delegation-based authentication, enabling secure access of hosts outside of the security perimeter

Components of SMA (cont.)

- The business needs of having secure network access
  - use of various available networks (LAN, WLAN, cellular)
  - capability to switch between them
  - multi-level security that works on top of any MAC layer
  - backward compatibility
  - automatic configuration
- The PKI part
  - company PKI servers
  - an employee's badge card with built-in smart chip
  - Registration Authority connected by SSL tunnels to the card reader
  - a card is inserted into the reader - request of a temporal certificate valid for 8 hours by supplying the badge certificate stored on the chip

Components of SMA (cont.)

- Information providing by NDS
  - mapping between host identity and address for internal users
  - Secure Lightweight Directory Access Protocol (SLDAP)
  - integration with the DNS (enabling updates)
- The location server
  - part of NDS
  - geographical coordinates of a user
  - coordinates of a policy decision daemon
  - coordinates of various middleboxes

Components of SMA (cont.)

- New clients authentication
  - a two-stage provisioning process
  - regular authentication with the DHT and AAA server (RADIUS) - obtaining an IP
  - contact with the Registration Authority - obtaining a temporal certificate over TLS
  - updating an identity to IP mapping in the directory (SLDAP)
- The first prototype of directory service at the SMA testbed
  - status updates directly from the LDAP server
  - query to the decision daemon - whether the client was authorized
  - LDAP updates the client's current IP to the DNS

Components of SMA (cont.)

- Extension of the directory deployment
  - the publish-subscribe paradigm
  - the Message Broker Infrastructure - message delivery
  - incoming data from a location server - sensors and RFID tags, the decision daemon, LDAP server, HIP daemons are notified according to their subscriptions

Components of SMA (cont.)

- Location-Enabled Network Service
  - tracking of workers and equipment
  - passive tag gates detect an object passing through
  - a WLAN positioning system determines user location
  - AeroScout active RFID tags possess a unique 802.11 MAC address
  - the LENS system is connected to the Boeing intranet
  - an integral part of the overall SMA testbed

SMA testbed at Boeing

- Richard Paine of Boeing
  - every step can be identified with the person and equipment
- Traditional approach
  - a worker personal stamp - mark of the parts attached to an airplane
  - an accident - the cause could be tracked to the part and person installing it
- Driving force behind the SMA in Boeing
  - the identification of airplane construction
  - the identification of parts automatic and secure
  - keep up with the schedule - the airplane gradual move through the assembly process

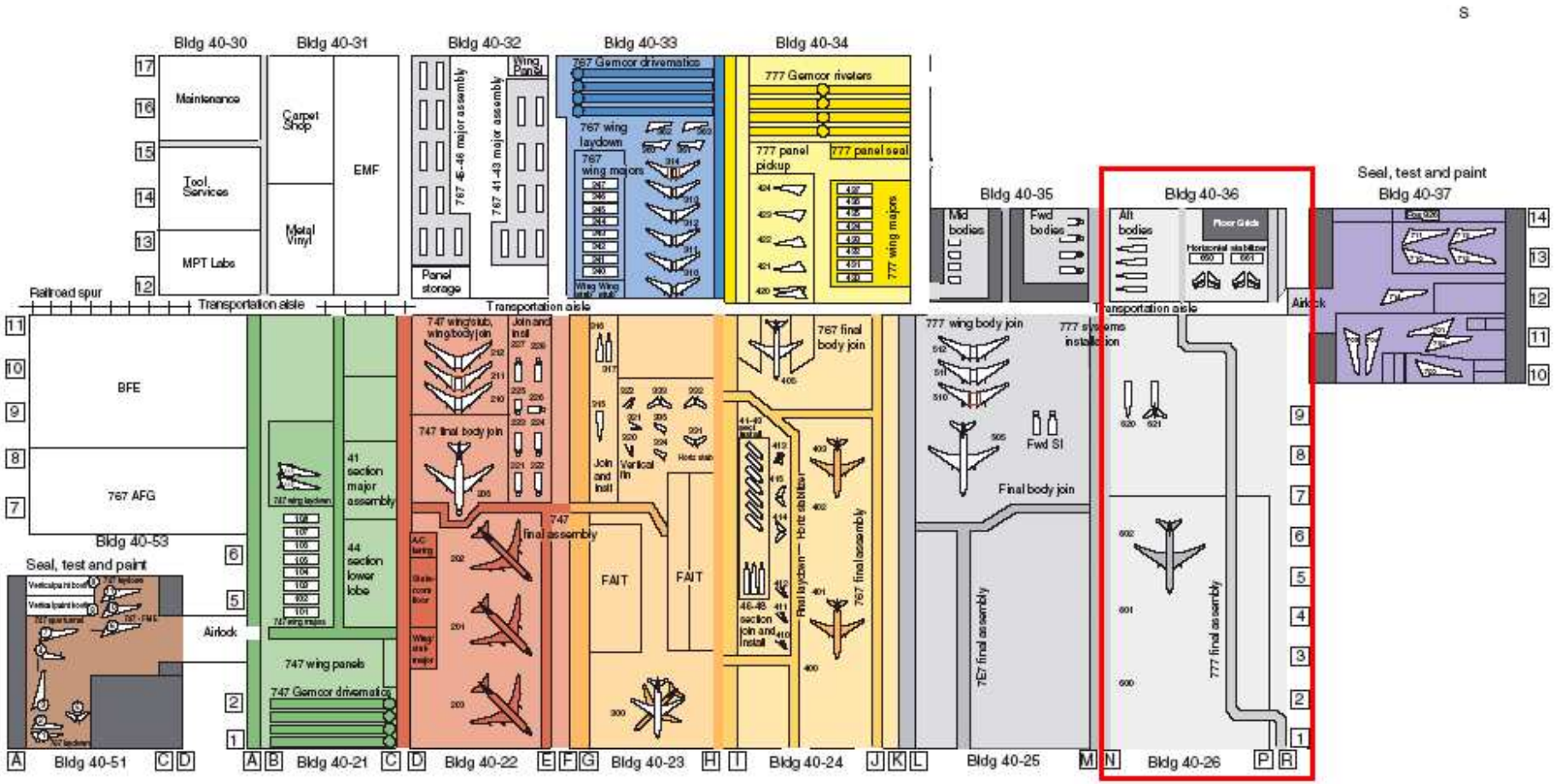
SMA testbed at Boeing (cont.)

- The current practice
  - logging the operation on the PC by mechanics
- Unsuccessful use of WLAN with portable devices
  - going out of network coverage - loss of VPN connectivity
  - the use of PC with LAN is preferable for workers
- SMA improves the manufacturing process
  - manual authentication is not necessary
  - reconnection is not needed
  - end-to-end packet authentication
  - prevention of spoofing
  - seamless WLAN routing

SMA testbed at Boeing (cont.)

- An SMA testbed is deployed at the Everett manufacturing site
  - assembling of Boeing 777 airplane
  - Crawlers robots moves components using Supervisory Control And Data Acquisition system
  - SMA/HIP Endbox is installed on Crawlers, connected to FactoryNet
  - the Endbox contains a SIM card - strong authentication over WLAN to the RA and Directory
  - the Endbox securely communicates to the Robot Controller over HIP SA
  - a HIP Bridge - legacy Ethernet equipment can connect to SMA components

Location of SMA testbed on Everett airplane factory [5]



[5] Photos ©Boeing. Used with permission.

HIP Endbox installed on the crawler platform under the plane

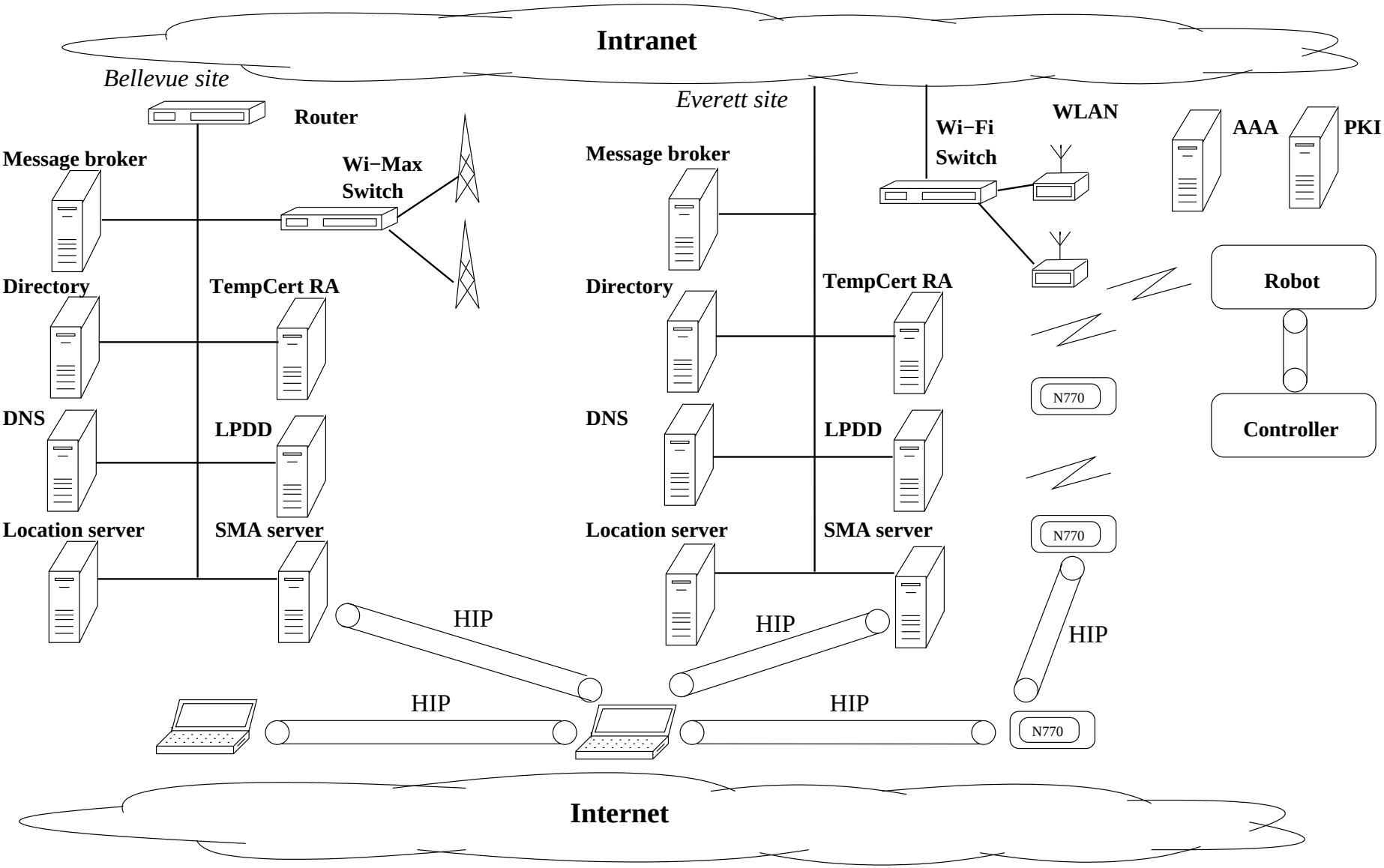
SMA testbed at Boeing (cont.)

- The secure handover between cellular and WLAN over HIP
  - demonstration on an SMA testbed in 2005
  - WLAN APs were connected to the Boeing intranet
  - traffic to the cellular network traveled through the public Internet
  - traffic enters the intranet through a Netscreen firewall
  - enabling of host moving outside of the security perimeter

SMA testbed at Boeing (cont.)

- The initial testbed version - 2004
- The SMA testbed at Boeing in 2007
  - two subnetworks: Bellevue and Everett
  - a company-wide AAA server (RADIUS) and PKI server
  - each subnetwork - Message Broker, Directory server, DNS server, Location information server, Wireless access switch, Temporal certificate Registration Authority, SMA AP
  - the Bellevue - a WiMAX wireless network (towers are connected to a switch)
  - the Everett - a Wi-Fi switch connected to several WLAN AP
  - Nokia N770 Internet Tablets - user terminals

Components of SMA testbed at Boeing



SMA testbed at Boeing (cont.)

- SMA mobile terminals
  - can connect over HIP to the SMA server in each subnetwork
  - can connect to each other
  - public Internet access to pass the intranet
  - Voice over WLAN (VoWLAN) - N770
- Advantages of the SMA deployment
  - secure mobility improvement
  - reduction of complexity and management cost
  - the possibility to access the intranet services outside of the security perimeter
  - no changes are required to routing infrastructure

SMA testbed at Boeing (cont.)

- Advantages of the SMA deployment
  - non-HIP hosts can be allowed in the deployment
  - the middleboxes are able to authenticate the traffic not based on changing IP
  - seamless integration of IPv4 and IPv6
  - PKI secures host and user identities from spoofing
  - setting a new host without a complete bureaucratic process

Live application migration

- A *virtual machine* (VM)
  - an instance of an operating system
  - virtual hardware is mapped to the real hardware of the host machine
  - several virtual machines can run on the same host machine
  - reduction of cost of hosting WWW servers
  - convenience of running Linux in a separate window after booting from Windows
- The *migration* of VM
  - migration from one host machine to another
  - useful - the host machine needs to be shut down for maintenance
  - operating without a noticeable break
  - load balancing

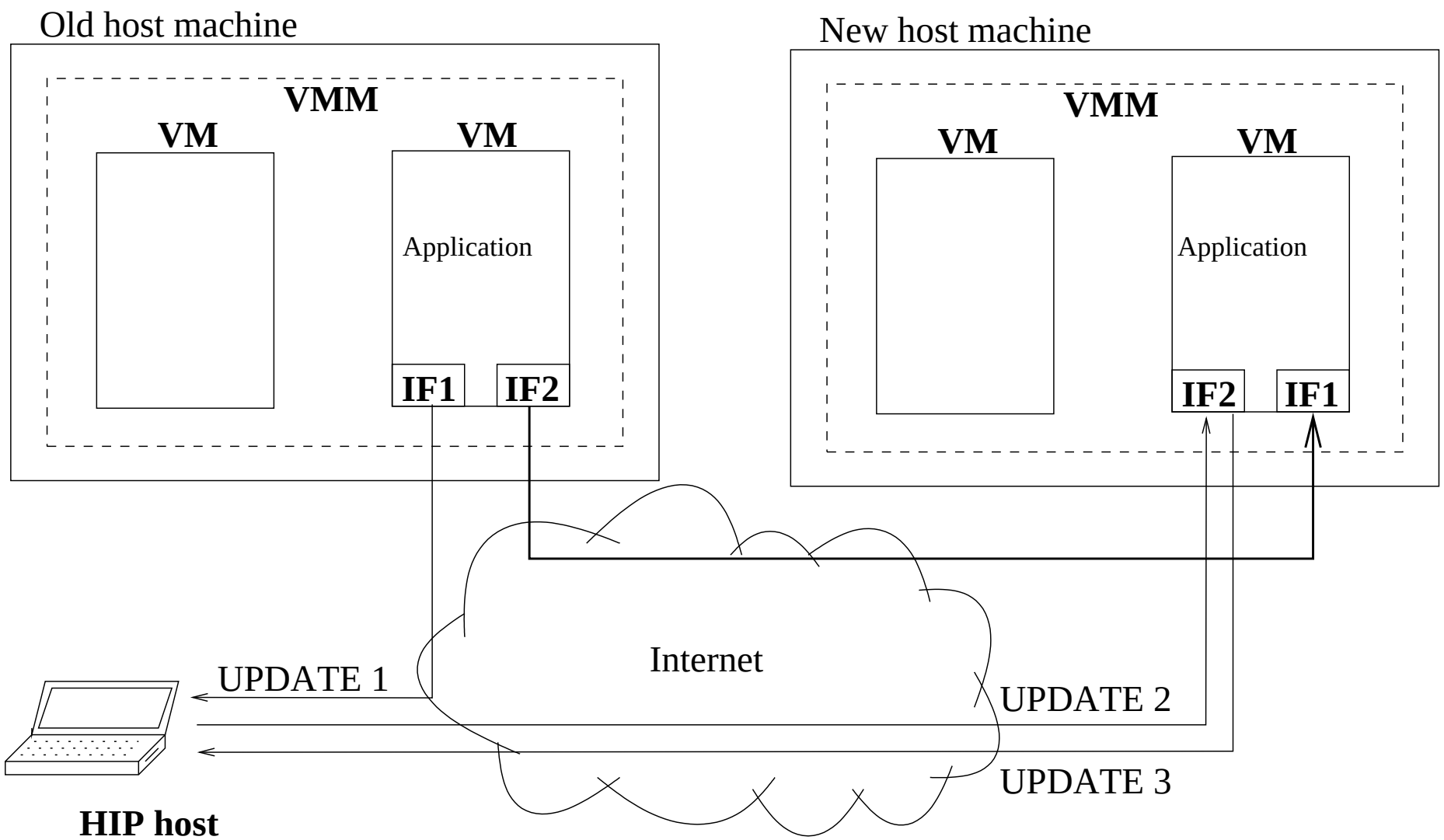
Live application migration (cont.)

- *Residual dependency*
  - new host machine needs the old host to operate
  - the network traffic to the VM can arrive to the old host machine
  - forwarding traffic to the new host machine is needed
  - the active sessions have not terminated - the old host machine cannot be brought down
- The *Virtual Machine Monitor*
  - management of virtual machine execution
- A full virtualization approach
  - the VM sees a virtual hardware
  - the VM can run unmodified operating system code
  - inefficient in performance

Live application migration (cont.)

- A para-virtualization approach
  - small modifications to the operating system
  - better performance is provided
- The Xen VMM
  - hypervisor - interface for VM hardware
- Xen migration
  - few hundred milliseconds within a single LAN
  - avoiding residual dependencies - VM migration moves the MAC address from the old to the new host
  - the possible migration area is limited to a LAN

Wide-area migration of Xen VM



Live application migration (cont.)

- A simple approach to wide-area VM migration
  - break-before-make
  - long break in network connectivity of a VM
  - using when a VM obtains an IP at the new host machine before moving
- Wide-area migration of Xen
  - two virtual Ethernet interfaces (one connected to the old and one to the new host machine)
  - VM is multihomed during the migration
- The timing of the migration
  - impact on the outage of applications

Live application migration (cont.)

- The connections are migrated before VM migration
  - the link layer tunnel is loaded with redirection traffic
  - reduction of the tunnel capacity
  - VM migration slows down
- Migrating connections after VM migration
  - increase of the latency
  - packet loss for applications
- VM migration should occur synchronously

Live application migration (cont.)

- Return routability test
  - verifying the ownership of the new IP
- Steps of a Xen VM migration
  - preparation
  - address configuration
  - migration starts
  - binding update
  - suspending
  - activation
  - traffic switch
  - closing

Live application migration (cont.)

- Host identity relocation
  - keys are on a separate security device - keys are not easily movable
  - residual dependency - dependence on signing packets
- Delegation certificates - an alternative approach
  - pseudonym public/private keys for VM identification
  - a certificate binds the pseudonym to the permanent identity
- Start of the migration process
  - pre-copying - the memory image is transferred to the new machine
  - a bandwidth-hungry operation - tens of seconds
  - dirty (recently modified) memory pages are transferred

Live application migration (cont.)

- A signal to the HIP daemon to being the UPDATE procedure
  - changing of current IP in DHT
  - the Xen VM and HIP daemon communication - a TCP and a raw IP socket
- The prototype implementation of VM wide-area migration
  - OpenHIP
  - OpenDHT for storing mappings of host identities to IP
  - two servers test the migration process
  - emulation by the Linux kernel
  - workload connection to the migrating VM by the PlanetLab clients
  - TCP and UDP traffics were tested

Live application migration (cont.)

- Increase of TCP window
  - several megabytes
  - high transmission rate is necessary for a memory image transfer
- TCP experiments
  - the VM was prepared for migration in 9 s
  - suspend interval of 170 ms
  - TCP ack loss - the overall outage on the connection of 370 ms
- UDP experiments
  - not sensitive to losses
  - the outage is less than 200 ms

Network operator viewpoint on HIP

- The organized use of HIP in a telecommunication network
  - a number of issues for network operator
- The flat rate model
  - the simplest charging scheme (fixed price for unlimited use of network)
  - easy to implement
  - popular with users
  - none issues with the use of HIP
  - most cellular and WLAN do not use flat rate - the high cost of installation and operation

Network operator viewpoint on HIP (cont.)

- The time-based model
  - charging of the time spent online
  - popular with WLAN hotspots
  - well function with HIP
- The data volume model
  - charging of the amount of received and transmitted data
  - often application in cellular networks (3G)
  - working in the presence of HIP

Network operator viewpoint on HIP (cont.)

- Content-based model
  - the most difficult from the HIP point
  - data is priced differently depending on its type
  - various data - different per-byte values to the customer
  - with HIP information is encrypted - the use of the model is impossible
  - the operator is less willing to promote the use of HIP
- A lawful interception of user traffic
  - a capability to eavesdrop user communication requirement
  - HIP support (user private keys are not revealed to the operator) - lawful interception is not possible

Network operator viewpoint on HIP (cont.)

- Key escrow
  - arrangement of providing operator with keys
- Termination of HIP association on the proxy
  - performing of lawful interception
  - collect of charging information
  - HIP association establishment to the traffic destination
- SIP for locating user and placing calls
- Integration of HIP and SIP is important

Network operator viewpoint on HIP (cont.)

- Central creation and maintaining of HI
  - 3G - the identifier is stored in SIM card (PIN protection)
  - the user identity is bound to the SIM
- HIP helps to switch between interfaces
  - smart phones have both 3G and WLAN interfaces
  - the same operator manages both networks - the authentication is easy
- Mobility and multihoming in a multi-operator environment
  - a challenging area
  - accounting and agreement issues
  - interface selection - the user phone or the operator network elements

Application interface

Using legacy applications with HIP

- IETF document (Henderson *et al* 2007)
  - possible methods to use HIP with applications
  - employ of the standard Berkeley socket API
- The first method - IP in applications
  - mapping between the host address and identity with the HIP layer
  - better support for *referrals*
  - a referral - an attempt to pass a locator to an application on another host

Using legacy applications with HIP (cont.)

- The second method - DNS resolution
  - address-compatible representations of host identity
- The third method - a HIT or LSI
  - no prior DNS resolution
- Channel binding
  - the advantage of using the HIT versus an IP
  - the application is bound to the cryptographic host name

Using IP addresses

- Application connect call with a IP
- External mechanism to configure HIP associations
  - an entry to the security policy database is added
  - mapping - wrapping the socket API or patching the OS
- Several ways of triggering the HIP association
- Static configuration
  - policy rules are pre-configured
  - the security level - the current use of IPsec policies

Using IP addresses (cont.)

- Reverse DNS lookup
  - the HIP layer can perform the reverse DNS mapping (from IP to the FQDN)
- DHT lookup
  - a DHT stores HIT indexed by the IP - the HIP layer can lookup the HIT (before sending the I1)
  - security and delay concerns are similar to DNS lookup
- Opportunistic mode
  - the HIP layer tries to send an I1 to all IP that an application connects
  - the security level is similar to plain IP (MITM)

Using IP addresses (cont.)

- Complete mobility support is not provided
  - the application uses IP calling the sendto()
  - the destination host changes the IP - connectivity breaks
  - a connect() call is used for a socket - the change of IP can be transparent
- Weaker security binding between the identity and locator
  - implemented internally at the HIP layer
  - not visible to the application or the user
- Better support for referrals
  - the case where an application passes its IP to a peer application

Using DNS resolution

- An approach to support HIP for applications using the DNS
- The resolver library can be modified to support HIP
- Use of a suffix
  - a suffix ".hip" for an application to request HIT or LSI
  - "www.example.com.hip" request - only HIT or LSI is returned
  - finer control of application needs in a user-friendly way
- Returning HIT and LSI instead of IP
  - transparent remove of IP from DNS reply
  - the user is not required to know anything about HIP
  - HIT or LSI is not found - IP can be returned

Using DNS resolution (cont.)

- The drawback of the DNS-based approach
  - lack of support of referrals
  - not all legacy applications use DNS
- The strong side - closer binding of the peer identity in the application
- Security properties depend on the use of DNSSEC
  - DNSSEC is properly used - the security level is high
  - DNSSEC is not used - the authentication level is similar to plain IP

Directly using HIT

- Best channel binding properties
- Highest security level
- The random-looking HIT format
  - cumbersome for human users to input the HIT in the application
- The HIP layer has to distinguish between IPv6s and HITs
  - an ORCHID prefix is used
  - pre-configured local mapping (e.g., in the /etc/hip/hosts) to lookup the IP
  - use of DNS to lookup the IP

API for native HIP applications. Overview of the design

- The standard Berkley API can be used with applications with LSI, HITs, or IPs
- Applications with HIP support can take full advantage of HIP function
- Using HITs in the interface may not be always possible in the future
  - OPP mode - the peer HIT is unknown for the application - some local identifier is needed
  - the HIT length of 128 bits could be insufficient if more attacks on hash function are found
  - multiple HITs can be used for a single HIP association when HIP multicast extensions are specified
  - session mobility or process migration - the binding between the identifier and the host identifier requires changes

API for native HIP applications. Overview of the design (cont.)

- Endpoint Descriptors (ED) are used to overcome limitations
  - EDs resemble - file similar to the UNIX file system interface
  - local significant of EDs file
  - ED - a small integer number
  - an index to the host identity database
- The Native API hides the transport port numbers from the application
  - the numbers are often changed by NATs in the network
  - new address family PF_SHIM to pass the ED
  - the API is placed between the application and transport layers
  - the shim layer is between the transport and network layers

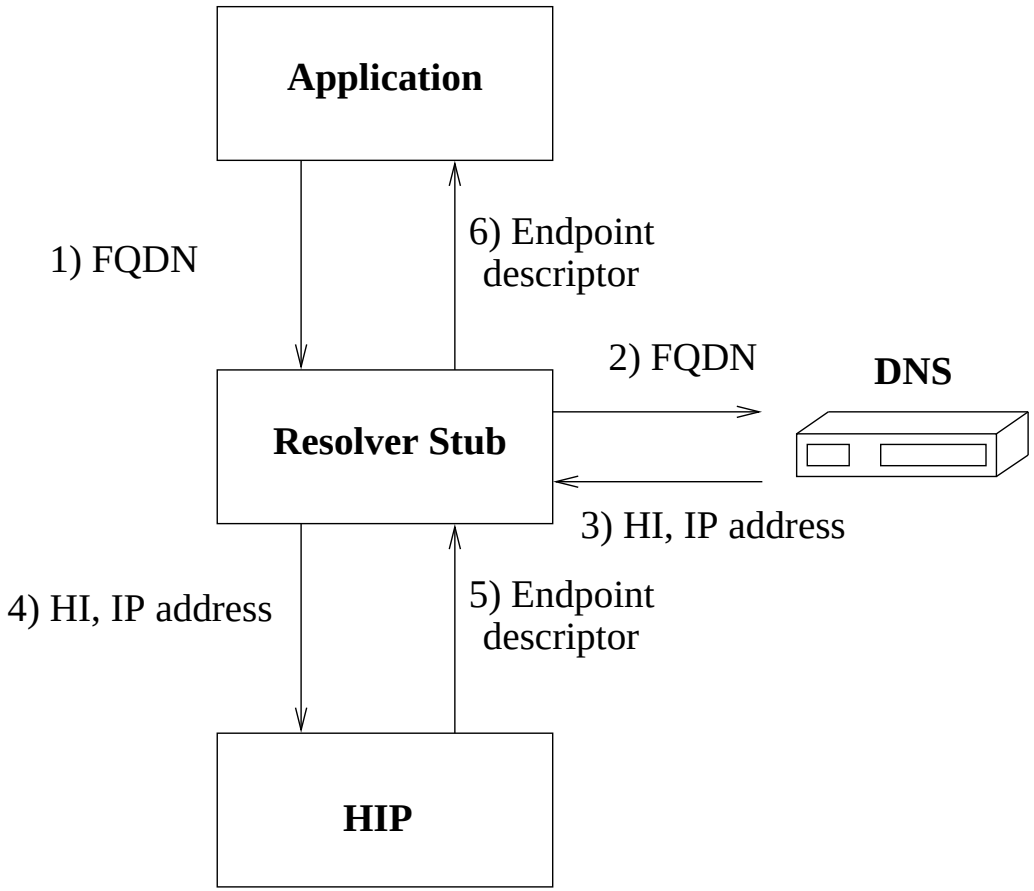
API for native HIP applications. Overview of the design (cont.)

- With Native API applications use human-friendly FQDNs
  - communication using HITs or IPs is possible
  - within the application - connections are identified by source ED, destination ED, source and destination port, the transport protocol number
  - the transport layer - the protocol number is fixed, the EDs are replaced by HI
  - the HIP shim layer - associations are identified by a pair of HIs
  - the network layer - the IPs for routing packets

API for native HIP applications. Overview of the design (cont.)

- PF_SHIM socket
  - bound to source and destination ED, source and destination port, the transport protocol
- Single HI can be associated with multiple EDs
  - binding can be established after the first connect call in the OPP mode
- A single ED can associate with several HIs - advanced scenarios

Native HIP API resolves a FQDN to an Endpoint Identifier



- the application call the resolver library
- FQDN as the argument
- the resolver stub - a query to a DNS
- set of HIs and locators available for the FQDN
- HIP layer stores the mapping
- HIP layer sends the ED to the resolver
- resolver returns the ED to the application

Interface specification

- The ED passing
  - *struct sockaddr_ed{*
    unsigned short int ed_family;
    in_port_t ed_port;
    sa_ed_t ed_val;}
  - ed_family - PF_SHIM
  - ed_val - the 4-byte value of ED; three macros: SHIM_ED_ANY, SHIM_ED_ANY_PUB, SHIM_ED_ANY_ANON
- The endpoint structure - application-specific endpoint identifiers
  - *struct endpoint {*
    se_length_t length;
    se_family_t family;};

Interface specification (cont.)

- The endpoint structure - application-specific endpoint identifiers
  - *struct endpoint_hip {*
    se_length_t length;
    se_family_t family;
    se_hip_flags_t flags;
    union {
    struct hip_host_id host_id;
    hit_t hit;
    } id; };
  - a HIP endpoint family field - EF_HI
  - flags field - SHIM_ENDPOINT_FLAG_ANON, SHIM_ENDPOINT_FLAG_PRIVATE

Interface specification (cont.)

- The HIP interface resolution function
  - *struct addrinfo {*
    int ai_flags; / e.g. AI_ED */*
    int ai_family; / e.g. PF_SHIM */*
    int ai_socktype; / e.g. SOCK_STREAM */*
    int ai_protocol; / usually just zero */*
    size_t ai_addrlen; / length of the endpoint */*
    *struct sockaddr *ai_addr; /* endpoint socket address */*
    *char *ai_canonname; /* canon. name of the host */*
    *struct addrinfo *ai_next; ;}; /* next endpoint */*
  - AI_ED is set to prevent compatibility problems
  - AI_ED_RVS - only the rendezvous service addresses are resolved

Interface specification (cont.)

- The HIP interface resolution function
  - AI_ED_NOLOCATORS is set to prevent the resolver from looking up the locators
  - AI_ED_ANY (PUB/ANON) - a single socket address is returned
- The getaddrinfo() function
  - *int getaddrinfo(const char *nodename,*
    *const char *servname,*
    *const struct addrinfo *hints,*
    *struct addrinfo **res)*
    *void free_addrinfo(struct addrinfo *res)*
  - zero on success
  - the socket addresses are placed to the pointer

Interface specification (cont.)

- Loading the endpoint identity from a file or string
  - *int shim_endpoint_pem_load(const char *filename, struct endpoint **endpoint)*
  - *int shim_endpoint_pem_load_str(const char *pem_str, struct endpoint **endpoint)*
  - a public or a private key is loaded from a source encoded with PEM
  - zero on success, non-zero on failure

Interface specification (cont.)

- The function getlocaled() - the ED creation for a local endpoint identity
  - *struct sockaddr_ed *getlocaled(const struct endpoint *endpoint,*
    *const char *servname,*
    *const struct addrinfo *addrs,*
    *const struct if_nameindex *ifaces, int flags)*
  - ifaces - a list of interfaces (NULL - interface selection is unlimited)
- The function getpeered() - the ED creation for an external identity
  - *struct sockaddr_ed *getpeered(const struct endpoint *endpoint,*
    *const char *servname,*
    *const struct addrinfo *addrs,*
    int flags)
  - servername - the port number copied to the sockaddr_ed
  - addrs - a pointer to the current IP of a peer or local host

Interface specification (cont.)

- The function getlocaledinfo() - a reverse mapping from EDs to HIs
  - *int getlocaledinfo(const struct sockaddr_ed *my_ed,*
    *struct endpoint **endpoint,*
    *struct addrinfo **addrs,*
    *struct if_nameindex **ifaces)*
  - my_ed - the ED that functions look up
  - the list of interfaces associated with the ED is returned
- The function getpeeredinfo() - a reverse mapping from EDs to HIs
  - *int getpeeredinfo(const struct sockaddr_ed *peer_ed,*
    *struct endpoint **endpoint,*
    *struct addrinfo **addrs)*
  - peer_ed - the ED that functions look up

Socket attributes

- Several HIP policy attributes can be set for a Native API socket
  - getsockopt() and setsockopt() calls are used
  - IPSEC_ESP_TRANSFORM sets the preferred ESP algorithm
  - IPSEC_SA_LIFETIME - a lifetime of IPsec SA in sec
  - SHIM_PROTOCOL is set to PF_HIP
  - SHIM_CHALLENGE_SIZE - the challenge size of the puzzle
  - SHIM_SHIM_TRANSFORM - the preferred SHIM algorithm
  - SHIM_DH_GROUP_IDS - the DH key group
  - SHIM_AF_FAMILY - the preferred locator family

Socket attributes (cont.)

- Generic attributes specified for SHIM protocols (HIP, SHIM6)
- The application handles the tasks
  - enabling the SHIM layer
  - management of locators
  - event triggers
  - configuring timeouts
  - hot-standby configuration
  - path exploration
  - delaying context establishment
  - locator queries

Socket attributes (cont.)

- The attributes defined for the SHIM layer
  - SHIM_ASSOCIATED - checking if the socket is using the SHIM layer (getsockopt())

    int optval;

    int optlen = sizeof(optval);

    getsockopt(fd, SOL_SHIM, SHIM_ASSOCIATED, &optval, &optlen);
  - SHIM_DONTSHIM defines if the SHIM layer provides multihoming for a socket (getsockopt() and setsockopt())
  - SHIM_HOT_STANDBY can be queried and set by the application to enable secondary location pair at the SHIM layer

Socket attributes (cont.)

- The attributes defined for the SHIM layer
  - SHIM_LOC_LOCAL_PREF queries or sets the local preferred locator for the socket *struct shim_locator* {
    hspace1cm uint8_t lc_family; / address family */*
    hspace1cm uint8_t lc_ifidx; / interface index */*
    hspace1cm uint8_t lc_flags; / flags */*
    hspace1cm uint8_t lc_preference; / preference value */*
    hspace1cm uint8_t lc_addr[16]; / locator */* }

Socket attributes (cont.)

- The attributes defined for the SHIM layer
  - SHIM_LOC_LOCAL_PREF set operation

```
struct shim_locator lc;
struct in6_addr ip6;
// ip6 = preferred IPv6 address
bzero(&lc, sizeof(shim_locator));
lc.lc_family = AF_INET6; /* IPv6 */
lc.lc_ifidx = 0;
lc.lc_flags = 0;
lc.lc_preference = 255;
memcpy(lc.lc_addr, &ip6, sizeof(in6_addr));
setsockopt(fd, SOL_SHIM, SHIM_LOC_LOCAL_PREF, &lc,
sizeof(optval));
```

Socket attributes (cont.)

- The attributes defined for the SHIM layer
  - SHIM_LOC_PEER_PREF queries or sets the locator for the peer
  - SHIM_LOC_LOCAL_RECV determines if the SHIM layer stores the destination locator of arriving packets

struct msghdr {
caddr_t msg_name; / optional address */*
u_int msg_namelen; / size of address */*
*struct iovec *msg_iov; /* scatter/gather array */*
u_int msg_iovlen; / # elements in msg_iov */*
caddr_t msg_control; / ancillary data, see below */*
u_int msg_controllen; / ancillary data buffer len */*
int msg_flags; / flags on received message */ };*

Socket attributes (cont.)

- The attributes defined for the SHIM layer
  - SHIM_LOC_PEER_RECV determines if the SHIM layer stores the source locator or arriving packets
  - SHIM_LOCLIST_LOCAL controls the list of local locators associated with Endpoint ID of the socket
  - SHIM_LOCLIST_PEER controls the list of remote locators associated with a socket
  - SHIM_APP_TIMEOUT determines the application timeout to detect a path failure

Socket attributes (cont.)

- The attributes defined for the SHIM layer
  - SHIM_PATHEXPLORE defines the interval and the maximum number of path exploration probes when a primary path fails

```
struct shim6_pathexplore pe;
pe.pe_probenum = 4; /* times */
pe.pe_keepaliveto = 10; /* seconds */
pe.pe_initprobeto = 500; /* milliseconds */
pe.pe_reserved = 0;
setsockopt(fd, SOL_SHIM, SHIM_PATHEXPLORE, &pe, sizeof(pe));
```

Socket attributes (cont.)

- The attributes defined for the SHIM layer
  - The shim_pathexplore
    struct shim_pathexplore {
    uint8_t pe_probenum; / # of initial probe */*
    uint8_t pe_keepaliveto; / Keepalive Timeout */*
    uint16_t pe_initprobeto; / Initial Probe Timeout */*
    uint32_t pe_reserved; / reserved */* };
  - SHIM_CONTEXT_DEFERRED_SETUP determines if the SHIM context is set up before or after initial packet exchange between the peer hosts

Integrating HIP with other protocols

Generalized HIP

- HIP implements the identifier-locator split
  - not all network engineers agree that a new namespace is needed
  - consensus - some level of indirections is needed
- Indirections in the Internet
  - Mobile IP, multi6, i3, FARA, TRIAD, IPNL, MOBIKE
- The HIP architecture is more general - other protocols can benefit from HIP
  - first change - requirement that HIP messages must carry certain parameters is relaxed
  - second change - a definition of profiles for the HIP BE is proposed
  - third change - type-length vector encoding is made extensible

Classification of proposals

- Upper-layer identifier
  - an identifier used at the application, session, transport layer
  - for compatibility identifiers should have an IPv4/v6 length of 32/128
  - can be routable at the existing IP layer
  - can be routable at the overlay layer on top of IP
  - for application supporting enhanced socket API identifiers of a format incompatible with IP addresses are possible
- Resolving the identifier
  - the upper-layer identifier is not the same as the destination address - resolution or binding to the proper address is required
  - binding can occur ”early” - in the sending host
  - binding can occur ”late” - in an overlay node in the network

Classification of proposals (cont.)

- Establishing context
  - the context establishment protocol - the mapping between upper-layer identifiers and locators
  - definition of common cryptographic keys and demultiplexing rules for data traffic
- Managing locators
  - the identity of the host typically remains stable
  - the active set of locators can change dynamically
  - proposals can support mobility, multihoming
  - a secure binding mechanism identifier and locator is requires to prevent locator hijacking

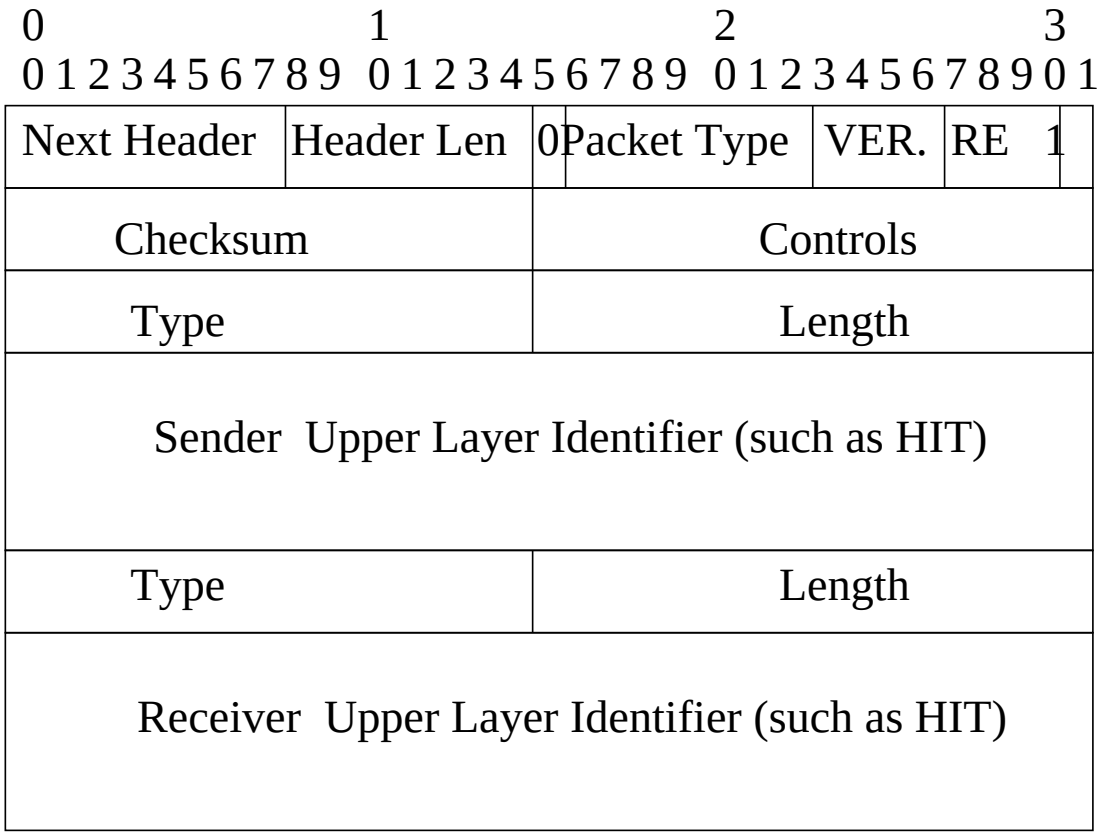
Classification of proposals (cont.)

- Tagging packets
  - a packet arrives to the receiver - some tag is needed to locate a correct context for processing the packet
  - HIP with ESP encapsulation - SPI serves as the tag

HIP implications

- Suggestion of the use of HIP is
  - general experimentation framework for multiple proposals towards separation of identifiers and locators
  - practical deployment of proposals is an open issue
- Support of the approach of generalizing HIP
  - the ESP encapsulation is removed from basic specification (IETF WG)
  - the ESP - one possible HIP encapsulation
  - other possible data packet encapsulation methods can include SRTP

Generalized HIP packet header



- a more general HIP header
- Type-Length encoding for representing identifiers
- HIT of a different size than 128 bits is allowed to use
- the use of other identifiers is possible

HIP implications (cont.)

- Implementation profile
  - existing HIP specification can be mapped to the generalized version
  - the HIP SIGMA-compliant DH exchange - the core of the HIP profile
  - an option of delaying the context establishment handshake is useful
  - the list of supported options in the R2 (TLV encoding, control bits)
  - one possible option - LHIP
- The base header of generalized HIP
  - source and destination identities encoded in TLV format - the only compulsory fields
  - each separate profile can include additional compulsory parameters

HIP implications (cont.)

- Practical use cases for extended HIP profile include NAT traversal and micromobility
- NAT traversal
  - NAT traversal extensions for HIP use UDP encapsulation
  - a common NAT-friendly profile can be defined for generalized HIP
- Micromobility
  - the base HIP mobility model - end-to-end mobility updates or a single RVS
  - efficient micromobility - a set of mobility anchor points is required
  - generalized HIP - micromobility profile common for MIP and base HIP

HIP implications (cont.)

- Securing the Mobile IP binding updates
  - possible application of generalized HIP
  - the present MIP specification - a binding management key through a return routability test from the home network to the mobile node
  - the key is necessary for secure context with arbitrary correspondent nodes
  - the mobile node changes IP - the key is invalid
  - generalized HIP - a mobile node can initiate BE using the home address as the source identity and correspondent node address as the destination identity
  - DH parameters are mandatory
  - the puzzle, nonce and signatures are optional

HIP implications (cont.)

- The shim6 IETF WG - multihoming solution
  - routable specially-formed IP as upper-layer identifiers
  - policies for selecting source and destination address
  - the context establishment protocol executed in parallel with data packets

The use of Session Initiation protocol

- The use of SIP infrastructure proxies
  - a rendezvous mechanism for HIP hosts
- The use of HIP to secure SIP traffic
- The extensions to SIP to exchange HITs between HIP hosts
- An active area of development in IETF - the use of HIP for the design P2PSIP

SIP as a rendezvous service

- SIP
  - an application-layer protocol
  - setting up sessions between users identified by URI
  - Uniform Resource Locators are similar to email addresses
  - often use for voice and video telephony, chat messaging
  - the UA locates the party and negotiate the session parameters
  - the data is delivered using a different protocol (RTP)
- SIP is mostly used for user-to-user communication
  - a rendezvous service is provided
  - a user with permanent identifier can be located in different places
  - the UA registers the user identifier at the current location to the SIP registrar server

SIP as a rendezvous service (cont.)

- The session initiation request
  - the SIP proxy server forwards the request to the UA
  - the SIP redirect server returns the current user location
  - the Initiator can directly contact the UA
- The SIP rendezvous mechanism
  - HIP hosts can locate each other
  - a SIP URI generating and publishing by HIP host
  - the use of SIP session to exchange the HITs and IPs
- The SIP session setup
  - request-reply messages negotiating the session parameters
  - INVITE message - IP, UDP ports, HITs

SIP as a rendezvous service (cont.)

- The use of HIP can be negotiated during the SIP session establishment
  - HIP hosts use SIP URI to locate peers with SIP servers - some hosts may not support HIP
  - the use of HIP can be not preferred (RTP voice traffic)
- RTP session
  - HIP is enabled - whole data packets are encrypted by ESP
  - the compressor and de-compressor are not able to process the headers
  - the use of SRTP is preferable (SRTP encrypts the data payload)

SIP as a rendezvous service (cont.)

- S/MIME provides the end-to-end security in SIP
  - the messages and some headers are encrypted and integrity-protected
  - the messages can be read only by the receiver
  - SIP self-signed certificates - opportunistic session security
- Leap-of-faith
  - the destination is assumed to be correct
  - establishment is completed - the identity is stored locally
  - verification of identity on subsequent connections
  - SIP and HIP are used together - execution at the SIP layer
  - the SIP layer is preferable for opportunistic security - once the user identity is verified it can be applied to several device identities

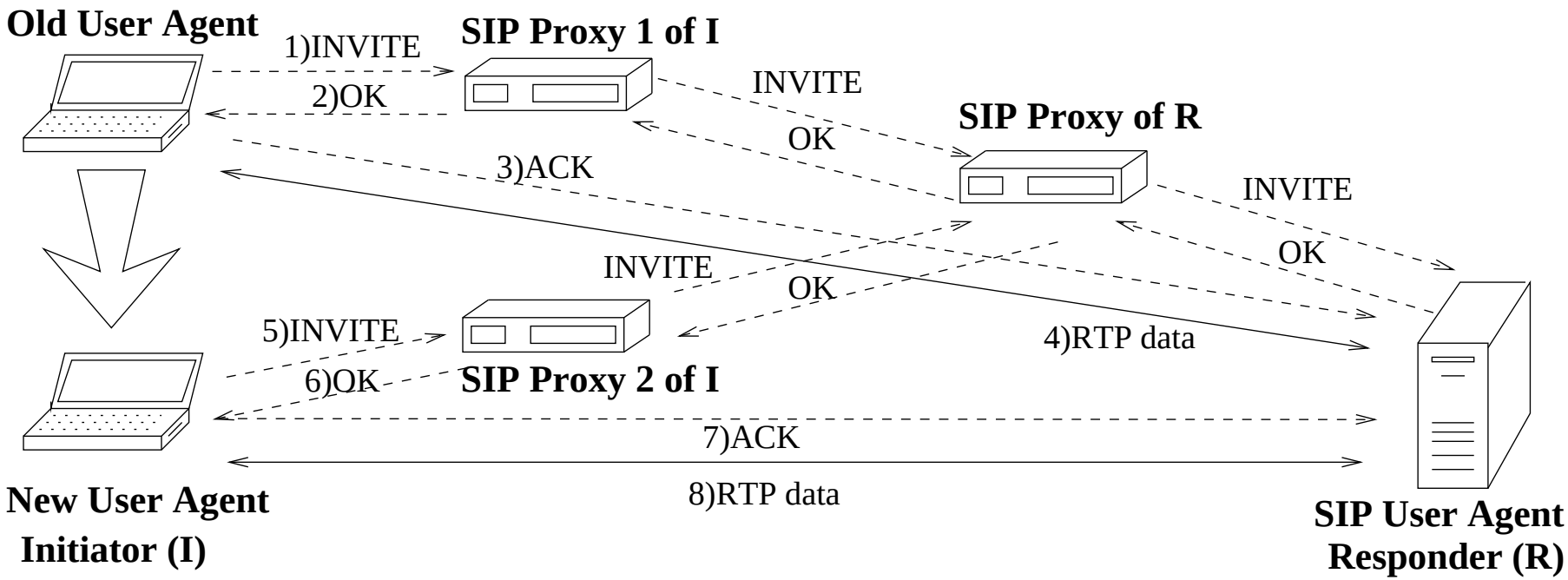
Complementary mobility

- HIP and SIP provide mobility on different layers
  - the joint use of protocols is not conflicting
  - the joint use is complementary
  - SIP provides mobility to users and sessions
  - a user moves from one device to another - the new host has a different host identifier
  - the HIP association cannot be moved from the old host
- Methods to overcome limitations
  - placing the host public/private keys on a portable storage device
  - delegation certificates

Complementary mobility (cont.)

- SIP supports session mobility
  - changing UA without loosing sessions
  - the change is implemented with the basic SIP mechanisms or Replaces
- Replaces
  - the user moves to a new device
  - Replaces are triggered by user - restoring existing session from the old device
  - the new UA send an INVITE message with a Replaces header
  - the header identifies session established with the old UA

Session mobility with SIP Replaces mechanism



- the new UA needs to re-establish a HIP association with the Responder UA and SIP proxies
- the old UA informs the peer UA that it is now reachable at the new UA
- SIP implements a REFER message (Globally Routable UA URI)
- the new UA receives a REFER - the new UA sends INVITE and ACK to the peer

Complementary mobility (cont.)

- Many of the SIP services use UDP
  - a datagram can be sent from any IP
- SIP mobility does not enable maintaining TCP session after the change of IP an a UA
  - the use of HIP - the SIP UA can keep all data connections during network mobility
  - the use of HIP - multihomed UA is enabled
- A SIP mechanism can handle the referrals
  - the mechanism can be used by HIP host instead of directly passing HITs
  - a SIP referral - the new UA can obtain peer HIT (DNS resolution)

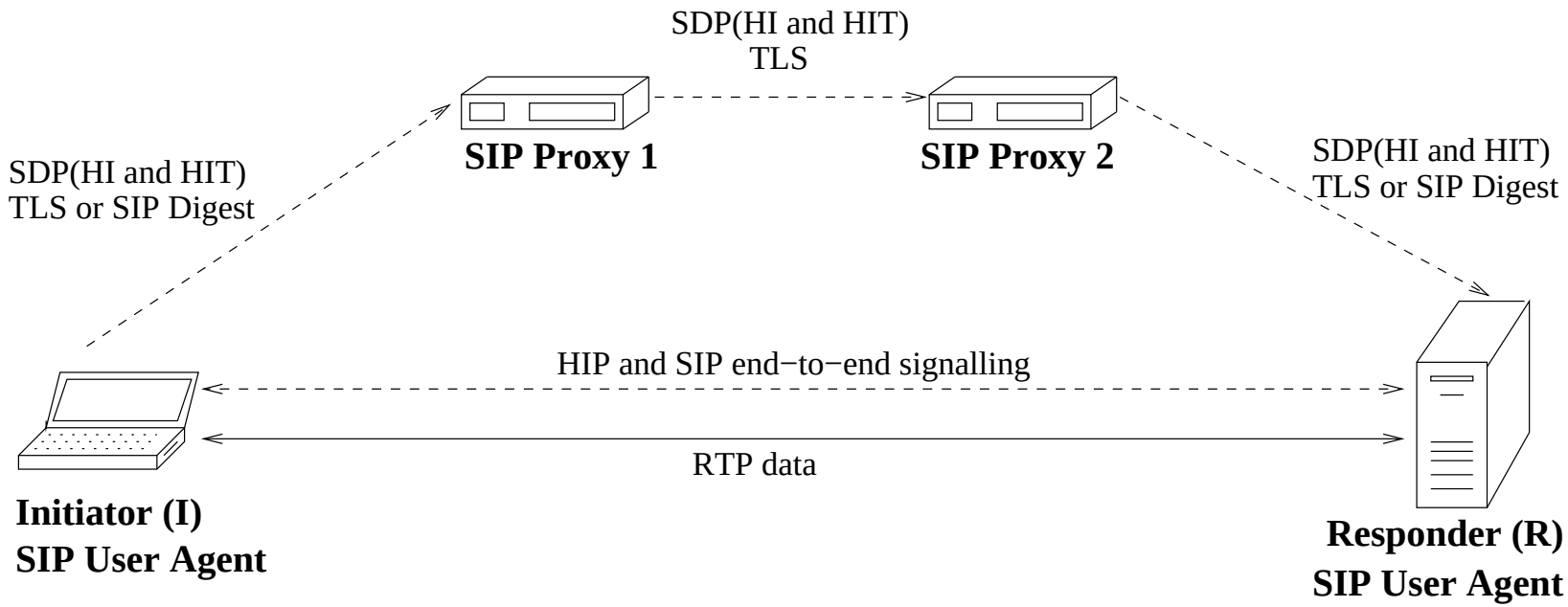
Complementary mobility (cont.)

- HIP can prevent exploiting the SIP to generate DoS
  - unsolicited traffic can be misused to send to a third party
  - the INVITE includes an IP that the Responder sends a reply to
  - the address of a victim can be put in the INVITE by malicious host
  - the ICMP ”Non Reachable” may not stop the attack (firewalls)
  - the use of HIP - the Responder can verify the IP of the Initiator

Securing SIP control traffic

- The use of HIP with SIP control messages
- The use of HIP for the data traffic after a SIP session establishment
- End-to-end data security can be always provided by HIP
- Hop-by-hop security in the SIP infrastructure

Combining HIP and SIP protocols



- separate data and control planes of a SIP session
- the control signaling is performed through a sequence of SIP proxies
- the data flows directly between communication hosts
- hop-by-hop communication is secured by TLS
- Digest authentication method is used

Securing SIP control traffic (cont.)

- The use of IPsec between proxies
  - a possible alternative to TLS
  - the application can not be certain that a given message was indeed sent over a protected connection
- With HIP IPsec option is more attractive for SIP
  - the applications certain that a message is delivered protected
  - the use of HIT implies that the message is secured by HIP
  - HIP supports certificate authentication - HIP can be used for agent-proxy and proxy-proxy hops
  - a CER packet can be sent after R1

Securing SIP control traffic (cont.)

- HIP can be useful for UA mobility
  - UA mobility - informing all associated UAs about new location (a hop-by-hop approach)
  - an UPDATE from the moving UA is relayed by several proxies
  - HIP is used - a mobile UA can inform its own proxy (HIP UPDATE)
  - an approach can reduce the load on other SIP proxies
- A modeling study of a hybrid mobility solution using HIP and SIP
  - the handover delay, signaling overhead of plain SIP, HIP/SIP, plain Mobile IP, Mobile IP/SIP were compared
  - the handover duration is dominated by a large DHCP delay
  - the plain SIP delay is highest
  - the signaling overhead of the HIP/SIP solution is found to be half that of plain SIP and a third that of Mobile IP/SIP

Securing SIP control traffic (cont.)

- HIP can help to resolve the problem of simultaneous mobility
  - both UAs move at the same time - SIP UPDATE are not useful (invalid destination)
  - HIP is used - UA can update its proxy with the current location information
  - Replaces mechanism can also resolve the problem of simultaneous mobility (more complex and less efficient)
- HIP can be useful for the instance identifier for UAs
  - Address of Record identifies a user
  - the instance identifier gives the location of a UA
  - a forking proxy uses instant identifiers to probe several UAs

Securing SIP control traffic (cont.)

- HIP can help to resolve the problem of simultaneous mobility
  - the use of HITs as instance identifiers
  - prevention of registration hijacking
  - the last-hop proxy can authenticate the UA just before relaying a session initiation message
  - authentication is currently implemented by maintaining a TLS connection between UA and the proxy
- HITs can not be used
  - software UAs that change a location device
  - software UAs that run simultaneously on the same device
  - several UAs running on the same device would not get unique identifiers from the same HIT

Securing SIP control traffic (cont.)

- Several types of instance identifier can be used by SIP
  - hardware UAs bound to the same host can use HITs as instance identifiers
  - mobile software UAs can use the standard SIP identifiers
- Possible synergies between HIP and SIP for middlebox traversal
  - Interactive Connectivity Establishment for traversing legacy NATs
  - SIP traffic benefits from using HIP to traverse HIP-aware NAT - additional security by verifying the HI of hosts

Session Description Protocol extensions

- A prior secure exchange of HI is required for HIP association establishment
  - the extensions to the SDP to be used to carry HI
- The initial SIP exchange
  - TLS is used to protect the exchange of HIs
  - control and data packets follow different paths - additional security is useful
  - an attacker must be able to intercept both signaling and data traffic
  - successful exchange of HIs requires only the integrity
  - TLS provides the integrity and confidentiality of SDP messages

Session Description Protocol extensions (cont.)

- The HI can be carried in SDP parameters ”k” or ”a”
- The structure of ”k” parameter
  - $k = \langle method \rangle : \langle encryption\ key \rangle$
  - $k = host - identity : \langle Host\ Identity \rangle$
  - $k = host - identity - tag : \langle Host\ Identity\ Tag \rangle$
- The ”a” parameter can be used as an alternative to ”k”
  - $a = \langle attribute \rangle : \langle value \rangle.$
  - $a = key - mgmt : host - identity \langle Host\ Identity \rangle$
  - $a = key - mgmt : host - identity - tag \langle Host\ Identity\ Tag \rangle$

Session Description Protocol extensions (cont.)

- The SDP only deals with media streams
  - several media streams can be involved with a single SIP session
  - one device can be used for receiving video, another for receiving an audio stream
  - several HI (one per device) need to be exchanged
  - SIP signaling header could carry HI or HIT
  - carrying in the ”Contact” field protected by S/MIME
- The SIP integrity mechanism
  - the delivery of HI form the Initiator to the Responder is protected
  - Response Identity Mechanism is under development by the SIP community in the IETF

Session Description Protocol extensions (cont.)

- Several mechanisms can be used to assert the identity of the Responder
  - the Responder replies with SDP message containing HI
  - an UPDATE message can be immediately sent - confirmation of the information sent in SDP
  - one additional RTT is required - drawback
- SIP mechanism based on hop-by-hop TLS security
  - all SIP proxies on the path are trusted - the authenticity of the Responder HI is provided
  - a MITM - replacing the Responder identity when relaying an SDP message
  - S/MIME can provide end-to-end integrity; not deployed for SIP

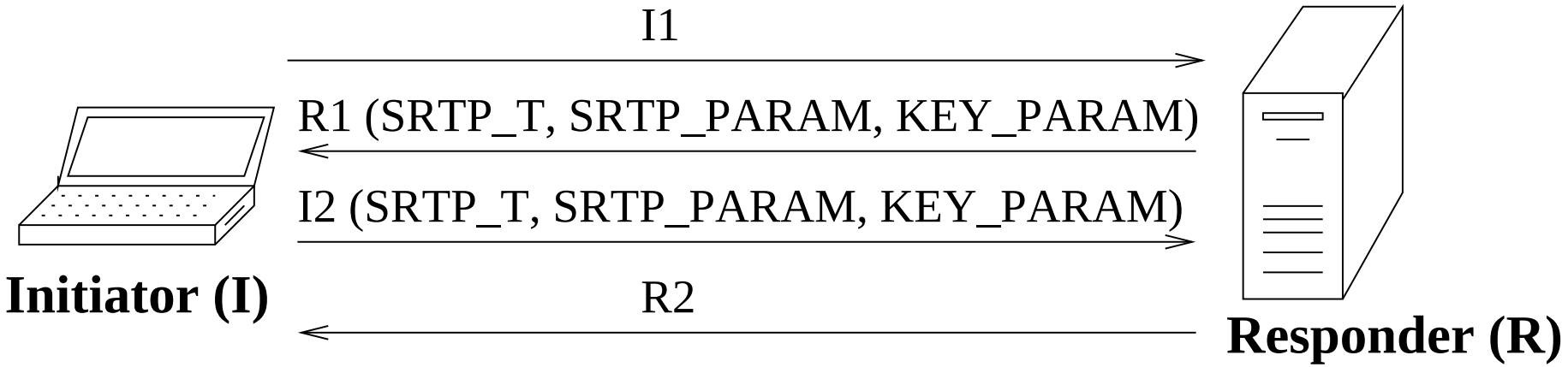
Encapsulating HIP data using SRTP

- An alternative experimental encapsulation - Secure Real-Time Protocol
  - security properties of encapsulation are similar to IPsec ESP
  - header compression is enabled
  - header compression is important for real-time application transmitting small packets over a slow link
- SRTP - an application protocol
  - operating over UDP
  - use together with SIP - a signaling mechanism
- SRTP - an extension profile for Real-Time Protocol
  - integrity, encryption is provided
  - protection against replay attacks specifically for real-time data

Encapsulating HIP data using SRTP (cont.)

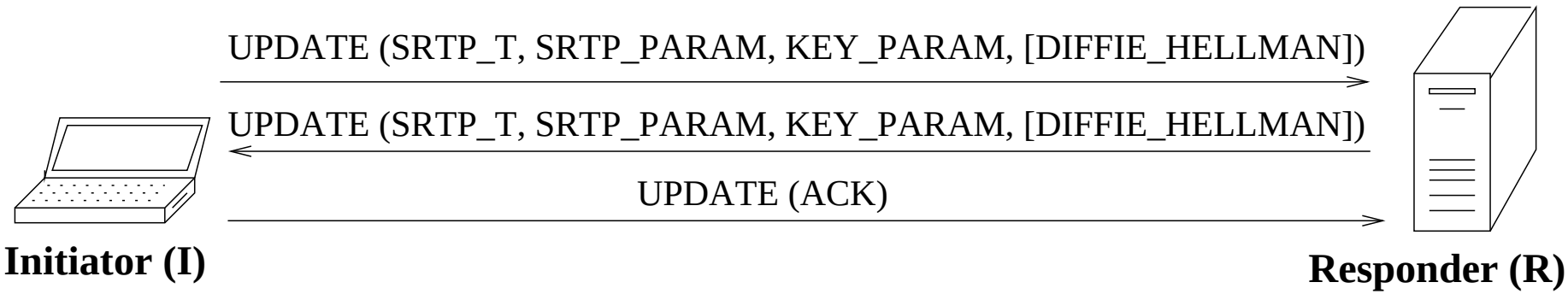
- SRTP protection
  - a mechanism to exchange keys is not included
  - protocol parameters necessary to establish an SRTP security context is not included
- HIP BE extension to carry SRTP-specific parameters
  - R1 - a list of available transforms at the Responder
  - I2 - selection of one of the transforms by the Initiator

HIP base exchange with SRTP



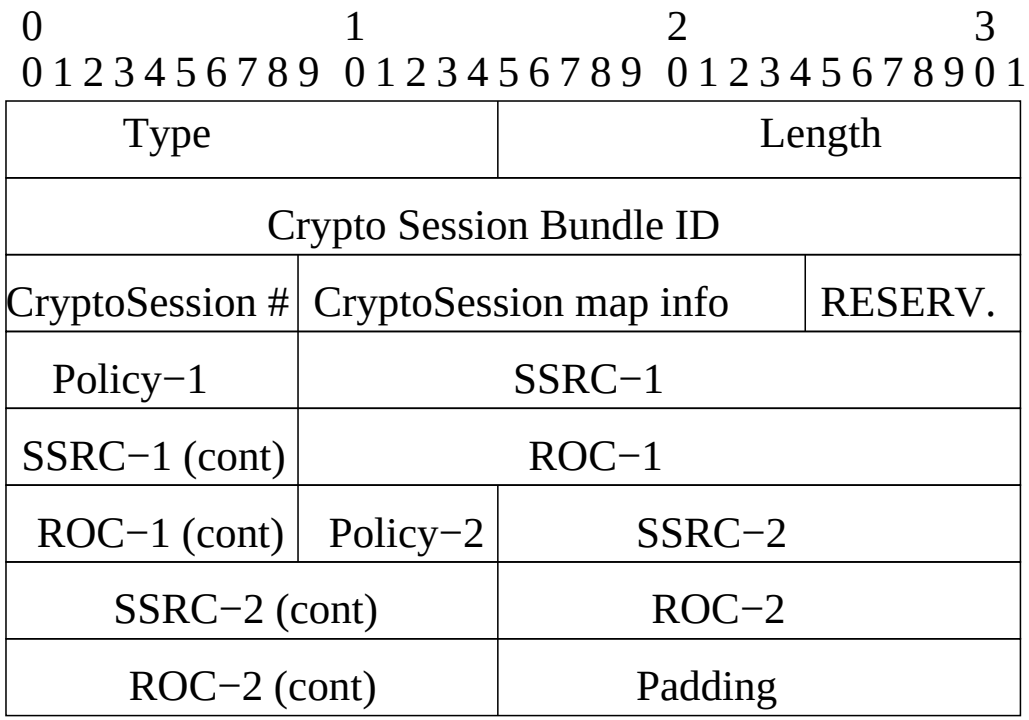
- R1 and I2 are extended with several SRTP-specific parameters
- SRTP_T - a timestamp to prevent replay attacks
- SRTP_PARAM - the mapping between Cryptographic Sessions and Synchronization sources
- KEY_PARAM contains SRTP_RAND (random string for key generation), SRTP_SP (list of security algorithms and transforms), SRTP_MKI (master key)
- SRTP_KEY (R1) includes a transform selected by the Initiator
- the Responder generates the session key from the master key and salt

HIP mobility update with SRTP



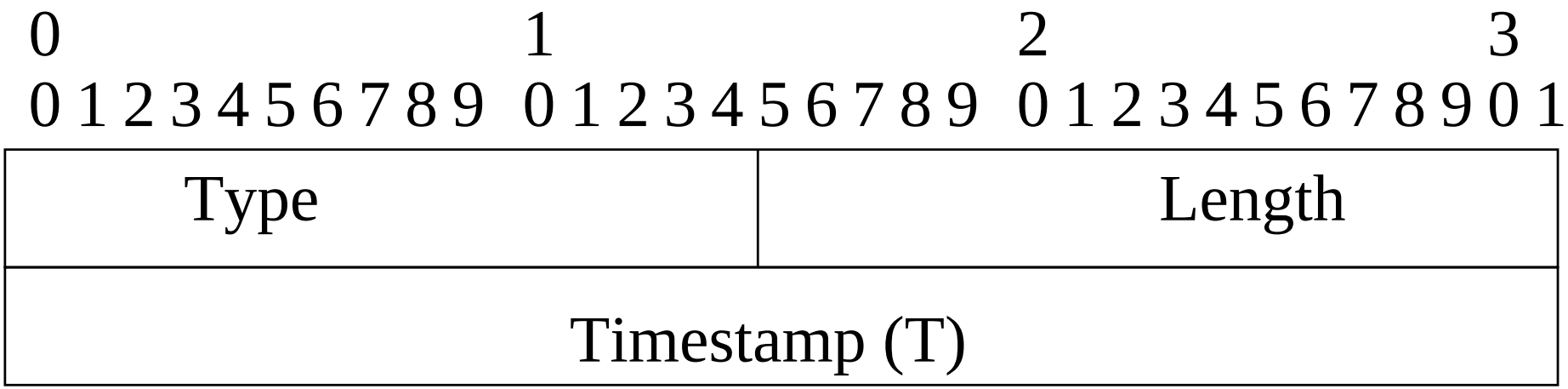
- initiating rekeying - new SRTP and a DH parameters for generating a new key
- using of the current master key can be continued
- generating only a new session key (the DH parameters is not sent)
- the rekeying process is protected by the retransmission mechanism
- the second UPDATE - a return routability check to the source of the first UPDATE
- the third UPDATE - an acknowledgment of successful rekeying

The HIP mapping parameter for SRTP



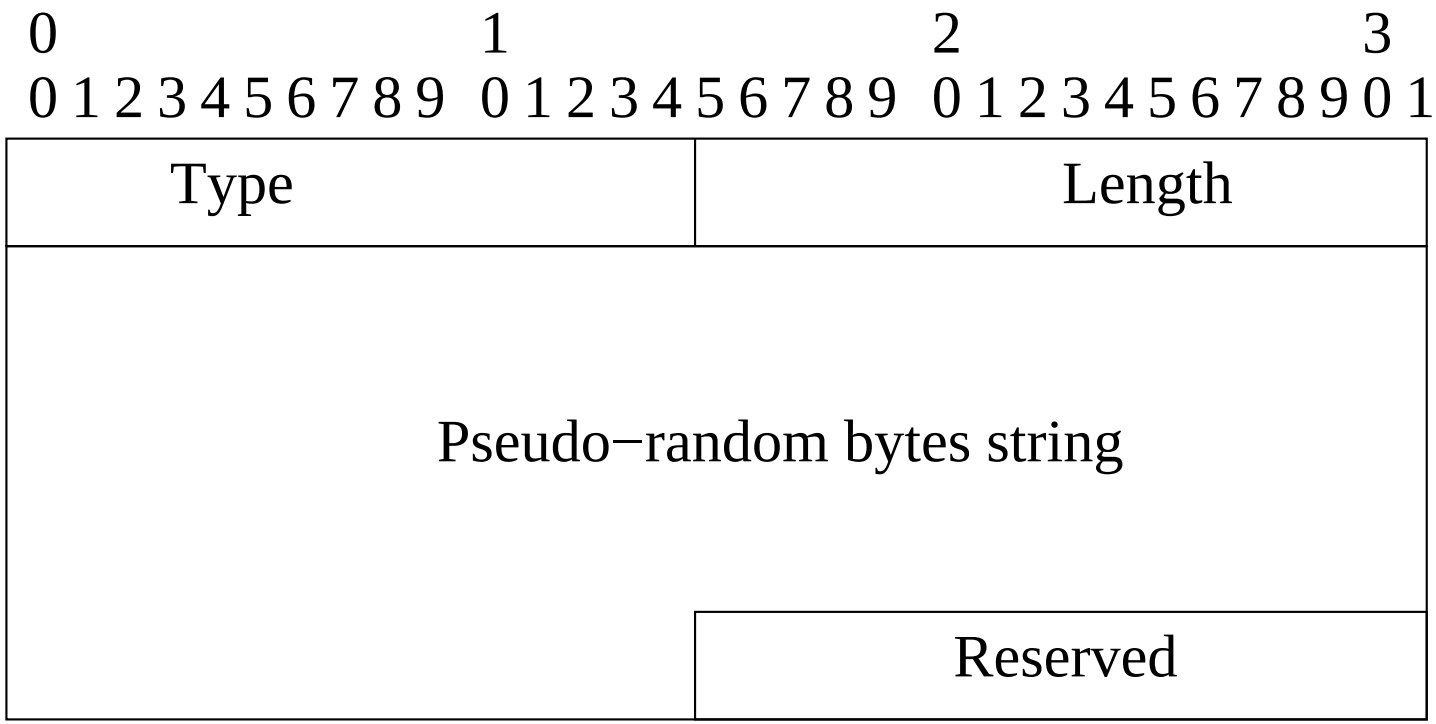
- SRTP_PARAM maps Crypto Session (CS) to SRTP association
- CSB - identifier randomly generated by the Responder
- the maximum number of session is 255, the type of 40000

The HIP timestamp parameter for SRTP



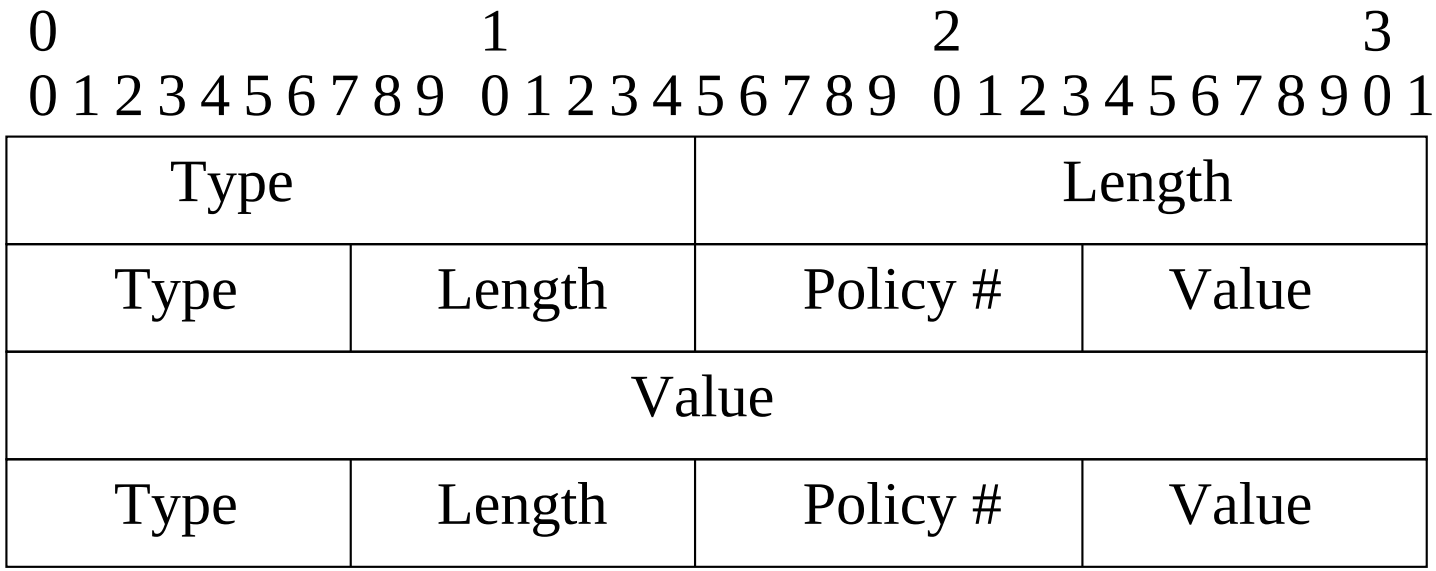
- SRTP_T contains a timestamp
- fixed length of 4 bytes, the type of 40001

HIP pseudo-random bitstring parameter for SRTP



- SRTP_RAND contains a master salt for generating a key
- the length of the pseudo-random bit string is 112 bits
- the length of 14 bytes, the type of 40002

The HIP policy parameter for SRTP

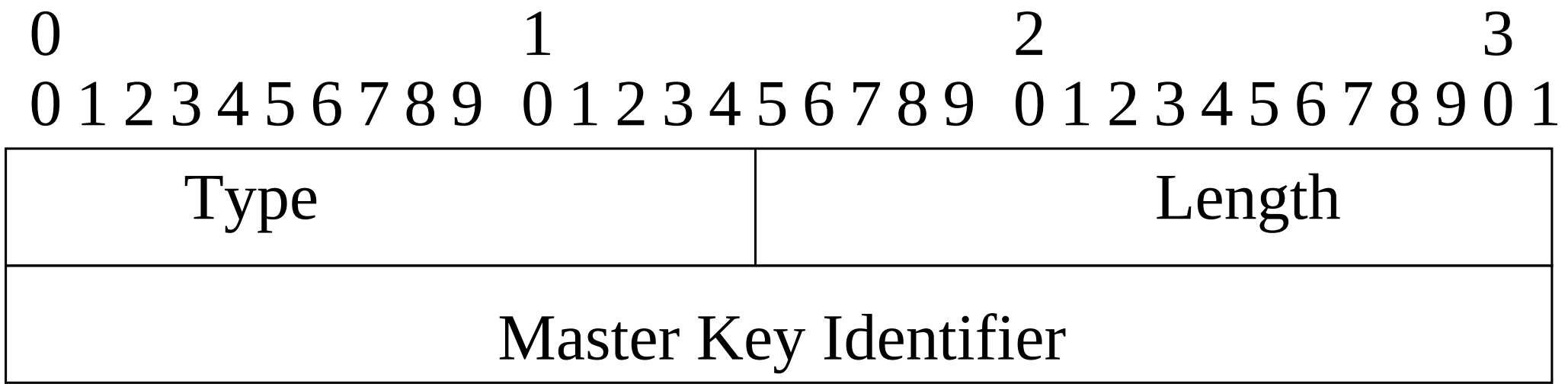


- SRTP_SP is composed of several security policies
- each policy defines cryptographic transforms and cypher algorithms
- the type of 40003
- the policies are concatenated using the TLV encoding
- the maximum number of policies is 255, the actual number is limited by the R1/I2 packet size

Parameters for SRTP security policies

Type	Length	Name	Value
1	1	Encryption transforms	NULL, AES-CM, AES-F8
2	2	The length of encryption session key	128
3	1	Authentication transforms	NULL, HMAC-SHA-1
4	2	The length of authentication session key	160
5	2	The length of a tag	80
6	4	The SRTP prefix length	Variable (default 0)
7	1	Pseudo Random Function for key generation	NULL, AES-CM
8	8	The rate of key generation	Variable (default 0)
9	8	Maximum packet lifetime for SRTP	Variable
10	8	Maximum packet lifetime for SRTCP	Variable
11	1	Forward Error Correction	2 bits

The HIP master key identifier parameter for SRTP



- SRTP_MKI determines the master key and its salt of the session key (an optional parameter)
- the type of 40004

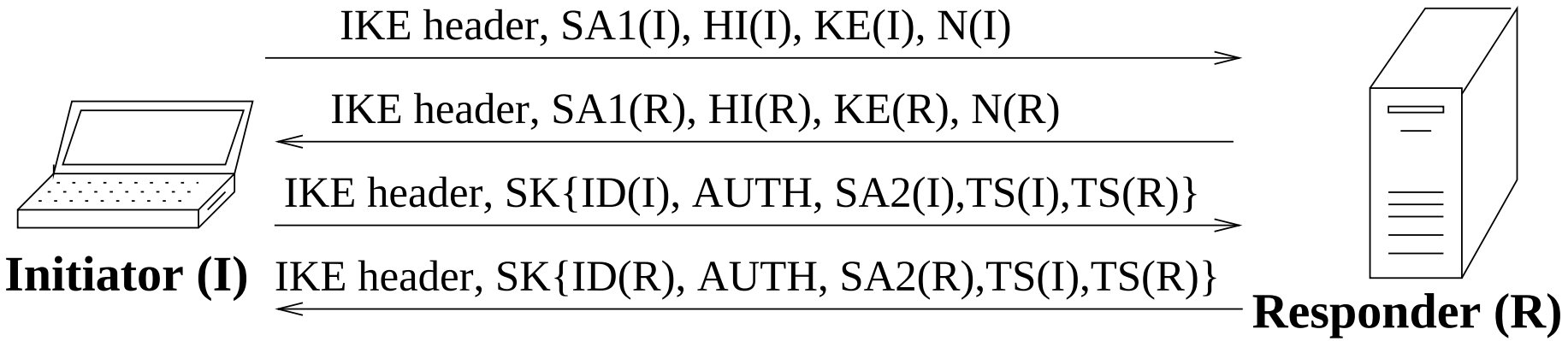
Encapsulating HIP data using SRTP (cont.)

- The HIP NOTIFY is extended - two error types
  - NO_SRTP_PROPOSAL_CHOSEN
  - INVALID_SRTP_TRANSFORM_CHOSEN
- The processing of SRTP-encapsulated packets (RFC 3711)
  - SRTP_PARAM, SRTP_T, SRTP_RAND, SRTP_SP, SRTP_MKI are protected with HMAC and a HIP signature
- The HIP BE establishes the SRTP association
  - processing a data stream - the interface between the HIP daemon and implementation is necessary
  - the SRTP processing code is integrated to a user application
  - the application needs to receive the session key form the HIP daemon

Replacing HIP base exchange with IKEv2

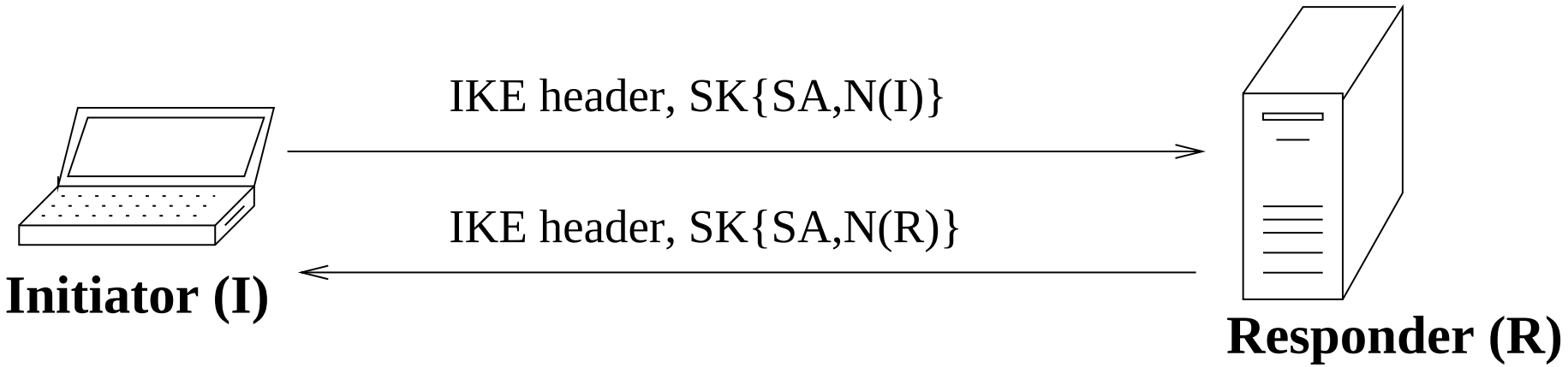
- The use of IKEv2 to set up ESP SA for HIP
  - IKE-H may provide better security
  - benefit from a larger implementation and deployment of IKEv2
- The IKE-H extension
  - the exchange of keying material
  - the exchange of mutual authentication
  - a readdressing exchange in case of mobility

The HIP base exchange with IKEv2



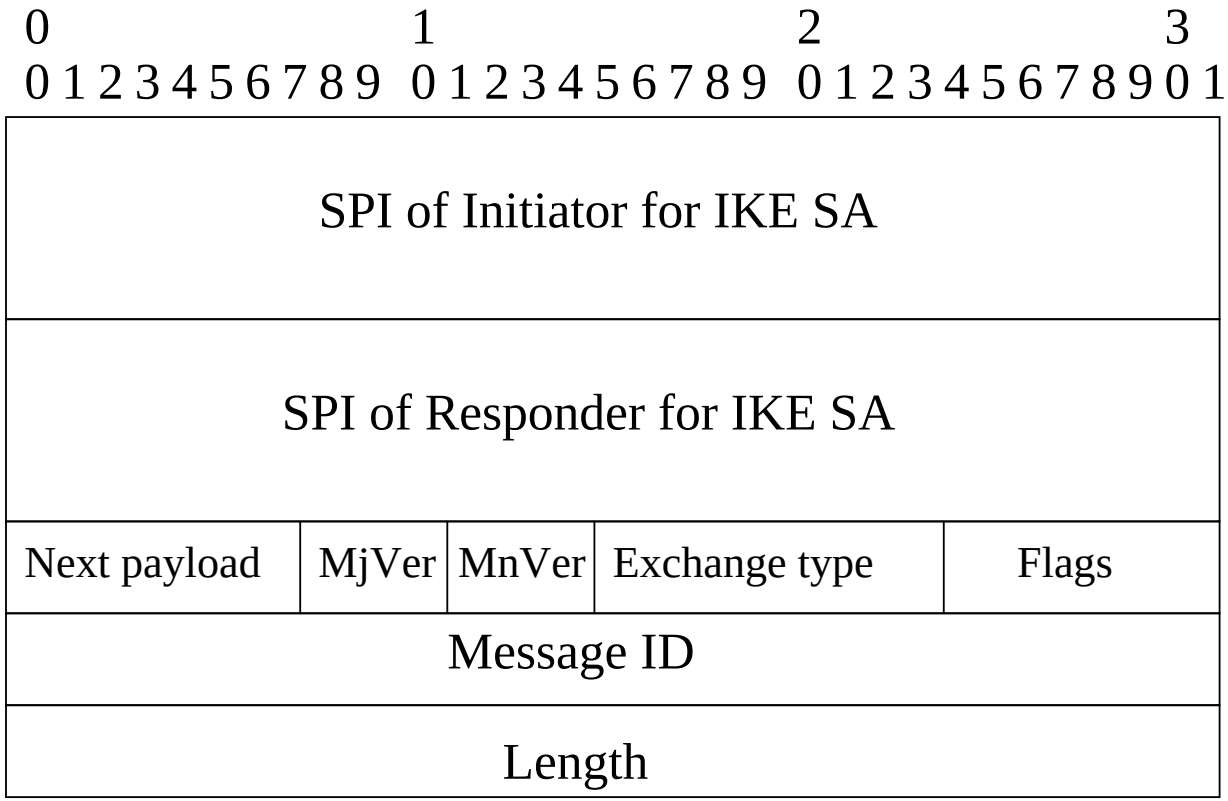
- four-message exchange
- the IKE_SA_INIT exchange (the first two messages) - HITs, nonces, transforms, DH
- the IKE_AUTH exchange (the last two messages) - a child SA is established
- ID(I) and ID(R) are identifiers of the Initiator and Responder
- AUTH - data for host authentication and the SA parameter
- TS(I) and TS(R) - the traffic selectors

The HIP rekeying with IKEv2



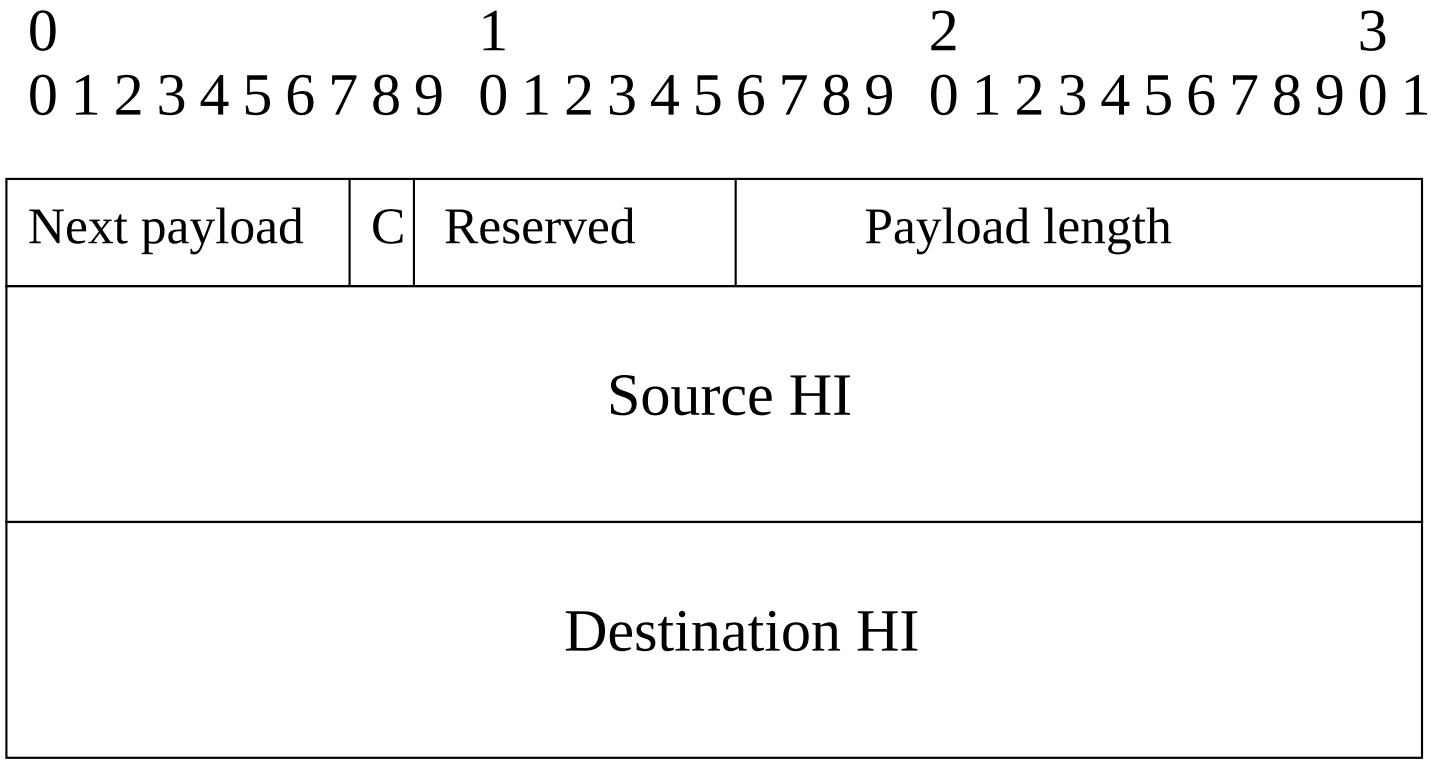
- a child exchange is created by exchanging new nonces and SA parameters
- a host identity can be changed during the process

The IKEv2 header with HIP extensions



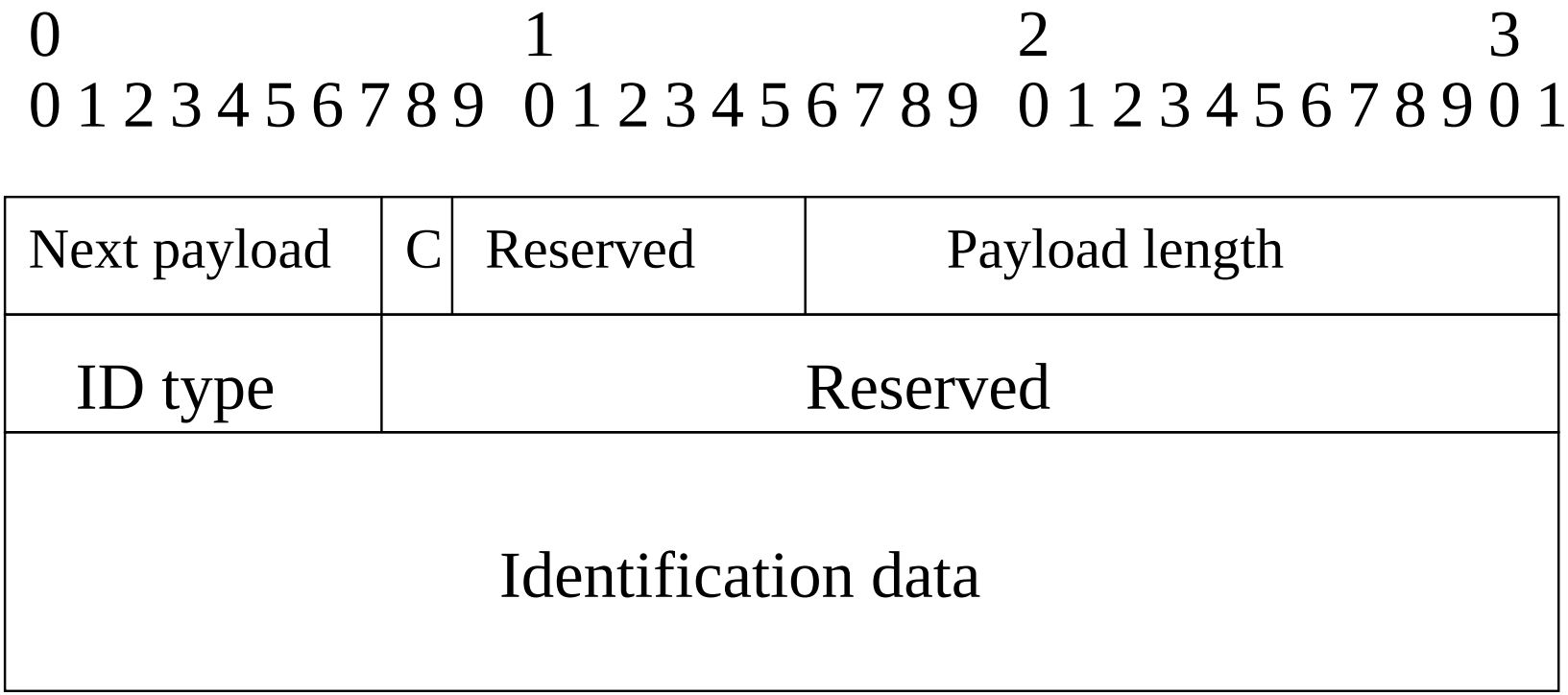
- H flag indicates the support of IKE-H
- MjVer and MnVer are major and minor versions of IKE implementation
- Exchange type - type of IKE exchange (IKE_SA_INIT, IKE_AUTH, CREATE_CHILD_SA)

The IKEv2 parameter for carrying a Host Identity



- HI payload for IKE-H
- the parameter is sent in the IKE_SA_INIT and during the rekeying
- the exchanged HITs - identifiers to SA for IKE-H

The IKEv2 parameter for carrying identification data



- the format of HIP-specific payload
- the Reserved field is set to zero on transmission and ignored on reception
- Identification data - a HIT

Replacing HIP base exchange with IKEv2 (cont.)

- The IP addresses (not HITs) are used with IKE-H
  - the IPsec security policies - determination of packets that are sent using IPsec
  - an outgoing packet requires IPsec processing - IKE-H daemon looks up a HIT corresponding to the destination IP
  - the HIT is not found - a new IKE-H is initiated
  - the HIT is found - the packet is encrypted and integrity protected
  - the HITs are replaced with IPs before passing the packet to the application

Mobile IP and HIP

- The pros and cons of using HIP, i3, Mobile IPv6, Hi3 are compared
- Mobility
  - simultaneous mobility and multihoming support
  - fault-tolerance
  - inverse mapping support
- Security
  - Denial-of-Service (DoS) resistance
  - end-to-end security and privacy
  - accountability and trust model
- Efficiency
  - routing efficiency
  - infrastructure cost

Mobile IP and HIP (cont.)

- The basic HIP
  - efficient and secure end-to-end connectivity
  - the HITs and IPs are known - additional infrastructure is not needed
  - no issues with infrastructure cost, accountability, trust, fault-tolerance
  - limited DoS protection - puzzle computation
  - mobility of one end point at a time is supported
  - no support of the reverse mapping
- HIP with a RVS
  - mobility of both end points
  - accountability and the trust model are preserved

Mobile IP and HIP (cont.)

- The advantages of i3
  - better DoS protection
  - support of simultaneous mobility
  - higher fault-tolerance when using a DHT
- The disadvantages of i3
  - reliance on an extensive infrastructure
  - server scalability
  - use of UDP
  - lack of traffic encryption
  - an overlay network on top of the Chord DHT - complex infrastructure
  - relaying all control traffic is burdensome

Mobile IP and HIP (cont.)

- The i3 could run on both UDP and TCP
  - only UDP is supported currently
  - maintaining many TCP connection is challenging
  - UDP packets do not traverse firewalls and NATs
- The basic i3 system
  - data encryption is not provided
  - lack of encryption and privacy for control packets
- A public i3 infrastructure
  - the possibility of malicious or misbehaving i3 nodes
  - a lack of trust of arbitrary i3 servers

Mobile IP and HIP (cont.)

- Secure-i3
  - constraints on the structure of triggers
  - prevention from misuse of triggers by third parties
  - diagnosing problems is challenging
- A combination approach of HIP and i3 - Hi3
  - protection from DoS
  - double-jump problem solving
  - an initial rendezvous service providing
  - hiding IPs until HIP authentication - additional protection from DoS
  - simultaneous mobility is supported - updated packets are sent via i3

Mobile IP and HIP (cont.)

- Hi3 inherits the challenges of the i3
  - trust
  - accountability
  - cost issues
- Basic Mobile IP
  - any DoS protection mechanisms are not provided
  - end-to-end security is not supported
  - no support of multihoming and co-existence of IPv4 and IPv6
  - end-to-end security can be added (IKE)
  - a business relationship between mobile node and home agent - clear trust and accountability model

Mobile IP and HIP (cont.)

- Mobile IP
  - proposals for supporting mobility between IPv4 and IPv6
  - serious work to address end-host multihoming with Mobile IP
  - a fault-tolerance point-of-view - home agent forms a single point of failure
  - an efficiency point-of-view - extra mobility-related headers
- Mobile IP and Hi3
  - capacity requirements are at the same level
  - administrative models are different

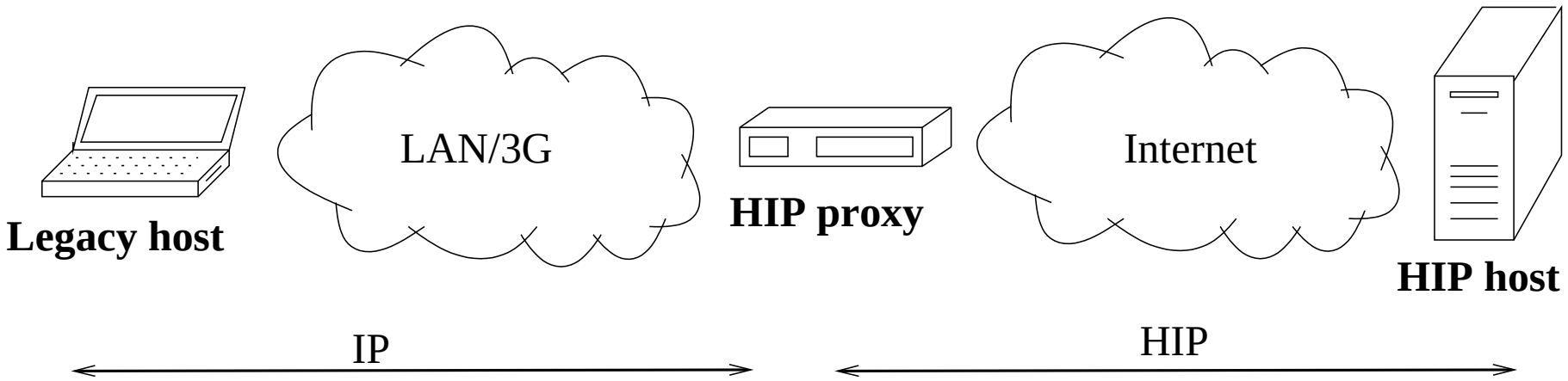
HIP proxy for legacy hosts

- Both communication hosts are required by HIP to support HIP
- Incremental deployment of HIP
  - most hosts remains HIP-unaware
- A HIP proxy can help to obtain HIP benefits
- The HIP proxy should handle four possible scenarios
  - a legacy mobile host contacting a legacy Internet host
  - a legacy mobile host contacting a HIP Internet host
  - a HIP mobile host contacting a legacy Internet host
  - a HIP mobile host contacting a HIP Internet host

Legacy mobile hosts

- The first scenario
  - a HIP proxy helps legacy hosts in a local to gain HIP benefits
  - example of deployment - an intranet of a company of 3G networks
  - benefiting from HIP without upgrading all local host
  - smooth transition to end-to-end HIP is possible

A proxy enables HIP benefits for legacy mobile hosts



Legacy mobile hosts (cont.)

- The advantage of using a HIP proxy
  - traffic protection in the public Internet
  - a great number of security risks in the public Internet
- Local or 3G networks are relatively secure
  - LAN - Ethernet switching technology
  - 3G - own security mechanism; own mobility management
- A mobile host attempts to communicate with HIP-aware host
  - a public/private key pair generation
  - the HIP proxy notices a DNS query
  - in the reply the proxy sees if any HITs are presented in the DNS record

Legacy mobile hosts (cont.)

- The AAAA record in the reply
  - HITs with an ORCHIDs prefix can be contained
  - a HIP-specific record can be present - host public key and a HIT
- No HITs are presented in the reply or proxy does not have any internal HIT-IP mapping
  - conclusion of the proxy - host is HIP-unaware
  - proxy can attempt to establish an OPP HIP
  - forwarding packet without further processing

Legacy mobile hosts (cont.)

- The HIP proxy receives a plain IP packet from a mobile host
  - checking an existence of available HITs for the destination IP
  - no available HITs - the packet is forwarded as is
  - HITs are available - the proxy looks up existing SA
  - no SA are found - internal packet caching
  - a BE is initiated to the peer
- A complicated scenario
  - the DNS record - an IP of the RVS together with peer's HIT
  - a legacy mobile host will send packets to the IP of the RVS

Legacy mobile hosts (cont.)

- Only a single connection to the RVS exists
  - the proxy can successfully forward ESP packets with HIT of the peer
- Several mobile hosts attempt to contact the peer using the same RVS
  - an ambiguous mapping
  - several destination HITs correspond to a single IP of the RVS
  - the proxy can drop or send the packets to all hosts - the hope that incorrect packets are rejected by transport or application protocol
- Resolving the conflict
  - modifying DNS reply that contains the HIT and RVS IP for hosts
  - acting as a NAT - insert a private IPv4 instead of a public IPv4 RVS; insert the destination HIT instead of IPv6

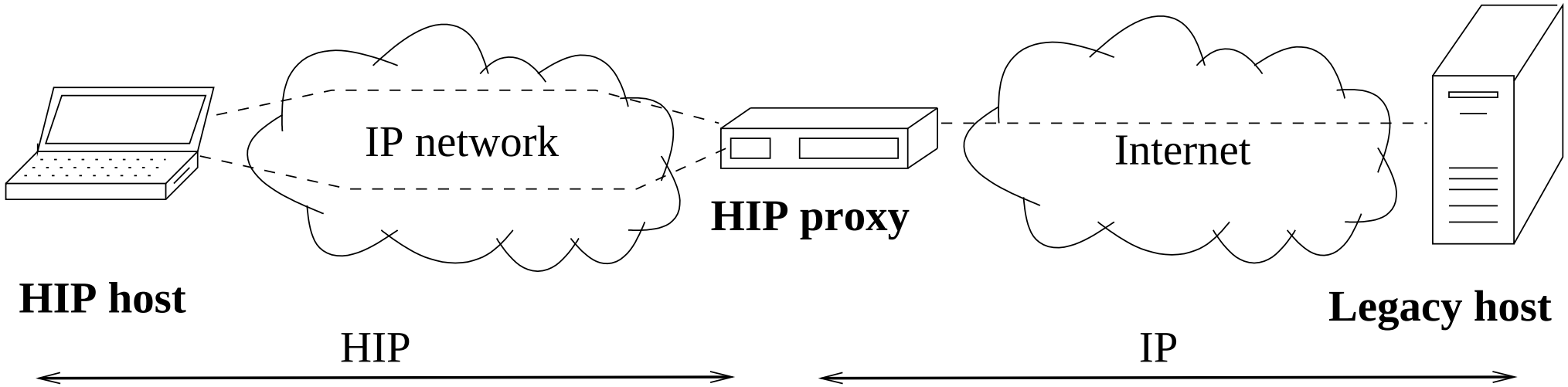
Legacy mobile hosts (cont.)

- The 3G network
  - own access control
  - encryption and mobility mechanisms
  - the mobile hosts are mostly lightweight devices
- Deploying HIP on lightweight devices
  - limited hardware capabilities
  - difficulty of performing a software update
  - lack of HIP implementations for Symbian
- Gateway GPRS Support Node - a natural place to implement the HIP proxy
- The proxy prototype - FreeBSD system OS

Legacy correspondent hosts

- The advantage of using a proxy
  - mobility and multihoming for the mobile host is provided
  - traffic protection on the path between the mobile hosts and the proxy
- Example
  - a mobile host uses WLAN for Internet access
  - the mobile host can connect to own HIP proxy in its home network
  - all air traffic is encrypted
  - moving between different WLAN networks is possible
- To gain some HIP benefits a mobile user needs a HIP proxy to communicate with the rest of world

A proxy enables HIP benefits for legacy remote hosts



Legacy correspondent hosts (cont.)

- An alternative to developing a HIP-specific proxy
  - adoption of a generic proxy
- An Overlay Convergence Architecture for Legacy Applications
  - running on multiple platforms (Windows, Mac OS, Linux)
  - a capturing mechanism for packets that a legacy application sends or receives
  - a generic overlay-independent sublayer - packet capturing and encapsulation
  - adaptations modules for several overlay architectures (i3 and HIP)
  - open-source - http://ocala.cs.berkley.edu/
  - the drawback - must be run both on the HIP and on the proxy host

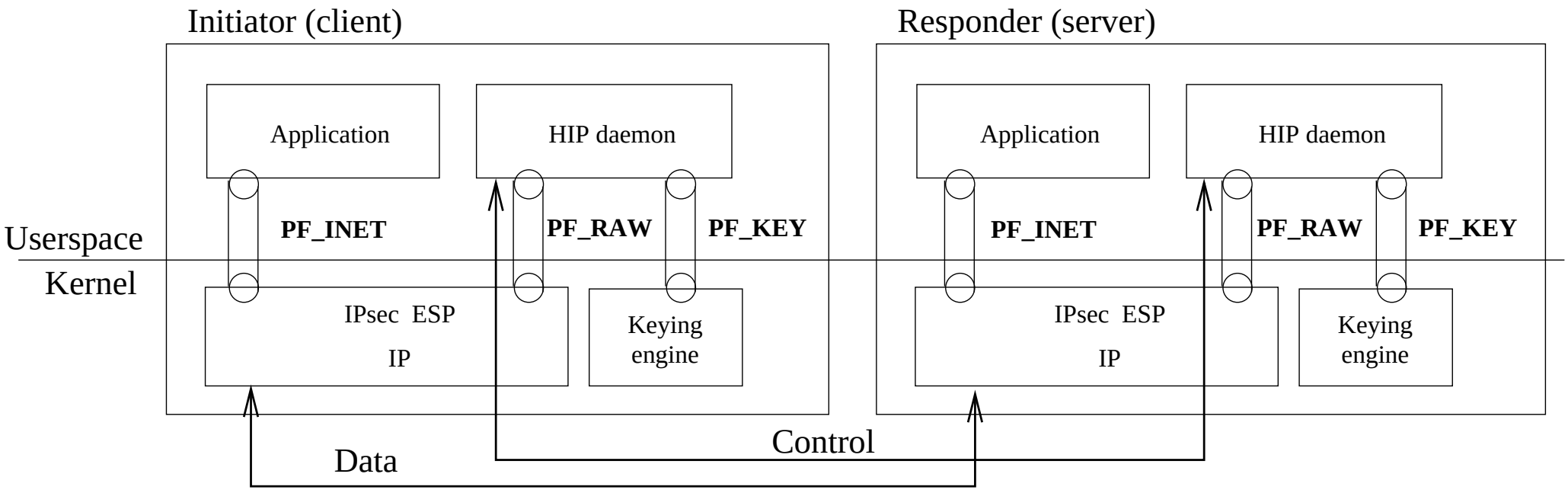
HIP Implementations

- HIP for Linux (HIPL)
  - developed by Helsinki Institute for Information Technology
  - runs on Linux
  - ongoing project to port to Symbian OS
  - originally a kernel IPv6-only implementation
  - IPv4 support is added in the user space

HIP Implementations

- OpenHIP
  - Boeing Phantom Works starts implementation
  - Linux, Windows, and Mac OS
  - a GPL license
- HIP for inter
  - developed by Ericsson NomadicLab
  - FreeBSD - the primary platform
  - both IPv4 and IPv6 are supported
  - ”Ericsson Finland Public Source” license

Internal structure of OpenHIP implementation



Graphical User Interface for managing HITs in HIPL