Secure Signaling in Next Generation Networks with NSIS

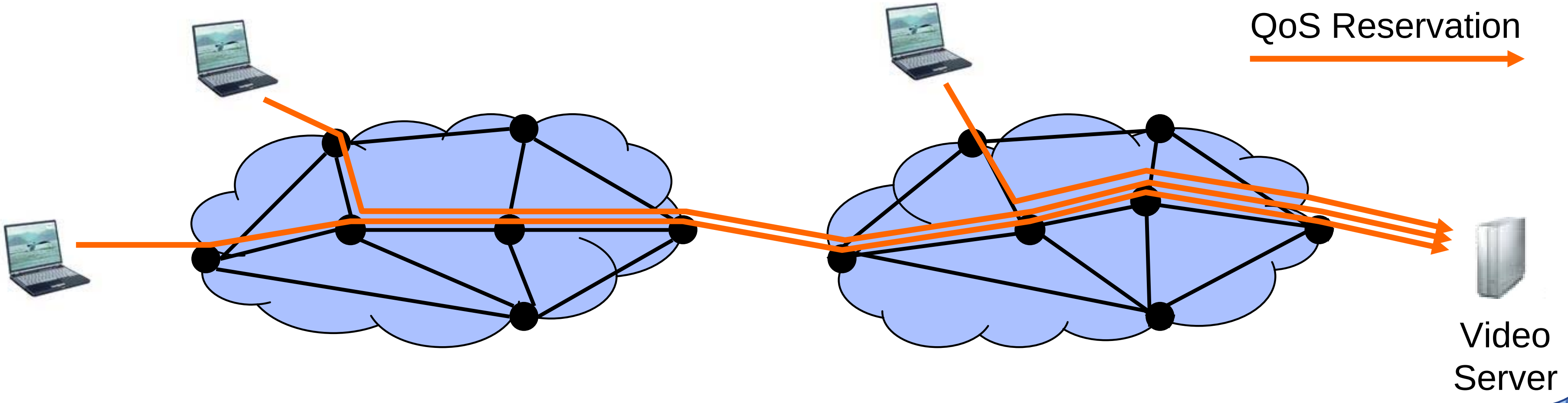
Roland Bless, Martin Röhricht

IEEE ICC 2009, Dresden

Institut für Telematik

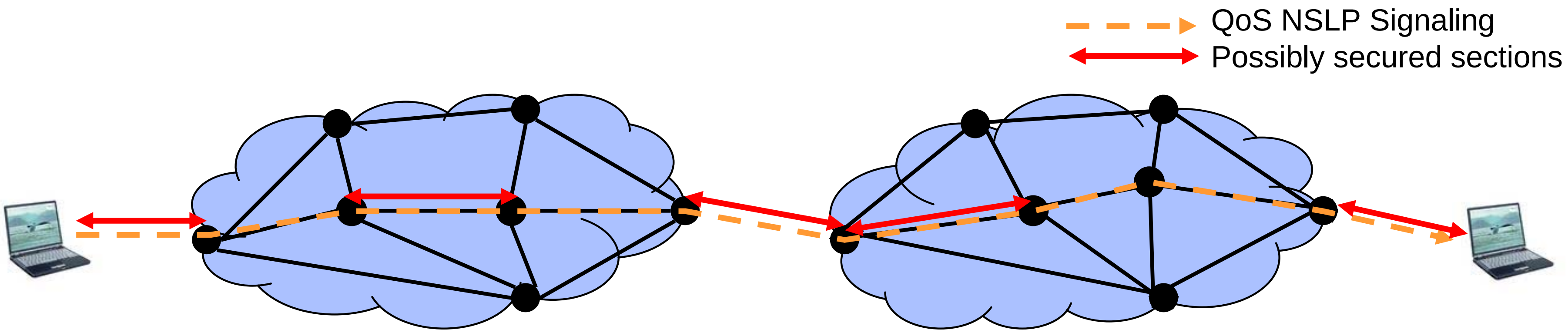
Motivation

- Signaling protocols important component for Next Generation Networks
  - Admission control for resource reservations
  - Management of network entities
  - RSVP → NSIS
- Security of signaling protocols important
  - QoS reservations
  - Firewall configurations
  - NAT traversal mappings



Next Steps in Signaling – Overview

- Two-layer approach
  - QoS or NAT/FW NSLP
  - NTLP, i.e. GIST
    - discovery of next signaling peer
    - signaling message transport (unreliable, reliable, secure)
- Channel security mechanisms at GIST level
  - Hop-by-hop based, not end-to-end
  - Multiplex several different sessions over one secured channel
  - No per-user authentication



Problem Statement

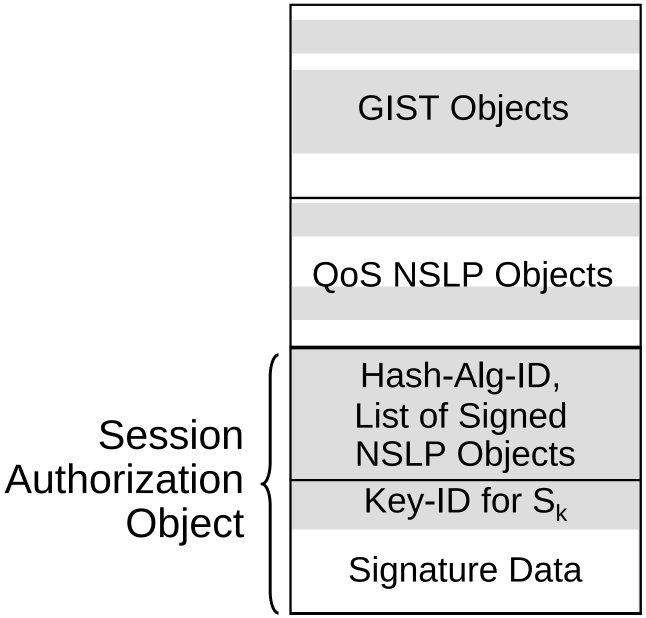
- No per-user or per-session authentication possible
  - No per-user authorization
  - No reliable and secure accounting
- Objective: provide integrity protection for every signaling message
- Session Authorization Policy Element
  - Relies on provision of authorization tokens from trusted third party
  - Opaque authorization token not sufficient
    - ▶ Not related to any signaling message objects

Main Challenges

- Add per-user authentication mechanism to Authorization Policy Element
- Integrity protection parts of signaling message
  - Some objects should still be modifiable by intermediate nodes
    - E.g. QoS parameter values
- Specify light-weight approach
  - Security shouldn’t add much additional (setup) delay
  - Thousands of signed signaling messages per node
    - Digital certificates not suitable

Proposal towards Authentic NSIS Signaling

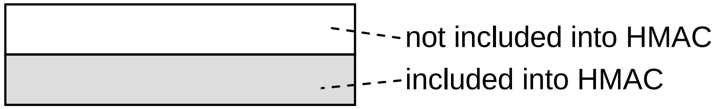
- Establish binding of authorization object and NSLP messages


GIST Objects
QoS NSLP Objects
Hash-Alg-ID, List of Signed NSLP Objects
Key-ID for S_k
Signature Data
Session Authorization Object

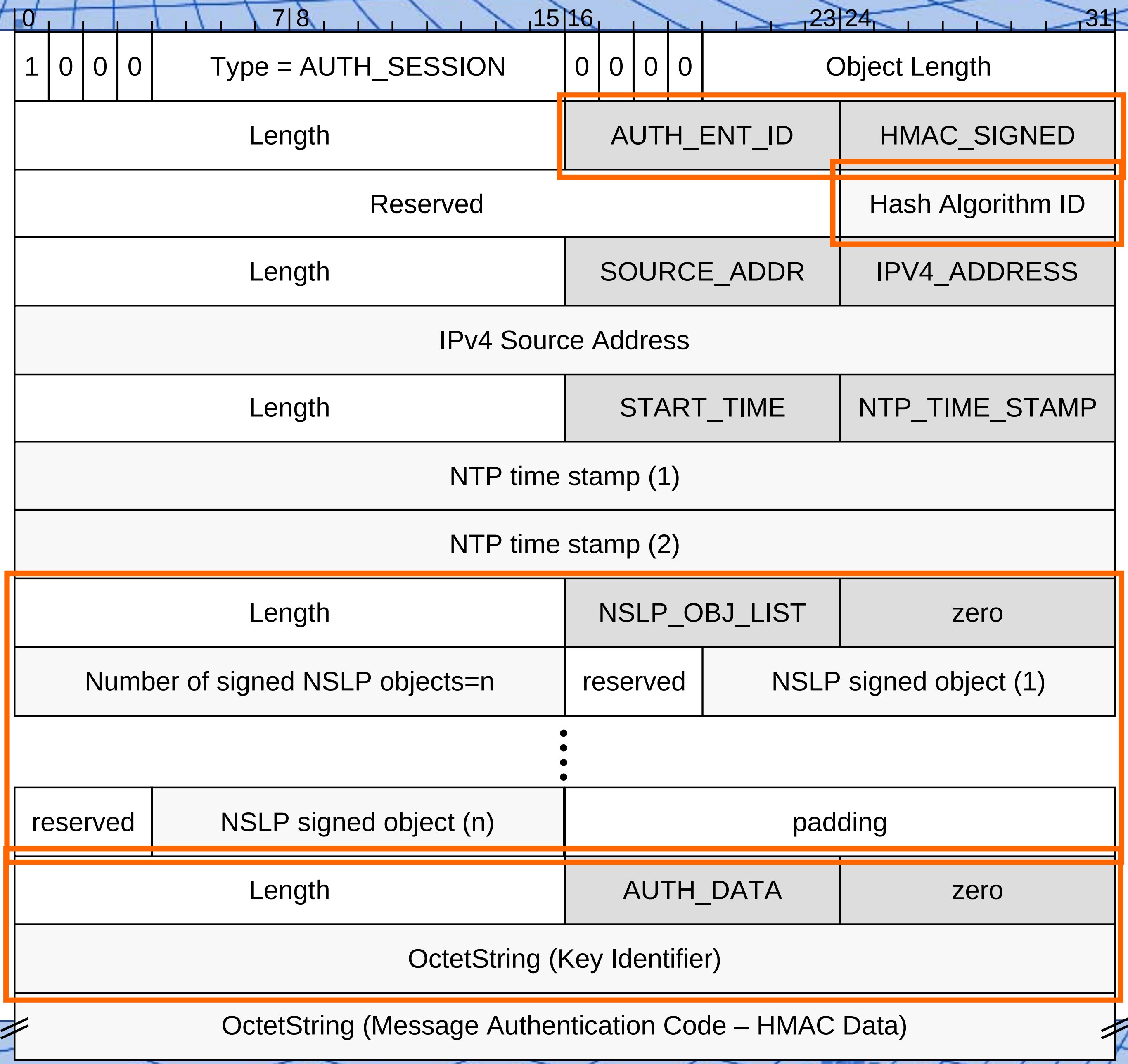
Protected parts of GIST PDU
e.g. Session ID, Message Routing Information

Protected parts of QoS NSLP PDU
e.g. INFO_SPEC, QSPEC

→ HMAC(S_k, Data)



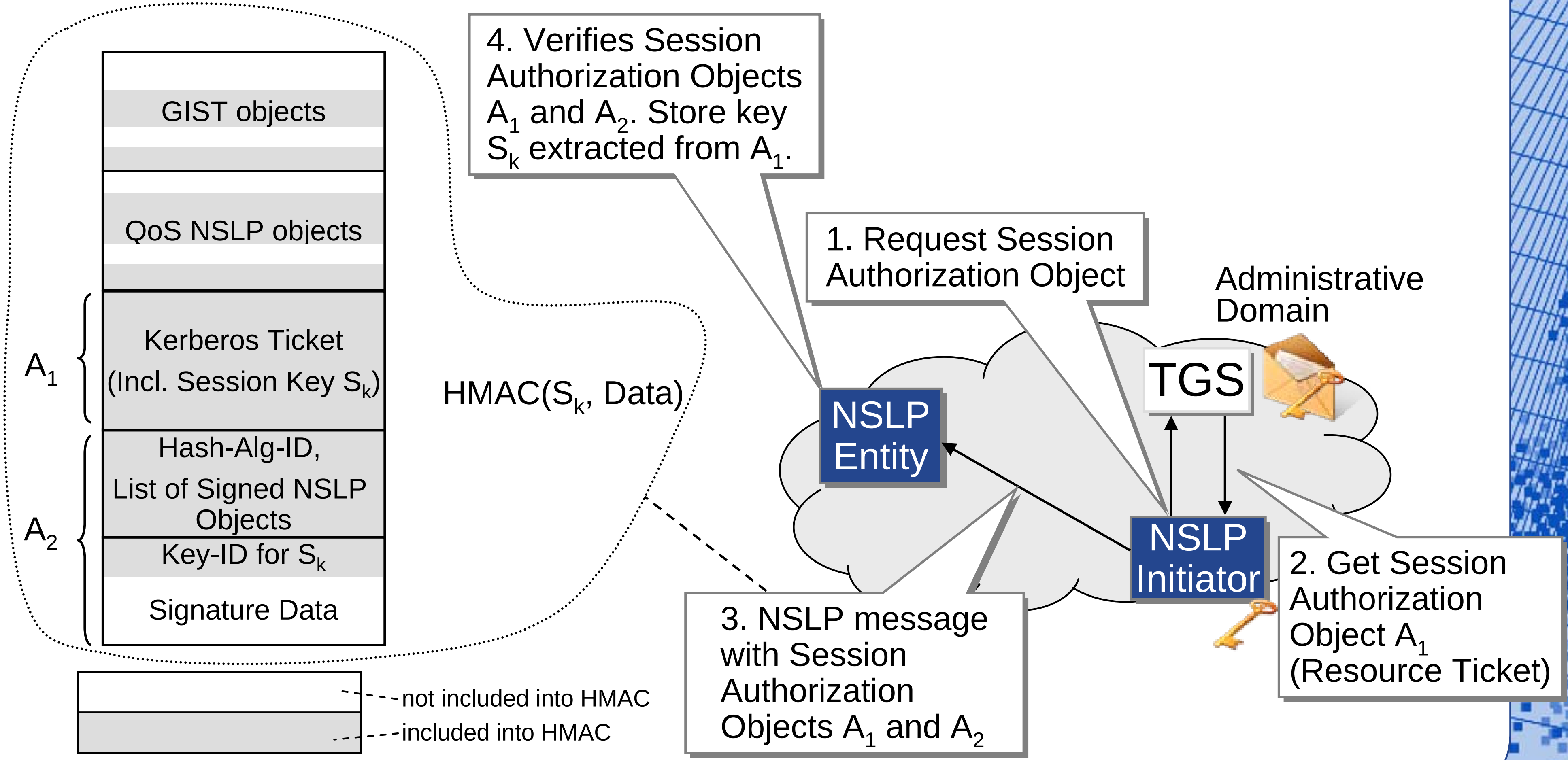
HMAC-based protection



<table>
<tr><td>0</td><td></td><td></td><td></td><td>7 | 8 ... 15</td><td>16</td><td></td><td></td><td></td><td>23</td><td>24 ... 31</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>Type = AUTH_SESSION</td><td>0</td><td>0</td><td>0</td><td>0</td><td colspan="2">Object Length</td></tr>
<tr><td colspan="5">Length</td><td colspan="5">AUTH_ENT_ID</td><td>HMAC_SIGNED</td></tr>
<tr><td colspan="10">Reserved</td><td>Hash Algorithm ID</td></tr>
<tr><td colspan="5">Length</td><td colspan="5">SOURCE_ADDR</td><td>IPV4_ADDRESS</td></tr>
<tr><td colspan="11">IPv4 Source Address</td></tr>
<tr><td colspan="5">Length</td><td colspan="5">START_TIME</td><td>NTP_TIME_STAMP</td></tr>
<tr><td colspan="11">NTP time stamp (1)</td></tr>
<tr><td colspan="11">NTP time stamp (2)</td></tr>
<tr><td colspan="5">Length</td><td colspan="5">NSLP_OBJ_LIST</td><td>zero</td></tr>
<tr><td colspan="5">Number of signed NSLP objects=n</td><td colspan="4">reserved</td><td colspan="2">NSLP signed object (1)</td></tr>
<tr><td colspan="11">⋮</td></tr>
<tr><td colspan="4">reserved</td><td>NSLP signed object (n)</td><td colspan="6">padding</td></tr>
<tr><td colspan="5">Length</td><td colspan="5">AUTH_DATA</td><td>zero</td></tr>
<tr><td colspan="11">OctetString (Key Identifier)</td></tr>
<tr><td colspan="11">OctetString (Message Authentication Code – HMAC Data)</td></tr>
</table>

Kerberos based Example

- Initial Session Authorization
  - Assumption: routers are “Kerberized” resources



1. Request Session Authorization Object

2. Get Session Authorization Object A_1 (Resource Ticket)

3. NSLP message with Session Authorization Objects A_1 and A_2

4. Verifies Session Authorization Objects A_1 and A_2. Store key S_k extracted from A_1.

NSIS-ka Suite

- Open Source C++-based, multi-threaded implementation for Linux
  - GIST
  - QoS NSLP
  - NATFW NSLP
- Well tested at Interop tests against different implementations
- Currently under active development
  - GIST-aware NAT-Gateways
  - Mobility support for/with MobileIPv6
  - Anticipated Handovers
  - Multicast Support
  - Integration into OMNeT++ simulation framework
- Code freely available: http://nsis-ka.org

Performance Evaluation

- Proposed integrity protection implemented and tested
- Benchmarks to determine overhead of HMAC computation
  - Intel Pentium IV 2.8GHz
  - Reading system clock at specific actions and keeping time stamps in memory
  - 50,000 runs measured in µs

Action	Min	Max	Mean	Stddev
Serialization	68.2	701.9	**69.1**	10.5
Serialization w. HMAC	89.4	718.1	**90.4**	8.3
Deserialization	74.4	705.6	**75.3**	8.8
Deserialization w. HMAC	97.6	746.3	**99.2**	9.8

- Creation of Session Authorization Object including HMAC computation
  - 30.8% overhead (Mean)
- HMAC verification and deserialization of PDU
  - 31.8% overhead (Mean)

Conclusion & Outlook

- Allows for user-based authentication
- Integrity protection of important parts of an NSLP message
- Uses resource efficient HMAC-based signatures
- Key exchange not per session required
  - Only per user
- No further backend communication needed by intermediate nodes for integrity checks
- Low communication overhead
- Not restricted to a particular NSLP

Thanks! Questions?

www.tm.uka.de/itm

Institut für Telematik