- to be on the safe side -

www.crysys.hu

Version Number Authentication and Local Key Agreement

<draft-dvir-roll-security-extensions-00.txt>

Levente Buttyán
Laboratory of Cryptography and System Security (CrySyS)
Budapest University of Technology and Economics

this is joint work with ***Amit Dvir, Tamás Holczer,*** and ***László Dóra***

work supported by the WSAN4CIP Project (www.wsan4cip.eu)
(funded by the European Comission within FP7)

Contents

- DIO message (broadcast) authentication
  - prevents misbehaving (compromised) nodes to become DODAG roots by sending out a DIO message for DODAG update with an increased Version Number

- Local key exchange
  - allows each node to establish a set of pairwise keys (shared with each of its local neighbor) and a cluster key (shared with all neighbors)

Motivations for DIO message authentication

- lack of physical protection + unattended operation → nodes may be accessed physically and get compromised
- by sending a well crafted DIO message, a compromised node can easily become the root of the DODAG and divert all traffic towards itself
- as each node rebroadcasts DIO messages, a single crafted DIO message can generate a lot of traffic and increase overall energy consumption
- pairwise keys shared between neighboring nodes as envisioned in the ROLL security framework does not solve the problem

Design considerations

- requirements
  - broadcast authentication
    - only the real DODAG root should be able to generate valid DIO messages, and all nodes should be able to authenticate them
  - efficiency
    - use symmetric key cryptography as much as possible
- trade-off
  - authenticate only the Version Number
    - this ensures that (re)construction of the DODAG can only be initiated by the real DODAG root
- approach
  - use a hash chain for authenticating the Version Number, and …
  - symmetric or asymmetric key message authentication for authenticating the root of the hash chain (and the static part of the DIO message)

Protocol overview

- the DODAG root generates a random number r, and computes a hash chain h (r), h(h(r)) = $h^2(r)$, …, $h^n(r)$
- the DODAG root distributes the root $h^n(r)$ of the hash chain to all nodes in the network by
  - including $h^n(r)$ in a DIO message
  - authenticating $h^n(r)$ and the static fields of the DIO message, such that all other nodes can be sure that $h^n(r)$ originates from the DODAG root
    - digital signature verifiable with the public key of the DODAG root
    - MAC computed with a globally shared key K that can be assumed to be non-compromised at the time of distributing and authenticating $h^n(r)$
- when the DODAG root sends out a DIO message with a new Version Number, it also releases the next hash chain value
  - note that the next hash chain value $h^{i-1}(r)$ cannot be computed from the last released value $h^i(r)$ due to the one-way property of the hash function h
- each node verifies that the new hash chain value hashes into the last received one, and stores the new value as the latest received hash chain value
- when the DODAG root runs out of hash chain values it generates a new chain and distributes its root as described before

The Broadcast Authentication option

C – Continues bit

if set then authentication data continues in the next option

H – Type of data in the Authentication Data field

0: not a hash value (i.e., signature or MAC)
1: hash chain root
2: current hash chain value

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type=10     | Option Length |C| H |Resvd    | Security Alg  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
:                    Authentication Data                        :
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

defines the algorithm that should be used to interpret the Auth Data field (hash fn, MAC fn, digital sig scheme)

Examples

- Broadcast Authentication options carrying a hash chain root and a digital signature on the hash chain root and the static fields of the DIO message:

```
+--+---+-+--+-+-------+
|10| 34|0|1 |0|  0x01 |
+--+---+-+--+-+-------+
|Hash Chain Root Value|
+---------------------+
+--+---+-+--+-+-------+
|10|255|1|0 |0|  0xC0 |
+--+---+-+--+-+-------+
|   IProt part 1      |
+---------------------+
+--+---+-+--+-+-------+
|10|136|0|0 |0|  0xC0 |
+--+---+-+--+-+-------+
|   IProt part 2      |
+---------------------+
```

- Broadcast Authentication option carrying the latest hash chain value:

```
+--+---+-+--+-+----------+
|10| 34|0|2 |0|   0x01   |
+--+---+-+--+-+----------+
|Current Hash Chain Value|
+------------------------+
```

Motivation for pairwise key establishment

- the RPL security framework envisions the use of cryptographic mechanisms on the links between neighboring nodes, but …
- it does not specify a key exchange method to set up the necessary keys

The LEAP protocol

- main assumptions:
  - any node will not be compromised within time T after its deployment
  - any node can discover its neighbors and set up keys with them within time $T_{kex} < T$
    - kex
      practice

- protocol phases:
  - key pre-distribution phase
  - neighbor discovery phase
  - link key establishment phase
  - key erasure phase

The LEAP protocol

- key pre-distribution phase
  - before deployment, each node is loaded with a master key K
  - each node u derives a node key K_u as $K_u = f(K, u)$, where f is a one-way function
- neighbor discovery phase
  - when a node deployed, it tries to discover its neighbors by broadcasting a "HELLO" message

$$u \rightarrow *: u$$

  - each neighbor v replies with

$$v \rightarrow u: v, \mathrm{MAC}(K_v, u|v)$$

  - u can compute $f(K, v) = K_v$, and verify the authenticity of the reply

The LEAP protocol

- link key establishment phase
  - u computes the link key $K_{uv} = f(K_v, u)$
  - v computes the same key
  - no messages are exchanged
  - note:

    u does not authenticate itself to v, but …
    - only a node that knows K can compute K_{uv}
    - a compromised node that tries to impersonate u cannot know K (see below)
- key erasure phase
  - time T after its deployment, each node deletes K and all keys it computed in the neighbor discovery phase

Integration into the RPL protocol

- LEAP protocol messages:
  u → * (local broadcast message): u
  v → u (response message): v, MAC(K_v, u|v)

- no explicit HELLO message is needed, any locally broadcasted RPL message from u can serve as the LEAP HELLO message
- LEAP response message can be piggy-backed as an option on a RPL message:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type=11     | Option Length |  Comp Algo    | MAC Function  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                          Response MAC                         :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                        Compressed Address                     :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Integration options

- S1: u → * DAO Multicast
  v → u DAO Unicast Ack
- S2: u → * DAO Multicast
  v → u DAO Multicast Ack
- S3: u → * DAO Multicast
  v → u DAO Multicast
- S4: u → * DIO Multicast
  v → u DIO Unicast
- S5: u → * DIS Multicast
  v → u DIO Unicast
- S6: u → * DIS Multicast or DIO Multicast
  v → u DIO Multicast
- S7: u → * New RPL Base Message
  v → u New RPL Base Message

Selection criteria

RPLM: The scheme should not introduce a new RPL message type.

RPLF: The scheme should not change RPL functionality.

EFFI: The scheme should be efficient (low communication and computation overhead).

STP: The local key agreement must be completed before the safe time period expires.

BN: The scheme must work when the network boots and when a new node joins the DODAG.

NEI: The scheme must find all of a node's neighbors.

MAND: The scheme should prefer mandatory RPL message types (i.e., DIO, DIS).

RELY: The scheme should not rely on DODAG or DODAGID.

Setting up a cluster key

- protocol:
  - node u generates a random key
  - for each neighbor, node u encrypts this random key with the pairwise key shared with that neighbor
  - node u sends the encrypted random key to each of its neighbors
- LEAP cluster key option:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type=12     | Option Length |  Key Length   | ENC Function  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                     Encrypted Cluster Key                     :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Status of implementation

- we implemented the RPL protocol on two platforms
  - Linux 2.6.36, user space
  - TinyOS 2.x
- we implemented the security extensions on Linux using the OpenSSL library
- the TinyOS implementation will use the TinyECC library
- our implementations will be used in the demos of the WSAN4CIP project
  - monitoring power grid lines and substations (Linux + WiFi)
  - monitoring a drinking water pipeline (TinyOS + TDMA based MAC)
- results of our experiments can be expected in the second half of 2011