draft-dvir-roll-security-authentication-01
and
draft-dvir-roll-security-key-agreement

Amit Dvir
Laboratory of Cryptography and System Security (CrySyS)
Budapest University of Technology and Economics

draft-dvir-roll-security-authentication-01
and
draft-dvir-roll-security-key-agreement

Amit Dvir
Laboratory of Cryptography and System Security (CrySyS)
Budapest University of Technology and Economics

draft-dvir-roll-security-authentication-01
and
draft-dvir-roll-security-key-agreement

Amit Dvir
Laboratory of Cryptography and System Security (CrySyS)
Budapest University of Technology and Economics

draft-dvir-roll-security-authentication-01
and
draft-dvir-roll-security-key-agreement

Amit Dvir
Laboratory of Cryptography and System Security (CrySyS)
Budapest University of Technology and Economics

draft-dvir-roll-security-authentication-01
and
draft-dvir-roll-security-key-agreement

Amit Dvir
Laboratory of Cryptography and System Security (CrySyS)
Budapest University of Technology and Economics

draft-dvir-roll-security-authentication-01
and
draft-dvir-roll-security-key-agreement

Amit Dvir
Laboratory of Cryptography and System Security (CrySyS)
Budapest University of Technology and Economics

draft-dvir-roll-security-authentication-01
and
draft-dvir-roll-security-key-agreement

Amit Dvir
Laboratory of Cryptography and System Security (CrySyS)
Budapest University of Technology and Economics

draft-dvir-roll-security-authentication-01
and
draft-dvir-roll-security-key-agreement

Amit Dvir
Laboratory of Cryptography and System Security (CrySyS)
Budapest University of Technology and Economics

draft-dvir-roll-security-authentication-01
and
draft-dvir-roll-security-key-agreement

Amit Dvir
Laboratory of Cryptography and System Security (CrySyS)
Budapest University of Technology and Economics

draft-dvir-roll-security-authentication-01
and
draft-dvir-roll-security-key-agreement

Amit Dvir
Laboratory of Cryptography and System Security (CrySyS)
Budapest University of Technology and Economics

this is joint work with

Levente Buttyán, Tamás Holczer and Dóra László
RPL – WG Vs CrySyS Security View

Both are important and should be implemented.

- A security framework.
- Link-by-link protection.
- External attacker, the basic assumption is trustful nodes.

After a while, a node in the field can be the problem
- Nodes may be accessed physically/logically and get compromised.
- Internal attacks by compromised nodes.
Example

Duration: 1/1/09 – 31/12/12,
Total effort: 370 PM, Total costs/funding: 4.0 / 2.8 Mio EUR
The review went well
Security/Efficiency Trade off (or why protect some and not all)

- An internal adversary can attack in many different ways.
- Not all attacks may have a major influence.
- Reconstructing the entire DAG, exhaust the nodes’ batteries, or eavesdropping a large part of the traffic can have major or even critical influence.
- One way to achieve attacker’s goals is to change/modify the Version Number or to change/modify the DODAG node Rank.
  - Modifying these may have a global effect.
- Therefore, in this draft we present a security scheme to prevent those attacks.
Version Number authentication

- The DODAG root generates a random number $v_n$, and computes a hash chain $v_{n-1} = h(v_n)$, $v_{n-2} = h(v_{n-1})$, ..., $v_0 = h(v_1)$

- The DODAG root distributes the root $v_0$ of the version hash chain to all nodes in the network by
  - Including $v_0$ in a DIO message
  - Authenticating $v_0$ and the static fields of the DIO message, such that all other nodes can be sure that $v_0$ originates from the DODAG root
    - Digital signature verifiable with the public key of the DODAG root
    - When the DODAG root sends out a DIO message with a new Version Number, it also releases the next version hash from the chain
      - Note that the next version hash $v_i$ cannot be computed from the last released value $v_{i-1} = h(v_i)$ due to the one-way property of the hash function $h$
Rank Authentication

Compute hash chain for every Version number hash chain element
Upon Version Number Update:

- **DODAG Root**
  - Version Number Update?
    - Y:
      - Rank = OF(root)
      - \( R_{\text{sender}} = h^{\text{Rank}}(x_i) \)
      - \( R_{\text{mrh}} = h(h(...h(x_{i+1}))) \)
    - N:
      - Send DIO: \( V_i, MAC_{V(i+1)}(R_{\text{mrh}}), R_{\text{sender}} \)

- **DODAG Node v**
  - Get DIO: \( V_i, MAC_{V(i+1)}(R_{\text{mrh}}), R_{\text{sender}} \)
  - \( R_{\text{check}} = h^{\text{Rank}_{\text{parent}}}(R_{\text{sender}}) \)
    - Y:
      - \( V_0 = h^{i}(V_i) \)
    - N:
      - Calculate \( MAC_{\text{check}} = MAC_{V_i}(R_{\text{check}}) \)
      - Y:
        - Rank monotonically increase!!
      - N:
        - Invalid Value
        - Rank = OF(v)
        - \( \Delta = \text{Rank}_v - \text{Rank}_{\text{parent}} \)
        - \( R_{\text{sender}} = h^{\Delta}(R_{\text{sender}}) \)
    - Send DIO: \( V_i, MAC_{V(i+1)}(R_{\text{mrh}}), R_{\text{sender}} \)
<table>
<thead>
<tr>
<th>Value</th>
<th>Code Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Version Number Hash Chain Value</td>
</tr>
<tr>
<td>1</td>
<td>Version Number Hash Chain Root</td>
</tr>
<tr>
<td>2</td>
<td>Rank Hash Chain Value</td>
</tr>
<tr>
<td>3</td>
<td>MAC of Max Rank Hash Value</td>
</tr>
<tr>
<td>4</td>
<td>Integrity Protection Value</td>
</tr>
<tr>
<td>else</td>
<td>Unassigned</td>
</tr>
</tbody>
</table>

Figure 6: Code Type
Pairwise key establishment – LEAP++

- **Different draft**

- **main assumptions:**
  - Any node will not be compromised within $T$ time after its deployment
  - Any node can discover its neighbors and set up keys with them within $T_{kex} < T$ time
    - typically, $T_{kex}$ is a few seconds, so these assumptions make sense in practice
  - Each node has a preshared secret key $K$ at boot/restart

- **protocol phases:**
  - key pre-distribution phase
  - neighbor discovery phase
  - link key establishment phase
  - key erasure phase
| Pairwise key | Hello Message - Any RPL message.  
|             | Response Message - Unicast DIS.  
|             | Ack Message (OPTIONAL) - Unicast DIO |
| Cluster Key | Hello Message - Any Unicast RPL message.  
|             | Ack Message (OPTIONAL) - Any Unicast RPL message |

---

Hello Message, $u \rightarrow *: (u)$
Response Message, $v \rightarrow u: (\text{MAC}(K_v, v|u))$
Ack Message, $u \rightarrow v: (\text{MAC}(K_{uv}, u|v))$

---

ENC Message, $u \rightarrow *: (\text{ENC}_{K_{uv}}(r))$
Ack Message, $v \rightarrow u: (\text{MAC}(K_{uv}, r))$
Conclusion

- We identified illegitimate Version Number increases and Rank value decreases as two powerful attacks against RPL.
- We proposed solutions that prevents both of the attacks based on stable mechanisms.
- We use public/private operations once in a long while; we use nearly only symmetric operations.
- For the key exchange, we are using preinstalled keys for a short time.
Answers:

- In theory, \( l \) should be 65535 to have a different value for each possible Rank value (Rank is 16 bits long [I-D.ietf-roll-rpl]). In practice, 256 is large enough as for most of the operations DAGRank [I-D.ietf-roll-rpl] is used (the DAG Rank of a node is the upper 8 bits of the Rank), which is 8 bits long.
  - If DAGRank is used when defining the hash chain, then all occurrences of Rank must be substituted by DAGRank in the sequel
- **Elliptic Curve Cryptography**
  - ECC-SECP256K1 with SHA-256