INTERNET-DRAFT A. Dvir
Intended Status: Experimental T. Holczer
Expires: June 25, 2011 L. Dora
L. Buttyan
January 14, 2011

<Version Number Authentication and Local Key Agreement>
<draft-dvir-roll-security-extensions-00.txt>

Abstract

Low power and Lossy Networks (LLNs) are a class of networks in which both the routers and their interconnects are constrained. LLN routers typically operate with constraints on processing power, memory, and energy (battery power). LLN router supported traffic flows include point-to-point, point-to-multipoint, and multipoint-to-point. The IPv6 Routing Protocol for LLNs (RPL) provides the mechanisms to support those traffic flows. The currently available security services in RPL will not protect against a compromised internal node that can also construct and disseminate fake messages. In this document, a service is described that prevents an internal attacker from impersonating a Destination Oriented Directed Acyclic Graph (DODAG) root. Moreover, the establishment and maintenance of any cryptographic key is out of the scope of the current RPL proposal. In this document a service that allows nodes to agree on local keys with their neighborhood is also presented.

http://www.ietf.org/shadow.html.

Copyright and License Notice

Table of Contents

1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119]. This document adopts and conforms to the terminology defined in [I-D.ietf-roll-terminology] and in [RFC4949].

In this document, 'compromised' refers to taking control over a node. 'Potential DODAG roots' are grounded DODAG roots and a small set of capable nodes that could become floating DODAG roots. 'Data authenticity' is the assurance about the source of transmitted information (and, hereby, that information was not modified in transit).

As they form networks, LLN devices often mix the roles of 'host' and 'router' when compared to traditional IP networks. In this document, 'host' refers to an LLN device that can generate but does not forward RPL [I-D.ietf-roll-rpl] traffic, 'router' refers to an LLN device that can forward as well as generate RPL traffic, and 'node' refers to any RPL device, either a host or a router.

2 Introduction

Low power and Lossy Networks (LLNs) consist largely of constrained nodes with limited processing power, memory, and sometimes energy, when they are battery operated. These routers are interconnected by unstable lossy links, typically supporting only low packet and data delivery rates. Another characteristic of such networks is that point-to-point is not the typical traffic pattern, but point-to-multipoint or multipoint-to-point are. Furthermore, such networks may potentially comprise up to thousands of nodes.

These characteristics offer unique challenges to a routing solution. The IETF ROLL Working Group has defined application-specific routing requirements for a Low power and Lossy Network (LLN) routing protocol, specified in [RFC5867], [RFC5826], [RFC5673], and [RFC5548]. Moreover, based on those standards, an IPv6 Routing Protocol for Low power and Lossy Networks (RPL) has been proposed [I-D.ietf-roll-rpl] and a security framework for RPL is described in [I-D-roll-security-framework].

Many LLN systems are deployed in unattended or remote locations, such as urban environments [RFC5548]. Hence, security mechanisms that provide confidentiality and authentication are critical for the operation of many applications. The currently available security services in RPL proposed in [I-D.ietf-roll-rpl] will not protect against a compromised internal node that can also construct and disseminate fake messages. Moreover, the establishment and maintenance of any cryptographic key is out of the scope of the current RPL proposal [I-D.ietf-roll-rpl]. Therefore, this document presents two new security services for RPL:

- o DIO Message Broadcast Authentication - secures the network from misbehaving nodes to become a DODAG root and to increase the Version Number.

- o Local Key Agreement - allows each node to agree on local keys with its neighborhood.

The implementation of the security services described in this document are OPTIONAL. A given implementation MAY support a subset (including the empty set) of the described security services; for example, the implementation could support Local Key Agreement, but not DIO Message Authentication. An implementer SHOULD clearly specify which security services are supported, and it is RECOMMENDED that implementers carefully consider security requirements and the availability of security mechanisms in their network.

3 Security Services

This section describes two protocols; the first enables nodes to authenticate DIO Messages. The second protocol enables nodes to a) agree on a pairwise key, with each of its neighbors; and b) generate and disseminate a cluster key, a shared key between a node and all of its neighbors. The hash functions, MAC functions, and the digital signatures used in the protocols are based on sections 10.1 and 10.9.2 of [I-D.ietf-roll-rpl], e. g., SHA-256 hash function specified in Section 6.2 of [FIPS180], message encoding rules of Section 8.1 of [RFC3447]. The elliptic curve cryptography (ECC) used in section 3.1 is based on section 2.7 of [SECG2]. The Counter with CBC-MAC (CCM) used in section 3.2, is described in [RFC3610]. Note that although [RFC3610] disallows the CCM mode with M=0, RPL explicitly allows the CCM mode with M=0 when used in conjunction with a signature, because the signature provides sufficient data authentication. Here, the CCM mode with M=0 is specified as in [RFC3610], but where the M field in Section 2.2 of [RFC3610] MUST be set to 0. The Hashed Message Authentication Mode (HMAC) in the protocols is described in [RFC4868].

3.1 DIO Message Authentication

A grounded DODAG offers connectivity to hosts that are required to satisfy the application-defined goal. An attacker may try to become a DODAG root by sending a well-constructed DIO message where the grounded flag is set. The scope of the current RPL security services is the link; therefore, the authenticity of the messages sent by the DODAG root relies on the trustworthiness of all intermediate nodes and the fact that none of the keys are compromised. Any key that is compromised allows an attacker to send an authentic DIO message that will be accepted by all the nodes. Therefore, a node that received the DIO message from the attacker will multicast to its neighbors the DIO message using uncompromised keys. The content of the message from the attacker will affect other nodes participating in the DODAG.

RPL [I-D.ietf-roll-rpl] allows the Version Number to be increased regularly or occasionally. Moreover, the whole network can be reconstructed by sending a DIO message with an increased Version Number. Therefore, preventing any misbehaving node from impersonating the actual DODAG root by increasing the Version Number is essential. In particular, only those parts of the DIO message that do not need to be updated when the nodes forward the DIO message can be protected. The static fields are the following:

- o DIO Base Object:
  - o RPLInstanceID
  - o G|A|T|MOP|Prf
  - o DODAGID
  - o Version
- o Routing Information (option)
- o DODAG Configuration (option)

By authenticating the DIO message, each node can securely forward the DIO message in order to bootstrap or update the DODAG.

The Authentication procedure starts/updates from a DODAG root toward the nodes as follows:

1. The DODAG root first generates a random number r.

2. The DODAG root calculates h(h...(h(h(r)))), also denoted by h^n(r), where h() is a hash function and n is the length of the chain. This value is called the hash chain root [L1981].

3. The DODAG root authenticates the h^n(r) value as well as the static fields using any supported integrity protection

algorithm (e. g., digital signature or a MAC function).

4. The DODAG root sends a DIO message with the authenticated value.

5. Each node receiving a DIO message verifies the authenticity of the static fields of the message.

6. If the message is authentic, the node saves the Version Number value (init or update value), the hash chain value (root or current chain value), and the integrity protection data (MAC value or signature) for future use, and multicast to all neighbors the DIO message after updating the fields as described in section 6.3.1. of [I-D.ietf-roll-rpl].

7. If the message is not authentic, the receiver MUST ignore the message.

In case the implementer decides to authenticate the hash chain root with an integrity protection mechanism, steps 1-7 MUST be implemented. If not, only steps 1-2 MUST be implemented. When digital signature is used, each node has to know the public signature verification key. When symmetric keys are used, all nodes must have a preshared key K. In order to minimize the computation time and memory usage of the hash chain, the implementer can use the technique in [OptHash] on the DODAG root side.

When the DODAG root increases the Version Number (by k from the initial Version Number value), the DODAG root reveals the value of h^(n-k)(r) and inserts this value in the DIO message with Broadcast Authentication Option. When node v receives the DIO message it can easily verify the message because, if the Version Number is increased by the DODAG root, h^k(h^(n-k)(r)) must be equal to h^n(r). For an attacker, computing the previous element h^(i-1)(r) knowing h^i(r) is hard when r is not known and h() is a cryptographic one-way function.

In order to authenticate the static fields of a DIO message and the Version Number, a DIO MUST carry one or more "Broadcast Authentication" options. A Broadcast Authentication option consists of the following fields:

```
        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                         :
       |                                                               |
       +                                                               +
       |                                                               |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Format of the Broadcast Authentication Option

Option Type: 0x0A (to be confirmed by IANA)

Option Length: 8-bit unsigned integer, variable length of the option in octets excluding the Type and Length fields.

C: Continues bit, the C bit is set whenever the signature/Hash/MAC output has length greater than maximum option data length; the receiver needs to merge it with the other Broadcast Authentication Options with the same H type until the C bit is unset.

H: 2-bit field, indicating which part of the hash chain is in the Authentication data field.

Bit Number	Hash Value Type
0	No Hash Value
1	Hash Root Chain Value
2	Current Hash Chain Value
3	Unassigned

Figure 2: Hash Value Type

Resvd: 5-bit unused field. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Security Algorithm: The Security Algorithm field specifies the encryption, MAC, and signature scheme used by the network. The high order bit (0x80) of the code denotes whether Integrity Protection has been enabled. The second high order bit (0xC0) of the code denotes whether the Integrity Protection is using symmetric or asymmetric key algorithms. Supported values of this field are as follows:

Bit Number	Security Algorithm
0x00	No Security Algorithm
0x01	SHA-256
0x02	SHA-512
0x80	HMAC-SHA-256
0x81	HMAC-SHA-512
0xC0	RSA with SHA-256
0xC1	ECC-SECP256K1 with SHA-256
else	Unassigned

Figure 3: Security Algorithm

Authentication Data: Contains the authentication data compatible with the Hash and Protection Type fields.

Unassigned bits of the Broadcast Authentication option are reserved. They MUST be set to zero on transmission and MUST be ignored on reception.

3.1.1 Sequence Diagram

The sequence diagram of the DIO Message Authentication has three parts: authentication procedure, Version Number update, and, admission of a new node in the DODAG.

```
Root                  Node v          Node u             New Node

  |  VN=n, HR, IP     |               |                       |
  |-------------->M#1|               |                       |
  |                  | VN=n, HR, IP  |                       |
  |                  |----------->M#2|                       |
  |                  |               |                       |
  :                  :               :                       :
  |                  |               |                       |
  | VN=n+i, CH       |               |                       |
  |-------------->M#3| VN=n+i, CH    |                       |
  |                  |----------->M#4|                       |
  |                  |               |                       |
  :                  :               :                       :
  |                  |               |                       |
  |                  |               | Unicast DIS           |
  |                  |               |M#5<-------------------|
  |                  |               |                       |
  |                  |               |VN=n+i, n, HR, CH, IP |
  |                  |               |------------------->M#6|
  |                  |               |                       |
```

Figure 4: Sequence Diagram of DIO Message Authentication

M - Message
VN - Version Number
n - Initial value of the Version Number
HR - Hash root chain value
CH - Node chain value
IP - Integrity protection

Messages #1 and #2 refer to the authentication procedure. The DIO messages (messages #1 and #2) consist of the following Broadcast Authentication Options (the format of the option is described in Figure 1):

o The value of the chain root, HR value:

```
+--+---+-+--+-+-------+
|10| 34|0|1 |0|  0x01 |
+--+---+-+--+-+-------+
|Hash Root Chain Value|
+---------------------+
```

o The integrity protection, IP value:

```
+--+---+-+--+-+-------+
|10|255|1|0 |0|  0xC0 |
+--+---+-+--+-+-------+
|   IProt part 1      |
+---------------------+

+--+---+-+--+-+-------+
|10|136|0|0 |0|  0xC0 |
+--+---+-+--+-+-------+
|   IProt part 2      |
+---------------------+
```

The length of the integrity protection value (3096 bits in this example) can be larger than the maximum length of the Authentication data.

Each DODAG node saves the IP value, Root value, and the initial Version Number (taken from the DIO message). Each DODAG node sends the DIO message to its neighbors.

In the case when a root wants to update the Version Number, the DIO messages (messages #3 and #4) consist of the following Broadcast Authentication Option(the format of the option is described in Figure 1):

o One of the node’s value of the hash chain, CH value:

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

Each DODAG node verifies the values as explained above and saves the current hash value and the current Version Number (taken from the DIO message).

In the case when a new node (newcomer) wants to join the DODAG, a node receiving a unicast DIS message (message #5) from the new node (newcomer) must reply with a DIO message (message #6), consisting of the following Broadcast Authentication Options(the format of the option is described in Figure 1):

o The root chain value (HR value, as sent in message #1 and #2):

```
+--+---+-+--+-+-------+
|10| 34|0|1 |0|  0x01 |
+--+---+-+--+-+-------+
|Hash Root Chain Value|
+---------------------+
```

o The current hash value (CH value, as sent in messages #3, #4):

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

o The integrity protection, IP value, as sent in message #1 and #2:

```
+--+---+-+--+-+----+
|10|255|1|0 |0|0xC0|
+--+---+-+--+-+----+
|   IProt part 1   |
+------------------+

+--+---+--+-+-+----+
|10|136|0|0 |0|0xC0|
+--+---+-+--+-+----+
|   IProt part 2   |
+------------------+
```

o The initial Version Number, VN value as sent in message #1 and #2:

```
+--+---+-+--+-+------+
|10|  3|0|0 |0|0x00  |
+--+---+-+--+-+------+
|Init Version Number |
+--------------------+
```

The new node saves the IP value, Root value, current Version Number (taken from the DIO message), and the initial Version

Number.

3.2 Local Key Agreement

Providing security is particularly challenging to LLN networks due to the resource limitations. If a group key is used for peer-to-peer communication, protection is provided only against outsider devices and not against potential malicious devices in the key-sharing group. However, local key agreements can be used despite the node limitation in order to authenticate MAC layer one-hop unicast and multicast for all neighbors' messages. The establishment and maintenance of any cryptographic key for security services is out of the scope of the current RPL proposal. This section describes two protocols, establishment of a pairwise key and establishment of a cluster key. Both protocols assume the following:

- o T is defined as the lower bound on the time for an adversary to compromise a node. T is measured from the boot/restart time of the node.

- o T is greater than the accumulated time required to construct a DODAG and the time to create local key agreements.

- o Each node has preshared key K at boot/restart.

3.2.1 Pairwise Shared Key

This section describes a pairwise shared key agreement protocol based on the Localized Encryption and Authentication Protocol (LEAP) [LEAP]. This section does not provide results on LEAP's performance or behavior, nor does it explain the algorithm's design in detail. Interested readers should refer to [LEAP].

The pairwise key agreement consists of the following steps:

- o Each node sets the safe period timer; the pairwise key agreement protocol assumes that the nodes are not compromised before this timer expires.

- o Each u node derives its own key Ku=MAC(K,u), K is a preshared master key, and u is the IPv6 address of the node.

- o Each Node u multicasts its identifier to all neighbors.

- o Each node v receiving the identifier from u, responds with message (v, MAC(Kv, u|v)).

- o The pairwise key Kuv is generated as: Kuv=MAC(Kv,u).
- o After the safe period timer expires, each node deletes the preshared key K (from its memory).
- o Each node has a set of pairwise keys, one for each neighbor.
- o In case of conflict, a node chooses the pairwise key generated by the node with the lower id.

Figure 5 presents the messages exchanged between two neighbors in the pairwise key agreement:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
u -> *, Multicast Message : u.
v -> u, Response Message: v, MAC(Kv, u|v)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Messages Flow of Pairwise Key Agreement

In order to realize the pairwise key agreement, the LEAP option is presented. A LEAP option consists of the following fields:

```
 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                     :
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: Format of the LEAP Response Option

Option Type: 0x0B (to be confirmed by IANA)

Option Length: Variable, length of the option in octets excluding the Type and Length fields.

Comp Algo: 8-bit field. In order to store a short version of the id (IPv6) a collision resistant hash function or the method used in Prefix Information Option(as described in section 6.7.1. of [I-D.ietf-roll-rpl]) can be used. The Compression Algorithm field indicates which (if any) compression algorithm is being used. The Compression Algorithm is encoded as in the table below:

Bit Number	Comp Algo
0x00	No Address
0x01	No Compression
0x02	SHA-1
0x03	SHA-256
0x04	Prefix Information
else	Unassigned

Figure 7: Compression Algorithm

MAC Function: 8-bit field, indicating which MAC function is being used (interested readers should refer to [PseuFun]). The length of the MAC Function is set by the algorithm. The MAC Function is encoded as in the table below:

Bit Number	MAC Function
0	HMAC-SHA-256
1	HMAC-SHA-512
else	Unassigned

Figure 8: MAC Function

Response MAC: The message authentication code is computed on the address of the sender, address of the recipient, and the key of the sender.

Compressed Address: Indicates the compressed IPv6 destination address. The sender truncates the compressed address from the Comp Algo result. The receiver can calculate the Compressed Address length by excluding the comp, MAC, and Response MAC fields from the Option length

The pairwise key establishment can be based on RPL messages, by piggy-backing the key agreement message on RPL messages. Implementers may choose to use the LEAP option on any of the one-hop bi-directional message exchanges done in RPL based on the design considerations of their implementation. Below are lists of design considerations, possible message exchange schemes, and a matrix summarizing which design considerations are covered by each message exchange scheme.

3.2.1.1 Message Design Considerations List

The design considerations are as follows:

- 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.

3.2.1.2 Message Exchange Schemes

The possible message exchange schemes that can be used to implement the key agreement protocol are as follows:

- S1: u -> * DAO Multicast
  v -> u DAO Unicast Ack

```
o  S2:  u -> * DAO Multicast
        v -> u DAO Multicast Ack

o  S3:  u -> * DAO Multicast
        v -> u DAO Multicast

o  S4:  u -> * DIO Multicast
        v -> u DIO Unicast

o  S5:  u -> * DIS Multicast
        v -> u DIO Unicast

o  S6:  u -> * DIS Multicast or DIO Multicast
        v -> u DIO Multicast

o  S7:  u -> * New RPL Base Message
        v -> u New RPL Base Message
```

In case the response message is a Multicast, the sender may add a number of IPv6 addresses. In order to save overhead, any algorithm to compress the addresses can be used, e. g., a collision resistance hash function, the method used in Prefix Information Option. Selecting at least one is mandatory in order to use the LEAP option.

3.2.1.3 Design Consideration vs. Message Exchange Scheme

The following matrix analyzes the design considerations vs. the message exchange schemes. The implementer needs to choose which scheme is most appropriate for its application requirements:

S	MES	RPLM	RPLF	EFFI	STP	BN	NEI	MAND	RELY
S1	DAO-M DAO-MA	+	- #0	+ #1	+	+	+	-	+
S2	DAO-M DAO-MA	- #2	+	+ #0	+	+	+	-	+
S3	DAO-M DAO-M	+	- #3	+ #4	+	+	+	-	+
S4	DIO-M DIO-U	+	- #5	+ #6	+	+	+	+	- #8
S5	DIS-M DIO-U	+	- #7	+ #6	+	+	+	+	-
S6	DIS-M DIO-M	+	+	+ #6	+	+	+	+	- #8
S7	NEW NEW	-	+	+ #9	+	+	+	-	-

Figure 9: Design Consideration vs. Message Exchange

#0 -   Acknowledgement of DAO Multicast required, while the RPL [I-D.ietf-roll-rpl] states that Ack is sent to unicast messages.

#1 -   The number of extra Ack messages is proportional to the number of neighbors. Those messages may potentially cause congestion and collisions.

#2 -   DAO-Multi-Ack is a new type.

#3 -   According to the RPL specification [I-D.ietf-roll-rpl], DAO Multicast is not sent automatically as a response to DAO Multicast.

#4 - The number of extra DAO Multicast messages is proportional to the number of neighbors. This number can be reduced with longer aggregated messages.

#5 - According to the RPL [I-D.ietf-roll-rpl], DIO Unicast is not sent automatically to DIO Multicast.

#6 - The number of extra DIO messages has an order of magnitude of the number of neighbors. Compared to other base messages, the length of a DIO message is longer.

#7 - According to the RPL [I-D.ietf-roll-rpl], the response message to DIS Multicast is DIO Multicast and not DIO Unicast.

#8 - Part of the DODAG construction.

#9 - The number of the extra new RPL messages is proportional to the number of neighbors.

For example, if S6 (using DIS Multicast and DIO Multicast) is selected for implementation, the following apply:

1. Each node periodically sends a DIS message before joining the DODAG (as described in section 17.2.1.1. of [I-D.ietf-roll-rpl]).

2. A non-DODAG node, a node that is not part of the DODAG, when receiving a DIS message, MUST ignore the message.

3. A non-DODAG node, when receiving a DIO message, follows the RPL.

4. A DODAG node, when receiving a DIS or DIO message during the Trickle interval, checks whether a pairwise key exists with the sender.

    4a. If not, the node adds a new LEAP option with the compressed address to its next DIO message, and copies the pairwise key it generates. The node also initializes a retransmission value, a maximum number each node will try to retransmit to a neighbor (can be different for different neighbors).

    4b. If a pairwise key exists, the node checks the retransmission value.

        I. If the retransmission value is greater than zero, the node adds a LEAP option to its next DIO message.

        II. Otherwise, it does not add a LEAP option.

        III. It always decreases the retransmission value.

3.2.2 Cluster Key

This section describes a cluster key agreement procedure based on the LEAP algorithm [LEAP]. This section does not provide results on LEAP’s performance or behavior, nor does it explain the algorithm’s design in detail. Interested readers should refer to [LEAP].

The cluster key establishment phase follows the pairwise key establishment phase. The cluster key agreement has the following steps:

- o Node u first generates a random key.
- o For each neighbor, node u encrypts this random key with the neighbor’s pairwise key.
- o For each neighbor, node u sends the encrypted random key.

The cluster key agreement can be realized with RPL messages; any RPL Unicast message is OPTIONAL. For example, a node sends a DAO unicast message with a Cluster Key Option that can carry the cluster key encrypted to each neighbor.

In order to generate a cluster key, an RPL message MUST carry a "Cluster Key" option. A Cluster Key option consists of the following fields:

```
 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                     :
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10: Format of the LEAP Cluster Key Option

Option Type: 0x0C (to be confirmed by IANA)

Option Length: Variable, length of the option in octets excluding the Type and Length fields.

Key Length: Variable, length of the Encrypted Cluster Key in octets.

ENC Function: 8-bit field, indicating which encrypted function is being used. The ENC Function is encoded as in the table below:

Bit Number	ENC Function
0	CCM with AES-128, M=0
else	Unassigned

Figure 11: Encryption Function

Encrypted Cluster Key: The encrypted value of the cluster key computed on the random key and the neighbor pairwise key.

4 Security Considerations

The security mechanisms in this standard extend the RPL security mechanisms, sections 6.1 and 10 of [I-D.ietf-roll-rpl]. Therefore, the security consideration described in section 18 of [I-D.ietf-roll-rpl] exists in this document. The scope of the current RPL security services is the link; the authenticity of the messages sent by the DODAG root relies on the trustworthiness of all intermediate nodes and the fact that none of the keys are compromised. The herein proposed DIO Message Authentication extends the data integrity and data origin authentication [RFC3552] into network level, by authenticating the static fields of the DIO message for all nodes in the DODAG.

The security mechanisms in RPL [I-D.ietf-roll-rpl] are based on symmetric-key and public-key cryptography, and use keys that are to be provided by higher/lower layer processes. However, the establishment and maintenance of these keys are out of the scope of the current RPL. The proposed local key agreement gives new procedures in order to establish and maintain pairwise and cluster keys for peer entity authentication [RFC3552]. The cryptographic protection using pairwise and cluster keys allows some flexibility and application specific tradeoffs between key storage and key maintenance costs versus the cryptographic protection provided.

The security services in this document are based on symmetric-key and public-key cryptography and assume a safe time interval after bootstrapping, during which an attacker cannot compromise a node.

The current RPL security services [I-D.ietf-roll-rpl] assume that a node wishing to join a secured network has been preconfigured with a shared key; for example, each node MAY use a secure message with KIM=0. Moreover, to join a secure RPL network, a node either listens for secure DIO messages or triggers secure DIOs by sending a secure DIS.

5 IANA Considerations

5.1 RPL Control Message Option

IANA is requested to create a registry for the RPL Control Message Options.

New values may be allocated only by an IETF Review. Each value should be tracked with the following qualities:

- o Value
- o Capability description
- o Defining RFC

The following bits are currently defined:

Value	Description	Reference
0x0A	Broadcast Authentication	This document
0x0B	LEAP Response	This document
0x0C	Cluster Key	This Document

RPL Control Message Options

5.2 New Registry for the Hash Value Type

IANA is requested to create a registry for the Hash Value Type Field, which is contained in the Broadcast Authentication option.

New values may be allocated only by an IETF Review. Each value should be tracked with the following qualities:

- o Value

- o Capability description

- o Defining RFC

The following bits are currently defined:

Value	Hash Value Type	Reference
0	No hash Value	This document
1	Hash Root Chain Value	This document
2	Current Hash Chain Value	This document
3	Unassigned	This document

Hash Field in Broadcast Authentication Option

5.3 New Registry for the Security Algorithm Type

IANA is requested to create a registry for the Security Algorithm Field, which is contained in the Broadcast Authentication option.

New values may be allocated only by an IETF Review. Each value should be tracked with the following qualities:

- o Value

- o Capability description

- o Defining RFC

The following bits are currently defined:

Value	Security Algorithm	Reference
0x00	No Security Algorithm	This document
0x01	SHA-256	This document
0x02	SHA-512	This document
0x80	HMAC-SHA-256	This document
0x81	HMAC-SHA-512	This document
0xC0	RSA with SHA-256	This document
0xC1	ECC-SECP256K1 with SHA-256	This document
else	Unassigned	This document

Security Algorithm Field in Broadcast Authentication Option

5.4 New Registry for the Comp Algo Type

IANA is requested to create a registry for the Comp Algo Field, which is contained in the LEAP Response Option.

New values may be allocated only by an IETF Review. Each value should be tracked with the following qualities:

o Value

o Capability description

o Defining RFC

The following bits are currently defined:

Value	Comp Algo	Reference
0x00	No Address	This document
0x01	No Compression	This document
0x02	SHA-1	This document
0x03	SHA-256	This document
0x04	Prefix Information	This document
else	Unassigned	This document

Comp Algo Field in LEAP Response Option

5.5 New Registry for the MAC Function Type

IANA is requested to create a registry for the MAC Function Field, which is contained in the LEAP Response Option.

New values may be allocated only by an IETF Review. Each value should be tracked with the following qualities:

- o Value
- o Capability description
- o Defining RFC

The following bits are currently defined:

Value	MAC Function	Reference
0	HMAC-SHA-256	This document
1	HMAC-SHA-512	This document
else	Unassigned	This document

MAC Function Field in LEAP Response Option

5.6 New Registry for the ENC Function

IANA is requested to create a registry for the ENC Function Field, which is contained in the LEAP Cluster Key Option.

New values may be allocated only by an IETF Review. Each value should be tracked with the following qualities:

- o Value
- o Capability description
- o Defining RFC

The following bits are currently defined:

Value	ENC Function	Reference
0	CCM with AES-128, M=0	This document
else	Unassigned	This document

ENC Function Field in LEAP Cluster Key Option

6 Acknowledgements

The research leading to these results has received funding from the European Community’s Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 225186. The authors would also like to acknowledge the review and comments from Yoav Ben Yehezkel.

7 References

7.1 Normative References

[I-D.ietf-roll-rpl]
Winter, T., Thubert, P., Brandt, A., Clausen, T., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., and J. Vasseur, "RPL: IPv6 Routing Protocol for Low power and Lossy Networks", draft-ietf-roll-rpl-17 (work in progress), December 2010.

[RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3552] Rescorla E., and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, March 2003.

[RFC3447] Jonsson, J., and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.

7.2 Informative References

[I-D.ietf-roll-terminology] Vasseur, J., "Terminology in Low power And Lossy Networks", draft-ietf-roll-terminology-04 (work in progress), September 2010.

[I-D-roll-security-framework] Tsao T., Alexander R., Dohler M., Daza V., and A. Lozano, "A Security Framework for Routing over Low Power and Lossy Networks", |%draft-ietf-roll-security-framework-03, (work in progress) December 2010.

[RFC5826] Brandt, A., Buron, J., and G. Porcu, "Home Automation Routing Requirements in Low-Power and Lossy Networks", RFC 5826, April 2010.

[RFC5867] Martocci, J., De Mil, P., Riou, N., and W. Vermeylen, "Building Automation Routing Requirements in Low-Power and Lossy Networks", RFC 5867, June 2010.

[RFC5548] Dohler, M., Watteyne, T., Winter, T., and D. Barthel, "Routing Requirements for Urban Low-Power and Lossy Networks", RFC 5548, May 2009.

[RFC5673] Pister, K., Thubert, P., Dwars, S., and T. Phinney, "Industrial Routing Requirements in Low-Power and Lossy Networks", RFC 5673, October 2009.

[RFC4949] R. Shirey, "Internet Security Glossary", RFC 4949, FYI 36, August 2007.

[RFC3610] Whiting, D., Housley, R., and N. Ferguson, "Counter with CBC-MAC (CCM)", RFC 3610, September 2003.

[RFC4868] Kelly, S., and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-

384, and HMAC-SHA-512 with IPsec", RFC 4868, May 2007.

[PseuFun] Goldreich, O., Goldwasser, S., and S. Micali, "How to Construct Random Functions", Journal of the ACM, Volume 33, Number. 4, 1986, pp 210-217.

[L1981] Lamport L., "Password Authentication with Insecure Communication", ACM Journal of Communications Volume 24 Issue 11, pp 770-772, Nov. 1981.

[LEAP] Zhu, S., Setia, S., and S. Jajodia, "LEAP: efficient security mechanisms for large-scale distributed sensor networks ", ACM conference on Computer and communications security, pp. 62-72, 2003.

[SECG2] D. R. L. Brown, "Standards for Efficient Cryptography Group (SECG), "SEC 2: Recommended Elliptic Curve Domain Parameters version 2.0", Version 2.0, January 2010.

[OptHash] Don, C., and M. Jakobsson, "Almost Optimal Hash Sequence Traversal", Fourth Conference on Financial Cryptography, 2002.

[FIPS180] National Institute of Standards and Technology, "FIPS Pub 180-3, Secure Hash Standard (SHS)", US Department of Commerce , February 2008, <http://www.nist.gov/itl/upload/fips180-3_final.pdf>.

Authors' Addresses

Amit Dvir
Laboratory of Cryptography and Systems Security (CrySyS)
Budapest University of Technology and Economics
BME-HIT, PO Box 91, 1521 Budapest
Hungary

EMail: azdvir@gmail.com

Tamas Holczer
Laboratory of Cryptography and Systems Security (CrySyS)
Budapest University of Technology and Economics
BME-HIT, PO Box 91, 1521 Budapest
Hungary

EMail: tamas.holczer@crysys.hu

Laszlo Dora
Laboratory of Cryptography and Systems Security (CrySyS)
Budapest University of Technology and Economics
BME-HIT, PO Box 91, 1521 Budapest
Hungary

EMail: laszlo.dora@crysys.hu

Levente Buttyan
Laboratory of Cryptography and Systems Security (CrySyS)
Budapest University of Technology and Economics
BME-HIT, PO Box 91, 1521 Budapest
Hungary

EMail: buttyan@crysys.hu