ARPD: Asynchronous random key predistribution in the LEAP framework for Wireless Sensor Networks

Andreas Achtzehn, Christian Rohner and Ioana Rodhe
Department of Information Technology,
Uppsala University, Box 337, SE-751 05 Uppsala
Email: andreas.achtzehn@rwth-aachen.de
{christian.rohner,ioana.ungurean}@it.uu.se

Abstract

In the LEAP framework for wireless sensor networks a set of keys is used to secure communication. LEAP distinguishes between unicast (pairwise) communication, group (cluster) communication and global (broadcast) communication. The keys used in pairwise communication are derived from an initial key K_I that nodes are equipped with prior to deployment and that is deleted after link setup. Further keys are distributed encrypted with these pairwise keys. It is a weakness that, if the initial key is ever disclosed, the whole network is compromised.
To lower the threat of K_I disclosure, we present a K_I-less scheme for key predistribution. Our scheme is based on random key predistribution, and proves to perform better in medium sized networks than previous proposals. It is resilient against node capture attacks and allows node to node authentication. Attacks against overlying protocols in the network are more difficult with this scheme.
We have conducted computations to show the feasibility of our scheme for networks up to a size of 1000 nodes. By introducing a key reuse system we are able to increase the probability of a successful link setup. We have included a security analysis that discusses our scheme's resistance against commonly known attacks.

1 Introduction

Protocols designed for sensor networks have to work securely in hostile environments. Encrypted communication is therefore necessary. Sensors nodes, though, are usually low power devices with limited memory capacity and low computation power. Processors used in sensor nodes fulfil the requirements for public key encryption [6], but energy consumption due to extensive calculations significantly decreases a node's lifetime [2]. Therefore, symmetric key encryption and authentication is often applied in wireless sensor networks.
The LEAP framework [13] distinguishes between pairwise communication, cluster communication, group communication, and communication between each node and the base station. Symmetric keys for each message class are derived with the help of a key K_I that is deleted after initial deployment. LEAP is secure if an adversary cannot derive K_I. If the actual time a node needs to establish pairwise keys after deployment in the hostile environment is smaller than the time an attacker needs to capture a node then a LEAP network is secure.
In the life cycle of a WSN it becomes necessary to add new nodes to the network. Nodes cease to operate due to empty batteries, electrial or mechanical failure and need replacement [12].
In LEAP, K_I is present in the hostile environment every time new nodes are added. This increases the threat of an attacker to retrieve K_I. The impact of such disclosure is significant. With K_I in his possession, an attacker can derive all pairwise keys and decrypt all unicast communication in the network. This is not limited to communication taking place after K_I is captured, all encrypted communication can be decrypted retroactively. Other keys can be derived under certain circumstances.
We have derived from our observation that protection of K_I is essential in the LEAP framework. We have identified node additions as a critical phase in a WSN's life cycle. Hence, we focus on developing a replacement for the keying scheme LEAP uses in the node addition phase.
We have studied the random pairwise keys scheme (*RPK*) proposed by Chan et al. [3] as an alternative to the basic LEAP algorithm. The scheme perfectly preserves the secrecy of the network in case of node capture as well as allowing node to node authentication. But, since node relations are predetermined, *RPK* limits the network size and the number of node additions. We have reviewed Chan

1-4244-1455-5/07/$25.00

et al.'s scheme and developed it into a new scheme.

Contribution. We introduce an asynchronous random key predistribution scheme called *ARPD* to be used in node addition phases instead of the basic LEAP keying scheme. It does not make use of K_I, therefore the network remains secure even if an attacker is able to capture a node. Like RPK, it provides *perfect resilience against node capture*, limiting network communication disclosure solely to communication including the captured node.
We show that ARPD does not limit the number of node addition phases and allows a controller to dynamically change the network's size.

Organization. The rest of this paper is organized as follows. Section 2 covers the basic LEAP algorithm as well as an introduction to the random keying scheme by Chan et al. In section 3 we present our new approach of establishing pairwise shared keys called *asynchronous random pairwise key distribution*(ARPD). Section 4 contains a discussion on the performance issues that arise in our protocol. A security analysis to evaluate robustness to various attacks against our protocol can be found in section 5. We complete this paper in Section 6 with conclusions.

2 Related work and background

In this section we will briefly study pairwise key establishment as the most important mechanism in the LEAP protocol.
To compare keying schemes more easily we will define four distinct phases in the life cycle of a wireless sensor network. They are described as *settling phase*, *pairwise key establishment phase*, *cluster key establishment phase*, and *mission phase*.
Because ARPD is based on ideas in random pairwise keying, we will continue in the later part of this section with a study of the random pairwise scheme of Chan et al.

2.1 Pairwise key establishment in LEAP

In sensor networks, it is unknown prior to deployment which two nodes will be able to communicate directly with each other. LEAP evades this obstacle by applying an algorithm for pairwise key establishment that makes it possible to build a secure link between any two nodes in the network. A commonly known secret, the initial key K_I, is necessary to exchange key credentials. The basic algorithm works as follows:

1. **Predistribution.** The controller generates the initial key K_I and stores it in every node. Each node u generates its master key K_u using K_I, so that $K_u = f_{K_I}(u)$. f is a secure pseudo-random function [5].
2. **Settling.** All nodes are spread randomly in the deployment area. Each node waits for a predefined time to assure that all nodes are in their final position. We refer to this as the *settling phase*.
3. **Neighbour Discovery.** The *pairwise key establishment phase* begins with a discovery message sent by each node u to reveal all nodes in its vicinity. Nodes reply to this message with an authenticated acknowledgement message. Verification is possible, since at this stage node u still holds K_I and can therefore generate K_v for each replying node v. We symbolize concatenation of values a and b as $a \mid b$. Node v uses a secure keyed hash function [9] $MAC(K, m)$ where K is the key and m is the message to authenticate.

$$\begin{aligned} u &\longrightarrow * : & u \\ v &\longrightarrow u : & v, MAC(K_v, u \mid v). \end{aligned}$$

4. **Pairwise Key Derivation.** Node u and node v generate their mutual pairwise key K_{uv} without further communication.

$$K_{uv} = f_{K_v}(u).$$

5. **Initial Key Deletion.** After successful establishment of secure connections with the neighbouring nodes, node u deletes the initial key K_I from its memory.

The pairwise key establishment sketched out above is followed by further key establishments which are outside the scope of this paper. Having concluded key establishment, the network goes into the *mission phase*.
Some nodes fail during the mission phase. To maintain functionality, the operator has to replace those nodes. In LEAP, new nodes execute the same algorithm the original population of nodes in the network used. They also carry K_I in their memory prior to finishing key establishment.

2.1.1 LEAP security

A number of security threats can be identified in this keying scheme. Most important, if K_I becomes known to an attacker at any time, he can derive any pairwise key between two nodes. No backward confidentiality is provided. So, encrypted traffic recorded earlier can be decrypted (including key establishment traffic).
Another threat can be identified in the ability of the attacker to arbitrarily add nodes to the network if in possession of K_I, influencing information retrieval based on majority votes [11]. Also, any node can be impersonated, opening the network to Sybil attacks [4].

A promising approach to lower the risk of K_I disclosure is shortening the time the key is available in the deployment area. In [14], Zhu et al. have addressed this issue by shortening the time, K_I is valid. They use a different K_I^t for each interval t in the lifetime of the sensor network. The advantage is obvious, as that disclosure of an initial keys valid for interval t doesn't allow for arbitrary node additions and supports *weak backward and forward confidentiality*. In opposite to the basic scheme with only one K_I the extended scheme needs additional measures to invalidate old initial keys, limits the number of node additions and increases memory consumption. Therefore, we propose replacing the node addition algorithm by a variant, that works without having any initial key stored in the nodes.

2.2 Random pairwise key predistribution

A possible substitute for the LEAP node addition algorithm is the random pairwise key predistribution scheme (RPK) of Chan, Perrig, and Song [3]. Each node is equipped with a set of keys that correspond to a subset of nodes in the network. If a node is later deployed in another nodes' neighbourhood, they can build a pairwise connection with the preloaded key. The practicality of this approach can be shown by applying random graph theory. Erdös and Rényi showed in the late 1950's that in order to have a connected graph for a number of nodes, each two nodes have to be connected only with a certain probability.

RPK has some salient features. It allows *node to node authentication*, so any node can ascertain the identity of the nodes that it is communicating with. Another feature of RPK is the perfect resilience against node capture. We define it as a property of the protocol that through the capture of any node solely communication including the captured node is revealed.
But, RPK bears two limitations that make it difficult to use in the LEAP framework. We assume that the controller can't change keying material in nodes that are already deployed. Also, we assume that the controller doesn't know which nodes have failed before he deploys new nodes. Therefore, the network size is predetermined since keying material for new nodes has to be stored beforehand. The number of node additions is consequently limited, since, if all node identities have been used, the network can't be further "refreshed" with new nodes.

3 ARPD for node additions

In this section we will present our new key establishment protocol called *asynchronous random pairwise key distribution*(ARPD) that evades the limitations on network addition and size in RPK while offering node to node authentication and perfect resilience against node capture. ARPD uses no common initial key, eliminating this single point of exploitation in LEAP.
We have designed ARPD to be used to add new nodes to the network. Some nodes might have ceased to work or there is a need to increase network size or density. Keying for the initial deployment of the network was conducted for example by using the basic LEAP pairwise keying algorithm as presented in section 2.1.
Our scheme works as follows:

1. **Predistribution.** The controller generates an individual generation key $K_u^{ps} = f_{K^{ps}}(u)$ and stores it in node u. Note that this key is a function of the node identifier u and a master key K^{ps} that is to be kept secret by the controller. Every node, including those that have been brought out in the initial deployment phase hold an individual generation key.
   In addition to the node's individual generation key, the controller stores m pairwise shared keys in the new node. These keys are used for establishing secure links with already deployed nodes. They are generated by applying the node identifier u to a function of the individual generation key of the appropriate node. Imagine that we want node u to be able to establish a secure link with the already deployed node v. The controller stores $K_{uv} = f_{K_v^{ps}}(u)$ in node u's key ring.

2. **Settling.** The new nodes are brought into the deployment area. They are spread, for example, by aerial scattering. Each node waits for a predefined time to assure that it is in its final position.

3. **Neighbour Discovery.** A discovery message sent out by each node u reveals all nodes in its vicinity. Node v generates an authenticated reply by sending u's and v's identifiers. Node v can generate K_{uv} using K_v^{ps}

$$\begin{aligned} u &\longrightarrow * : & u \\ v &\longrightarrow u : & v, MAC(K_{uv}, u \mid v). \end{aligned}$$

4. **Node Authentication.** Node u can verify v's message if it was equipped with K_{uv} in advance. As a reply, it sends an authenticated message in which the node identifiers are inverted. This enables node v to verify if u holds K_{uv}.

$$u \longrightarrow v : u, MAC(K_{uv}, v \mid u)$$

5. **Key Deletion.** Node u erases all pairwise keys it hasn't used during step 4.

The algorithm provided above yields secure links with all neighbours node u shares a key with. To connect the remaining immediate neighbours, other methods like *multipath reinforcement* [1] have to be applied. Multipath reinforcement needs at least two neighbours to already be connected to the node.

4 Performance analysis

In a random scheme like ARPD, a node can only connect with a certain probability to its neighbours. In this section we study this probability and the limitations introduced through random keying and present a key reuse scheme to extend the usability of our scheme.

4.1 Section notation and assumptions

Throughout this section we use the following notation: n is the number of nodes in the network, n' is the average number of nodes within a node's communication range. Each node stores m keys, k is the number of distinct keys in a node's neighbourhood.
We study the feasability of our scheme up to a network size of 1000 nodes. As suggested in [7] where Hwang and Kim compare keying schemes, we have examined sparse distributions of down to 10 neighbour nodes.

4.2 Probability of connectivity

Our scheme limits the network size by the size of the memory in a single node. This is because a node has to store a certain amount of keys in its key ring before deployment to connect to at least two neighbours in its later deployment environment.
In order to do multipath reinforcement, a node has to share keys with at least two neighbours. Based on the number of neighbours and the number of keys stored in the node, this probability is

$$p^{multi} = 1 - \frac{n + k(m-1) - m + 1}{n - k - m + 1} \prod_{i=0}^{n'-1} \frac{n - m - i}{n - 1}$$

The probability to connect drops as expected if the size of the network is increased.

4.2.1 Key reuse

To increase the maximum network size in a memory constrained scenario and still keep the connection probability, we propose to reuse generation keys and thereby reduce the number of unique keys in the network. We define $K_u^{cs} = f_{K^{ps}}(\lfloor \frac{u}{r} \rfloor)$ as the *generation cluster key* where r denotes how many nodes share this key (*reuse factor*). Naturally, if one of the nodes storing this key is compromised, the links of all nodes with the same key get compromised. We have concentrated on reuse factors r of 2 and 3 for the network size assumed above.

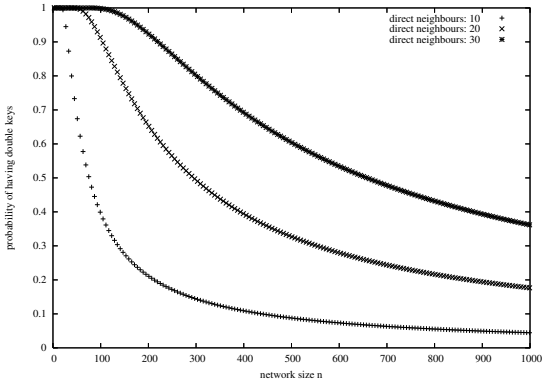

Figure 1. Probability to have at least one key twice in the physical neighbourhood.

reuse factor 2. The probability to have x keys twice in the physical network neighbourhood can be calculated to

$$p_2^{overlap}(x) = \begin{cases} \dfrac{\binom{\lfloor \frac{n}{2} \rfloor}{x} \binom{\lfloor \frac{n}{2} \rfloor - x}{n' - 2x} 2^{n'-2x}}{\binom{n}{n'}} & \text{if } x > n' - \frac{n}{2} \\ 0 & \text{else} \end{cases}$$

As one can see from Figure 1, the probability to have more than one node with the same K_u^{cs} nearby is especially high for smaller networks. It therefore needs anticipation. The number of expected keys in the physical neighbourhood of a node is

$$k = \sum_{x=0}^{\lfloor \frac{n'}{2} \rfloor} p_2^{overlay}(x) * (n' - x)$$

reuse factor 3. Similar to the calculation done above, we can derive the probability to have x triples (three nodes sharing the same key) and y doubles (two nodes sharing the same key) as

$$p_3^{overlap}(x,y) = 3^{n'-3x-y} \frac{\binom{\lfloor \frac{n}{3} \rfloor}{x} \binom{\lfloor \frac{n}{3} \rfloor - x}{y} \binom{\lfloor \frac{n}{3} \rfloor - x - y}{n' - 3x - 2y}}{\binom{n}{n'}}$$

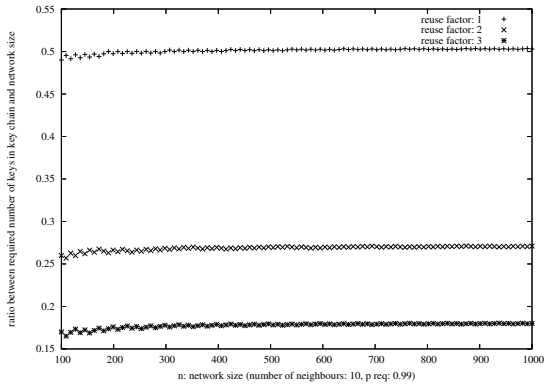

Figure 2. Ratio between the number of keys to be stored in the key ring of a new node and the size of the network if a connection probability of at least 99 percent is to be guaranteed.

if $n' - \frac{n}{3} < y - 2x$, 0 otherwise. The expected number of keys k is

$$k = \sum_{x=0}^{\lfloor \frac{n'}{3} \rfloor} \sum_{y=0}^{\lfloor \frac{n'-3x}{2} \rfloor} p_3^{overlap}(x,y)(n' - 2x - y)$$

4.2.2 Choice of reuse factor

We have shown in section 4.2.1 how we can improve the scheme by introducing a key reuse scheme. We will now look closer at how many keys we have to store in the newly deployed node.
The probability of having at least two keys, thus being able to do multipath reinforcement, is

$$p_r^{multi} = 1 - \frac{\lceil \frac{n}{r} \rceil + k(m-1) - m + 1}{\lceil \frac{n}{r} \rceil - k - m + 1} \prod_{i=0}^{k-1} \frac{\lceil \frac{n}{r} \rceil - m - i}{\lceil \frac{n}{r} \rceil - 1}$$

taking the reuse factor r into account. Figure 2 shows the ratio between the number of keys to be stored in a node to guarantee a 99 percent connection probability and the size of the network. It is particulary interesting to see that for large network sizes this ratio is non-varying. Therefore, the maximum size of a wireless sensor network that uses *ARPD* as a keying scheme grows linearly with the size of memory provided by a single node.

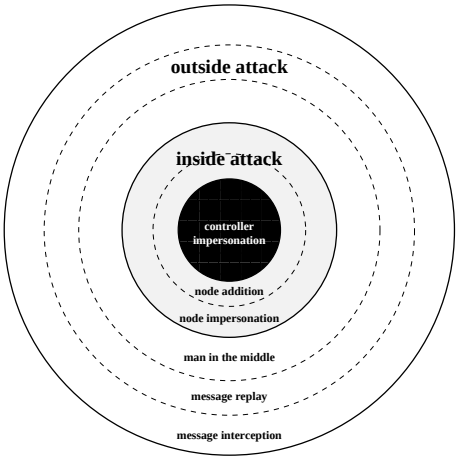

Figure 3. A security model for WSNs.

5 Model-driven security analysis

In this section we will analyze the most commonly used attacks against WSNs and estimate their impact on a ARPD employing LEAP network. We apply an intuitive model to outline the impact of each security threat, which we present first.

5.1 Threat model

A first basic differentiation of attacks can be made looking at the prerequisite of node capture. In our security model we differ between *outside attacks* and *inside attacks* to reflect the aspect of node capture, one of the most common threats. Inside attacks are considered to be more harmful to overall network security, since at that time the attacker can already gain access to some information stored in the network. Also, if the attacker has gained access to the inside of the network, he can raise attacks against overlying protocols. With no further argument, if the network controller is under the control of the attacker, the network has to be considered completely insecure.
We incorporate this observation in a graphical representation of our security model in Figure 3. The closer a successfully carried out attack is to the center of the circle, the more information in the network is revealed or can even be altered.
Outside attacks can further be broken down into three levels of severity. Message interception is at the lowest level, followed by message replay and at the highest level one can find man in the middle attacks. In inside attacks, our model focuses on attacks against communication in the network. Of course, if a node is compromised, it can influence other nodes on higher layers of the application. In order to in-

filtrate higher levels of the application running on the network's nodes, the attacker needs to be able to either gain full control over a node or add new nodes to the network. Aggregation and agreement protocols still perform correctly in presence of a limited number of malicious nodes. Still, these protocols have to be adjusted to the security performance of the underlying keying mechanism.

5.2 Attack evaluation

Our scheme secures the network at link layer level against inside and outside attacks. In this section we therefore conduct experiments on this level and evaluate their impact.

Message injection during node addition. An attacker can send neighbour discovery messages to nodes since these messages are not encrypted or authenticated. If sensors use techniques to save energy by powering down components, the advantage of these techniques can't be used anymore. This *sleep-deprivation attack* is among others studied in [10].

Some approaches can be made to lower the impact. Most of them include setting thresholds so the nodes stop processing messages. Unfortunately, this opens the network for denial-of-service attacks.

Forged acknowledgements. Our keying scheme is robust to forged acknowledgements. If the attacker forges the reply of an already deployed node to the new node, the new node will discard it due to the wrong MAC.

Wormhole and sinkhole attacks. Wormhole and sinkhole attacks have been studied in [8]. Building a wormhole between to adjacent parts of the network is impossible once nodes are deployed since they drop remaining keying material. No increased message retrieval due to forged low-latency links can therefore be achieved.

Message replay. ARPD doesn't protect against message replay. Further measures like introducing nonces have to be taken at higher layers.

6 Conclusions

We have presented ARPD, an asynchronous pairwise key establishment scheme based on random keying. To proof its functionality, we have applied it to node addition phases in the LEAP framework. ARPD decreased the overall threat time, the time an attacker can potentially derive K_I, to a constant.

We conducted a number of calculations on the connection probability in an ARPD environment. We showed that due to the limitation in memory, it might become necessary to employ key reuse. We have presented a simple method to distribute keys among several nodes, so called generation clusters, and make it easy to the newly deployed node to identify cluster affiliation. Our calculations indicated that key reuse can significantly improve memory consumption.

The security analysis introduced a security model to make the impact of different varieties of attacks comparable. We split up the analysis into an analysis of inside and outside attacks. Our studies showed that the ARPD protocol is resilient against any kind of outside attacks. The impact of key disclosure in an inside attack is limited to keys that are used solely in communication with the captured node. Therefore, the key framework remained perfectly secure in node capture attack.

References

[1] R. Anderson, C. Haowen, and A. Perrig. Key infection: Smart trust for smart dust, 2001.

[2] D. W. Carman, P. S. Kruus, and B. J. Matt. Constrains and approaches for distributed sensor network security. Technical report, NAI Labs, September 2000.

[3] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, page 197, Washington, DC, USA, 2003. IEEE Computer Society.

[4] J. Douceur. The sybil attack, 2002.

[5] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[6] N. Gura, A. Patel, A. Wander, H. Eberle, and S. Shantz. Comparing elliptic curve cryptography and rsa on 8-bit cpus. *Lecture Notes in Computer Science*, 2004.

[7] J. Hwang and Y. Kim. Revisiting random key predistribution schemes for wireless sensor networks. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 43–52, New York, NY, USA, 2004. ACM Press.

[8] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 113–127, May 2003.

[9] H. Krawczyk, M. Bellare, and R. Canetti. Hmac: Keyed-hashing for message authentication, 1997.

[10] J. Krishnaswami. Denial-of-service attacks on battery-powered mobile computers, 2003.

[11] L. Patrick and R. John. A formally verified algorithm for interactive consistency under a hybrid fault model. Technical report, 1993.

[12] R. Szewczyk, J. Polastre, A. M. Mainwaring, and D. E. Culler. Lessons from a sensor network expedition. In H. Karl, A. Willig, and A. Wolisz, editors, *EWSN*, volume 2920 of *Lecture Notes in Computer Science*, pages 307–322. Springer, 2004.

[13] S. Zhu, S. Setia, and S. Jajodia. Leap: efficient security mechanisms for large-scale distributed sensor networks. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 62–72, New York, NY, USA, 2003. ACM Press.

[14] S. Zhu, S. Setia, and S. Jajodia. Leap+: Efficient security mechanisms for large-scale distributed sensor networks. *ACM Trans. Sen. Netw.*, 2(4):500–528, 2006.