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. If the initial key is ever disclosed, the whole network is compromised.**

To lower the threat of K_I disclosure, we present a novel 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.

***Index Terms*—Wireless sensor networks, Network-level security and protection**

I. INTRODUCTION

Protocols designed for sensor networks have to work securely in a hostile environment like a battle field. 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 [1], 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 [3] 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 [4].

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. [5] as a candidate for replacing 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 et al.'s scheme and developed it into a new scheme.

Contribution. We introduce an asynchronous random key predistribution scheme called *ARPD* that can be used in node addition phases instead of the basic LEAP keying scheme. It needs no initial key 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 II covers the basic LEAP algorithm as well as an introduction to the random keying scheme by Chan et al. In section III we present our new approach of establishing pairwise shared keys called *asynchronous random pairwise key distribution*(ARPD). Section IV contains a discussion on the performance issues that arise in our protocol. A security model to evaluate robustness to various attacks against our protocol can be found in section V. We complete this paper in Section VI with conclusions.

II. RELATED WORK

In this section we will briefly study pairwise key establishment in the LEAP protocol. Pairwise key establishment is essential since all other keys are transmitted over the medium through pairwise communication.

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.

A. 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 [6].
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 out 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 [7] $MAC(K, m)$ where K is the key and m is the message to authenticate.

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

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.

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 [8]. Also, any node can be impersonated, opening the network to Sybil attacks [9].A promising approach to lower the risk of K_I disclosure is shortening the time the key is available in the deployment area. In [10], 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 support *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.

B. 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 [5]. 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 so 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.

III. 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 limitation 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 II-A.

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}

   $$\begin{aligned} u \longrightarrow * : &\quad u \\ v \longrightarrow u : &\quad v, MAC(K_{uv}, u \mid v). \end{aligned}$$
4) **Node Authentication.** Node u can verify v's message if it was equipped with K_{uv} in advance. As a reply, it sends and 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* [11] have to be applied. Multipath reinforcement needs at least two neighbours to already be connected with the node.

IV. 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.

A. Section notation and assumptions

Throughout this section we use the following notation:

n	number of nodes in the network
n'	average number of nodes within a node's communication range
m	number of pairwise keys in a node's key ring
k	expected number of mutual keys in the neighbourhood
r	key reuse factor
p^{multi}	probability to be able to establish a sufficient number of connections to do multipath reinforcement
$p_2^{overlap}(x)$	probability for a node to have x keys twice in the neighbourhood if a reuse factor of 2 is applied
$p_3^{overlap}(x, y)$	probability for a node to have x keys twice and y keys three times in the neighbourhood if a reuse factor of 3 is applied

We study the feasability of our scheme up to a network size of 1000 nodes. As suggested in [12] where Hwang and Kim compare keying schemes, we have examined sparse distributions of down to 10 neighbour nodes.

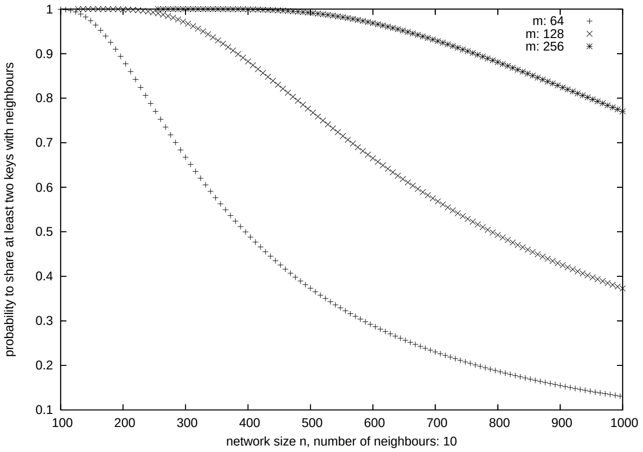
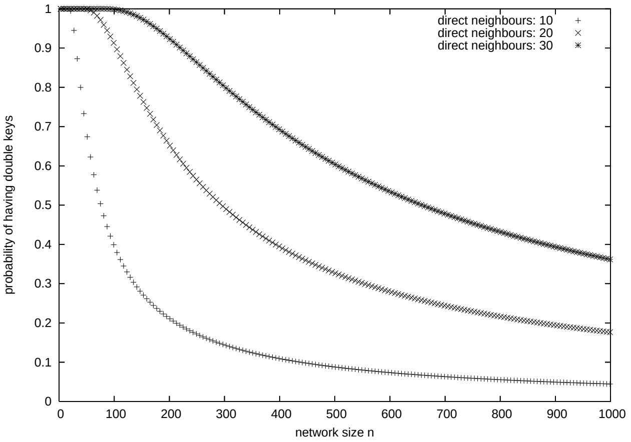
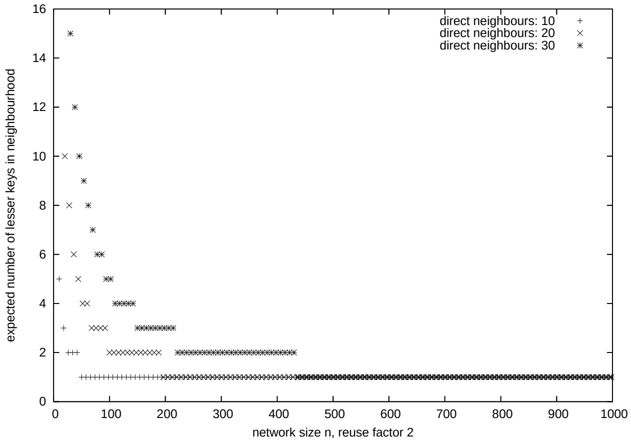


Fig. 1. Probability to share enough keys with vicinity to perform multipath reinforcement. Each node has ten neighbours, probability is sketched for key rings of 64,128, and 256 keys.



(a) Probability to have at least one key twice in the physical neighbourhood.



(b) Excepted number of key losses.

Fig. 2. Changes in neighbourhood in an environment with reuse factor 2.

B. 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

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

Figure 1 shows p^{multi} for selected key ring sizes. The probability to connect drops as expected if the size of the network is increased.

1) Key reuse: To increase the maximum network size in a memory contrained 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 will call a group of nodes that share the same generation key a *generation cluster*, consequently the generation key is furthermore to be called *generation cluster key*.

We define $K_u^{cs} = f_{K^{ps}}(u \mod r)$ as the *generation cluster key* where r denotes how often a key is assigned to nodes in the network (*reuse factor*).

Key reuse comes at the cost of exposing parts of the network in case of node capture. If one of the nodes of a generation cluster is compromised, the links of all nodes in that cluster get compromised.

We have concentrated on reuse factors r of 2 and 3 for the network size assumed above.

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

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

As one can see from Figure 2(a), the probability to have more than one member of a *generation cluster* in the physical neighbourhood is especially high for smaller networks. The number of expected keys in the physical neighbourhood of a node is

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

We show in Figure 2(b) that in a small size neighbourhood like ten neighbours, the expected number of keys in a worst case approximation is almost equal to the number of neighbours minus one if the network consists of more than 48 nodes. For this expected number of keys to be achieved in a denser network with 20 nodes in the vicinity, the network size should be greater than 192 nodes. In case of 30 neighbouring nodes, the network should be larger than 436 nodes.

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

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

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

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

2) Choice of reuse factor: We have shown in section IV-B1 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 in order to connect it to the network with high probability and in how far the reuse factor influences this choice.

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 into account. Figure 3 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.

V. 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 have developed a model to outline the impact of each security threat, which we present first.

A. A security threat model for WSNs

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.

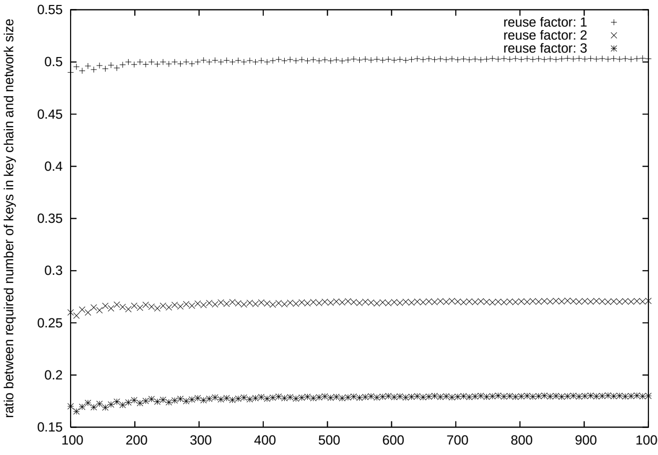


Fig. 3. Ratio between the number of keys to be stored in the key ring of a new node and the size of the network if a connection probability of at least 99 percent is to be guaranteed.

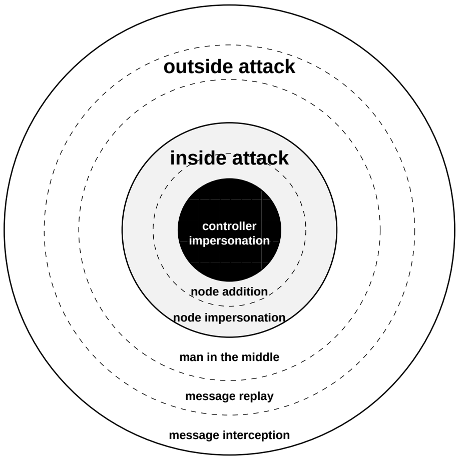


Fig. 4. A security model for WSNs.

We incorporate this observation in a graphical representation of our security model in Figure 4. 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.

The graphical representation is not to be misinterpreted as that it is strictly considered more difficult to conduct an outside attack in comparison to an inside attack.

Outside attacks can be further broken down into three levels of severity. At the first level, message interception is set. Message interception is seen as the interception of (eventually encrypted) messages on the physical medium. Attacks on this level might allow for pattern analysis and derivation of information structure. Message replay requires the attacker to simulate a sending node's radio characteristics. An attacker needs to intercept message traffic first in order to have it replayed. Some protocols employ a way to ensure message freshness to avoid message replay. Finally, man in the middle attacks require the attacker to not only intercept and send messages, but also to prevent nodes from receiving the original message.

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 infiltrate 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 be able to add new nodes to the network. As we have pointed in the introduction, 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.

B. Outside attacks against ARPD

Outside attacks are based on analysis of traffic pattern, message tampering or cryptoanalysis. Our scheme secures the network at link layer level. 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 studied in [13], [14], [15].

Some approaches can be made to lower the impact of *discovery message injection*. First, the number of acknowledgements sent by each node within a certain threshold time can be limited so that an attacker can't make the nodes use processing power extensively. Though this approach lowers energy consumption, it opens the possibility of a denial of service attack. An attacker can trick nodes into rejecting justified node addition requests. The approach is still adequate, since we consider an attacker to have full control over the broadcast medium and therefore to be able to prevent communication (including node join requests).

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.

C. Inside attacks against ARPD

Attacks to routing protocols. Inside attacks can be carried out on multiple layers of the application. Attacks to the routing of messages in a wireless sensor network have been of particular interest to research. Two of these attacks, namely *sinkhole attacks* and *wormhole attacks*, have been studied by Karlof and Wagner [16].

Attacks on the routing layer of the network protocol are difficult to prevent. Hu, Perrig and Johnson [17] present two schemes for ad hoc networks to defend against wormhole attacks. Neither of them is suitable for wireless sensor networks because they either require additional hardware in form of GPS receivers or tight time synchronisation.

If an attacker is able to capture a node preliminary to deployment, he can use the key material to connect two distant parts of the networks. Once this connection is established, overlaying routing protocols will decide to use the captured node as a preferred route.

ARPD makes this kind of attack more difficult for several reasons. Every node deletes keying material that was not used during pairwise key establishment phase. Therefore, an attacker cannot use an already deployed node to build sinkholes or wormholes. Keying material and node identity are indivisible bound to each other. So, it is not possible for the adversary to create new node entities for attacking routing protocols.

Node impersonation. ARPD implements a proof of identity by using keying material that is bound to the node's identity. This feature is a strong way to defend against various attacks. Nodes added to the network can authenticate already deployed nodes since only those nodes can generate the pairwise keys. Nodes deployed in earlier stages can authenticate the new nodes because of the bondage property. Node to node authentication in accordance with the definition above is therefore possible.

Once a misbehaving (because tampered with) node is revealed, other nodes have to isolate it and render it useless to the attacker. This can be achieved by keeping a list of misbehaving nodes in each nodes memory. Additions to the list must be authenticated, so to keep attackers from arbitrarly adding other nodes to this list.

VI. 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] 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.

[2] D. W. Carman, P. S. Kruus, and B. J. Matt, “Constrains and approaches for distributed sensor network security,” NAI Labs, Tech. Rep., September 2000. [Online]. Available: http://download.nai.com/products/media/nai/zip/nailabs-report-00-010-final.zip

[3] 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*. New York, NY, USA: ACM Press, 2003, pp. 62–72.

[4] R. Szewczyk, J. Polastre, A. M. Mainwaring, and D. E. Culler, “Lessons from a sensor network expedition.” in *EWSN*, ser. Lecture Notes in Computer Science, H. Karl, A. Willig, and A. Wolisz, Eds., vol. 2920. Springer, 2004, pp. 307–322.

[5] 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*. Washington, DC, USA: IEEE Computer Society, 2003, p. 197.

[6] O. Goldreich, S. Goldwasser, and S. Micali, “How to construct random functions,” *J. ACM*, vol. 33, no. 4, pp. 792–807, 1986.

[7] H. Krawczyk, M. Bellare, and R. Canetti, “Hmac: Keyed-hashing for message authentication,” , United States, 1997.

[8] L. Patrick and R. John, “A formally verified algorithm for interactive consistency under a hybrid fault model,” Tech. Rep., 1993.

[9] J. Douceur, “The sybil attack,” 2002. [Online]. Available: citeseer.ist.psu.edu/douceur02sybil.html

[10] S. Zhu, S. Setia, and S. Jajodia, “Leap+: Efficient security mechanisms for large-scale distributed sensor networks,” *ACM Trans. Sen. Netw.*, vol. 2, no. 4, pp. 500–528, 2006.

[11] R. Anderson, C. Haowen, and A. Perrig, “Key infection: Smart trust for smart dust,” 2001. [Online]. Available: citeseer.ist.psu.edu/anderson01key.html

[12] J. Hwang and Y. Kim, “Revisiting random key pre-distribution schemes for wireless sensor networks,” in *SASN ’04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*. New York, NY, USA: ACM Press, 2004, pp. 43–52.

[13] M. Pirretti, S. Zhu, N. Vijaykrishnan, P. McDaniel, M. Kandemir, and R. Brooks, “The sleep deprivation attack in sensor networks: Analysis and methods of defense,” in *Conference on Innovations and Commercial Applications of Distributed Sensor Networks*, October 2005, p. to appear, best Paper Award. [Online]. Available: http://www.gigascale.org/pubs/741.html

[14] F. Stajano and R. Anderson, “The resurrecting duckling: Security issues for ad-hoc wireless networks,” 1999, pp. 172–194. [Online]. Available: citeseer.ist.psu.edu/stajano99resurrecting.html

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

[16] 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*, May 2003, pp. 113–127. [Online]. Available: citeseer.ist.psu.edu/article/karlof02secure.html

[17] Y. Hu, A. Perrig, and D. Johnson, “Packet leashes: A defense against wormhole attacks in wireless ad hoc networks,” Department of Computer Science, Rice University, Tech. Rep., 2001. [Online]. Available: citeseer.ist.psu.edu/hu01packet.html