

# A Tree-Based Approach for Secure Key Distribution in Wireless Sensor Networks

Erik-Oliver Blaß  
Institute of Telematics  
University of Karlsruhe  
Germany  
blass@tm.uka.de

Michael Conrad  
Institute of Telematics  
University of Karlsruhe  
Germany  
conrad@tm.uka.de

Martina Zitterbart  
Institute of Telematics  
University of Karlsruhe  
Germany  
zit@tm.uka.de

## ABSTRACT

Providing security for wireless sensor networks is a challenging task. Besides physical restrictions like limited computing abilities and available memory storage, a major problem results from the lack of any infrastructure components in a sensor network. Central components for security like eg. key-distribution centers or Central Authorities (CA) are impossible to realize. It is therefore difficult to distribute encryption keys necessary for secure communication among sensors. This paper presents a novel approach to securely distribute keys to sensors joining the network. The approach is self organizing and minimizes memory consumptions as well as radio transmissions.

## 1. INTRODUCTION

As wireless sensor networks are becoming more and more popular, questions about their security arise. There is a large number of scenarios where the data exchanged between sensor nodes is critical eg. in health applications. Microcontroller-based tiny hardware sensors are not capable of performing complex security protocols known from the PC/Internet world. However the major problem for sensor networks is the absence of central infrastructure components. In this context, key distribution becomes a delicate problem. A malicious node can learn or even forge plaintext keys sent from one node to the other and is thus able to decrypt encrypted messages or forge new messages. Due to the lack of infrastructures key-distribution centers or CAs are not feasible in sensor networks. This paper presents the basic idea for a new key distribution mechanism in sensor networks. It uses the fact that communication in sensor networks follows a certain tree-like scheme, *aggregation*, allowing memory efficient and energy saving but still secure key distribution. New sensor nodes joining the network are able to autonomously share the keys they need to accomplish their mission.

## 2. RELATED WORK

Most work dealing with key distribution in wireless sensor networks assumes that there is the need for every sensor node to be able to securely communicate with arbitrary other nodes from the network. This is a very strong assumption that might not be realistic in a real world sensor scenario (cf. section 3). A typical representative for this class of distribution schemes is presented in [1]. Every sensor node receives a huge subset of an even larger set of *pool-keys* from the user. With a certain probability it is now

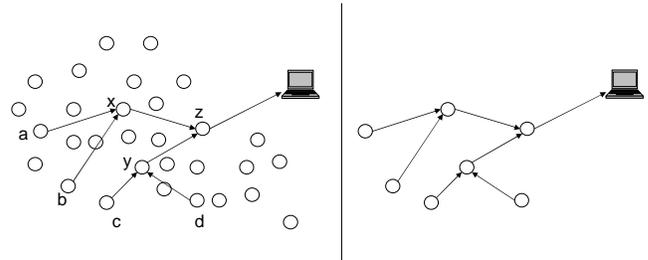


Figure 1: a) Example data aggregation in a sensor network b) Aggregation as a tree-like hierarchy

possible that two nodes willing to communicate have at least one common key in their subset. In such a way however a lot of memory is wasted, which is especially critical for low memory sensor devices with eg. only 4KByte RAM. Other work focuses on the use of a base station to distribute keys, eg. [2]. Depending on a base station for every key exchange is an unrealistic assumption, as in a real world sensor network this base station might not be available at all times, especially not for each and every key exchange.

## 3. AGGREGATION

Typically sensors report measured data, eg. temperature, towards a *data sink*. On the way to the sink data can be aggregated by so called *aggregation nodes*. These nodes are able to collect data from other sensors nodes and process them, for example computing a mean value and forward the *aggregate* to the sink. Figure 1 a) shows an example network. Sensor nodes *a* and *b* measure the temperature in room 1 at different positions, eg. at the top and the floor, and nodes *c* and *d* measure the temperature in room 2 respectively. Represented as a laptop, the sink is however only interested in the mean temperature of the complete building. Therefore a tree-like scheme has to be established for sensor communication. Aggregation node *x* collects temperature measurements from nodes *a* and *b* and computes their mean value forwarding this to aggregation node *z*. Aggregation node *y* does the same for node *c* and *d*. Finally node *z* computes the mean temperature for the whole building, i.e. two rooms, and reports it to the sink. As shown in figure 1 b) this communication scheme forms a hierarchy, sensor nodes (vertexes) and communications paths (edges) form a graph, more precisely a tree. The question whether communication within this tree is overlay communication or

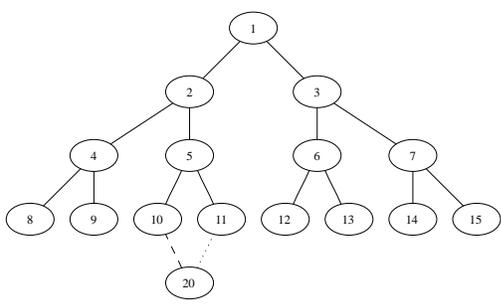


Figure 2: A sample aggregation tree

not, i.e. if there are multiple *hops* necessary to reach  $x$  from  $a$ , is negligible at this point.

Contrary to assumptions made in related work, the major observation here is that in this typical scenario, sensor nodes do not have to communicate arbitrarily with each other node. Nodes talk to each other only by a communication scheme, i.e. an aggregation tree. Sensor node  $a$  for example has to exchange data only with node  $x$ , therefore  $a$  needs a shared key with  $x$ , aggregation node  $y$  needs a key with node  $z$ . On the other hand communication is unlikely to happen between nodes from other categories like eg. communications between a temperature sensor and a light sensor. Also communication between nodes within the same category will never happen. Nodes  $a$  and  $b$  as well as  $c$  or  $d$  will most likely never exchange data among each other. Again: They might *transport* or *forward* data in multi-hop situations but there is no need for an applied *end-to-end* communication. However to ensure authenticity it will also be necessary for eg. the sink to verify that certain received aggregated data has been aggregated in a correct way, whatever that means – think of data origin verification. As an example, to verify aggregated data from  $z$  the sink will have to talk to  $x$  and  $y$ . To check whether  $x$  aggregated correctly, the sink has to talk to  $a$  and  $b$ . It is not in the scope of this work to introduce an efficient algorithm for authentic data aggregation, but important to point out that therefore secure communication has to be established not only between the vertexes in the aggregation respectively communication graph: Keys are also necessary between a vertex and its grandfather, great-grandfather and so on. So eg. sensor  $a$  will need a shared key with  $x$  but also different shared keys with  $z$  and the sink to enable possible additional authenticity verifications.

#### 4. TREE-BASED KEY DISTRIBUTION

Let us assume for now that aggregation in sensor networks forms a complete binary tree without loss of generality. Before a new sensor joins the network, it must be paired by the user or a *MasterDevice*. The pairing is essential for the node to obtain its new position inside the aggregation tree, to identify the parent i.e. the first aggregation node. The user knows node positions because of their duty or use within the network. The distribution scheme is defined inductively. Shown in figure 2, a complete binary tree exists before a new node starts to join. This node is now paired by the user. The user identifies, that this node will have to communicate with aggregation node 10 because of its use. For this the user assigns the new node a new id  $ID_{node}$ . As the new node will be a child of 10,  $ID_{node}$  could be 20 or 21 to comply with the binary tree scheme. As the new node is the first child of

10, it becomes 20. Now 10 is the *primary parent*  $P_1$  of 20. Furthermore the user computes a *secondary parent*  $P_2$  for 20

$$\text{using: } P_2 = \begin{cases} ID_{node}/2 + 1 & \text{if } ID_{node} \text{ is even} \\ (ID_{node} + 1)/2 & \text{if } ID_{node} \text{ is odd} \end{cases}$$

$P_2$  for 20 is therefore node 11. The user can now handout two *tickets* (including keys) to 20 that allow secure communication for 20 with 10 and 11. This can be done efficiently as each node in the network might share a pairwise different key with the user – thus allowing the user to securely access distinct nodes. As 20 can now establish secure channels to parents 10 and 11, it will ask them, which other aggregation nodes are on the way to the sink 1. Even in the presence of one cheater 20 will come to know 5, 2 and 1. The next step is to build secure channels to these nodes by securely exchanging shared secrets with them, first of all with 5. The main idea is, that 20 generates a new key  $K$  and splits it into two parts  $K_1$  and  $K_2$ , these parts are encrypted with the key for 10 or 11 respectively and sent to 10 and 11 to forward them towards 5. As  $K_1$  is encrypted from 20 with the key shared with 10 only 10 can decrypt it. Then 10 encrypts  $K_1$  with the secret key 10 shares with 5 and sends the result to 5. On the other hand 11 does the same with  $K_2$ . Finally 5 can decrypt both transmissions from 10 and 11 and restore  $K$ . As neither 10 nor 11 comes to know the other part of  $K$ ,  $K$  is finally secretly transmitted from 20 to 5 even in the presence of one malicious node. Also changing  $K_1$  or  $K_2$  maliciously would not help any node as this would only deny communication between 20 and 5, but impersonation attacks are not possible. To secure communication between 20 and 2 or 1, the same procedure can be repeated. 20 sends one half of the encryption key  $K$  to 10, the other one to 11. As the aggregation tree was build inductively both have already a secure channel to node 2 and can transfer their part of  $K$  to node 2 directly (10s parents were 5 and 6).

An analysis of this mechanisms complexity shows, that each setup of a secure communication channel only needs 4 symmetric encryptions and 4 communication steps inside aggregation *overlay*, completely independent from the position of the new node inside the tree or the depth of the tree. Also each node has to store only keys from other nodes, which are absolutely necessary because of its mission.

#### 5. CONCLUSION

In contrast to related publications in wireless sensor networks, this work concludes that there is no need to distribute keys between *arbitrary* sensor nodes. Instead this paper states that such need is an unnecessary strong and superfluous assumption, as real-world sensor networks communication is often a tree-like aggregation towards the sink. This paper then presents first steps to an efficient algorithm for a secure key distribution within this aggregation tree. The algorithm uses only a constant number of symmetric encryptions and stores only keys a sensor node needs anyhow because of its mission.

#### 6. REFERENCES

- [1] L. Eschenauer and V. Gligor. A key management scheme for distributed sensor networks. In *ACM CCS*, 2002.
- [2] A. Perrig, R. Szewczyk, V. Wen, D. E. Culler, and J. D. Tygar. SPINS: security protocols for sensor networks. In *Mobile Computing and Networking*, 2001.