On Active Attacks on Sensor Network Key Distribution Schemes

Stefan Dziembowski*, Alessandro Mei**, and Alessandro Panconesi***

University of Rome *La Sapienza*

Abstract. This paper concerns sensor network key distribution schemes (KDS) based on symmetric-key techniques. We analyze the problem of active attacks against such schemes. By active attacks we mean those attacks, where the adversary can maliciously disturb the communication between the sensors. We observe that the active adversary that captured even a small number of sensors, can anyway get a full control over the network, no matter how strong the KDS is. Therefore we conclude that the best scheme in this context is the one based on the method of Blöm (1984) (which guarantees perfect secrecy of the keys, as long as the number of corrupted sensors is small).

1 Introduction

Wireless sensor networks (WSNs) are a new promising technology that emerged a few years ago. A WSN consist of a large number of intelligent nodes which are low-cost, low-power and small. Their goal is to produce globally meaningful information from local data obtained by individual nodes. A typical node is equipped with a sensing unit (for temperature, light, sound, vibration, stress, weight, pressure, humidity, etc., depending on the application), a weak wireless link and processor with a small amount of memory. It is powered by a small battery that will usually not be replaced when it gets exhausted. WSNs have one or more points of control called *base stations*, which are nodes that are orders of magnitude more powerful and often serve as an interface to some other networks. In a typical application the sensors connect to each other to communicate the outcomes of their measurements. Later they may send some statistics about these values to the base station (this is called *in-network processing* [7]), or perform some actions depending on the messages that he received (this is called *in-network control* [14]). The principal characteristics of the WSNs, that make them different from the other types of networks, are: (1) *low computing power*: the total amount of processor cycles that a given sensor can use during its lifetime is severely constrained (because the computation costs energy); hence the sensors are not able to perform a large number of cryptographic public-key operations (like public-key encryption, or digital signatures);

* Supported by the European Research Council *Starting Grant*, Project Number: 207908. Part of this work was done when this author was receiving a Marie-Curie Fellowship 2006-024300-CRYPTOSENSORS.

** Partially funded by the FP7 EU project "SENSEI, Integrating the Physical with the Digital World of the Network of the Future", Grant Agreement Number 215923, www.ict-sensei.org.

*** Supported by *Progetto FIRB Italia-Israele*.

S. Dolev (Ed.): ALGOSENSORS 2009, LNCS 5804, pp. 52–63, 2009.

(2) *limited communication capability*: since transmitting information costs energy also the total amount of data that the sensor can send during its lifetime is small; (3) *limited transmission range*: since each sensor's wireless link has range that covers only a limited number of other nodes, the sensors can communicate with each other only if the distance between them is small, and therefore every sensor has only few neighbors in the network; (4) *small memory*: in order to keep the prices of the sensors low, the manufacturers usually equip it only with small memory; (5) *lack of tamper-resistance*: since tamper resistant hardware is expensive to construct, the sensor are usually easy to reverse-engineer, i.e., anybody who captures a sensor may get a full information about its internal state; (6) *mobility and unknown topology*: in many cases the sensors frequently change their physical positions, moreover, even if the network is static, its topology may be unknown at the moment of the deployment (the sensor may be even distributed in a random way: e.g. they may be dropped from an airplane).

In many cases sensor networks have to be designed with security in mind. In particular, one often has to guarantee that the messages exchanged between the sensors remain *secret* even if the adversary eavesdrops on the communication between the nodes. In practice, however, we usually have to deal with stronger attacks in which the adversary may also be able disturb the behavior of the network by introducing fake messages, or modifying the legitimate ones. In many applications guaranteeing integrity of the output produced by the network is actually more important than its secrecy. This is especially true when the data that the network collects is anyway available publicly (like e.g. the temperature, or air pressure), and the main goal of the adversary may be to alter the outcome of the measurement.

Some papers already considered the problem of active attacks. In particular, some of them [4,13] proposed schemes that included mechanisms for message authentication- which is a technique that allows a node in the network to ensure that a given message M, that supposedly originates from some other node P_i, really comes from P_i. However, up to our knowledge, none of them fully analyzed the problem of active attacks. This is quite unfortunate, since in general security against an active adversary is trickier and harder to analyze than the one against the passive (eavesdropping only) adversary. In this paper we discuss the problem of active attacks against the sensor-network key distribution schemes (focusing on the problem of message authentication), and we evaluate security of the existing protocols. Our conclusion is that, when the active attacks are considered, the most appropriate scheme is the scheme of Blöm [2].

Organization of the paper. In the next section we provide a short introduction to the key distribution in sensor networks. Then, in Sect. 3, we describe informally the security issues related to the active attacks on the key distribution schemes, and we argue that the scheme of Blöm is more suitable when such attacks are considered. The formal definitions are provided in Sect. 4, and the scheme (based on the one of Blöm) is constructed in Sect. 5.

2 Key Distribution in Sensor Networks

Any secure communication requires a secret cryptographic key. Of course, one cannot simply equip all the sensors (in a given network) with the same key S, since the sensors

are not tamper-proof, and therefore by capturing and reverse-engineering one sensor the adversary could learn S. Going to the other extreme, one could give to each sensor P_i a separate key $S_{i,j}$ to communicate with each P_j, but this would be impractical, since it would require each sensor to store large amounts of data, linear in the number of all sensors in the network. Therefore usually a more sophisticated *key distribution scheme (KDS)* is needed. Unfortunately, the standard key-distribution schemes used for other types of networks either require interaction between the nodes and a trusted center (see e.g. [10]), which is too expensive to be implemented on low-powered devices, or rely on energy-consuming public key-techniques, and therefore cannot be used on typical sensors.

The problem of designing key-distribution schemes for the sensor networks attracted a lot of attention over the last couple of years. We will later define the notion of a key-distribution scheme more formally, but for now let us assume that such a scheme consists of (1) a method for distributing the key material between the sensor (a key material of each sensor P_i is called its *key ring*, and denoted S_i), and (2) a protocol that allows pairs (P_i, P_j) of sensors to establish a common secret key $S_{i,j}$ (since we are interested only in symmetric cryptography we will always have $S_{i,j} = S_{j,i}$). In principle, such a protocol could involve a number of steps in which P_i and P_j exchange messages. In this paper, however, for the sake of simplicity, we will assume that (unless explicitly stated otherwise) all the schemes are *non-interactive*, i.e. P_i can compute $S_{i,j}$ just from his key-ring and the identity of P_j. This can be done without loss of generality for the following reasons: (1) most of the interactive schemes can be easily converted to the non-interactive ones (for example in [13,11] it is shown how to remove the need of interaction from the scheme of [6]), and (2) in practical applications, for the efficiency reasons, one would probably anyway consider only the non-interactive schemes. Schemes in which every pair of sensors is able to establish such a common key will be called *complete*. We will say that a scheme is t*-resilient* if each key $S_{i,j}$ remains secret even if an adversary captured less than t sensor and learned their key rings (of course, we have to assume that he did not capture P_i and P_j).

In general, the existing approaches for constructing the KDS's can be divided in two classes. The first class originates from a method proposed in a different context in [2] and relies on simple linear-algebraic tools. For any parameter t it allows to construct a complete t-resilient scheme in which the key ring of each sensor is of a size $\alpha \cdot t$, where α is the length of each $S_{i,j}$ (all $S_{i,j}$'s are of equal length). If the adversary captures t sensors then the security of the [2] scheme breaks down entirely, as from the key rings of any t sensors one can compute the key rings of all the remaining ones. In [1] it was shown that the scheme of [2] is optimal in the class of complete schemes, in the following sense: in every complete t-resilient key KDS the size of the key ring needs to be at least $\alpha \cdot t$.

The second class of such methods derives from the first paper that explicitly considered the problem of key distribution in sensor networks [6]. The key idea is to relax the requirement of *completeness* and t*-resiliency*, so that the lower bound of [1] does not hold. More precisely, [6], and the follow-up papers [3,15,13] construct β*-incomplete* γ*-partially-resilient* schemes, where β*-incompleteness* means that for a randomly chosen pair (P_i, P_j) of distinct sensors the probability that P_i and P_j can establish a common

key $S_{i,j}$ is equal to β (which may be less than 1), and γ-*partial-resiliency* means that for a randomly chosen pair (P_i, P_j) of distinct sensors the probability that P_i and P_j can establish a common key $S_{i,j}$ *and* this key remains secret for the adversary, is equal to γ, which, of course, is usually a function $\gamma(x)$ of the number x of captured sensors.

The scheme of [2] can be viewed as a special case of such a generalized notion, if we set $\beta := 1$, and $\gamma(x)$ equal to

$$\gamma^t_{\mathsf{Blom}}(x) := \begin{cases} 1 \text{ if } x < t \\ 0 \text{ otherwise.} \end{cases} \tag{1}$$

In [6] it is shown how to construct a scheme with $\beta \in (0, 1)$ and γ being some function, slowly decreasing to 0. Let us now compare the scheme of [6] with a t-resilient scheme of [2]. Clearly, as long as the number x of captured sensors is below the the threshold t, the scheme of [2] is better than the one of [6]. However, for $x \geq t$ the scheme of [6] beats [2], as it can still offer a certain degree of security, while [2] is broken completely. It was argued in [6] that the problem of the incompleteness of the scheme can be bypassed by the routing techniques: if two sensors are not able to establish a common key they can always ask some other sensors to route the messages for them.

Encryption. Of course, key distribution is just a first step in constructing cryptographic protocols in which the $S_{i,j}$'s are used. For example, $S_{i,j}$'s may be treated as keys for symmetric encryption—in this case we can define an *encryption-key distribution scheme* (denoted *Enc-KDS*) as a functionality that consists of the following procedures:

1. a procedure for distributing key material among the sensors (let S_i denote the key material of a sensor P_i), and
2. for a pair (P_i, P_j) of sensors a procedure $Enc_{i,j}$ that, given a key ring S_i of P_i and any message M, produces a ciphertext $C = Enc_{i,j}(S_i, M)$. This procedure is executed by P_i. The ciphertext C can later be decrypted by sensor P_j using a procedure $Dec_{i,j}$. We should always have $Dec_{i,j}(S_j, C) = M$.
   In case of the incomplete schemes such a procedure may exist only for certain subset of the set of all the pairs of sensors.

Typically, Enc-KDS is constructed on top of a KDS scheme in a following straightforward way. Let Θ be a KDS and let (E, D) be some standard symmetric encryption scheme, where $C = E_K(M)$ denotes a result of encrypting M with a key K, and $D_K(C)$ denotes the result of decrypting C with key K. Then, one can construct an Enc-KDS, where (1) the key material $S_1, \ldots, S_w$ is distributed in the same way as in Θ; (2) in order to compute $Enc_{i,j}(S_i, M)$ the sensor P_i first calculates $S_{i,j}$ (as in Θ) and then sets $Enc_{i,j}(S_i, M) := E_{S_{i,j}}(M)$; and (3) in order to compute $Dec_{i,j}(S_j, C)$ the sensor P_j calculates $S_{i,j}$ and then sets $Dec_{i,j}(S_j, C) = D_{S_{i,j}}(C)$.

Authentication. It was suggested in some papers [4,13] that in a similar way one can achieve message authentication, by combining a KDS with a *message authentication code* (MAC) scheme. Recall that MAC is a scheme that consists of (1) a procedure Auth that, given a secret key K and a message M produces a *tag* $T = \mathsf{Auth}_K(M)$, and (2) a procedure Ver that given a key K, a message M and a tag T verifies if T is a valid

tag for M under the key K (see [9], or [5] for a formal definition). Obviously the keys $S_{i,j}$ distributed in the KDS can be used as keys for a MAC. A resulting scheme will be called a *MAC-key distribution scheme*. Formal security definition of a MAC-KDS appears in Sect. 4.

3 Active vs. Passive Security of the Key Distribution Schemes

So far we did not say anything about the security of the Enc- and MAC-KDS schemes. To reason formally about it we use a notion of an *adversary* that attacks the scheme and we define what is his goal and what he is allowed to do in order to achieve it, i.e. what is the attack model. We will provide such a definition in Sect. 4. For now, let us just discuss it informally. First, for the purpose of this discussion, assume that the goal of the adversary is simply to guess as many keys $S_{i,j}$ as he can: clearly if the adversary knows $S_{i,j}$ then he can decrypt the messages sent between P_i and P_j (if $S_{i,j}$ is used for encryption), or fabricate messages (if $S_{i,j}$ is used for authentication), although of course there may also exist other (than just guessing the keys) ways of breaking the protocol. The definition of the attack model depends on the type of a scheme whose security we are considering. When defining security of the Enc-KDS schemes we can restrict ourselves to an adversary that is only *passive*, i.e. he is only allowed to:

1. selectively capture some sensors, and learn all their internal data, including their key material, and the entire history of the execution (we will also say that a sensor was *corrupted* by the adversary, and otherwise we will call him *honest*),
2. eavesdrop all the messages sent by the sensors.

In case of the MAC-KDS schemes we should consider an adversary that is *active*, i.e., besides of capturing some sensors, and passively listening to the messages sent between the remaining ones, he has also the ability to interfere with the transmission. In other words, additionally to what is described in Points 1 and 2 above, he can

3. prevent some messages from arriving, and
4. fabricate new messages, send them to the honest sensors.

Note, that such an adversary can simulate also other attacks that are not explicitly mentioned above. For example, our adversary can alter any message M sent between two sensors (P_0 and P_1, say) in the following way. Suppose that his goal is to modify M by transforming it into some $\xi(M)$. He can achieve it by (a) intercepting M (he can do it by Point 2 above), (b) preventing M from arriving to P_1 (see Point 3), and (c) fabricating $M' := \xi(M)$ and sending it to P_1 (Point 4).

As another example consider an attack in which the adversary "takes control over some sensor P_i". This can be simulated as follows. First, the adversary captures P_i (Point 1). Then, he starts blocking all the messages that P_i may send (Point 3). At the same time he starts simulating P_i, controlling his execution in an arbitrary way. Observe that (by Point 2) he can eavesdrop all the messages that are sent to P_i, and forward them to his simulated copy of P_i. He can also fabricate messages claiming that they come from P_i (Point 4).

Our approach is different from the one used in the practitioners' community, where it is common to design protocols that are secure against one particular class of active attacks. One example of such a class are the so-called *Sybil attacks*, where a single node may present multiple identities to the other sensors in the network (such additional identities are called the *Sybil nodes*). Other types of active attacks that were considered in the literature include the *sinkhole attacks*, *wormhole attacks*, *replay attacks*, etc. (see e.g. [8]) All of these attacks can be easily simulated by our adversary.

Our, more general approach has the following advantages. First, Points 2–4 correspond to a real-life attack scenario where an adversary is equipped with a laptop and an antenna (such an attack was already described in [8]). Using this tools he may easily eavesdrop and disturb the communication in the network by inserting new messages, or preventing the legitimate ones from arriving to the destination by jamming the communication. Hence, we follow a good tradition in cryptography of being prudent by granting the adversary in the formal model as much power as he can realistically obtain in real life.

Moreover, a formal security framework allows us to give a precise evaluation of the protocols that we construct. Many previous papers in this area used just informal descriptions of the threat model, which in some cases led to ad-hoc constructions without precise security properties. Consider e.g. the idea of [11] to use random key predistribution to protect against the Sybil attacks. In the protocol of [11] some subset of the sensors jointly verify the identity of another sensor P_i by testing if he knows certain keys from the key pool (call these sensors the *verifiers*). The problem with the solution presented in [11] is that it does not take into account that some of the verifiers may actually also be controlled by the adversary. Such "corrupted verifiers" could e.g. falsely confirm that P_i passed the test, or, which may be even worse, accuse the honest sensors of being the Sybil nodes. Whether this attack is legitimate or not is not clear from the model of [11].

3.1 The Power of Active Attacks

One may wonder what is the additional power of the active attacks. We now argue that active security significantly differs from the passive one. First, observe that if the active attacks are considered, then some of the ideas that were valid in passive scenario break down completely. Consider e.g. the suggestion of [6] to route the messages between sensors that cannot establish a key, via some other sensors. An active adversary can simply capture a small number of sensors, make virtual copies of them (in his laptop, say), and then, by impersonating the captured sensors, volunteer to route the messages between any two parties that are not able to establish a key. Clearly, we can assume that the attacker has a more powerful antenna then the sensors, so he is able to jam messages from any legitimate sensors that may also want to do routing. In this way the system becomes completely broken. Of course the fact that the simple routing does not work in this context does not mean that the entire approach based on [6] fails, as there may exist applications that do not require the network to be complete. It is however unclear how to create them without using public-key cryptography. The second difference is more fundamental. We describe it below.

How Many Corruptions Can We Tolerate? Let $\mathcal{P}$ be the set of all sensors in the network, and let w be the degree of every node in the network (for simplicity assume that the network graph is regular). Suppose we use the strongest KDS possible, i.e. every sensor P_i is simply equipped with a separate key $S_{i,j}$ to communicate with any other sensor P_j (call it: a *perfect* KDS). As we argued in Sect. 2 such a scheme is too memory-consuming to be practical, but assume for a moment that we use it, just for the sake of this argument. Now, assume that the adversary corrupted some random set $\mathcal{P}'$ of sensors (let $x := |\mathcal{P}'|$). Thus, the adversary knows the keys $\mathcal{S}' = \{S_{i,j} \text{ such that } P_i \in \mathcal{P}' \text{ and } P_j \in \mathcal{P}\}$. Of course if the adversary is passive then the only damage that he can do is that he can decrypt the messages encrypted with the keys in $\mathcal{S}'$, which he trivially knows anyway (since he corrupted the sensors that know this values).

The situation in case of the *active* security is very different. Suppose, e.g., that the sensors connect to their neighbors in order to learn the output of their measurements. Now, assume that the keys $S_{i,j}$ are used for the message authentication. Let P_{i_0} be some sensor, and let $\mathcal{W}$ be the set of its neighbors. The active adversary can simply create "virtual copies" of the sensors from $\mathcal{P}'$, pretend that they are the "new" neighbors of P_{i_0}, and start sending arbitrary messages from them to P_{i_0}, authenticating them with appropriate keys from $\mathcal{S}'$. Moreover, since he fully controls the communication he can jam the messages coming from the "legitimate" neighbors (i.e. those in $\mathcal{W}$). Therefore if $x \geq w$, then the adversary can completely substitute the set of neighbors of P_{i_0}, by the "virtual" sensors that he controls. Of course the adversary may simultaneously perform the same attack against many other honest sensors, not just against P_{i_0}, and hence he may take a total control over the network! Let us now look at some possible remedies for this problem and discuss why in most of the cases they fail.

Detection of the change of neighbors. One natural idea may be to instruct the sensors to take special actions when the set of their neighbors changes rapidly. This idea may work in some cases, however it breaks down completely in case of the mobile networks, where the neighbors change frequently. Also, the adversary may choose to slowly substitute the legitimate neighbors with the "virtual" ones. Since the sensors are anyway expected to die at some point, because of the battery exhaustion, this will not raise any alarm.

Node replication detection. One could also think about introducing a mechanism in which the honest sensors jointly detect if a given sensor does not claim to be a neighbor of too many of them at the same time. This is called *node replication detection*, and there exist techniques for achieving it [11,12]. Unfortunately the techniques of [11] (see Sect. 2.2) assume that the sensors can connect to the trusted center (which would trivialize the problem of key distribution), and the techniques of [12] rely on the public-key cryptography, and it is questionable if they can be built just using the symmetric-key cryptography. The main difficulty is that the adversary (that controls some corrupted sensors) may insert false "accusations" against some honest sensors, and it is unclear how to resolve such disputes. Note also that the centralized solutions (e.g. reporting the set of neighbors to the base station) do not work here, since we are interested in schemes that do not rely on the interaction with the base station.

Location-dependent KDS. Another idea that one could consider is to make the key-rings dependent on the geographic location of the sensor. In this way, a corrupted sensor could

claim to be a neighbor of only those sensors that are physically close to him. This of course would require the network to be completely static and the positions of the sensors would need to be known before the network is deployed. In such a case, however, much simpler solutions exist: instead of inventing a sophisticated key distribution scheme just give to each sensor a separate key to communicate to each of its neighbors (which are known beforehand).

In Sect. 3 we argued that the assumption that the adversary fully controls the communication corresponds to a practical attack when the attacker is equipped with a laptop and an antenna. Hence, one could consider the following countermeasure against the attack described above: let each sensor listen to all the messages he can hear, and verify that the sensor that claims to be his neighbor does not claim to be a neighbor of too many other sensors in the network. Of course, the need of receiving and analyzing the messages would increase the energy consumption. Moreover, this countermeasure works only if the adversary has just one static antenna that is used for broadcasting messages to all the sensors in the network. This method fails if the adversary can change the physical location of his antenna and decrease the strength of the signal, so that only the sensor that are close can hear it. He may also use several weaker antennas, or he can distribute his own sensors that would serve as "relay stations". This should be quite economical, as it is believed that the main advantage of the sensor networks will actually be the low prices of the sensors.

So far, we assumed that the scheme that we use is perfect (i.e. every sensor P_i knows an independent key $S_{i,j}$ to communicate with each P_j). In this paper we are interested in schemes that are not perfect, and hence may be partially resilient and incomplete. Clearly, partial resiliency may only help the adversary, and therefore the attack described above remains valid (actually, as we show in the next section partial resiliency causes even some additional problems). Let us now assume that the scheme is β-incomplete. This means that, in particular, each sensor P_{i_0} can establish a connection with (on average) $\beta \cdot w$ of its neighbors. On the other hand, also (on average) $\beta \cdot x$ sensors, out of x sensors that the adversary corrupted, can establish a connection with P_{i_0}. Thus if $\beta \cdot x \geq \beta \cdot w$ (which is trivially equivalent to $x \geq w$), then again the adversary can substitute the honest neighbors of P_{i_0} with the sensors controlled by him. Hence we get the following.

Moral 1. *In case of active security it makes sense to consider only adversaries that corrupt less than w sensors.*

A Problem with Partial Resiliency. In this section we argue that in case of the partially resilient incomplete schemes the situation gets even worse than what was described in Sect. 3.1. Suppose we have some γ-partially-resilient β-incomplete KDS, and assume that the adversary corrupted some number x of sensors. The difference $\beta - \gamma(x)$ corresponds to a probability that a link between two sensors exists and it is broken, and, of course, the smaller it is, the better for us. Let now ask the following question: how large difference between β and $\gamma(x)$ can we tolerate? We now argue that the answer is very different for the passive and for the active security.

Clearly in case of a passive adversary even large values of $\beta - \gamma(x)$ can still be tolerated. For example if $\beta = 60\%$ and $\gamma(x) = 40\%$ then the adversary can eavesdrop

on average only on $(\beta-\gamma(x))/\beta = 1/3$ links, which for some applications may be still be acceptable.

Now, let us examine the case of active security. Again, consider some sensor P_{i_0}, let $\mathcal{W}$ be the set of its neighbors, and let $w = |\mathcal{W}|$. Clearly P_{i_0} can directly communicate with $\beta \cdot w$ sensors (on average). However, the set of sensors that could potentially communicate with P_{i_0}, if they were its neighbors, is much larger, and on average equal to $\beta \cdot n$. Out of this set, there are on average $(\beta - \gamma(x)) \cdot n$ sensors P_j that can communicate with P_{i_0}, *but* the adversary knows the key S_{j,i_0}. Hence the adversary can do the following.

1. Determine[1] the set $\mathcal{P}''$ of sensors such that for every $P_j \in \mathcal{P}''$ the adversary knows S_{j,i_0}. As we argued before the expected size of $\mathcal{P}''$ is $(\beta - \gamma(x)) \cdot n$.
2. For each $P_j \in \mathcal{P}''$ create a "virtual copy" of it, and claim to P_{i_0} that it is his new neighbor. Since the adversary knows S_{j,i_0} he can easily impersonate P_j, and send (in his name) to P_{i_0} any message he wants. He can also jam the communication of P_{i_0} with the legitimate neighbors.

Hence, if $(\beta - \gamma(x)) \cdot n \geq \beta \cdot w$ then the adversary can replace all the honest neighbors of P_{i_0} with the sensors controlled by him. Of course, as in Sect. 3.1, he can perform the same attack not just against not just against P_{i_0}, but against all the sensors in the network. Thus we conclude with the following.

Moral 2. *In case of active security it makes sense to consider only adversaries that corrupt such a number x of sensors that $\gamma(x) \cdot n$ is smaller than $(\beta - \gamma(x)) \cdot w$.*

For example suppose that $n = 10.000, w = 100$ and $\beta = 50\%$. Even if $\gamma(x)$ is very close to β, say: $\gamma(x) = 49\%$, then the security is broken, since $\beta \cdot w = 50$ and $(\beta - \gamma(x)) \cdot n = 100$. In the full version of this paper [5] we provide an analysis of some of the schemes existing in the literature, showing that, for the realistic values of the parameters the value of $(\beta - \gamma(x)) \cdot n$ is much larger than w.

The bottom-line is that as long as the active security is considered, the right choice of a KDS is simply the original scheme of [2] (see Sect. 2). By setting $t = w$ (where w is the typical number of neighbors of a sensor) we get a scheme that is resilient against an adversary that corrupted less than w sensors (and where the key-ring of each sensor is $w \cdot \alpha$). This should suffice, as, by Moral 1, if the adversary corrupts w sensors then anyway he can take a full control over the network. Hence, no matter how strong our KDS is, the security is in this case completely broken anyway. Of course storing $w \cdot \alpha$ bits may be expensive for larger values of w. However, in most of the applications the size of the sensor's memory needs to be anyway linear in w, since the sensor most likely needs to store some information about each of the neighbors with which it connects.

In the next section we formally define the authentication-key distribution schemes. Then, in Sect. 5 we construct a scheme, based on the method of [2], that is secure in this framework. As we remarked above, we believe that this is the right choice of a MAC-KDS. In the full version of this paper [5] we also analyze possibility of constructing a secure authentication scheme basing on the approach of [6].

[1] Clearly, the adversary can compute $\mathcal{P}''$ in polynomial time, unless some special intractability assumptions are introduced.

4 Definition

In this section we give a formal definition of a MAC-KDS. A similar definition was already proposed in [4], however we belive that our approach is slightly simpler. We start with a functional definition. Consider a group of *sensors* $\mathcal{P} = \{P_1, \ldots, P_n\}$. A *MAC-key distribution scheme (MAC-KDS) for* $\mathcal{P}$ is a tuple of algorithms $\Phi = (\mathsf{Distr}, \mathsf{Auth}, \mathsf{Ver})$, where:

- Distr is a randomized algorithm that outputs a sequence $(S_1, \ldots, S_n)$, where each $S_i \in \{0,1\}^*$ is interpreted as a key ring of P_i.
- Auth is a deterministic algorithm that takes as input a tuple (P_i, P_j, S_i, M) (where $P_i, P_j \in \mathcal{P}$ and $M \in \{0,1\}^*$) and outputs a *tag* $T \in \{0,1\}^*$, or a special symbol $\perp$. We will always have that either $\mathsf{Auth}(P_i, P_j, S_i, M) \neq \perp$ for every M, in which case we say that *a link between P_i and P_j exists*, or that $\mathsf{Auth}(P_i, P_j, S_i, M) = \perp$ for every M, in which case we say that a *a link between P_i and P_j does not exist*. Moreover, we require that a link between P_i and P_j exists if and only in a link between P_j and P_i exists.
- Ver is a deterministic algorithm that takes as input a pair (P_i, P_j, S_j, M, T) and outputs OK or $\overline{\mathsf{OK}}$. We require that if a link between P_i and P_j exists, then it always holds that $\mathsf{Ver}(P_i, P_j, S_j, M, T) = \mathsf{OK}$, where $T = \mathsf{Auth}(P_i, P_j, S_i, M)$.

Additionally, the above algorithms take as input a *security parameter* α. We say that Φ is β-incomplete, if the probability that a link between any two distinct sensors exists is equal to β.

The security of KDS was already informally discussed in Sect. 3. In our definitions we will be as pessimistic as we can—i.e. in order to make our definitions stronger, we will give to the adversary as much power as possible. We will be also very generous in defining what counts as the success of the adversary. Our definition will follow a common style in cryptography, namely we will define a game between a polynomial-time (in α) Turing machine $\mathcal{A}$, called an *adversary* and an oracle Υ_Φ. The oracle will internally simulate the execution of the key-distribution scheme Φ. First, the oracle runs Distr to obtain the key-rings $(S_1, \ldots, S_w)$. Then, the adversary attacks the scheme by issuing requests to the oracle. Below we describe the requests that the adversary can issue.

Corruption requests (“$corrupt$”). The adversary can issue less than t requests denoted $corrupt(P_i)$ (where P_i is any sensor). The meaning of this request is that the adversary wants learn the internal state of P_i. The oracle Υ_Φ replies with S_i. This requests can be chosen adaptively, i.e. they can depend on what $\mathcal{A}$ has seen so far.

Authentication request (“$auth$”). As described in Sect. 3 we grant to the adversary power to fully control the communication. To be realistic we should also assume that he can influence the contents of the messages that are sent, e.g. by causing a certain event on which the sensors are supposed to react. To capture this we assume that the adversary may simply perform a *chosen-message attack*, i.e. he may choose the messages that are sent by the honest sensors. Since we can also assume that the adversary can prevent some messages from arriving to destination, when analyzing security we can just restrict ourselves to the messages that the adversary has

chosen. We will model this attack by giving the adversary a right to issue a request $auth(P_i, P_j, M)$, to which the oracle replies with $\mathsf{Auth}(P_i, P_j, S_i, M)$.

Testing requests ("*try*"). The adversary is allowed to fabricate any message that he wants, and send it some sensor P_j. By observing the reaction of P_j he may obtain some information about its private data. We model this by allowing $\mathcal{A}$ to issue a request of $try(P_i, P_j, M, T)$ whose meaning is "the adversary wants to know if P_j accepts T as a tag on a message M coming from P_i". The oracle replies with $\mathsf{Ver}(P_i, P_j, S_j, M, T)$. In practical scenarios the adversary may get this information e.g. by sending (M, T) to P_j (claiming it comes from P_i), observing the actions of P_j, and checking if P_j replies with some error message.

Consider some execution of $\mathcal{A}$ against an oracle Υ_Φ. We will say that a link between P_i and P_j is *broken* if the adversary managed to fabricate *at least one* message that P_j accepts as originating from P_i, i.e. if he at least once issues a request $try(P_i, P_j, M, T)$ such that the oracle replied OK. To exclude trivial ways of achieving this goal, we require that the adversary never issued any of the following requests: $auth(P_i, P_j, M)$, $auth(P_j, P_i, M)$, $corrupt(P_i)$, and $corrupt(P_j)$. We will say that the scheme is γ-*partially-secure* if for every x, and for every polynomial-time adversary $\mathcal{A}$ that corrupts less than x sensors we have that

$$P\left(\text{the number of links that exist and are not broken is at most } \gamma(x) \cdot \tfrac{n(n-1)}{2}\right) \quad (2)$$

is negligible[2] in α. We have chosen to multiply $\gamma(x)$ by $\frac{n(n-1)}{2}$ to remain consistent with the notation used in Sect. 2.

5 The Construction Based on the Scheme of [2]

In this section we construct a MAC-KDS scheme Φ^t_{Blom} that is γ^t_{Blom}-secure, where γ^t_{Blom} was defined in Eq. (1). Our construction is based on the scheme of Blöm [2]. The main building blocks are: a *message authentication code* scheme $\mathsf{MAC} = (\mathsf{MAC.Auth}, \mathsf{Mac.Verify})$ and a hash function H, which will be modeled as a random oracle (this concepts are defined in [9], see also [5]). Let F be a Galois Field $\mathrm{GF}(2^\alpha)$, and let V be an $w \times m$ Vandermonde matrix, over F, i.e.:

$$V = \begin{bmatrix} V_1 \\ \vdots \\ V_w \end{bmatrix} := \begin{bmatrix} 1^0 & 1^1 & \cdots & 1^{m-1} \\ & & \vdots & \\ w^0 & w^1 & \cdots & w^{m-1} \end{bmatrix}$$

The scheme $\Phi_{Blom} = (\mathsf{Blom.Gen}, \mathsf{Blom.Auth}, \mathsf{Blom.Ver})$ is defined as follows.

- The $\mathsf{Blom.Gen}$ algorithm first chooses a random symmetric $m \times m$-matrix D with entries in F. Then, it sets $U = (U_1^T, \ldots, U_w^T) := (V \cdot D)^T$. The share of each P_i is U_i.

[2] A function μ is negligible if for every $c > 0$ there exists x_0 such that for every $x > x_0$ we have $|\mu(x)| < x^{-c}$.

- The $\mathsf{Blom.Auth}(P_i, P_j, S_i, M)$ algorithm first computes $S_{i,j} := U_i \cdot V_i$. Then it outputs $\mathsf{MAC.Auth}(H(S_{i,j}), M)$.
- The $\mathsf{Blom.Ver}(P_i, P_j, S_j, M, T)$ algorithm computes $S_{j,i} = U_j \cdot V_j$ and outputs $\mathsf{Mac.Verify}(H(S_{j,i}), M, T)$.

Denote $S := U \cdot V$. Clearly S is symmetric. Hence $S_{i,j} = S_{j,i}$ and the output of Blom.Ver is always OK on the legitimate tags. The original scheme of [2] did not require hashing the key. This is needed in our case, since we use the values $S_{i,j}$ as keys for the MAC, and without hashing our protocol would be vulnerable to the *related key attacks*, sice the keys $S_{i,j}$ would not be independent.

Lemma 1. *The scheme Φ_{Blom} constructed above is γ_{Blom}-secure.*

The proof appears in the full version of this paper [5].

References

1. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: FOCS 1997, p. 394 (1997)
2. Blom, R.: An optimal class of symmetric key generation systems. In: Beth, T., Cot, N., Ingemarsson, I. (eds.) EUROCRYPT 1984. LNCS, vol. 209, pp. 335–338. Springer, Heidelberg (1985)
3. Chan, H., Perrig, A., Song, D.: Random key predistribution schemes for sensor networks. In: IEEE Symposium on Security and Privacy (May 2003)
4. Du, W., Deng, J., Han, Y.S., Varshney, P.K., Katz, J., Khalili, A.: A pairwise key predistribution scheme for wireless sensor networks. ACM Transactions on Information and System Security 8(2), 228–258 (2005)
5. Dziembowski, S., Mei, A., Panconesi, A.: On active attacks on sensor network key distribution schemes. full version of this paper (to appear)
6. Eschenauer, L., Gligor, V.D.: A key-management scheme for distributed sensor networks. In: ACM CCS 2002, pp. 41–47 (2002)
7. Heidemann, J., Silva, F., Intanagonwiwat, C., Govindan, R., Estrin, D., Ganesan, D.: Building efficient wireless sensor networks with low-level naming. In: SOSP 2001, pp. 146–159. ACM, New York (2001)
8. Karlof, C., Wagner, D.: Secure routing in wireless sensor networks: attacks and countermeasures. Ad Hoc Networks 1(2-3), 293–315 (2003)
9. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. Chapman & Hall/Crc Cryptography and Network Security Series. Chapman & Hall/ CRC (2007)
10. Neuman, B.C., Ts'o, T.: Kerberos: An authentication service for computer networks. IEEE Communications 32(9), 33–38 (1994)
11. Newsome, J., Shi, E., Song, D., Perrig, A.: The sybil attack in sensor networks: analysis & defenses. In: IPSN 2004, pp. 259–268. ACM, New York (2004)
12. Parno, B., Perrig, A., Gligor, V.: Distributed detection of node replication attacks in sensor networks. In: IEEE Symposium on Security and Privacy, pp. 49–63 (2005)
13. Pietro, R.D., Mancini, L.V., Mei, A.: Energy efficient node-to-node authentication and communication confidentiality in wireless sensor networks. Wirel. Netw. 12(6), 709–721 (2006)
14. Silberstein, A., Yang, J.: Many-to-many aggregation for sensor networks. In: IEEE 23rd International Conference on Data Engineering, 2007. ICDE 2007, pp. 986–995 (2007)
15. Zhu, S., Xu, S., Setia, S., Jajodia, S.: Establishing pairwise keys for secure communication in ad hoc networks: A probabilistic approach. In: ICNP 2003, p. 326 (2003)