AUTHENTICATED KEY EXCHANGE WITH GROUP SUPPORT FOR WIRELESS SENSOR NETWORKS

PETR ŠVENDA
MASARYK UNIVERSITY, BRNO, CZECH REPUBLIC
SVENDA@FI.MUNI.CZ
AND VÁCLAV MATYÁŠ
MASARYK UNIVERSITY, BRNO, CZECH REPUBLIC
MATYAS@FI.MUNI.CZ

Abstract. Our work targets the area of authenticated key exchange for wireless sensor networks. Probabilistic key pre-distribution schemes were developed to deal with limited memory of a single node and high number of potential neighbours. We present a new idea of group support for authenticated key exchange that substantially increases the resilience of an underlaying probabilistic key pre-distribution scheme against the threat of node capturing.

Key words. security, wireless sensor network, key distribution, node capture resilience

AMS subject classifications. 68M10, 68M12, 68M14, 94A60, 94A62

Technical advances in micro-electro-mechanical systems technology, digital electronics and wireless communications have enabled the development of low-cost, low-power, multifunctional devices that are small in size and communicate only at short distances. These devices can be used to form a new class of applications, Wireless Sensor Networks (WSNs). WSNs consist of a mesh of a several powerful devices (denoted as *base stations, sink or cluster controller*) and a high number ($10^3 - 10^6$) of a low-cost devices (denoted as *nodes or motes*), which are constrained in processing power, memory and energy. The nodes are equipped with an environment sensor (e.g., heat, pressure, light, movement). Events recorded by the sensor nodes are locally collected and then forwarded to a base station (BS) using multi-hop paths for further processing.

Wireless networks are widely used today and they will spread even more with increasing number of personal digital devices that people are going to use in near future. Sensor networks form just a small fraction of future applications, but they abstract some of the new concepts in distributed computing.

WSNs are considered for and deployed in a multitude of different scenarios such as emergency response information, energy management, medical monitoring, wildlife monitoring or battlefield management. Resource-constrained nodes render new challenges for suitable routing, key distribution, and communication protocols. Still, the notion of sensor networks is used in several different contexts. There are projects targeting development of very small and cheap sensors (e.g. [2]) as well as research in middleware architectures [24] and routing protocols (AODV [7], DSR [8], TORA, etc.) for self-organising networks – to name a few.

No common hardware architecture for sensor nodes WSN is postulated and will depend of the target usage scenario. Currently available hardware platforms for sensor nodes ranges from Mica Motes [1] equipped with 8-bits Atmel ATmega 128L down to Smart Dust motes [2] with their total size around $1mm^3$ and extremely limited computational power. No tamper resistance of node hardware is assumed so far.

Security often is an important factor of WSN deployment, yet applicability of some security approaches is often limited. *Terminal sensor nodes* can have no or little physical protection and should therefore be assumed as *untrusted.* Also, *network topology knowledge is limited* or not known in advance. Due to the limited battery power, communication traffic should be kept as low as possible and most operations should be done locally, not involving the trusted BS.

The main contribution of our work is a protocol design for authenticated key exchange with an improved resilience against the threat of node capturing over existing probabilistic pre-distribution schemes. A short paper outlining the protocol with only some relevant issues has been published as [22] and a detailed treatment of this work is available in our technical report [23]

0.1. Node-compromise attacker model. Common attacker model in the network security area is the extension of the classic Needham-Schroeder[1] model [18] called the node-compromise model [11, 5, 9, 10], described by the following additional assumptions:

A1: The key pre-distribution site is trusted – Before deployment, nodes can be pre-loaded with secrets in a secure environment.

A2: The attacker is able to capture fraction of deployed nodes – No physical control over deployed nodes is assumed. An attacker is able to physically gather nodes either randomly or selectively based on additional information about the nodes role and/or carried secrets.

A3: The attacker is able to extract all keys from a captured node – No tamper resistance of nodes is assumed. This lowers the production cost and enables production of a high number of nodes, but call for novel approaches in security protocols.

0.2. Secure link communication. Secure link communication is the building block for most of the security functions maintained by the network. Aggregation of the data from separated sensors needs to be authenticated, otherwise the attacker can inject his own bogus information. Routing algorithms need to utilize authentication of packets and neighbours to detect and drop malicious messages and thus prevent network energy depletion and route messages only over trustworthy nodes. Data encryption is vital for preventing the attacker from obtaining knowledge about actual value of sensed data and can also help to protect privacy of the sensed environment. On top of these common goals, secure and authenticated communication can be used to build more complex protocols designed to maintain reliable sensing information even in a partially compromised network. The generally restricted environment of WSNs is a challenge for designing of such protocols.

In a static WSN, nodes are assumed to have a fixed position and a relatively static set of neighbours. New nodes are only introduced during the redeployment, to replenish the nodes with exhausted batteries. Authentication and key exchange are performed with direct neighbours only, and thus with a very small subset (typically 5-40 nodes) of the total amount of nodes. However, neighbours of a particular node typically are not known before the deployment. Pre-distribution of pairwise authentication keys is thus not possible in this scenario due to the potentially high number of neighbours and limited memory of a single node. The following text will provide a summary of related work in the area of key (pre-)distribution.

0.2.1. Random key pre-distribution. Most common key pre-distribution schemes expect that any two nodes can always establish a link key if they appear as physical neighbours within their mutual transmission range. This property can be weakened in such a way that two physical neighbours can establish the link key only with a

[1]An intruder can interpose a computer on all communication paths, and thus can alter or copy parts of messages, replay messages, or emit false material.

certain probability, which is still sufficient to keep the whole network connected by secured links. A trade-off between the graph connectivity of link keys and the memory required to store keys in a node is introduced. If the network detects that it is disconnected, a range extension through higher radio power can be performed to increase the number of physical neighbours (for the price of higher energy spending).

The idea of random key pre-distribution for WSNs is introduced for the first time by Eschenauer and Gligor [11] (referred to as EG scheme) and is based on a simple but elegant idea. At first, a large key pool of random keys is generated. For every node, randomly chosen keys from this pool are assigned to its (limited) key ring, yet these *assigned keys are not removed from the initial pool.* Hence the next node can also get some of the previously assigned keys. Due to the birthday paradox, probability of sharing at least one common key between two neighbours is surprisingly high even for a small size of the key ring (e.g., 100 keys). That makes this EG scheme suitable for memory-constrained sensor nodes. Resilience of known pre-distribution schemes against the threat is evaluated by Chan et al. [5]. Attacker obtains some keys from the initial key pool by picking and reverse-engineering captured nodes. These keys are then used to decrypt eavesdropped messages. The success rate of decryptions depends on the number of compromised keys (nodes). We argue that multiple occurrence of the same key on the multiple nodes used for authentication purposes is not a real problem if we ensure that a new node comes from the same deploying authority rather than actually care about the identity of the node itself (can be viewed as a variation of group authentication).

Chan et al. [5] extends the EG schema by q-composite random key pre-distribution, requiring at least q shared keys instead of one (referred to as q-EG). Link key is constructed using hash function from at least q shared keys. The number of a required shared keys makes it exponentially harder for an attacker to compromise the link key with a given subset of already compromised keys, but also lowers the probability of establishing a link key. If the node key ring size m is fixed, total size of key pool S must be reduced to preserve same key establishment probability, and thus the attacker obtains a larger fraction of S from a single node. A formula for optimum tradeoff is given. Impact of multi-path key reinforcement, introduced in [3] together with q-EG is studied. The random pairwise scheme is also described (see 0.2.2).

Pietro et al. [21] extends the EG scheme using pseudo-random generation of key indexes rather than completely random (referred to as seed-based key deployment). The advantage is that two neighbours can compute identification (not the key value) of their shared keys only from their node identifications with no additional communication messages. A co-operative version of the seed-based key deployment protocol is described, performing secrecy amplification with a set of common neighbours of participants A and B. A chooses randomly the set of B-neighbours (mediators C_i) and asks them for computation of $HMAC(ID_A, K_{C_iB})$. Resulting values from each mediator are XORed together with the original key value K_{AB} and used as the new key value. Node B can compute new key value only from the information who were the mediators used, with no additional messages.

As one key is known to more than two nodes, node-to-node authentication cannot be provided in contrast to the pairwise key pre-distribution. The q-EG scheme [5] provides significantly better node-capture resilience than basic EG [11] until some threshold is reached.

0.2.2. Pairwise key pre-distribution. Pairwise key pre-distribution scheme is a scheme where a given key is shared between two nodes. In a basic pairwise scheme, each node shares a unique key with every other node in the network (referred to as (n-1) pairwise scheme). This scheme is perfectly resilient against the node capture[2], but is poorly scalable and has high memory requirements. Note that perfect resilience against the node capture does not mean that an attacker cannot obtain a significant advantage by combining keys from the captured nodes (e.g., collusion attack [17]).

Chan et al. in [5] propose a modification of the basic pairwise scheme (referred to as CPS scheme). Based on the required probability p that two physical neighbours will share a key, unique pairwise keys for X are generated, but only for m other randomly chosen nodes. In a contrast to the EG scheme, node-to-node authentication can be performed. Total number of nodes in a network is limited by $n = m/p$. Support for distributed revocation of a compromised node is proposed. During the initialisation phase, each node Y_i sharing key with node X obtains also a secret voting information, which can be used against malfunction node X when detected. The vote can be then broadcasted and node X marked by Y_i as revoked if the number of received votes exceeds a specific threshold value. Merkle hash tree is used to decrease storage needs. A masking mechanism that allows only direct neighbours of X to vote against X serves as a prevention to the revocation attack, where an attacker uses captured votes against legal nodes. A valid vote key can be constructed after deployment only if the masked key is combined (e.g., XORed) with some secret information carried by X.

A key pre-distribution scheme (referred as Blom scheme) that allows any pair of nodes to find a pairwise secret key is proposed by Blom [4]. Blom scheme requires substantially less memory than (n-1) pairwise key scheme and still allows for computing pairwise keys between each two nodes. However, Blom scheme is perfectly resilient only if not more than λ nodes are compromised (λ-secure property). If only one global key space of Blom scheme is used, λ must be unwieldy high and so does the required memory to resist against the node capture. Scalability of such approach is then poor.

A solution based on multiple key spaces is proposed by Du et al. [9] (referred to as DDHV scheme). Instead of one global key space a large key pool S of key spaces KS_i is generated and m randomly chosen key spaces KS_i are assigned to each node, analogically as for the EG scheme (see 0.2.1). The basic Blom scheme is used for each separate key space. Whole approach can be viewed as a combination of the EG key pool scheme and single space approaches like Blom's one. Probability that two nodes can establish a pairwise key is equal to the probability that they share at least one key space.

DDHV scheme provides a very good node capture resilience in comparison to EG and CPS schemes until some threshold value of total number of compromised nodes is reached. Then the whole network rapidly becomes completely insecure.

Hwang and Kim [14] revisit the basic random pre-distribution EG scheme (0.2.1), CPS scheme (0.2.2) and DDHV (0.2.2) using the giant graph component theory by Erdös and Réney to show that even when the number of a node's neighbours is small, most nodes in the whole network stay connected. If the network connectivity requirements are weakened only to some big graph component (e.g., 98% of nodes), substantial improvements of local connectivity or lower memory requirements can be obtained. Results for various trade-offs between connectivity, key ring size and security are presented [14]. Because of optimal network capacity, average node degree between 5 and 8 is suggested. Local flooding and mediator support approaches are proposed to establish link keys between two yet unconnected nodes.

[2]No other keys are compromised but from the captured nodes.

The hypercube pre-distribution based on multiple key spaces of Blom's polynomial is proposed by Liu and Ning [15]. Prior to deployment, the nodes are arranged in a virtual hypercube (so-called grid in two-dimensional case) and shared Blom's polynomials are assigned to all nodes having same coordinates within a given dimension (same row or column for grid case). This scheme is inspected in more details in sections 0.3.2 and 5 as it is closely related to our work.

Summary of random pre-distribution schemes covering EG scheme, q-EG scheme, CPS scheme and multi path key reinforcement can be found in [6].

Impact of node replication (Sybil) attack against EG, q-EG and Blom scheme is evaluated by Fu et al. [12]. The work evaluates how much can an adversary gain after injecting certain number of replicated nodes and which scheme is most resilient against the replication attack, both through theoretical and experimental results. It is shown that success of the replication attack grows with the network density.

A novel collusion attack against the pairwise key pre-distribution schemes is presented by Moore [17]. Compromised nodes are sharing their secrets to increase probability that one of them will be able to establish the link key with its neighbours. This attack differs from the Sybil attack as node identities are not randomly generated, but instead are reused according to the available pairwise keys. A distributed voting scheme can be undermined by a 5% colluding minority since this minority is able to establish approximately one half of valid communication channels.

The pairwise key schemes can provide node-to-node authentication, but support a lower number of nodes in the network in comparison to the EG scheme. Combination of the ideas from EG scheme and Blom scheme (DDHV scheme) provides better resilience against node capture, until some threshold is reached (see Figure 2.1).

0.3. Seed-based pre-distribution. Seed-based pre-distribution is an extension of a given pre-distribution scheme, introduced by Pietro et al. in [21]. Rather than completely random, a pseudo-random generation is used to determine key indexes of the keys that will be assigned to a given node. The advantage is that two neighbours can compute identification (not the key value) of their shared keys only from their node identifications with no additional communication messages. Suitable key assignment rule for probabilistic pre-distribution can be constructed in the following way:

1. Generate an initial key pool with $poolSize$ keys inside.
2. Generate a random identity ID_x for a new node from large space (e.g., 16B).
3. Use ID_x as the initial seed for a pseudo-random generator and generate the set of pseudo-random values $R_i, i \in \{1, ..., ringSize\}$.
4. For each R_i calculate $IDK_i = R_i$ modulo $poolSize$.[3]
5. For each IDK_i, assign the node ID_x with the IDK_i-th key from the initial key pool.

This process is directly usable for the EG pre-distribution scheme. With small changes, it can be also used with others. The key thing here is the fact that identity of keys carried by the node can be computed by other locally, without any additional communication except for retrieving the target node's ID.

0.3.1. Selective node capture attack. The seed-based pre-distribution suffers from an important weakness with respect to the attacker capable of performing selective node capture as described by Huang et al. [13]. As the identification of all

[3]Note that for equal probability of all possible values of IDK_i, the greatest common divisor of maximum value of R_i and $poolSize$ should be equal to $poolSize$.

carried keys can be computed from a node's ID alone, knowledge of a node's ID can be utilised by an attacker to selectively capture such nodes that maximise the number of compromised keys. To prevent this, the following defense can be used, inspired by Merkle's work on puzzles [16]. Existing nodes in the network do not use their original IDs for communication, they use fresh randomly generated identifiers instead. The original ID of a particular node as used for the seed-based pre-distribution is only known to the node's direct neighbours as follows from the proposed scheme below.

Key discovery between direct neighbours deployed in the first round is not based on the exchange of nodes' IDs, but on a more communication expensive exchange of "puzzles" created using carried keys. At first, both neighbouring nodes A and B generate separate random challenges N_A (node A) and N_B (node B) and exchange them in plaintext. Node A then computes set C_A of "puzzles" $C_{Ai} = MAC_{K_i}(hash(0|N_A|N_B))$ using all its keys. A similar set C_B is computed as $C_{Bi} = MAC_{K_i}(hash(1|N_B|N_A))$ by the node B. The way how the values N_A and N_B are combined serves as a protection against an active attacker trying to obtain a valid MAC with the key K_i applied to a selected value. Note that the size of sets C_A and C_B is equal to the node ring size. Both sets C_A and C_B are exchanged between A and B. The node A then locally computes the set C'_B in the same way C_B is constructed, but using its own keys and then checks C'_B and C_B for intersecting values. The keys used by A to create intersecting values are the keys shared with the node B. A similar process is used by the node B. The shared keys are used to establish a secure channel. Note that an attacker does not obtain any information about the keys carried by any node during this process. Original node's ID is exchanged later only if a secure channel can be set up using shared keys between neighbours. An attacker thus does not have any information about a particular node's ID until she captures the node itself, or one of its direct neighbours.

Note that there can be a significant communication overhead when puzzles are exchanged. This can be reasonable as it has to be performed only once before the secure link is established. Identity of a new node joining the group can be then broadcasted over secure links only by one of the group members.

0.3.2. Hypercube pre-distribution . The extension protocol for probabilistic polynomial pre-distribution called hypercube scheme is presented in [15]. Before the deployment, the nodes are arranged into a layered hypercube-like structure (grid in case of two-dimensions). The basic pool of key spaces is generated, same as for the multi-space polynomial scheme [9]. Then nodes with the same index within a given dimension (rows and columns in 2-dimensional case, shown on Figure 0.1) are assigned with a polynomial from the same key space and thus are able to establish shared pairwise key directly. Nodes are then deployed and perform ordinary neighbour discovery phase. If two nodes A and B wish to establish a pairwise key, all indexes of nodes within all dimensions are compared. If at least one index is shared then a key can be directly established. Otherwise other nodes are asked for cooperation such that virtual path from A to B can be formed $(A, C_1, C_2, \ldots C_n, B)$, where A is able to establish direct pairwise key with C_1 (they have same index in at least one dimension), C_i with C_{i+1} and C_n with B. New pairwise key is then generated on A and transported with re-encryptions over intermediate nodes C_j to node B. The proposed scheme assumes that compromised nodes/links are known and thus the non-compromised path can be selected. With the growing dimension of the hypercube, number of such paths is significant. If at least one non-compromised path exists, secure pairwise key can be established. Knowledge of compromise nodes/links is vital for the scheme

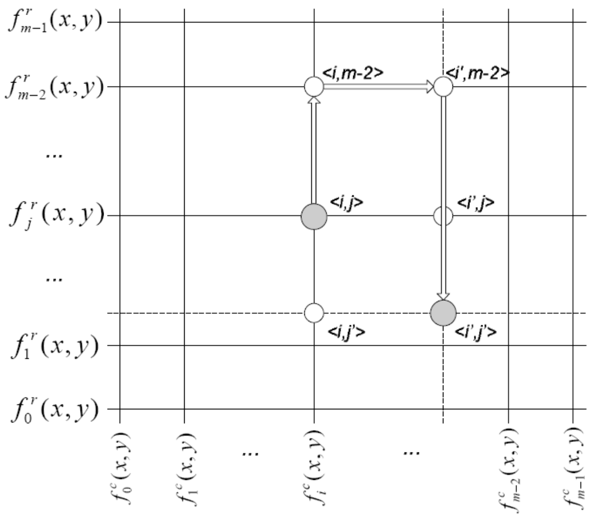

Fig. 0.1. *Polynomial assignment in two-dimension hypercube (grid). Nodes in the same row or column can establish key directly, otherwise with the help of neighbours in virtual grid. Figure taken from [15].*

– otherwise all possible paths must be tried with related significant communication overhead to obtain level of node capture resilience analyzed by authors of the scheme. Moreover, some matching between virtual hypercube layout and physical layout of nodes after deployment should be maintained. Otherwise nodes close on a virtual path can be in distant parts of the network physically, connectable only by the multi-hop communication and key exchange then poses a significant communication overhead.

The node capture resilience is significantly increased by the hypercube scheme as presented on Figure 0.2. Comparison with our group supported scheme (will be described in next section) is presented in section 5.

1. Group supported key exchange. We aim to achieve secure authenticated key exchange between two nodes, where the first node is integrated into the existing network, in particular knows IDs of its neighbours and has established secure link keys with them. The main idea comes from the behavior of social groups. When Alice from such a group wants to communicate with a new incomer Bob, then she asks other members whether anybody knows him. The more members know him, the higher confidence in the profile of the incomer. A reputation system also functions within the group – a member that gives references too often can become suspicious, and those who give good references for malicious persons become less trusted in the future (if the maliciousness is detected, of course).

1.1. Authenticated key exchange with group support. We adapt this concept to the environment of a wireless sensor network. The social group is represented by neighbours within a given radio transmission range. The knowledge of an unknown person is represented by pre-shared keys. There should be a very low probability of an attacker to obtain a key value exchanged between two non-compromised nodes and thus compromise further communication send over this link. Key exchange authentication is implicit in such sense that an attacker should not be able (with a high probability) to invoke key exchange between the malicious node and a non-compromised

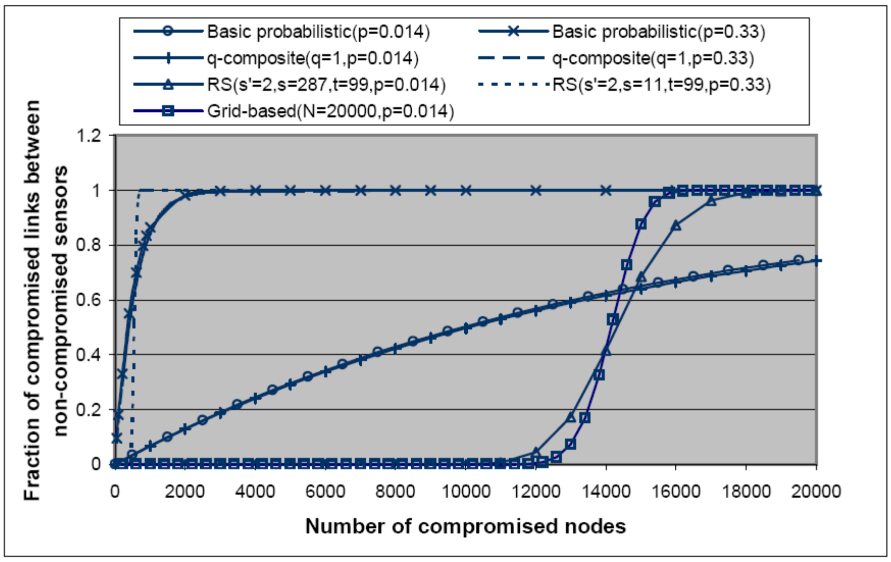

FIG. 0.2. *Node capture resilience of hypercube scheme, ring size equal to 200 keys. Figure taken from [15].*

node. Only authorised nodes should be able to do so.

In short, A asks her neighbours inside group around him to provide "onion" keys that can also be generated by the newcomer B. The onion keys are generated from a random nonce R_{ij}, R_B and keys pre-shared between A's neighbours and B. All of these onion keys are used together to secure the transport of the key K_{AB}. The valid node B will be able to generate all onion keys and then to recover K_{AB}.

Note that the key exchange secured only by the basic EG scheme is a special case of group supported protocol with the group size equal to one.

The protocol consists of the following steps:

1. The node B generates a random nonce R_B and sends message ("hello",B,R_B) to A.
2. The node A does for each neighbour node N_i, including itself, the following:
   (a) Based on the seed based pre-distribution, A is able to decide without any communication overhead whether N_i shares any key with B. Steps 2b to 2d are then executed only if there are any shared keys.
   (b) A sends ID of B and R_B to N_i using a secure channel shared with N_i.
   (c) N_i computes ID(s) $\{ID_{i1}, ID_{i2}, \ldots, ID_{im}\}$ of keys shared between B and N_i. Again, this can be done without any additional communication due to the seed-based pre-distribution.
   (d) For each shared key ID_{ij}, N_i generates a random nonce R_{ij} and computes onion key $K\prime_{ij} = hash(K_{ID_{ij}}, R_{ij}, R_B)$. $(K\prime_{ij}, ID_{ij}, R_{ij})$ is sent back using the secure channel between A and N_i.
3. If the number of distinct shared keys among all neighbours is less than the threshold given as a preset global parameter by the network owner, A refuses to exchange the key with B and quits.
4. A generates key K_{AB} that she wishes to exchange securely with B.

5. A applies all onion keys K'_{ij} over the K_{AB} value in the onion encryption fashion, $EK' = E_{K'_{11}}(E_{K'_{12}}(\ldots((K_{AB})\ldots))$. The order of application of keys K'_{ij} is given by the order of respective ID_{ij} within B's key ring and can be encoded as a bit mask *bitMask* indicating which keys were used.[4] This is done without any additional communication and with a small message.
6. A sends to B message $\{M|MAC_{K_{AB}}(M)\}$, where $M = \{R_B|\{R_{11}, \ldots, R_{ij}\}|bitMask|EK'\}$ with R_{ij} sequenced by the order of usage given by *bitMask*.
7. B uses *bitMask* to determine which keys from its key ring were used for the generation of onion keys. B then uses R_{ij} and R_B to generate onion keys as described in step (2d) and removes onion encryption layers step by step. Recovered key K_{AB} is then used to check integrity of the original message M.
8. B sends back to A the value $C_R = MAC_{K_{AB}}(hash(R_B, R_{11}|R_{12}|\ldots R_{ij}))$ as a confirmation of a correctly and completely decrypted message M.
9. A verifies the correctness C_R and then sets K_{AB} as a node to node key for communication with node B.

1.2. Probabilistic authentication. This protocol can be also used as a building block for probabilistic entity authentication. Probabilistic authentication means that a valid node can convince others with a high probability that it knows all keys related to its ID. A malicious node with a forged ID will fail (with a high probability) to provide such a proof. We propose to use the term *probabilistic authentication* for authentication with following properties:

1. Each entity is uniquely identifiable by its ID.
2. Each entity is linked to the keys with the identification publicly enumerable[5] from its entity ID.
3. A honest entity is able to convince another entity that she knows keys related to her ID with some (high) probability. For practical purposes, this probability should be as high as possible to enable authentication between as many other nodes as needed.
4. Colluding malicious entities can convince other entities that they know keys related to an ID of a not-captured node only with a low probability.

This approach to authentication enables a tradeoff between security and computation/storage requirements for each entity. As a potential number of neighbours in WSNs is high, memory of each node is severely limited and an attacker can capture nodes, this modification allows us to obtain *reasonable* security in the WSN context.

No modification of the proposed protocol core is necessary for authentication purposes: if the node B wants to authenticate itself to A, then it will do so by sending its ID, which will initialize a key exchange as described above. Node A accepts authentication only if B is able to respond with a valid C_R in the 8th step.

However, special attention must be paid to the actual meaning of the authentication in case of a group supported protocol. Firstly, as we assume that a part of the group can be compromised, verification of B's claims can be based on a testimony of compromised neighbours. Secondly, node A has a special role in the protocol execution

[4]As only the keys from B's ring can be used, the bit mask will have the size of *ringSize* bits. Due to the seed-based pre-distribution, keys in B's ring can be ordered by the sequence of the key identity generation, e.g. the first value obtained from the pseudo-random generator corresponds to the first key and to the first bit in the bit mask. Value '1' of the i-th bit of the mask signalizes that i-th key was used for onion encryption.

[5]Only the key identification can be obtained from entity ID, not the key value itself.

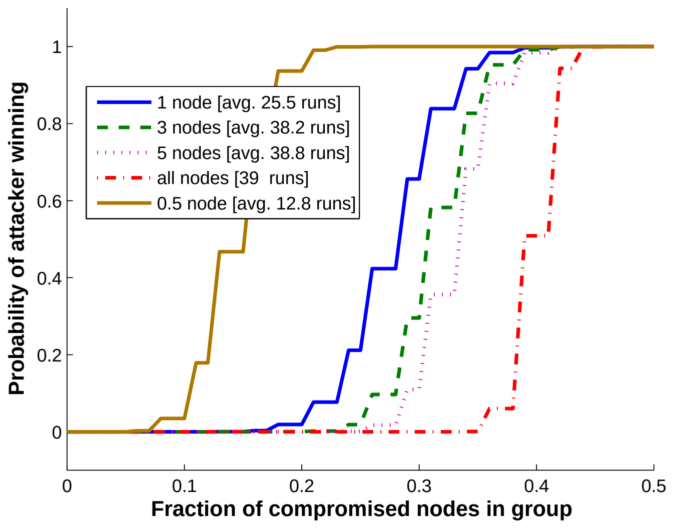

FIG. 1.1. *Probability of an attacker winning in majority decision voting described in 1.2.1 and number of expected runs of the protocol w.r.t. number of nodes selected as central by each member of a group (39 nodes per group).*

and can be compromised as well. The neighbours should not rely on an announcement from node A that node B was correctly authenticated to it. The special role of A can be eliminated if the protocol is re-run against all members of the group with a different central node each time. Yet this would introduce an additional communication overhead compared to the approach where a single member of the group will announce result of the authentication to others. Note that the number of messages for a single protocol run is reasonably low due to the seed-based pre-distribution.

1.2.1. Probabilistic authentication with majority decision. The following tradeoff between security and communication overhead can be introduced: The protocol is re-run k-times only with randomly selected central nodes and a majority rule is applied in order to obtain a result for the whole group. The random selection resilient against certain fraction of compromised nodes can be done as follows:

1. Every node broadcasts a set of p randomly selected IDs of its neighbours.
2. Each selected node performs the protocol as a central node with B and distributes the result using authenticated channel to each of the requesting nodes.
3. Every node then separately applies majority rule over p responses from its set, obtains partial result M_i and then broadcast it through an authenticated channel.
4. The majority rule is applied again over all partial results M_i by every node to obtain the final result of the authentication.

Note that the requirement of randomly selecting the "central" nodes is critical, otherwise an attacker can force a selection of compromised nodes only and will be certainly successful when controlling at least $\lceil k/2 \rceil + 1$ nodes inside group. See Fig. 1.1 for probability of attacker's success for different values of p and of the compromised fraction of a group.

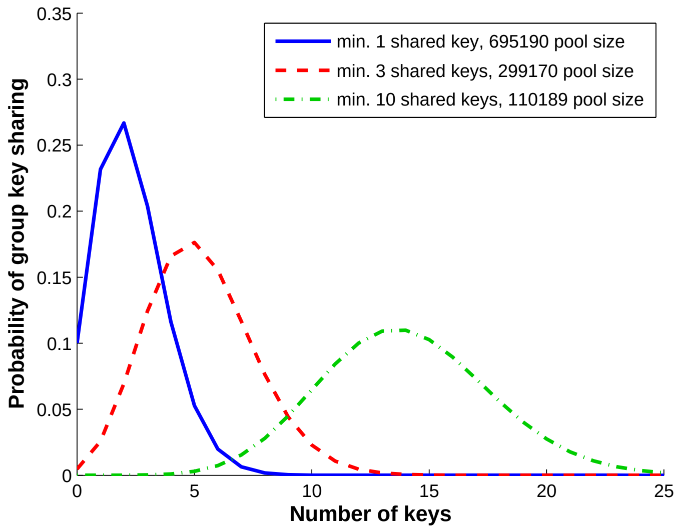

FIG. 1.2. *The distribution of probability of group key sharing (group size 40 nodes, ring size 200 keys, 90% constant probability of sharing at least required number of keys). Note that the pool size decreases with a growing minimum of required shared keys.*

1.3. Evaluation of the communication and computation overhead. Number of additional (to a basic key exchange between two nodes only) messages is at most equal to 2 * number of applied onion keys. We require two additional messages per single neighbour that shares at least one key with B. Most commonly, a neighbouring node will share only one key with B (otherwise pool size can be set significantly larger for particular settings). Threshold of minimum of required keys for exchange is given by a global preset security parameter T. There can be key exchanges with more onion keys applied. For a fixed pool size, more applied keys mean lower probability that an attacker will be able to decrypt a message so more applied keys mean higher communication overhead (more neighbours to be contacted), but also direct increase in the exchange security. Most commonly, there will not be significantly more applied keys than the preset parameter T (otherwise the pool size can be again set significantly larger). As a result, there will be only about twice as many additional messages than the required preset treshold T for minimum of required keys. Our experiments reveal this T is most commonly expected to be between 1 to 5, and the communication overhead is then very reasonable.

The size of messages exchanged between A to N_i in steps 2b and 2d is small, only the message sent from A to B in step 6 is larger. This message carries information about used keys and random nonces R_i. The information about used keys takes m bits as explained in step 5, where m is the ring size. E.g., for a scenario with a 3-key treshold, 200 keys in the ring and 16-byte identificators/nonces/keys, this message will be around 120 bytes[6].

The computation overhead consists of additional encryptions/decryptions, hash function computations and random number generation. There will be additional encryptions given by the number of applied onion keys. Same holds for number of

[6] 4 * 16B nonces + 25B used keys + 16B K_{AB} + 16B $MAC_{K_{AB}}(M)$

decryptions and random number generations.

We would like to stress that the overhead is independent of the group size (larger group allows to set up a higher treshold for minimum shared keys). If more than T keys are shared between the group and B, then the overhead will increase (by a fraction of additional shared keys) but the exchange will have a higher probability of being secure.

2. Group support with EG scheme. Our work has been motivated by poor node-capture resilience (NCR) of known PKPSs. We evaluate NCR improvements obtained by the group support protocol with the EG scheme as the underlaying PKPS as well as providing details of calculating relevant probabilities. The analytical results were experimentally verified by series of simulations with 40000 nodes on network simulator.

Lemma 1: Keys capture probability – EG scheme.
The probability that an attacker will know each key from k randomly chosen keys after capturing c random nodes, where m is size of the node's key ring and S is the key pool size:

$$P_{KC}(k,c) = \left(1 - \left(1 - \frac{m}{S}\right)^c\right)^k$$

Proof: The probability that a particular key will not appear in a key ring of any captured node is equal to $(1 - \frac{m}{S})^c$. The complement gives us the probability that a particular key will be captured and this must hold for each of the k keys.

Lemma 2: Group key share probability.
The probability that a group G of n randomly chosen nodes will share exactly k keys with another randomly chosen distinct node B:

$$P_{GKS}(k,n) = \binom{m}{k} * \left(1 - \left(\frac{S-m}{S}\right)^n\right)^k * \left(\frac{S-m}{S}\right)^{n*(m-k)}$$

Proof: The probability that a particular key from B's key ring is shared with group G is equal to 1 - probability that this key is shared between B and no G_i node. Probability that a particular key is not shared between B and G_i is $\frac{\binom{S-1}{m}}{\binom{S}{m}} = \frac{S-m}{S}$. There is n such G_i nodes, thus $P_{keyNotShared\_1} = (\frac{S-m}{S})^n$. Then the probability that exactly k keys from B key ring are shared is equal to $P_{keyShared\_k} = \binom{m}{k}(1-P_{keyNotShared\_1})^k * (P_{keyNotShared\_1})^{m-k}$, where $(1 - P_{keyNotShared\_1})^k$ stands for exactly k keys being shared while at the same time remaining $(m-k)$ of keys in key ring are not shared. $\binom{m}{k}$ stands for the number of positions where shared keys can be placed inside B's key ring.

Lemma 3: Onion decryption probability.
The probability that an attacker will know all the keys shared between the node X (with a random ID) and group G of n randomly selected non-compromised nodes after capturing c randomly selected nodes. At least $minK$ keys are used for onion encryption:

$$\sum_{i=minK}^{ringSize} P_{KC}(i,c) * P_{GKS}(i,n)$$

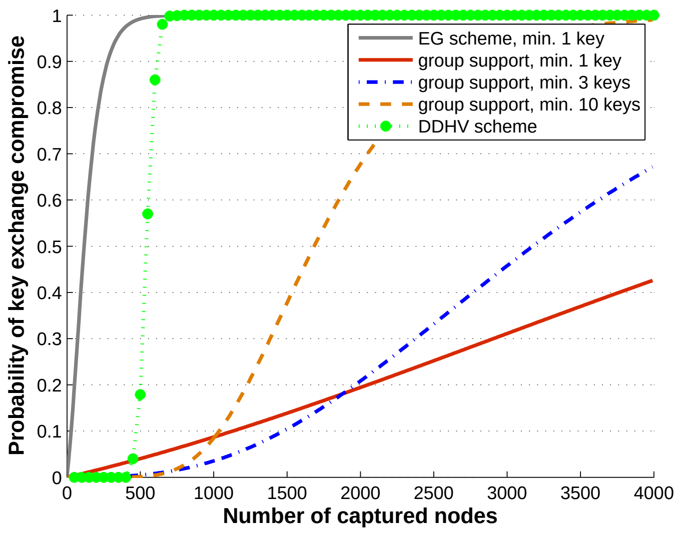

Fig. 2.1. *Node capture resilience of the basic EG scheme and the variant with group supported key exchange. Key ring size is fixed to 200 keys, group size to 40 nodes. The pool size differs with minimum of required shared keys to maintain a constant probability 90% that the exchange will be possible.*

Proof: The probability that G will share exactly i keys with X is given by $P_{GKS}(i,n)$. Probability that the attacker will know these i keys after capturing c nodes is given by $P_{KC}(i,c)$. No more than $ringSize$ keys can be shared.

2.1. Options and settings. Node capture resilience can be evaluated for particular parameters of a sensor network according to the lemmas from section 2. We assume that the maximum number of keys carried by a single node is fixed and given as a manufacturing parameter. More keys carried in a node generally imply better resilience for any PKPS.

We choose to fix this parameter to 200 keys to enable a comparison with the multi-space pairwise keys scheme [9] (DDHV scheme for short). DDHV scheme with 0.33 connection probability is perfectly secure until around 400 nodes are captured. Then it quickly becomes insecure, having more than 98% communication insecure when 700 nodes are captured.

As shown in Fig. 2.1, our protocol with 40 neighbours and required minimum of 3 shared keys results only in 0.25% compromised exchanges when 400 nodes are captured. When 700 nodes are captured, only around 1.3% exchanges are compromised. More than 3000 nodes need to be captured to compromise half of the key exchanges. The relation between the increasing number of required shared keys and the resilience is as follows: If the pool size and ring sizes are fixed, then a higher value of minimum of required shared keys implies a decrease in the probability of the group sharing enough keys with the new node. To increase this probability, the pool size must be decreased. Smaller pool size implies a higher fraction of keys captured by the attacker after a node compromise.

In case that even better resilience for a low number of captured nodes is required, minimum of required shared keys can be increased. For example, when 10 keys are

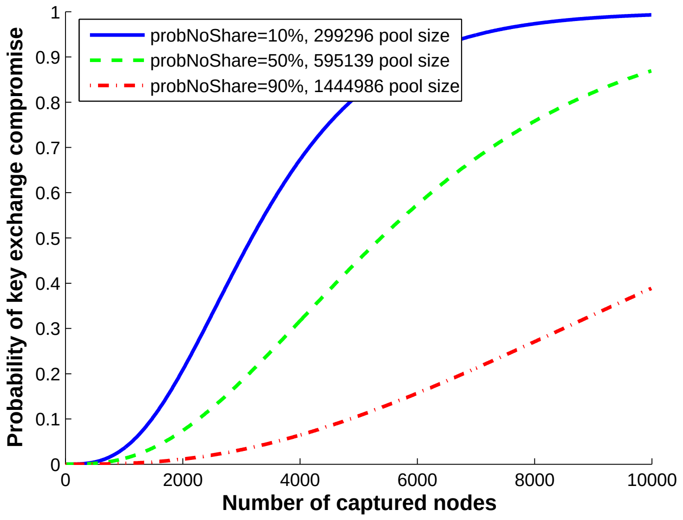

FIG. 2.2. *Impact of varying the probability of impossible key exchange to node capture resilience for minimum required keys $T = 3$. The network parameters are same as for Fig. 2.1. The group enlarged to 160 nodes will decrease respective $probNoShare$ values to 0.0001% / 0.2% / 18.43%, keeping the pool size and node-capture resilience same.*

required, there is only around 0.025% compromised exchanges for 400 captured nodes. However, there is a tradeoff introduced: half of the compromised exchanges is reached faster (around 1700 captured nodes).

As we target static WSNs with immobile nodes (after the deployment), we choose to calculate the appropriate pool size value such that there is $probNoShare = 10\%$ probability *not* to find the required amount of shared keys. In such cases, the protocol will abort in the 3rd step. This situation can be solved by increasing the group size by involving neighbours two hops away at the cost of additional communication (see 3 for details). Based on the usage scenario, $probNoShare$ can be set to a higher value, e.g. if the node can move to another location or if fraction of nodes can be "sacrificed" for sake of better network resilience. This increases the pool size and thus consequently increases the node capture resilience. Example of impact for minimum of 3 required keys is shown on Fig. 2.2.

The significant increase of NCR can be obtained when $probNoShare = 90\%$ is set. However, this means that in the basic version of protocol, only 10% of new nodes will be able to join to the group. This can be acceptable in the scenario with mobile nodes. Otherwise, group enlargement (see section 3) can be performed.

We would like to stress out that our protocols do not provide defense against Sybil [19] or collusion [17] attacks, where direct clones of the captured node are populated over the network. Here we rely on some replication detection algorithm like [20] by Parno et al. Design of such efficient protocol with a reasonable communication overhead is still an open question.

3. Group enlargement. If a node is capable to remember IDs of nodes that are two hops away (direct neighbours of its direct neighbour), then the size of the group

1 minimum key required		3 minimum keys required	
basic group	enlarged group	basic group	enlarged group
10%	0.1%	10 %	$< 10^{-5}$ %
50 %	6.25 %	**50 %**	**0.16 %**
90 %	65.59 %	90 %	18.43 %

TABLE 2.1
Decrease of probability of impossible key exchange probNoShare when also the nodes two hops away are used. Basic group consists of 40 nodes, the enlarged group of 160 nodes (assumed to be reachable in two hops). Results valid for both EG and multi-space polynomial underlaying pre-distribution schemes.

will increase fourfold. Yet the number of additional messages will increase only twice due to the need for message routing (two hops instead of one). The total number of contacted nodes will remain the same and – due to the seed-based pre-distribution – no messages are required to compute IDs of nodes that will be contacted. The group enlargement can be employed in the following scenarios:

- There are too few direct neighbours for creating a group capable of executing the protocol with a required node capture resilience.
- Not enough keys are shared between the group and the incoming node B.

The table 2.1 shows the impact of the group enlargement in case of 1, resp. 3 minimum required keys with 40 nodes reachable in one hop. Note that the increase in probability of possible key exchange is more significant with more minimum keys required.

Together with the results shown in Fig. 2.2, one can set a larger key pool, obtain better NCR and ask nodes two hops away in case of missing keys. E.g., when the basic group has 40 members and T = 3 minimum keys are needed, probability of possible key exchange can be set to 50%. Then approximately a half of the requests will invoke the need for group enlargement (approximately 160 nodes in the enlarged group) and only less than 0.2% of valid nodes will be rejected in total.

Even after group enlargement, the communication overhead remains reasonable and proportional to the required security given by the threshold T. Only such nodes that are actually sharing key with incoming node B are contacted. Increased messages comes only from the fact that nodes in the enlarged group are not reachable directly, but more intermediate nodes must be involved in a multi-hop communication. For example, if the group consist from the nodes up to 2-hop away, communication overhead will increase twice with the energy consumption distributed regularly over nodes in the group. On the other side, the storage overhead increases more significantly, as each node must remember IDs of the nodes two hops away. Note, that IDs of direct neighbours is most probably stored for routing purposes anyway.

The tradeoff between communication and storage overhead can be introduced here: the central do not store all IDs of the nodes up to two hops away, but rather ask his direct neighbours to provide list of their neighbours for the expense of few additional messages only when necessary for protocol run. However, possibility of an attacker injecting fake ID through a compromised node must be considered.

4. Group support with multi-space polynomial scheme. Basic evaluations of group supported protocol properties were provided, with EG scheme as the underlaying pre-distribution scheme. Other pre-distribution can be transparently used as well. Here we will discuss polynomial-based pre-distribution introduced in [9]. In-

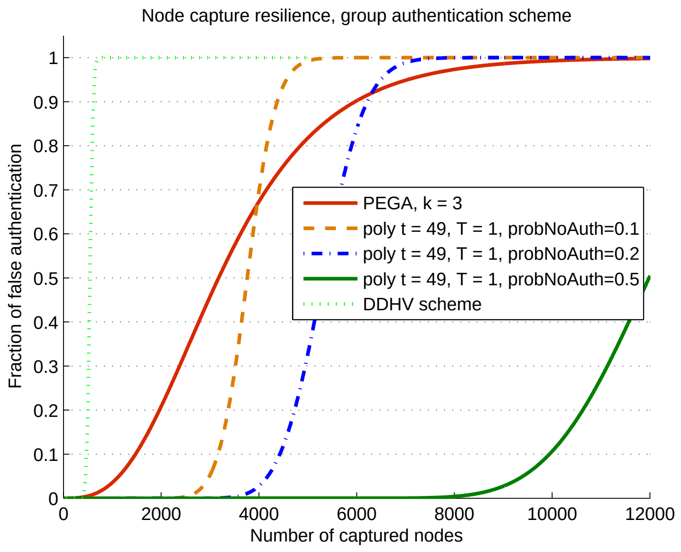

FIG. 4.1. *Node capture resilience of group supported scheme with Blom's treshold secret sharing scheme as an underlaying pre-distribution. Group size 40 nodes. Ring size is equal to 200 keys.*

stead of assigning direct keys as in the EG scheme, we select Blom's key space from key spaces pool during the pre-deployment. Selection is again done according to seed-based pre-distribution. For each selected space, Blom's polynomials are generated. To keep the same ring size, there can be up to $numberOfPolynomials = \lfloor m/c \rfloor$ polynomials, where m is node ring size and c is degree of Blom's polynomial. We choose to fix $c = t + 1$, where t is Blom's threshold security parameter to minimize memory footprint of one polynomial. Limitation is that when using this settings, only up to c nodes can be assigned by different polynomial share from one polynomial space and this may limit total supported network size. However, as we will be using group support, the probability of sharing key space between two nodes will be much lower than probability in the original multi-space polynomial scheme and thus the supported network size remains sufficient.

Lemma 4: Probability of i shares compromise.
Probability, that attacker will be able to compromise exactly i shares of particular polynomial after capturing of c nodes, where S is the key pool size, m' is number of polynomials in one node's key ring ($m' = m/d$) and d is degree of polynomial:

$$P_{CS}(i) = \binom{c}{i} * \left(\frac{m'}{S}\right)^i * \left(1 - \frac{m'}{S}\right)^{c-i}$$

Proof: Particular polynomial cannot be compromised until an attacker compromise at least $(t + 1)$ shares of this polynomial (proof given in [4]), where t is pre-specified threshold. The probability that given polynomial was chosen for a sensor node is $\frac{m'}{S}$. To compromise exactly i shares of this polynomial, share from this polynomial must be captured exactly i times from the captured nodes with share and not being present on remaining $c - i$ nodes. There is $\binom{c}{i}$ ways how the i compromised shares can be

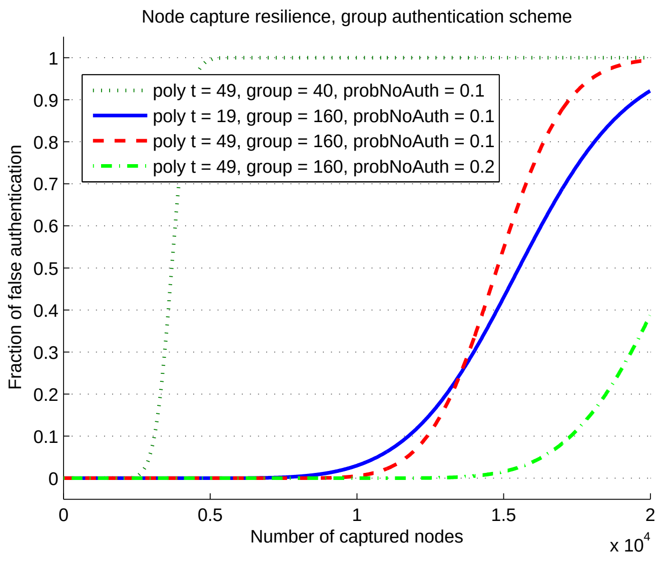

FIG. 4.2. *Node capture resilience of group supported scheme with Blom's treshold secret sharing scheme as an underlaying pre-distribution. Group size 160 nodes. Ring size is equal to 200 keys.*

distributed among c captured nodes.

Lemma 5: Probability of polynomial compromise.
The probability that an attacker will know all shares necessary to compromise every polynomial from k randomly chosen polynomials after capturing c random nodes is given by:

$$P_{KP}(t) = \left(1 - \sum_{i=0}^{t} P_{CS}(i)\right)^k$$

Proof: The sum gives as the probability, that attacker will compromise less or equal to t shares from particular polynomial. If the attacker compromise more than t shares, than particular polynomial is compromised and this must hold for every of the k polynomials.

4.1. Impact of polynomial security threshold. Impact of different degree of the polynomials is shown of Figure 4.3 (basic group with 40 neighbours) and Figure 4.4 (enlarged group with 160 neighbours). Again, the original ring size is assumed to allow for up to 200 ordinary keys and the number of selected key spaces and number of stored polynomials on every node are set to fit this memory restriction as $numberOfPolynomials = \lfloor 200/c \rfloor$. Tested polynomial degrees were 5 (40 polynomials per node), 10 (20), 20 (10), 40 (5), 50 (4) and 66 (3). Larger degrees of polynomial were not tested as the resulting node capture resilience do not increase for our settings.

4.2. Impact of minimal shared keys threshold. Similarly to the group supported scheme with EG pre-distribution, threshold of minimal keys can be required in step 3 of the protocol. As we are using multi-space polynomial scheme now, the

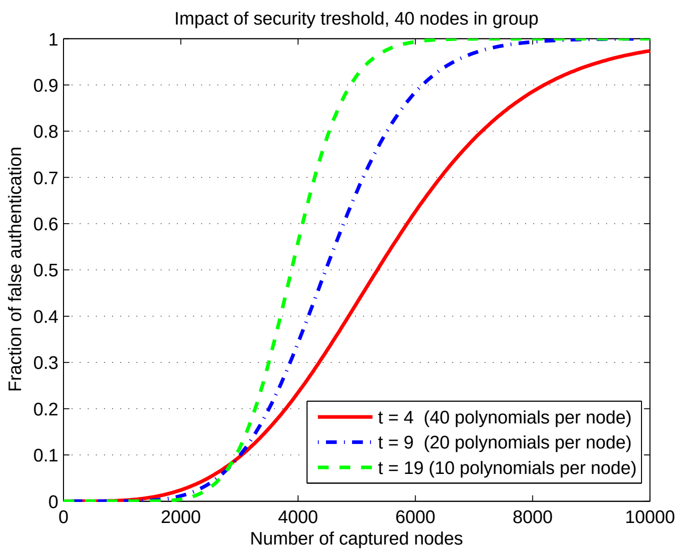

Fig. 4.3. *Impact of number of polynomial security threshold. Group size is 40 nodes. Ring size is equal to 200 keys.*

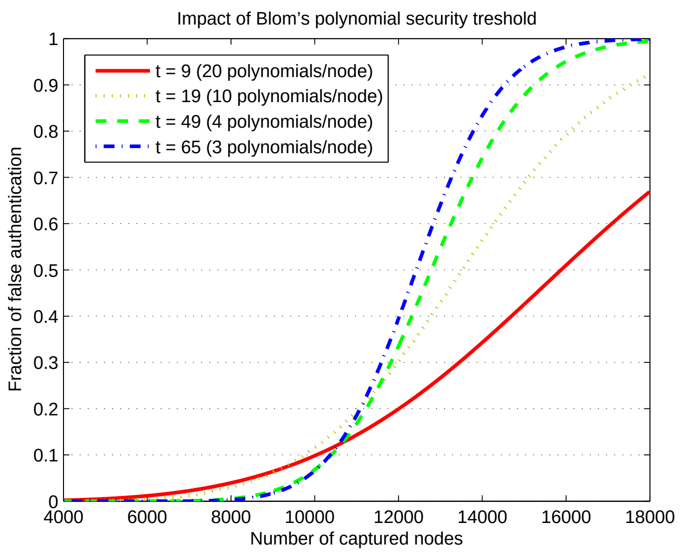

Fig. 4.4. *Impact of number of polynomial security threshold. Group size is 160 nodes. Ring size is equal to 200 keys.*

threshold of minimal shared key spaces is checked. The results are significantly different from the EG scheme as shown on Figures 4.5 (basic group with 40 neighbours) and Figure 4.6 (enlarged group with 160 neighbours). Increased threshold of minimal required shared keys does not improve the node capture resilience. As the single polynomial requires more memory to store than single key in EG scheme requires, there is a lower number of key spaces in the key pool for polynomial scheme than the

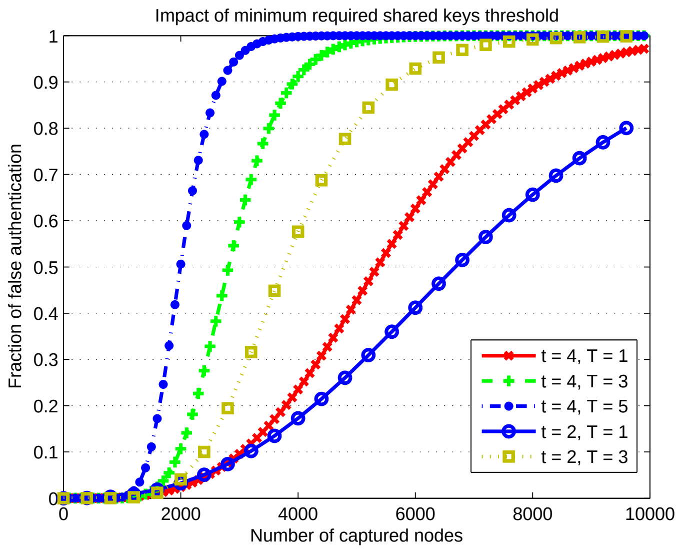

FIG. 4.5. *Impact of number of minimum required keys during the group support protocol. Group size is 40 nodes. Ring size is equal to 200 keys.*

number of simple keys in key pool in case of the EG scheme. Increased threshold of minimal required shared keys (more than one) thus implies a significant decrease of the pool size to maintain fixed probability that key exchange will be possible. Resulting node capture resilience against an attacker randomly capturing the nodes is weakened. However, this threshold increases the resiliency against an attacker who tries to find the part of the network where he is able to introduce malicious node with forged ID. With increased threshold there is a higher probability that the group will know at least one key connected to this forged ID, but unknown to an attacker. The threshold thus should be set according to expected threats to particular network. The analysis shows that a lower degree of polynomial is better for higher number of minimal required shared keys (see Figure 4.5).

5. Comparison with hypercube scheme. Direct comparison between group supported and hypercube scheme [15] is not really possible, as both schemes use different assumptions and resulting node capture resilience depends on several input variables used for the analysis.

The hypercube scheme is more suitable for structured deployments with position of nodes such that real length of paths (number of hops) during key establishment is reasonable small to prevent unwanted communication overhead. Additionally, the actual compromise status of links must be known as well, otherwise all possible paths between two nodes must be used to establish a new pairwise key to obtain indicated node capture resilience. Usage of all paths is possible, but results in a very high communication overhead. The scheme has low storage overhead, as the ID of nodes necessary to form key establishment path can be computed from ID of source and target node. However, the storage of status of compromised links implies an additional overhead.

The group supported scheme is more suitable for scenarios with randomly deployed dense networks. The group support is formed from neighbouring nodes only

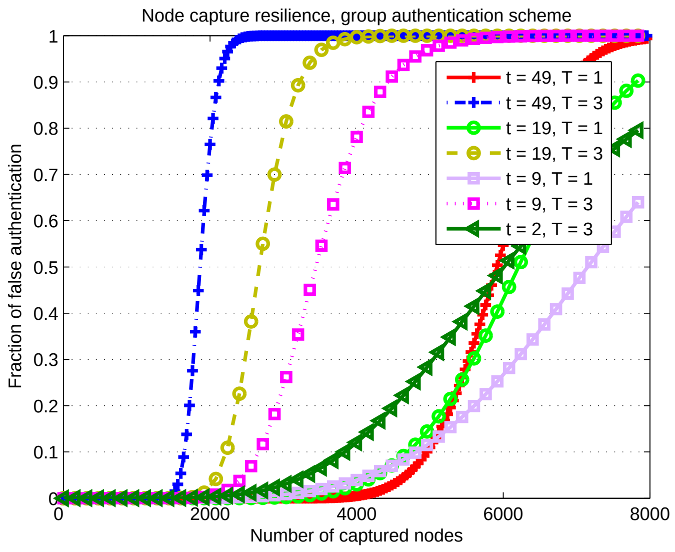

FIG. 4.6. *Impact of number of minimum required keys during the group support protocol. Group size is 160 nodes. Ring size is equal to 200 keys.*

	Group supported	**Hypercube based**
Communication overhead	Only messages to the direct radio neighbours.	If no special deployment is used, then nodes that have to be contacted can be in any part of the network.
Storage overhead	Basic keys + ID of nodes inside the group.	Basic keys only + compromise status of links.
Knowledge of compromised links	Not required.	High communication overhead if not known.
Deployment pattern	Not required, group always formed from the direct neighbours.	If not, than high communication overhead (distant nodes).
Usability with other PKPS	Any, but node capture resilience may vary. Differences only in process of selection keys (seed-based).	Analyzed for Bloom polynoms, but can be adjusted. Node capture resilience may vary.
Selective node capture	Threat as the node ID can be used to enumerate carried keys.	Low threat, an attacker may target nodes on short paths.

TABLE 5.1
Comparison of the properties of group supported and hypercube-based predistribution schemes.

and thus does not create a long communication paths. Due to seed-based predistribution, communication overhead is low, but IDs of nodes within group must be stored on each node (storage already used for routing purposes can be used to lower this overhead). Knowledge of link key compromise status is not required and group supported scheme is tolerant to a partial group compromise.

6. Possible attacks and defenses. Increased resilience does not come for free and involvement of neighbours opens possibility for new attacks. Impact and defenses against such attacks are discussed, with respect to passive and active attackers.

6.1. Passive attacker. A passive attacker can either randomly or selectively [13] compromise a fraction of nodes, extract all their keys, and access the relayed communication, but the compromised nodes *do not misbehave* in the protocol execution.

1. An attacker may try to selectively capture nodes based on the knowledge of their IDs in order to obtain most keys for its forged node ID – as described in section 0.3.1, actual IDs of nodes distributed in the first round can be kept secret just between a node and its direct neighbours, never transmitted over a non-encrypted channel.
2. An attacker will use a node with a random forged ID – based on previous analytical results, group of nodes will have a very high probability to share at least one key that the attacker does not know and thus to detect cheating.
3. An attacker will try to generate a random forged ID, for which he knows most of the keys (for feasibility evaluation – see 2, Lemma 2 – results for 200 keys in a ring show that such attack is computationally infeasible even when the attacker knows 1/3 of the initial key pool.)
4. An attacker will select such a position within the network so that neighbouring nodes only know the keys known to the attacker – there are the following defenses:
   (a) At least *minAuthKeys* are required to enable the key exchange. This security parameter prohibits poorly secured exchanges.
   (b) Use of fresh random identifier instead of original ID as described in section 0.3.1 for selective capture of nodes above to minimize attacker knowledge about nodes in network.
5. An attacker will use node(s) with same ID(s) as the captured one(s) – known as the Sybil attack. Our protocol offers no defense here, and we rely on other replication detection mechanisms.

6.2. Active attacker. An active attacker can not only do all that the passive attacker can, e.g. extract secrets from captured nodes, but also place them back to the field and actively control them during the protocol execution.

1. An attacker will compromise one of A's neighbours and supply an incorrect onion key value when asked for keys causing rejection of a valid node B. After rejection of node B, A can initiate the compromise detection phase: A gradually removes onion keys from EK' to detect when B will be able to successfully decrypt K_{AB}, detecting the incorrect onion key supplier.
2. An attacker will compromise some node N and relay part of the protocol messages for node X with a forged ID to neighbours of node N pretending that there is an authentication process between N and X going on to obtain correct onion keys usable elsewhere. Here the following policy can be introduced as a defense: N_i will not respond to N until a 'hello from X' packet from the first step of the protocol is received. The random nonce R_{ij} generated by each node prevents creation of same onion key multiple times.
3. An attacker may try to insert bogus messages impersonating some party participating in the protocol. Integrity, confidentiality and freshness of all messages from step 2 are protected by the pre-existing link secure channel. The message from step 6 is integrity-protected by the key K_{AB}. Integrity can be checked backwards after a successful recovery of the key K_{AB}. Note that a denial of service by a corruption of this message is possible here (A and B will not be able to establish shared key). However, if an attacker is able to

modify the original transmission and to insert his modified message, he can achieve the same goal only by garbling anyway. Integrity of the message in step 8 is protected with the key K_{AB}, with implications same as for step 6.

7. Conclusions. We propose a novel idea for key exchange and entity authentication based on the random pre-distribution scheme. Results of this enhancement of the EG pre-distribution scheme [11] and polynomial scheme by [9] show that a substantial node capture resilience can be obtained. Probabilistic key pre-distribution schemes generally exhibit the property that the probability of key sharing rapidly increases with the number of keys in a node's key ring. Our group supported protocol exploits this behaviour to create large virtual key rings. However, this improvement does not come for free. Firstly, some additional communication is required. We have shown that substantial improvements in the node capture resiliency can be obtained with a reasonable communication overhead that is proportional to the minimum of required shared keys (security parameter) rather than the size of supporting group. Secondly, a node relies on the security of previously established link keys with its neighbours. This opens a possibility for additional attacks, which were discussed together with possible countermeasures.

Combination of the group supported protocol with multi-space polynomial pre-distribution provides a better node capture resiliency if the polynomial scheme can be efficiently computed on the nodes. See [15] for discussion of efficient implementation.

REFERENCES

[1] Crossbow Technology, Inc. *http://www.xbow.com/.*

[2] Smart dust project website. *http://robotics.eecs.berkeley.edu/∼pister/SmartDust/.*

[3] Ross Anderson, Haowen Chan, and Adrian Perrig. Key infection: Smart trust for smart dust. In *Proceedings of the Network Protocols (ICNP'04), 12th IEEE International Conference, Washington, DC, USA*, 2004.

[4] Rolf Blom. An optimal class of symmetric key generation systems. *EUROCRYPT '84, LNCS 209*, pages 335–338, 1984.

[5] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy (SP'03), Washington, DC, USA*, pages 197–214. IEEE Computer Society, 2003.

[6] Haowen Chan, Adrian Perrig, and Dawn Song. Key distribution techniques for sensor networks. pages 277–303, 2004.

[7] Elizabeth M. Royer Charles E. Perkins. Ad hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, 1999.

[8] Josh Broch David B. Johnson, David A. Maltz. Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In Charles E. Perkins, editor, *Ad Hoc Networking*, pages 139–172. Addison-Wesley, 2001.

[9] Wenliang Du, Jing Deng, Yunghsiang S. Han, and Pramod K. Varshney. A pairwise key pre-distribution for wireless sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'03), Washington, DC, USA*, pages 42–51, 2003.

[10] Wenliang Du, Jing Deng, Yunghsiang S. Han, and Pramod K. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. *IEEE INFOCOM 2004, Hong Kong*, 2004.

[11] Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'02), Washington, DC, USA*, pages 41–47, 2002.

[12] Huirong Fu, Satoshi Kawamura, Ming Zhang, and Liren Zhang. Replication attack on random key pre-distribution schemes for wireless sensor networks. *IEEE Information Assurance Workshop, West Point, USA*, 2005.

[13] Dijiang Huang, Manish Mehta, Deep Medhi, and Lein Harn. Location-aware key management scheme for wireless sensor networks. *SASN'04, Washington, DC, USA*, pages 29–42, 2004.

[14] Joengmin Hwang and Yongdae Kim. Revisiting random key pre-distribution schemes for wireless sensor networks. In *Proceedings of the ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'04), Washington DC, USA*, pages 43–52, 2004.

[15] Donggang Liu and Peng Ning. Establishing pairwise keys in distributed sensor networks. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 52–61, New York, NY, USA, 2003. ACM Press.

[16] Ralph C. Merkle. Secure communications over insecure channels. *Commun. ACM*, 21(4):294–299, 1978.

[17] Tyler Moore. A collusion attack on pairwise key predistribution schemes for distributed sensor networks. In *Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'06), Washington, DC, USA*, 2005.

[18] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM, vol. 21, issue 12*, pages 993–999, 1978.

[19] James Newsome, Elaine Shi, Dawn Song, and Adrian Perrig. The sybil attack in sensor networks: Analysis & defenses. In *Proceedings of the third international symposium on Information processing in sensor networks (IPSN'04), Berkeley, California, USA*, pages 259–268, 2004.

[20] Bryan Parno, Adrian Perrig, and Virgil Gligor. Distributed detection of node replication attacks in sensor networks. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy (SP'05), Washington, DC, USA*, pages 49–63, 2005.

[21] Roberto Di Pietro, Luigi V. Mancini, and Alessandro Mei. Random key-assignment for secure wireless sensor networks. *1st ACM Workshop Security of Ad Hoc and Sensor Networks Fairfax, Virginia*, pages 62–71, 2003.

[22] Petr Švenda and Václav Matyáš. Authenticated key exchange with group support for wireless sensor networks. *The 3rd Wireless and Sensor Network Security Workshop; 4th IEEE International Conference on Mobile Ad-hoc and Sensor Systems, Pisa, Italy*, pages 21–26, 2007.

[23] Petr Švenda and Václav Matyáš. Key distribution and secrecy amplification in wireless sensor networks. *Technical Report, FIMU-RS-2007-05, Masaryk University, Brno*, 2007.

[24] Eiko Yoneki and Jean Bacon. A survey of wireless sensor network technologies: research trends and middleware's role. *Technical Report, UCAM 646, Cambridge*, 2005.