FI MU

Faculty of Informatics
Masaryk University, Brno

Key Distribution and Secrecy Amplification in Wireless Sensor Networks

by

Petr Švenda
Václav Matyáš

FI MU Report Series **FIMU-RS-2007-05**

Copyright © 2007, FI MU **September 2007**

**Copyright © 2007, Faculty of Informatics, Masaryk University.
All rights reserved.**

**Reproduction of all or part of this work
is permitted for educational or research use
on condition that this copyright notice is
included in any copy.**

Publications in the FI MU Report Series are in general accessible via WWW:

`http://www.fi.muni.cz/reports/`

Further information can be obtained by contacting:

**Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic**

Key Distribution and Secrecy Amplification in Wireless Sensor Networks

Petr Švenda
Faculty of Informatics, Masaryk University
xsvenda@fi.muni.cz

Václav Matyáš
Faculty of Informatics, Masaryk University
matyas@fi.muni.cz

November 19, 2007

Abstract

This report targets the area of wireless sensor networks, and in particular their security. 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.

We also propose a new method for automatic protocol generation which utilizes Evolutionary Algorithms (EA). The approach is verified on the automatic generation of secrecy amplification protocols for wireless sensor networks. All human-designed secrecy amplification protocols proposed so far were re-invented by the method. A new protocol with better fraction of secure links was evolved. An alternative construction of secrecy amplification protocol was designed which exhibits only linear (instead of exponential) increase of needed messages when the number of communication neighbours is growing. As a message transmission is a battery expensive operation, this more efficient protocol can significantly save this resource.

1 Introduction

Recent advances in micro-electro-mechanical systems technology, wireless communications, and a digital electronics 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. [SD]) as well as research in middleware architectures [YB05] and routing protocols (AODV [Cha99], DSR [DBJ01], 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 [MIC] equipped with 8-bits Atmel ATmega 128L down to Smart Dust motes [SD] with their total size around $1 mm^3$ 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, commu-

nication traffic should be kept as low as possible and most operations should be done locally, not involving the trusted BS.

The main contribution of this report is a protocol design for authenticated key exchange with an improved resilience against the threat of node capturing over existing probabilistic pre-distribution schemes and method for automatic design of secrecy amplification protocols based on Evolutionary Algorithms.

This report presents several key distribution schemes for WSNs and discuss its properties, namely the node capture resilience. We then focus on extension protocols, that are able to improve the security for the price of an additional communication. The protocol described in section 2 uses combination of group support from neighbour nodes and probabilistic pre-distribution to provide authenticated key exchange and introduce the concept of probabilistic authentication. Section 3 describes the framework for an automatic generation of other types of an extension so-called secrecy amplification protocols. The resulting protocols with their analysis and efficiency are presented in for two patterns of the partially compromised networks as well as comparison with protocols designed using conventional "human" design. This is followed by a conclusions in section 4 of this report.

Parts of this report originated in conference papers previously published in [CS05] (PUSH amplification protocol), [SO05] (early stage of group supported protocol without probabilistic pre-distribution), [SM07] (group supported protocol) and [Sve07] (basic framework for automatic protocol generation).

1.1 Node-compromise attacker model

Common attacker model in the network security area is the extension of the classic Needham-Schroeder[1] model [NS78] called the node-compromise model [EG02, CPS03, DDHV03, DDHV04], described by the following additional assumptions:

A1: The key pre-distribution site is trusted – Before deployment, nodes can be preloaded with secrets in a secure environment. Our work aims to omit this phase as is a cheaper to produce identical nodes (even at the memory level).

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

[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.

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.

The attacker model is in some cases (Key Infection [ACP04]) weakened by the following assumption:

A4: For the short interval the attacker is able to monitor only a fraction of a links – This assumption is valid only for a certain period of time after deployment and then reverts to stronger attacker with ability to eavesdrop all communication. The attacker can selectively eavesdrop only a fraction of links and the rational reason behind this assumption is based on specifics of WSNs:

a) Locality of eavesdropping – low communication range of nodes allows for the frequent channel reuse within the network and detection of extremely strong signal, so it is not possible for an attacker to place only one eavesdropping device with a highly sensitive and strong antenna.

b) Low attacker presence during deployment – low threat in most scenario during first few seconds before he realizes, that target area is in use. If the attacker's nodes are already presented in given amount in the target location, we can deploy network with density and node range such, that ratio between legal nodes and attacker's eavesdropping devices is such, that secure network can be formed.

Note that the attacker model for WSNs is stronger than the original Needham-Schroeder, because nodes are not assumed to be tamper resistant and attacker is able capture them and extract all carried sensitive information.

1.2 Cryptographic issues in network bootstrapping and main results

Security protocols for WSNs deal with very large networks of very simple nodes. Such networks are presumed to be deployed in large batches followed by a self-organising phase. The later is automatically and autonomously executed after a physical deployment of sensor nodes.

Deployment of a sensor network can be split into several phases. The following list is adjusted to discern important processes of key pre-distribution, distribution and key exchange protocols. The main phases are as follows:

1. Pre-deployment initialisation – performed in trusted environment.
2. Physical nodes deployment – random spreading (performed manually, from plane, ...) of sensors over a target area in one throw or in several smaller batches.
3. Neighbour discovery – sensors are trying to find their direct neighbours (nodes which can be directly reached with radio) and establish communication channels.
4. Key setup – key discovery or key exchange between direct neighbours.
5. Key update – periodic update of initial keys based on events like secrecy amplification, join to cluster, node revocation or new nodes redeployment.
6. Establishment of point-to-point keys – the final goal is always to transmit data securely from sensors to one of a few BS. Point-to-point keys are pairwise keys between sensors and BS (or distant sensors).
7. Message exchange – the production phase of the network.

Main issues in the area of the key distribution for WSNs can be summarised as follows:

Master key scheme – One of the simplest solutions for key establishment is to use one network-wide shared key. This approach has minimal storage requirements, unfortunately a compromise of even one single node in the network enables decryption of all traffic. The master key scheme has no resilience against node capture. As recognised in [CPS03], this approach is suitable only for a static network with tamper resistant nodes.

Full pairwise key scheme – A contrast to the master key scheme, where a unique pairwise key exists between each two nodes. As shown in [DDHV03], this scheme has got a perfect resilience against node capture. However, this approach is not scalable as each node needs to maintain $n - 1$ secrets, where n is the total number of nodes in the network.

Asymmetric cryptography – Usage of PKI for WSNs is often assumed as unacceptably expensive in terms of special hardware, energy consumption and processing power [PST+02, EG02, ACP04]. Usage of asymmetric cryptography can lead to energy exhaustion attacks by forcing the node to frequently perform expensive

signature verification or creation [ANL02]. Energy efficient TinyPK architecture based on elliptic curves is proposed in [WKC+04]. Maintaining fresh revocation list, its verification and attacks like collusion attack [Moo05] remains a concern.

Base station as trusted third party – Centralised approaches like the SPINS architecture [PST+02] use the BS as a trusted mediator when establishing pairwise key between two nodes. Each node initially share the unique key with BS. This approach is scalable and memory efficient, but has high communication overhead as each key agreement needs to contact the BS and this causes non-uniform energy consumption inside the network [OS06].

Probabilistic pre-distribution – Various variants of random pre-distribution were proposed to ensure that neighbours will share a common key only with a certain probability, but still high enough to keep whole network connected [EG02, CPS04, CPS03, DDHV03]. During the pre-deployment initialisation, keys for each node are randomly chosen and assigned from a large key pool without replacement. After deployment, nodes search in their key rings for shared key(s) and use it as a link key, if such key(s) exist. Variants based on threshold secret sharing provide a better resilience against the node capture.

Deployment knowledge pre-distribution – Efficiency of probabilistic pre-distribution can be improved if certain knowledge about the final node position or likely neighbours is known in advance. Ring keys selection processes based on node physical distribution allocation are proposed in [DDHV04, CYZ04, LN03b]. The nodes that have a higher probability to be neighbours have higher probability to share a common key.

Key infection approach – Unconventional approach that requires no pre-distribution is proposed in [ACP04]. The weakened attacker with limited ability to eavesdropping is assumed for a short period after the deployment. Initial exchange key exchange between neighbours is performed in plaintext and then the number of compromised keys is further decreased by secrecy amplification techniques [ACP04, KKLK05, CS05].

Impact of Sybil and collusion attack – Powerful attacks against known key pre-distribution protocols are presented in [NSSP04, Moo05]. In the Sybil attack, the attacker is able to insert many new bogus nodes equipped with secrets extracted

from the captured nodes. In the collusion attack, compromised nodes are sharing their secrets to highly increase the probability of establishing a link key with an uncompromised node. This attack shows that a global usage of secrets in a distributed environment with lack of physical control poses a serious threat, which is hard to protect against.

Our work relates mainly on problem of node capture resilience of key distribution protocols as all known suitable protocols are to some extent vulnerable against variants of such attack [NSSP04, Moo05, DDHV03]. We aim to show that substantial improvements in resilience against node capture can be achieved when a group of neighbouring nodes cooperates in additional protocol. In the first case, the group of neighbours creates large virtual keyring and thus boosts resiliency against the scenario where each node has only its own limited keyring available. Probabilistic pre-distribution is used as the underlaying key distribution method here. In the second case, the group is propagating fresh new keys and thus helps to potentially secure again some of the compromised links. Secrets that are valid only locally are introduced, serving as a protection against Sybil-like attacks. Here, we based our work on the novel concept of plaintext key exchange and additional secrecy amplification introduced in [ACP04] and extended in [KKLK05].

1.3 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 detect majority of the malicious nodes 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 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.

1.3.1 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 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. Even when the number of keys carried by each node is relatively small, probability that two nodes will share at least one key is surprisingly high (e.g., 60% probability to share at least one key if each node carries 100 keys and the initial key pool has 10000 keys). 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 [EG02] (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 as described in (1) above is evaluated in [CPS03]. 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 (2) 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).

[CPS03] 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 by [ACP04] together with q-EG is studied. The random pairwise scheme is also described (see 1.3.2).

[PMM03] 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 $HMAC(ID_A, K_{C_i B})$. 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 [CPS03] provides significantly better node-capture resilience than basic EG [EG02] until some threshold is reached.

1.3.2 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 require-

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

ments. 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 [Moo05]).

In [CPS03], a modification of the basic pairwise scheme (referred to as CPS scheme) is proposed. 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 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 in [Blo84]. 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 in [DDHV03] (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 1.3.1). 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 [HK04] revisit the basic random pre-distribution EG scheme (1.3.1), CPS scheme (1.3.2) and DDHV (1.3.2) 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 [HK04]. 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.

The hypercube pre-distribution based on multiple key spaces of Blom's polynomial is proposed in [LN03a]. 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 1.8.1 and 2.6 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 [CPS04].

Impact of node replication (Sybil) attack against EG, q-EG and Blom scheme is evaluated in [FKZZ05]. This paper 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 in [Moo05]. 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 6).

1.4 Limited neighbour knowledge schemes

Previous schemes presume that the probability that any two nodes will appear as physical neighbours is equal for any two nodes (network is *flat*). Yet in many practical scenarios, some probabilistic deployment knowledge about the node's final grounding position can be available *a priori*. An additional setup phase before random key pre-distribution is introduced, exploiting this knowledge.

[LN03b] presents two schemes that are using certain deployment knowledge. The CPS scheme is used (see 1.3.2), but pairwise keys are generated only for such nodes, which have high probability to be physical neighbours after the deployment, based on some probability density function of deployment error with respect to node's expected position (referred to as closest pairwise key scheme). An extended version using a pseudo-random function (PRF) for pairwise key computation is introduced. During deployment, pairwise key K_{uv} between nodes u and v is created as $K_{uv} = PRF_{K_v}(id_u)$, where K_v is the master key for node v. The node u (called slave node) carries the value K_{uv} directly, whereas node v (called master node) carries function PRF and its key value K_v. Thus node v is able to compute K_{uv} after deployment for any (possibly later deployed) node u.

The second scheme uses threshold key distribution protocol based on bivariate polynomials. Target deployment area is divided into sectors with a suitable size. Multiple key spaces are used, each one for a group of neighbour sectors. Pairwise keys are generated only for the nodes from close sectors inside a corresponding key space.

In [DDHV04] the target deployment area is divided into sectors, each corresponding to a group of nodes, which are to be deployed from the same position. For each group, there is a key pool constructed in such way that there are overlaps with key pools for neighbouring groups. Only groups with direct neighbouring sectors share some amount of keys.

Similar approach that divides the original single group of all nodes to a network into smaller subgroups is presented in [CYZ04]. Nodes that will share the same "mission task" and should be able to communicate with each other are placed in the same subgroup.

A key management scheme using attack probabilities is presented in [CPS05]. Nodes are divided into different subgroups based on the expected attack probability. Groups with higher probability are equipped with more keys drawn from a larger key pool

than nodes from a less risky group. If such attack probabilities are known in advance, substantial improvements of the node capture resilience can be obtained.

In [HMMH04] the assumption about a random node capture and resulting resilience of previously proposed schemes against such capture is pointed out as too weak. A selective node capture algorithm is presented and vulnerability of known schemes against such type of capture is examined. Vulnerability of known schemes against node fabrication attack is pointed out. Grid-group deployment scheme that should be more resistant to selective node capture and node fabrication is proposed. The target area is divided into multiple squares areas (zones) based on the expected nodes deployment pattern. A distinct key pool is generated for each zone, using the DDHV scheme (see 1.3.2) with fixed value of $\tau = 2$ (called I-Scheme). No more than λ rows from one key space can be distributed among nodes, thus the attacker can never reconstruct whole key space from captured nodes and cannot add new fabricated nodes with keys of unused rows from the given key space. Additionally, each node obtains a special pairwise key (eight in total) for a randomly chosen "bridge" node from each one neighbour zone (called E-scheme) during pre-deployment phase. Nodes are then deployed uniformly over the assigned zone. Key establishment inside the zone (I-scheme) follows the original DDHV scheme. Missing pairwise keys can be established using neighbours (from the current zone) or bridge node(s) using a multi-hop key establishment method.

All presented schemes use the same idea of dividing originally flat network into smaller subparts based on the pre-deployment knowledge about the nodes that are more likely to communicate (location, mission task) or require more or less resilience against the node capture based on attack probability (if some parts of the network are likely to have more compromised nodes). Degree of partitioning can substantially increase of connection probability, keeping node's key ring at the same size. Alternatively, size of the initial key pool can be increased keeping the connection probability same, resulting in an increased resilience against the node capture.

1.5 Seed-based pre-distribution

Seed-based pre-distribution is an extension of a given pre-distribution scheme, introduced by [PMM03]. 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 communi-

cation 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 keys carried by the node can be computed by other locally, without any additional communication except for retrieving the target node's ID.

1.5.1 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 in [HMMH04]. As the identification of all 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 [Mer78]. 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" $C_{Ai} = MAC_{K_i}(hash(0|N_A|N_B))$

[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 $poolSize$.

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.

1.6 Key infection

A modified attacker model, where an attacker is not able to eavesdrop all communication lines, is introduced in [ACP04]. Under the assumptions of this model, a plaintext key exchange protocol with no pre-distributed keys is described. An attacker is assumed to have (black) nodes with the same receiver quality as our (white) nodes have. After the deployment, physical neighbours exchange a link key value with no additional security. The transmission range is increased by small steps (referred to as *whispering*) until a neighbour can hear a key value (when its hardware is capable to do so). An attacker will compromise a link key when being able to record the key exchange transmission.

Two methods which should further decrease the fraction of compromised keys (generally called *secrecy amplification*) are described. These methods can be viewed as an extension of the basic plaintext exchange and are discussed in more details in section 1.8. Generally, secrecy amplification works better in more dense networks.

A variant of initial key exchange (denoted as COMODITY) without secrecy amplification is presented in [KKLK05]. The node A sends same key K_1 to nodes B and C

in plaintext. Then K_1 is used to secure the distribution of initial key material $E_{K_1}(B|K_2)$ and $E_{K_1}(C|K_3)$ between (A, B) and (A, C). Final key between (A, B) is constructed as $K_{12} = hash(K_3, hash(K_2, K_1))$. Formal security proof of the proposed scheme is presented in the paper. No estimate of the number of compromised link keys is given.

In the key infection approach, weakened attacker model is necessary for the first stage (plaintext key exchange) and in some cases also during secrecy amplification. Length of this interval, together with the resilience of used exchange and subsequent secrecy amplification, determines the cost for an attacker to successfully attack the network.

In section 3, we focus on an automatic generation of such protocols using Evolutionary Algorithms to generate candidate solutions and a network simulator to evaluate them. Such approach enables us to find personalized protocols that work well against a given attacker and her tactics, avoiding unnecessary messages and thus also significantly reducing the communication overhead and making Secrecy Amplification protocols more practical.

1.7 Key distribution and resulting compromise patterns

Key distribution schemes behave differently when the network is attacked and partially compromised. We will focus on two types of network compromise patterns:

1.7.1 Random compromise pattern

The first one is the *Random compromise pattern*, where the probability that a given link is compromised is almost independent of other links. Especially, whether a link to particular node is compromised should be almost independent of a compromise of another links from the same node. This compromise pattern may arise when a probabilistic pre-distribution and later variations are used and an attacker extracts keys from several randomly captured nodes.

Note that in the case of probabilistic pre-distribution, the compromise status of links from same node is still slightly correlated – if one link is compromised, other links from the same node may be established using same key(s) as the compromised one. This correlation quickly decreases with the size of key ring on each node and e.g., for 200 keys in ring is negligable. For links constructed from pre-distributed symmetric cryptography keys holds if the link $A \rightarrow B$ is compromised, then also $A \leftarrow B$ is compromised.

1.7.2 Key Infection pattern

Compromised networks resulting from Key Infection key distribution [ACP04] form the second inspected pattern. Here, link keys are exchanged in plaintext and an attacker can compromise them if the transmission can be recorded. The weakened attacker model assumes that an attacker is not able to eavesdrop all transmissions, yet has a limited number of restricted eavesdropping nodes in the field. The closer is the link transmission to the listening node and the longer the distance between link peers, the higher the probability of a compromise. Typically, if the eavesdropping node is close to the legal node, most of the links to the latter can be compromised. Note that there can be a difference between the compromise status of the link $A \rightarrow B$ and the link $A \leftarrow B$. This is another difference from the Random compromise pattern.

1.8 Extension protocols

Extension protocol is an additional process executed by the nodes in the network after the basic link key establishment. We will target such protocols that involve other nodes in the network (typically the direct neighbours) as a group of supporting entities to improve the security of the partially compromised network. Here, we provide short overview of selected extension protocols related to key distribution that are related to our work presented in section 2 and 3.

1.8.1 Hypercube pre-distribution

The extension protocol for probabilistic polynomial pre-distribution called hypercube scheme is presented in [LN03a]. 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 [DDHV03]. Then nodes with the same index within a given dimension (rows and columns in 2-dimensional case, shown on Figure 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

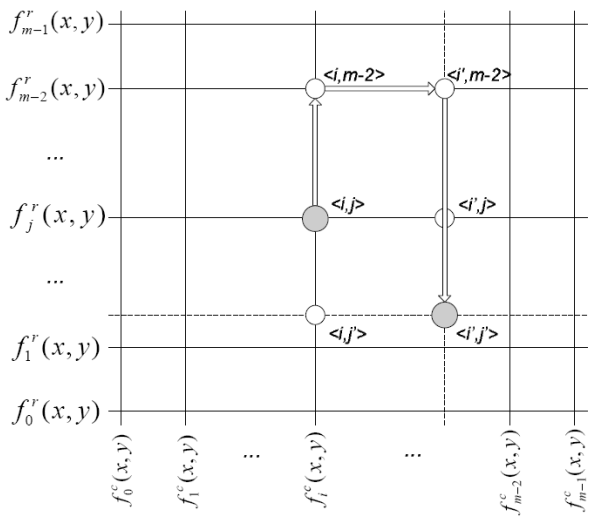


Figure 1: Polynomial assignment in two-dimension hypercube (grid). Nodes in the same row or column can establish key directly, otherwise with the help of neighbours in virtual grid. Figure taken from [LN03a].

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 – 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 2. Comparison with our group supported scheme (will be described later) is presented in section 2.6.

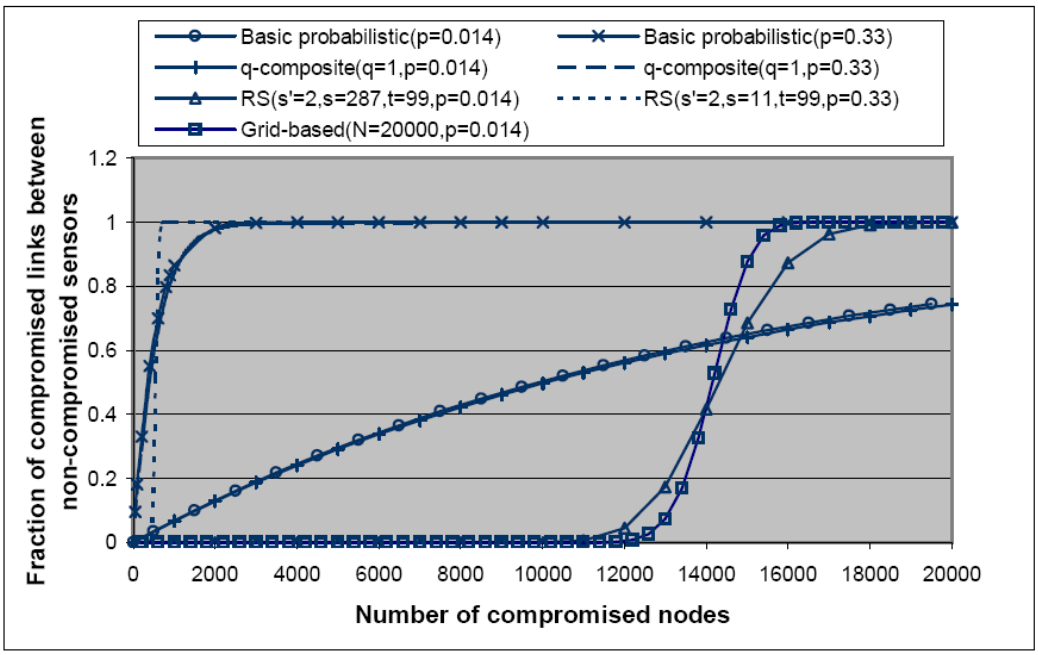


Figure 2: Node capture resilience of hypercube scheme, ring size equal to 200 keys. Figure taken from [LN03a].

1.8.2 Secrecy amplification

Secrecy amplification protocols were originally introduced for the Key Infection plaintext key exchange, but can be used also for partially compromised network resulting from a node capture.

In the *multi-path key establishment*, node A generates q different random values and sends each one along a different path over node(s) C_i to node B, encrypted with existing link keys (will be denoted as the PUSH protocol). All values combined together with the already existing key between A and B are used to create the new key value. An attacker must eavesdrop all paths to compromise the new key value. Second method, called *multi-hop key amplification*, is basically a 1-path version of the multi-path key establishment with more than 1 intermediate node C_i. Simulations in [ACP04] for attacker/legal nodes ratio up to 5% are presented, showing that the plaintext key exchange followed by the secrecy amplification is suitable within this attacker model.

A variant of initial key exchange (denoted as COMODITY) without secrecy amplification is presented in [KKLK05]. The node A sends same key K_1 to nodes B and C in plaintext. Then K_1 is used to secure distribution of initial key material $E_{K_1}(B|K_2)$ and $E_{K_1}(C|K_3)$ between (A, B) and (A, C). Final key between (A, B) is constructed as

$K_{12} = hash(K_3, hash(K_2, K_1))$. Formal security proof of the proposed scheme is presented. No estimate of the number of compromised link keys is given.

A variant of the PUSH protocol called PULL protocol is presented in our work [CS05]. Initial key exchange is same as for the PUSH protocol, but node C decides to help improving secrecy of the key between nodes A and B instead of node A as in the PUSH protocol. This decreases the area affected by the attacker eavesdropping node and thus increases the number of non-compromised link keys. Impact of key composition called mutual whispering on subsequent amplification is examined. Mutual whispering is key exchange where a pairwise key between A and B is constructed simply as $K_{12} = K_1 \oplus K_2$, where K_1 is the key whispered from A to B and K_2 from B to A. Experimental results show that mutual whispering followed by the PUSH protocol gives us equivalent fraction of secure links as basic whispering followed by the PULL protocol for Key Infection compromise pattern. Repeated iterations of the PULL protocols lead to strong majority of secure links even in the networks with up top 20% of attackers' eavesdropping nodes. Graphical representation of the PUSH and PULL protocol messages is shown on Figure 3.

Comparing PUSH, PUSH and COMODITY protocols, COMODITY requires the shortest period of weakened attacker model (transmission of only one key K_1), but K_1 can be intercepted from a larger distance than keys in PUSH and PULL protocols. This results in a higher probability of attacker interception. An additional problem exploitable by an attacker is such that an intermediate node W_2 knows the value of key between W_1 and W_3. PUSH protocol results in a significantly lower number of compromised link keys than PULL, especially for a dense enough network (more than 15 average neighbours).

An impact of PUSH, PULL, mutual whispering and new automatically found protocols (described in section 3) for Random compromise pattern and Key Infect compromise pattern are shown on Figure 16 and 17. The PULL protocol provides better results than PUSH protocol for the Key Infection pattern, but has no advantage in the Random pattern. Mutual whispering improves security in the Key Infection pattern, but no improvement is visible for the Random pattern. Combination of mutual whispering with the PUSH protocol gives the same results as the PULL protocol alone in the Key Infection pattern. See [CS05] for a more detailed comparison of protocols and the impact of repeated runs of amplification (not shown here).

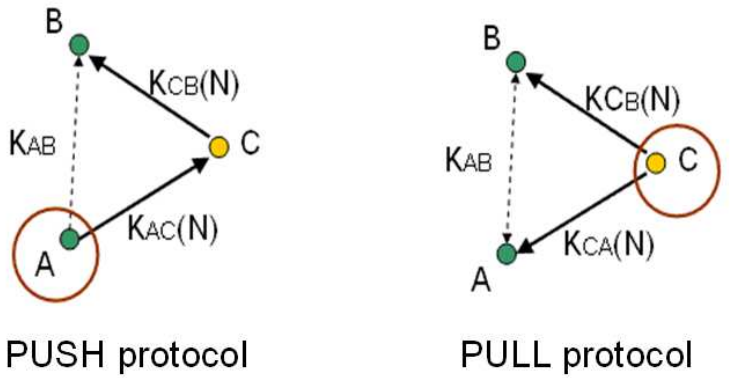


Figure 3: Graphic representation of the simplest version PUSH and PULL amplification protocols with only one intermediate node C. Red circle highlights the node generating fresh new random secret N. K_{xy} is the existing directional key between nodes x and y.

These results should demonstrate that amplification protocols may significantly increase the fraction of secure links and have a different impact for different compromise patterns. The protocols can be also combined together, but the impact of such composition is dependent on a particular compromise pattern and not necessarily beneficial. As each protocol requires significant number of messages, their inefficient combination should be avoided. Automated design of an efficient is proposed in section 3.

2 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).

2.1 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 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 $K'_{ij} = hash(K_{ID_{ij}}, R_{ij}, R_B)$. $(K'_{ij}, ID_{ij}, R_{ij})$ 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.

2.2 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:

[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.

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 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.

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

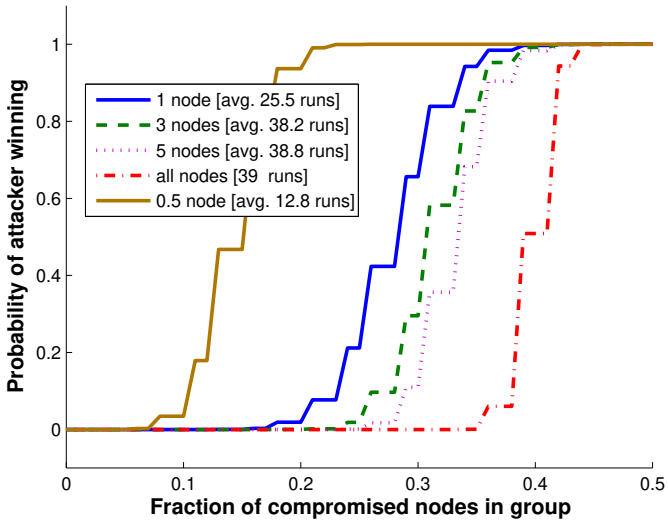


Figure 4: Probability of an attacker winning in majority decision voting described in 2.2.1 and number of expected runs of the protocol w.r.t. number of nodes selected as central by each member of a group (39 nodes per group).

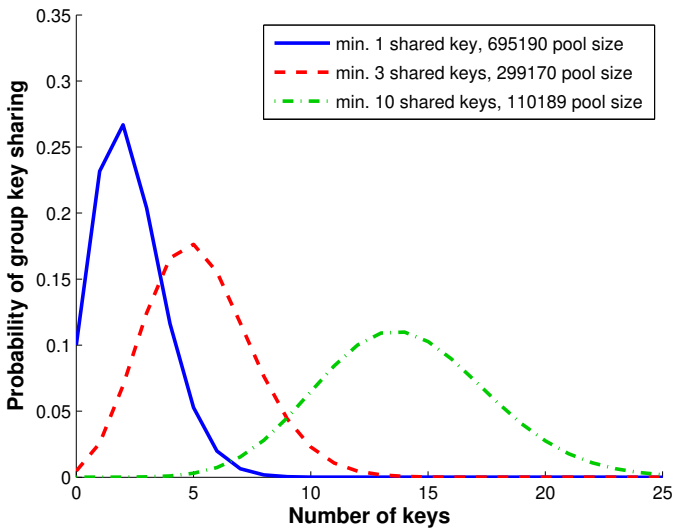


Figure 5: The distribution of probability of group key sharing (group size 40 nodes, ring size 200 keys, 90% constant probability of sharing at least required number of keys). Note that the pool size decreases with a growing minimum of required shared keys.

2.2.1 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. 4 for probability of attacker's success for different values of p and of the compromised fraction of a group.

2.2.2 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 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.

2.3 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 results were experimentally verified by series of simulations with 40000 nodes on our simulator described in section 3.3.2.

2.3.1 Evaluation of node capture resilience improvements

Lemma 1: Keys capture probability – EG.
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:

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

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

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:

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

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.

2.3.2 Options and settings

Node capture resilience can be evaluated for particular parameters of a sensor network according to the lemmas from 2.3.1. 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.

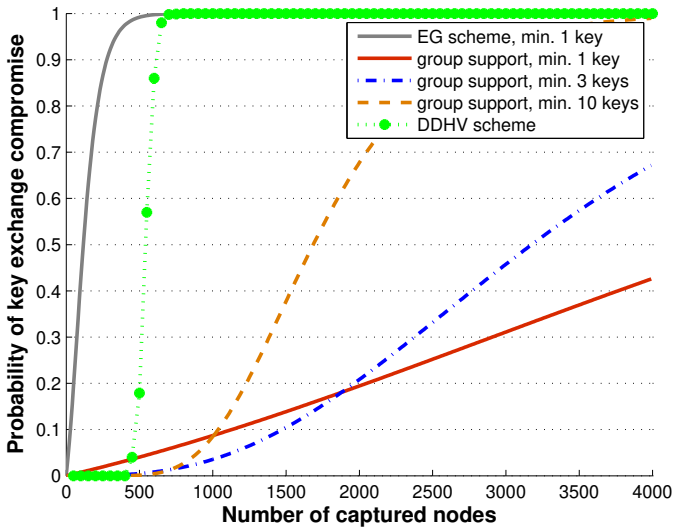


Figure 6: Node capture resilience of the basic EG scheme and the variant with group supported key exchange. Key ring size is fixed to 200 keys, group size to 40 nodes. The pool size differs with minimum of required shared keys to maintain a constant probability 90% that the exchange will be possible.

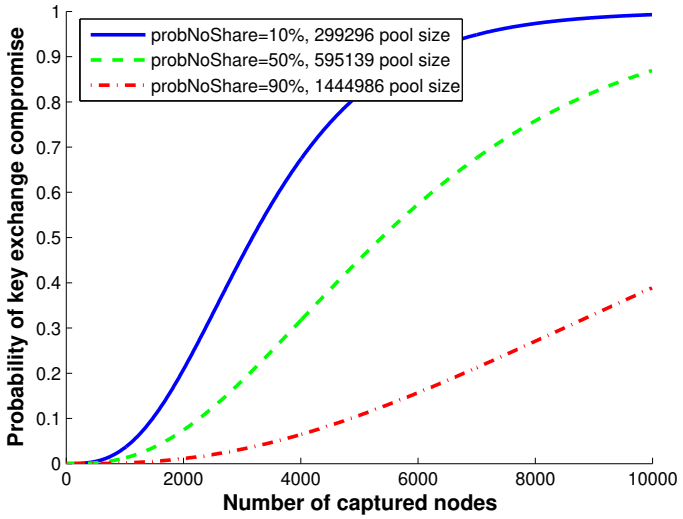


Figure 7: Impact of varying the probability of impossible key exchange to node capture resilience for minimum required keys T = 3. The network parameters are same as for Fig. 6. The group enlarged to 160 nodes will decrease respective probNoShare values to 0.0001% / 0.2% / 18.43%, keeping the pool size and NCR same.

We choose to fix this parameter to 200 keys to enable a comparison with the currently best-performing multi-space pairwise keys scheme [DDHV03] (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. 6, 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 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 $\mathtt{probNoShare} = 10\%$ 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 2.4 for details). Based on the usage scenario, $\mathtt{probNoShare}$ 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. 7.

The significant increase of NCR can be obtained when $\mathtt{probNoShare} = 90\%$ 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 2.4) can be performed.

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 1: Decrease of probability of impossible key exchange $\mathtt{probNoShare}$ 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.

We would like to stress out that our protocols do not provide defense against Sybil [NSSP04] or collusion [Moo05] attacks, where direct clones of the captured node are populated over the network. Here we rely on some replication detection algorithm like [PPG05]. Design of such efficient protocol with a reasonable communication overhead is still an open question.

2.4 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 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 $\mathtt{B}$.

The table 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. 7, 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.

2.5 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 [DDHV03]. Instead 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 poly-

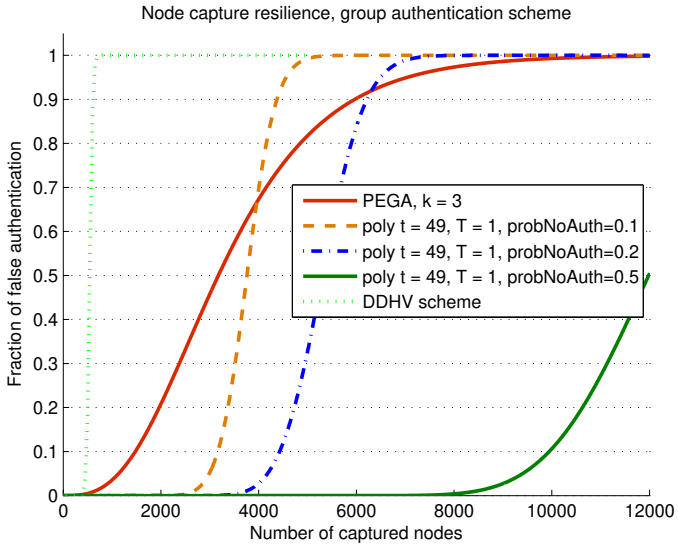


Figure 8: Node capture resilience of group supported scheme with Blom's treshold secret sharing scheme as an underlaying pre-distribution. Group size 40 nodes. Ring size is equal to 200 keys.

nomial 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.

2.5.1 Evaluation of node capture resilience improvements

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 [Blo84]), 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

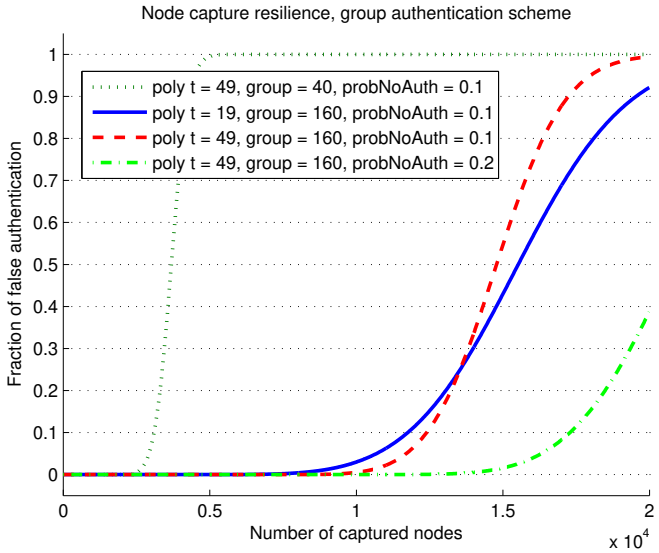


Figure 9: Node capture resilience of group supported scheme with Blom's treshold secret sharing scheme as an underlaying pre-distribution. Group size 160 nodes. Ring size is equal to 200 keys.

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.

2.5.2 Impact of polynomial security threshold

Impact of different degree of the polynomials is shown of Figure 10 (basic group with 40 neighbours) and Figure 11 (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

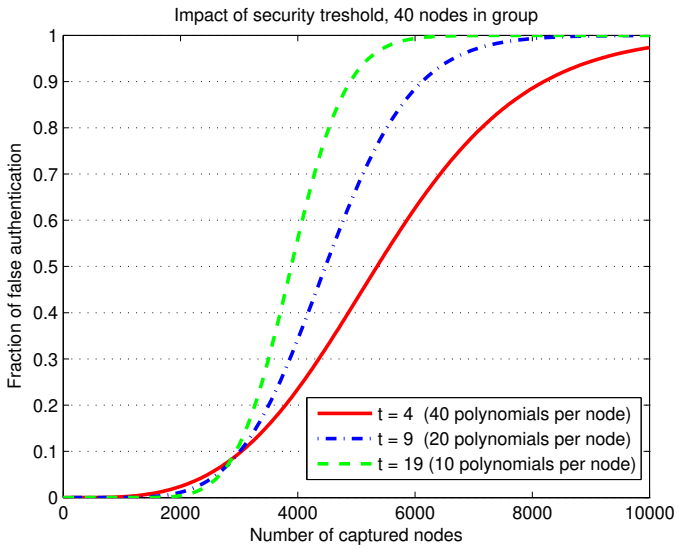


Figure 10: Impact of number of polynomial security threshold. Group size is 40 nodes. Ring size is equal to 200 keys.

restriction as $\mathtt{numberOfPolynomials} = \lfloor 200/c \rfloor$. 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.

2.5.3 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 threshold of minimal shared key spaces is checked. The results are significantly different from the EG scheme as shown on Figures 12 (basic group with 40 neighbours) and Figure 13 (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 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 to 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

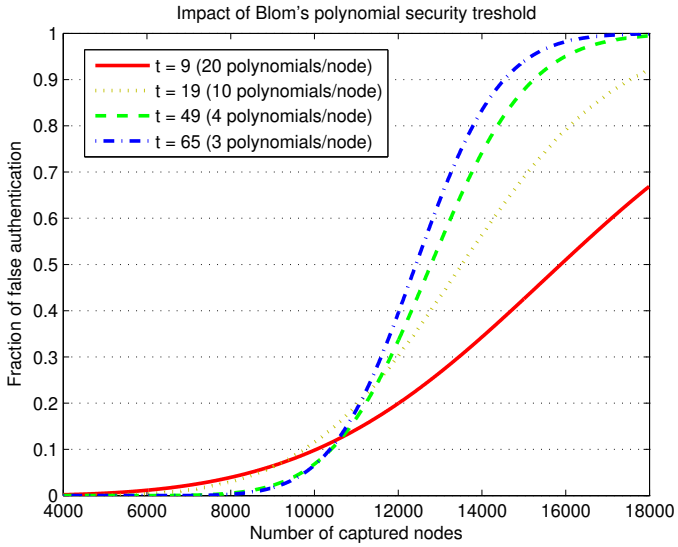


Figure 11: Impact of number of polynomial security threshold. Group size is 160 nodes. Ring size is equal to 200 keys.

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 12).

2.6 Comparison with hypercube scheme

Direct comparison between group supported and hypercube scheme 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

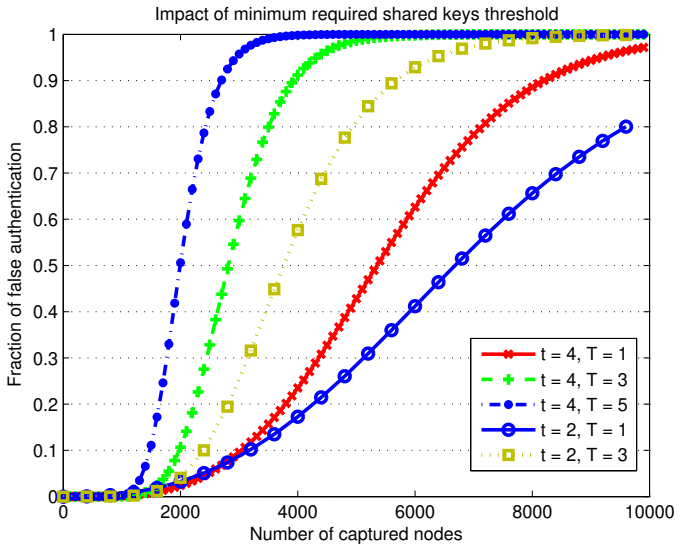


Figure 12: Impact of number of minimum required keys during the group support protocol. Group size is 40 nodes. Ring size is equal to 200 keys.

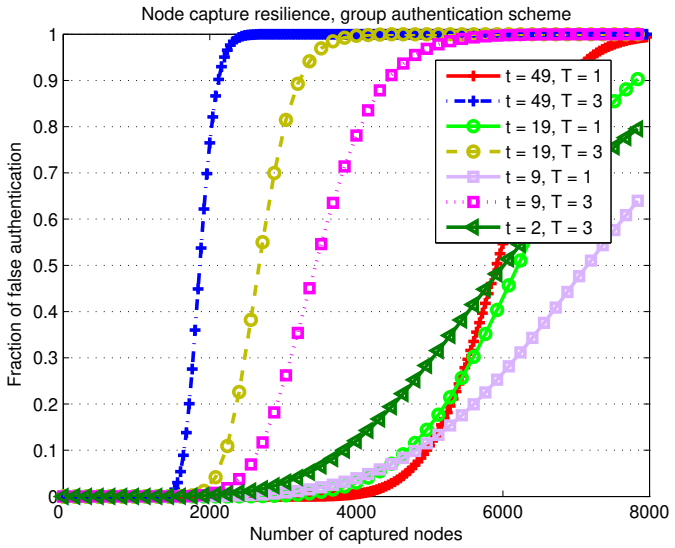


Figure 13: Impact of number of minimum required keys during the group support protocol. Group size is 160 nodes. Ring size is equal to 200 keys.

	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 2: Comparison of the properties of group supported and hypercube-based predistribution schemes.

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 and thus does not create a long communication paths. Due to seed-based pre-distribution, 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.

2.7 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.

2.7.1 Passive attacker

A passive attacker can either randomly or selectively [HMMH04] 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 1.5.1, 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 2.3.1, 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 1.5.1 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.

2.7.2 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.

2.8 Summary of the protocol

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 [EG02] and polynomial scheme by [DDHV03] 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 predistribution provides a better node capture resiliency if the polynomial scheme can be efficiently computed on the nodes. See [LN03a] for discussion of efficient implementation.

Future work will thoroughly examine other possible attacks against the proposed protocol, namely the deeper analysis of impact of compromised neighbours and selective node capture.

3 Localized secrets and secrecy amplification

The uncertainty about direct neighbours identity after the deployment prior to it naturally results in such property of the key distribution schemes that most of the nodes in the network should be able to establish shared key (either directly or with support from other nodes). At the same time, this property implies one of the main problems for maintaining secure network in presence of the adversary with ability to capture nodes. As the extracted secrets can be commonly used in any part of the network, various Sybil and replication attacks can be mounted (e.g., to join an existing cluster with a captured node clone). Moreover, even when the compromised node is detected and its keys are revoked, revocation list must be maintained for the whole network (e.g., if the revocation list is maintained in a completely decentralized way then ultimately every node must store the copy of the list). A common way and good practice to introduce localized secrets is not to use pre-distributed keys for ordinary communication, but only to secure key exchange of fresh new keys, which are locally generated only by nodes involved in exchange. If the usage of the pre-distributed keys is allowed only for a certain time (or treated with more scrutiny later), such practice can limit the impact of the node capture as the localized keys have no validity outside the area of their generation. An attacker is forced to mount his attack only during a limited time interval and it is thus reasonable to expect that the number of compromised nodes will be lower. When such localized secrets are established with the support from other (potentially compromised) nodes,

secrecy of the resulting key can be violated. To tackle this threat, secrecy amplification protocols were proposed.

3.1 Secrecy amplification protocols

Secrecy amplification protocol (will be denoted as SA; also known as privacy amplification protocol) describes an additional process executed by nodes within a network after the basic link key establishment. Fresh new secrets are locally generated and distributed using existing links with associated security. As a result new link keys are constructed. These are different from pre-shared secrets. Especially in WSNs, secrets usable only locally should be preferred due to possibility for various Sybil-like attacks (spread of malicious clones of captured legal nodes). Moreover, some links can be secured, even when the original link was compromised as was already discussed in 1.8.

The protocols described in section 1.8.2 can be extended both in the number of used distinct paths (multi-path key amplification) and their lengths (multi-hop key amplification). In the multi-path amplification, several different random values are sent along different paths to A and to B, encrypted with existing link keys (the original PUSH and PULL are 2-path versions, the mutual whispering is also 2-path, where these paths overlap). All values combined together with the already existing key between A and B are used to create a new key value. An attacker must eavesdrops all paths to compromise the new key value. Multi-hop key amplification involves several intermediate nodes C_i between A and B (mutual whispering is 1-hop (no intermediate node), the PUSH and PULL are 2-hop).

What is commonly unknown to the nodes in the network is the fact which links are actually compromised. Still, we can execute the amplification protocol "just to be sure", even when the link between A and B is secure against the attacker (but we do not know that). If we create a new link key as $K'_{AB} = f(K_{AB}, K')$, where K_{AB} is the original link key and f is cryptographically strong one-way function, we will obtain a secure link if either the original link is already secure or K' can be securely transported to both A and B over some existing path. Such process poses a significant communication overhead as number of such paths is significant, but may also significantly improve the overall network security.

Eventually, more iterations of the amplification protocol can be performed. Link keys security can be further improved as links newly secured in the previous iteration can help to secure a new link in the next one.

There is no difference between passive and active attackers for amplification protocol. An active attacker controlling the node is equivalent to a passive one that compromised all links to the node and thus all passing messages are intercepted. A denial-of-service attack can be mounted if intermediate nodes propagate incorrect values, but will be detected after the construction of a new link key, because two non-compromised nodes will not be able to establish a functional key. By gradually removing keys used in the construction, they can spot the node or path which contributed the defective key and ignore it for later protocol runs. The opposite attack must be considered as well as two compromised nodes may blame legal node for providing the incorrect key. The jammed link is equivalent to a missing connection.

3.2 Automatic protocol design

In this report, we propose a way how new SA protocols can be automatically generated for an arbitrary compromise pattern with a minimal effort from a human designer. Evolutionary Algorithms (EAs) are used to construct new protocols. Candidate protocols are evaluated using a network simulator.

Designing new protocols is a time consuming process and present flaws may remain undetected for a long time. Various formal verification tools currently exist to verify the correctness of a proposed protocol (see [Mea03] for an extended review). Automatic protocol generation (APG) was proposed to automatically generate new protocols with desired properties using a brute-force space search and with their correctness verified by formal tools [SBP01]. Unfortunately, there are still limits due to the rapid increase of possible configurations of non-trivial protocols.

However, the formal verification approach can be avoided for APG if a new protocol can be securely composed from simpler (secure) protocols. See [Cho06] for an extended overview of possible approaches to automatic protocol generation and protocol composition. Fortunately, this is the case of SA protocols – SA protocol specifies the way how fresh key values are propagated and combined by parties involved. Thus an SA protocol can be viewed as a composition of few simpler protocols. Namely, we need only a protocol for secure message exchange between two nodes sharing a secret key and a secure composition of two or more values.

This is an important difference to former approaches to APG – as the composition of selected secure protocols will be also secure (see [Cho06] for such protocols; note that composition is not secure in general), we can skip the formal verification of the compos-

ite all together. Instead, we have to verify how many keys from freshly generated secrets will be compromised by an attacker after a SA protocol execution. An attacker is able to eavesdrop the content of some SA messages as he knows some of the keys used (partially compromised network). This is a deterministic process – if we know exactly which keys are known to the attacker – and thus can be simulated. Even if we know only the expected fraction of compromised keys and the average pattern of compromised links, we can perform a probabilistic evaluation. As the number of nodes and links in WSNs is expected to be high, such average case will be a reasonable approximation of secure links after SA execution in a real network.

Although the space of potential protocols is still large, we can substitute a formal verification tool by a network simulator with faster evaluation. Moreover, we will obtain a smoother indication how good a candidate protocol is. Instead of binary indication "secure or flawed", we will get the number of links additionally secured by a particular protocol[7]. Hence we can use some kind of informed search instead of an exhaustive search. In our case, we will use Evolutionary Algorithms.

3.3 Evolutionary Algorithms intro

Evolutionary Algorithms (EAs) are stochastic search algorithms inspired by Darwin's theory of evolution. Instead of working with one solution at a time (as random search, hill climbing and other search techniques), these algorithms operate with the *population* of *candidate solutions* (candidate SA protocol in our case). Every new population is formed by genetically inspired operators such as *crossover* (part of protocol's instruction are taken from one parent, rest from another one) and *mutation* (change of instruction type or one of its parameter(s)) and through a selection pressure, which guides the evolution towards better areas of the search space. The Evolutionary Algorithms receive this guidance by evaluating every candidate solution to define its *fitness value*. The fitness value (fraction of secure links here), calculated by the *fitness function* (network simulator here), indicates how well the solution fulfills the problem objective (improving network security here). In addition to the classical optimization, EAs have been utilized to create engineering designs in the recent decade. For example, computer programs, electronic circuits, antennas or optical systems are designed by *genetic programming* [KIAK99]. In contrast to the conventional design, the evolutionary method is based on the generate&test approach that modifies properties of the target design in order to obtain the

[7] In degenerated case, this can still be only "0% or 100%" links secure.

required behavior. The most promising outcome of this approach is that an artificial evolution can produce intrinsic designs that lie outside the scope of conventional methods. In this work, we will use linear genetic programming (LGP) to generate the protocols. LGP represents a candidate program as a sequence of instructions [BNKF98].

3.3.1 Primitive instructions set

Each party (sensor node) in the protocol is modeled as a computing unit with a limited number of memory slots, where all local information is stored. The memory slot can be loaded with a) random value, b) encryption key and c) message. The following primitive instructions are defined, having one or two parameters N_x indicating the node(s) that will execute a given instruction (e.g., local generation of a random value will have only one node parameter; sending a message between nodes will have two parameters) and up to three parameters R_x for the identification of used memory slots. These instructions were selected such that allow us to describe all published SA protocols and are using only (cryptographic) operations available on real nodes. A candidate SA protocol is represented as an array of bytes.

The instructions set is as follows:

- NOP – No operation is performed.
- RNG N_a R_i – Generate a random value and store the result in the slot R_i.
- CMB N_a R_i R_j R_k – Combine values from slots R_i and R_j and store results in the slot R_k. The combination function may vary on the application needs (e.g., cryptographic hash function such as SHA-1).
- SND N_a N_b R_i R_j – Send a value from node N_a to N_b. The message is taken from N_a's slot R_i and stored in N_b's slot R_j.
- ENC N_a R_i R_j R_k – Encrypt a value from the slot R_i using the key from slot R_j and store encrypted result in slot the R_k.
- DEC N_a R_i R_j R_k – Decrypt a value from the slot R_i using the key from the slot R_j and store decrypted result in the slot R_k.

Each instruction has an additional boolean switch, which can turn the operation off (equivalent of NOP), without changing the instruction itself. This allows the EA to temporarily disable or enable a single instruction. Node identifications N_a and N_b can

be either fixed (the index) in case of node-oriented protocols or distance-relative in a group-oriented protocol. These variants are discussed later in sections 3.4 and 3.5.

Using this set of primitive instructions, a simple plaintext key exchange can be written as {RNG N_1 R_1; SND N_1 N_2 R_1 R_1;}, a PUSH protocol as {RNG N_1 R_1; SND N_1 N_3 R_1 R_1; SND N_3 N_2 R_1 R_1;}, a PULL protocol as {RNG N_3 R_1; SND N_3 N_1 R_1 R_1; SND N_3 N_2 R_1 R_1;} and a multi-hop version of PULL as {RNG N_3 R_1; SND N_3 N_1 R_1 R_1; SND N_3 N_4 R_1 R_1; SND N_4 N_2 R_1 R_1;}.

Note that the protocol space is extremely large even for small protocols with six instructions for four nodes, each with six memory slots only, having more than 10^{21} possible configurations[8].

3.3.2 Network simulator

Candidate protocols are evaluated using our own simulator that was developed specifically for security analyses of the key distribution protocols and message routing. The simulator is capable of performing:

- Random or patterned deployment of a network with up to 10^5 nodes together with neighbours discovery, secure links establishment and simple routing of messages.
- Deployment of attacker's nodes and their eavesdropping impact on the network.
- Evaluation of the number of secure links of published protocols for secrecy amplification of "Key Infection" approach (see [ACP04] for details).
- Evaluation of the number of secure links of probabilistic key pre-distribution protocols as described in [CPS04].
- Support for the execution of amplification protocols in the metalanguage using primitive instructions (see section 3.3.1).
- Support for Evolutionary Algorithms employed in an automatic generation of protocols in the metalanguage and consequent simulation(s) with the fraction of secure links as a fitness value. Implementation of the EA is based on the GALib package[9].

[8] $(6 \times 4 \times 6 \times 6 \times 6)^6$

[9] GALib - C++ Genetic Algorithms Library

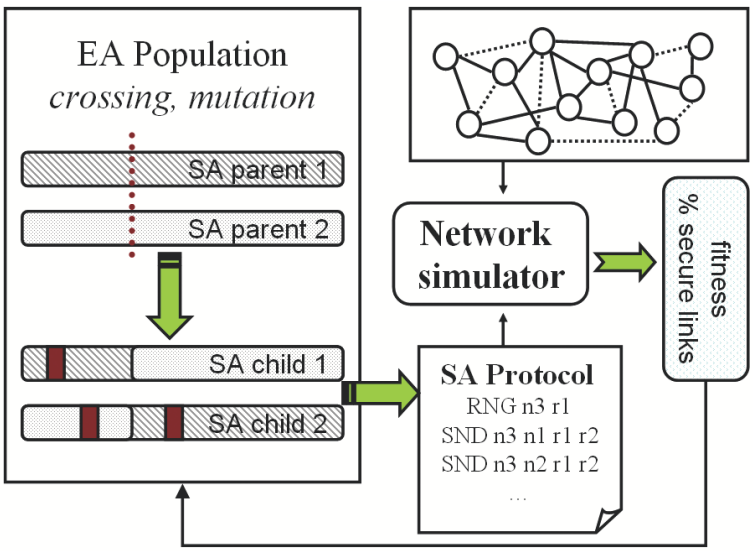


Figure 14: Automatic protocol generation process with fitness evaluation. The new population of genotypes is transcribed into candidate protocols. Using our network simulator and a given partially compromised network (dotted links), the fitness value (fraction of secured links) is calculated for each candidate protocol.

3.4 Evolution of node-oriented protocols

In this part, we focus on an automatic generation of amplification protocol for the fixed number of k parties, i.e. the same scenario as used in [ACP04, CS05]. Such protocol is executed for all possible k-tuples of neighbours in the network. Note that the number of such k-tuples can be high[10], especially for dense networks (e.g. more than 10 direct neighbours) and resulting communication overhead is then significant. However, this approach provides an upper bound on the success rate of a given protocol as no k-tuple is omitted.

Initially, we generate few protocols with several hundred randomly selected primitive instructions. These candidate protocols form the initial population for the EA. Every protocol is then simulated and the number of secured links serves as a fitness value. Protocols with the best fitness value are selected to serve as parents for the next generation which is created by applying crossover and mutation operators. New population is thus consisting of better parts of the previous one (an offspring protocol combined from two good parents will be possibly better than parents). Also, new "ideas" are

[10] E.g., $(total_nodes * avg_neigh) * (avg_neigh - 1) * msg_per_protocol_execution$ for a three-party protocol, where avg_neigh is the average number of neighbours.

introduced by rare mutations. Commonly, first generations are not able to secure any additional link, but as evolution proceeds, there are more and more secured links. The evolution can be stopped when a sufficiently good protocol is found or the best fitness value has stagnated for some time.

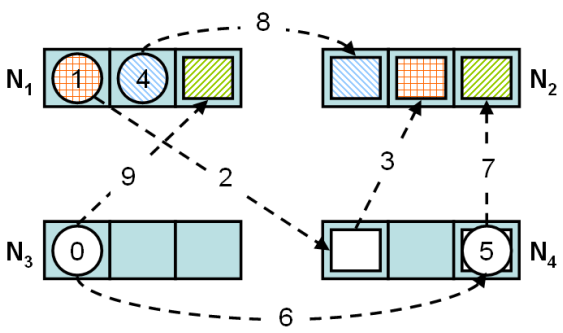
The fitness landscape[11] for node-oriented protocols seems to have a "cascade-like" form. There are only a few significant fitness values in the search space. A small population with the rapid mutation is suitable for solving such problems (Cartesian Genetic Programming (CGP) [MS06]). Candidate protocols contain up to 200 primitive instructions. Each party has 8 memory slots for storing intermediate values. The population size is 5. The mutation operator is applied with the probability 10%. Similarly to the CGP, crossover is not used. Simulations are performed for three distinct network deployments (the average fraction of secured links is used as the resulting fitness value), each with 100 legal and 20 eavesdropping nodes having 9 white neighbours on average. The protocol success rate was verified using larger networks with 2500 white nodes at the end of evolution.

The best performing 4-party protocol discovered by EA was found within 4 days on a 3GHz processor in the 62786th generation. The protocol consists of the instructions shown on Figure 15. This is a "pruned" version of the original 200-instructions long protocol found by evolution. Importance of each instruction was tested by its temporal disabling – if the instruction is important, then the fitness decreases and the instruction is preserved; otherwise, it is discarded from the protocol. Typically, only 5-10% instructions contribute to the fitness value (i.e., there is analogy to exons and junk DNA in the human genome).

This protocol can be further post-processed – only three memory slots are actually required on each node instead of eight that was given to use for evolution.

All amplification protocols we are aware of at the start of our work were rediscovered here by EA. The simple key transfer between neighbours is encoded in steps {4,8}. The PUSH protocol by [ACP04] is encoded in steps {1,2,3}. The PULL protocol by [CS05] is encoded in steps {0,6,9}. The multi-hop version [CPS04] of PULL amplification is encoded in steps {0,6,7,9}. Moreover, the new protocol outperforms existing amplification protocols, as shown on Figures 16 and 17.

[11] The fitness values for each possible instance in the search space. We cannot compute whole landscape in a reasonable time – if we can, then there is no need for any EA – we can obtain a fitness maximum directly.



#	instructions
0	RNG N_3 R_1
1	RNG $N_1 R_1$
2	SND $N_1 N_4 R_1 R_2$
3	SND $N_4 N_2 R_2 R_2$
4	RNG $N_1 R_3$
5	RNG $N_4 R_4$
6	SND $N_3 N_4 R_1 R_4$
7	SND $N_4 N_2 R_4 R_3$
8	SND $N_1 N_2 R_3 R_8$
9	SND $N_3 N_1 R_1 R_6$

Figure 15: Evolved node-oriented 4-party secrecy amplification protocol. This is a pruned version from a 200 instruction protocol, no other post-processing was applied. A circle denotes RNG instruction, an arrow denotes SND instruction and a box a transmitted value. Values shared between N_1 and N_2 are the same color.

The evolved protocol also exhibits an interesting feature of "polymorphic" instruction. At the first look, instruction 5 (RNG N_4 R_4) seems to be redundant as a newly generated random value by node N_4 stored in the slot R_4 is immediately overwritten by the instruction 6 (SND N_3 N_4 R_1 R_4). However, in the case when node N_3 is not a direct neighbour of node N_4, the message in instruction 6 cannot be transmitted and R_4 is not overwritten. The exact behavior of the consequent instruction 7 will vary as R_4 can be filled either with a newly generated random value or the value received from node N_3. Such kind of "polymorphic" instructions enable protocol execution even when only a limited number of nodes is reachable. However, it would be hard for a human designer to propose such protocol as a dependency between the instructions and neighbour layout is rather complex, especially for group-oriented protocols (discussed later).

Note that automatic design of the node-oriented protocols with 5+ parties was not possible as the number of messages to be simulated grows exponentially with the number of parties involved. The simulator is not able to evaluate such protocols fast enough to obtain a fitness value and prevents evolution to proceeds towards better solution in reasonable time.

An interesting result is that despite the fact that encryption (ENC) and decryption (DEC) is included in the set of primitive instructions, none of them was used in the evolved protocols. There can be multiple reasons for this: At first, useful usage of the ENC and DEC instruction may exist, but the evolution was not able to find it. Second, more probable reason can arise from the applied evolution speed-up trick used during

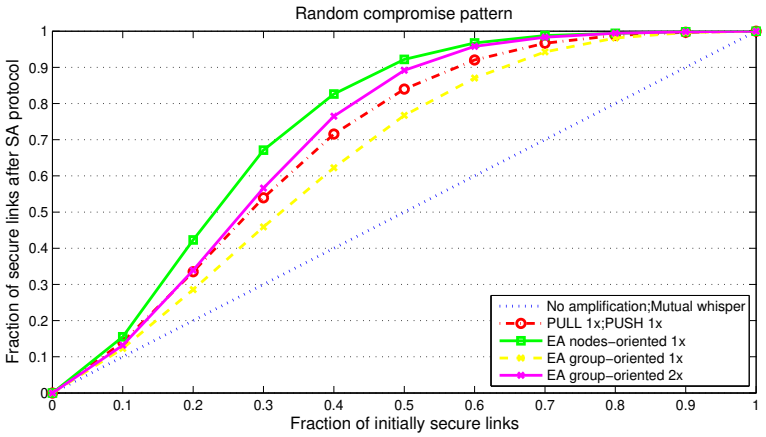


Figure 16: Increase in the number of secured links after SA protocols in the Random compromise pattern. The PUSH and PULL protocols give same results, mutual whispering does not improve security at all.

the evaluation of the candidate protocol. If the link already has some assigned key, this key is transparently used for encryption, as it is obviously useful thing to do (if the key is compromised we will obtain the same result as sending message un-encrypted, but if the key is secure then message secrecy will be protected). Series of the evolutions was performed for the case when the transparent link encryption was not used. Evolution was significantly slower to achieve the same fraction of secured links, but essentially developed link encryption by existing keys anyway.

3.5 Evolution of group-oriented protocols

As we have already mentioned, node-oriented protocols introduce a high communication overhead – all k-tuples of neighbours must be executed by such protocol. Another issue is an unknown number of direct neighbours and their exact placement. All neighbours can theoretically participate in the protocol and help to improve the fraction of secure links, but it is much harder to design an efficient protocol for ten nodes instead of three or four nodes. And finally, due to the random placement of nodes in the sensor networks, the number of direct neighbours may vary significantly and protocol for the fixed number of parties can even fail to have enough participants. Due to the broadcast nature of the wireless transmission, nodes' geographic position also influences the result of amplification.

We present a different approach to the design of amplification protocols. Identification of the parties in protocol is no more "absolute" (e.g., node number "1"), but

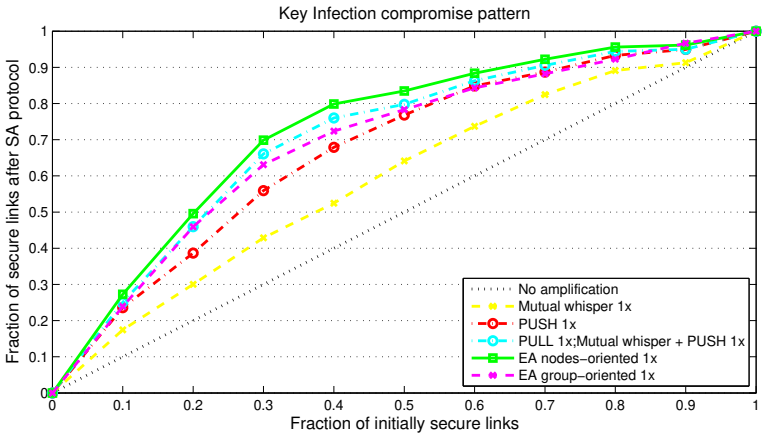


Figure 17: Key Infection compromise pattern. Average 8.0 white neighbours. The PULL protocol provides better results than the PUSH protocol here and combination of mutual whispering with the PUSH protocol gives the same results as PULL alone.

given by the relative distance from other parties (we will use distance from two distinct nodes). We assume that each node knows the distance to its direct neighbours. This distance can be approximated from the minimal transmission power needed to communicate with a given neighbour. If the protocol has to express the fact that two nodes N_i and N_j are exchanging a message over the intermediate node N_k, only relative distances of such node N_k from N_i and N_j are indicated in the protocol (e.g., $N_{(0.3_0.7)}$ is a node positioned 0.3 of the maximum transmission range from N_i and 0.7 from N_j; EAs are searching for distance values.) Based on the actual distribution of the neighbours, the node closest to the indicated distance(s) is chosen as the node N_k. There is no need to re-execute the protocol for all k-tuples as the protocol can utilize all neighbours in a single execution and thus significantly reduce the communication overhead. The relative position of nodes can be expressed as well. The variation in an actual number of direct neighbours poses no problem here – the protocol parties will always be found (but its actual position may be more or less dissimilar from relative distances indicated in the protocol).

The group protocol evaluation process is more complex than for the node-oriented protocols, but the number of totally exchanged messages is significantly lower.

The evaluation is based on the following rules:

1. Every node in the network is processed separately and independently once in the role of a central node N_C for each amplification iteration. Only direct neighbours of N_C (group) are involved in the protocol execution.

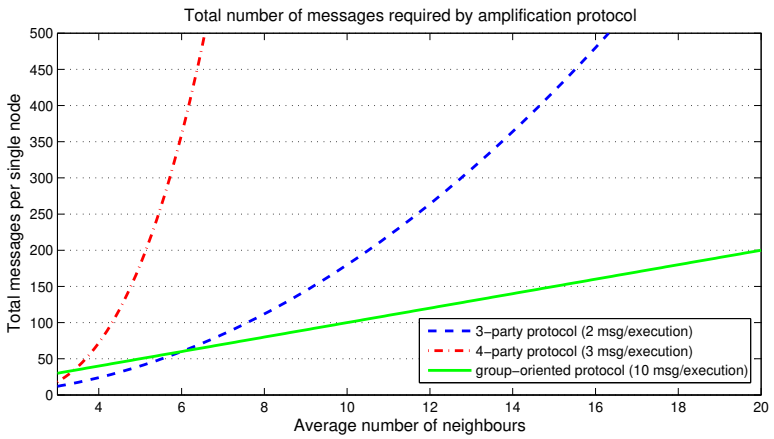


Figure 18: Total number of messages per single node required to perform a 3-party, 4-party and group-oriented secrecy amplification protocol.

2. A separate protocol execution is performed once for each direct neighbour that will have a special role and will be denoted as N_P (e.g., if there are 10 direct neighbours around N_C, then there will be only 10 protocol executions with the same central node N_C, each one with a different N_P). This is a key difference from nodes-oriented protocols, and thus saves a considerable communication overhead.

3. The memory slots of the neighbours involved (for the same N_C) are *not* cleared between the protocol executions. This enables the evolution to find a protocol that propagates values among a group of neighbours.

4. The node N_P provides a list of distances from all its neighbours (as the minimal transmission power needed to communicate with a given neighbour) to node N_C. Based on the actual deployment of nodes in the group, parties from the protocol are replaced by real identification of the nodes closest to the relative identification from N_C and N_P.

5. When the next node is executed as a central node N_C, the nodes memories are cleared (memory values cannot propagate between executions with a different central node N_C).

Note, that the spared messages come from the change of the secrecy amplification rules, not the Evolutionary Algorithms itself. The role of the Evolutionary Algorithm is to find a protocol that will operate in the described restricted scenario and still be able to perform comparably to the node-oriented protocols in terms of secure links. Figure

18 compares the number of necessary messages for the three/four-party node-oriented protocol and the group-oriented protocol constructed using the above described process.

For the purpose of evolution speedup, we introduced the automatic actions that do not have to be evolved as they are obviously beneficial:

- Each message transmission (SND instruction) is transparently encrypted using existing link key (which can be either secure or compromised by eavesdropping) even when not stated explicitly in the protocol.
- The shared values later used for the creation of new link keys are automatically found in memory slots of N_C and its neighbours N_x at the end of each execution for a fixed node N_C. Again, this speeds up the search. In the actual execution of the protocol, this can be achieved efficiently using Bloom filters [Blo70] without a transmission of the values or better by the post-processing of an evolved protocol (re-order of memory slots and additional CMB instructions).

As for node-oriented protocols, more iterations (amplification repeats) can be executed. For the purpose of evaluation, the results within one iteration are independent and may influent only the next iteration, not the current one (links secured during an actual iteration will not help to secure other links during the same iteration). At the end of each iteration, the link security status is evaluated and updated. Evaluation process is thus not dependent on the processing nodes order inside the simulator.

Well performing group-oriented protocols with the fraction of secure links comparable to node-oriented protocols are usually evolved in 10^5 generations (see Figures 16 and 17 for the performance of evolved protocols). Such a protocol has typically 10-15 important instructions and uses neighbours from 5-7 different areas. The SND instruction is the most common one, forming 60-80% of instructions of discovered protocols. Example of such evolved protocol is presented on Figure 19 and its analysis in section 3.6. Around 80% of EA runs are able to provide protocols with almost the same success ratio that differ in their instruction order (remaining runs stay in some sub-optimal local maximum). So there is not only one "best" protocol – instead, most of EA runs provide some useful amplification protocols. This may serve as a kind of obfuscation – a "new" protocol can be evolved for each new network. In contrary to node-oriented protocols, instructions of the evolved protocols are more difficult to understand as the parties are not directly specified any more.

3.5.1 Methods for analysis of evolved protocols

Various techniques like real-time visualization of message transmission, analysis of memory store/load sequences or visualization of probable areas of relatively identified parties (see Figure 20) can be used to recognize actual purpose and importance of the instructions.

Real-time visualization of message transmission – Initial protocol overview can be obtained by observing the visual representation of the protocol execution with a set of nodes with fixed position. One limitation of this approach is that only execution for a given distribution of nodes is obtained and the behaviour of instructions for different nodes distribution can be overlooked.

Instructions cross-dependency using pruning-like process – Fitness reduction effect can be studied to identify groups of instructions with cross-dependent fitness. As the protocol has already been pruned, removing every single instruction I will cause a fitness decrease. An additional pruning process over the reduced protocol (without I) gives us the difference in the fitness gain for every of remaining instruction (some instructions may be completely be removed if they have no function without I). The higher the decrease in the fitness gain by a particular instruction J, the stronger the dependency of J on removed I.

Analysis of memory store/load sequences – As described in section listing primitive instructions, each party has a limited number of memory slots that are used to store intermediate values. A chain of memory slots connected by the edges representing a particular instruction can be established for graph-like visualization of process. More precisely, if there is an instruction I that reads from a memory slot M_i and writes in a memory slot M_j, we can connect state M_i with M_j in graph by an edge with label I.

Probable areas for position of parties identified by the relative distance – Visualization of areas, where nodes referenced in the protocol will most probably positioned is an important source of information how the protocol works. Note that these areas are not static for all nodes N_P, but differ significantly with the distance between central node N_C and its special partner for protocol N_P. A change of the position and the shape of areas with distance between N_C and N_P also reveals information how the fresh key values are propagated in the group. Using this

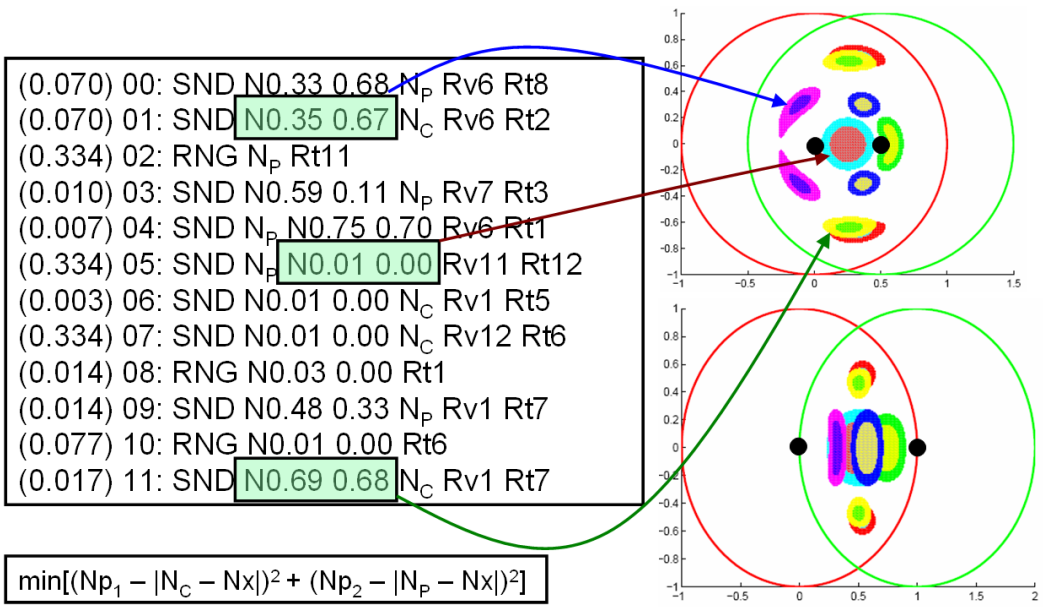


Figure 19: Example group oriented secrecy amplification protocol found by the evolution. Selected node-relative identification of involved parties are displayed as the geographically most probable areas, where such nodes will be positioned.

technique, we can conclude that instruction 3 sends a value stored by the instruction 9 in the previous run of the protocol when N_P was around 0.6 of the maximum transmission range far from N_C and in the layout area B. The reason is that in this distance, the layout area B overlaps with position of the node N_P and these two parties in the protocol are most probably mapped to the same node.

Note that removal of a single instruction I can not only decrease the fitness value, but also a fitness gain to other instruction(s) J. There are two reasons for this behavior: 1) Instruction I was really harming the fitness gain from instruction J, but the caused harm is lower than the fitness gain and thus I remains in the pruned protocol. 2) Instruction J is able to compensate (at least partially) the loss caused by I's removal and it is able to secure some links originally secured by the instruction I. Analysis of separate instructions shows that the second case is much more common. Evolved protocol thus exhibits "defense in depth" property – when some instructions cannot be executed (due to missing, unreachable or compromised party), other instructions are able to (partially) compensate for the decrease in the number of secured links. Similar behavior was also observed for the evolved nodes-oriented protocol.

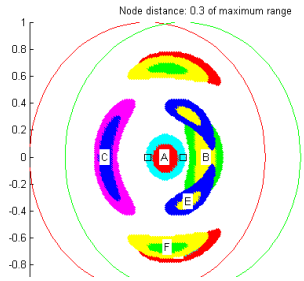
3.6 Functional analysis of evolved group-oriented protocol

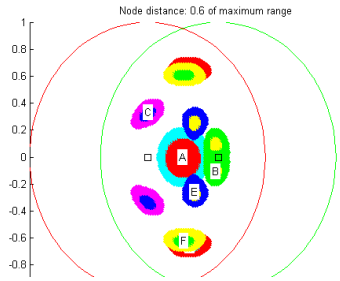
Using the instruction cross-dependency technique, we can conclude that the group of instructions {2,5 and 7} is responsible for most of the secured links. A new random value is generated in the node N_P and sent to the node N_x positioned in the middle between N_P and N_C. Node N_x then re-sends this value to the node N_C. This group of instructions is responsible for securing 80-95% of all secured links (depending on the average number of neighbours and the number of attacker's black nodes / compromised links). When instructions {2,5 and 7} are removed from the protocol, fraction of secured links decreases by about 20-40%, rest of links being secured by the other group of instructions {0,1,10 and 11} which secures part of the links originally secured by the removed instructions.

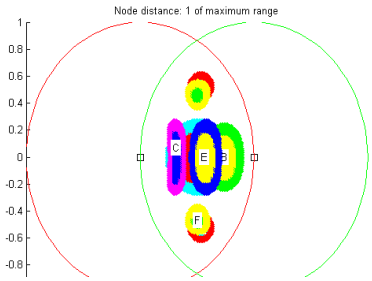
The fraction of secure links remains almost stable when more amplification iterations are executed with these two groups ({2,5,7} and {0,1,10,11}) of instructions separated. But when both groups of instructions are used together (original protocol), improvement by more iterations of the amplification protocol is possible. A 5-10% increase in number of secured links can be expected when the number of iterations is increased from 2 to 5. Note that obtaining even small improvements here is difficult, because we are very close to the theoretical maximum of secure links for a given number of black nodes/compromised links.

First two SND instructions may appear useless (no value is available in the memory slot 6 for the first run of the protocol), but as the protocol is executed repeatedly for all nodes within a group, this value can actually be present in memory slot 6 from a previous execution as a result of the instruction 7 or 10. Again, evolution is able to include such "overlapping executions" in the protocol, while this might be difficult for a human designer.

Surprisingly, the most important intermediate node is not positioned in the middle between two nodes (area A) trying to establish it keys. Instead, most probable position for that intermediate node is area C shown on Figure 20. Note that the area C is differently positioned based on the distance between nodes N_C and N_P. When these two nodes are close to each other then C is "behind" the node N_C (Figure 20, part a)). As the nodes move away from each other, the area C moves around the N_C to the position shown on Figure 20, part b). When both nodes are very close to the maximum transmission range then C is in one third of the distance between N_C and N_P, closer to N_C (Figure 20, part c)).


Node distance: 0.3 of maximum range
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
C
A
B
E
F


Node distance: 0.6 of maximum range
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
C
A
B
E
F


Node distance: 1 of maximum range
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
C
E
B
F

[SM07] Petr Svenda and Václav Matyáš. Automatic construction of secrecy amplification protocols. *The 3rd Wireless and Sensor Network Security Workshop, IEEE Computer Society Press. Los Alamitos, CA*, pages 21–26, 2007.

[SO05] Petr Svenda and Martin Osovsky. A forward onion encryption scheme for wireless sensor networks. *MEMICS 2005*, pages 38–44, 2005.

[Sve07] Petr Svenda. Automatic construction of secrecy amplification protocols. *MEMICS 2007*, 2007.

[WKC+04] Ronald Watro, Derrick Kong, Sue-fen Cuti, Charles Gardiner, Charles Lynn, and Peter Kruus. TinyPK: Securing sensor networks with public key technology. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks (SASN'04), Washington, DC, USA*, pages 59–64, 2004.

[YB05] Eiko Yoneki and Jean Bacon. A survey of wireless sensor network technologies: research trends and middleware's role. *Technical Report, UCAM 646, Cambridge*, 2005.