Faculty of Informatics
Masaryk University

The link key security in wireless sensor networks

Dissertation thesis

Petr Švenda

Brno, September 2008

I would like to dedicate this thesis to my beloved Linďa.

Acknowledgements

I like to express my very deep appreciation to my supervisor Vashek Matyáš for his ubiquitous care and support for and far beyond the study matters. Besides technical discussions, ideas, tireless proof reading and encouragement to pursue a wide range of research topics, he was able to create and maintain a pleasant atmosphere in our research group.

I am very graceful for suggestions, contributions and help from several other people. To Dan Cvrček for his significant work on new types of key infection protocols and forward onion encryption, which lead to the basic idea for group supported protocol. To Martin Osovský for his work on the analytical evaluation of performance for the group supported protocol. To Lukáš Sekanina for his continuous discussions and help with both technical and theoretical aspects of evolutionary algorithms used for the automatic generation of secrecy amplification protocols and attacker strategies. To Jirka Kůr for his part of work on attack strategy generation. To Nicholas Svenda for help with grammar corrections. And to my colleagues Honza Krhovják and Marek Kumpošt for our joint progress in graduate life.

I also like to thank all anonymous reviewers for their valuable comments and ideas which hopefully improved my research and presentation skills and highlighted weaknesses and open questions.

I like to thank my family, especially my parents Petr and Zdenka Svendovi for their persistent support all over my studies. And finally, I am very thankful to my wife Linda and daughter Zuzanka for all the overtime hours spent in the lab which they tolerated.

Abstract

This dissertation thesis targets the area of wireless sensor networks (WSNs), in particular their security of link key establishment. We focus on how link keys can be established in memory and computation restricted environment of WSNs, how link security behaves under a selected attack, and what methods can be used to strengthen their resilience against compromise. We based our work on the assumption that partial compromise in the WSNs is inevitable and network architecture should be prepared to cope with related security issues. We work with two basic link key establishment concepts based on symmetric cryptography – memory efficient probabilistic pre-distributions [21, 11, 10] and lightweight key exchange without pre-distributed secrets [3, 29, 16]. Both key distribution concepts behave differently when the network is attacked. We study resulting compromised patterns and propose two separate mechanisms based on support from neighbouring nodes for improving the network resiliency – one for the probabilistic pre-distribution and second for the Key Infection.

The first mechanism uses group support for authenticated key exchange to substantially increase the resilience of an underlying probabilistic key pre-distribution scheme against the threat of node capturing. The resiliency of probabilistic pre-distribution schemes generally increases if more keys can be put into key ring on every single node, but such an increase is limited by the node storage capacity. The proposed protocol creates a large virtual key ring in an efficient and secure way from the key rings of separate nodes. The proposed protocol itself is resilient against partial compromise inside a group of neighbours.

The second proposed mechanism improves the fraction of secure links after compromise of some links due to attacker eavesdropping when key exchange between neighbours is made in plaintext (Key Infection approach). Our proposed mechanism from the family of secrecy amplification protocols exploits the non-uniformity of link compromise patterns in Key Infection and provides a significantly better fraction of secure links than previously published protocols, especially for denser networks. We additionally provide detailed evaluation of existing secrecy amplification protocols with respect to network density, repeated iterations, composition of protocols and different quantities of attacker eavesdropping nodes on our network simulator – previous works dedicated little attention to these aspects.

We later realized that the secrecy amplification protocols can also be used to improve the fraction of secure links and strengthen node capture resilience for probabilistic pre-distribution. It works even better here than for the Key Infection approach for which the secrecy amplification protocols were originally proposed for. A network with half of its links compromised can be made reasonable secure

with less than 10% of compromised links when the secrecy amplification protocols are applied. However, some combinations of secrecy amplification protocols that worked for Key Infection do not work for probabilistic pre-distribution (do not increase number of secure links) and thus only impose unnecessary communication overhead. Instead of analyzing each separate compromise pattern arising from the combination of a particular key distribution method and attacker strategy, we proposed an automated approach based on the combination of a protocol generator and network simulator.

We utilize evolutionary algorithms (EA) [7] to facilitate guided search for well-performing secrecy amplification protocol created as a series of elementary instructions. Every candidate protocol is evaluated on our network simulator for a particular compromise pattern. The protocols with a better fraction of secure links are used as templates ("parents") for the next generation of candidate protocols. Using this method, we were able to automatically re-invent all human-designed secrecy amplification protocols proposed so far and find a new protocol that outperforms them. With respect to classical human-made protocols, an increase in number of secure links was obtained by the efficient combination of the simpler protocols and an unconventional interleaving of elementary instructions that enable protocol execution even when one of the participants is out of reach of the radio transmission.

The practical disadvantage of secrecy amplification protocols is the number of necessary messages resulting in high communication overhead during the link key establishment. We propose an alternative construction which exhibits only linear (instead of exponential) increase of necessary messages when the number of neighbours in communication range (network density) is growing. As a message transmission is a battery expensive operation, this more efficient protocol can significantly save this resource. Designing secrecy amplification protocol in such scenario is more difficult as more parties are involved and the relative positions of the nodes must be taken into account as well. Again, we used described an automatic protocol search to find a protocol for a message restricted scenario with comparable performance to protocols with original assumptions.

Finally, we explore the dark side and propose a new concept for automatic search for attack strategies with demonstrative applications to link key security for probabilistic pre-distribution and Key Infection approaches. The similar framework as for protocols generation was used and candidate attacker strategy is combined from elementary operations and evaluated on network simulator or in real system. Attacker strategies that increase the number of compromised links with respect to several deterministic algorithms or random case were found. Our framework can be used to improve the success of an attacker with the ability to perform selective actions and even to provide novel and unconventional attacks.

Table of Contents

1 Introduction 1

1.1 Target of interest . . . 2

1.1.1 Target scenarios for this thesis . . . 3

1.1.2 Node-compromise attacker model . . . 5

1.1.3 Cryptographic issues in network bootstrapping and main results . . . 6

1.2 Developed network simulator . . . 8

1.2.1 Simulator architecture . . . 9

1.2.2 Network deployment . . . 10

1.2.3 Support for secrecy amplification protocols . . . 10

1.2.4 Support for probabilistic pre-distribution . . . 11

1.3 Structure of the thesis . . . 11

2 Authenticated key exchange with probabilistic pre-distribution 13

2.1 Related work . . . 13

2.1.1 Random key pre-distribution . . . 14

2.1.2 Pairwise key pre-distribution . . . 15

2.1.3 Limited neighbour knowledge schemes . . . 16

2.1.4 Seed-based pre-distribution . . . 18

2.1.5 Selective node capture attack . . . 18

2.1.6 Hypercube pre-distribution . . . 20

2.2 Group supported key exchange . . . 20

2.2.1 Authenticated key exchange with group support . . . 22

2.2.2 Probabilistic authentication . . . 24

2.2.3 Probabilistic authentication with majority decision . . . 26

2.2.4 Evaluation of the communication and computation overhead . . . 26

2.3 Group support with EG scheme . . . 27

2.3.1 Evaluation of node capture resilience improvements . . . 27

2.3.2 Options and settings . . . 28

2.3.3 Group enlargement . . . 30

2.4 Group support with multi-space polynomial scheme . . . 31

2.4.1 Evaluation of node capture resilience improvements . . . 33

2.4.2 Impact of polynomial security threshold 33
2.4.3 Impact of minimal shared keys threshold 35
2.4.4 Comparison with hypercube scheme 35
2.5 Possible attacks and defenses 37
2.5.1 Passive attacker 37
2.5.2 Active attacker 38
2.6 Summary of the protocol 39

3 Secrecy amplification 40
3.1 Related work 41
3.1.1 Plaintext key exchange – whispering 42
3.1.2 Basics of secrecy amplification protocols 43
3.1.3 The mutual whispering protocol 44
3.1.4 The Push protocol 44
3.1.5 The Commodity protocol 45
3.1.6 The Pull protocol 45
3.2 Analysis of secrecy amplification protocols 46
3.2.1 Network settings and simulation setup 47
3.2.2 Discussion of simulation results 48
3.2.3 Transmission overhead 51
3.2.4 Compromise success for eavesdropping nodes 52
3.3 Key Infection analysis conclusions 53
3.4 Compromise patterns of key distribution 54
3.4.1 Random compromise pattern 55
3.4.2 Key Infection compromise pattern 55
3.5 Generation of secrecy amplification protocols 56
3.5.1 Composition from simple secure protocols 56
3.5.2 Evolutionary algorithms 57
3.5.3 Primitive instructions set 57
3.6 Generation of node-oriented protocols 58
3.6.1 Overview of the method 59
3.6.2 Parameters of experiments 59
3.6.3 Results for node-oriented protocols 60
3.7 Generation of group-oriented protocols 62
3.7.1 Overview of the method 62
3.7.2 Results for group-oriented protocols 65
3.7.3 Methods for analysis of evolved protocols 66
3.7.4 Functional analysis of evolved group-oriented protocol 68
3.8 Parameters used for LGP 69
3.9 Automatic protocol generation conclusions 70

4 Evolution of attacker strategies **71**
4.1 Related work 71
4.2 Automatic design of attacks 72
4.2.1 General concept 73
4.2.2 Strategy description 74
4.2.3 Fitness function construction 74
4.3 Evolution of attack strategies 75
4.3.1 Re-combination of the existing attacks 75
4.3.2 Improvement (optimization) of known attack strategy 76
4.3.3 Finding novel attack strategies 76
4.3.4 Promising areas 76
4.4 Applications 77
4.4.1 Optimal eavesdropping pattern 77
4.4.2 Selective node capture 79
4.4.3 Incomplete fitness function specifications 81
4.5 Conclusions for attack strategies 82

5 Concluding remarks **83**
5.1 Future research opportunities 86

Bibliography **89**

Chapter 1

Introduction

Advance in miniaturization of electronics opens the opportunity to build devices that are small in scale, can run autonomously on battery and can communicate on short distances via wireless radio. 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, sinks or cluster controllers*) 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. These nodes are typically equipped with an environment sensor (e.g., heat, pressure, light, movement). Events recorded by the sensor nodes are locally collected and then forwarded using multi-hop paths to a base station (BS) for further processing.

Wireless networks are widely used today and they will become even more widespread with the increasing number of personal digital devices that people are going to be using in the 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 and 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 the development of very small and cheap sensors (e.g., [2]) as well as research in middleware architectures [62] and routing protocols (AODV [13], DSR [17], TORA, etc.) for self-organising networks – to name a few.

No common hardware architecture for WSN is postulated and will depend on the target usage scenario. Currently available hardware platforms for sensor nodes range from Mica Mote2 [1] or TMote Sky equipped with 8/16-bit processor with less than 10 MHz clock frequency down to Smart Dust motes [2] with their total size around $1mm^3$ and extremely limited computational power. No tamper resistance of the node hardware is usually assumed.

Security often is an important factor of WSN deployment, yet the 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 limited battery power, communication traffic should be kept as low as possible and most operations should be done locally, not involving the more powerful and (possibly) trusted base station.

In this thesis, we will focus only on a small subset of security issues, namely robust link key establishment between neighbouring nodes within a radio communication range. We focus on schemes with low memory requirements and without the involvement of the trusted base-stations, resilient against partial compromise of the network due to eavesdropping or node capture. We also inspect the opposite side, designing attacker strategies to increase attacker success in the compromise of the network.

The main contribution of this thesis is as follows. A protocol design for authenticated key exchange with improved resilience against the threat of node capturing over existing probabilistic pre-distribution schemes presented in Chapter 2. A detailed analysis of the performance of existing secrecy amplification protocols for the Key Infection key establishment approach, with an additional protocol with a better fraction of secured links than the existing ones is presented in Chapter 3 (Section 3.1). Our method for automatic design of secrecy amplification protocols for an ordinary compromise pattern based on evolutionary algorithms is presented in Chapter 3 (Section 3.5). An automatic attacker strategy generation framework for attacks against secrecy amplification protocols and probabilistic pre-distribution are presented in Chapter 4.

Parts of this thesis originated in our research previously published in paper [16] (the Pull amplification protocol), papers [60, 58] (group supported protocol with Eschenauer and Gligor pre-distribution), chapter in book [59] (group supported protocol combined with multiple key spaces predistribution by Du et al.), paper [57] and chapter in book [50] (basic framework for automatic protocol generation). The rationale behind the assumption of the availability of truly random number generators (TRNGs) with a sufficient speed for secrecy amplification is based on our work on TRNGs in restricted mobile environments [32] and [31].

1.1 Target of interest

Secure link communication is the building block for large part of the security functions maintained by the network. 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. The aggregation of the data from separate sensors needs to be authenticated; otherwise an attacker can inject his own bogus information. Routing algorithms need to utilize the authentication of packets and neighbours to detect and drop malicious messages and thus prevent network energy depletion and route messages only over trustworthy nodes. On top of these common goals, secure and authenticated communication between neighbours can be used to build more complex protocols designed to maintain reliable sensing information even in a partially compromised network. The generally restricted environment of WSNs is a challenge for the design such protocols.

1.1.1 Target scenarios for this thesis

Early stage of the wireless sensor networks design with sparse real-world implementations naturally results in wide range of the assumptions in theoretical works about network architecture (static, dynamic, possibility of redeployments, degree of possible disconnection), topology (completely decentralized, with clusters, various tree-like structures), size (from tens of nodes up to hundreds of thousands of nodes) and expected lifetime (short time and task specific networks vs. long term ubiquitous networks embedded in the infrastructure). Various assumptions are made about the properties of sensor nodes, namely their mobility (fixed, slowly moving, frequent and fast change), available memory, computation power (symmetric cryptography only, regular use of asymmetric cryptography), energy restrictions (power grid, replaceable/non-replaceable batteries) or availability of special secure hardware (no protection, secure storage, trusted execution environment). Different assumptions are made also about network operation mode (continuous, seasonal, event-driven), how are queries issued and propagated (all nodes report in certain intervals, on demand reading) and how are data from sensor nodes transported back to base stations (direct communication, data aggregation). Finally, attacker model ranges from fully fledged adversary with capability to monitor all links and selectively compromise nodes to more restricted attacker with limited resources, presence and priory knowledge about attacked network.

Here, we will define assumptions about network and nodes that will be used in this thesis.

Network architecture – We assume mostly static network with large number of nodes ($10^3 - 10^6$) deployed over large area without the possibility for a continuous physical surveillance of the deployed nodes. Redeployments are possible for group supported protocol in Chapter 2 and secrecy amplification protocols applied with probabilistic pre-distribution (Section 3.5), but not for Key Infection approach. We studied networks with densities (number of neighbours in reach of radio transmission) from very sparse (e.g., two nodes on average) up to very dense (forty nodes on average) networks.

Network topology – We assume network topology that requires link keys between direct neighbours (e.g., for link encryption, data aggregation...), leaving more advanced topology issues related to routing and data transmission unspecified. Although is possible in principle to use proposed protocols to establish also peer to peer (p2p) keys with distant nodes, we focus on link key establishment and do not explicitly discuss p2p keys as additional assumptions about network topology and routing algorithms are then needed.

Network lifetime – We focus on networks with long expected lifetime where most decisions should be done locally and the message transmissions should be as low as possible. A higher communication load is expected for the initial phase after the deployment or during the redeployments, when new link keys are established. Section 3.7 specifically targets message overhead of secrecy amplification protocol to improve overall network lifetime.

Base stations – Base stations are not addressed much in our work as we target link keys between direct neighbours without involvement of base station itself. We assume that only few base stations are presented in the network and high majority of the ordinary sensor nodes cannot use advantage provided by possible trusted and powerful positioned closely to base station.

Degree of centralism – We focus on networks where security related decisions during the key establishment should be done without a direct involvement of a base station and the nodes are not assumed to communicate with base station(s) later on regular basis (e.g., for later verification of passed key establishment process). A base station may eventually take part in the process (e.g., during the redeployment), but is not assumed to communicate directly with each separate node or act as mediator between nodes.

Nodes mobility – We target scenarios for which make sense to establish link keys. These links do not necessarily remain static during whole network lifetime. The protocol described in Chapter 2 assumes an existence of link keys with direct neighbours, therefore high mobility of nodes is not desirable here or link keys must be re-established when necessary. On the other side, nodes mobility provides better opportunity for selection of parameters of the network than for case of fixed immobile nodes. Relatively static set of neighbours should remain present at least during execution of secrecy amplification protocols. Provided simulations in Chapter 3 assume that nodes do not change their position during the protocol.

Communication medium – We target nodes with wireless communication with assumption of non-directional antenna with controllable transmission power for Key Infection approach. An ideal propagation of the signal is assumed in the simulations. The communication medium and antenna properties are not relevant for the part of the work focused on probabilistic pre-distribution.

Computation power – We target nodes capable to transparently use symmetric key cryptography, but not the asymmetric cryptography. Even when encryption/verification with asymmetric cryptography keys might be possible on hardware in principle (e.g., verification of signature in two seconds on Mica2 motes [53]), protocols proposed in this thesis generally require processing of high number but small messages. Usage of asymmetric cryptography then significantly increases the total time required to finalize link key establishment. Still, secrecy amplification protocols can be used atop of a basic link key exchange facilitated by an asymmetric cryptography.

Memory limitations – Our work targets devices with limited memory. Protocol described in Chapter 2 requires memory in order of kilobytes, secrecy amplification protocols from Chapter 3 requires storage for values exchanged only between direct neighbours, totally less than kilobyte for a reasonably dense network.

Energy source – We assume that nodes energy is limited and therefore nodes should not communicate too often and most of the decisions should be made locally. Energy limitation is additional reason why symmetric cryptography is preferred over asymmetric in this thesis as detection of corrupted message with integrity protection based on symmetric cryptography is much more efficient.

Tamper resistant hardware – We do not assume existence of any secure hardware on the sensor node side as such protection increases cost significantly, especially for large scale networks we are interested in. All secrets from captured nodes are assumed to be compromised. The automatic search for selective eavesdropping particulary exploits the possibility for a key extraction.

Pre-distributed secrets – We work with the pre-distributed secrets in form of probabilistic pre-distribution in Chapter 2 and with part of secrecy amplification protocols. No pre-distributed secrets are assumed when Key Infection approach is used. Secrecy amplification protocols alone do not require any pre-distributed secrets, only availability of a suitable (pseudo-)random generator is required.

Routing algorithm – We assume existence of suitable routing algorithm to propagate message in closed geographical area, usually inside group of direct neighbours and we abstract from algorithms specifics in our work. If multiple paths for delivery of parts of fresh key during secrecy amplification are used, mechanism for packet confirmation should exist. Otherwise composed key will be corrupted if one or more parts lost in transmission.

1.1.2 Node-compromise attacker model

The common attacker model in the network security area is an extension of the classic Needham-Schroeder model [42] called the node-compromise model [21, 10, 19, 20]. Original model assumes that 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. Extended model is 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. Part of our work aims to omit this phase completely as it is cheaper to produce identical nodes (even at the memory level).

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

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

The attacker model is in some cases (Key Infection [3]) weakened by the following assumption:

A4: For a 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 we have to consider a stronger attacker with the ability to eavesdrop all communication. The attacker with a limited number of eavesdropping devices can eavesdrop only a

fraction of links and the rational reason behind this assumption is based on specifics of WSNs:
a) Locality of eavesdropping – the low communication range of nodes allows for a frequent channel reuse within the network and detection of extremely strong signals, 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 – a low threat in most scenarios during first few seconds before the attacker realizes what target area is in use. If the attacker nodes are already present in a given amount in the target location, we can deploy a network with density and node range such that the the ratio between legal nodes and the attacker's eavesdropping devices is such that a secure network can be formed.

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

1.1.3 Cryptographic issues in network bootstrapping and main results

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

The 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. Not all steps need to be executed for different key establishment approaches. The main phases are as follows:

1. Pre-deployment initialization – performed in a trusted environment. Keys can be pre-distributed during this phase.
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 – nodes are trying to find their direct neighbours (nodes that can be directly reached with radio) and to establish communication channels.
4. Neighbour authentication – authentication of neighbours with pre-shared secrets, via trusted base station, etc.
5. Key setup – key discovery or key exchange between direct neighbours.
6. Key update – periodic update of initial keys based on events like secrecy amplification, join to cluster, node revocation or new nodes redeployment.

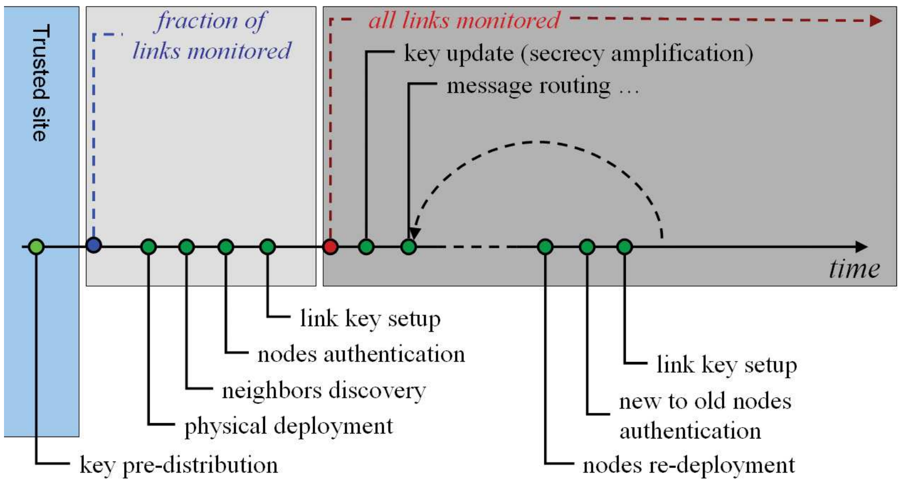


Figure 1.1: Network lifetime with highlighted phases relevant for the key establishment. Periods of attacker eavesdropping capability are depicted for the Key Infection model.

7. Establishment of point-to-point keys – the final goal is to transmit data securely from sensors to one of a few base stations. Point-to-point keys are pairwise keys between sensors and base stations (or distant sensors).

8. Message exchange – the production phase of the network.

The main issues in the area of key link establishment for WSNs can be summarized as follows. We will discuss selected issues in more details in the respective chapters.

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 the compromise of even a single node in the network enables the decryption of all traffic. The master key scheme has no resilience against node capture. As recognised in [10], this approach is suitable only for a static network with tamper resistant nodes. An additional work extended master key scheme with an assumption of secure erase after certain time period [63] and limit impact of master key compromise [18].

Full pairwise key scheme. A contrast to the master key scheme, where a unique pairwise key exists between any two nodes. As shown in [19], this scheme has 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. The usage of PKIs for WSNs is often assumed as unacceptably expensive in terms of special hardware, energy consumption and processing power [46, 21, 3]. The usage of asymmetric cryptography can lead to energy exhaustion attacks by forcing the node to frequently perform expensive signature verification or creation [4]. Energy efficient architecture based on elliptic curves is proposed in [61, 53] with signature verification in order of small seconds. The maintenance and verification of a fresh revocation list and attacks like the collusion attack [40] remain a concern.

Base station as a trusted third party. Centralised approaches like the SPINS architecture [46] use the BS as a trusted mediator when establishing a pairwise key between two nodes. Each node initially shares a unique key with the base station. This approach is scalable and memory efficient, but has a high communication overhead as each key agreement needs to contact the BS, causing non-uniform energy consumption inside the network [44].

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 the whole network connected [21, 11, 10]. 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/them as a link key, if such key(s) exist. Variants based on threshold secret sharing provide better resilience against the node capture [19, 34].

Deployment knowledge pre-distribution. The efficiency of probabilistic predistribution can be improved if certain knowledge about the final node position or likely neighbours is available in advance. Ring keys selection processes based on node physical distribution allocation are proposed in [20, 9, 35]. The nodes that have a higher probability to be neighbours have keys assigned in a such way to have higher probability to share a common key.

Key infection approach. An unconventional approach that requires no predistribution is proposed in [3]. The weakened attacker with limited ability to eavesdrop is assumed for a short period after 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 [3, 29, 16].

Impact of Sybil and collusion attack. Powerful attacks against known key predistribution protocols are presented in [43, 40]. 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 the global usage of secrets in a distributed environment with no or little physical control poses a serious threat, which is hard to protect against.

1.2 Developed network simulator

I have developed network simulation software which was extensively used for simulation results presented in this thesis. Development of the new simulator instead of using an existing one was motivated by two main reasons:

No suitable simulator was available for simulation of large scale networks when we started our work as we aimed for simulations of up to hundreds thousands nodes.

The simulation of large networks requires significant speed and memory optimization that are hard achieve for a general purpose simulator. Our purpose-built simulator provides the level of abstraction suitable for a particular tasks, lowering computation and memory requirements.

Simulation speed is a critical factor for most of our experiments, especially for the usage in automatic protocol generation with evolutionary algorithms (see Section 3.5) where up to hundreds of thousands full network simulations must be performed in a reasonable time. With our purpose-built simulator, we have been able to focus on and simulate only relevant operations, obtaining significant performance improvements over a general purpose simulator.

The development of a new simulator has some drawbacks as well. Besides significant time being invested in the design and implementation, the new simulator makes the comparison with/of results with other research more difficult. We were aware of this fact and provided analytical results (group supported key exchange, Section 2.2) to complement simulation results. Secrecy amplification protocols found by automatic generation approach can be readily implemented and performance results can be compared in another simulators. We also made source codes of our simulator publicly available[1].

The capabilities of our simulator will be discussed in the following sections.

1.2.1 Simulator architecture

Our simulator has been developed for Microsoft Windows 2K/XP/Vista operating systems in C++, currently having almost forty thousand lines of code. The simulator can be compiled both for 32-bit and 64-bit Intel/AMD architectures, with the 64-bit version providing some performance advantage.

A pseudo-random generator based on the MD5 hash function is used to provide pseudo-random data where required, starting from a specified seed. Once the initial seed is fixed, the simulator steps are fully deterministic, except for random operations used internally by evolutionary algorithms where determinism is undesirable. Truly random data is taken from the QRBG service [2] or from data extracted by our approach described in [31]. Simulations are repeatable if the seed is preserved.

The parameters of a simulation can be set either from the GUI or can be provided from a configuration file. Once a simulation is started, all configuration parameters are saved together with the used random seed, making an exact repetition of a particular simulation possible.

The simulator can be run with all settings given from the command line. Support for execution on either local or remote machines with centralized management is implemented. Speedup of extensive simulations with naturally parallel operations can be distributed over

[1]http://www.fi.muni.cz/∼xsvenda/s3.html
[2]http://random.irb.hr/

several machines with real-time reporting to the central management application (e.g., each deployment is simulated on a separate machine).

Support for batch sequential execution of several simulations is also implemented. The start of a particular simulation can be delayed until the previous simulations finish. The selected characteristics of network deployment can be automatically iterated over a specified range with fixed steps, providing separate simulations (e.g., increasing network density within a given range with a defined increase step).

Detailed data produced during simulation is written into files with format suitable for later processing with the Matlab software. Several Matlab scripts were created for further processing, data visualization and graph plotting.

1.2.2 Network deployment

The deployment of nodes is done in a specified two-dimensional rectangle area, every node having a position in the x and y axes with float number precision. Nodes can be deployed either randomly or according to a freely specifiable pattern. Every node has a definable transmission range with connection to nodes in range established during neighbour discovery. Eavesdropping nodes can be deployed similarly to legal nodes.

Multiple deployments can be performed at the same time. Subsequent operations are performed over all deployments with some simulation results averaged. The averaging of results over multiple deployments decreases the dependence of simulation statistics on the particular placement of nodes. This property is of a special importance for automatic protocol generation, where over-learning on a particular deployment is undesirable.

1.2.3 Support for secrecy amplification protocols

The deployment of the nodes can be followed by establishment of link keys between neighbours in transmission range with several supported key exchange mechanisms (maximum screaming, whispering, random and probabilistic pre-distribution). Links are marked either secure or compromised based on the attacker model and actual distribution of eavesdropping nodes (Key Infection) or node capture (probabilistic pre-distribution). Such mesh of links with different secrecy state (secure/compromised) forms a starting point for secrecy amplification protocols. These protocols are either directly implemented in the code (used for performance comparison in Section 3.1) or can be specified in a metalanguage from script (slower but more flexible option) as a sequence of elementary instructions (see Section 3.5.3).

Metalanguage-based execution of amplification protocols is used for automatic protocol generation as described in Section 3.5. The simulator contains a built-in protocol generator based on evolutionary algorithms (The GALib package[3] was used for the implementation). Generated protocols are executed in the deployed networks and evaluated with a fraction of

[3]http://lancet.mit.edu/ga/

secure links used as an indicator of protocol quality. See Section 3.5 for more details about the whole process. Interrupted protocol generation can be restarted from a temporal state stored in files generated during simulation.

1.2.4 Support for probabilistic pre-distribution

Support for probabilistic key pre-distribution is implemented with evaluation of the number of secure links after a compromise of specified number of randomly or selectively captured nodes. The basic schemes described in [11] and [21] are implemented. The simulator is optimized for fast queries with respect to the a subset of shared keys and the evaluation of the compromise impact of node capture. Simulations from this part were used as additional verification of analytical results proposed in Chapter 2.

Support for the automatic generation of node capture attacks is implemented (see Chapter 4 for details). The subset of nodes selected either randomly or selectively is marked as compromised, keys are extracted and the compromise impact on the network links is evaluated. The implementation of evolutionary algorithms based on the GALib package is used.

1.3 Structure of the thesis

The text of this thesis is structured into four main parts. The introduction, outline of my research focus and a high-level description of the developed simulator are provided in this first chapter.

The second chapter presents several key pre-distribution schemes for WSNs and discusses their properties, namely the resilience against the node capture attack. We then focus on extension protocols that are able to improve the security for the price of additional communication. The protocol described in Section 2.2 uses a combination of group support from neighbour nodes and probabilistic pre-distribution to provide authenticated key exchange, and here we introduce also the concept of probabilistic authentication. The group of neighbours creates a large virtual keyring and thus boosts resiliency against the scenario where each node has only its own (limited) keyring available. Analytical and simulation results are provided, together with a detailed comparison with node-capture resilience of other existing schemes.

The third chapter is devoted to secrecy amplification protocols. Analysis with simulation results of existing secrecy amplification protocols and a new protocol proposed by us is provided in Section 3.1. Section 3.5 extends secrecy amplification to a partially compromised network resulting from capture of nodes with keys distributed by probabilistic pre-distribution. A flexible framework for automatic generation of secrecy amplification protocols for an ordinary compromise pattern is described with results presented for two specific patterns arising from Key Infection and probabilistic pre-distribution key establishment. A combination of evolutionary algorithms and network simulator is used to generate and test candidate amplification protocols. A different execution model for amplification protocols is introduced with linear instead of exponential increase of message overhead with increasing network density. An automatic approach is then used to generate a well-performing protocol for the new execution model with a fraction of secured links comparable to older (message expensive) protocols.

The fourth chapter then describes a framework for the automatic search for attacker strategies, again based on the combination of evolutionary algorithms for the construction of candidate attacks and the network simulator for evaluation of its success. The proposed framework is used to generate deployment patterns for eavesdropping nodes to maximize fraction of compromised links for the Key Infection key establishment and selective node capture strategy to increase the fraction of compromised links for probabilistic key predistribution.

The overall conclusion with an outline for future work is presented in the final chapter.

Chapter 2

Authenticated key exchange with probabilistic pre-distribution

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 link 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 keys is thus not possible in this scenario due to the potentially high number of neighbours and limited memory of a single node.

This chapter will provide a summary of related work in the area of probabilistic key pre-distribution, approach suitable for the memory constrained nodes and propose extension protocol executed atop given probabilistic pre-distribution method. Proposed protocol employ the support from the neighbouring nodes to create the large virtual key rings in efficient and compromise tolerant way. We will analyze and simulate increase in node capture resilience when the proposed protocol is used.

The idea of group supported protocol with Eschenauer and Gligor pre-distribution was published in paper [60] and extended in paper [58]. Combination of the protocol with multiple key spaces predistribution by Du et al. was published as a chapter in book [59].

2.1 Related work

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. 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).

2.1.1 Random key pre-distribution

The idea of random key pre-distribution for WSNs is introduced for the first time by Eschenauer and Gligor [21] (referred to as EG scheme) and is based on a simple but elegant idea. At first, a large key pool of random keys is generated. For every node, randomly chosen keys from this pool are assigned to its (limited) key ring, yet these *assigned keys are not removed from the initial pool*. Hence the next node can also get some of the previously assigned keys. Due to the birthday paradox, probability of sharing at least one common key between two neighbours is surprisingly high even for a small size of the key ring (e.g., 100 keys). That makes this EG scheme suitable for memory-constrained sensor nodes. Resilience of known pre-distribution schemes against the threat is evaluated by Chan et al. [10]. Attacker obtains some keys from the initial key pool by picking and reverse-engineering captured nodes. These keys are then used to decrypt eavesdropped messages. The success rate of decryptions depends on the number of compromised keys (nodes). We argue that multiple occurrence of the same key on the multiple nodes used for authentication purposes is not a real problem if we ensure that a new node comes from the same deploying authority rather than actually care about the identity of the node itself (can be viewed as a variation of group authentication).

Chan et al. [10] extends the EG schema by q-composite random key pre-distribution, requiring at least q shared keys instead of one (referred to as q-EG). Link key is constructed using hash function from at least q shared keys. The number of a required shared keys makes it exponentially harder for an attacker to compromise the link key with a given subset of already compromised keys, but also lowers the probability of establishing a link key. If the node key ring size m is fixed, total size of key pool S must be reduced to preserve same key establishment probability, and thus the attacker obtains a larger fraction of S from a single node. A formula for optimum tradeoff is given. Impact of multi-path key reinforcement, introduced in [3] together with q-EG is studied. The random pairwise scheme is also described (see Section 2.1.2). The q-EG scheme [10] provides significantly better node-capture resilience than basic EG [21] until some threshold is reached.

Pietro et al. [47] extends the EG scheme using pseudo-random generation of key indexes rather than completely random (referred to as seed-based key deployment). The advantage is that two neighbours can compute identification (not the key value) of their shared keys only from their node identifications with no additional communication messages. A co-operative version of the seed-based key deployment protocol is described, performing secrecy amplification with a set of common neighbours of participants A and B. A chooses randomly the set of B-neighbours (mediators C_i) and asks them for computation of $HMAC(ID_A, K_{C_iB})$. Resulting values from each mediator are XORed together with the original key value K_{AB} and used as the new key value. Node B can compute new key value only from the information who were the mediators used, with no additional messages.

2.1.2 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[1], but is poorly scalable and has high memory requirements. Note that perfect resilience against the node capture does not mean that an attacker cannot obtain a significant advantage by combining keys from the captured nodes (e.g., collusion attack [40]).

Chan et al. in [10] propose a modification of the basic pairwise scheme (referred to as CPS scheme). Based on the required probability p that two physical neighbours will share a key, unique pairwise keys for X are generated, but only for m other randomly chosen nodes. In a contrast to the EG scheme, node-to-node authentication can be performed. Total number of nodes in a network is limited by $n = m/p$. Support for distributed revocation of a compromised node is proposed. During the initialisation phase, each node Y_i sharing key with node X obtains also a secret voting information, which can be used against malfunction node X when detected. The vote can be then broadcasted and node X marked by Y_i as revoked if the number of received votes exceeds a specific threshold value. Merkle hash tree is used to decrease storage needs. A masking mechanism that allows only direct neighbours of X to vote against X serves as a prevention to the revocation attack, where an attacker uses captured votes against legal nodes. A valid vote key can be constructed after deployment only if the masked key is combined (e.g., XORed) with some secret information carried by X.

A key pre-distribution scheme (referred as Blom scheme) that allows any pair of nodes to find a pairwise secret key is proposed by Blom [8]. Blom scheme requires substantially less memory than (n-1) pairwise key scheme and still allows for computing pairwise keys between each two nodes. However, Blom scheme is perfectly resilient only if not more than λ nodes are compromised (λ-secure property). If only one global key space of Blom scheme is used, λ must be unwieldy high and so does the required memory to resist against the node capture. Scalability of such approach is then poor.

A solution based on multiple key spaces is proposed by Du et al. [19] (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 Section 2.1.1). 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.

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

Hwang and Kim [27] revisit the basic random pre-distribution EG scheme (2.1.1), CPS scheme (2.1.2) and DDHV (2.1.2) 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 [27]. 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 by Liu and Ning [34]. 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 2.1.6 and 2.4.4 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 [11].

Impact of node replication (Sybil) attack against EG, q-EG and Blom scheme is evaluated by Fu et al. [24]. The work evaluates how much can an adversary gain after injecting certain number of replicated nodes and which scheme is most resilient against the replication attack, both through theoretical and experimental results. It is shown that success of the replication attack grows with the network density.

A novel collusion attack against the pairwise key pre-distribution schemes is presented by Moore [40]. Compromised nodes are sharing their secrets to increase probability that one of them will be able to establish the link key with its neighbours. This attack differs from the Sybil attack as node identities are not randomly generated, but instead are reused according to the available pairwise keys. A distributed voting scheme can be undermined by a 5% colluding minority since this minority is able to establish approximately one half of valid communication channels.

The pairwise key schemes can provide node-to-node authentication, but support a lower number of nodes in the network in comparison to the EG scheme. Combination of the ideas from EG scheme and Blom scheme (DDHV scheme) provides better resilience against node capture, until some threshold is reached (see Figure 2.6).

2.1.3 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.

Two schemes that are using certain deployment knowledge are presented in [35]. The CPS scheme is used (see Section 2.1.2), 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 [20] 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 [9]. 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 [12]. 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 [26] 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 Section 2.1.2) 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.

2.1.4 Seed-based pre-distribution

Seed-based pre-distribution is an extension of a given pre-distribution scheme, introduced by [47]. Rather than completely random, a pseudo-random generation is used to determine key indexes of the keys that will be assigned to a given node. The advantage is that two neighbours can compute identification (not the key value) of their shared keys only from their node identifications with no additional communication messages. Suitable key assignment rule for probabilistic pre-distribution can be constructed in the following way:

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

2.1.5 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 [26]. 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 [38]. Existing nodes in the network do not use their original IDs

[2]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 $poolSize$.

Notation	Description
$\|$	concatenation operator
B	identification of node that likes to authenticate to existing group
A	identification of node that will authenticate node B
N_i	identification of neighbours within communication range
ID_{ij}	identification of keys shared between N_i and B
$K_{ID_{ij}}$	shared key with ID ID_{ij}
K'_{ij}	one-time onion key generated by N_i for shared key with ID ID_{ij}
R_B	random nonce generated by node B
R_{ij}	random nonce generated by node N_i for key K_{ij} B
K'_{xy}	new link key shared between nodes x and y after secrecy amplification
$H(.)$	application of cryptographical strong one-way hash function H
$MAC(.)$	application of cryptographical strong message integrity function
$E_{K_{xy}}(.)$	symmetric encryption function using key K_{xy}

Table 2.1: Notation used for authenticated key exchange with group support. Function H should be realized as HMAC construction [6] instead of simple hash function to prevent extension attacks.

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 during the first contact 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" $C_{Ai} = MAC_{K_i}(H(0|N_A|N_B))$ using all its keys. A similar set C_B is computed as $C_{Bi} = MAC_{K_i}(H(1|N_B|N_A))$ 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.

2.1.6 Hypercube pre-distribution

The extension protocol for probabilistic polynomial pre-distribution called hypercube scheme is presented in [34]. 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 [19]. Then nodes with the same index within a given dimension (rows and columns in 2-dimensional case, shown on Figure 2.1) are assigned with a polynomial from the same key space and thus are able to establish shared pairwise key directly. Nodes are then deployed and perform ordinary neighbour discovery phase. If two nodes A and B wish to establish a pairwise key, all indexes of nodes within all dimensions are compared. If at least one index is shared then a key can be directly established. Otherwise other nodes are asked for cooperation such that virtual path from A to B can be formed $(A, C_1, C_2, \ldots C_n, B)$, where A is able to establish direct pairwise key with C_1 (they have same index in at least one dimension), C_i with C_{i+1} and C_n with B. New pairwise key is then generated on A and transported with re-encryptions over intermediate nodes C_j to node B. The proposed scheme assumes that compromised nodes/links are known and thus the non-compromised path can be selected. With the growing dimension of the hypercube, number of such paths is significant. If at least one non-compromised path exists, secure pairwise key can be established. Knowledge of compromise nodes/links is vital for the scheme – 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 2.2. Comparison with our group supported scheme (will be described later) is presented in Section 2.4.4.

2.2 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).

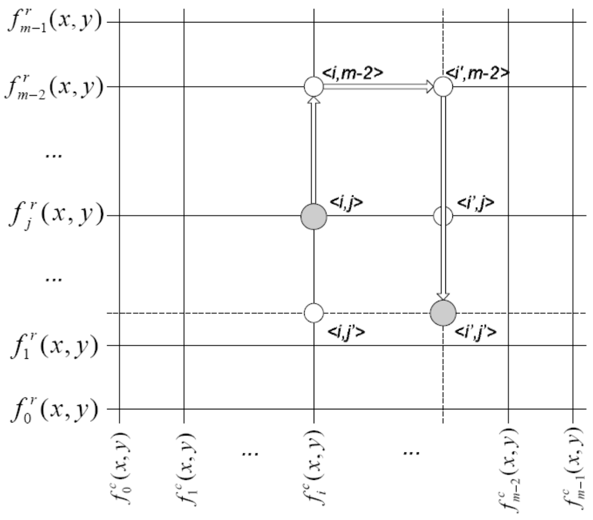


Figure 2.1: Polynomial assignment in two-dimension hypercube (grid). Nodes in the same row or column can establish key directly, otherwise with the help of neighbours in virtual grid. Figure taken from [34].

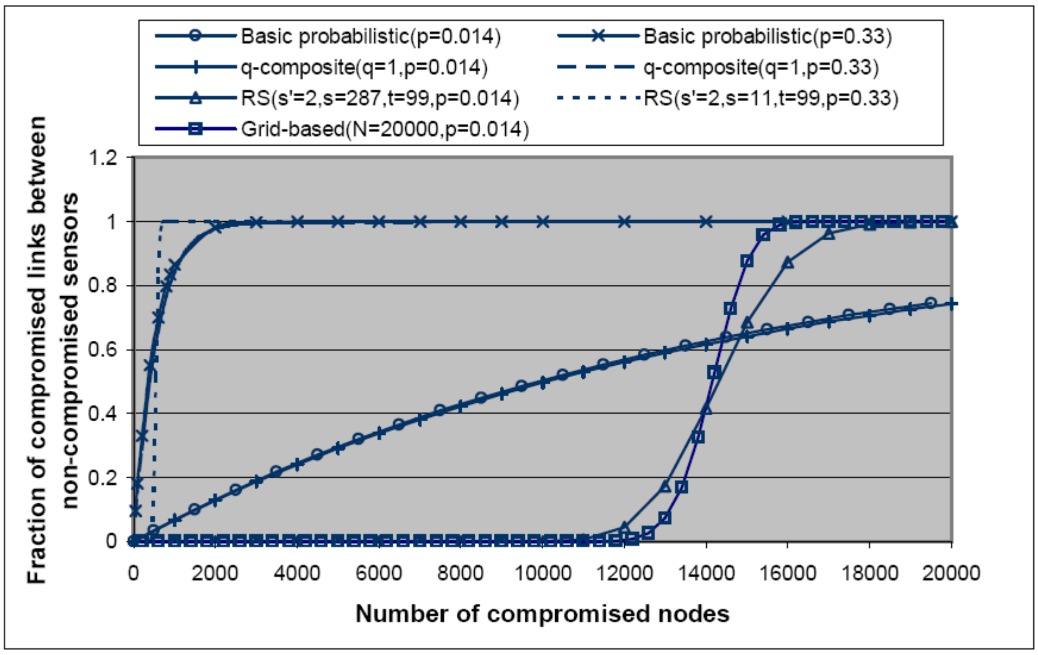


Figure 2.2: Node capture resilience of hypercube scheme, ring size equal to 200 keys. Figure taken from [34].

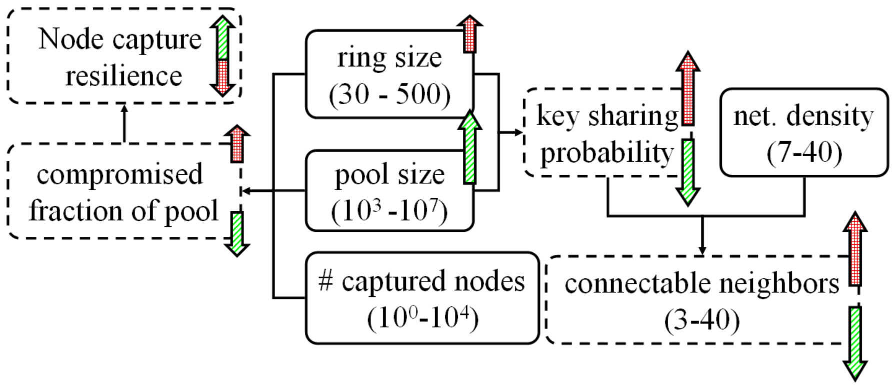

Node capture resilience
compromised fraction of pool
ring size (30 - 500)
pool size (10^3 - 10^7)
# captured nodes (10^0-10^4)
key sharing probability
net. density (7-40)
connectable neighbors (3-40)

Figure 2.3: Cohesion between key ring and key pool size. Particular settings of the network are depicted in the solid boxes, resulting properties in the dashed boxes. If network settings remain fixed, increase of the ring size increase the key sharing probability which in turn increase the number of connectable neighbours, but also increase compromised fraction of key pool when fixed number of nodes are captured thus decreasing overall node capture resilience. Increase of pool size influence the node capture resilience in exactly opposite way. Moderate increase of the key ring size allows to increase pool size highly, obtaining better node capture resilience as a result.

2.2.1 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 $K'_{ij} = H(K_{ID_{ij}}, R_{ij}, R_B)$. $(K'_{ij}, ID_{ij}, R_{ij})$ 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.[3] 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. The recovered key K_{AB} is then used to check the integrity of the original message M.
8. B returns to A the value $C_R = MAC_{K_{AB}}(H(R_B, R_{11}|R_{12}|\ldots R_{ij}))$ 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.

[3] 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 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.

2.2.2 Probabilistic authentication

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

1. Each entity is uniquely identifiable by its ID.
2. Each entity is linked to the keys with the identification publicly enumerable[4] 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 the potential number of neighbours in WSNs is high, the 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 the 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.

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

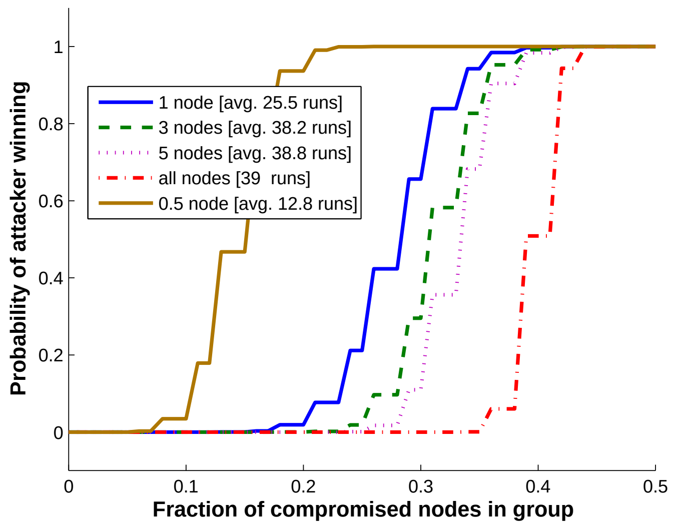


Figure 2.4: Probability of an attacker winning in majority decision voting and number of expected runs of the protocol w.r.t. number of nodes selected as central by each member of a group (39 nodes per group).

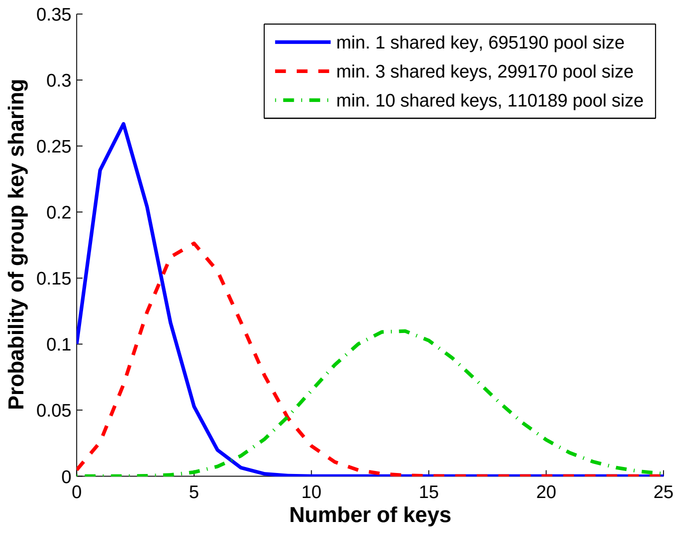


Figure 2.5: The distribution of probability of group key sharing (group size 40 nodes, ring size 200 keys, 90% constant probability of sharing at least required number of keys). Note that the pool size decreases with a growing minimum of required shared keys.

2.2.3 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 an authenticated channel to each of the requesting nodes.
3. Each node then separately applies the majority rule over p responses from its set, obtains the partial result M_i and then broadcasts 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 Figure 2.4 for the probability of the attacker's success for different values of p and of the compromised fraction of a group.

2.2.4 Evaluation of the communication and computation overhead

The 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). The 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 a 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 threshold T for the minimum number 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 the messages exchanged between A and 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[5].

[5] 4 * 16B nonces + 25B used keys + 16B K_{AB} + 16B $MAC_{K_{AB}}(M)$

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. The same holds for the number of decryptions and random number generations.

We would like to stress that the overhead is independent of the group size (a larger group allows for the setup of a higher threshold 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.

In the following paragraphs, we will examine success of group-supported protocols with two schemes for probabilistic pre-distribution.

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 the calculation of relevant probabilities. The results were experimentally verified by a series of simulations with 40000 nodes on our simulator described in Section 1.2.

2.3.1 Evaluation of node capture resilience improvements

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

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

Proof: The probability that a particular key will not appear in the 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

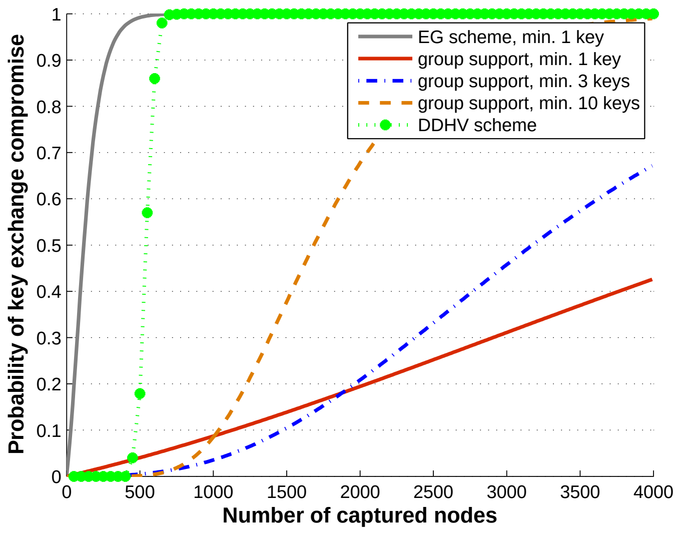


Figure 2.6: Node capture resilience of the basic EG scheme and the variant with group supported key exchange. Key ring size is fixed to 200 keys, group size to 40 nodes. The pool size differs with minimum of required shared keys to maintain a constant probability 90% ($probNoShare$) that the exchange will be possible.

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 are n such G_i nodes, thus $P_{keyNotShared\_1} = (\frac{S-m}{S})^n$. Then the probability that exactly k keys from the 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 the remaining $(m - k)$ of keys in the 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

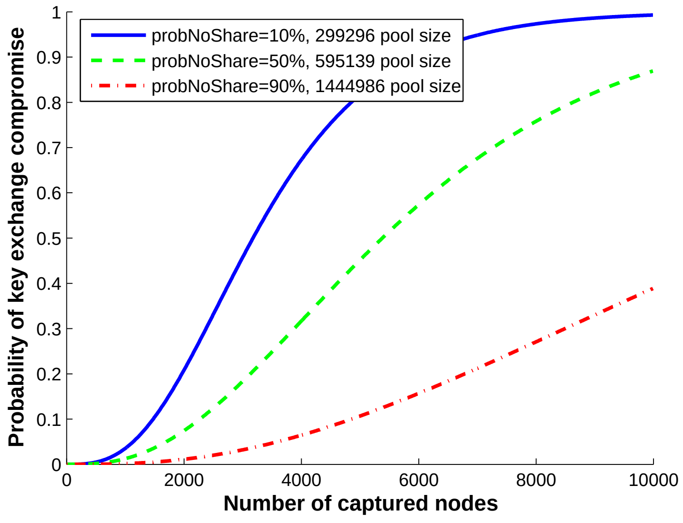


Figure 2.7: The impact of varying the probability of impossible key exchange to node capture resilience for minimum required keys T = 3. The network parameters are same as for Figure 2.6. The group enlarged to 160 nodes will decrease respective $probNoShare$ values to 0.0001% / 0.2% / 18.43%, keeping the pool size and node-capture resilience same.

a single node is fixed and given as a manufacturing parameter. More keys carried in a node generally imply better resilience for any PKPS.

We choose to fix this parameter to 200 keys to enable a comparison with the multi-space pairwise keys scheme [19] (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 Figure 2.6, 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 $probNoShare$ = 10% probability *not* to

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

Table 2.2: Decrease of the probability of impossible key exchange $probNoShare$ when the nodes two hops away are also 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.

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 Section 2.3.3 for details). Based on the usage scenario, $probNoShare$ can be set to a higher value, e.g. if a node can move to another location or if a fraction of the nodes can be "sacrificed" for the sake of better network resilience. This increases the pool size and thus consequently increases the node capture resilience. An example of the impact of a minimum of 3 required keys is shown on Figure 2.7.

A significant increase of NCR can be obtained when $probNoShare = 90\%$ is set. However, this means that in the basic version of the protocol, only 10% of new nodes will be able to join the group. This can be acceptable in the scenario with mobile nodes. Otherwise, group enlargement (see Section 2.3.3) can be performed.

We would like to stress that our protocols do not provide defense against Sybil [43] or collusion [40] attacks, where direct clones of the captured node are populated over the network. Here we rely on some replication detection algorithm like [45]. The design of such a protocol with high efficiency and reasonable communication overhead is still an open question.

2.3.3 Group enlargement

If a node is capable of remembering the 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 twofold due to the need for message routing (two hops instead of one). The total number of contacted nodes will remain the same and – due to the seed-based pre-distribution – no messages are required to compute IDs of nodes that will be contacted. The group enlargement can be employed in the following scenarios:

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

The Table 2.2 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 Figure 2.7, 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 nodes that are actually sharing key with the incoming node B are contacted. An increase in the number of 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 consists of 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 the IDs of nodes two hops away. Note, that the IDs of direct neighbours are most probably stored for routing purposes anyway.

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

2.4 Group support with multi-space polynomial scheme

Basic evaluations of group supported protocol properties were provided, with the EG scheme as the underlaying pre-distribution scheme. Other pre-distribution schemes can be transparently used as well. Here we will discuss polynomial-based pre-distribution as introduced in [19]. Instead of assigning direct keys as in the EG scheme, we select Blom's key space from the 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 ring size the same, there can be up to $numberOfPolynomials = \lfloor m/c \rfloor$ polynomials, where m is the node ring size and c is the degree of Blom's polynomial. We choose to fix $c = t + 1$, where t is Blom's threshold security parameter to minimize the memory footprint of one polynomial. The limitation is that when using this setting, only up to c nodes can be assigned by different polynomial share from one polynomial space and this may limit the 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 the probability in the original multi-space polynomial scheme and thus the supported network size remains sufficient.

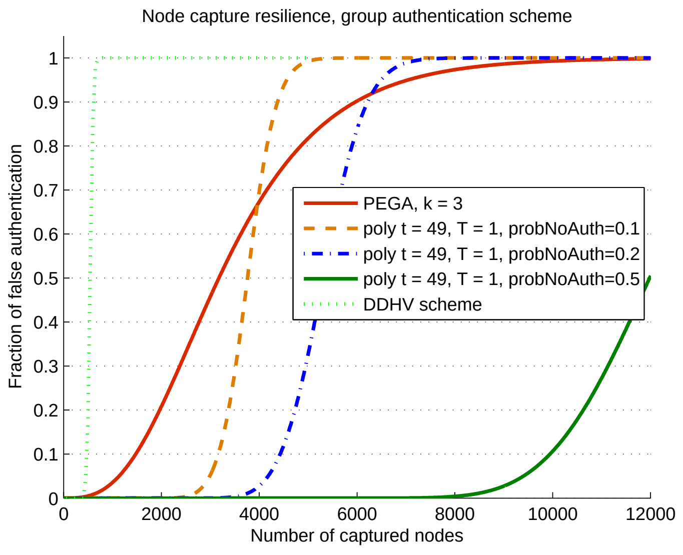


Figure 2.8: Node capture resilience of the group supported scheme with Blom's treshold secret sharing scheme as an underlaying pre-distribution. Group size 40 nodes. Ring size is equal to 200 keys.

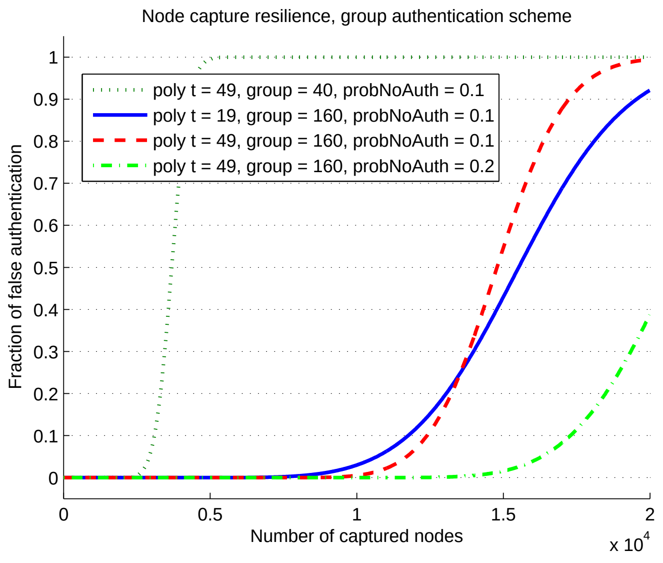


Figure 2.9: Node capture resilience of the group supported scheme with Blom's treshold secret sharing scheme as an underlaying pre-distribution. Group size 160 nodes. Ring size is equal to 200 keys.

2.4.1 Evaluation of node capture resilience improvements

Lemma 4: Probability of i shares compromise.
Probability that an 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: A particular polynomial cannot be compromised until an attacker has compromised at least $(t+1)$ shares of this polynomial (proof given in [8]), where t is pre-specified threshold. The probability that a given polynomial was chosen for a sensor node is $\frac{m'}{S}$. To compromise exactly i shares of this polynomial, the 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 in which the i compromised shares can be 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: This sum gives the probability that an attacker will compromise less than or equal to t shares from a particular polynomial. If the attacker compromises more than t shares, then the particular polynomial is compromised and this must hold for every one of the k polynomials.

2.4.2 Impact of polynomial security threshold

Impact of different degree of the polynomials is shown of Figure 2.10 (basic group with 40 neighbours) and Figure 2.11 (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 the number of stored polynomials on every node are set to fit this memory restriction as $numberOfPolynomials = \lfloor 200/c \rfloor$. Tested polynomial degrees were 5 (40 polynomials per node), 10 (20), 20 (10), 40 (5), 50 (4) and 66 (3). Larger degrees of polynomial were not tested as the resulting node capture resilience does not increase for our settings.

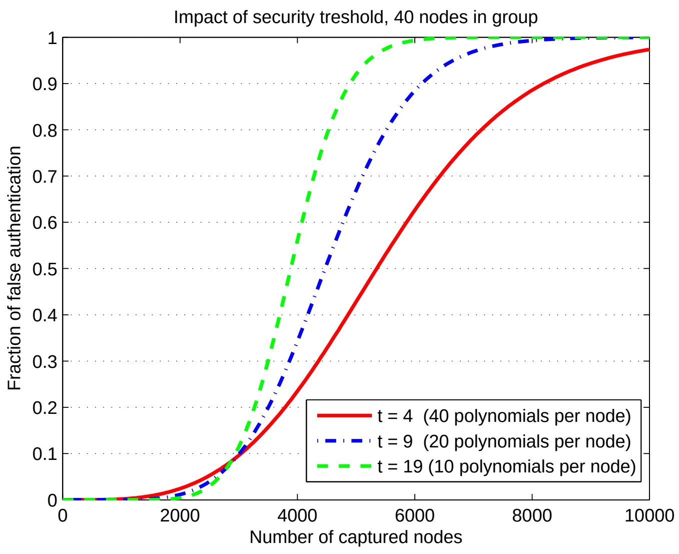


Figure 2.10: Impact of number of polynomial security threshold. Group size is 40 nodes. Ring size is equal to 200 keys.

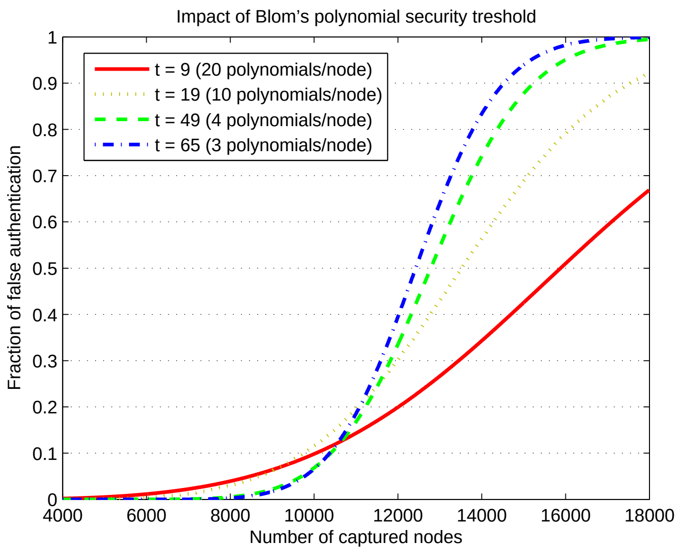


Figure 2.11: Impact of number of polynomial security threshold. Group size is 160 nodes. Ring size is equal to 200 keys.

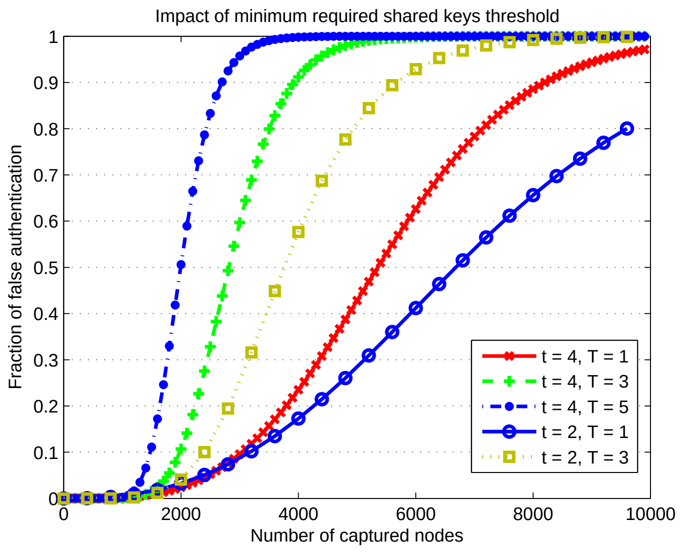


Figure 2.12: Impact of number of minimum required keys during the group support protocol. Group size is 40 nodes. Ring size is equal to 200 keys.

2.4.3 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 2.12 (basic group with 40 neighbours) and Figure 2.13 (enlarged group with 160 neighbours). Increasing the 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. An increased threshold of minimal required shared keys (more than one) thus implies a significant decrease of the pool size to maintain fixed probability that key exchange will be possible. The resulting node capture resilience against an attacker randomly capturing the nodes is weakened. However, this threshold increases the resiliency against an attacker who tries to find the part of the network where he is able to introduce malicious node with forged ID. With increased threshold there is a higher probability that the group will know at least one key connected to this forged ID, but unknown to an attacker. The threshold thus should be set according to expected threats to particular network. The analysis shows that a lower degree of polynomial is better for higher number of minimal required shared keys (see Figure 2.12).

2.4.4 Comparison with hypercube scheme

Direct comparison between group supported and hypercube scheme [34] is not really possible, as both schemes use different assumptions and resulting node capture resilience depends on several input variables used for the analysis.

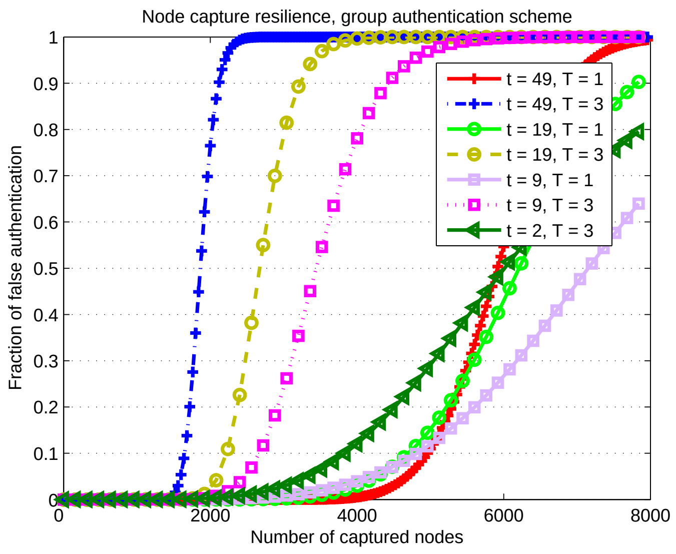


Figure 2.13: Impact of number of minimum required keys during the group support protocol. Group size is 160 nodes. Ring size is equal to 200 keys.

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

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 node capture resilience indicated in original paper. Usage of all paths is possible, but results in a very high communication overhead. The scheme has low storage overhead, as the ID of nodes necessary to form key establishment path can be computed from ID of source and target node. However, the storage of status of compromised links implies an additional overhead.

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

2.5 Possible attacks and defenses

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

2.5.1 Passive attacker

A passive attacker can either randomly or selectively [26] 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 2.1.5, 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, a group of nodes will have a very high probability of sharing at least one key that the attacker does not know and thus also a very high probability of detecting cheating.

3. An attacker will try to generate a random forged ID, for which he knows most of the keys (for feasibility evaluation – see Section 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) The use of a fresh random identifier instead of the original ID as described in Section 2.1.5 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.

2.5.2 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 in 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.

2.6 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 [21] and polynomial scheme by [19] show that a substantial node capture resilience can be obtained. Probabilistic key pre-distribution schemes generally exhibit the property that the probability of key sharing rapidly increases with the number of keys in a node's key ring. Our group supported protocol exploits this behaviour to create large virtual key rings. However, this improvement does not come for free. Firstly, some additional communication is required. We have shown that substantial improvements in the node capture resiliency can be obtained with a reasonable communication overhead that is proportional to the minimum of required shared keys (security parameter) rather than the size of supporting group. Secondly, a node relies on the security of previously established link keys with its neighbours. This opens a possibility for additional attacks, which were discussed together with possible countermeasures.

Combination of the group supported protocol with multi-space polynomial pre-distribution provides a better node capture resiliency if the polynomial scheme can be efficiently computed on the nodes. See [34] for discussion of efficient implementation.

Chapter 3

Secrecy amplification

The uncertainty about the identity of direct neighbours prior to the actual deployment naturally results in such a property of the key distribution schemes that most of the nodes in the network should be able to establish a shared key (either directly or with support from other nodes). This alone is a serious challenge for memory- and battery-limited nodes. At the same time, this property implies one of the main problems for maintaining a secure network in presence of an adversary with the ability to compromise link keys (e.g., via node capture or eavesdropping). If 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, the 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 a copy of the list). A common way and good practice to introduce localized secrets is to not use pre-distributed keys for ordinary communication, but only to secure the key exchange of fresh keys, which are locally generated only by nodes involved in the 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 of other (potentially compromised) nodes, the secrecy of the resulting key can be violated. To tackle this threat, secrecy amplification protocols were proposed.

This chapter starts with discussion of the principles, properties and performance results of several secrecy amplification protocols including those proposed by us for a partially compromised network resulting from plaintext key distribution known as Key Infection [3]. We will provide detailed comparison is based on simulations resulting from an S3 simulator (see Section 1.2 for description). Later we discuss the applicability of secrecy amplification protocols to different types of partially compromised networks resulting from a node capture in probabilistic key pre-distribution. The differences between characteristics of compromise (so-called compromise patterns) are described, and the impact of these differences on the success of secrecy amplification protocols is examined. To cover additional compromise patterns, we turn from manual design of the protocols and focus on an automatic generation of such protocols. We are using a combination of evolutionary algorithms that generate candidate solutions and a network simulator that evaluates them. Such an approach enables us to find personalized protocols that work well for a particular key distribution method and against a given attacker and his tactics, avoiding unnecessary messages and thus significantly reducing the communication overhead and making secrecy amplification protocols more practical. An automated approach helps us to find the new protocol with a better fraction of secured links than all published. Finally, we propose a novel principle of secrecy amplification protocols design. This design exhibits linear instead of exponential increase of protocol messages with increasing network density. An automated approach was used to design new protocols with a comparable fraction of secured links to the original (message expensive) approach.

Analysis of the Pull protocol is based on our paper [16], but is significantly rewritten to incorporate information from additional experiments and to better fit with the remaining chapter text. Initial results for automatic generation of secrecy amplification protocols were published in [57] and is accepted for publication as a chapter in book [50].

3.1 Related work

Secrecy amplification protocols (also known as privacy amplification protocols) were introduced by [3] for weaker attacker model together with the plaintext key exchange as a lightweight key establishment method (so-called Key Infection) for wireless sensor networks. This approach does not require any pre-distributed keys. Nodes are simply distributed over the target deployment plane and perform discovery of neighbours by radio. Every node then generates a separate random key for its each neighbour and sends it *un-encrypted* (as no keys are pre-shared) over a radio link. This key is then used as a basic link key to encrypt subsequent communication. This scheme has minimal memory requirements, requires no pre-distribution operations in a trusted environment and every node can essentially establish a key with any other nodes. Perfect node capture resilience is achieved as any two nodes share different keys. If no attacker with eavesdropping capability is present during such plaintext key exchange then all keys will be secure. On the other side, if an attacker is able to eavesdrop all plaintext exchanges then the scheme will be completely insecure as all keys will be compromised. A modified attacker model based on the assumption that the attacker has limited material/financial resources is used instead. Basic assumption is that not all links are eavesdropped.

The first assumption of the model is the similarity of devices used as network nodes and eavesdropping nodes, especially radio sensitivity. We will reason about this assumption a bit: The higher sensitivity of the radio implies higher energy consumption and eavesdropping nodes will require stronger battery sources implying an increased attack cost. Additionally, the relatively small radio range of legal nodes enables frequent radio channel reuse. An eavesdropping node with a highly sensitive antenna will receive signals from several parallel transmissions on the same channel, rendering the received cumulated signal unreadable. The assumption of similar radio sensitivity is therefore reasonable.

3.1.1 Plaintext key exchange – whispering

Communication between two neighbours might be performed with full radio transmission power (we will call this transmission as *maximal screaming mode*). This mode of transmission is suboptimal both from energy-efficiency and security points of view. Current sensor platforms allow us to control the transmission power to some degree to save on node battery, and this feature can be used to facilitate plaintext key exchange which can be eavesdropped only in a limited area.

We will use the term *whispering* for this message transmission mode between two nodes that is performed with the minimal transmission power necessary to communicate. If the sending node is using lower power than this minimal value, then the receiving node is not able to receive messages successfully with its own antenna. The minimal power strength can be obtained from the following process: node starts sending a hello message with the minimal possible power. If no response is received within a defined time-frame, then power is repeatedly increased by small steps until the particular neighbour can hear the transmission and responds.

This minimal transmission power is used later to exchange the link key in plaintext. An attacker will compromise a link key when he is able to record this key exchange transmission. If the eavesdropping device has the same quality of receiver as legal nodes (antenna, signal amplification) then the eavesdropping device must be positioned at equal or smaller distance to the sending node from the receiving node to eavesdrop transmission (if signal propagation is an ideal sphere). This assumption will be used for simulations of secrecy amplification protocols.

As the plaintext key exchange takes place immediately after network deployment, the attacker's eavesdropping devices must be present from the very beginning, actually placed in the deployment field *before* the deployment of the network. If the exact deployment field is not known in advance, the attacker must cover larger areas by its eavesdropping nodes than the owner of the network. The ratio between legal nodes and attacker's eavesdropping nodes will be then unbalanced toward a higher number of legal nodes. This forms the second assumption of the model.

In real deployment, several hello messages should be sent with the same transmission power, as wireless signal propagation might vary, transmission with minimum possible power is desirable, and several lost messages during the key exchange (short messages themselves) can be tolerated. Also, the key exchange can be actually in the hello message. If multiple messages are used, then a different random key should be used for each message to limit the time frame when a particular key value is in the air, possibly vulnerable to eavesdropping. The receiving node will respond with a hash of the key from the received message to confirm the exchange of the key.

In the Key Infection approach, a weakened attacker model is necessary for the first stage (plaintext key exchange) and in some cases also during secrecy amplification. The attacker then reverts to the mode where all transmissions are eavesdropped by an attacker. The length of this interval, together with the resilience of the used exchange and subsequent secrecy amplification, determines the cost for an attacker to successfully attack the network.

3.1.2 Basics of secrecy amplification protocols

A secrecy amplification protocol is an additional scheme executed by the nodes in the network after the basic link key establishment, plaintext key exchange in case of the Key Infection. Fresh new secrets are generated locally and distributed using existing links with associated security state (secure/compromised). As a result, new link keys are constructed. These are different from original pre-shared or exchanged secrets. Especially in WSNs, secrets usable only locally should be preferred due to the possibility of various Sybil-like attacks. Moreover, some links can be secured, even when the original link was compromised.

What is commonly unknown to the network nodes is the identity of links that are actually compromised. Still, we can execute the amplification protocol as the second layer of defence, 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 $K'_{AB} = H(K_{AB}, K)$, where K_{AB} is the original link key, K is a fresh key exchanged during amplification protocol and H is a 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 the 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. The security of link keys can be further improved as links newly secured in the previous iteration can help to secure a new link in the next iteration.

There is no difference between passive and active attackers for the secrecy amplification protocol with respect to the number of secured links. An active attacker controlling a node is equivalent to a passive one that has compromised all links to the node, thus intercepting all passing messages. 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 the 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 inverse attack must be considered as well as two compromised nodes may blame a legal node for providing an incorrect key. A link jammed by an adversary is equivalent to a missing connection rendering path unusable for secrecy amplification.

Secrecy amplification protocols can be categorized based on:

Number of distinct paths used to send parts of the final key – if more than one path is used then the protocol performs so-called *multi-path amplification*. An attacker must eavesdrop all paths to compromise the new key value. If two nodes A and B exchange a new key directly in one piece, then only one path is used. Note that multiple virtual paths can be constructed over one physical path [56].

Number of involved intermediate nodes per single path – basic key exchange between A and B requires no intermediate node. If at least one intermediate node is used then the protocol performs so-called *multi-hop amplification*. The path is compromised if an attacker is able to eavesdrop at least one link on the path.

Notation	Description
$\vert$	concatenation operator
A, B	identification of nodes for which link key is strengthened during secrecy amplification
C_i	identification of intermediate node(s) used during secrecy amplification
N_C	identification of central node during group-oriented secrecy amplification protocols
N_P	identification of node with special role during group-oriented secrecy amplification protocols
N_{d1_d2}	distance relative identification of a node with distance $d1$ from N_C and $d2$ from N_P
k_{xy}	key exchanged in plaintext from node x to node y
K_{xy}	pairwise key shared between nodes x and y
K'_{xy}	new pairwise key shared between nodes x and y after secrecy amplification
$H(.)$	application of cryptographical strong one-way hash function H
$E_{K_{xy}}(.)$	symmetric encryption function using key K_{xy}

Table 3.1: Notation used for secrecy amplification protocols. Function H should be realized as HMAC construction [6] instead of simple hash function to prevent extension attacks.

3.1.3 The mutual whispering protocol

The simplest secrecy amplification protocol is based on a combination of keys exchanged between two nodes. Mutual whispering secrecy amplification constructs the new key between A and B simply as $K'_{AB} = H(k_{AB}, k_{BA}, K_{AB})$, where k_{AB} is the key exchanged (whispered) from A to B, k_{BA} from B to A, and K_{AB} is an already existing shared key for the link between A and B if such key exists, otherwise only keys k_{AB} and k_{BA} are combined. This protocol was not explicitly mentioned in [3], but was actually used for simulations there (based on provided source code). Mutual whispering is a one-hop two-path protocol – no intermediate node is used and keys from two paths are combined (from A to B and from B to A – these paths overlap).

$$A \to B : (A, B, k_{AB})$$
$$B \to A : (B, A, k_{BA})$$
$$A, B \text{ compute } K'_{AB} = H(k_{AB}|k_{BA}|K_{AB})$$

Table 3.2: Message diagram for mutual whispering protocol.

3.1.4 The Push protocol

The multi-hop (two-hop) and multi-path (number of neighbours reachable from both A and B) secrecy amplification protocol was described in [3]. Node A generates q different random values (key parts) and sends each one along a different path over an intermediate node(s) C_i to node B, encrypted with an existing link key(s). All values combined together with the existing key shared between A and B are used to create the new key value. If at least one path is not compromised, the resulting key will be secure. Simulations in [3] for attacker/legal nodes ratio up to 5% are presented, showing that the plaintext key exchange followed by the Push protocol is suitable within this attacker model.

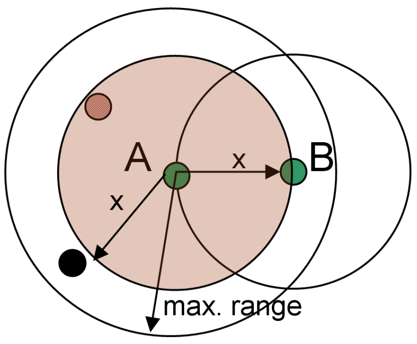


Figure 3.1: Minimal transmission power used during whispering between (green) nodes A and B. Attacker's node positioned inside the inner circle (red node) is able to eavesdrop the transmission between A and B whereas node positioned outside (black node) is not.

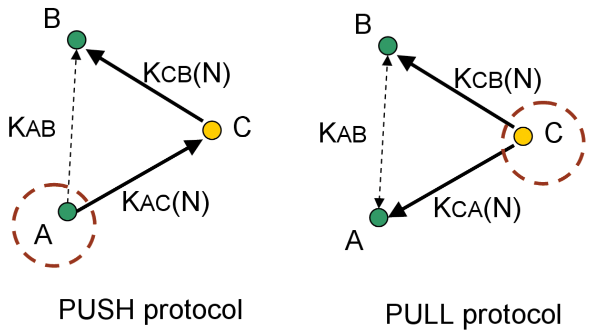


Figure 3.2: Graphic representation of the simplest version of Push and Pull amplification protocols with only one intermediate node C. Red dashed circle highlights the node generating a fresh new random secret N. K_{xy} is the existing directional key between nodes x and y.

$$\begin{aligned}
&A \rightarrow C_i : E_{K_{AC_i}}(A, B, N_i)\\
&C_i \rightarrow B : E_{K_{C_iB}}(A, B, N_i)\\
&A, B \text{ compute } K'_{AB} = H(K_{AB}|N_i)
\end{aligned}$$

Table 3.3: Message diagram for two-hop version of the Push protocol.

3.1.5 The Commodity protocol

A variant of initial key exchange mixed with the Push protocol (will be denoted as Commodity) without explicit secrecy amplification is presented in [29]. Node A sends the same key k_i to nodes B and C_i in plaintext using whispering. Then k_i is used to secure distribution of initial key material $E_{k_i}(A, B, K_{i2})$ between (A, B), $E_{k_i}(C_i, A, K_{i3})$ between (C_i, A) and $E_{k_i}(C_i, B, K_{i3})$ between (C_i, B). The final key shared between (A, B) is constructed as $K_{AB} = H(K_{i3}, H(K_{i2}, k_i))$. Formal security proof of the proposed scheme is presented in the paper. The fraction of secured links will be lower than for the Push protocol as the transmission of initial exchange is performed with a higher transmission power (maximum of transmissions required to reach both B and C from A) and therefore is more likely to be compromised. We exclude the Commodity protocol from more detailed analysis, as it is only a variant of the Push protocol, does not provide secrecy amplification as separate and the fraction of secure links will be lower than for the Push protocol alone.

3.1.6 The Pull protocol

A variant of the Push protocol called Pull protocol is presented in our work [16]. The initial key exchange is same as for the Push protocol, but node C_i generates fresh secrets which

$$\begin{array}{l}
A \rightarrow B : (A, B, C_i, k_i) \\
A \rightarrow C_i : (A, B, C_i, k_i) \\
A \rightarrow B : E_{k_i}(A, B, K_{i2}) \\
C_i \rightarrow A : E_{k_i}(C_i, A, K_{i3}) \\
C_i \rightarrow B : E_{k_i}(C_i, B, K_{i3}) \\
A, B \text{ compute } K_{AB} = H(K_{i3}, H(K_{i2}, k_i))
\end{array}$$

Table 3.4: Message diagram for two-hop version of the Commodity protocol.

are used to improve the secrecy of the key shared between nodes A and B instead of node A as in the Push protocol. The basic idea is that the area where eavesdropping nodes must be positioned to successfully compromise the link key is smaller than for the Push protocol. The resulting fraction of compromised keys is then lower as an attacker has a smaller chance to place eavesdropping nodes properly.

$$\begin{array}{l}
C_i \rightarrow A : E_{K_{C_iA}}(A, B, N_i) \\
C_i \rightarrow B : E_{K_{C_iB}}(A, B, N_i) \\
A, B \text{ compute } K'_{AB} = H(K_{AB}|N_i)
\end{array}$$

Table 3.5: Message diagram for two-hop version of the Pull protocol.

3.2 Analysis of secrecy amplification protocols

Previous work [3] and [29] dedicated little attention to the behaviour of proposed protocols for different network densities, networks with a higher number of eavesdropping nodes or repeated iterations of amplification. Simulations were performed only for small sizes of the network. Work [29] provided no analysis of the fraction of secured links at all.

We focused on the development of an optimized simulator capable of simulating networks up to hundreds thousands of nodes with variable deployment field size, network density, number of eavesdropping nodes and repetition of amplification process. The simulator aims to simulate such networks in reasonable time to provide more detailed performance estimation, used also for our Pull protocol. See Section 1.2 for simulator details.

The initial results of our simulator for the Push protocol had the same dynamics, but the absolute numbers were different from the original results presented in [3] quite substantially – up to 50-100% of the original results. We asked authors for their original simulator and inspected the source code. We found that their implementation is correct, but they use a mesh with very small resolution to position the nodes and the total number of nodes used was also quite low. When we increased the implicit number of nodes in their simulator, the results varied in tens of percent from the results presented in the original paper. The results presented in this section come from our simulator.

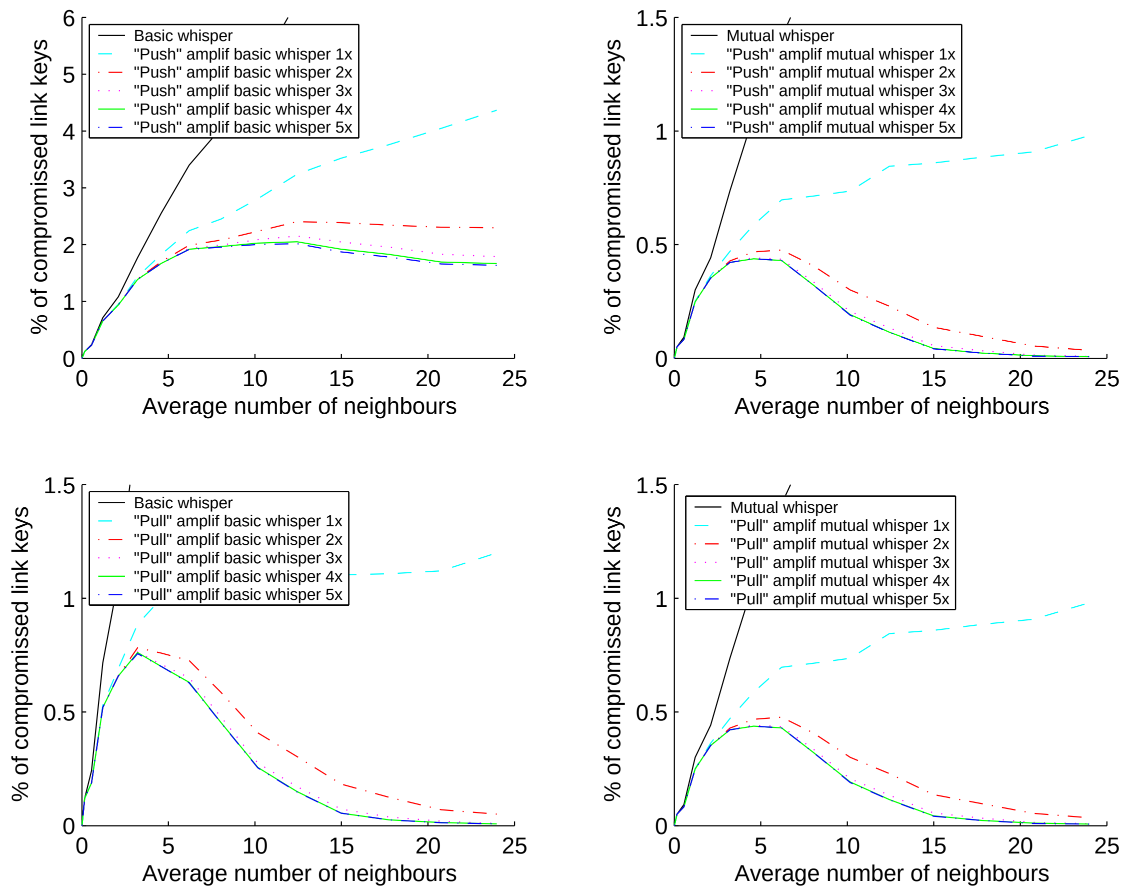


Figure 3.3: Fractions of compromised link keys with 1% of eavesdropping nodes for multiple iteration of the protocols. The Push protocol applied over basic whispering (upper left) and over mutual whispering (upper right) is shown. The Pull protocol applied over basic whispering (lower left) and over mutual whispering (lower right) is shown.

3.2.1 Network settings and simulation setup

Legal and attacker nodes are randomly distributed over a pre-defined area. The neighbour discovery phase is performed for each legal node based on its transmission range. Attacker nodes act just as passive communication eavesdroppers – they represent a passive adversary, but they immediately share all information eavesdropped by any of them so that they can instantly combine key values sent over different paths.

We used network sizes between 10^4 to 10^5 of legal nodes deployed over a square plane with side equal to 25 distance units. Actual variation of average number of neighbours was achieved by changing the transmission range for legal nodes (values between 0.1 and 0.5 were used).

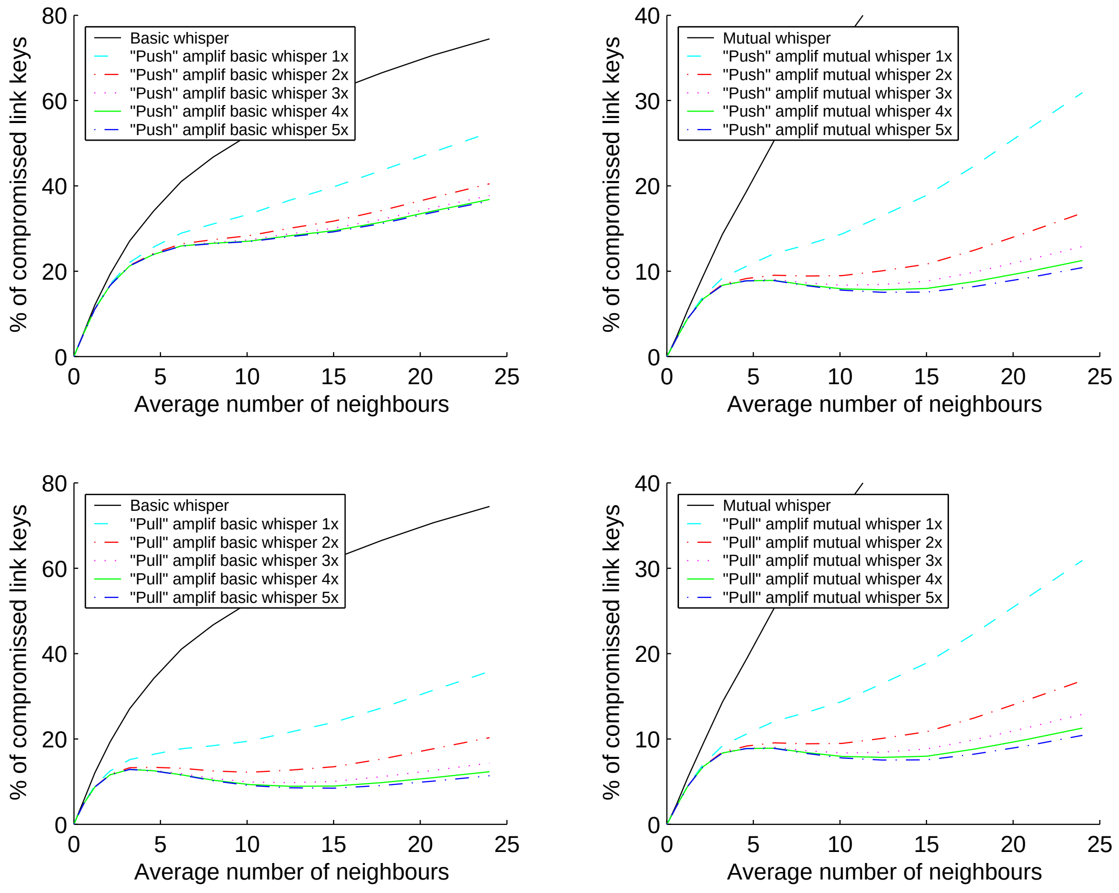


Figure 3.4: Fractions of compromised link keys with 20% of eavesdropping nodes. Results are presented for Push, Pull and mutual whispering protocols, executed after the initial key exchange (basic whisper) or after the initial key exchange with the mutual whisper amplification applied first. The Push protocol applied over basic whispering (upper left) and over mutual whispering (upper right) is shown. The Pull protocol applied over basic whispering (lower left) and over mutual whispering (lower right) is shown.

The simulations were performed with an increasing density of the networks and the resulting graphs are averaged results from at least five distinct simulation runs (note that the high number of nodes in the network and a large deployment plane provides reasonable independence of simulation results from a particular placement of the nodes – therefore five to ten simulations with different nodes layout are sufficient to provide a reasonable average).

3.2.2 Discussion of simulation results

The set of graphs presented in Figures 3.3, and 3.4provides simulation results for all inspected protocols and some selected combinations. Dependency of the fraction of secured links on network density and number of eavesdropping nodes are inspected. Performance of maximum screaming and whispering alone is shown and serves as a baseline (number of secure links if no secrecy amplification protocol is executed). The second set of results is generated from

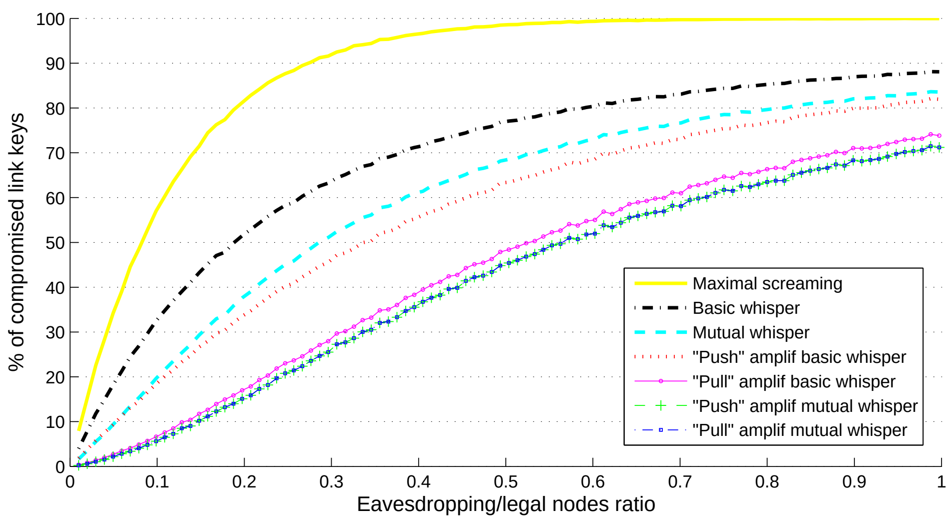


Figure 3.5: Fraction of compromised keys with increasing fraction of eavesdropping nodes. Network density for legal nodes is 8 neighbours on average.

execution of mutual whispering with both Push and Pull protocols over network. Finally, the execution of Push and Pull protocols over network with already performed mutual whispering is examined. Figures 3.5 and 3.2.2 show the behaviour of the protocols with the increasing number of eavesdropping nodes for two network densities with eight and fifteen neighbours respectively on average.

The first set of graphs (Figure 3.3) shows results from a network of 10^4 nodes with 1% of eavesdropping nodes (i.e., there are 100 eavesdropping nodes).

The amplification results are naturally getting worse with an increasing number of eavesdropping nodes in the network as more links are compromised during the initial plaintext key exchange. The second set of graphs (Figure 3.4) shows results for the same settings (10^4 of legal nodes), but with a significantly increased number of eavesdropping nodes at 20% (i.e., there are 2000 eavesdropping nodes).

Based on simulation results for these two scenarios, following findings were observed:

Repetition of secrecy amplification iterations significantly increases the total number of secure links, especially for the lower rates of eavesdropping nodes. The results presented in [3] were done for only one iteration of the amplification protocol. As new links are secured in the first iteration, following iterations have a better starting position than the first one (more secure links). Simulations showed that the increase of secure links can be very significant, decreasing the number of compromised links effectively to zero for scenarios with a low attacker presence and a dense enough network (e.g., 15 neighbours on average). Significant improvement by repetition is possible also for a strong attacker presence (20%).

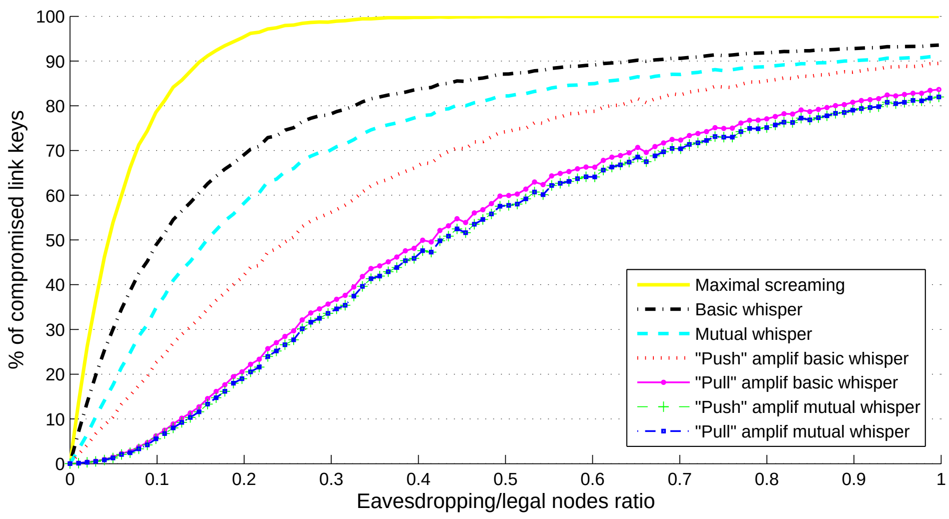


Figure 3.6: Fraction of compromised keys with increasing fraction of eavesdropping nodes. Network density for legal nodes is 15.2 neighbours on average.

Reasonable number of repetitions within tested scenarios is between two and four as additional repetitions do not provide significant increase. The Figures 3.3 and 3.4 show results for up to five iterations of amplification protocols, where only the last two iterations provide a small improvement.

Success of amplification fluctuates with network density. The actual number of links secured by the secrecy amplification protocol as well as the relative factor of improvement (rate between number of secure links before and after amplification) depends heavily on the network density and may fluctuates.

This behaviour is caused by interplay between two factors – a) amplification protocols generally work better with a higher density of the network; b) the attacker will initially compromise a higher fraction of links in dense networks (regardless of the number of eavesdropping nodes). The baseline number of compromised links after the plaintext key exchange (no amplification applied yet) increases quickly with the increasing density of network.

For very sparse networks (up to five nodes in transmission range on average), protocols requiring intermediate nodes often do not find such intermediates between two original nodes and only factor b) applies and so the fraction of compromised links is increasing. With a moderate density of network, amplification protocols have enough intermediates and factor a) is dominating over b), causing a decrease in compromised links. For very dense networks (more than 20 neighbours on average), factor b) will eventually overweight factor a) gain, if enough eavesdropping nodes are present. For a low attacker presence (e.g. 1% of eavesdropping nodes), factor b) never overweight a). For scenarios with a significant attacker presence (e.g., 20%), increasing network density after some threshold does not help to decrease the fraction of compromised links. Such situation is best seen in the graph of the Pull protocol with basic whispering on Figures 3.3 and 3.4.

The Pull protocol outperforms the Push protocol when whispering is used. One iteration of the Pull protocol provides significantly more secure links than one iteration of the Push protocol (factor of two for sparse networks up to factor of four for dense networks). Multiple iterations work significantly better with the Pull protocol than for the Push protocol, especially for 1% eavesdropping nodes. The threshold point of network density where the number of compromised links starts to decrease is also lower (around five for the Pull protocol and around eight for the Push protocol).

Combination of Mutual whispering with other amplification protocol might increase the number of secured links. Combination of the Push protocol with mutual whispering provides exactly the same fraction of secured keys as the Pull protocol alone. The reason comes from the geographical distribution of areas where eavesdropping nodes must be positioned to successfully compromise the link key during amplification. Intersection of compromise areas (area where the attacker eavesdrops a key if positioned inside) for mutual whispering and the Push protocol is exactly same as for the Pull protocol alone. Consequently, amplification success is same as well. This behaviour introduces the idea of efficient composition (stacking) of several secrecy amplification protocols – we will discuss this issue later in Section 3.5.

Mutual whispering is better than Push/Pull protocols alone for sparse networks. Mutual whispering requires no intermediate node and therefore is not impacted by the possibility of unreachable intermediates, which might happen frequently for sparse networks. The combination of a protocol without intermediates (e.g., mutual whispering) with a protocol that uses intermediates (e.g., the Pull protocol) should be generally preferred.

3.2.3 Transmission overhead

The number of messages necessary to execute Push and Pull protocols is same. Also the probability that new key K'_{AB} can be established (not necessarily a secure key) using mediator C_i remains the same as both protocols use at least one intermediate node. Intermediate node C_i is unusable for amplification when no path from A to B over C_i exists. Specifically for the two-hop version of Push and Pull protocols (one intermediate node), C_i cannot be used if C_i is not neighbour of A and B at the same time. This situation can occur with the same probability for both Push and Pull protocols as node identities for the Push protocol can be viewed as permutation of the Pull protocol. Similar situation holds for multi-hop versions of Push and Pull protocols.

Further improvement is possible for the two-hop Pull protocol if the eavesdropping node density is assumed to remain same from the initial plaintext key exchange between neighbours to the amplification phase as well. Messages exchanged during the amplification phase then do not need to be encrypted, as each legal node transmits messages with exactly the same strength as for the initial key exchange. Intermediate C_i can simply transmit the value N_i using one transmission with strength equal to the stronger value used for A or B, preserving the same compromise ratio. If the attacker is able to receive a stronger signal, the new key will be compromised anyway as the attacker already has one of the underlying link key k_{C_iA}

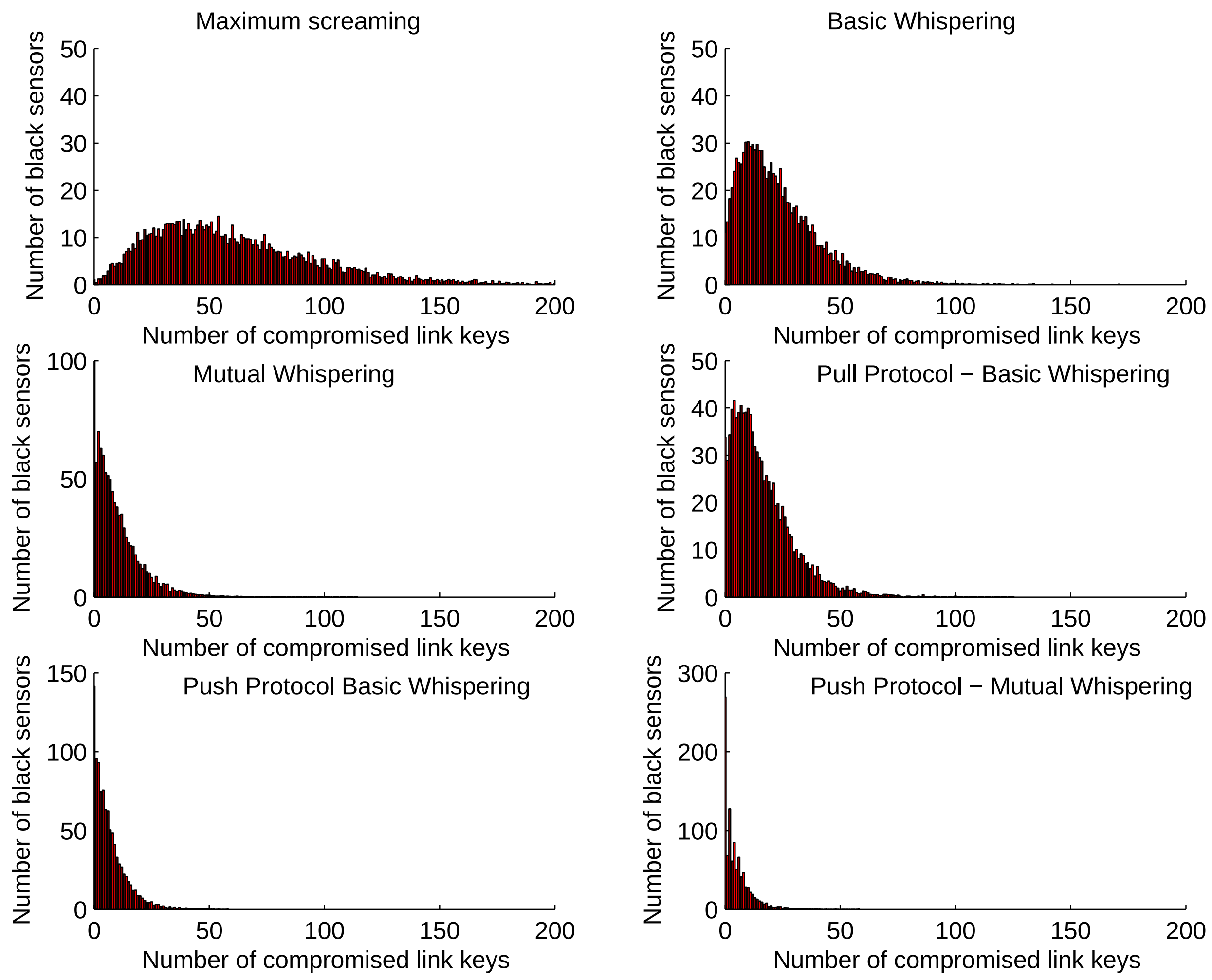


Figure 3.7: Distribution of eavesdropping node success rates.

or k_{C_iB}. One message exchange is thus spared, resulting in a 50% decrease in the number of total messages in case of the two-hop Pull protocol. Note that this optimization cannot be used for multi-hop version of the Pull protocol with more than one intermediate.

3.2.4 Compromise success for eavesdropping nodes

The set of graphs on Figure 3.7 shows the number of eavesdropping nodes, which were able to compromise a particular number of link keys for different secrecy amplification protocols. The results are generated from networks of 10^5 legal nodes with 1000 (1%) eavesdropping nodes. Every variation of the protocols we have been studying is provided in a separate graph.

The first graph covers the situation when keys are sent in clear with the maximum transmission power. The success rate of compromised keys corresponds to a Poisson distribution. Basic whispering shifts the mean value strongly towards a low number of compromised link keys per eavesdropping node, but still almost all eavesdropping nodes are usually responsible for the compromise of several link keys.

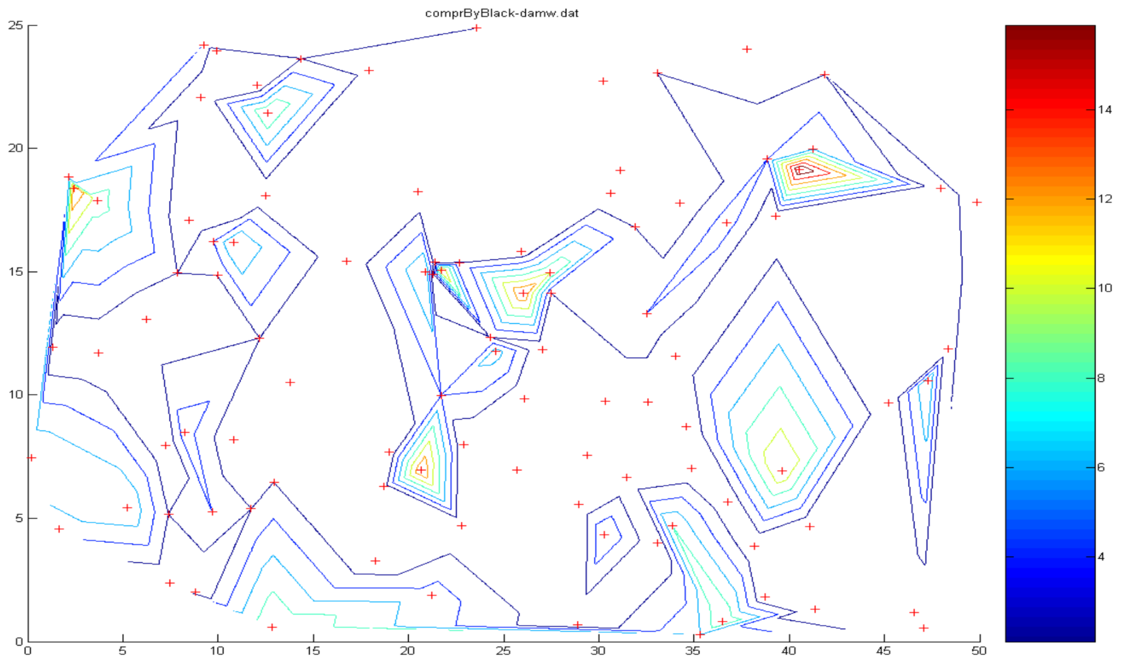


Figure 3.8: Example of non-uniform distribution of the link key compromise for the Pull protocol. Two-dimensional deployment plane is displayed with number of compromised link keys as third axis. Red crosses show position of the eavesdropping nodes. Legal nodes are not shown for the clarity reasons.

Secrecy amplification protocols have a notable impact. A large fraction of eavesdropping nodes is not able to compromise a single key, and the number of really successful eavesdropping nodes (nodes responsible for compromise of a large number of link keys) is rapidly decreasing with the value of compromised link keys. The last graph shows the amplification protocol combined with mutual whispering (Push and Pull protocols have the same results for such combination) where about 300 out of 1000 eavesdropping nodes are not able to eavesdrop a single key. The uneven distribution of compromised links also indicates that there might be large areas in the network without compromised link keys. Visualization of eavesdropping nodes layout together with the number of compromised links on Figure 3.8 provide an example of such non-uniform compromise for the Pull protocol. Such non-uniformity can be exploited to further increase the number of secure links and provides a background for the inspection of different compromise patterns as we later discuss in Section 3.4.

3.3 Key Infection analysis conclusions

One of the goals of the work was to verify simulations from [3] and to provide detailed information about protocols behaviour for the various network parameters. We believe that the results we introduced confirm very good resistance of amplification protocols against a local adversary. The presented results provided an insight in the published protocols and introduced a new secrecy amplification protocol with better faction of secured links.

The most important findings can be summarized as follows. The secrecy amplification generally works better with denser networks, but one cannot improve the ratio of secure keys with density over certain threshold when a certain density and attacker nodes fraction threshold was reached. Multiple iterations of secrecy amplification protocols provide a significant increase in the fraction of secure links with about three repetitions being reasonable. The combination of protocols is possible and a proper combination improves the number of secured links. The combination of the Push protocol with mutual whispering provides same results as the Pull protocol alone for network densities except for sparse networks. Sparse networks should combine mutual whispering with multi-paths/hops protocols to prevent situation with missing intermediate nodes.

Comparing Push, Push and Commodity protocols, Commodity requires the shortest period of the weakened attacker model (transmission of only one key k), but k can be intercepted from a larger distance than keys in Push and Pull protocols. An additional problem exploitable by an attacker is such that an intermediate node C_i knows the value of key between A and B. The Push protocol results in a significantly lower number of compromised link keys than the Pull protocol, especially for denser networks (more than 15 neighbour average).

Secrecy amplification protocols can make a network almost completely secure, when 1% of eavesdropping nodes is assumed. Even when 20% of eavesdropping nodes are present and each legal node has two eavesdropping nodes in transmission range on average, there are still 90% of link keys secure.

Secrecy amplification protocols were originally introduced for the Key Infection plaintext key exchange, but can be used also for a partially compromised network resulting from a node capture for the probabilistic pre-distribution and other partially compromised networks. This idea is later described in Section 3.5.

3.4 Compromise patterns of key distribution

Different key distribution schemes behave differently when the network is under attack targeted to disturb a link key security. The impact on link key security differs based on the attack strategy used. In case of node capture, all links to captured node are compromised. If some probabilistic pre-distribution scheme like [21, 10, 19] is used then some additional links between non-compromised nodes become compromised as well. An eavesdropping of the exchanged key in the Key Infection approach [3] does not compromises nodes directly, but compromises links in reach of eavesdropper radio instead. The characteristics of a particular compromise pattern may significantly influence the success rate of the secrecy amplification executed later. We will focus on two types of network compromise patterns – Random compromise pattern and highly correlated Key Infection pattern.

3.4.1 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 a 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.

3.4.2 Key Infection compromise pattern

Compromised networks resulting from Key Infection key distribution [3] 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.

The impact of Push, Pull, mutual whispering and new automatically derived protocols (as described in Sections 3.6 and 3.7) for Random and Key Infect compromise patterns are compared in Figure 3.12 and 3.13. The Pull protocol provides better results than the 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 Section 3.5.2 for a more detailed comparison of the protocols and the impact of repeated runs of secrecy amplification.

This short survey should demonstrate that amplification protocols may significantly increase the fraction of secure links (e.g., from only 50% secure to more than 90% secure) and can be combined together. But the impact of such composition is dependent on a particular compromise pattern and is not necessarily beneficial. As each protocol requires a significant number of messages, their inefficient combination should be avoided. Moreover, a change in the compromise pattern may render existing secrecy amplification protocol inefficient. Then, a time-consuming analysis and some design effort are needed to find a new protocol.

3.5 Generation of secrecy amplification protocols

In this part, we propose a method that enables the secrecy amplification protocols to be designed automatically (i.e., with a minimal effort from a human designer) for an arbitrary compromise pattern. Evolutionary algorithm, and linear genetic programming (LGP) in particular, is used to construct new protocols. Candidate protocols are evaluated using a network simulator.

3.5.1 Composition from simple secure protocols

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 [37] for an exhaustive review). Automatic protocol generation (APG) was proposed to automatically generate new protocols with desired properties using a brute-force space search; protocols' correctness is then verified by formal tools [52]. 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 [15] for an a good overview of possible approaches to automatic protocol generation and protocol composition. Fortunately, this is also the case of secrecy amplification protocols because they specify the way how fresh key values are propagated and combined by the parties involved. Thus a secrecy amplification 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 [15] for such protocols; note that a composition is not secure in general), we can skip the formal verification of the composite all together. Instead, we have to verify how many keys from freshly generated secrets will be compromised by an attacker after a secrecy amplification protocol execution. An attacker is able to eavesdrop the content of some secrecy amplification messages as he knows some of the keys used (a partially compromised network is assumed due to possibility of an attacker to capture nodes or eavesdrop fraction of a communication). 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 secrecy amplification execution in a real network.

By substitution of a formal verification tool by a network simulator for a faster evaluation, we additionally obtain a smoother indication how good a candidate protocol is. Instead of binary indication "secure or flawed", we will obtain the number of links additionally secured by a particular protocol (in degenerated case, this can still be only "0% or 100%" links secure). Hence we can use some kind of informed search instead of an exhaustive search.

3.5.2 Evolutionary algorithms

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 secrecy amplification 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 EAs 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* [30]. 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 required behavior. The most promising outcome of this approach is that an artificial evolution can produce innovative 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 [5].

3.5.3 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 set of primitive instructions is defined in such a way that each of the instructions has 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 with the aim to describe all published secrecy amplification protocols and use only (cryptographic) operations available on real nodes. A candidate secrecy amplification protocol is represented as a program composed of these instructions and modeled 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 on node N_a into slot R_i.
- CMB N_a R_i R_j R_k – Combine values from slots R_i and R_j and store the results in slot R_k. The combination function may vary on the application needs (e.g., a 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 slot R_i using the key from slot R_j and store encrypted result in slot R_k.
- DEC N_a R_i R_j R_k – Decrypt a value from slot R_i using the key from slot R_j and store decrypted result in slot R_k.

Each instruction has an additional boolean switch, which can turn the operation off (to equivalent of NOP), without changing the instruction itself. This allows the LGP to temporarily disable or enable some instructions. 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 3.6 and 3.7.

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;}[1], the 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;}, the 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 and four nodes (each with six memory slots only) there are more than 10^{21} possible configurations[2]. Proper restrictions might limit the total space size, but such limitation requires some knowledge about the target environment and relations between protocol and compromise pattern. Our goal is to create the method which requires only description of compromise pattern in form suitable to simulator, rest being done by proposed automated method.

3.6 Generation of node-oriented protocols

In this part, we focus on the automatic generation of amplification protocols for a fixed number of k parties, i.e. the same scenario as used in [3, 16]. 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[3], 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.

[1]New key is generated on node N_1 into slot R_1 and then send to node N_2 and stored in its slot R_1.

[2]$(6 \times 4 \times 6 \times 6 \times 6)^6$

[3]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.

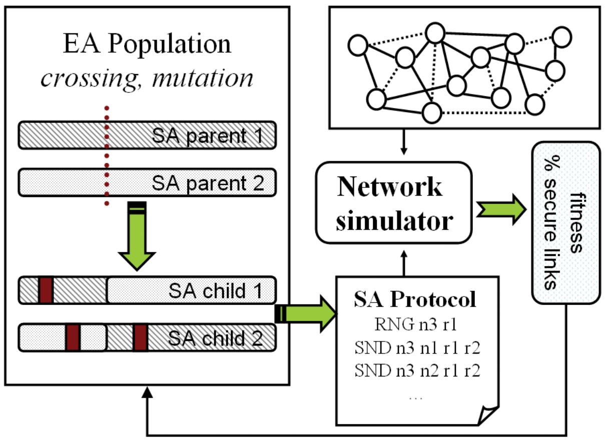


Figure 3.9: Automatic protocol generation process with fitness evaluation. The new population of genotypes is transcribed into candidate protocols. Using our network simulator and a given partially compromised network (dotted links), the fitness value (fraction of secured links) is calculated for each candidate protocol.

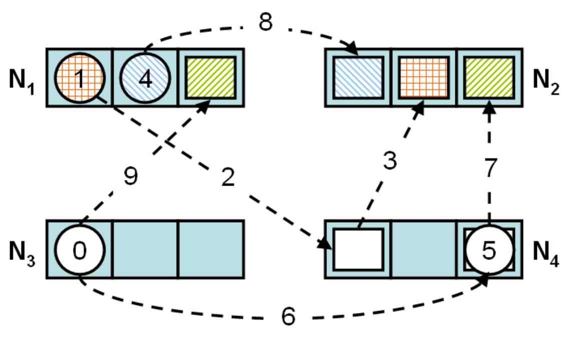
3.6.1 Overview of the method

Initially, we generated five protocols with 200 randomly selected primitive instructions. These candidate protocols form the initial population for the LGP. Every protocol is then simulated on our network simulator 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. Protocols of the first generation are not usually 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.

We like to stress that usage of LGP is not the only possibility. Several other methods for generation of candidate protocols for simulator evaluation might be used as well including the brute-force search. We chose evolutionary algorithms as these have been already successfully used in WSN (although for a different purpose like optimal node placement [22]), usually exhibiting significantly faster convergence towards solution than brute-force search.

3.6.2 Parameters of experiments

The following reference setting of LGP and simulator was used: target plane was 3x3 units large with 100 deployed legal nodes. Each node has 0.5 unit maximum transmission range, which results in 8.2 legal neighbours on average. For Key Infection scenario, there was 10 attacker's eavesdropping nodes. For Random compromise pattern, 50% of links were randomly marked as compromised. Each party has 8 memory slots for storing intermediate values and candidate protocol was limited by 200 elementary instructions. Simulations are performed for three distinct network deployments, the average fraction of secured links is



#	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 3.10: 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 and hatching.

used as the resulting fitness value. The number of nodes was intentionally kept low to make the simulation as fast as possible. The functionality of the evolved protocol was later verified on much larger network with 4000 legal nodes.

3.6.3 Results for node-oriented protocols

The best performing 4-party protocol discovered by LGP was produced within 4 days on a 3GHz processor in the 62786th generation. The protocol consists of the instructions shown on Figure 3.10. This is a “pruned” version of the original 200-instructions long protocol found by evolution. Importance of each instruction was tested[4] by its temporal disabling (pruning) – 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). Figure 3.11 shows a typical graph of fitness values for one particular run. The experiment was repeated 30 times where 14 LGP runs provided protocols with the same fitness as the best performing one (protocols differed only in the instruction order and used memory slots, otherwise functionally identical) before the 100000th generation when the evolution was stopped.

This protocol can be further post-processed. Only three memory slots are actually required on each node instead of eight slots that were available to LGP.

All amplification protocols we were aware of at the beginning of our work were re-discovered here by LGP. The simple key transfer between neighbours is encoded in steps {4,8}. The Push protocol by [3] is encoded in steps {1,2,3}. The Pull protocol by [16] is encoded in steps {0,6,9}. The multi-hop version [11] of the Pull protocol is encoded in steps {0,6,7,9}. Moreover, the new protocol outperforms existing amplification protocols in fraction of secure links, as shown in Figures 3.12 and 3.13.

[4]After the end of the LGP search as a post-processing, not impacting the evolution itself.

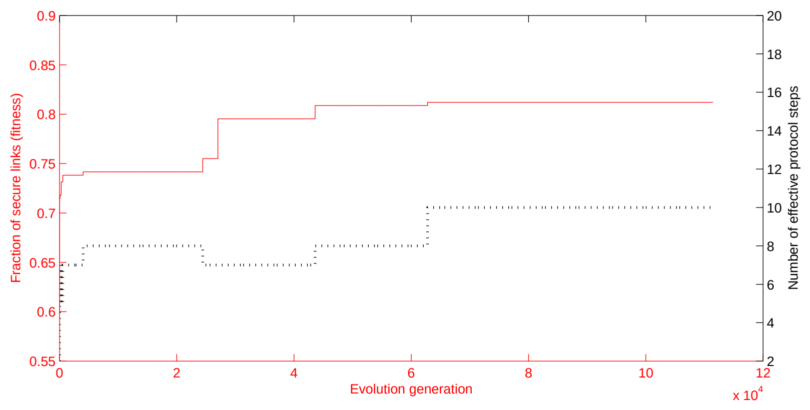


Figure 3.11: Progress of the best fitness value and the number of effective instructions of the best node-oriented protocol. Solid non-decreasing line shows fraction of secure links for best performing protocol in each generation and dashed line show number of effective steps for this protocol.

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, i.e. nodes N_3 and N_4 cannot directly communicate via a radio link, 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 enables the protocol execution even when only a limited number of nodes is reachable. It would be hard for a human designer to propose such a protocol as dependency between the instructions and neighbour layout is rather complex, especially for group-oriented protocols (discussed later in Section 3.7).

Note that the automatic design of the node-oriented protocols with 5+ parties (nodes that take part in single execution of the protocol, independent of the network size) was not possible in the proposed framework because the number of simulated messages grows exponentially with the number of parties involved. The simulator is not able to evaluate such protocols fast enough to obtain a fitness value. Slow evaluation prevents the evolution to proceed towards better solutions in a 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. Secondly, more probable reason can arise from the setting that we applied to speed up the evaluation of candidate protocols. If the link already has some assigned key, this key is transparently used for encryption, as it is obviously a 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 LGP runs were performed for the case when the transparent link encryption was *not* used. Evolution was significantly slower to achieve the same fraction of

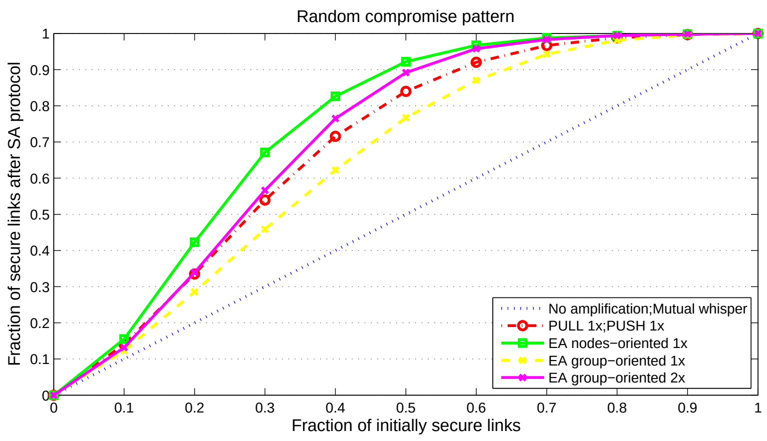


Figure 3.12: An increase in the number of secured links after secrecy amplification protocols in the Random compromise pattern. The Push and Pull protocols give the same results; mutual whispering does not improve security at all. Evolved group-oriented protocols will be described in Section 3.7. As can be seen, strong majority of secure links ($> 90\%$) can be obtained even when the initial network had one half of compromised links.

secured links, but the link encryption using existing keys was essentially developed anyway via the ENC and DEC instructions.

3.7 Generation 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 protocols. 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 without unnecessary message transmissions instead of three or four nodes. Due to the broadcast nature of the wireless transmission, nodes' geographic positions also influence the result of a secrecy amplification protocol. And finally, due to the random placement of nodes in the sensor networks, the number of direct neighbours may vary significantly; the protocol constructed for a fixed number of parties can even fail with an insufficient number of participants.

3.7.1 Overview of the method

We present a different approach to the design of secrecy amplification protocols with respect to established scenarios used in [3] and [16] (that are denoted as node-oriented protocols in this work). Identification of the parties in the protocol is no more "absolute" (e.g., node number 1, 2, 3), but it is given by the relative distance from other parties (we will use the distance from two distinct nodes). It is assumed that each node knows the distance to its direct neighbours. This distance can be approximated from the minimal transmission

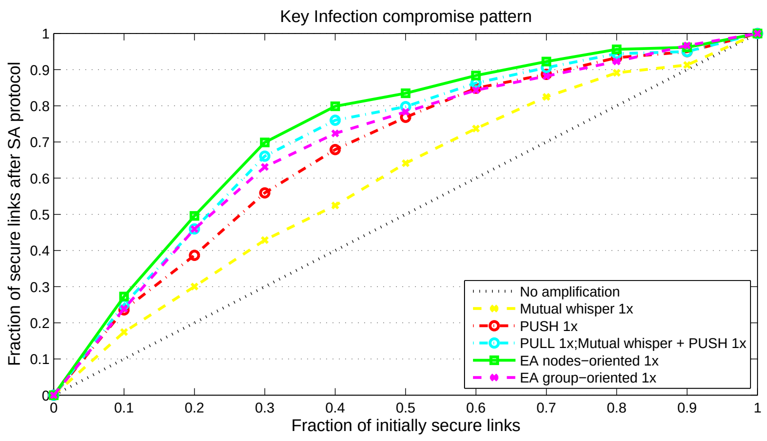


Figure 3.13: Key Infection compromise pattern (8 legal neighbours in average). The Pull protocol provides better results than the Push protocol. The combination of mutual whispering with the Push protocol gives the same results as the Pull protocol alone.

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). In other words, LGP still operates the same instructions of the protocol as in the case of node-oriented protocols, but additionally also with the distance values to identify the involved nodes. Based on the actual distribution of the neighbours in the field, 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 their actual positions may be slightly different from relative distances indicated in the protocol).

The evaluation process of a group protocol is more complex than for the node-oriented protocols, but the number of totally exchanged messages is significantly lower. Note that the spared messages come from the change of the secrecy amplification evaluation rules, not the LGP itself. The role of LGP is to find a protocol, which will operate in the restricted scenario with much less messages (with respect to node-oriented protocols, where all k-tuples are executed) and which will still be able to perform comparably to the node-oriented protocols in terms of the number of secure links.

The evaluation is based on the following rules:

1. Every node in the network is separately and independently processed 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.
2. A separate protocol execution is performed once for each direct neighbour (node in the radio transmission range), this neighbour will have a special role in this execution

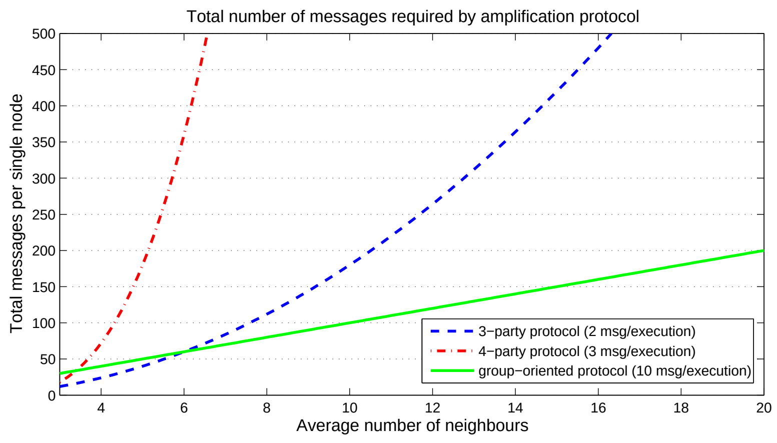


Figure 3.14: Total number of messages per single node required to perform a 3-party, 4-party node-oriented and group-oriented secrecy amplification protocol. Even when group-oriented protocol utilizes significantly more messages per single execution, total number of messages is smaller.

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 instead of $\binom{10}{k}$ for node-oriented). This is a key difference from node-oriented protocols, and cuts the communication overhead considerably.

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 (keys) 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 of the protocol are replaced by real identification of the nodes which are positioned as close as possible to the relative identification given by N_C and N_P in the protocol.

5. When the next node is executed as a central node N_C, the memory slots of all direct neighbours are cleared (memory values *cannot* propagate between executions with a different central node N_C) as such process requires non-trivial synchronization in real network.

Figure 3.14 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 with an 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 [8] without a transmission of the values or, even 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 influence 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 – the ordering of the actions of nodes in the simulator thus does not impact the results). 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.

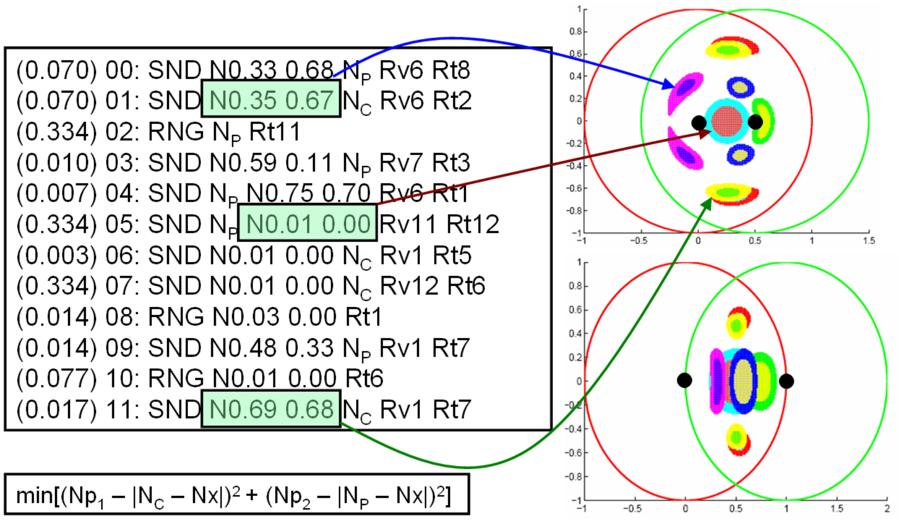
3.7.2 Results for group-oriented protocols

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 3.12 and 3.13 for the performance of evolved group-oriented protocols). Example of such evolved protocol is presented in Figure 3.15. Such a protocol has typically 10-15 important instructions and uses neighbours from 5-7 geographically different areas. The SND instruction is the most common one, forming 60-80% of instructions of discovered protocols. There is not only one "best" protocol – instead, most LGP runs provide some useful amplification protocols which differ in their instruction order.

In contrast to node-oriented protocols, instructions of the evolved protocols are more difficult to understand as the parties are not directly specified any more. Various techniques such as real-time visualization of message transmission, analysis of memory store/load sequences or visualization of probable areas of relatively identified parties (see Figure 3.18) can be used to recognize actual purpose and importance of the instructions.

Again, interesting and rather unexpected "tricks" were introduced by evolution. Firstly, two SND instructions in an example protocol shown in Figure 3.15 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 in example protocol. Evolution was able to include such "overlapping executions" in the protocol even when not explicitly designed to, while this might be difficult for a human designer.

Surprisingly, the most important intermediate node (node that is responsible for the majority of newly secured links between nodes N_C a N_P) is not positioned in the center between these two nodes (i.e. in area A in Figure 3.18 a)) which would reflect the assumption that shorter links have a smaller probability to be compromised in Key Infection pattern. Instead, the most probable position for that intermediate node is area C shown in Figure 3.18 a). Note that position of area C (and so intermediate node) depends on the distance between nodes



```
(0.070) 00: SND N0.33 0.68 NP Rv6 Rt8
(0.070) 01: SND N0.35 0.67 NC Rv6 Rt2
(0.334) 02: RNG NP Rt11
(0.010) 03: SND N0.59 0.11 NP Rv7 Rt3
(0.007) 04: SND NP N0.75 0.70 Rv6 Rt1
(0.334) 05: SND NP N0.01 0.00 Rv11 Rt12
(0.003) 06: SND N0.01 0.00 NC Rv1 Rt5
(0.334) 07: SND N0.01 0.00 NC Rv12 Rt6
(0.014) 08: RNG N0.03 0.00 Rt1
(0.014) 09: SND N0.48 0.33 NP Rv1 Rt7
(0.077) 10: RNG N0.01 0.00 Rt6
(0.017) 11: SND N0.69 0.68 NC Rv1 Rt7
```

$\min[(Np_1 - |N_C - Nx|)^2 + (Np_2 - |N_P - Nx|)^2]$

Figure 3.15: 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. Function shown at the bottom is used to select best fitting node based on its position with respectd to N_C and N_P. N_{p1} is distance from N_C and N_{p2} is distance from N_P specified in a protocol.

N_C and N_P. When these two nodes are close to each other then C is "behind" node N_C (Figure 3.18 a)). As the nodes move away from each other, area C moves around N_C to the position shown in Figure 3.18 b)). When both nodes are very close to the maximum transmission range then C is located in one third of the distance between N_C and N_P, closer to N_C (Figure 3.18 c)).

3.7.3 Methods for analysis of evolved protocols

Various techniques such as real-time visualization of message transmission, analysis of memory store/load sequences or visualization of probable areas of relatively identified parties (see Figure 3.18) can be used to recognize actual purpose and importance of the instructions.

Real-time visualization of message transmission – Elementary protocol functionality can be obtained by observing the visual representation of the protocol execution with a set of nodes with fixed positions. 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 – The fitness reduction effect can be studied to identify groups of instructions with cross-dependent fitness values. As the protocol has already been pruned, removing any single instruction I means a fitness decrease. An additional pruning process over the reduced protocol (without I) gives us the difference in the fitness gain for each of the remaining instruction (some instructions may be completely removed if they have no function without I). The

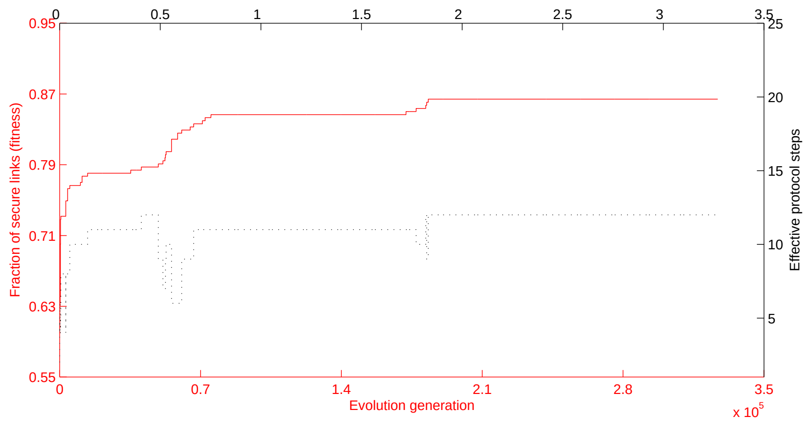


Figure 3.16: Progress of increase in fitness value and number of effective instructions of group-oriented protocol evolution. Solid non-decreasing line shows fraction of secure links for best performing protocol in each generation and dashed line show number of effective steps for this protocol.

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 previously, 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 the process (see Figure 3.17 for example). More precisely, if there is instruction I that reads from memory slot M_i and writes to memory slot M_j, we can connect vertices M_i and M_j in the graph by an edge labeled by I. Resulting graph can be then analyzed to obtain the indication of paths where the values propagates during the protocol execution.

Probable areas for parties identified by the relative distance – The visualization of the areas, where nodes referenced in the protocol will be positioned with high probability is an important source of information how a given protocol works. Note that these areas are not static for all nodes N_P, but differ significantly with the distance between the 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 the information how the fresh key values are propagated in the group. Using this technique, we can derive (see Figure 3.18) that instruction 3 sends a value stored by the instruction 9 in the previous run of the protocol when the position of N_P is around 0.6 of the maximum transmission range of N_C and in the layout area B. The reason is that in this distance the layout area B overlaps with the position of node N_P and these two parties of the protocol are most probably mapped to the same physical node.

Note that removal of a single instruction I can not only decrease decrease the overall fitness value, but it can also increase the contribution to fitness value of 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

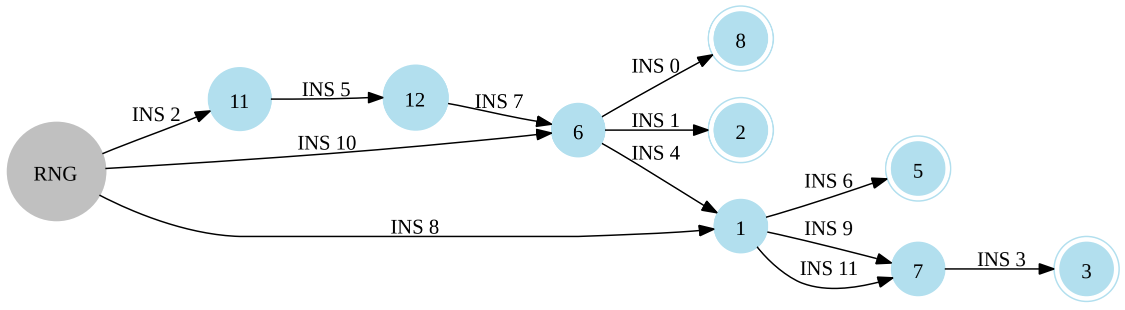


Figure 3.17: Memory chain for example group-oriented protocol from Figure 3.15. Circles constitute memory registers with labeled edges standing for instruction responsible with register manipulation. The memory registers used for the final link key construction are Double-circled.

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, i.e. 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 node-oriented protocol. In this task, evolutionary design provides not only the required functionality, but also robust solutions.

3.7.4 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 eavesdropping nodes / compromised links). When instructions {2,5 and 7} are removed from the protocol, the fraction of secured links decreases by about 20-40%, with the 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 kept 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 the 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 eavesdropping nodes/compromised links.

First two SND instructions may appear useless (no value is available in memory slot 6 for the first run of the protocol), but as the protocol is executed repeatedly for all nodes within

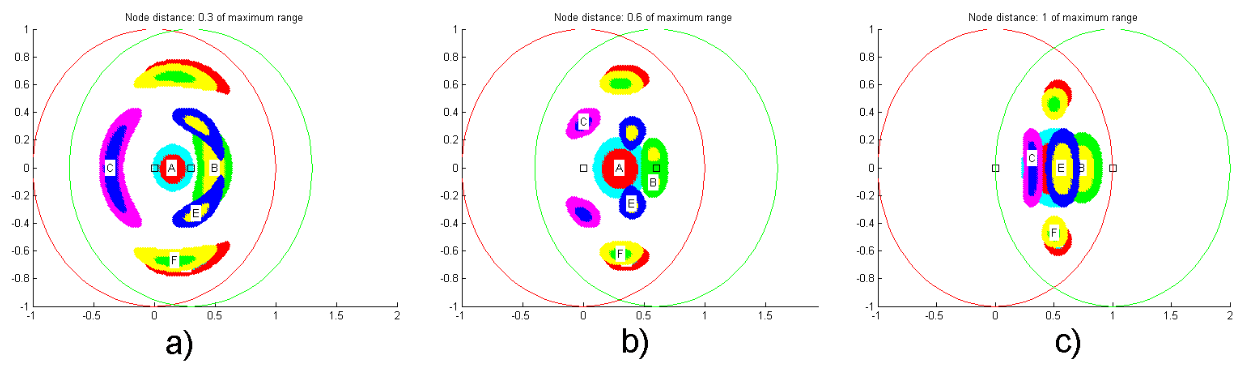


Figure 3.18: Layout of areas for potential parties when the distance between the central node N_C and node N_P is a) 0.1 of the maximum transmission range, b) 0.6 range and c) the maximum transmission range.

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 3.18. 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 3.18, part a)). As the nodes move away from each other, the area C moves around N_C to the position shown on Figure 3.18, 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 3.18, part c)).

3.8 Parameters used for LGP

This section summarizes used parameters and setting for linear genetic programming used to generate candidate protocols.

Based on random sampling test, the fitness landscape[5] for node-oriented protocols seems to be highly non-linear with only few significant fitness values in the search space. A small population with a rapid mutation is suitable for solving such problems (similarly to evolution of digital circuits using Cartesian Genetic Programming (CGP) [39]). The population size was fixed to 5 individuals. The mutation operator is applied with the 10% probability. Similarly to the CGP, crossover is not used (series of experiments did not shown improvements in evolution convergence when crossover was used).

[5]The fitness values for each possible instance in the search space. Note that 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.

The fitness landscape for the group-oriented protocols seems to be smoother than for node-oriented protocols. We utilized 20 individuals in the population and a single point crossover operator applied with the probability 70%. Crossover points allowed at the level of instructions only. Mutation with a 5% rate was used. Fitness evaluation was significantly faster (as significantly less messages had to be simulated) than for the node-oriented candidate protocols. Therefore, significantly more generations could be used. Steady state replacement rule (GASteadyStateGA in GALib) for the worst 1/3 of actual population is used to maintain population size for both types of the protocols.

3.9 Automatic protocol generation conclusions

We examined the area of automatic design of secrecy amplification protocols and their relation to the underlying key distribution protocol in wireless sensor networks. Some secrecy amplification protocols may work well for the networks with randomly compromised links (e.g., resulting from node capture for probabilistic pre-distribution), but may give a sub-optimal performance when applied to more correlated compromise patterns arising from distribution approaches such as Key Infection. Moreover, some steps of the secrecy amplification protocol may be pointless for a given compromise pattern as they do not improve secrecy of any link – and thus impose only an unnecessary message overhead.

We have described a more flexible approach based on the fact that the effectiveness of secrecy amplification protocols can be automatically evaluated using a network simulator. Linear genetic programming was used to search for new protocols. We were able to rediscover all published protocols for secrecy amplification we are aware of, and to find a new protocol that outperforms existing. The new protocol operates with four parties, but is able to operate even when only three parties are available. A single iteration of the secrecy amplification protocol can increase secure links from 60% to more than 95% for the Random and 88% for the Key Infection compromise pattern.

A significant disadvantage of existing secrecy amplification protocols is their high communication overhead because the number of required messages grows exponentially with the number of direct neighbours. By moving from node-oriented protocols to group-oriented protocols and using evolutionary design approach, we were able to find protocols with the fraction of secured links comparable to node-oriented protocols, but with only a linear (instead of exponential) increase of required messages with respect to the increasing number of neighbours. This is especially important for dense networks with more than 10 neighbours.

Chapter 4

Evolution of attacker strategies

The fundamental asymmetry between the attacker's and defender's position is that an attacker needs to find only one successful attack option where the defender should take care of all possibilities. This asymmetry is similar to the relation between a guided search without processing the entire search space and an exhaustive space search. We based our work on the assumption that guided search for new attacks is generally possible – at least on the same level as searching for defenses against an attacker. While we speak of search for attack strategies, this approach has clear benefits for defenders as well – discovery and study of new attacks should help the defender to build a better protection.

The advantage of automatic search provides us with the possibility to reliably examine all configurations within a constrained search space as formal verification does (see [37] for an exhaustive review). And so instead of using brute force search, we can search through even larger spaces using some form of guided or even random search – yet with an open question of assurance that a relevant configuration was not missed.

4.1 Related work

So far, automated constructions of attacks were proposed mainly for construction of testing beds for Intrusion Detection Systems (IDSs) or optimization of parameters for known attacks. Automated construction of attack graphs is proposed in [51] using symbolic model checking algorithms with example application to network security, where potential violation of safety property is constructed from four atomic attacks. And all possible attack vectors are constructed.

In [36], virtual network is used to capture whole network traffic with traces of known attacks from vulnerability databases. This traffic logs can be later used to test particular IDS. In principle, recombination of several attacks can be run in parallel to produce more obscured network traffic and more successful attacks (against a particular IDS) can be found.

Automatic generation and analysis of attacks against IDS systems is proposed in [48]. Formal transition rules are specified to transform the attack footprint from one known to a particular IDS to one that bypasses the detection. Soundness property of rules ensures that only valid attacks are derived; therefore method allow easy evaluation whether the attack was detected or not. Several significant bugs were found in the well known Snort IDS using this method.

Formal derivation method capable of generating polymorphic blending attacks that use encryption to hide the attack code is proposed in [23]. IDSs are modeled as finite state automata and problem of finding suitable encryption key that would not trigger IDS detection in an incoming packet is shown to be NP-complete. A hill climbing heuristic method is used to search through a potentially large space of possible encryption keys for near optimal solution.

Simulation with discrete event system specification (DEVS) is used to automatically generate attacks by recombination from several groups of shell commands in [33]. All possible combinations of the commands valid within given constraints are generated via DEVS and attacks are obtained as a paths between initial and compromised state.

It was an earlier work on evolutionary design of secrecy amplification protocols with a suspiciously high fraction of secured links (typically 100%) that lead us to a deeper inspection of the protocol with such a high performance. Here we discovered either our program mistake or incomplete specification of the evaluation function that was exploited by the evolutionary algorithms. Repetition of this behaviour then lead us farther to the idea of using evolutionary algorithms to search not only for defenses (like the secrecy amplification protocol) but also as a tool searching for new attacks (mistakes in code or incomplete specification). See Chapter 3 for details on automatic protocols design, examples of discovered protocols, their performance comparison and used settings of evolutionary algorithms.

4.2 Automatic design of attacks

We propose to use an automatic attack strategy generator together with simulated or real execution environment to generate and test large amount of candidate attacks. Additionally, we propose to use evolutionary algorithms (EAs) instead of brute-force or random search over the space of the possible attacks.

Automatization of the whole process and close binding to the real execution environment offer the possibility to find out low-level attacks aware of the system context rather than their high-level abstraction only.

Note that – due to their nature – EAs are more suitable to find the attacks rather than to prove that none exists as an attacker needs to find only one attack vector (analogy to a guided search) whereas the defender must count with all possible attacks (analogy to exhaustive search).

4.2.1 General concept

We have developed a general concept for automatic design of attacks. It consists of the following sequence of actions:

1. The candidate strategies are generated from elementary rules using specified mechanisms like:
   - *Educated guess* – field expert selects combinations of elementary rules that might work.
   - *Exhaustive search* – all possible combinations of elementary rules are sequently examined and evaluated. Inefficient for large search spaces.
   - *Random search* – combination of elementary rules is selected at random. No information about quality of previous selection is used during following one.
   - *Guided search* – actual combination of the rules is improved according to some rating function able to compare between quality of previous and newly generated candidate strategy. We will use the evolutionary algorithms for this task.
2. Candidate strategy is translated from the metalanguage used for generating into the domain language which can be interpreted by a simulator or real system.
3. Candidate strategy is executed inside simulated or real environment.
4. Impact of the attack is measured by a fitness function (see Section 4.2.3 for a further discussion).
5. The whole process is repeated until a sufficiently good strategy is found or the search is stopped.

Details are as follows. Prior to actual generation we have to inspect the system and to define basic methods how an attacker may influence the system (create/modify/discard messages, capture nodes, predict bits, etc.) and what constitutes a successful attack. Subsequently we decompose each basic method into a set of elementary rules and identify its parameters (e.g., modification of x-th byte in the message, delay a message certain time x, capture a particular node, ...). These elementary rules serve as basic building blocks of new attack strategies. The more detailed the level of basic methods is, the wider the possibility to express attacks, but at the price of a slower progress when searching for an attack as the search space is also larger. Having these blocks, we can start generating the strategies.

Note that described concept cannot provide a proof that the system is secure (under some assumptions) as formal verification or provable security does. In the worst case, we might still end up with only repeated random guesses of the possible attack strategies. However, practical experience with the usage of EAs proved to be usually much more efficient than random search. Combination with an accurate simulator (e.g., network simulator) or even real system (e.g., real network) allows us to work with a better representation of the target system than with an abstract model used for formal verification.

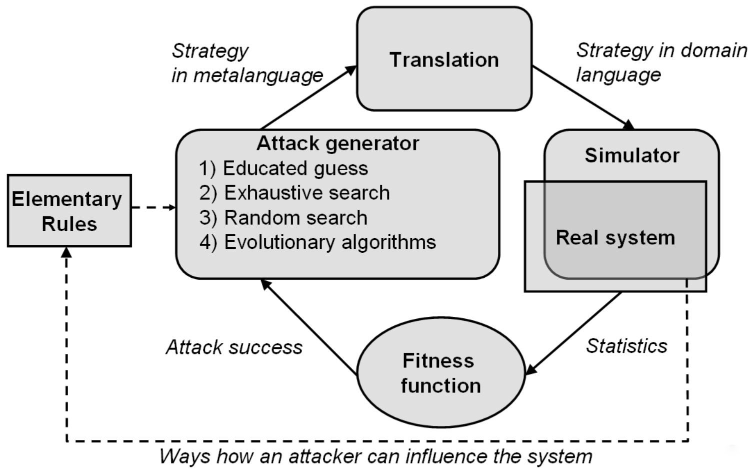


Figure 4.1: Automatic attack generation process with success evaluation. A new attack strategy is generated in a metalanguage from elementary rules created for a specific environment. Based on the used evaluation context, strategy is transcribed from the metalanguage into actions in the target environment. Statistics about attack are obtained and evaluated using a fitness function to provide guidance for the next generation of attacks.

In this work, we use mainly linear genetic programming (LGP) to generate the attack strategies where LGP represents a candidate strategy as a sequence of instructions (steps) [5] and classical version genetic algorithms to optimize free variables of the attack strategy. The additional discussion about usage of EAs in area of security for automatic generation of secrecy amplification protocols can be found in Section 3.5.

4.2.2 Strategy description

Attack strategy produced by the generator is typically described using some proprietary abstract language (*metalanguage*). The form of this language is determined by the generating mechanism and its properties (in case of EAs it has to provide simple and efficient ways for evolution and its operations). This language not have to be suitable for evaluation environment. Hence the generated strategy has to be translated into into *domain language*, which can be executed by the simulator or real execution environment. Note that the existence of translation step also supports use of simulators or generators, so the existing systems can be used for both generating and execution. The same as for simulator holds for real systems. Here the need for translation is even more obvious.

4.2.3 Fitness function construction

The tricky part and key to successful usage of EAs is specification of the proper fitness function. Fitness function must fulfill the following conditions:

Capture the progress towards the optimum – the fitness value of candidate solutions within relevant properties must capture the relationship between actual quality of candidate solution and intended goal we would like to achieve. E.g., if we like to identify 100 nodes to compromise to capture most keys, replacing one node by another within this set will probably change the amount of captures keys only a little. Some problems have straightforward metrics usable as the fitness function like the proportion of secret and compromised messages, average time to delivery message, fraction of compromised secrets or fraction of system resources available to legitimate users (DoS attack). But others may be more difficult to tackle – e.g., when we try to generate exploit for buggy code, how can we measure progress towards a successful attack? Still, some indirect metrics might exists like number of instructions executed above processing of the legal request.

Sufficient granularity – if the fitness function outputs only two values like "0% keys compromised" and "100% keys compromised", there is no potential for evolution to gradually increase the quality of the solution. Either the solution for 100% compromised keys is directly found by a chance or the solution is no closer to optimum than any other 0% solution. Smoother the function is usually the better.

Fast to compute – evaluation of a single candidate solution must be fast enough to evaluate 10^2 to 10^6 or more candidates in reasonable time. The exact time constraint depends heavily on the solved problem, but evaluation of one candidate should typically be completed in the order of seconds or less. The faster the evaluation is, the higher is the fraction of examined search space and the better is the chance to find a satisfactory solution.

4.3 Evolution of attack strategies

The described concept does not need to generate complete attack strategies starting from very basic rules. In the simplest case, new attack strategies are generated only as a recombination of already existing generic elementary attacks (e.g, replay a message, change the IP address in a packet header, capture a node). EAs are searching only for a sequence of such elementary attacks that together lead successful attack. If we give more freedom to EAs by increasing the granularity of rules, which means we decompose the generic attacks into more elementary rules (e.g., modification of X-th bit of message regardless of the structure of message), we get more possibilities. Results range from improvements of existing attacks by optimization of their parameters up to finding completely novel attacks. Note that the transition between recombination-only and novel attacks is not discrete as it depends on the granularity of the elementary attacks we are using, and the level of freedom we allow is often relative to the solved problem.

4.3.1 Re-combination of the existing attacks

Generic attacks are written as a sequence of elementary rules and EAs create combination only at a generic attack level, not on the rules level. Pre-specified generic attacks also serve as an significant evolution speed-up as it is not necessary for evolution to develop known attacks from scratch. Example generic attacks can be replay, reflection or interleave message attacks, forged IP addresses in a packet header, forged ARP packets, captured packets in a promiscuous mode or claim fake identity. Generic attacks alone may or may not be a successful attack strategy alone. E.g., if the target of an attacker is DoS for a selected computer, then a forged ARP packet alone is often sufficient. In the case of data traffic exposure, it must be combined with a subsequent packet capture of the redirected traffic.

4.3.2 Improvement (optimization) of known attack strategy

In this case, a particular attacker's strategy is known in advance (e.g., capture and extract keys from some nodes and use them to compromise communication), we are only optimizing parameters of the strategy (e.g., which particular nodes should be captured). This is the most common usage of EAs in other domains – as a tool for parameter optimization.

This approach is suitable when we know some parameterizable tradeoff based attack and we like to obtain better attacker success than with existing parameters. Alternatively, basic principle of attack is known but actual parameters must be set according to (complex) relations of actual environment (e.g., particular network deployment and type of applied secrecy amplification protocol).

4.3.3 Finding novel attack strategies

If we focus on the granularity of elementary rules, we can extend the re-combination approach to find novel attacks mechanism. If we do not restrict ourselves only to known attacks and their parameters, but introduce more general rules describing what else might an attacker be able to observe and manipulate inside the system, EAs might be able to evolve a completely novel attack. However, as the additional rules also increase the search space, the evolution progress will often be slower than in previous cases and with an uncertain outcome. But the ability of EAs to come up with unique solutions is beyond human capabilities as was demonstrated in the area of hardware circuits [54] and may lead to novel attack strategies difficult to be conceived by a human expert.

4.3.4 Promising areas

Not all areas have the same potential for automatic search within the described concept. Systems with straightforward and accurate fitness functions (like the fraction of compromised messages) are generally more suitable. EAs typically works well within systems with complex relations depending on multiple input variables, where the fitness landscape[1] is not discrete but contains local minimums and maximums with gradual transition. In case of attack strategies, it is important to have a gradual decrease in security after an attack instead of

[1]Virtual hyper plane of fitness values for all possible points inside search space.

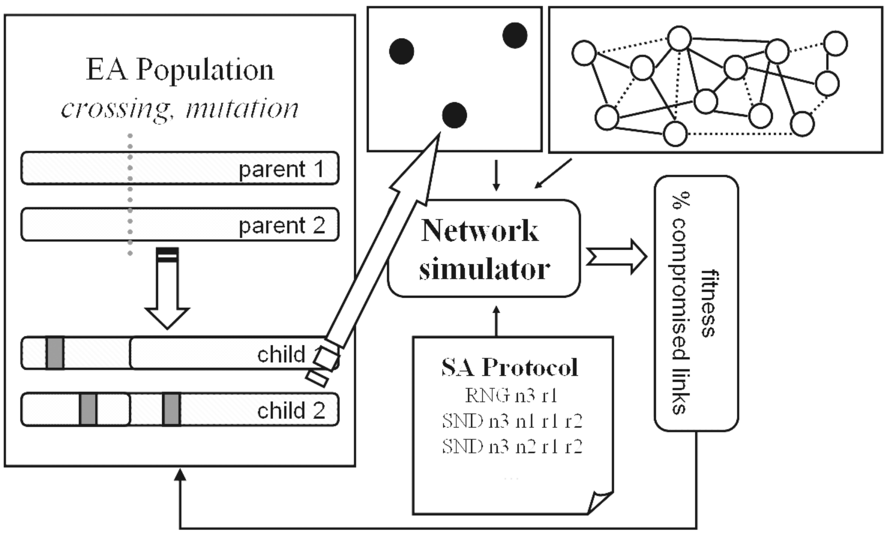


Figure 4.2: Overview of eavesdropping pattern generation.

only “0% or 100% compromised”. In discrete cases, EAs are just as effective as random search (might be still useful under some circumstances). Particularly suitable are the environments with already existing partial compromise due to security/resources tradeoff that can be unbalanced by a better attack strategy like is the case of wireless sensor networks.

We expect that recombination and optimization of known attacks will provide the most useful results. But different “way of thinking” of EAs may lead to unexpected and surprising discoveries of novel attacks. Here, the success rate will be highly dependent on the proper choice of the elementary rules used to build up the attack strategy.

4.4 Applications

Our inspiration for this work came from research on automatic protocol design in Section 3.5 and we applied the described concept to search for attacks in the domain of wireless sensor networks. But the concept is not limited to WSNs and can be used for other applications as well if suitable evaluation function can be constructed for particular environment and settings.

4.4.1 Optimal eavesdropping pattern

Lightweight key distribution presented in [3] requires no pre-distributed keys as link keys are exchanged directly in plaintext between neighbours “in situ” with secrecy amplification protocol executed afterwards. Weakened attacker with limited ability to eavesdrop in the network is assumed, where attacker’s eavesdropping nodes are on equivalent technical level (radio sensitivity) to legitimate nodes of the network owner, but present only in a fraction amount (results for 1-5% ratio for which a reasonably secure network can be set were origi-

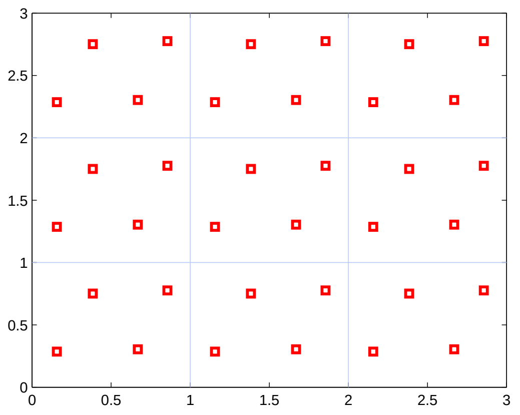


Figure 4.3: Deployment 4-nodes pattern for eavesdropping nodes found by an evolution. Nine neighbouring cells are displayed to show the pattern tying.

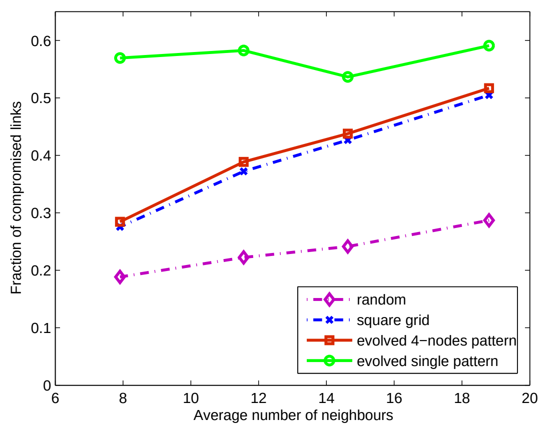


Figure 4.4: Comparison between eavesdropping nodes placement, grid-like pattern and evolved patterns with and without information about position of legitimate nodes. Eavesdropping to legal nodes ratio = 20%.

nally presented, and then improved up to 20% in [16]). The attacker's success is influenced by the placement of eavesdropping nodes. Original results presented in [3, 16] were based on an assumption that eavesdropping and legitimate nodes are both deployed in a random fashion. However, placing nodes in specific pattern may consistently provide better results than for the random case. We can increase the number of secured links when evolving the pattern for legitimate nodes or increase the number of eavesdropped links when evolving the pattern for eavesdropping nodes.

We performed automatic search of an attacker that can precisely deploy its nodes (e.g., manually) using our new concept. Our network simulator (see Section 1.2) was used, and candidate attack strategies were used to encode the eavesdropping deployment pattern. Square deployment field was assumed for simplicity, divided into k^2 equivalent cells, where k is the number of cells per axis. The same deployment pattern was used for every cell. The deployment patterns can have different number of nodes within. Having n eavesdropping nodes in total, the pattern for single cell will have $p = n/k^2$ nodes. A single genome is used for encoding positions (x and y axis) of p nodes within one cell. A pattern with more nodes can be thus created either by increasing the total number of nodes and keeping the same number of cells, or by decreasing the number of cells and keeping number of nodes stable.

Fitness function is based on our simulation result as a fraction of compromised link keys after executing a plaintext key exchange and fixed amplification protocol (we used the Pull protocol as best performing yet simple amplification protocol with low message overhead). When evolving the pattern for eavesdropping nodes, a higher fraction of compromised links implies a better fitness for a given genome.

Two possible scenarios were examined. In the first scenario, actual deployment of legitimate nodes is random and not known to the attacker in advance. One of the well performing patterns for this scenario with group of four nodes is on Figure 4.3. The comparison of the evolved pattern with naïve grid-like distribution shows only a slightly better result for the evolved pattern. This suggests that the amount of the eavesdropping nodes up to 30% of legitimate nodes is not sufficient to form any significantly more successful pattern capable to counter the effects of the secrecy amplification. Instead, eavesdropping pattern is trying to uniformly cover whole area to maximize the probability that an eavesdropping node will be the closest neighbour of some legitimate node and thus eavesdropping all links to it (secrecy amplification will fail in such layout). Impact of the regular eavesdropping pattern increases with an increasing number of non-attacker neighbours. This behavior is caused by the fact that longer some links are the more likely are they to be compromised in Key Infection approach.

In the second scenario, the attacker knows the distribution of legitimate nodes in advance and can place eavesdropping nodes according to this knowledge. Such scenario fits situations when secrecy amplification is used together with probabilistic key pre-distribution like [21, 11, 10] in later stages of the network lifetime and when the attacker knows a fraction of the used key pool. A simple experiment with a single pattern for the whole (fixed) deployment of white nodes shows that significantly better results can be obtained (with respect to unknown deployment). A highly successful pattern of eavesdropping nodes is evolved not only to cover an area as large as possible, but also to joint effort of eavesdropping nodes to cancel an effect of secrecy amplification based on local positions of legitimate nodes. Such attack would be very successful in case when legitimate nodes do not start key exchange and amplification right after their deployment, leaving attacker some time to obtain information about the network topology and to distribute its limited number of eavesdropping nodes accordingly.

Fitness landscape seems to be much smoother than in case for evolution of protocols. More over, only two variables (axis x and y) are used for each node and visualization of single genom is much more understandable than for evolution of secrecy amplification protocols. Best results were obtained using higher number of genom in population (20 individuals), mutation with rate 0.1 and single point cross-over operator used with crossing probability 50%. A crossing operator combines new offspring from k nodes of pattern defined by the first parent and $p - k$ nodes from the second parent.

4.4.2 Selective node capture

Pre-distribution of the keys is not an easy task in the context of WSNs due to limited memory, large set of potential neighbours, susceptibility to node capture and battery-expensive communication. Novel pre-distribution schemes were proposed, including probabilistic pre-distribution [21] and later variations [11, 10, 19, 34, 58] where a random subset of the initial key pool is assigned to each node (without replacement). Two randomly selected nodes can have surprisingly high probability to find at least one shared key, but an attacker can also recover the original key pool by capturing only a fraction of the deployed nodes. Yet the typical attacker strategy is not to capture as many keys as possible, but to compromise enough data traffic with least possible effort. In contrast to the limited eavesdropping model for the previous attack, the assumption here is that an attacker is able to monitor all transmissions and capture few selected nodes as well. An optimization algorithm designed only to maximize the number of captured keys (when identifications of keys carried by every node are known to the attacker, like in the case of seed-based pre-distribution [47]) might not be an optimal strategy as it is not taking the network topology into an account. Different schemes have different node capture resilience and results presented in original papers typically assume the random capture of nodes.

We used our concept to generate a selective node capture strategy that is significantly more successful at the whole network level (selective node capture is an easy task if we want to compromise only selected link(s)). For simplicity, we used original the probabilistic pre-distribution by Eschenauer and Gligor [21] (will be denoted as EG or 1-EG) with initial pool size 96359 keys and 19393 keys for later variation [11] of EG that requires at least 3 shared keys to establish link key (denoted as 3-EG). The ring size with 200 keys is used to maintain probability of connection equal to 33% as the most common settings used evaluations in relevant existing papers [21, 11, 19]. The same method can be used for more complicated schemes and we expect comparable results. EA population with 20 individuals was used, with 5% mutation rate and 50% crossover rate. Every gene of individual encode one node ID selected to capture with genom length equal to total nodes to capture.

We compare four different results from 1) random node capture, 2) capture based on deterministic algorithm maximizing number of extracted keys with high occurrence inside network, 3) capture based on deterministic algorithm maximizing number of keys most commonly used to form link inside network and 4) nodes selected to capture by EA using proposed concept. For simulation purposes, network with 1200 nodes was used, with the attacker compromising fraction of them (from 30 to 150 nodes). The average density of the network was 9 neighbours within one's node transmission range. Such number of neighbours yields to 3 directly connectable neighbours when predistribution settings 33% probability of sharing key is used. Note that in order to find an optimal node capture using brute force just for 30 nodes requires $\binom{1200}{30} \cong 6.2 * 10^{59}$ enumerations.

Deterministic algorithm for case 2) and 3) is constructed as follows: During each iteration, frequency of occurrence as a) number of nodes carrying a particular key in its keyring (maximization of captured key set) or b) number of links secured with a particular key (maximization of compromised links) is computed for each key from the original key pool. More common keys have one of the higher values of occurrence. If a key is already compromised then its value is set to zero. Significance value for each node is computed as a sum of values of occurrence for keys carried by this node. For a given iteration of an algorithm, the node with a highest significance value being selected for capture. No node can be selected twice (until all keys are captured) as its value decreases to zero in the next iteration due to the zero value of occurrence assigned to keys compromised from its keyring.

The results for the random case are an average from ten random selections of nodes. To allow for a fair comparison with automatic design, the result for random capture should be taken as the best value obtained from a random selection of subsets of nodes for the same time as was given to the evolution. We performed such evaluation and the results were only around 10% better than the average from ten selections only and still lower than method which maximize captured keys. Therefore we did not plot these results into figures for clarity reasons.

Figure 4.5 shows results for basic version of EG scheme with different amounts of captured

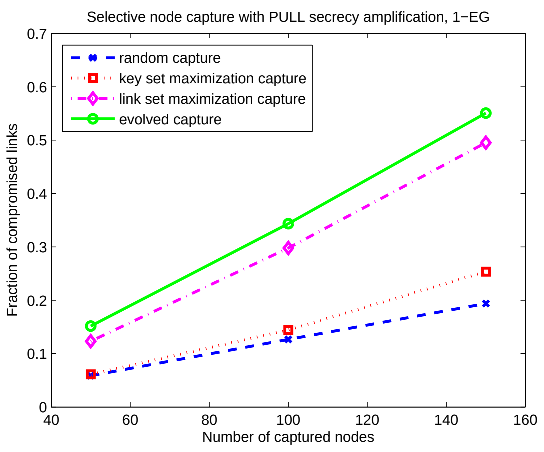


Figure 4.5: Attacker's success for different selective node capture techniques for EG predistribution with initial pool size 96359 keys and ring size 200 keys with the Pull secrecy amplification protocol.

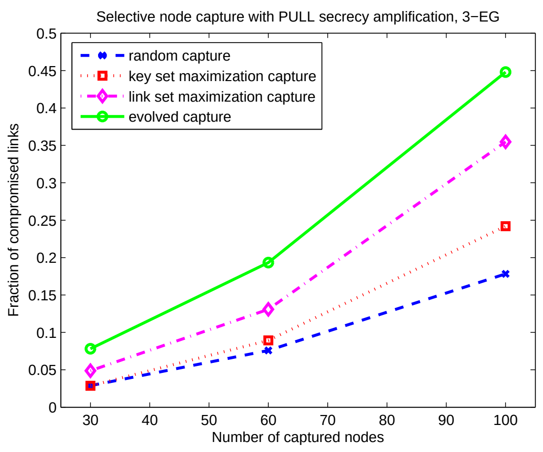


Figure 4.6: Attacker's success for different selective node capture techniques for EG predistribution with at least 3 keys required for link establishment with initial pool size 19393 keys and ring size 200 keys with the Pull secrecy amplification protocol.

nodes (50, 100, 150) and the Pull secrecy amplification protocol [16] executed at the top of the established links after the key discovery phase. A significant improvement with the use of EA can be seen over random capture and slightly better results than link capture maximization algorithm as well. Maximization of captured key set is not an efficient strategy as it does not take into the account actual distribution of nodes. Figure 4.6 shows results for the same settings, but with the 3-EG instead of 1-EG. Here we present results for compromise of 30, 60 and 100 nodes as 3-EG scheme generally provides a better node capture resilience than 1-EG for a smaller number of compromised nodes (see [11] for detailed comparison) both having approximately same resilience for 100 nodes (for random capture). Improvement of EA over deterministic link maximization algorithm is more significant here (ranging from 30 to 60%) as the relation between captured keys and compromised links is more complex here – if all keys used to establish a link key are not compromised together, this link remains secure.

4.4.3 Incomplete fitness function specifications

We also encountered an interesting situation when the implicit assumptions were made about the system, but these assumptions were not realized in our simulator. Except for the search for a deployment pattern for the eavesdropping nodes, we also tried to automatically develop a deployment pattern for legal nodes that maximize the number of secure links when the particular secrecy amplification protocol is executed. The basic settings were similar to search for an eavesdropping pattern, the vector of node's positions was used to encode a deployment pattern for legal network nodes. Evolutionary algorithms quickly provided a very good solution – the deployment pattern with almost no compromised links was found, regardless of the number of eavesdropping nodes. This successful pattern was very simple – all nodes were positioned were close to each other almost in one single point, rendering resulting links very short and therefore hard to compromise by eavesdropping. Obviously, such solution does not fulfil the original target of a network owner to sense uniformly deployment field (implicit assumption). An exploit of such incomplete specification is the common problem when working with simulated environment and automatic search. Provided results should be therefore rigorously verified, e.g., by human analysis and separate simulation code.

4.5 Conclusions for attack strategies

We have proposed novel concept for automatic generation of attack strategies based on combination of evolutionary algorithms and our network simulator. The main advantage of the approach is possibility to test and find well working attacks in close to real or even real usage.

The usability of proposed concept was verified on two attack vectors for wireless sensor networks, but is not limited only to this area. Firstly, well performing pattern for the deployment of eavesdropping nodes was developed as an attack against Key Infection plaintext key distribution [3] having roughly twice as many compromised links than random deployment. Secondly, several variations of attacks based on selective node capture were examined. Approximately 50-70% increase in the number of compromised links was obtained with respect to random node capture (for the whole network) or 25-30% decrease in the number of necessary nodes to capture lowering cost of attack. These two groups of attacks are examples of automatic optimization of known attacks.

Chapter 5

Concluding remarks

In this thesis, we focused on the issues related to link key security in wireless sensor networks. We examined both defender and attacker side and focused mainly on the automated methods for generation of proper strategies as a way how to overcome the problem of wide range of different assumptions currently made about these networks.

We first survey the probabilistic pre-distribution schemes suitable for use in the memory-and energy-restricted environment of wireless sensor networks. We then focus on the aspect of the resilience of key pre-distribution schemes against node capture, proposing an extension protocol in Chapter 2. This protocol utilizes group support from direct neighbours to provide authenticated key exchange with significantly better node capture resilience than that of an underlying probabilistic pre-distribution scheme. A large virtual key ring is created from neighbours' keys and is maintained in an efficient way with a low communication overhead. Node capture resilience for two probabilistic pre-distribution schemes (EG scheme [21] and multi-space polynomial scheme [19]) is analyzed and simulated. Approximately up to 10000 captured nodes can be tolerated in dense networks with 40 neighbours and key ring memory is able to store up to 200 keys when multi-space polynomial pre-distribution as an underlying scheme is used. This work was published in article [60] as a version with the Eschenauer and Gligor probabilistic pre-distribution and in [58] with analytical results. Combination of the protocol with multiple key spaces pre-distribution by Du et al. was published as book chapter in [59].

The third chapter deals with the issue of localized secrets as a preventive defence against the Sybil-like attacks. Localized secrets are introduced by a secrecy amplification mechanism that propagates new keys over the multiple paths involving network neighbours in a specified way. Moreover, such protocols are able to secure links that were previously compromised by an attacker.

In Section 3.1, we provided detailed simulations of published secrecy amplification protocols and one protocol proposed by ourselves with respect to different network densities, different number of eavesdropping nodes and combination of separate secrecy amplification protocols. Our simulations provided several noteworthy results which can be used for planning of networks configuration. Secrecy amplification protocols generally work better for higher number of neighbours (denser networks) until significant presence (15-20%) of eavesdropping nodes occur. Such density can be temporarily obtained by increasing radio transmission range just for the time of secrecy amplification. Secrecy amplification protocols should be repeated several times (around 2-4 times) as links secured in previous iteration help to secure new links in the following iteration. Secrecy amplification protocols that require intermediate nodes should be combined with protocols without intermediates if network is sparse by the nature or average number of neighbours fluctuates as intermediates might be out of transmission range of involved nodes. A proper composition provides significant improvement of number of secured links. This work was published in [16], with background for assumption of availability of reliable random number generator for secrecy amplification published in [32] and [31].

The success rate of separate eavesdropping nodes is inspected with the conclusion of highly non-uniform compromise pattern for Key Infection key distribution approach. With this result on mind, we extended the secrecy amplification protocols from Key Infection to the more researched area of probabilistic pre-distribution in Section 3.5. The different key distribution approaches result in the different compromise patterns when attacker captures some nodes and extracts their secrets or eavesdrop communication. The performance (number of secured links) of a particular amplification protocol may vary between such patterns. Human design of an efficient protocol without unnecessary steps for a particular pattern is time consuming. We proposed a framework for automatic generation of personalized amplification protocol with effective-only steps. Evolutionary algorithms are used to generate candidate protocols and our network simulator then provides metric of success in terms of secured links.

The approach was verified on two compromise patterns that arise from Key Infection approach and probabilistic key pre-distribution. For these patterns, all published protocols we were aware of at the beginning of our work (described in Section 3.1) were rediscovered and a new protocol that outperforms them was found. More than 90% of secure link can be obtained after a single run of secrecy amplification protocol even in a network with half of compromised links. We also demonstrated that secrecy amplification protocols are not limited only to relative specific plaintext Key Infection approach distribution model [3]. According to our simulations, secrecy amplifications actually works even better with much more researched probabilistic pre-distribution schemes [21, 19, 34] providing a higher increase in the number of secured links.

The practical disadvantage of established design of secrecy amplification protocols (so called node-oriented) is a significant communication overhead, especially for dense networks. In Section 3.7, we propose a group-oriented design, where possibly all direct neighbours can be included in a single protocol run. Only a very small fraction of messages is necessary to obtain a comparable number of secured links with respect to the node-oriented design. Moreover, a linear increase of necessary messages instead of exponential increase with increasing density of the network is obtained. This makes our approach practically usable for networks where energy-expensive transmissions should be avoided as far as possible. Automatic generation framework was used to generate well performing protocol in the message restricted scenario. The work on automatic generation of secrecy amplification protocols was published in article [57] and as a chapter in book [50].

We explore the possibility for automatic generation of attacks in the fourth chapter. The idea is based on fundamental asymmetry between the attacker and the defender, where the attacker needs to find only one attack path where the defender must secure all of them. A brute-force search over the space of possible attack paths is then more suitable approach for the defender. An attacker can make an informed search for possible attack without inspecting all possibilities. The general concept uses some generator of candidate attacks from elementary actions and execution environment to evaluate the impact of the candidate attack. This concept cannot be used for any type of attacks – the existence of some metric to evaluate attack success in a reasonable time is necessary. In our case, we used evolutionary algorithms as the candidate attacks generator and the network simulator to evaluate them. We focused on two applications relevant in context of this thesis to provide proof of applicability of proposed concept.

An optimal deployment pattern for eavesdropping nodes for Key Infection key distribution with applied secrecy amplification was inspected first. The vector of position for eavesdropping nodes (deployment pattern) was generated by evolutionary algorithms and immediately simulated. The deployment patterns with a significant improvement in the number of compromised links (attacker goal) with respect to random placement of eavesdropping nodes were found. The biggest improvement was obtained when an attacker has a priori knowledge about position of network nodes is known in advance, as one can expect.

The second inspected application was a search for the optimal selective node capture against network with probabilistic key pre-distribution and with applied secrecy amplification protocol. The size of the network prohibits the brute-force examination of all possible subsets even for small number of selectively captured nodes to find optimal solution. The vector with fixed size provides identifications of the nodes that should be selective captured. Keys are extracted from selected nodes and resulting number of compromised links in the network was evaluated. We compared automatically found set of nodes with sets generated at random, set by an algorithm that tries to maximize the number of unique keys captured from nodes and by algorithm that greedily tries to maximize the number of compromised links per single node. The keys maximization algorithm provides only moderate improvement with respect to random capture, but the links maximization algorithm performs significantly better. The automatically found set according to our concept provided a slightly better result than the link maximization algorithm for probabilistic pre-distribution [21] with one key threshold required for link establishment ($1-EG$) and a significantly better result when applied to three keys threshold ($3-EG$,[10]). It is still possible that a better node capture algorithm can be developed for each particular scenario, but our automated concept provides much faster and flexible solution – if the pre-distribution and impact of node capture can be simulated then automated search can be run with well-performing results usually found.

Significant work was done in the development of the fast network simulator used as the source of simulation results for issues tackled in this thesis. We decided to develop new simulator instead of using an existing one as the speed of simulation is critical for automated search for new secrecy amplification protocols and attacker strategies where testing of hundreds of thousands of candidate instances must be performed in a reasonable time period. Source code of the simulator is publicly available, but we do not intend to provide a general purpose simulator. Instead, we focus on high speed and possibility for distribution of computation over a network of computers. Yet, some personalization of the simulator is possible without changing its code – namely specification of network distribution pattern and specification of arbitrary secrecy amplification protocol in metalanguage, provided in the form of an interpreted script.

Our work was supported by research grants from Grant Agency of Czech Republic (Integrated approach to education of PhD students in the area of parallel and distributed systems, Cryptographic random and pseudo-random number generators), National Security Authority (R&D Projects in Cryptography and Cryptanalysis) and European 6th framework programme (NoE Future of Identity in the Information Society).

5.1 Future research opportunities

The problem we encountered with group-supported probabilistic protocol is the dependence of the key establishment schemes on some node replication detection scheme. Such scheme must work efficiently in highly decentralized networks but with low communication overhead. Approaches described in or [43] or probabilistic detection described in [45] provide a partial solution, but with communication overhead quickly increasing with the increasing network size. A combination of replication detection with limitation of area where duplicated nodes are of some use from the attackers point of view may provide solution. Such limitation can be introduced based on the geographic limitation of valid node occurrence area (a node will be rejected if try to connect in part of the network other than assigned geographic area) or position in the network hierarchy (node is rejected outside assigned virtual cluster). A key distribution technique with only locally usable keys like the Key Infection approach [3] can be used as well, but possibility for redeployments and interconnection with existing nodes is limited. We encourage studying efficient solutions based on probabilistic rather than deterministic detection

The possibility for node authentication in the context of symmetric cryptography is usually perceived as viable only if the key used for authentication is known to no other nodes besides the authenticating and verifying node. Such assumption is common even in works that explicitly deal with partially compromised networks and probabilistic protocols [3, 41]. The notion of probabilistic authentication in the context of wireless sensor network was used in [25] for message authentication and in [43, 58] for node authentication. We expect that the difference between compromised and incorrectly authenticated node is negligible for many scenarios. And if the network is able to properly function when partially compromised, it should be possible to function when a limited number of malicious nodes are incorrectly authenticated. Is it really necessary to authenticate separate nodes, especially when prevention of the Sybil attack is such a hard task? We propose to study schemes, where the probability of successful authentication decreases with increasing number of cloned nodes that use (part of) particular authentication credentials. One of the possible direction might be usage of anonymous money [14] where multiple spending of same "coin" leads to loss of anonymity. Multiple exposures of the same authentication credentials (by Sybil or cloned nodes) then reveal secret information usable to blacklist cloned node. The scheme should be adapted to decentralized nature of sensor networks and should be based on the probabilistic rather than deterministic detection with relaxation of the memory storage requirements necessary to detect multiple credentials exposure at the same time.

We expect to see a wider application of secrecy amplification protocols as the next layer of defence. Secrecy amplification protocols proved to increase overall network security substantially, at least in both inspected compromise patterns. A highly insecure network with half of its links compromised can be turned into a reasonably secure network with more than 90% of links secure – node capture resilience for probabilistic schemes is then significantly improved. Future analysis of new key distributions schemes should discuss not only the direct impact on the network security when certain number nodes are captured, but also how many links remain compromised after the application of secrecy amplification protocol. For example, more advanced threshold cryptography probabilistic schemes like [19] gain less from secrecy amplification than simpler schemes like [21], because networks with the threshold scheme have very good resilience alone against node capture until critical number of nodes is captured. After this threshold network quickly become completely insecure with almost all links compromised, situation where secrecy amplification protocols cannot operate successfully.

We see the potential of highly decentralized operations coordinated within group of neighbours as a basic principle for governing for the distributed networks with lack of centralized control, especially when an economic-based approach to network security is applied and part of the network is assumed to be compromised. The way that separate nodes in the network should cooperate (protocols) to accomplish certain tasks can be very simple or very complicated, depending on various aspects like the degree of decentralization, available information about the operating environment and especially solved problems. On one extreme, we can see very simple rules of behaviour, with interaction only with direct neighbours in the case of cellular automata [49], still resulting in very complex global behaviour. If properly designed, such systems may provide significant resilience against intentional or random failure and provide the overall robustness of the network. However, the communication overhead to accomplish certain task is usually substantial with respect to centralized or hierarchical systems. This becomes an issue, when applied to a scenario with energy-limited nodes. The large-scale wireless sensor networks based on devices like smart dust [2] are step in this direction – nodes are energy-limited but should work at least partially in a decentralized fashion. Such nodes are still significantly more powerful than simple cells in case of cellular automata. Yet efficient combination of possible node actions can be hard for a human designer. The automatic design and optimization of rules may provide a flexible way of obtaining near-optimal parameters of network behaviour for a particular task, like the combination of evolutionary algorithms with cellular automata [28].

A wide range of future research directions is possible for an attacker strategies generator based on evolutionary algorithms. We developed a working framework and tested few initial scenarios that confirmed the usability of the framework. In general, the application of the approach to optimization problems provides usually usable results, but the real appeal is in the automated search for novel attacks rather than improvement of existing attacks. We preliminarily probed this option for attacks against geographic routing protocols. Two types of results were obtained. Several already known attacks principles were rediscovered, including base station impersonation, beacons forging, selective message forwarding and dropping, and selective collisions on MAC layer and overloading of neighbours' message buffers. Economic tradeoffs based attacks, where only a very small fraction of malicious nodes was able to affect majority of communication, were especially successful.

A combination of the attack generator (either random or based on informed search like evolutionary algorithms) with a real system instead of the simulator is possible. The only requirement on the real system is the existence of an accurate and relatively stable fitness function with reasonable speed of evaluation. We see the potential in areas such as attacks against routing in real network architectures, bypassing intrusion detection systems (already researched [48, 23, 33]) or artificial manipulation of person characteristics (e.g., reputation status) in massive social networks. The fundamental advantage of work with a real system is the absence of the necessity for the abstraction of the system realized in a simulator and the potential to work out the attacks both on the design and the implementation level.

Another appealing idea is a continuous evolution of an attacker strategy as a response to the variability of the environment and applied defences in real systems. Such an approach is commonly used for real-time evolution in embedded devices, e.g., software filters for recognition of speed limit signs continuously adapted to actual weather conditions [55]. In fact, this is the behaviour seen in gross granularity in the never-ending confrontation between attackers and defenders like virus creators with antivirus companies. Instead of having a fixed scenario and a well-performing attacker strategy for it, an attacker can run a continuous search for new strategies and use the one that is performing best at a given time. Such fine-grained approach can underpin even subtle changes in network topologies, fluctuation in the network load or the improvement of defence strategies.

Bibliography

[1] Crossbow Technology, Inc. *http://www.xbow.com/ [Last Access: 2008-08-31].*

[2] Smart dust project website. *http://robotics.eecs.berkeley.edu/~pister/SmartDust/ [Last Access: 2008-08-31].*

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

[4] Tuomas Aura, Pekka Nikander, and Jussipekka Leiwo. Denial of service in sensor networks. *IEEE Computer, Issue 10*, pages 54–62, 2002.

[5] Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, and Frank D. Francone. *Genetic Programming – An Introduction.* Morgan Kaufmann Publishers, San Francisco, CA, 1998.

[6] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. pages 1–15. Springer-Verlag, 1996.

[7] Peter J. Bentley. *Evolutionary Design by Computers.* Morgan Kaufmann Publishers, San Francisco, CA, 1999.

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

[9] Shaobin Cai, Xiaozong Yang, and Jing Zhao. Mission-guided key management for ad hoc sensor network. *PWC 2004, LNCS 3260*, page 230237, 2004.

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

[11] Haowen Chan, Adrian Perrig, and Dawn Song. *Key distribution techniques for sensor networks.* Kluwer Academic Publishers, Norwell, MA, USA, 2004. ISBN 1-4020-7883-8.

[12] Siu-Ping Chan, Radha Poovendran, and Ming-Ting Sun. A key management scheme in distributed sensor networks using attack probabilities. *GLOBECOM 2005, St. Louis, USA*, 2005.

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

[14] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In *CRYPTO '88: Proceedings on Advances in cryptology*, pages 319–327, New York, NY, USA, 1990. Springer-Verlag New York, Inc. ISBN 0-387-97196-3.

[15] Hyun-Jin Choi. Security protocol design by composition. In *University of Cambridge, technical report UCAM-CL-TR-657, GB*, 2006.

[16] Dan Cvrček and Petr Švenda. Smart dust security - key infection revisited. *Security and Trust Management 2005, Italy, ENTCS*, pages 10–23, 2005.

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

[18] Jing Deng, Carl Hartung, Richard Han, and Shivakant Mishra. A practical study of transitory master key establishment for wireless sensor networks. In *SECURECOMM '05: Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks*, pages 289–302, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2369-2.

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

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

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

[22] Konstantinos P. Ferentinos and Theodore A. Tsiligiridis. Adaptive design optimization of wireless sensor networks using genetic algorithms. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 51(4):1031–1051, 2007.

[23] Prahlad Fogla and Wenke Lee. Evading network anomaly detection systems: formal reasoning and practical techniques. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 59–68, New York, NY, USA, 2006. ACM. ISBN 1-59593-518-5.

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

[25] Ashish Gehani and Surendar Chandra. Past: Probabilistic authentication of sensor timestamps. In *ACSAC '06: Proceedings of the 22nd Annual Computer Security Applications Conference on Annual Computer Security Applications Conference*, pages 439–448, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2716-7.

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

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

[28] Rajarshi Das James P. Crutchfeld, Melanie Mitchell. The evolutionary design of collective computation in cellular automata. In *Evolutionary Dynamics, Exploring the Interplay of Selection, Neutrality, Accident, and Function.* New York: Oxford University Press, 2002.

[29] Yong Ho Kim, Mu Hyun Kim, Dong Hoon Lee, and Changwook Kim. A key management scheme for commodity sensor networks. *ADHOC-NOW 2005, LNCS 3738*, pages 113–126, 2005.

[30] J. R. Koza, F. H. Bennett III., D. Andre, and M. A. Keane. *Genetic Programming III: Darwinian Invention and Problem Solving.* Morgan Kaufmann Publishers, San Francisco, CA, 1999.

[31] Jan Krhovják, Petr Švenda, and Vashek Matyáš. The sources of randomness in mobile devices. In *Proceeding of the 12th Nordic Workshop on Secure IT System*, pages 73–84. Reykjavik University, October 2007.

[32] Jan Krhovják, Petr Švenda, Vashek Matyáš, and Ludek Smolík. The sources of randomness in smartphones with Symbian OS. In *Security and Protection of Information 2007*, pages 87–98. University of Defence, May 2007.

[33] Jong-Keun Lee, Min-Woo Lee, Jang-Se Lee, Sung-Do Chi, and Syng-Yup Ohn. Automated cyber-attack scenario generation using the symbolic simulation. In *AIS 2004*, pages 380–389, 2004.

[34] Donggang Liu and Peng Ning. Establishing pairwise keys in distributed sensor networks. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 52–61, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-738-9.

[35] Donggang Liu and Peng Ning. Location-based pairwise key establishments for static sensor networks. *1st ACM Workshop Security of Ad Hoc and Sensor Networks Fairfax, Virginia*, pages 72–82, 2003.

[36] Frederic Massicotte, Francois Gagnon, Yvan Labiche, Lionel Briand, and Mathieu Couture. Automatic evaluation of intrusion detection systems. In *ACSAC '06: Proceedings of the 22nd Annual Computer Security Applications Conference on Annual Computer Security Applications Conference*, pages 361–370, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2716-7.

[37] Catherine Meadows. Formal methods for cryptographic protocol analysis: emerging issues and trends. In *IEEE Journal on Selected Areas in Communications, Volume 21, Issue 1*, pages 44–54, 2003.

[38] Ralph C. Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21(4):294–299, 1978.

[39] Julian Miller and Peter Thomson. Cartesian Genetic Programming. In *Proc. of the 3rd European Conference on Genetic Programming EuroGP2000*, LNCS 1802, pages 121–132. Springer-Verlag, 2000.

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

[41] Tyler Moore. Cooperative attack and defense in distributed networks. In *University of Cambridge, Technical report UCAM-CL-TR-718*. University fo Cambridge, 2008.

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

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

[44] Stephan Olariu and Ivan Stojmenoviĉ. Design guidelines for maximizing lifetime and avoiding energy holes in sensor networks with uniform distribution and uniform reporting. In *Proceedings of the 25th IEEE Conference on Computer Communications (INFOCOM'06)*, 2006.

[45] Bryan Parno, Adrian Perrig, and Virgil Gligor. Distributed detection of node replication attacks in sensor networks. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy (SP'05), Washington, DC, USA*, pages 49–63, 2005. ISBN 0-7695-2339-0.

[46] Adrian Perrig, Robert Szewczyk, J.D. Tygar, Victor Wen, and David E. Culler. Spins: Security protocols for sensor networks. *Wireless Networks 8/2002, Kluwer Academic Publishers*, pages 521–534, 2002.

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

[48] Shai Rubin, Somesh Jha, and Barton P. Miller. Automatic generation and analysis of nids attacks. In *ACSAC '04: Proceedings of the 20th Annual Computer Security Applications Conference*, pages 28–38, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2252-1.

[49] Joel L. Schiff. *Cellular Automata: A Discrete View of the World.* Wiley & Sons, Ltd, 2008. 0-470-16879-X.

[50] LukáŠ Sekanina, Zdeněk Vašíček, Richard Ružička, Michal Bidlo, Jiří Jaroš, and Petr Švenda. *Evolucni hardware – in final preparation.* Academia, Praha, Czech Republic, 2009.

[51] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M. Wing. Automated generation and analysis of attack graphs. In *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy*, page 273, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7695-1543-6.

[52] Dawn Xiaodong Song, Sergey Berezin, and Adrian Perrig. Athena: A novel approach to efficient automatic security protocol analysis. *Journal of Computer Security*, 9(1/2):47–74, 2001.

[53] Piotr Szczechowiak, Leonardo B. Oliveira, Michael Scott, Martin Collier, and Ricardo Dahab. Nanoecc: Testing the limits of elliptic curve cryptography in sensor networks. In *LNCS 4913*, pages 305–320, 2008.

[54] Adrian Thompson. *Hardware Evolution: Automatic design of electronic circuits in reconfigurable hardware by artificial evolution.* Distinguished dissertation series. Springer-Verlag, 1998. ISBN 3-540-76253-1.

[55] Jim Torresen, W. Jorgen Bakke, and Luk Sekanina. Recognizing speed limit sign numbers by evolvable hardware. *Lecture Notes in Computer Science*, 2004(3242):682–691, 2004.

[56] Harald Vogt. Exploring message authentication in sensor networks. *ESAS 2004, LNCS 3313*, pages 19–30, 2005.

[57] Petr Švenda. Automatic construction of secrecy amplification protocols. *3th Workshop Mathematical and engineering methods in computer science, MEMICS 2007*, 2007.

[58] Petr Švenda and Václav Matyáš. Authenticated key exchange with group support for wireless sensor networks. *The 3rd Wireless and Sensor Network Security Workshop, IEEE Computer Society Press. Los Alamitos, CA*, pages 21–26, 2007. ISBN 1-4244-1455-5.

[59] Petr Švenda and Václav Matyáš. *From Problem to Solution: Wireless Sensor Networks Security (chapter in book).* Nova Science Publishers, New York, USA, 2008. ISBN 978-1-60456-458-0.

[60] Petr Švenda and Martin Osovský. A forward onion encryption scheme for wireless sensor networks. *MEMICS 2005*, pages 38–44, 2005.

[61] 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.

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

[63] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. Leap: efficient security mechanisms for large-scale distributed sensor networks. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 62–72, New York, NY, USA, 2003. ACM. ISBN 1-58113-738-9.