Evolutionary Design of Message Efficient Secrecy Amplification Protocols

Tobiáš Smolka[1], Petr Švenda[1], Lukáš Sekanina[2], and Vashek Matyáš[1⋆]
{xsmolka,svenda,matyas}@fi.muni.cz sekanina@fit.vutbr.cz

[1] Masaryk University, Faculty of Informatics, Czech Republic
[2] Brno University of Technology, FIT, IT4Innovations Centre, Czech Republic

Abstract. Secrecy amplification protocols are mechanisms that can significantly improve security of partially compromised wireless sensor networks (e.g., turning a half-compromised network into the 95% secure one). The main disadvantage of existing protocols is a high communication overhead increasing exponentially with network density. We devise a novel family of these protocols exhibiting only a linear increase of the communication overhead. The protocols are automatically generated by linear genetic programming (LGP) connected to a network simulator. After a deep analysis of various characteristics of this new family of protocols, with a special focus on the tuning of LGP parameters, new and better group-oriented protocols are discovered by LGP. A multi-criteria optimization is then utilized to further reduce the communication overhead down to 1/2 of the original amount while maintaining the original fraction of secure links.

1 Introduction

Wireless sensor networks (WSNs) are networks of resource-constrained battery-powered nodes that can communicate over short distances via wireless radio. The applications of such networks vary from environment monitoring to battlefield management and often require resistance against unauthorized reading, modification or generation of monitored information. To achieve this goal, encryption and message authentication techniques with shared symmetric keys between the communicating parties can be used. This is, however, a challenging task, since the nodes are usually distributed in an untrusted environment. An attacker can extract all keys from a physically captured node and easily intercept large fraction of communication in the network.

Secrecy amplification (SA) is a post-deployment technique for improving the security of communication in partially compromised networks. It can be employed in situations, where there are keys established between the nodes in the network, but some of them may be compromised. SA exploits the fact that a group of neighbouring nodes can cooperate and establish a new key derived

⋆ Final work on this paper undertaken as a Fulbright-Masaryk Visiting Scholar at Harvard University.

from multiple old keys. The new key will be secure in case when at least one of the old ones was not compromised. The concept was initially introduced in [2] and further improved in [6, 8].

In this work, we present newly discovered group-oriented secrecy amplification protocols with better performance than previously published node-oriented protocols. The protocols are automatically generated by linear genetic programming (LGP) [4] connected to a network simulator. We chose LGP because it is suitable for evolution of short domain-specific programs as shown by e.g. [3]. The discovery was made possible by deeply analysing and tweaking the evolutionary search and also by introducing another criterion for protocol optimization – a total number of exchanged messages. The goal of this paper is to analyze the impact of LGP parameters on the quality of evolved protocols and find better ones.

The paper is organized as follows: The next section provides a short introduction to secrecy amplification in WSNs and reviews previous work on automatic design of these protocols with Evolutionary Algorithms. Section 3 analyses the impact of different parameters such as population size, mutation and crossover rate, length of chromosome or number of generations on the performance of LGP. Section 4 describes new protocols found in long-running experiments with tuned LGP and analyses their performance and robustness. Section 5 introduces a multi-criteria optimization and shows that further reduction of totally exchanged messages in the protocols is possible. Conclusions are given in Section 6.

2 Previous work

Secrecy amplification (SA) was initially introduced in [2] for the Key Infection (KI) key establishment, in which the keys are exchanged between the neighbours in plaintext. In case an attacker places an eavesdropping node close to a legitimate one, it is able to intercept all the keys exchanged with that node. The concept of SA can also be used when the compromised links are distributed randomly. Such a compromise pattern may result from the probabilistic distribution scheme of Eschenauer and Gligor (EG) [7].

The protocols presented in [2] and [6] use an "absolute" identification of the nodes (e.g., node number `1`, `2`, `3`.) If there are k parties (nodes) in the protocol and n neighbours of node on average then one run of the protocol must be executed for all k-tuples of neighbours leading to $\binom{n}{k}$ executions per single node – a huge communication overhead. The number of totally exchanged messages increases exponentially with the number of neighbours and is significant for WSNs where 6-15 neighbours are usually assumed. We will denote such protocols as node-oriented (NO).

A different approach to the design of amplification protocols was presented in [8]. Identification of the parties in protocol is given by the relative 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 trans-

mission 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). Based on the actual distribution of the neighbours, the node closest to the indicated distance(s) is chosen as the node N_k for particular protocol run. There is no need to re-execute the protocol for all k-tuples (as for NO protocols) as all neighbours can be involved in a single execution, reducing communication overhead significantly. See [8] for a detailed description of evaluation process for group-oriented protocols.

2.1 Evolution of amplification protocols

In order to improve the fraction of secure links and to decrease the necessary communication overhead (the number of messages), a new method for automatic generation of protocols was introduced in [8]. The method utilized linear genetic programming and a network simulator for evaluation of candidate amplification protocols with resulting fraction of secure links taken as fitness value. The use of LGP is especially important in case of group-oriented protocols, since the design of such a protocol is not a trivial task and to the best of our knowledge, no human-designed group-oriented protocol was published yet.

Instruction set: Each party (a real node in network) in the protocol is modelled as a computing unit with a limited number of memory slots. Each memory slot can contain either a random value, encryption key or message. Each candidate protocol is modelled as a program composed of instructions from a specific instruction set given in Table 1. This instruction set was chosen because it enables to express all previously known amplification protocols and to utilize only operations available on real sensor nodes such as TelosB [5].

Table 1: Instruction set for amplification protocols.

NOP	No operation is performed.
RNG N_a R_i	Generate a random value on node N_a into slot R_i.
SND N_a N_b R_i R_j	Send a value from R_i on node N_a to slot R_j on N_b.
CMB N_a R_i R_j R_k	Combine values from slots R_i and R_j on node N_a and store the result to R_k (e.g., cryptographic hash function like SHA-3).
ENC N_a R_i R_j R_k	Encrypt a value from R_i on node N_a using the key from R_j and store the result to R_k.
DEC N_a R_i R_j R_k	Decrypt a value from R_i on node N_a using the key from R_j and store the result to R_k.

Using this set of primitive instructions, a simple plaintext exchange of new key can be written as {RNG N_1 R_1; SND N_1 N_2 R_1 R_1;}, a PUSH protocol [2] as

{RNG N_1 R_1; SND N_1 N_3 R_1 R_1; SND N_3 N_2 R_1 R_1;}, a PULL protocol [6] 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 [6] 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;}. All these protocols are node-oriented. Group-oriented protocols are longer and more complicated, see [8].

Previous LGP results: LGP rediscovered previously published protocols and also new and better performing protocols were found. The best performing node-oriented 4-party secrecy amplification protocol found [8] consists of 10 effective instructions with performance shown in Figure 3. Note that the LGP objective was to optimize the number of secure links only, not the number of messages. Additionally, only limited computing resources were available to obtain these results. We will show in next sections that better protocols (in terms of the number of secure links and messages needed) can be obtained with improved LGP settings and more computational resources.

3 LGP tuning and exploring the design space

This section describes the initial version of LGP and network simulator, together with heavily resource-consuming experiments conducted to determine the most suitable parameters of LGP that are necessary for finding new group-oriented protocols in Section 4. Distributed computation via BOINC (Berkeley Open Infrastructure for Network Computing) [1] with around 250 CPU cores was used to provide the performance necessary for all experiments[3].

3.1 Experimental setup

Basic LGP setup: LGP operates with the instruction set given in Table 1. The size of chromosome is 100 instructions. Every node contains 12 memory slots. The initial population is generated randomly. The mutation operator randomly picks an integer and generates a new value at its position. The crossover operator is applied at the level of instructions. A new population is formed using a tournament selection. The fitness function is defined as a fraction of secure communication links. The impact of various parameters of LGP on the performance is investigated in Section 3.2.

Network simulation: Candidate protocols are simulated in a network of 100 legitimate nodes. During the evaluation, each amplification protocol was independently executed on 5 deployments, each of which with different placement of the nodes. This way the candidate protocol was prevented from optimizing on one particular network deployment and provided results usable also in networks

[3] Raw data in searchable format from all experiments are available at web page http://www.fi.muni.cz/~xsvenda/papers/EuroGP2012/ and additional experiments with examples of protocols found will be available in parallel technical report.

with a higher number of nodes (we kept the number of nodes intentionally low so network simulation is executed fast enough). The nodes were always placed uniformly over a square area and each node had approx. 10 legitimate neighbours on average (over all deployments).

The fraction of initially secured links was intentionally set to 30%, so it is reasonably difficult to increase the fraction of secure links. In the EG compromise pattern, this number was used directly when deciding whether the link was initially compromised or not. In the KI pattern [2], the portion of compromised links was affected by the number of attacker's nodes in the network. This number was determined experimentally (30 attacker nodes), so the resulting fraction of compromised links was close to the desired level.

3.2 LGP performance

Population size and mutation rate: First, the most suitable values of the population size and probability of mutation were sought. We tested 10 different combinations of the number of generations (num_{gen}) and population size (pop_{size}) for 20 different mutation probabilities. The values of num_{gen} and pop_{size} were chosen so that value $num_{gen} * pop_{size}$ and consequently also the number of fitness evaluations remains constant ($num_{gen} * pop_{size} = 40\ 000$). Each combination of parameters (p_{mut}, num_{gen}, pop_{size}) was instantiated in 20 independent runs and the averages of the best fitness values are shown in Figure 1a and Figure 1b. Clearly, LGP performs better for small probability of mutation (between 0.005 and 0.05) and smaller population sizes (between 10 and 20). Values $p_{mut} = 0.02$ and $pop_{size} = 15$ were used in next experiments based on these results.

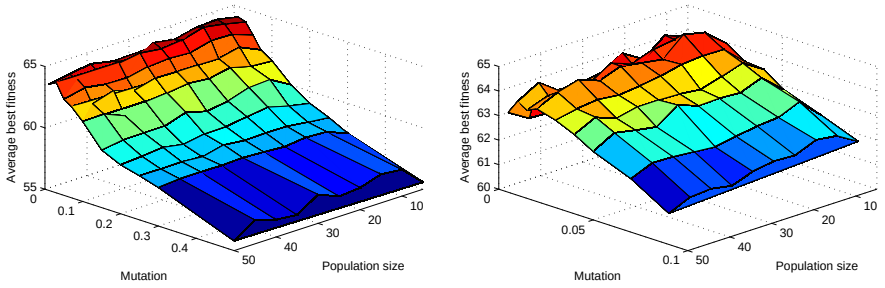


Fig. 1: Average of the best fitness values calculated from 20 independent runs using different mutation probabilities and population sizes: (a) all experiments, (b) zoomed.

Similarly, the effect of crossover (p_{cross}) was investigated using 20 independent runs where p_{cross} was between 0 and 1. This experiment showed that the

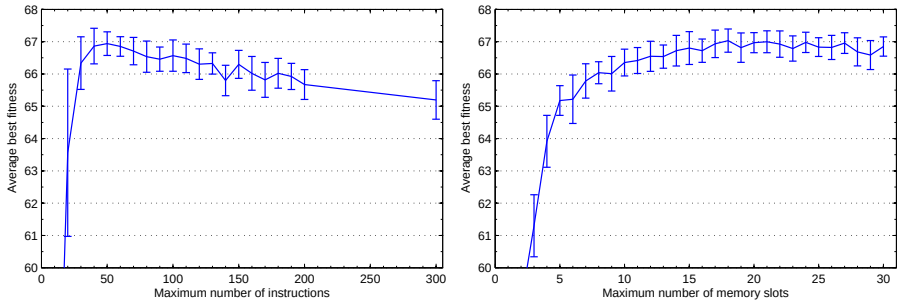
crossover has no significant impact on performance and thus crossover has not been used in other experiments.

Chromosome length: Long chromosomes imply large search spaces that are usually difficult to search. They also lead to long programs that can take a considerable time to be executed. Previous work utilized 200 instructions; however only around 10 instructions were effectively used in evolved protocols [8]. In order to find a reasonable value for the maximum length of chromosome (ins_{max}), we fixed all the parameters except ins_{max} and num_{gen}. The values for ins_{max} and num_{gen} were chosen such that the value $ins_{max} * num_{gen}$ remains constant and equals to 1 200 000 (and therefore, the overall computational time is constant for different runs).

The average fitness value computed using the best fitness values obtained from 20 independent runs is depicted in Figure 2a. The experiment shows a sharp drop in the achievable fitness when less than 30 instructions ins_{max} are available. The performance is increasing for values of ins_{max} between 40 and 100 and slowly decreasing with bigger ins_{max} afterwards. This result is correlated with observed average number of effective instructions (i.e., the instructions that actually contribute to the fitness value, usually around 30, in our case 32 instructions at maximum) in best protocols such as EG_{best} evolved with $ins_{max} = 100$ or more. Note that doubling the number of available instructions ins_{max} will roughly double the time necessary to simulate a single protocol, but it will not impact the resulting protocol as usually only around 30 instructions are effective and we can automatically identify them. On the basis of these results, the following experiments were initialized to allow 50 (100 in some cases) instructions at maximum. The evaluation time has been significantly reduced without affecting the quality of evolved protocols.

Memory slots: In the next experiment we analyzed the impact of limiting the number of memory slots ($slot_{max}$) which the evolved protocols can use. LGP parameters remain identical to the previous experiment ($ins_{max} = 100$). The average fitness values are depicted in Figure 2b for $slot_{max} = 1 \ldots 30$. The experiment revealed that the protocols require at least 10 memory slots to achieve a reasonable performance.

Number of generations: The fitness increases with additional generations, but only to some extend. In order to determine the number of generations needed for reaching a reasonably performing protocol, 20 runs were executed for 53 340 generations ($ins_{max} = 100$, $slots_{max} = 12$). In this experiment, all runs reached 60% secure links after 1 067 generations, 65% after 8 529 generations and 66% after 15 991 generations. These 20 runs reached 67.72% on average, the worse one stagnated at 66.8%. In comparison, 60% secure links is the level which no random search was able to reach even after 14 000 generations (assuming the same population size).



(a) Number of instructions (b) Memory slots

Fig. 2: Best fitness obtained from 20 independent runs for various limits in the number of instructions (a) and memory slots (b).

4 Discovering new group-oriented protocols

The search for new protocols utilized the best-performing parameters found in previous experiments. Complete setting is given in Table 2. For each of the compromise scenarios (KI, EG), we performed 20 independent runs and allowed a sufficient number of generations. The best performing protocol for each scenario was then taken and further analysed.

4.1 Long-running experiments

The best protocols are denoted as KI_{best} and EG_{best}. Protocol KI_{best} was discovered after 125 hours of computation in 330 641 generations (note that one generation required 1.4 seconds on a single 3000+ MHz core). EG_{best} was found after 87 hours in 165 365 generations (1.9 sec/generation on a single 3000+ MHz core). The protocols exhibit 69.12% (KI_{best}) and 60.07% (EG_{best}) secured links on average across 5 deployments they were trained for.

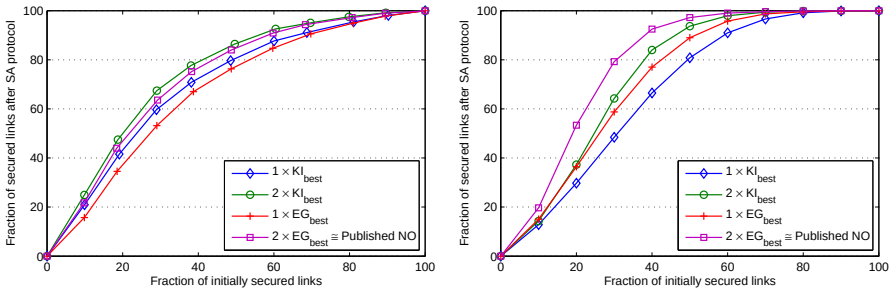
4.2 Performance of evolved secrecy amplification protocols

The protocols KI_{best} and EG_{best} performed very well in the deployment(s) they were trained for. However, in order to get more accurate estimate of protocols' performance, one needs to test them on different deployments. We did so by evaluating each of the protocols in 100 random networks for each of 9 levels of compromise (10% ... 90% stepped by 10%), two compromise patterns (KI, EG) and three different average numbers of legitimate neighbours (5, 10, 15). One additional repetition of amplification was also tested. In total, we evaluated each of the protocols in 10 800 independent scenarios.

The comparison of average reached fraction of secured links after secrecy amplification in networks with 10 neighbours are shown in Figure 3. Two amplification repeats of discovered protocols have almost identical performance to our

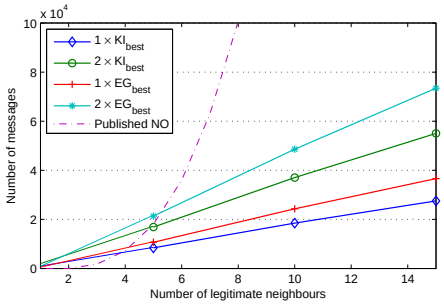
		KI	EG
Simulator	Number of deployments	5	5
	Number of legitimate nodes	100	100
	Number of malicious nodes	30	-
	Average number of legitimate neighbours	9.88	10.28
	Average number of initially secured links	32.5%	30.7%
LGP	Probability of mutation	0.02	0.02
	Probability of crossover	0.00	0.00
	Size of population	15	15
	Maximum number of instructions	50	100
	Number of memory slots	12	12
	Number of generations	1 216 000	200 000
	Average time of single run on 1 CPU	467h	106h

Table 2: Parameters of LGP and network simulator used in long running experiments.



(a) KI compromise pattern

(b) EG compromise pattern



(c) Total number of messages

Fig. 3: The performance of discovered secrecy amplification protocols. $2 \times EG_{best}$ stands for two repetitions of EG_{best}.

previously published node-oriented protocol [8]. Similar results were obtained also for 5 and 15 neighbours – with higher number of neighbours the differences between protocols' average performance are becoming larger.

The protocol evolved for KI and then used in EG performs significantly worse than the protocol evolved directly for EG scenario (and vice versa). This implies that the structure of the problem solved for KI and EG is different and a separate protocol should be evolved for different initial key distribution method. Note that such a result is not problematic for the network owner as used scenario (KI, EG or other) is known to him/her in advance and the owner can therefore select/evolve a corresponding protocol.

Note that the best performing group-oriented protocols are able to increase the fraction of secure links from 30% to almost 70% in KI or 80% in EG. In real deployment, one would usually like to achieve 85% or more secure links to provide a strong majority of secure links. Majority voting will then almost always outcompete a potential attacker. Such a percentage can be achieved when initial fraction of secure links is 40-50% in KI and 30-40% in EG scenario.

Figure 3c shows the total number of required messages for mentioned protocols. Note that the total number of messages in the group oriented protocols is independent of compromise scenario (the number of transmitted messages is always the same).

4.3 Robustness of discovered protocols

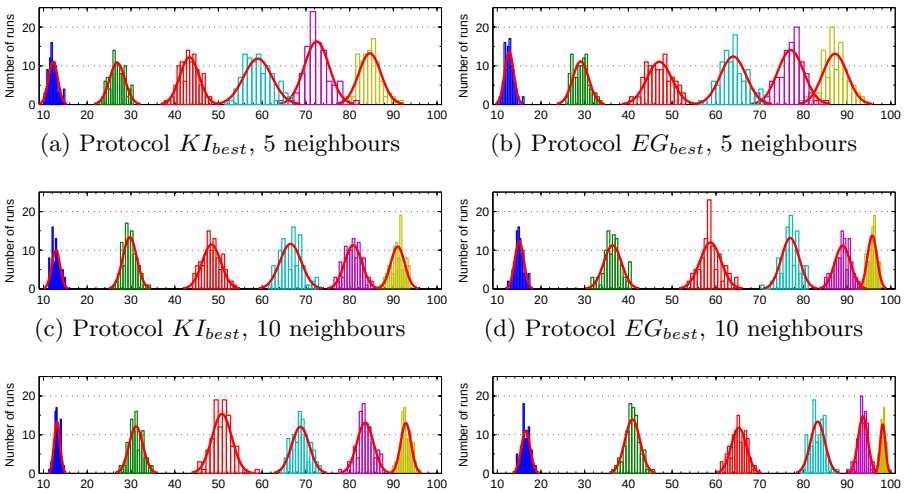
Protocol robustness against the change in underlying parameters w.r.t. parameters used during evolution is examined in this section. We focused on robustness against the change in key distribution method (KI_{best} used in EG scenario), the change in initial fraction of secure links and the change in layout of nodes in a particular deployment.

The average performance of the protocols was already shown in Figure 3. However, from the averages one can not conclude directly whether the discovered protocols are robust against changes in deployment. The EG scenario was chosen for the analysis because it allows to precisely set the fraction of initially secured links[4].

The results of the analysis are depicted in Figure 4. There are 6 histograms for each of the protocols and the average number of neighbours. Each histogram shows the distribution of resulting fraction of secure links for a given fraction of initially secured links (from left to right $10\% \ldots 60\%$).

For 100 evaluations the distributions are similar to the normal one. It can be also seen that KI_{best} has slightly worse performance than EG_{best} for all tested configurations of neighbours (note that the experiment was performed for EG scenario). With a higher number of neighbours, the differences between protocols and also the performance of individual protocols are increasing.

[4] The initial fraction of secured links in KI scenario depends on the number of attacker's nodes. Particular layout of nodes in deployment slightly varies between different deployments.



(a) Protocol KI_{best}, 5 neighbours (b) Protocol EG_{best}, 5 neighbours

(c) Protocol KI_{best}, 10 neighbours (d) Protocol EG_{best}, 10 neighbours

(e) Protocol KI_{best}, 15 neighbours (f) Protocol EG_{best}, 15 neighbours

Fig. 4: Fraction of secured links reached for a given fraction of initially secured links (histograms represent from left to right 10% . . . 60%).

5 Multi-criteria optimization

Results presented in [8] and extended in Section 4.2 were obtained with the fitness function reflecting only the fraction of secure links without taking into account the number of exchanged messages. Although the overall number of exchanged messages is relatively small for group-oriented protocols, natural question is if this number can further be decreased by additional optimization.

5.1 Weighted fitness

The communication overhead is measured as a counterpart to fraction of messages transmitted by the protocol (lower the better) during simulation to the theoretical maximum of messages when every instruction in protocol would send one message.

We propose to combine the fraction of secure links f_1 and fraction of messages f_2 using two weighting coefficients w_1 and w_2 ($fitness = w_1 f_1 + w_2 f_2$). In order to analyze the impact of different ratios of weights, 20 independent runs were executed for multiple different weights ($90 : 10 \ldots 0 : 100$), always spanning 2 000 generations. We also set the lower bound for fraction of secured links to 50% so the evolution was forced to search only for meaningful protocols (50% can be easily achieved even by a random search).

The results of experiments are shown in Figure 5. From left to right, LGP was forced to optimize the protocols more and more for the fraction of secured links. It can be clearly seen that with any additional increase of the security in the

network, the total number of messages is also non-trivially increased. However, this increase is non-linear when the weight assigned to fraction of secure links is higher than 90. Based on this result, we decided to perform another long search for a message-optimal group-oriented protocol and compare its performance with previously found KI_{best}.

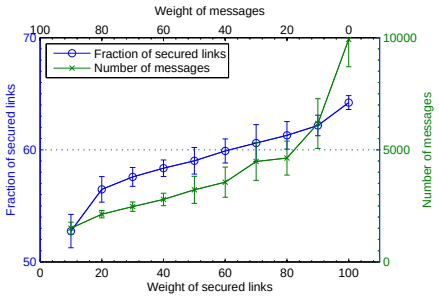


Fig. 5: Impact of criteria weights on fraction of secure links and the number of messages.

5.2 Optimizing the number of messages

Previous experiments documented a high correlation between the number of secured links and the number of messages used. Since we were interested in protocols with a reasonable security, we decided to use the weights 90 : 10 in favour of secured links (according to Figure 5). In order to find a message-optimized protocol, we executed 20 independent runs and took the best performing protocol (denoted as KI_{msg}). It was found after 90 hours in 234 000 generations.

We evaluated the performance of KI_{msg} against KI_{best} and previously published node-oriented protocols in the setup that was described in Section 4.2. Surprisingly, the evaluation over 100 different networks and different fractions of initially compromised network showed that KI_{msg} has comparable performance in KI to previously found KI_{best} protocol. More importantly, with two amplification repeats instead of one it also exhibits almost identical performance to previously published node-oriented protocols. However, since the protocol was optimized not only for security but also for low total number of messages, KI_{msg} uses only 50% of messages to achieve similar performance.

6 Conclusions

Secrecy amplification protocols turned to be one of the most promising ways how a WSN with a significant number of compromised links can be turned into

secure one for the price of additional messages exchanged. Human-designed and message intensive node-oriented version [2] of these protocols were extended by group-oriented approach in [8].

In this paper, we performed a detailed analysis of LGP in the task of evolutionary design of group-oriented protocols. By careful setting of LGP parameters, suitable setting of network simulator parameters and utilization of distributed computation, new protocols were discovered that outperform the previously published ones. The analysis of robustness of discovered protocols in scenarios different from those available during evolution confirmed that group-oriented protocols are robust against the change in the initial fraction of secure links. We have observed that these protocols are less robust against the change in selection of initial key distribution method. However, this selection is under control of the network owner who can, therefore, select/optimize the group-oriented protocol for preferred key distribution.

Additionally, we focused on further reduction of the communication overhead. A multi-criterial optimization was conducted where not only secure links but also the number of messages was optimized. It was shown that newly found group-oriented protocols outperform the node-oriented protocols while still requiring an order of magnitude less messages, e.g., $\pm$ 1/20 in common scenarios. Future work will be devoted to applying truly multi-criteria optimization algorithms such as NSGA-II and implementation of evolved protocols on real nodes.

Lukáš Sekanina was supported by the IT4Innovations Centre of Excellence CZ.1.05/1.1.00/02.0070. Other authors were supported by the GAP202/11/0422 and Centre of Excellence GAP202/12/G061 of the Czech Science Foundation. The experiments were supported by computational resources at Institute of Computer Science, Masaryk University.

References

1. Anderson, D.: Boinc: A system for public-resource computing and storage. In: Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on. pp. 4–10. IEEE (2004)
2. Anderson, R., Chan, H., Perrig, A.: Key infection: Smart trust for smart dust. In: ICNP 2004. pp. 206–215. IEEE (2004)
3. Bernardi, P., Sánchez, E., Schillaci, M., Squillero, G., Reorda, M.S.: An effective technique for minimizing the cost of processor software-based diagnosis in socs. In: IEEE Date2006: Design, automation and test in Europe. pp. 412–417 (2006)
4. Brameier, M., Banzhaf, W.: Linear Genetic Programming. Springer Verlag, Berlin (2007)
5. CrossBow: Telosb. http://www.willow.co.uk/TelosB_Datasheet.pdf [24/1/2012]
6. Cvrcek, D., Svenda, P.: Smart dust security-key infection revisited. Electronic Notes in Theoretical Computer Science 157(3), 11–25 (2006)
7. Eschenauer, L., Gligor., V.: A key-management scheme for distributed sensor networks. pp. 41–47 (2002)
8. Švenda, P., Sekanina, L., Matyáš, V.: Evolutionary design of secrecy amplification protocols for wireless sensor networks. In: Proceedings of the second ACM conference on Wireless network security. pp. 225–236. ACM (2009)