Evolutionary Design of Secrecy Amplification Protocols for Wireless Sensor Networks

Petr Švenda
Faculty of Informatics
Masaryk University
Brno, Czech Republic
svenda@fi.muni.cz

Lukáš Sekanina
Faculty of Information Technology
University of Technology
Brno, Czech Republic
sekanina@fit.vutbr.cz

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

ABSTRACT

We propose a new method for automatic generation of secrecy amplification protocols for wireless sensor networks, utilizing evolutionary algorithms. We were able to rediscover all published protocols for secrecy amplification we are aware of, and found a new protocol that outperforms the existing ones. An alternative construction of secrecy amplification protocols with a comparable fraction of secure links to that of the original "node-oriented" approach was also designed. This new construction exhibits only linear (instead of exponential) increase of necessary messages when the number of communication neighbours grows. This efficient protocol can significantly reduce the sensor battery power consumption because of the decreased message transmission rate. We used a combination of linear genetic programming and a network simulator in this work.

Categories and Subject Descriptors

C.2.4 [**Computer-communication networks**]: Distributed networks; K.6.5 [**Computing Milieux**]: Security and Protection

General Terms

Security, Performance

Keywords

Evolutionary algorithms, key establishment, secrecy amplification protocols, wireless sensor networks

1. INTRODUCTION

Advances in miniaturization of electronics creates the opportunity to build devices that are small in scale, can run autonomously using only battery power and can communicate over short distances via wireless radio. These devices can be used to form a new class of applications, Wireless Sensor Networks (WSNs). WSNs are considered for and deployed in a multitude of different scenarios such as emergency response information, energy management, medical monitoring, wildlife monitoring or battlefield management. Resource-constrained nodes pose new challenges for suitable routing, key distribution, and communication protocols. Security is often an important factor of WSN deployment, yet the applicability of some security approaches is often limited. *Terminal sensor nodes* can have little or no physical protection and should therefore be assumed as *untrusted*. Also, *network topology knowledge is limited* or unknown in advance. Due to the limited battery power, communication traffic should be kept as low as possible and most operations should be done locally, not involving the trusted base stations (BS). We focus on the basic problem of secure link key establishment, specifically on the strengthening of link security after an ordinary key establishment scheme like probabilistic pre-distribution [10] or Key Infection [1] in partially compromised networks.

This paper proposes a new method for automatic generation of secrecy amplification protocols [1] for WSNs, which utilizes linear genetic programming (LGP) [4]. The proposed framework consists of a protocol generator realized by LGP and a network simulator that computes the fitness value (quality metric) for candidate protocols. The link keys established after the execution of a secrecy amplification protocol might be then used as a building block for a broad range of application security goals like multi-hop message secrecy and authentication or secure aggregation.

The paper is organized as follows: the next section provides a short introduction to wireless sensor networks, highlights related security issues and provides overview of related work. Section 3 describes the proposed framework for automatic generation of secrecy amplification protocols and defines the elementary rules used by evolutionary algorithms to build novel protocols. Our proposed method and an example of a protocol evolved for a fixed number of parties is presented in Section 4. Section 5 describes a different approach to the protocol construction that significantly decreases the number of messages which has to be produced. Conclusions are given in Section 7.

2. WIRELESS SENSOR NETWORKS AND THEIR SECURITY

Security protocols for WSNs deal with very large networks of very simple nodes. Such networks are presumed to be deployed in large batches followed by a self-organizing phase.

WiSec'09, March 16–18, 2009, Zurich, Switzerland.

The latter is automatically and autonomously executed after a physical deployment of sensor nodes. Data sensed by nodes are collected and sent to a base station and then presented to users. To protect the content of sensed data and prevent insertion of fake information, mechanisms such as encryption and message authentication are used. These techniques usually require secret key material to be shared between communicating parties that must be kept secret from an attacker. In this work, we focus on usage of symmetric cryptography, as secrecy amplification protocols are usually based on this, even when a possibility for combination with asymmetric cryptography (e.g., for initial key exchange) exists.

A common attacker model (with respect to key management) used in the network security arena is an extension of the classic Needham-Schroeder[1] model [17] called the node-compromise model [10], described by the following additional assumptions: 1) The key pre-distribution site (if used) is trusted, i.e. before deployment, nodes can be pre-loaded with secrets in a secure environment. 2) The attacker is able to capture a fraction of deployed nodes because no physical control over deployed nodes is assumed. 3) The attacker is able to extract all keys from a captured node, i.e. no tamper resistance of nodes is assumed. The attacker model is in some cases (Key Infection [1]) additionally weakened by the assumption that 4) for a short interval the attacker is able to monitor only a fraction of links and then it reverts to being a stronger attacker with ability to eavesdrop all communication.

The assumption of no tamper resistance lowers the production cost and enables the production of a high volume of nodes. On the other hand, it requires novel approaches to security protocol design. The aim is to build a reasonably secure network even in the presence of an attacker who can obtain secrets for a part of the network by capturing some of the nodes or eavesdropping part of the key exchanges. Note that different key distribution techniques have different abilities to withstand such attacks.

2.1 Compromise patterns of key distributions

Secure link communication is the building block for most security services maintained by a WSN. Here, we focus on two types of initial key establishment. First is probabilistic key pre-distribution introduced by Eschenauer and Gligor [10] (referred to as the EG scheme) and extended later by [5, 18, 9, 14] and others with improved node capture resilience. Second is a plaintext key exchange called Key Infection with restricted attacker model introduced in [1]. Key distribution schemes behave differently when the network is attacked and partially compromised. We will focus on two types of network compromise patterns:

2.1.1 Random compromise pattern

This compromise pattern may arise when a probabilistic key pre-distribution scheme [10] and later variants are used and an attacker extracts keys from several randomly captured nodes. The EG scheme 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*. Due to the birthday paradox, the probability of sharing at least one common key between two neighbours is surprisingly high even for a key ring of relatively small size. That makes this EG scheme suitable for memory-constrained sensor nodes.

When an attacker captures several nodes, links to nodes other than those captured are potentially compromised as extracted keys from captured nodes recover a proportion of the original key pool. The probability that a given link secured by shared keys 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 other links to the same node. Note that in the case of probabilistic pre-distribution, the compromise status of links from a given node is still slightly correlated because if one link is compromised, other links from the same node may be established using the same key(s) as the compromised one. This correlation quickly decreases with the size of key ring on each node (e.g., it is negligible for 200 keys in the ring). It holds for links constructed from pre-distributed symmetric cryptography keys that if link $A \rightarrow B$ is compromised, then also $A \leftarrow B$ is compromised as the same set of keys is used.

2.1.2 Key Infection pattern

Compromised networks resulting from Key Infection distribution [1] form the second inspected pattern. Here, link keys are exchanged in plaintext (no keys are pre-distributed) and an attacker can compromise them if the transmission can be recorded by an attacker's eavesdropping device. 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 the link transmission is 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 $A \leftarrow B$ as the eavesdropping node positioned outside the virtual sphere of radio transmission range centered on node A with diameter equal to distance between A and B will not be able to compromise link $A \rightarrow B$ but still might be able to compromise link $A \leftarrow B$. This is another difference from the Random compromise pattern.

2.2 Secrecy amplification

Substantial improvements in resilience against node capture or key exchange eavesdropping can be achieved when a group of neighbouring nodes cooperates in an additional *secrecy amplification* protocol after the initial key establishment protocol. This concept was originally introduced in [1] for the Key Infection plaintext key exchange, but can be also used for a partially compromised network resulting from node capture in probabilistic pre-distribution schemes. Several secrecy amplification protocols were published.

In *multi-path key establishment*, node A generates q different random values and sends each one along a different path via node(s) C_i to node B, encrypted with existing link keys. This operation will be denoted as the PUSH protocol. All values combined together with the already existing key between A and B are used to create the new key value. An attacker must eavesdrop all paths to compromise the new key value. A second method, called *multi-hop key amplification*,

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

is basically a 1-path version of the multi-path key establishment with more than one intermediate node C_i. Simulations for attacker/legal nodes ratios of up to 5% in [1] show that plaintext key exchange followed by secrecy amplification is sufficient to achieve a network with more than 90% of secure links within this attacker model.

A variant of initial key exchange (denoted as COMODITY) without secrecy amplification was presented in [12]. Node A sends the same key K_1 to nodes B and C in plaintext. Then, K_1 is used to secure distribution of initial key material $E_{K_1}(B|K_2)$ and $E_{K_1}(C|K_3)$ between (A, B) and (A, C)[2]. The final key between (A, B) is constructed as $K_{12} = hash(K_3|hash(K_2|K_1))$

A variant of the PUSH protocol, called the PULL protocol, was presented in [8]. The initial key exchange is identical to the PUSH protocol. However, node C decides to help improving the secrecy of the key between nodes A and B instead of node A making such decisions as in the PUSH protocol. This in turn decreases the area affected by the attacker eavesdropping node and thus increases the number of non-compromised link keys (valid for Key Infection distribution).

The impact of a key composition mechanism called *mutual whispering* on subsequent amplification was also examined [8]. Mutual whispering is a key exchange where a pairwise key between A and B is constructed simply as $K_{12} = K_1 \oplus K_2$, where K_1 is the key whispered [3] from A to B and K_2 from B to A. Experimental results show that mutual whispering followed by the PUSH protocol gives us the equivalent fraction of secure links as basic whispering followed by the PULL protocol for Key Infection compromise pattern. Repeated iterations of the PULL protocols lead to a strong majority of secure links even in networks where up to 20% of nodes are the attackers' eavesdropping nodes. Note that the assumption that an attacker controls only a fraction of nodes (e.g., 10%) is reasonable, as an attacker must place his nodes before the network is deployed and therefore the density of the deployed legal network can be set to achieve the desired ratio. A detailed analysis of secrecy amplification protocols with respect to the network density and number of eavesdropping nodes is presented in [20].

The impact of PUSH, PULL, mutual whispering and new automatically derived protocols (as described in Sections 4 and 5) for Random and Key Infect compromise patterns are compared in Figure 4 and 5. 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. A combination of mutual whispering with the PUSH protocol gives the same results as the PULL protocol alone in the Key Infection pattern. See [8] for a more detailed comparison of the protocols and the impact of repeated runs of secrecy amplification (not shown here).

[2]Notation $E_{K_1}(B|K_2)$ stands for node identification (B) concatenated with value of key K_2 and resulting message encrypted (E) with key K_1.

[3]Transmission is performed with the minimal radio strength necessary to communicate between two nodes, therefore nodes more distant from the sending node are not able to hear the transmission.

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 an existing secrecy amplification protocol inefficient. As a result, a time-consuming analysis and some design effort are needed to find a new protocol.

3. PROPOSED METHOD

In this paper, 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.

3.1 Composition of simple secure protocols

Designing new protocols is a time consuming process and any resulting flaws may remain undetected for a long time. Various formal verification tools currently exist to verify the correctness of a proposed protocol (see [15] 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 [19]. 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 [7] for an a good overview of possible approaches to automatic protocol generation and protocol composition. Fortunately, this is also the case for secrecy amplification protocols because they specify the way in which fresh key values are propagated and combined by the parties involved. Thus, a secrecy amplification protocol can be viewed as a composition of a 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 [7] 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 capturing nodes or eavesdropping a fraction of all communications). 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 substituting a formal verification tool with a network simulator for faster evaluation, we additionally obtain a smoo-

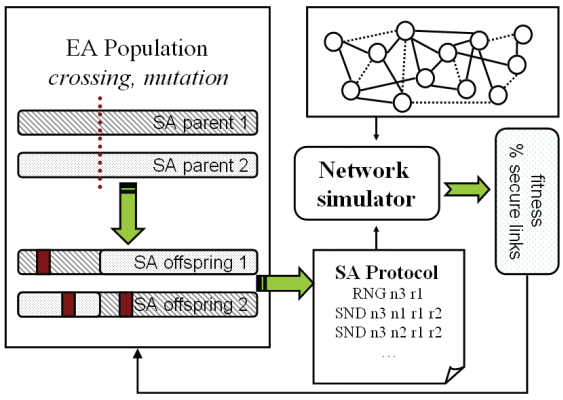


Figure 1: Automatic protocol generation process with fitness evaluation. The new population is created using crossover and mutations. Genotypes are transcribed into candidate protocols. Using the network simulator and a given partially compromised network (dotted links), the fitness value (fraction of secured links) is calculated for each candidate protocol.

ther indication how good a candidate protocol is. Instead of a binary indication "secure or flawed", we will obtain the number of links additionally secured by a particular protocol[4]. Hence we can use some kind of informed search instead of an exhaustive search.

3.2 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 do), these algorithms operate with the *population* of *candidate solutions* (candidate secrecy amplification protocols in our case). Every new population is formed by genetically inspired operators such as *crossover* (a part of protocol's instruction are taken from one parent, the rest from another one) and *mutation* (the 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 (in our analysis, the fraction of secure links), calculated by the *fitness function* (network simulator), indicates how well the solution fulfills the problem objective (improving network security). 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* [13]. In contrast to 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 [2].

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

3.2.1 Primitive instructions set

Each party (sensor node) in the protocol is modeled as a computing unit with a limited number of memory slots, where all local information is stored. The memory slot can be loaded with a) random value, b) encryption key and c) message. The 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 of describing 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 instruction 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-related in a group-oriented protocol. These variants are discussed later in Sections 4 and 5.

Using this set of primitive instructions, a simple plaintext key exchange can be written as {RNG N_1 R_1; SND N_1 N_2 R_1 R_1;}[5], a PUSH protocol as {RNG N_1 R_1; SND N_1 N_3 R_1 R_1; SND N_3 N_2 R_1 R_1;}, a PULL protocol as {RNG N_3 R_1; SND N_3 N_1 R_1 R_1; SND N_3 N_2 R_1 R_1;} and a multihop 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[6]. Proper restrictions might limit the to-

[5]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.

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

tal space size, but such limitation requires some knowledge about the target environment and the relationship between protocol and compromise pattern. Our goal is to create a method which requires only a description of the compromise pattern in a form suitable for the simulator, with the remainder being done by our proposed automated method.

3.2.2 Genetic operators

The mutation operator is applied at the level of integers (bytes) that encode a protocol. Every resulting (mutated) instruction is always valid. The instruction code is selected from the set of valid instruction codes; the parameters always remain in the correct range. As the mutation operator is applied with a given probability to every component of a primitive instruction (such as instruction code, parameters, boolean execution switch) multiple components of one instruction might, in principle, be modified during one mutation of the genome (candidate protocol). The crossover operator is applied at the level of instructions (no crossing point inside the instruction is allowed). The resulting instruction always has a valid form.

Several invalid states might occur as a result of mutation or crossover operator:

- **Reading from uninitialized memory slot** occurs when the instruction uses the value from a particular memory slot as an input and when no value was stored in this slot previously. When the usage of uninitialized slot is detected during protocol execution, the instruction is skipped and not executed. Some cases of this invalid reading can be detected during protocol post-processing at design time. However, some uninitialized memory slots might result from a message transmission error or node unreachability. A practical implementation should initialize all memory slots using a predefined value in order to detect this invalid state easily.

- **Sending message to permanently unreachable node** – actual layout of nodes deployed in the field might make it impossible to send a message defined in the protocol (SND instruction) to a permanently unreachable target node. Such situations cannot be usually detected at the design time. It results in missing an expected value in the memory slot of the target node, which could potentially cause a reading of an uninitialized memory slot. If the target node is permanently unreachable, such instruction will not increase the fitness value during the protocol simulation/evolution. The instruction is discarded from the resulting protocol during protocol post-processing (pruning), as will be explained later.

- **Combination of a new key value from different key subparts on involved nodes** – the resulting new key will be different on nodes involved in the protocol and thus unusable for subsequent encryption. Such situation might occur as a result of invalid sequence of instructions (detectable at the design time) or as a result of failed message transmission that fails to set a proper value to the target node memory slot. Some instances of such invalid state are automatically removed during post-processing as such key cannot contribute to overall protocol fitness value and is thus discarded. A practical implementation should verify if key subparts are identical on the communicating nodes before a new key is combined and used.

3.2.3 Network simulator

Candidate protocols are evaluated using our own simulator that was developed specifically for security analyses of key distribution protocols and message routing. We designed our own simulator, as the speed of simulation is a critical factor in the automatic generation process with evolutionary algorithms, where hundreds of thousands of whole network simulations must usually be conducted to obtain a secrecy amplification protocol that performs reasonably well. Commonly used simulators like ns2[7] work with an unnecessary level of details for our purposes, for example, with radio signal propagation or MAC layer collisions. They are unable to simulate networks with 100+ nodes in a matter of seconds. However, these common simulators might be used later to further test the discovered secrecy amplification protocols found using the method described in this work.

Our simulator is capable of performing:

- Random or patterned deployment of a network with up to 10^5 nodes together with neighbour establishment, secure links establishment and simple routing of messages.

- Evaluation of the number of secure links of probabilistic key pre-distribution protocols as described in [6]. Deployment of attacker's nodes and their eavesdropping impact on the network and evaluation of the number of secure links of published protocols for secrecy amplification of Key Infection approach (see [1] for details).

- A support for the evolutionary algorithms employed in an automatic generation of protocols. Protocols are described in the metalanguage of proposed primitive instructions (see Section 3.2.1) and consequently simulated to get the fraction of secure links as a fitness value (see Section 3.2). The implementation of the LGP is based on the GALib package[8].

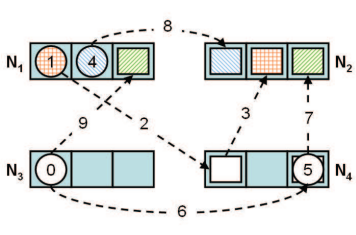
4. NODE-ORIENTED PROTOCOLS

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 [1, 8]. 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[9], 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.

4.1 Overview of the method

Initially, five protocols were generated; each of them consisting of 200 randomly selected primitive instructions. These candidate protocols form the initial population for the LGP.

[7] http://nsnam.isi.edu/nsnam/index.php/Main_Page
[8] GALib – C++ Genetic Algorithms Library.
[9] 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.



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

Every protocol is then simulated on our network simulator and the number of secured links serves as a fitness value. The 2/3 best-ranking protocols serve as parents for the next generation, which is created by applying crossover and mutation operators. Protocols from 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 the usage of evolutionary algorithms is not the only possibility how to generate protocols. We chose evolutionary algorithms as they have been already successfully used in WSN (although for a different purpose like the optimal node placement [11]), usually exhibiting significantly faster convergence towards solution than a brute-force search.

4.2 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. In this settings, the average success of the PULL protocol is 93.70% for three amplification iterations and 94.24% of secured links for ten iterations (assumed as an upper bound). Each party has 8 memory slots for storing intermediate values and a candidate protocol was limited by 200 elementary instructions. Simulations were performed for three distinct network deployments, the average fraction of secured links is 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.

4.3 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 in Figure 2. This is a "pruned" version of the original 200-instructions long protocol found by evolution.

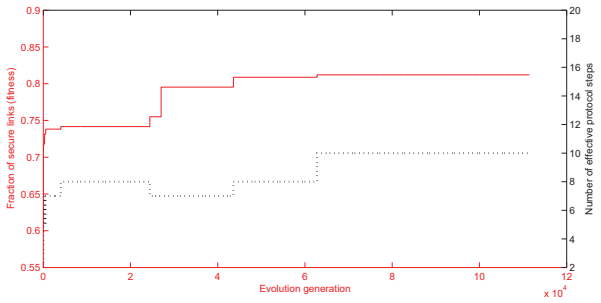


Figure 3: The typical progress of the fitness value and the number of effective instructions of the node-oriented protocol for one evolution run. Solid line shows fitness value (fraction of secure links) for the best candidate protocol in the particular generation and dotted line shows the number of effective protocol steps (after pruning) of the best candidate.

The importance of each instruction was tested[10] 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 shows a typical graph of fitness values for one particular run. 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 [1] is encoded in steps {1,2,3}. The PULL protocol by [8] is encoded in steps {0,6,9}. The multi-hop version [6] of PULL amplification 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 4 and 5.

The evolved protocol also exhibits an interesting feature of "polymorphic" instruction. At first inspection, 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 5).

Note that the automatic design of the node-oriented protocols with 5+ parties (nodes that take part in single exe-

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

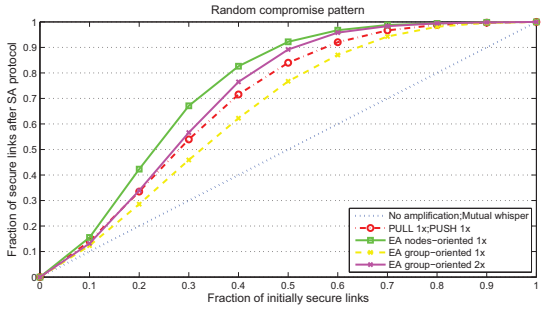


Figure 4: An increase in the number of secured links after secrecy amplification protocols in the Random compromise pattern. The PUSH and PULL protocols give the same results; mutual whispering does not improve security at all. Evolved group-oriented protocols will be described in Section 5. As can be seen, strong majority of secure links ($> 90\%$) can be obtained even when the initial network had one half of compromised links.

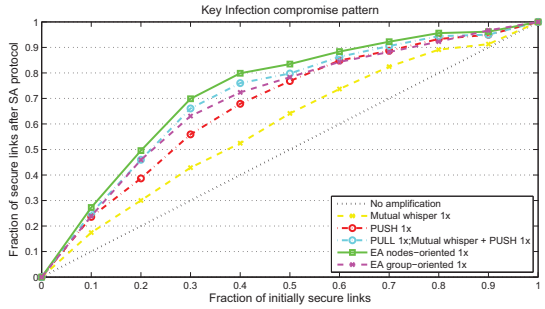


Figure 5: Key Infection compromise pattern (8 legal neighbours on average). The PULL protocol provides better results than the PUSH protocol. The combination of mutual whispering with the PUSH protocol gives the same results as the PULL protocol alone.

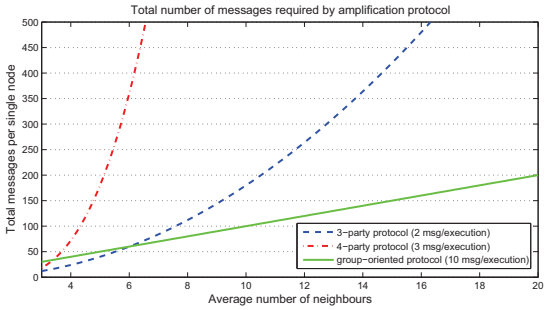


Figure 6: Total number of messages per single node required to perform a 3-party, 4-party node-oriented and group-oriented secrecy amplification protocol. Even when a group-oriented protocol utilizes significantly more messages per single execution, the total number of messages is smaller.

cution 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, a useful usage of the ENC and DEC instructions may exist, but the evolution was not able to find it. Secondly, a more probable reason could have arisen 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). A series of LGP runs were performed for the case when the transparent link encryption was *not* used. Evolution was significantly slower in achieving the same fraction of secured links, but the link encryption using existing keys was essentially developed anyway via the ENC and DEC instructions.

5. GROUP-ORIENTED PROTOCOLS

As we have already mentioned, node-oriented protocols introduce a high communication overhead – all k-tuples of neighbours must be involved in the execution of 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. Finally, due to the random placement of nodes in the sensor networks, the number of direct neighbours may vary significantly; a protocol constructed for a fixed number of parties can even fail due to there being an insufficient number of participants.

5.1 Method

We present a different approach to the design of secrecy amplification protocols with respect to established scenarios used in [1] and [8] (that are denoted as node-oriented protocols in this work). Identification of the parties in the protocol is no longer "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 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 with the distance values to identify the nodes involved. 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 total number of 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). Yet such protocol must 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 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 $\frac{10}{k}$ 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 6 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.

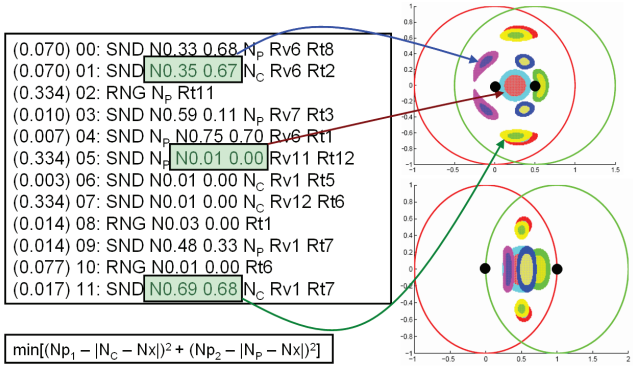


Figure 7: Example of an evolved group oriented secrecy amplification protocol. Selected node-relative identification (distance from N_C and N_P) of involved parties are displayed as the geographically most probable areas, where such nodes will be positioned (right part of the Figure). The number in brackets before each instruction gives the fitness loss when the instruction is removed from the protocol. The formula at the bottom is used to calculate deviation of the node in the field from the distance values stated in the protocol, where N_{P1} is distance from the node N_C and N_{P2} is the distance from N_P, respectively. Two probabilistic layouts for nodes positions are shown – upper layout is when distance between N_C and N_P is 0.6 of maximum transmission range. The lower layout is for maximum transmission range.

For the purpose of evolutionary 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 [3] without a transmission of the values or better by the post-processing of an evolved protocol (re-order of memory slots and additional CMB instructions).

As for node-oriented protocols, more iterations (amplification repeats) can be executed. For the purpose of evaluation, the results within one iteration are independent and may 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.

5.2 Results for group-oriented protocols

Efficient group-oriented protocols with a similar fraction of secure links comparable to node-oriented protocols were usually evolved in 10^5 generations (see Figures 4 and 5 for the performance of evolved group-oriented protocols). An example of such an evolved protocol is presented in Figure 7. 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 8) can be used to recognize the actual purpose and importance of the instructions (see Section 5.4 for more details).

Again, interesting and rather unexpected "tricks" were introduced through evolution. Firstly, two SND instructions in an example protocol shown in Figure 7 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 8 b)) 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 8 b). Note that position of area C (and so intermediate node) depends on the distance between nodes N_C and N_P. When these two nodes are close to each other then C is "behind" node N_C. As the nodes move away from each other, area C moves around N_C to the position shown in Figure 8 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 the N_C (Figure 8 c)).

Note that removal of a single instruction I from the pruned version (all instructions are necessary) of the protocol can not only 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 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. Thus, an evolved protocol exhibits a "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.

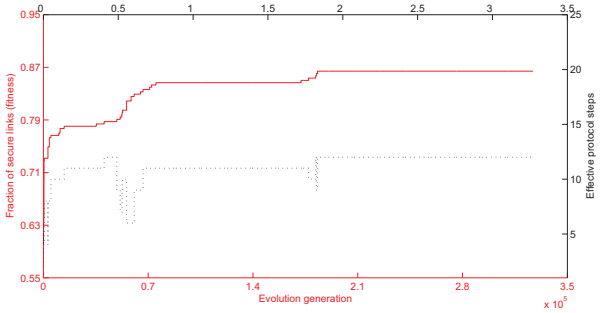


Figure 9: The typical progress of the fitness value and the number of effective instructions of the group-oriented protocol for one evolution run. Solid line shows fitness value (fraction of secure links) for the best candidate protocol in the particular generation and dotted line shows the number of effective protocol steps (after pruning) of the best candidate.

5.3 Parameters used for LGP

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

Based on random sampling test, the fitness landscape[11] for node-oriented protocols seems to be highly rugged with only a 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) [16]). The population size was fixed to 5 individuals. The mutation operator is applied with 10% probability. Similarly to the CGP, crossover is not used (series of experiments did not shown improvements in evolution convergence when crossover was used).

The fitness landscape for 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[12] applied with the probability 70%. 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 the actual population is used to maintain population size for both types of the protocols.

[11]The fitness values for each possible instance in the search space. Note that we cannot compute the whole landscape in a reasonable time – if we could, then there would not be any need for any EA – we could obtain the solution, i.e. the fitness maximum, directly.

[12]Crossover points allowed at the level of instructions only.

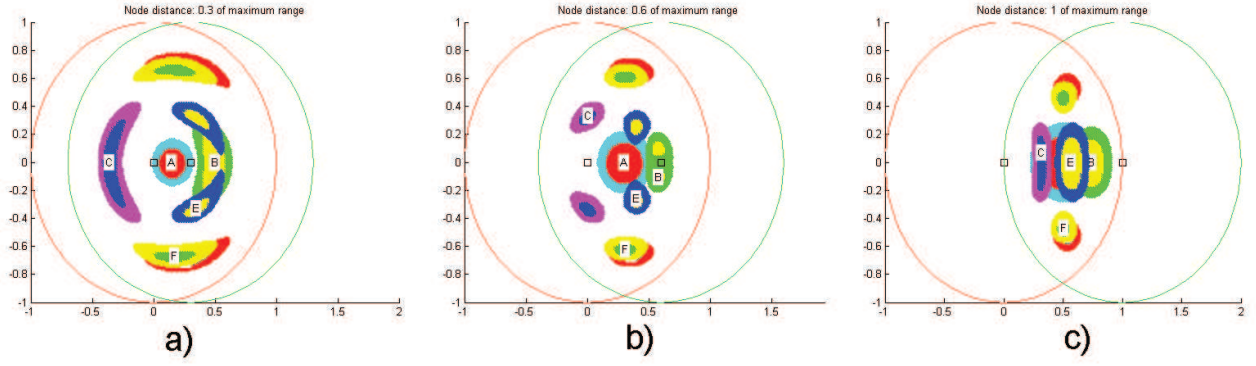


Figure 8: Layout of areas for potential parties when the distance between the central node N_C and node N_P is a) 0.1 of the maximum transmission range, b) 0.6 range and c) the maximum transmission range.

5.4 Methods for analysis of evolved protocols

Various techniques 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 node distributions 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 remaining instruction (some instructions may be completely removed if they have no function without I). The higher the decrease in the fitness gain by a particular instruction J, the stronger the dependency of J on the removed I.

- **Analysis of memory store/load sequences** – As described in section 3.2.1, 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 this process (see Figure 10 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. The resulting graph can then be analyzed to obtain an indication of paths the values propagate during the protocol execution.

- **Probable areas for parties identified by the relative distance** – Visualization of the areas where nodes referenced in the protocol will, with a high probability, be positioned, 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 8 b)) 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.

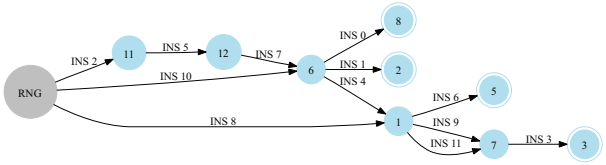


Figure 10: Memory chain for example group-oriented protocol from Figure 7. Circles constitute memory registers with labeled edges standing for instruction responsible with register manipulation. The memory registers used for the final link key construction are double-circled.

6. FUTURE WORK

This work presents results for secrecy amplification protocols applied in environments with randomly compromised links or a localized compromising attacker. Future work will cover the possibility of protocol generation for selective link compromise patterns and performance of protocols evolved for the randomly compromised links when applied to an environment with selective link compromise patterns.

Additionally, investigation of the attacker's compromise strategy is an interesting and complex problem, especially when various properties of deployment scenarios, including nodes relative positions, are to be considered. The protocols automatically designed in our work exhibited increased tol-

erance for unreachable nodes. Also, the group-oriented design requires significantly smaller numbers of messages, but might be more vulnerable to a selective compromise strategy as a relatively small number of possible paths are used for fresh secret propagation. Automated design of an attacker strategy against existing secrecy amplification protocols is another open research question.

The "tricks" found/discovered in EA generated protocols with good properties may be subsequently used for non-automatic design of new protocols. As EA is limited only by the defined constraints (here an implementation in simulator), building blocks in EA generated algorithm can be novel and surprising. But note that really useful tricks are occurring rarely and obfuscated versions of already known design principles/building blocks are much more often. A thorough analysis of evolved protocols is therefore necessary before any evolved building block or perceived "trick" is used in non-automatic protocol design. Particular caution must be taken when defining a set of rules for estimating protocol quality by a simulator. Incomplete definition of the fitness function or simulated constraints might result in a well-performing, but practically invalid protocol. While developing our simulator, we experienced several protocols securing suspiciously high fraction of links. After deeper inspection of such protocol, we discovered either our programming error or an incomplete specification of the fitness function that was exploited by the EA in well-performing, but practically invalid protocol.

On the other side, a too restrictive set of simulator rules is not appropriate as well as it is (possibly unnecessarily) limiting the search space for the EA. If the novel "tricks" are of the main interest, one may start with a rather benevolent set of rules giving the EA a large search space. New rules can be then be iteratively added according to inspections of the evolved protocols, in order to restrict violations of yet undefined practical constraints. We will focus on such methods in our further work.

7. 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 in networks with randomly compromised links (e.g., resulting from node capture for probabilistic pre-distribution), but may give a suboptimal performance when applied to more correlated compromise patterns arising from distribution approaches such as Key Infection. Moreover, some steps in the secrecy amplification protocol may be pointless for a given compromise pattern as they do not improve the 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 protocols. 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 an evolutionary design approach, we were able to find protocols where the fraction of secured links is comparable to node-oriented protocols, but with only a linear (instead of exponential) increase in the required messages with respect to the increasing number of neighbours. This is especially important for dense networks with more than 10 neighbours.

8. ACKNOWLEDGMENTS

We would like to thank Dan Cvrček for fruitful discussions about the manual design of secrecy amplification protocols and to Geraint Price for his help with the final version of our paper. Petr Švenda was partially supported by the Czech Science Foundation grant GD102/05/H050 Integrated approach to education of PhD students in the area of parallel and distributed systems. Lukáš Sekanina was supported by the the Research Plan No. MSM 0021630528 Security-Oriented Research in Information Technology.

9. REFERENCES

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

[2] W. Banzhaf, P. Nordin, R. Keller, and F. Francone. *Genetic Programming – An Introduction*. Morgan Kaufmann Publishers, San Francisco, CA, 1998.

[3] R. Bloom. An optimal class of symmetric key generation systems. In *Proceedings of EUROCRYPT'84, Springer-Verlag LNCS 209*, pages 335–338, 1984.

[4] M. Brameier and W. Banzhaf. *Linear Genetic Programming*. Springer, Berlin, 2007.

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

[6] H. Chan, A. Perrig, and D. Song. Key distribution techniques for sensor networks. In *Wireless sensor networks*, pages 277–303, Norwell, MA, USA, 2004. Kluwer Academic Publishers.

[7] H.-J. Choi. Security protocol design by composition. In *Technical report UCAM-CL-TR-657, University of Cambridge*, 2006.

[8] D. Cvrcek and P. Svenda. Smart dust security - key infection revisited. *International Workshop on Security and Trust Management 2005, ENTCS Vol. 157, Italy*, pages 10–23, 2005.

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

[10] L. Eschenauer and V. 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.

[11] K. P. Ferentinos and T. A. Tsiligiridis. Adaptive design optimization of wireless sensor networks using genetic algorithms. *Comput. Netw.*, 51(4):1031–1051, 2007.

[12] Y. H. Kim, M. H. Kim, D. H. Lee, and C. Kim. A key management scheme for commodity sensor networks. *ADHOC-NOW 2005, LNCS 3738*, pages 113–126, 2005.

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

[14] D. Liu and P. 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.

[15] C. 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.

[16] J. Miller and P. Thomson. Cartesian Genetic Programming. In *Proc. of the 3rd European Conference on Genetic Programming EuroGP2000*, volume 1802 of *Lecture Notes in Computer Science*, pages 121–132. Springer-Verlag, 2000.

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

[18] R. D. Pietro, L. V. Mancini, and A. 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.

[19] D. X. Song, S. Berezin, and A. Perrig. Athena: A novel approach to efficient automatic security protocol analysis. *Journal of Computer Security*, 9(1/2):47–74, 2001.

[20] P. Švenda and V. Matyáš. Key distribution and secrecy amplification in wireless sensor networks. In *Technical Report, FIMU-RS-2007-05, Masaryk University, Brno*, 2007.