A new approach to secrecy amplification in partially compromised networks

Radim Ošťádal[1], Petr Švenda[2], and Václav Matyáš[2]
ostadal@mail.muni.cz {svenda, matyas}@fi.muni.cz

[1] Government CERT, Czech Republic
[2] Masaryk University, Brno, Czech Republic

Abstract. Usage of various key (pre-)distribution schemes (KDSs) in networks with an active attacker results in a partially compromised network where some fraction of keys used to protect link communication is known to the attacker. The secrecy amplification protocols were proposed to secure again some previously compromised communication links by using non-compromised paths to deliver new secure keys. Design of suitable secrecy amplification protocols remains a challenge in scenarios where a trade-off between necessary resources (e.g., energy necessary for transmission of message) and improvement in the number of secure links must be balanced. We inspect classes of secrecy amplification protocols known as node-oriented and group-oriented protocols proposed for use in wireless sensor networks (WSN).
We combine analysis of given protocol participant placement via a simulator and manual post-processing to provide a simpler, practically usable hybrid protocol with less steps and lower communication overhead, yet still better in terms of re-secured links than previously proposed protocols.

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

1 Introduction

Secure link communication is the building block for many security services maintained by a wireless sensor network (WSN). Secure link is usually achieved by a secret key shared between communicating parties, requiring suitable key management techniques. Common assumption in WSNs is the inevitability of a partial compromise in a network when nodes can be captured and keys extracted from the memory as no tamper resistance is usually assumed.

Our work targets scenarios where a link between nodes can be compromised yet the nodes themselves are not. A typical example comes with schemes based on symmetric cryptography, where the attacker learns a fraction of used keys, resulting in a partially compromised network. 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* (SA) 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 [7]. A secrecy amplification protocol can be executed to secure again some of the compromised links, resulting in a less compromised network. SA protocols were shown to be very effective, but for the price of a significant communication overhead. Our aim is to provide SA protocols that can secure a high number of links, but require only a small number of messages and are easy to execute and synchronize parallel executions in the real network – properties not found together in previously published SA protocols.

Also, we like to challenge the ways how performance of key distribution schemes is currently judged. If SA protocols are efficient enough to be used, performance of key distribution schemes should be also compared with the option that an SA protocol will be applied.

The contributions of our work are: 1) Detailed analysis of impact of different node placement in previously published SA protocols; 2) design of a new class of SA protocols combining advantages of previously known SA protocols; and 3) a concrete efficient SA protocol outperforming previously published ones, together with its analytical and experimental evaluation.

This paper is organized as follows: the next section provides a short introduction to wireless sensor networks and compromise patterns resulting from different KDSs and attack strategies. Section 3 highlights related security issues and provides an overview of related work on node and group oriented secrecy amplification protocols. Section 4 describes the proposed approach of hybrid protocols taking the best from reviewed technique. Upper bound for secrecy amplification and our new manually constructed hybrid protocol are presented in Section 5. Section 6 provides a comparison with (so far) best node- and group-oriented protocols based on overall success rate. Conclusions are given in Section 7. The Appendix A provides detailed settings and observations of hybrid protocol properties in terms of number of amplifications and messages.

2 Partial network compromise

A wide range of key distribution, establishment and management techniques were proposed (see [4] for an overview). Distinct key distribution schemes behave differently when a network is under attack targeted to disturb link key security. The impact on link key security differs based on the attack strategy used. Although various schemes significantly differ in the way how keys are distributed and managed, similar compromise patterns can be detected. A compromise pattern provides us with a conditional probability that link Y is compromised when other link X is compromised after a relevant attack.

The characteristics of a particular compromise pattern may significantly influence the success rate of the secrecy amplification executed later. We will per-

form analysis of secrecy amplification protocols according to the following two possible compromise patterns, but our work can be extended to additional patterns as well.

2.1 Random compromise pattern

Random compromise pattern arises when a probabilistic key pre-distribution scheme [7] and later variants [3, 6, 8] are used and an attacker extracts keys from several randomly captured nodes.

In case of a node capture, all links to the captured node are compromised. If some probabilistic pre-distribution scheme like [3, 7] is used, then some additional links between non-compromised nodes become compromised as well. Probabilistic key pre-distribution schemes exhibit almost uncorrelated pattern resulting from node capture and extraction of randomly selected keys.

2.2 Key infection compromise 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 on 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's peers, the higher the probability of a compromise. Typically, if the eavesdropping node is close to a legal node, most of the links to the latter can be compromised.

An eavesdropping of the exchanged key in the key infection approach [1] does not compromise nodes directly, but compromises links in the reach of eavesdropper's radio instead. Key infection distribution forms a significantly correlated pattern due to locality of eavesdropping – links close to the eavesdropper have a higher probability of being compromised.

3 Secrecy amplification

Several secrecy amplification protocols were previously published and can be grouped according to general principles of their construction. 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 on all paths to compromise the new key value. A second method, called *multi-hop key amplification*, is basically a 1-path version of the multi-path key establishment with more than one intermediate node C_i.

3.1 Node-oriented protocols

Node-oriented security amplification protocols were firstly introduced in [1] and later enhanced and expanded in [5]. Node-oriented protocol is executed for all

possible k-tuples of neighbours in the network. Note that the number of such k-tuples can be high, especially for dense networks (e.g., more than 10 direct neighbours) and resulting communication overhead is significant[3].

A variant of the PUSH protocol, called the PULL protocol, was presented in [5]. 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 [5]. 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[4] from A to B and K_2 from B to A. 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 was presented in [10].

One of the most advanced node-oriented protocols was defined in [11], using the method for automatic generation of secrecy amplification protocols, which utilized linear genetic programming (LGP) [2]. A detailed analysis showed that the protocol consists of previously defined mutual whispering, PUSH protocol, PULL protocol and also the multi-hop version of PULL amplification. We are using those protocols as a base for construction of more advanced hybrid protocols.

3.2 Group-oriented protocols

In group-oriented protocols, an 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 are using the distance from two distinct nodes). It is assumed that each node knows the approximate 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., $N_{0.30\_0.70}$ 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

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

[4] 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.

neighbours, the node closest to the indicated distance(s) is chosen as the node N_k for a particular protocol run. There is no need to re-execute the protocol for all k-tuples (as was the case for node-oriented protocols) as all neighbours can be involved in a single execution, reducing the communication overhead significantly. See [11] for a detailed description of group-oriented protocols.

Note that inferring the relative distance from the received signal strength indication (RSSI) is usually a burden with errors resulting from the generally unreliable propagation of wireless signal and also as the relation between distance and RSSI is not linear. Relative distances used in group-oriented protocols are robust against moderate inaccuracies as a precise node position is not required for a protocol to succeed.

4 Hybrid protocols

In this paper, we propose a new kind of protocols that combine advantages of both node- and group-oriented protocols. A protocol consists of several primitive instructions as described later in Section 5.1. Its construction is based on knowledge gained from analysis of node-oriented and group-oriented protocols.

Both mentioned types of secrecy amplification protocols covered in Section 3 have their advantages and disadvantages. As described previously, node-oriented protocols exhibit polynomial increase of messages with respect to the number of neighbours in the network. An additional issue is unknown number of direct neighbours and their placement. A protocol prepared for a fixed number of parties could fail due to lack of participants.

The group-oriented protocols do not share those issues and they show only a linear increase in messages sent with respect to the number of neighbours. The main difficulty is their complexity and complicated analysis of their behaviour. They consist of multiple times more instructions when compared with node-oriented protocols (e.g., the best performing group-oriented protocol presented in [9] has 41 instructions and might include cooperation of up to 34 nodes. Compare this to the PUSH protocol with 3 instructions and only 3 nodes involved.). Those are issues limiting practical implementation and further adoption.

Hybrid protocols proposed in this work show only a linear increase in messages sent with respect to the number of neighbours and do not require storing multiple values transmitted during the protocol execution, easing synchronization during parallel runs occurring in a real network. They are using relative distance from special nodes N_C and N_P in the same way as group-oriented protocols. They contain a lower number of instructions and their construction, analysis and implementation are simpler than for group-oriented protocols.

Steps of a hybrid protocol are as follows:

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 might be 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).
3. 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, parties of the protocol are replaced by real identification of the nodes that are positioned as close as possible to the relative identification given by N_C and N_P in the protocol.
4. The key is updated after every protocol execution and only between nodes N_C and N_P. Also the memory slots of all participants are cleared.

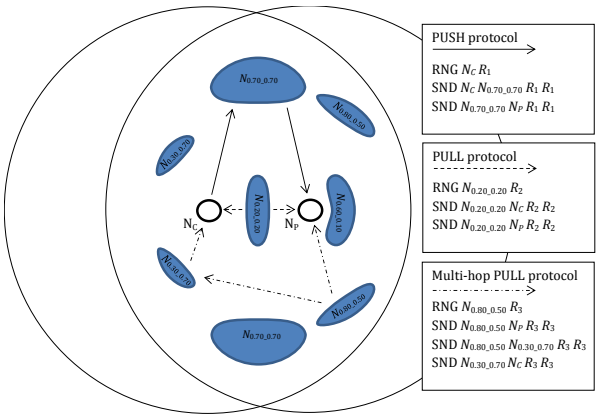

Fig. 1. An example of instructions of a basic hybrid secrecy amplification protocol. The PUSH, PULL and multi-hop version of PULL protocol are included. Selected node-relative identification (distance from N_C and N_P) of involved parties are displayed as the geographically most probable areas, where such nodes will be positioned. A probabilistic layout is shown for the case where the distance between nodes N_C and N_P is 0.5 of the maximal transmission range.

We construct protocols with application of knowledge from node-oriented protocols and statistical data about the most suitable placement of participating intermediate nodes. A hybrid protocol is executed for every pair of neighbouring nodes instead of every k-tuple as in node-oriented protocols. Other participating intermediate nodes are used for transmission of n different values for shared key update in the same fashion as previously described basic node-oriented protocols (mutual whispering, PUSH, PULL, multi-hop versions of PUSH and PULL). A visualisation of an example protocol can be seen in Figure 1. Participating intermediate nodes are not required to store any forwarded values and can erase

them as soon as a message with the amplification value is forwarded to the next node towards destination. This allows for a simple synchronization even within large and dense WSNs. We incrementally improve the results of the protocol utilizing the greedy search approach for intermediate node(s) placement. All evaluations are performed on our reference network consisting of 100 legal nodes (7.5 legal neighbours on average) with originally 50% links compromised.

5 Optimal node placement

As was demonstrated in previous work, secrecy amplification protocols are able to provide a significant increase in secure links, e.g., from 50% of originally secured links to more than 90%. To achieve such an improvement, there is a considerable overhead in communication and on-node processing. In the subsequent section, we use a combination of different analysis techniques backed by large data sets about merits of different positions of intermediate nodes in hybrid protocols obtained from the simulator for selected compromise patterns.

5.1 Network simulator

New hybrid protocols proposed throughout this work are evaluated using the same simulator that was developed specifically for security analysis of key distribution protocols and message routing by the authors of [11]. Commonly used simulators like ns2 or OMNeT++ work with an unnecessary level of details for our purposes (e.g., radio signal propagation or MAC layer collisions), significantly slowing evaluation of given network scenarios. The simulator is able to simulate a secrecy amplification protocol on fifty networks with 100 nodes each in about 3 seconds when executed on one core CPU @ 1.7 GHz.

The 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 [4]. 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).

Protocols evaluated in the simulator are described in a metalanguage of proposed primitive instructions. Each party (a real node in network) in the protocol is modelled as a computing unit with a limited number of memory slots, where all local information is stored. Each memory slot can contain either a random value, encryption key or message. Protocol instructions were selected with the aim of describing all published secrecy amplification protocols and use only (cryptographic) operations available on real nodes.

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 on node N_a and store the result to R_k. The combination function may vary on the application needs (e.g., a cryptographic hash function such as SHA-3).
– 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.
– 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.

5.2 Upper bound for amplification success

At first, an upper bound for amplification success (maximum number of secure links achievable by any sort of secrecy amplification protocol) can be established. A given link between nodes A and B is securable if there exists at least one secure path (no links on such a path are compromised) between nodes A and B via other nodes C_i.

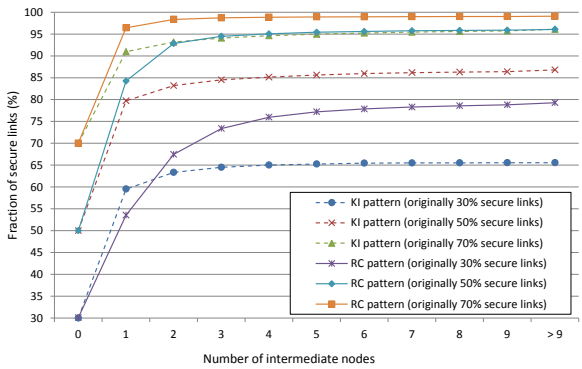

Fig. 2. Maximal possible increase in the number of secured links with dependency on the number of intermediate nodes. Results are displayed for both random compromise (RC) pattern and key infection (KI) compromise pattern. As can be seen, strong majority of secure links ($> 90\%$) can be obtained even when the initial network compromise is 50% (for RC pattern) or 70% (for KI pattern).

The upper bound of secured links can be achieved by a secrecy amplification protocol by sending fresh new keys via all possible paths between any two nodes. If there is a secure link, it will be used. However, there are two main practical limitations to such approach. First is the extraordinary high number of such paths even in a small network, resulting in unacceptable transmission overhead. Second is a high fragility to packet loss – as a network is usually not aware when

particular links are compromised, fresh keys must be sent and combined together (e.g., via hash function) from all such paths – if even a single fresh key is lost, A and B will not be able to establish the same secure key value.

We used modified Floyd-Warshall algorithm to establish an upper bound for given network. The Floyd-Warshall algorithm is a graph analysis algorithm for finding the shortest paths in a weighted, directed graph. A single execution of the algorithm will find the shortest path between all pairs of vertices. As the precise compromise pattern for a given network is not known in advance (depends on an attacker, secrecy amplification protocol, exact placement of nodes, etc.), we perform multiple evaluations for different networks to obtain an average result. Details of used algorithm are available in Section A.

In the rest of this paper, we will focus more on the random compromise pattern as there are more key distribution schemes resulting in this pattern after an attack and also a bigger potential for improvement in fraction of secure links.

5.3 Intermediate nodes placement

We have used fifty random deployments of our reference network for the evaluation of the best placement of an intermediate node. We need to be able to address a sufficient number of distinct neighbours from two specific nodes N_C and N_P. We chose granularity of 0.01 and 0.03, respectively, as they allow us to address 10 000 (0.01 granularity used for both N_C and N_P) and 1 156 (0.03 granularity) different neighbours, respectively. Both numbers give us a satisfactory number of distinct positions in two-dimension plane as we do not expect the network density more than 100 neighbouring nodes. The granularity 0.01 is used for placement of one intermediate node w.r.t. PUSH and PULL protocols. The granularity 0.03 is used for multi-hop versions of PUSH and PULL protocols as the placement of two intermediate nodes has to be found and it presents (already high) computational demand increased by two orders.

In our protocol design, the basic protocols taken over from the node-oriented approach are used. Those are mutual whispering, simple PUSH and PULL protocols, and their multi-hop versions. Focusing on the random compromise pattern, mutual whispering does not provide us with any improvements and there is no difference between the PUSH and PULL protocol (see [5] for reasoning). The same holds also for multi-hop versions of these.

We incrementally select five intermediate nodes for PUSH and PULL protocols with the greedy search approach. Choosing the sixth node would give us only a negligible improvement with respect to the portion of secure links. We have evaluated every possible placement and the resulting number of secure links after the protocol execution. We were able to get the 83% of secure links with a single amplification. Final results are shown in Figure 6 in Appendix.

A different approach for analysis of multi-hop versions of PUSH and PULL protocol is necessary as it is not feasible to evaluate all possible particular results due to exponential state explosion with every additional intermediate node. We inspect three cases closely: 1) Specific placement of two nodes where the portion

of secure links is biggest. 2) The average number of secure links calculated across all results for the first intermediate node placement is biggest. 3) The average number of secure links calculated across all results for the second intermediate node placement is biggest. In all cases, the best results were observed for placement of both intermediate nodes into the same position, effectively reducing the number of intermediate nodes. We interpret this in a way that a standard PUSH/PULL protocol gives us better or the same results as a multi-hop version of the PUSH/PULL protocol. The reason is that in the random compromise pattern, less intermediate hops mean a lower probability that compromised link will be used. Less intermediate hops also mean a lower number of all possible paths, but hybrid protocol is constructed so that not all paths are taken anyway.

A secrecy amplification protocol might be iterated multiple times for the same pair of nodes. As new links are secured in the first iteration, following iterations have a better starting position than the first one, potentially securing additional links. The final protocol works comparably with the node-oriented version of PUSH/PULL protocols on the random compromise pattern and consists of 5 independent sub-protocols (each corresponds to one PUSH/PULL protocol), resulting in 15 instructions. With initial 50% compromised links and two amplifications, we managed to get the network with 92.5% links secured, with three amplifications 94% secured links. When network has 40% initially compromised links, 91.9% links are secured after one amplification and 97.6% after three amplifications.

The search for intermediate node placement was conducted with simulations executing with only one amplification of particular basic protocols. As we expect more than one amplification of our final protocol (e.g., three) will be used (see results above), we also inspected difference when three amplifications are used for the search. Resulting graphs share the overall shape with those presented in Figure 6. The difference of the lowest and the highest success rate of the protocol is generally lower using three amplifications of the protocol than in the case of using only one, but overall difference is not significant enough to change instructions in the proposed protocols.

5.4 Constructing new protocol

The resulting protocol is shown in Figure 3. There are still several ways for its optimization. We focus on minimization of the communication overhead. The tool we use is *protocol pruning*. Protocol pruning is a process of progressively removing every primitive instruction from the protocol and evaluating the change in the success ratio after a modified protocol execution. It gives us the loss of secured links when the instruction is removed from the protocol.

We were able to iteratively remove sub-protocols 4 and 5 with the success ratio loss of only 0.0012 when employing three amplifications. This means that the fraction of secured links is reduced only by 0.12 percent, which is negligible. Removing the first block means reduction by 0.9 percent, which is also a relevant trade-off given the fact that two messages are spared in every execution. We tested the final protocol on five hundred of deployments with an average success

ratio 93.21% (secured links). Removing block two or three causes the success ratio loss of 17.0% or 14.5%, respectively.

#	instructions
0	RNG $N_{0.32\_0.85}$ R_1
1	SND $N_{0.32\_0.85}$ N_C R_1 R_1
2	SND $N_{0.32\_0.85}$ N_P R_1 R_1
3	RNG $N_{0.69\_0.98}$ R_2
4	SND $N_{0.69\_0.98}$ N_C R_2 R_2
5	SND $N_{0.69\_0.98}$ N_P R_2 R_2
6	RNG $N_{0.01\_0.39}$ R_3
7	SND $N_{0.01\_0.39}$ N_C R_3 R_3
8	SND $N_{0.01\_0.39}$ N_P R_3 R_3
9	RNG $N_{0.56\_0.70}$ R_4
10	SND $N_{0.56\_0.70}$ N_C R_4 R_4
11	SND $N_{0.56\_0.70}$ N_P R_4 R_4
12	RNG $N_{0.89\_0.01}$ R_5
13	SND $N_{0.89\_0.01}$ N_C R_5 R_5
14	SND $N_{0.89\_0.01}$ N_P R_5 R_5

Fig. 3. Best performing hybrid protocol (HP_{BEST}).

#	instructions
0	RNG N_C R_2
1	SND N_C $N_{0.69\_0.98}$ R_2 R_2
2	SND $N_{0.69\_0.98}$ N_P R_2 R_2
3	RNG N_C R_3
4	SND N_C $N_{0.01\_0.39}$ R_3 R_3
5	SND $N_{0.01\_0.39}$ N_P R_3 R_3

Fig. 4. Final hybrid protocol (HP_{FINAL}).

It is not efficient to remove more instructions as the success ratio loss becomes excessive. However, we can achieve a higher success ratio gain by increasing the number of repetitions of an amplification protocol. As we are taking the number of messages sent as the primary measure of how demanding a protocol is, we compare the efficiency of our final protocol (4 messages transmitted per protocol execution and 3 amplifications) with only one sub-protocol (2 messages transmitted per protocol execution and 6 amplifications). The best performing sub-protocol was the first one from Figure 3. It was able to secure nearly 90% links in the network, but this is worse than our original protocol. It is not possible to substitute any of the remaining instructions by amplifications preserving the performance of the protocol. Even if more amplifications would be able to do so, the communication overhead would exceed our final protocol.

To simplify synchronization of parallel protocol executions, we put full control over the protocol execution to the central node. As there is no difference between the PUSH and PULL protocol for the random compromise pattern, originally used PULL sub-protocols could be replaced by the PUSH ones. Both nonce generations are performed by the central node N_C and consequently transmitted to the intermediate node N_K. N_K only forwards the nonce to N_P and can forget it immediately. This will help with management of parallel protocol execution. The final protocol is shown in Figure 4.

We tested the performance on fifty larger networks with 1 000 nodes and initially with 50% links marked as compromised. The average number of neigh-

bours was 7.5. The performance of the protocol was very similar as computed on smaller networks, with deviations of success ratio only around 0.5%.

6 Success rate

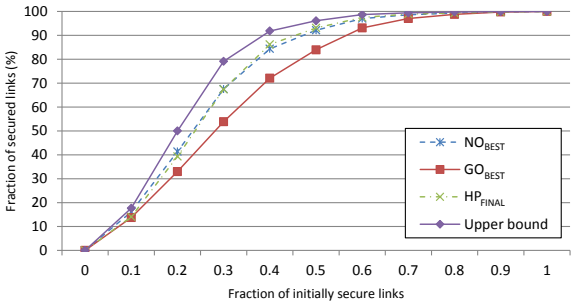

Fig. 5. An increase in the number of secured links after secrecy amplification protocols in the random compromise pattern. The best performing node-oriented protocol [11] is denoted as NO_{BEST}. The best performing group-oriented protocol [9] is denoted as GO_{BEST}. The pruned version of best hybrid protocol HP_{BEST} consists of 6 instructions (4 are SEND) is denoted as HP_{FINAL}. HP_{FINAL} is executed with 3 amplifications as it requires a comparable communication overhead. As can be seen, a strong majority of secure links ($> 90\%$) can be obtained even when the initial network had one half of compromised links.

The impact of the best known node- and group-oriented protocols together with our final hybrid protocol (as described in Section 4) for the random compromise pattern is compared in Figure 5. The NO_{BEST} performs slightly better than our hybrid protocol for the fraction of 20% initially secured links. For 40% and more, the HP_{FINAL} provides the best results among the tested protocols. The overall success rate is also very close to the theoretical reachable maximum computed by the modified Floyd-Warshall algorithm in Section 5.2.

The more detailed comparison covering number of required messages, the impact of repeated secrecy amplifications and details of practical implementation for the TelosB hardware platform with the TinyOS 2.1.2 operating system and tested on our laboratory test-bed with 30 nodes positioned atop of nine interconnected offices can be found in Annex A.

7 Conclusions

Our work presented in this paper demonstrates that hybrid amplification protocols can provide better trade-off between security and efficiency than currently

known approaches for secrecy amplification. Hybrid protocols show only a linear increase in messages sent with respect to the number of neighbours and do not require storing multiple values transmitted during the protocol execution. Proposed hybrid protocols also contain fewer instructions and their construction, analysis and implementation are simpler than for group-oriented protocols. The synchronization of the protocol steps and parallel execution on multiple nodes is easier than for previous approaches, making this approach practically usable on current platforms like TelosB as demonstrated by our prototype implementation.

Acknowledgments

This work was supported by the GAP202/11/0422 project of the Czech Science Foundation.

References

1. Anderson, R., Chan, H., and Perrig, A. Key infection: Smart trust for smart dust. In *12th IEEE International Conference on Network Protocols* (2004), IEEE, pp. 206–215.
2. Brameier, M. F., and Banzhaf, W. *Linear Genetic Programming (Genetic and Evolutionary Computation).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
3. Chan, H., Perrig, A., and Song, D. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy* (2003), pp. 197–213.
4. Chan, H., Perrig, A., and Song, D. *Wireless Sensor Networks.* Kluwer Academic Publishers, Norwell, MA, USA, 2004.
5. Cvrček, D., and Švenda, P. Smart dust security-key infection revisited. In *Electronic Notes in Theoretical Computer Science* (2006), vol. 157, Elsevier, pp. 11–25.
6. Di Pietro, R., Mancini, L. V., and Mei, A. Random key-assignment for secure wireless sensor networks. In *1st ACM Workshop on Security of Ad Hoc and Sensor Networks* (2003), ACM, pp. 62–71.
7. Eschenauer, L., and Gligor., V. A key-management scheme for distributed sensor networks. In *9th ACM Conference on Computer and Communications Security, Washington, DC, USA* (2002), ACM, pp. 41–47.
8. Liu, D., and Ning, P. Establishing pairwise keys in distributed sensor networks. In *10th ACM Conference on Computer and communications security* (2003), ACM Press, pp. 52–61.
9. Smolka, T., Švenda, P., Sekanina, L., and Matyáš, V. Evolutionary design of message efficient secrecy amplification protocols. In *12th European Conference on Genetic Programming* (2012), pp. 194–205.
10. Švenda, P., and Matyáš, V. *Key distribution and secrecy amplification in wireless sensor networks, Technical report FIMU-RS-2007-05.* Masaryk university, Czech Republic, 2007.
11. Švenda, P., Sekanina, L., and Matyáš, V. Evolutionary design of secrecy amplification protocols for wireless sensor networks. In *Second ACM Conference on Wireless Network Security* (2009), pp. 225–236.

A Protocol evaluation

A.1 Upper bound for amplification success - details

Let us define the graph $G(V, E)$, where V represents the nodes in our WSN deployment and E represents neighbour relationships. After basic whispering (for the key infection compromise pattern) or after random key pre-distribution (for the random compromise pattern), we assign weight one to the edge between nodes N_i and N_j if and only if there exists a secure link key established between those nodes. If the link key is compromised, we assign weight equal to infinity.

After the Floyd-Warshall algorithm execution, we obtain the shortest path between all nodes and we can interpret the results as follows:

- If the nodes N_i and N_j are neighbours and there exists a shortest path between them, the link can be secured. The length of this path reduced by one is also the minimum number of intermediate nodes needed to secure the link.
- If the nodes N_i and N_j are neighbours and there is no shortest path calculated, the link cannot be secured.

We carried out several experimental calculations. Every calculation was conducted on fifty random deployments of our reference network. As can be seen in Figure 2, there is a significant difference between two inspected compromise patterns. In the random compromise pattern, we are able to secure significantly more link keys than in the key infection compromise pattern. We can explain this situation by the fact that in the key infection compromise pattern, the compromised links are concentrated in particular areas around eavesdropping nodes and it is more probable that such links cannot be secured. It can be also seen that the most benefit can be gained using two intermediate nodes. With more nodes, the increase in secure links fraction is very small.

A.2 Number of amplifications

Different classes of secrecy amplification protocols use different capabilities to improve security throughout the network. A node-oriented protocol sends key updates via every possible neighbour or neighbours by a simple protocol. Group-oriented protocols share key updates inside the bigger group of cooperating nodes identified by relative distances. Hybrid protocols use sub-protocols (similarly to node-oriented), relative distances (similarly to group-oriented) and additionally utilize several repetitions of the whole process. Figure 7 shows the performance of particular protocols in our reference network with respect to the number of amplifications.

The most important observation is that with an increasing number of amplifications, the difference in success among distinct types of protocols is smaller, being negligible when four or more amplifications are used. That fact implies that protocol repetitions can substitute other methods used in node- and group-oriented protocols. We expect that three amplifications of a protocol will be a

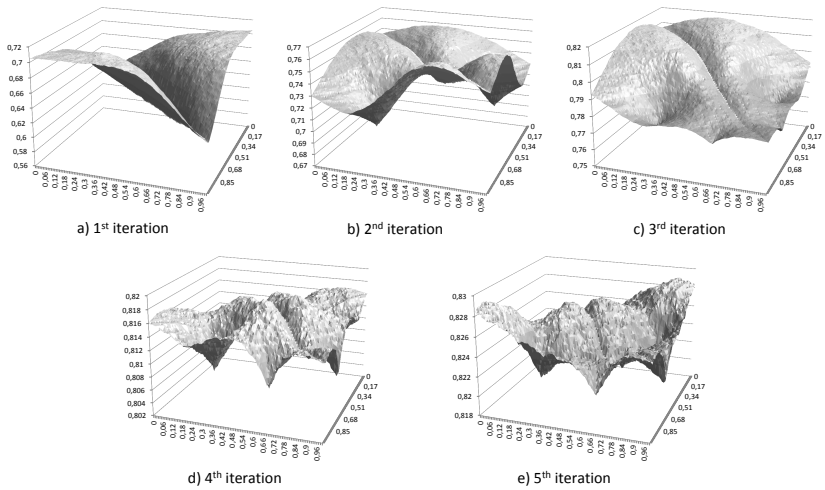

Fig. 6. Iterative search of intermediate nodes for PUSH/PULL protocols. Horizontal axis depicts distance from the node N_P, depth axis depicts distance from the node N_C, vertical axis represent the portion of secure links (quality of protocol with given position of nodes, scaled between 0 and 1). Based on our results, the first intermediate node is selected as $N_{0.32\_0.85}$ with the resulting success ratio 0.71 (a), second node as $N_{0.69\_0.98}$ with the success ratio 0.77 (b), third node as $N_{0.01\_0.39}$ with the success ratio 0.815 (c), fourth node as $N_{0.56\_0.70}$ with the success ratio 0.82 (d) and fifth node as $N_{0.89\_0.01}$ with the final success ratio of 0.83 (e).

proper compromise between a overall success of the hybrid protocol and the communication overhead it requires.

A.3 Number of messages

The best performing node-oriented protocol was presented in [11]. The protocol consists of 10 instructions (6 are SEND) and requires participation of 4 different nodes. We will refer to it as NO_{BEST}. The best performing group-oriented protocol was presented in [9]. It has 41 instructions (24 are SEND) and might include cooperation of up to 34 nodes (but does not require such a number of distinct nodes). We will refer to is as GO_{BEST}. Our final hybrid protocol consists of 15 instructions (10 are SEND) before pruning and it might include cooperation of up to 7 nodes. We will refer to this protocol as HP_{BEST}. The pruned version of our protocol consists of 6 instructions (4 are SEND) and it might include cooperation of up to 4 nodes. We will refer to this protocol as HP_{FINAL}.

Figure 8 shows the number of messages sent by every node in protocol execution in our reference network with 7.5 legitimate neighbours on average. It can be seen that our final protocol has less messages sent with 5 amplifications than node- or group-oriented protocols with a single execution. As the communication

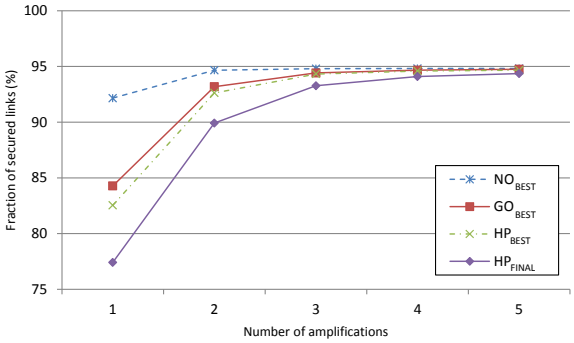

Fig. 7. Random compromise pattern (7.5 legal neighbours on average). Displayed results correspond to 50% originally compromised links. Node-oriented protocol performs better than others with only one amplification. Hybrid protocols perform comparably when a higher number of amplifications is performed.

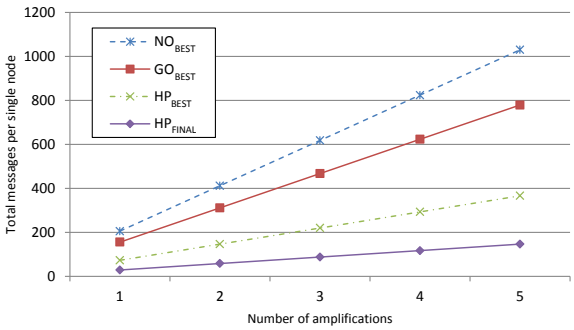

Fig. 8. Total number of messages per single node required to perform the best node-oriented and the best group-oriented secrecy amplification protocols (7.5 neighbours on average assumed). The hybrid protocol even with five amplifications sends considerably less messages than node or group oriented protocols with a single execution.

overhead is our primary metric for comparison of protocols, the proposed hybrid protocol gives more than 94% secure links (on average) from original 50% in our reference networks. This is a better result than node- or group-oriented protocol can provide with the same communication overhead.

The number of messages also depends on the number of participating parties and the average number of neighbours. Node-oriented protocols exhibit a polynomial increase of messages with respect to the number of neighbours in the network and an exponential increase of messages with respect to the number of communicating parties in the protocol execution. Group-oriented protocols exhibit only a linear increase of messages and the same dynamics holds for hybrid protocols. The growth in the number of messages depends on the count of SEND instructions within a protocol. This favours hybrid protocols even with a higher number of amplifications.

A.4 Test-bed results

To prove the practicality of the proposed protocol, we ran a prototype implementation for the TelosB hardware platform with the TinyOS 2.1.2 operating system and tested it in our laboratory network with 30 nodes positioned atop of nine interconnected offices.

Every node acts in three different roles according to a currently received message: *master* (being node N_C from the hybrid protocol), *slave* (being node N_P) and *forwarder* (being intermediate node). The implementation contains six phases executed mostly in parallel on all nodes in a network with a partial synchronization required only during the radio distance discovery:

1. A discovery of radio distance to neighbours – every node N_i periodically broadcasts *AM_MEASURE* message that is received together with the corresponding RSSI by its neighbours during the defined time-frame. Once broadcasting is finished, neighbours of N_i can compute the average RSSI value from the received packets, forming radio distance to N_i. Radio distance can be also computed from the RSSI of regular packets sent during ordinary network traffic, saving necessity to transmit special *AM_MEASURE* messages. This phase can be executed in parallel for all nodes with utilization of random back-offs between *AM_MEASURE* messages on different nodes to limit packet collisions.
2. A broadcast of measured distances to node's neighbours – once radio distances to other nodes are established, neighbours are notified about values measured by node N_i by the message *AM_DISTANCES* containing pairs of node's identification and its measured RSSI together with identification of measuring node. If node N_i receives the *AM_DISTANCES* message from a node that is its neighbour, measured values are stored locally. When node N_i receive measurements from all neighbours, next phase can be executed. Synchronization of remaining phases with other nodes is not required.
3. A computation of mapping to real nodes – mapping between nodes denoted in the hybrid protocol description and real nodes according to radio dis-

tances is performed locally. E.g., instead of node with $N_{0.69\_0.98}$ identification, a particular node N_i is selected. Note that mapping from the RSSI values distributed according to the logarithmic log-normal shadowing model of wireless signal propagation to the linear distance from a sending node is required. A different mapping model can be used where appropriate.

4. An execution of the hybrid protocol – node N_C executes the protocol as *master* to a selected neighbouring *slave* node N_P via intermediate *forwarder* nodes. Node N_C prepares its message with the sub-key as well as the routing path towards node N_P and sends it by the message *AM_SECAMPLIF*. Intermediate nodes act as simple forwarders with link transmission protected by already existing link keys. This phase can be executed in parallel for all nodes.
5. A verification phase – node N_C asks node N_P whether all sub-keys transmitted during the hybrid protocol execution or some were lost (e.g., due to packet loss) using message *AM_VERIFY*. If any sub-key is missing, a relevant sub-protocol for this sub-key is executed again.
6. A combination phase – all sub-keys, together with the existing link key between nodes N_C and N_P, are combined together using cryptographic hash function, forming the new shared link key. Optionally, a key confirmation can be executed before the old key is replaced by the new key value.

The hybrid protocol implementation has a small memory footprint – additional $(N*41)$ bytes of RAM are required (where N is the number of neighbours) and less then 3KB of additional code in EEPROM. Less then $(N*4*23+N*2*5+28)$ bytes of payload divided into about $(N*6)$ messages are transmitted on average during hybrid protocol execution by every single node (including verification messages, but excluding messages send during radio distance discovery phase and retransmission of lost messages). When 10 neighbours on average are assumed, around 1 KB of payload is transmitted by every node during secrecy amplification by the proposed hybrid protocol. *Master* node stores the current state of the hybrid protocol executed with the selected *slave* node, the *slave* node stores only received sub-keys and *forwarder* node stores no additional value. Due to the parallelization possibility, execution of hybrid protocols from the same *master* to different *slave* nodes can be interleaved without having long message buffers on a single node.

Times required to finish different phases are highly dependent on the network density and the signal propagation characteristics of the surrounding environment resulting in a different packet loss ratio. The prototype implementation performed was intended to verify memory, computational, transmission and synchronization requirements, not to provide detailed performance results for different environments and settings. Still, reasonable estimates about time required to finish separate phases can be inferred from experiments performed with our laboratory test-bed.

The radio discovery (phase one) took most of the time to complete as multiple *AM_MEASURE* messages had to be sent from every node in the network to obtain a reliable averaged RSSI value. Required time is roughly minutes or tens

of minutes to finish, depending on the required precision and network density (influencing the length of necessary random back-off to limit packet collisions). A broadcast of measured RSSI (phase two) is fast and requires only one or two messages, unless a high number of neighbours is present (more than 20). A mapping computation (phase three) is a fast local computation taking less than 1 second for a node with 10 neighbours and the optimized hybrid protocol with two sub-protocols described in Section 3. An execution of the optimized hybrid protocol (step four) takes 1-2 seconds, extending to tens of seconds when the packet loss is high and the verification phase (phase five) has to be executed repeatedly. Combination of received values by a hash function (phase six) is local and negligible.