On Secrecy Amplification Protocols

Radim Ošťádal, Petr Švenda, Vashek Matyáš

To cite this version:

Radim Ošťádal, Petr Švenda, Vashek Matyáš. On Secrecy Amplification Protocols. 9th Workshop on Information Security Theory and Practice (WISTP), Aug 2015, Heraklion, Crete, Greece. pp.3-19, 10.1007/978-3-319-24018-3_1 . hal-01442550

HAL Id: hal-01442550
https://hal.inria.fr/hal-01442550

Submitted on 20 Jan 2017

On Secrecy Amplification Protocols

Radim Ošťádal, Petr Švenda, and Vashek Matyáš
ostadal@mail.muni.cz {svenda, matyas}@fi.muni.cz

Masaryk University, Brno, Czech Republic

Abstract. We review most important secrecy amplification protocols that are suitable for ad-hoc networks of devices with limited resources, providing additional resistance against various attacks on used cryptographic keys without necessity for asymmetric cryptography. We discuss and evaluate different designs as well as approaches to create new protocols. A special focus is given to suitability of these protocols with respect to different underlying key distribution schemes and also to open issues.

1 Introduction

Ad-hoc networks of nodes with limited capabilities often handle sensitive information and security of such networks is a typical baseline requirement. Such networks consist of a high number of interacting devices, price of which should be as low as possible – limiting computational and storage resources. On top of the limited capability of the devices, there usually comes also the requirement of avoiding expensive tamper resistance. As a detection of an attack with limited resources is quite difficult, systems secure by design with a strong focus on autonomous self-defense are desired. Lightweight security solutions are preferable, providing a low computational and communication overhead. When considering key management in nodes of limited capabilities, symmetric cryptography is the preferred approach, yet with a low number of predistributed keys. While all results we present can be applied to general ad-hoc networks, we present them directly on wireless sensor networks (WSNs) as typical representatives.

Our work targets scenarios with ad-hoc networks where a link between particular nodes can be compromised, yet the nodes themselves are not. A typical example comes with some schemes based on symmetric cryptography, requiring suitable key distribution schemes (KDSs). During the attack, an 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. A strong majority of secure links ($> 90\%$) can be obtained even when the initial network compromise is at 50% [15]. This technique can be utilized in a broad range of scenarios, even if the particular results depend on a particular key distribution scheme and attack strategy.

The *contributions* of our work are: 1) a comparative review of all SA protocols (we are aware of), together with unified notation and taxonomy; 2) extensive multicriterial evaluation of all these protocols; and 3) identification of open research challenges in this area.

The SA 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 of [6]. SA protocols were shown to be very effective, yet for the price of a significant communication overhead. The overall aim is to provide SA protocols that can secure a high number of links yet require only a small number of messages and are easy to execute and synchronize in parallel executions in the real network.

Let us briefly present the principles of SA protocols and their most important features. Due to an attacker action, the communication link between nodes A and B secured by a link key K can be compromised. When the group of neighbouring nodes of A and B cooperates in an additional protocol, communication link(s) protected by the previously compromised key K can be secured again, if a new key K' can be securely transported to both nodes. If this is the case, there has to exist at least one non-compromised path. The exact way the new key value K' is transported specifies a particular secrecy amplification (SA) protocol.

The network owner usually does not know which concrete link key was compromised by an attacker and which was not. SA can be executed as a response to a partial compromise already happened or as a preventive measure for potential future compromise. SA can be also executed as another layer of protection even if a particular link key might not be compromised at all. Different key distribution schemes and related attacks correspond to different compromise patterns as described in Section 1.1, influencing how successful a SA protocol will be. SA protocols can try all possible paths, yet for the price of a huge communication overhead. Proposed SA protocols therefore aim to find a good tradeoff between the number of paths tried and the probability of finding at least one secure path.

SA protocols consist of the following principal steps:

1. Selection of neighbouring nodes participating in a given SA protocol.
2. Generation of new key values (shares).
3. Transport of key values (shares) via transport path or multiple paths according to the given SA protocol.
4. Combination of transported key values (shares) and existing old key into a new link key with an appropriate one-way function. New key will be secure if either old key or at least one of shares was previously secure.

The paper roadmap is as follows: the next subsection provides a short introduction to networks where a partial compromise is inevitable and one has to deal with compromise patterns resulting from different key distribution schemes and corresponding attack strategies. Section 2 provides a unified taxonomy of SA protocols and surveys previous work. Section 3 evaluates properties of SA protocols based on performance, memory and transmission overhead as well as ease of synchronization during massively parallel executions. Section 4 highlights open research problems and conclusions are provided in Section 5.

1.1 Partial network compromise

A wide range of key distribution, establishment and management techniques was proposed for sensor networks, see [3] for an overview. Distinct key distribution schemes behave differently when a network is under an attack targeted to disturb link key security. Although various schemes differ significantly 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 another link X is compromised after a relevant attack. The characteristics of a particular compromise pattern may significantly influence the success rate of an SA executed later. We will perform analysis of SA protocols according to the following two most prominent compromise patterns, but our work can be as well extended to additional patterns.

1.2 Random compromise pattern

The random compromise pattern arises when a probabilistic key pre-distribution scheme of [6] and many later variants of [2, 5, 8, 9] 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 a probabilistic pre-distribution scheme is used, then some additional links between non-compromised nodes become compromised as well. Probabilistic key pre-distribution schemes exhibit an almost uncorrelated pattern resulting from node capture and extraction of randomly selected keys.

1.3 Key infection compromise pattern

Compromised networks resulting from key distribution based on the idea of "key infection" [1] and later extended by [4, 7, 15] and others 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 eavesdropped by the attacker. 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 attacker node and the longer the distance between link peers, the higher the probability of a compromise. An eavesdropping of the exchanged key in the key infection approach of [1] does not compromise nodes directly, but compromises links in the reach of eavesdropper's radio instead.

2 Protocol survey

Different classes of SA protocols use different capabilities to improve security throughout the network. Although all SA protocols aim to setup new (possibly more secure) link key, three main distinct classes of SA protocols exist:

1. **A node-oriented protocol** sends key updates via every possible neighbour or neighbours by a simple protocol. Note that node-oriented protocol is executed for all possible k-tuples of neighbours in the network. A number of such k-tuples can be high, especially for dense networks.

2. **A group-oriented protocol** shares new key values inside a bigger group of cooperating nodes identified by their geographical areas in the form of relative distance to selected nodes.
3. **A hybrid-design protocol** uses sub-protocols (similarly to node-oriented), relative distances (similarly to group-oriented) and additionally utilize several repetitions of the whole process to achieve required success rate.

A summary of published protocols follows, with all details available in [13].

notation	description
A, B	identification of nodes for which the link key is strengthened during SA
C_i	identification of intermediate node(s) used during SA
N_C	identification of the central node during group-oriented SA protocols
N_P	identification of the node with a special role during group-oriented SA protocols
N_{d1_d2}	relative distance identification of a node with distance d_1 from N_C and d_2 from N_P
R_i	identification of a memory register
H	cryptographic one-way hash function
protocol instruction	description
NOP	no operation
$RNG\ N_a\ R_i$	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 the 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

Table 1: *Notation used for secrecy amplification (SA) protocols.*

2.1 Used notation

SA protocols can be described in the common form of message exchanges and operations executed on communicating nodes. Alternatively, each node in the protocol can be 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. SA protocol is then a sequential series of primitive instructions, manipulating values in memory slots and exchanging values between nodes. Some protocols require only one memory slot, but protocols with more than five different memory slots were also published. Latter case is more suitable to describe non-deterministic protocols without a fixed set of communicating peers and with execution differing at actual network layout and nodes positions (e.g., group-oriented protocols). Table 1 summarizes the used notation.

Using this set of primitive instructions, a simple plaintext exchange of a new key for node-oriented protocols can be written as {RNG N_1 R_1; SND N_1 N_2 R_1 R_1;}, a Push protocol [1] 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 [4] as {RNG N_3 R_1; SND N_3 N_1 R_1 R_1; SND N_3 N_2 R_1 R_1;}, a multi-hop version of Pull [4] 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;} and a multi-hop version of Push

[1] as {RNG N_1 R_1; SND N_1 N_3 R_1 R_1; SND N_3 N_4 R_1 R_1; SND N_4 N_2 R_1 R_1;}. Group-oriented and hybrid-design protocols consist from of the same type of instructions, but are typically longer and more complex, see [13].

2.2 Node-oriented protocols

The multi-hop (two-hop) and multi-path (number of neighbours reachable from both A and B) SA protocol was described in [1]. Node A generates q different random key values and sends each one along a different path over an intermediate node(s) C_i to node B, encrypted with existing link key(s). Key infection compromise pattern was assumed and simulations for attacker/legal nodes ratio up to 5% are presented, showing that the plaintext key exchange followed by the Push protocol is suitable within this attacker model. More detailed and precise simulations were later performed in [4]. The Push protocol is used as a basis for an establishment of the intra-group link keys between multiple nodes belonging to different groups when more structured deployment is assumed [10]. Multi-hop version of the Push protocol is analyzed in [11]. For the comparison, we assume Push protocol with one (denoted[1] as NO_3PUSH04) and two (NO_4PUSH04) intermediate nodes.

A variant of the Push protocol called Pull protocol was presented in [4]. The initial key exchange is same as for the Push protocol, but node C_i generates fresh key values that are used to improve secrecy of the key shared between nodes A and B instead of node A as in the Push protocol. The basic idea here is that the area where eavesdropping nodes must be positioned to successfully compromise the link key is smaller than for the Push protocol. The resulting fraction of compromised keys is then lower as an attacker has a smaller chance to place eavesdropping nodes properly. For the comparison, we assume Pull protocol with one (denoted as NO_3PULL05) and two (NO_4PULL05) intermediate nodes.

A variant of initial key exchange mixed with the Push protocol (denoted as Commodity) without explicit SA is presented in [7] together with formal security proof. We omit the Commodity protocol from the comparison as it is only a variant of the Push protocol, does not provide SA as a separate operation and the fraction of secured links is lower than for the Push protocol alone.

A linear genetic programming in combination with network simulator was used to design a node-oriented protocol [15] for the key infection pattern. Due to the nature of stochastic algorithms, a protocol was initially designed with up to 100 instructions storing intermediate values into up to 12 memory registers. Then was processed to omit all unused instructions and memory registers (based on performance provided by a network simulator), resulting in the 10 instruction protocol for four nodes (denoted as NO_EA09 for comparison).

As already mentioned, node-oriented protocols introduce a high communication overhead – all k-tuples of neighbours must be involved in a single execution

[1] For the rest of the paper, we will name protocols consistently in the form *protocolClass_protocolVariantYearOfPublication*, with additional compromise pattern designation when protocol was designed specifically for that pattern. E.g., NO_3PUSH04 means node-oriented protocol, Push variant with 3 participants, published in 2004.

of such a protocol. Another issue is an unknown number of direct neighbours and their exact placement. All neighbours can theoretically participate in the protocol and help to improve the fraction of secure links, but it is much harder to design an efficient protocol for ten nodes without unnecessary message transmissions instead of three or four nodes. 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 an insufficient number of participants.

In short, the main advantage of node-oriented protocols is simple synchronization of multiple protocol executions running in parallel and generally low memory overhead. The main disadvantage is the high number of messages transmitted, especially for the dense networks (see section Section 3.3 for details).

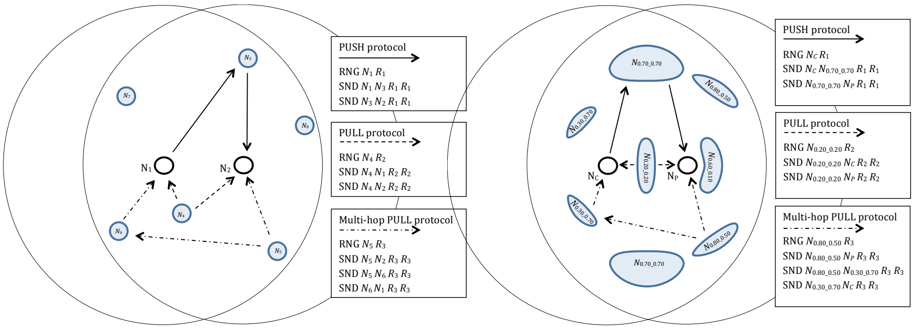
N_3
N_7
N_8
N_1
N_2
N_4
N_6
N_5
PUSH protocol
RNG N_1 R_1
SND N_1 N_3 R_1 R_1
SND N_3 N_2 R_1 R_1
PULL protocol
RNG N_4 R_2
SND N_4 N_1 R_2 R_2
SND N_4 N_2 R_2 R_2
Multi-hop PULL protocol
RNG N_5 R_3
SND N_5 N_2 R_3 R_3
SND N_5 N_6 R_3 R_3
SND N_6 N_1 R_3 R_3
$N_{0.70\_0.70}$
$N_{0.80\_0.50}$
$N_{0.30\_0.70}$
$N_{0.20\_0.20}$
$N_{0.60\_0.10}$
N_C
N_P
$N_{0.30\_0.70}$
$N_{0.80\_0.50}$
$N_{0.70\_0.70}$
PUSH protocol
RNG N_C R_1
SND N_C $N_{0.70\_0.70}$ R_1 R_1
SND $N_{0.70\_0.70}$ N_P R_1 R_1
PULL protocol
RNG $N_{0.20\_0.20}$ R_2
SND $N_{0.20\_0.20}$ N_C R_2 R_2
SND $N_{0.20\_0.20}$ N_P R_2 R_2
Multi-hop PULL protocol
RNG $N_{0.80\_0.50}$ R_3
SND $N_{0.80\_0.50}$ N_P R_3 R_3
SND $N_{0.80\_0.50}$ $N_{0.30\_0.70}$ R_3 R_3
SND $N_{0.30\_0.70}$ N_C R_3 R_3

Fig. 1: ***Left:*** *An example of instructions of a several node-oriented SA protocols. The Push, Pull and multi-hop version of Pull are included. A distance between nodes N_C and N_P is 0.5 of the maximal transmission range.* ***Right:*** *An example of instructions of a basic hybrid SA 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 geographic most probable areas, where such nodes will be positioned. A probabilistic layout shown is for the case where the distance between nodes N_C and N_P is 0.5 of the maximal transmission range. Notation used is according to the Table 1.*

2.3 Group-oriented protocols

In group-oriented protocols, an identification of the parties in the protocol is no longer “absolute” (e.g., node designation A, B, C), 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 a 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 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. Detailed description of group-oriented protocols is provided in [15].

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.

The protocol described in [15] consists of twelve instructions (denoted as GO_EA09 for comparison), but protocols with a better success rate were also generated by [14] (GO_EA12_KI and GO_EA12_RP). Group-oriented protocols consist of multiple times more instructions when compared with node-oriented protocols.

Due to the stochastic nature of the linear genetic programming used to generate group-oriented protocols, many different group-oriented protocols can be constructed based on the defined evaluation metric (fitness function). Evaluation metric can guide genetic programming towards protocols not only maximizing the fraction of secured links, but also to lower the number of messages exchanged, see [14]. In principle, new protocols can be generated for a particular usage scenario on demand, which is an interesting option.

In summary, the main advantage of the group-oriented protocols is a significantly lower (compared to node-oriented protocols) number of messages transmitted. The main disadvantage is the complicated synchronization of the parallel executions and also complicated security analysis due to the high number of nodes involved (e.g., the best performing group-oriented protocol presented in [14] 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.).

2.4 Hybrid-design protocols

Hybrid protocols [12] combine properties of both node- and group-oriented protocols. A protocol consists of several primitive instructions as described in Table 1. They were constructed with an application of knowledge from node-oriented and group-oriented protocols (thus hybrid design) and statistical data about the most suitable placement of the participating intermediate nodes.

A hybrid protocol is executed for every pair of neighbouring nodes instead of every k-tuple – same approach as in case of group-oriented protocols. Other participating intermediate nodes are used for transmission of n different values in the same fashion as previously described basic node-oriented protocols. Participating intermediate nodes are not required to store any forwarded values and can erase them as soon as a message with the value is forwarded to the next node towards destination. This allows for a simpler synchronization even within large and dense networks.

Steps of a hybrid protocol are similar to those of group-oriented protocols and also exhibit only a linear increase in the number of messages sent with respect to the number of neighbours. The main difference is independence of separate SA protocols executions and the fact that the key is updated only between nodes N_C and N_P in the last step. Relative distance from special nodes N_C and N_P is also used in the same way as for group-oriented protocols. Hybrid protocols contain a lower number of instructions and their construction, analysis and implementation are simpler than for group-oriented protocols.

Hybrid-design protocols optimized separately for key infection (denoted as HD_PULLPUSH14_KI) and random compromise (HD_PULLPUSH14_RP) patterns as well as for better tradeoff between overall success rate and number of messages (HD_PULLPUSHOPT14_KI and HD_PULLPUSHOPT14_RP) were proposed in [12].

As was observed early in [4], multiple repetitions of an SA protocol can additionally improve the number of secured links, yet for the price of additional multiplication of the total number of required messages. Hybrid-design protocols designed in [12] use three repetitions with the total number of messages still lower then for node- and group-oriented design.

In summary, the main advantage of the hybrid-design protocols is simple synchronization of parallel executions and low number of messages. The main disadvantage is the longer execution time due to multiple amplifications repetitions (but with possibility for parallel executions).

2.5 Comparison of general characteristics

Published SA protocols can be compared through several distinct characteristics:
Rules for selection of protocol participants – what neighbours and how they are included in SA protocol has a profound effect on the total number of protocol executions, overall number of messages transmitted and paths tested. Early protocols involved all neighbours indiscriminately (node-oriented) whereas later designs involved only nodes selected based on their relative positions w.r.t. to nodes controlling protocol execution (group-oriented and hybrid), resulting in probabilistic selection of nodes.
Design approach – early protocols were designed manually [1, 4, 7]; later came design with simulator-aided search for protocol settings, with stochastic optimization (genetic programming) [15] with semi-automatic postprocessing [12].
Number of involved intermediate nodes per single path – basic key exchange between A and B requires no intermediate node. If at least one intermediate node is used then the protocol performs so-called *multi-hop amplification*. The path is compromised if an attacker is able to eavesdrop at least one link on the path. If more then one intermediate node is involved, a suitable end-to-end routing protocol must be available.

Communication overhead – significant metric influencing protocol practicality due to energy-intensive radio transmissions necessary to transmit new key values during an amplification protocol. Communication overhead can be proportional to the network density (number of neighbours). The lower the number of messages, the faster the amplification phase is and the lower are the energy requirements.

Number of required repetitions – an additional iteration of a SA protocol can provide better results as links newly secured in a previous iteration can be used in the current one. Some protocols are simpler, but expect multiple repetitions whereas others expect only a single iteration.

Synchronization requirements – a SA protocol is usually not executed only between two nodes in the whole network, but between many different nodes in parallel. Degree of required synchronization is therefore an important characteristic, influencing speed of the SA phase as well as memory requirements on every node.

Number of distinct paths used to send new key values – if more than one path is used then the protocol performs so-called *multi-path amplification.* An attacker must eavesdrop all paths to compromise the new link key. If two nodes A and B exchange a new key directly in one piece, then only one path is used. Basically all SA protocols can be classified as multi-path to some extent if different intermediates or multiple repetitions are assumed.

Protocol	# intermediates	design approach	#primitive instructions	#msg/execution	#msg/node(4 neigh)	#msg/node(7.5 neigh)	#msg/node(20 neigh)	#repetitions	synchronization (1-3)
NO_3PUSH04 [1]	1	M	3	2	24	98	784	1-2	2
NO_3PULL05 [4]	1	M	3	2	24	98	784	1-2	2
NO_4PUSH04 [1]	2	M	4	3	72	804	21509	1-2	2
NO_4PULL05 [4]	2	M	4	3	72	804	21509	1-2	2
NO_EA09 [15]	2	A	10	6	144	1609	43019	1	2
GO_EA09 [15]	1-8	A	12	9	36	68	183	1	3
GO_EA12_KI [14]	1-31	A	35	23	92	173	467	1	3
GO_EA12_RP [14]	1-33	A	41	24	96	180	487	1	3
HD_PULLPUSH14_KI [12]	1-4	A	14	9	108	203	548	3	1-2
HD_PULLPUSHOPT14_KI [12]	1-2	A/M	6	4	48	90	244	3	1
HD_PULLPUSH14_RP [12]	1-5	A	15	10	120	225	609	3	1-2
HD_PULLPUSHOPT14_RP [12]	1-2	A/M	6	4	48	90	244	3	1

Table 2: *Basic characteristics of SA protocols. M/A means manual/automatic design approach respectively. Synchronization 1/3 means easy/difficult.*

2.6 Practical implementation

Practical implementation on the real nodes in existing work was provided only for hybrid-design protocols [12] on the TelosB hardware platform with the TinyOS 2.1.2 OS and tested with 30 nodes. The hybrid-design protocol implementation had 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. Times required to finish different phases was highly dependent on the network density and the signal propagation characteristics of the surrounding environment. The radio discovery took most of the time to complete as multiple messages had to be sent from every node in the network to obtain a reliable averaged RSSI value. Reported time was roughly minutes or tens of minutes to finish. Rest of the protocol took only few seconds. The node-oriented protocols requires similar instructions to be executed on every node, therefore memory and computational requirements would be comparable. The radio discovery phase does not require lengthly averaging of RSSI values, but number of protocol runs is significantly higher, depending on the network density.

3 Comparison of protocol performance

SA protocols are able to provide a significant increase in secure links, e.g., from 50% of originally secured links to more than 90% [15]. To achieve such an improvement, there is a considerable overhead in communication and on-node processing. In the subsequent section, we compare and evaluate all SA protocols we are aware of – w.r.t. to various metrics, including fraction of secure links newly secured by a protocol, communication and memory overhead, synchronization requirements. All comparisons are done on different compromise patterns.

Different initial settings can be used as a basis for the comparison, resulting in high number of combinations where SA protocols can be evaluated. First axis is formed by the selected initial compromise pattern – either random compromise or key infection pattern. Second axis is formed by the network characteristics, most importantly by the network density. For the comparison, we provide only selected combinations with complete results available[2].

3.1 Reference network and simulator

The following reference setting of simulator was used: network has 1000 deployed legal nodes and each node has 0.5 unit maximum transmission range. Target plane is a 13.8x13.8, 10.0x10.0 and 6.0x6.0 unit large that result in 4.0, 7.5 and 20.3 legal neighbours on average for networks with low, normal and high density respectively. Both random compromise and key infection patterns (see Section 1.1) were examined.

The evaluation of presented protocols is done using the same simulator that was developed specifically for security analysis of key distribution protocols and

[2] http://crcs.cz/papers/wistp2015

message routing by the authors of [15]. 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 SA protocol on fifty networks with 1000 nodes each in about 5 seconds when executed on one core CPU @ 2.7 GHz. Compare this to several minutes necessary to process only one network on OMNeT++ simulator.

Protocols evaluated in the simulator are described in a metalanguage of proposed primitive instructions, see the second part of Table 1 for more details.

3.2 Upper bound for amplification success

A modified Floyd-Warshall algorithm can be used to establish an upper bound for a given network, no matter what type of SA protocol is used. A single execution of the algorithm will find the shortest path between all pairs of vertices. When a graph is formed only from secure links, existence of the path between two nodes also implies possibility to transport and establish secure new key. As the precise compromise pattern for a given network is not known in advance (depends on an attacker, particular SA protocol, exact placement of nodes, etc.), we perform multiple evaluations for different networks to obtain an average result. As a side effect, we will also obtain lowest number of intermediate nodes necessary to transport new secure key.

There is a significant difference between two inspected compromise patterns. In the random compromise pattern, significantly more link keys can be secured than in the key infection compromise pattern. Also, that most benefit can be gained using only two intermediate nodes. With more nodes, the increase in secure links fraction is very small. Complete results of these experiments are provided in [13].

3.3 Number of messages

The number of messages sent during the protocol execution mainly depends on the protocol type. Nonetheless, it 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 particular protocol.

Figure 2 shows the number of messages sent by every node in the protocol execution on networks with different average number of legitimate neighbours. It can be seen that node-oriented protocols have advantage for networks with low density about 4 neighbours in average. The group-oriented and hybrid protocols are more suitable for dense networks.

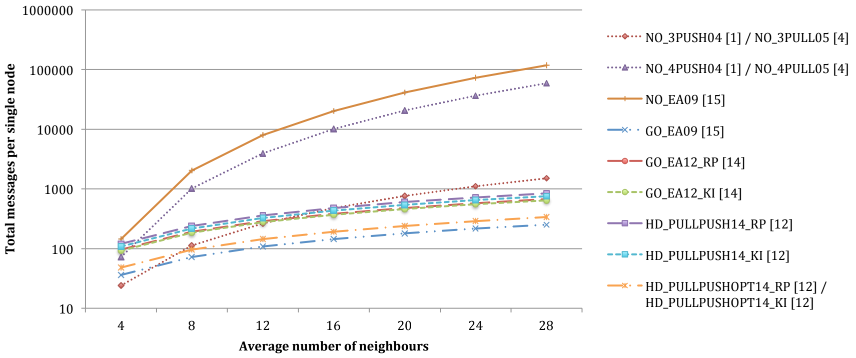

Fig. 2: *Total number of messages per single node required by a particular SA protocol. Even when group-oriented protocols utilise more messages per single execution and hybrid protocols utilise several protocol repetitions, the total number of messages is smaller than in case of node-oriented protocols for networks with higher density. Number of messages grows polynomially with the number of neighbouring nodes for node-oriented protocols compared to linear increase in case of group-oriented and hybrid protocols. Note the logarithmic scale of the y-axis.*

3.4 Success rate

We compare and evaluate all published SA protocols we are aware of w.r.t. to the fraction of secure links secured by particular protocol and also we compare the protocol effectiveness, which means the number of newly secured links for one message sent.

All SA protocols perform better with a rising density of network. The improvement is bigger for the random compromise pattern than for key infection (where the compromised links are concentrated in particular areas around eavesdropping nodes).

The impact of tested protocols for the random compromise pattern is compared in Figure 3, with additional results including the key infection compromise pattern in [13]. The HD_PULLPUSH14 protocols give us the best results regarding the overall success rate for both random compromise and key infection patterns regardless of the network density. NO_EA09 and HD_PULLPUSHOPT14 perform similarly, but there is a big advantage for the HD_PULLPUSHOPT14 considering the communication overhead of both protocols. There is no difference between NO_3PUSH04 and NO_3PULL05 protocol in case of random compromise pattern. NO_3PULL05 performs slightly better than NO_3PUSH04 on key infection. Both protocols are constantly in the lower half of success rate rating for both random compromise and key infection compromise patterns, however we can take advantage of their effectiveness for networks with low density where they present the best improvement compared to number of messages sent.

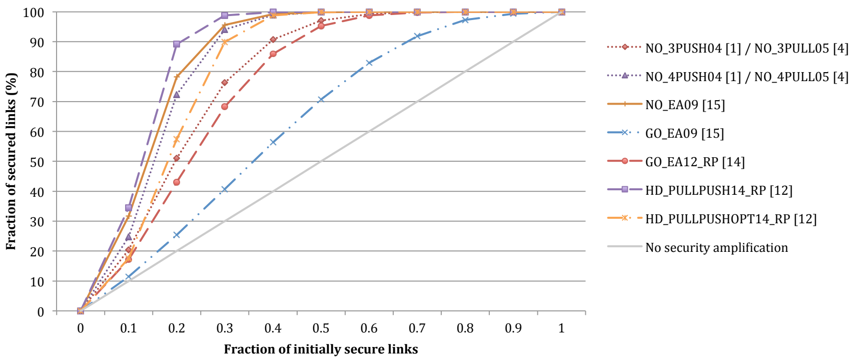

Fig. 3: *Increase in the number of secured links after SA protocols in the random compromise pattern on network with 20.3 legal neighbours on average. With increasing number of neighbouring nodes the general effectiveness of protocol grows. As can be seen, a strong majority of secure links (> 90%) can be obtained even when the initial network had 80% of compromised links. The best performing protocol is HD_PULLPUSH14_RP and it sends only little bit more messages than GO_EA12_RP. As can be observed, the 4-party node-oriented protocols show very good results on networks with high density. The least successful protocol is GO_EA09 because it was optimized for key infection pattern.*

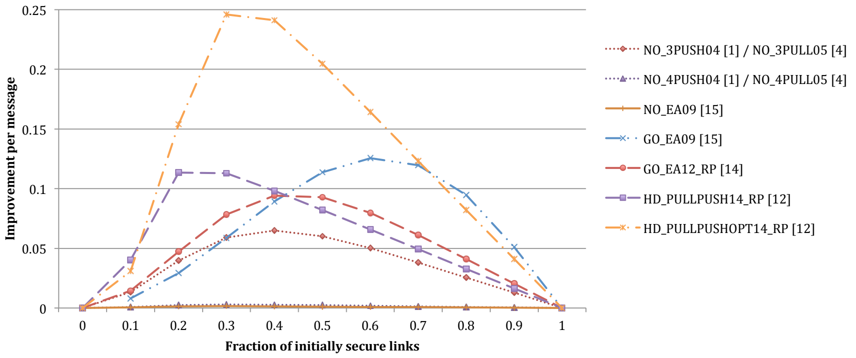

Fig. 4: *Increase in the number of secured links divided by the number of exchanged messages during the protocol execution (random compromise pattern, 20.3 legal neighbours on average). Node-oriented protocols send significantly more messages with rising network density. This stands especially for 4-party node-oriented protocols, which are the least effective. The best tradeoff shows group-oriented and hybrid protocols, while HD_PULLPUSH14_RP also outperforms the rest of protocols with regards to success rate.*

An increase in the number of secured links for one message sent during the protocol execution is showed in Figure 4 for random compromise pattern. Efficiency of node-oriented protocols with respect to improvement per message rate decreases with rising network density and remain more constant for group-oriented and hybrid protocols. The NO_3PUSH04 and NO_3PULL05 protocols are the most efficient for network with low density regardless the compromise pattern. They perform worse for a higher network density, but they are still better than 4-party node-oriented protocols. For networks with normal and high density, the most efficient protocol is HD_PULLPUSHOPT14. HD_PULLPUSH14 and GO_EA12 present very similar results regardless the network density or compromise pattern. They are in the middle spectrum compared to the rest of protocols. 4-party node-oriented protocols NO_EA09, NO_4PUSH04 and NO_4PULL05 give the worst results regarding the efficiency per message. It drops very quickly with rising network density.

4 Open research questions

So far, we inspected two compromise patterns in detail – the highly correlated key infection pattern for which the term secrecy amplification was originally coined, and the random compromise pattern without a significant correlation. As we have demonstrated, differences in the patterns have a significant impact on the success rate of SA protocol, rendering some parts of protocol vital for one pattern ineffective in another one. Is there a better approach than testing all possible SA protocols to obtain well performing and message efficient protocol? Can we analyze the compromise pattern and directly select an appropriate SA protocol?

We examined compromise patterns relating directly to the link keys randomly extracted from a nodes or eavesdropped by an attacker. Other attacker models have to be considered, based on attacker's interaction with a node. We considered that all keying material could be exfiltrated and the the node may continue working in an unchanged manner. Yet what if the attacker installs some malware and the node is under her control? How can that malware affect the behaviour of the node and what will be resulting compromise pattern?

The SA protocols were evaluated mostly for a flat network topology, where no node has a special status (e.g., cluster head) and initial keys were established in the same way for all nodes (e.g., same number of predistributed keys). More optimal protocols might be designed when these differences are taken into account. For example, if some nodes are equipped with a tamper resistant hardware (smartcards), but others are not, routing more messages via more resistant nodes during the SA can secure more links per messages transmitted. The different communication paths can be selected once a SA protocol is used inside cluster-based networks.

The SA phase usually takes a predefined time interval, provides fresh session keys and then finishes. But what if SA is performed in a continuous manner, producing fresh keys during the whole network lifetime? As a network in the production phase is usually exchanging many messages, the continuous SA may "piggyback" on these transmissions using already transmitted values without an additional message overhead. Some directions were already proposed in [11], but new problems need to be solved – how to maintain consistency of the current key on communicating nodes without an additional overhead, especially when the wireless transmission medium with a high packet loss is used? Can an attacker adapt his strategies like a selective node capture during the longer time-frame?

In the principle, the more paths are used to distribute key shares, the better is the chance to find a non-compromised one. But as the new key is constructed from all key shares, a missing or corrupted key share will render the resulting key incorrect. Therefore, the tradeoff between the resulting confidentiality (probability of establishing the non-compromised key) and integrity (probability of establishing a same value of shared key) exists. Yet, this perspective was not yet inspected in detail, with existing publications focusing mainly on the confidentiality part of the schemes. Protocols for threshold cryptography could be used to limit the impact of the corrupted key share, but these have to be executable with significant performance limitations.

5 Conclusions

Secrecy amplification protocols can significantly improve the fraction of secure links in partially compromised networks. These protocols were originally introduced for the key infection plaintext key exchange, but can be used also for a partially compromised network resulting from a node capture for the probabilistic pre-distribution and other partially compromised networks.

Node-oriented protocols are simple to execute in synchronized parallel executions and able to secure a high number of previously compromised links, but require a significant transmission overhead. Group-oriented protocols significantly decrease the transmission overhead and still provide a high number of secured links, but synchronization of multiple runs of secrecy amplification protocols executed in parallel between multiple nodes is their critical issue. Hybrid-design protocols share similar internal design with group-oriented protocols, but exhibit a significantly simpler synchronization of parallel executions. Multiple repetitions are generally required to obtain the same success rate as for other designs, but a lower number of messages in a single iteration provides a lower transmission overhead in total.

Even though every SA protocol class has its advantages and disadvantages, we identified several patterns that hold for both key infection and random compromise patterns. The HD_PULLPUSH14 protocols showed us the best results regarding the overall success rate. Its optimised version HD_PULLPUSHOPT14 is the most efficient protocol for networks with normal and high density. For networks with low density, the NO_3PUSH04 and NO_3PULL05 protocols are the most efficient.

SA protocols can make a network almost completely secure (more than 95% of secure links) when 60% of links are initially secure (probabilistic pre-distribution) or less than 10% ratio of eavesdropping nodes are present (key infection). When appropriate, SA should be executed as an additional strengthening mechanism after a basic key establishment.

References

1. Ross Anderson, Haowen Chan, and Adrian Perrig. Key infection: Smart trust for smart dust. In *12th IEEE International Conference on Network Protocols*, pages 206–215. IEEE, 2004.
2. Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, pages 197–213, 2003.
3. Haowen Chan, Adrian Perrig, and Dawn Song. *Key Distribution Techniques for Sensor Networks, Wireless Sensor Networks, ISBN 1-4020-7883-8, Kluwer Academic Publishers.* 2004.
4. Daniel Cvrček and Petr Švenda. Smart dust security-key infection revisited. In *Electronic Notes in Theoretical Computer Science*, volume 157, pages 11–25. Elsevier, 2006.
5. Roberto Di Pietro, Luigi V. Mancini, and Alessandro Mei. Random key-assignment for secure wireless sensor networks. In *1st ACM Workshop on Security of Ad Hoc and Sensor Networks*, pages 62–71, 2003.
6. Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In *9th ACM Conference on Computer and Communications Security, Washington, DC, USA*, pages 41–47. ACM, 2002.
7. Yong Ho Kim, Mu Hyun Kim, Dong Hoon Lee, and Changwook Kim. A key management scheme for commodity sensor networks. *4th International Conference on Ad Hoc and Wireless networks, LNCS 3738*, pages 113–126, 2005.
8. Donggang Liu and Peng Ning. Establishing pairwise keys in distributed sensor networks. In *10th ACM Conference on Computer and communications security*, pages 52–61. ACM Press, 2003.
9. Donggang Liu, Peng Ning, and Rongfang Li. Establishing pairwise keys in distributed sensor networks. *ACM Trans. Inf. Syst. Secur.*, 8(1):41–77, February 2005.
10. Zhihong Liu, Jianfeng Ma, Qiping Huang, and SangJae Moon. Storage requirements for key distribution in sensor networks. In *Second International Conference on Sensor Technologies and Applications*, pages 631–638, 2008.
11. Zhihong Liu, Jianfeng Ma, Qingqi Pei, Liaojun Pang, and YoungHo Park. Key infection, secrecy transfer, and key evolution for sensor networks. *IEEE Transactions on Wireless Communications*, 9(8):2643–2653, 2010.
12. Radim Ošťádal, Petr Švenda, and Václav Matyáš. A new approach to secrecy amplification in partially compromised networks. In *Security, Privacy, and Applied Cryptography Engineering – 4th International Conference, SPACE 2014, LNCS 8804*, pages 92–109, 2014.
13. Radim Ošťádal, Petr Švenda, and Václav Matyáš. *On Secrecy Amplification Protocols – Extended version, Technical report FIMU-RS-2015-01.* Masaryk university, Brno, Czech Republic, 2015.
14. Tobiáš Smolka, Petr Švenda, Lukáš Sekanina, and Vashek Matyáš. Evolutionary design of message efficient secrecy amplification protocols. In *12th European Conference on Genetic Programming*, pages 194–205, 2012.
15. Petr Švenda, Lukáš Sekanina, and Václav Matyáš. Evolutionary design of secrecy amplification protocols for wireless sensor networks. In *Second ACM Conference on Wireless Network Security*, pages 225–236, 2009.