Robust Secret Sharing for End-to-End Key Establishment with Physical Layer Keys Under Active Attacks

Stefan Pfennig, Sabrina Engelmann, Elke Franz and Anne Wolf

Abstract In recent years, there has been an increasing interest in physical layer key generation. Most of the approaches require that the communication partners who wish to generate a common secret key share a physical channel. That raises the question whether and how physical layer point-to-point keys can be used for a secure end-to-end key exchange over multiple hops. However, a multi-hop communication implies that an advanced attacker model has to be taken into consideration. In former work, we introduced possibilities for an end-to-end key exchange under the consideration of outsiders and passive insiders. Within this paper, we now focus on active insiders. The basic idea is to use a robust secret sharing scheme. We discuss the applicability and security of this method under different attack scenarios. Our investigations show that a secure key exchange is even possible with a high number of attackers in the system. Simulations allow an estimation of the expected costs in terms of communication overhead.

1 Introduction

Confidentiality and integrity of transmitted data are essential requirements on any data transfer. These protection goals are usually enforced by means of cryptography. Besides security, efficiency of data transmission plays an important role as well.

S. Pfennig (✉) · S. Engelmann · E. Franz · A. Wolf
Technische Universität Dresden, SFB 912 – Highly Adaptive Energy-Efficient Computing, Dresden, Germany
e-mail: stefan.pfennig@tu-dresden.de

S. Engelmann
e-mail: sabrina.engelmann@tu-dresden.de

E. Franz
e-mail: elke.franz@tu-dresden.de

A. Wolf
e-mail: anne.wolf@tu-dresden.de

M. Baldi et al. (eds.), *Proceedings of the 2nd Workshop on Communication Security*, Lecture Notes in Electrical Engineering 447,
DOI 10.1007/978-3-319-59265-7_2

Under the consideration of efficiency, symmetric cryptography is of special interest since it offers a performance that is magnitudes better in comparison to asymmetric cryptography. However, symmetric cryptography has the disadvantage that it requires the prior secure exchange of a secret key between the communication partners. This usually requires that a trusted third party is involved in the key exchange or that the communication partners already own a common secret key [4].

In recent years, there has been an increasing interest in physical layer key generation as an alternative to the common key exchange. Usually, a common channel between the two parties who wish to generate a secret key is assumed. This assumption implies, however, that only point-to-point keys can be established. There are also approaches that use a relay in the key generation what allows generating keys over more than one hop but a secret end-to-end key exchange over an arbitrary number of hops still remains a problem.

In former work, we studied possibilities for a secure end-to-end key exchange over multiple hops based on physical layer point-to-point keys [10]. The basic idea is to generate sub keys, to select different paths for the transmission of these sub keys, and to compute the secret key locally at the sender and receiver as XOR sum of all sub keys. As long as there is at least one trustworthy path, the establishment of a secret key is possible. As attacker model, we considered outside attackers as well as passive insiders (forwarders who are nice but curious).

The contribution of this paper is to consider also active insiders as attackers. We introduce a basic idea for a secure key exchange under this attacker model and discuss variations of the basic idea that are suitable for different scenarios regarding the number of passive and active insiders in the system. For the discussed solutions, we also evaluate the computational and communication overhead.

The paper is structured as follows. In Sect. 2, we explain our system model composed of the network as well as the attacker model. Section 3 gives an overview on the physical layer key generation. In Sect. 4, we explain the utilization of the physical layer keys to establish end-to-end keys on higher layers under active attackers. Section 5 concludes and gives an outlook.

2 System Model

2.1 Network Model

For the investigations within this paper, we assume a similar system model as introduced in [10]. Two communication partners, a sender $\mathscr{S}$ and a receiver $\mathscr{R}$, wish to establish a common secret key $k_{\mathscr{S}\mathscr{R}}$ for the protection of their data transmission by means of symmetric cryptography. Sender and receiver have to communicate over multiple hops, i.e., there is no direct link between these two parties. All messages are transmitted over intermediate nodes (relays) which we call forwarders $\mathscr{F}_{i,j}$ in the following. The assumed topology is shown in Fig. 1.

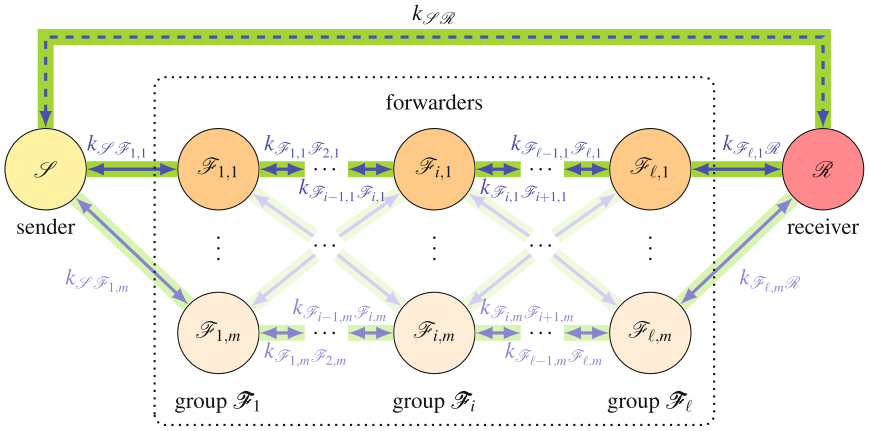


Fig. 1 System model similar to [10]

The forwarders are organized in ℓ different groups $\boldsymbol{\mathcal{F}}_1, \boldsymbol{\mathcal{F}}_2, \ldots, \boldsymbol{\mathcal{F}}_\ell$. Within each group $\boldsymbol{\mathcal{F}}_i$, there are m forwarders $\mathcal{F}_{i,1}, \mathcal{F}_{i,2}, \ldots, \mathcal{F}_{i,m}$. Each node of a certain group $\boldsymbol{\mathcal{F}}_i$ has direct links to all nodes on the neighboring groups $\boldsymbol{\mathcal{F}}_{i-1}$ and $\boldsymbol{\mathcal{F}}_{i+1}$. Thus, sender and receiver have direct links to all nodes in group $\boldsymbol{\mathcal{F}}_1$ and $\boldsymbol{\mathcal{F}}_\ell$, respectively.

This model is inspired by the envisioned topology of the high-performance low-energy computing platform HAEC that is currently under development [6]. The HAEC topology consists of a number of boards with optically connected compute nodes. In our model, the nodes of one board establish one group. All nodes of adjacent boards can directly communicate by means of wireless links.

For the investigation in this paper, we focus on the links between boards since we assume that these wireless links are used for the generation of physical layer keys. Particularly, we assume that each pair of nodes directly connected in our model has generated a physical layer point-to-point key $k_{\mathcal{S},\mathcal{F}_{1,j}}$, $k_{\mathcal{F}_{i,j},\mathcal{F}_{i+1,j'}}$ and $k_{\mathcal{F}_{\ell,j},\mathcal{R}}$ with $j, j' \in \{1, 2, \ldots, m\}$ and $i \in \{1, 2, \ldots, \ell - 1\}$ (Fig. 1) according to physical layer key generation mechanisms described in [2, 10]. The physical layer keys are used to protect the exchange of the end-to-end key $k_{\mathcal{S}\mathcal{R}}$ between sender and receiver.

2.2 Attacker Model

Basically, we assume that sender and receiver are trustworthy. They want to establish a common secret key in the presence of possible attackers. Accordingly, we also assume that adjacent nodes are trustworthy with respect to the generation of their physical layer keys since they are interested in protecting communication with their neighbors. However, forwarders are not necessarily trustworthy with respect to the end-to-end key exchange. To clearly describe our assumptions regarding the possible

attackers with respect to the end-to-end key exchange, we focus on the behavior of the attacker, his area of control, and his role within the system.

Regarding the *behavior*, we distinguish between passive and active attackers. A passive attacker only eavesdrops communicated messages, while an active attacker is able to disturb the communication. We especially focus on the case that an active attacker modifies or drops messages.

The *area of control* of an attacker defines how many links or forwarding nodes he controls. There can also be multiple attackers in the system. Regarding confidentiality, the worst case is given if these attackers cooperate; however, this case is equivalent to an attacker that controls all the links and nodes the single attackers can control. Similarly, it is sufficient to give the number of nodes controlled by one or more active attackers. For the discussions in this paper, it is only interesting to characterize the area of control of passive and active attackers individually.

The *role* of an attacker can be discussed at different levels [10]. For the physical layer key generation, an important distinction is whether the adjacent nodes who want to generate a point-to-point key have knowledge about the channel to the attacker (insider) or not (external entity). Since we discuss an end-to-end key exchange based on already established physical layer keys, we focus on attackers on the network layer. At this level, we can distinguish between insiders who participate in the transmission of data between sender and receiver and outsiders who do not participate. Of course, insiders are forwarders who transmit data. Outsiders may be other nodes in the network not selected as forwarders or external entities who may observe or jam the wireless communication, i.e., who access the links in our model. In our former investigations in [10], we considered outsiders and passive insiders only.

In this paper, we also consider active insiders; particularly, we focus on insiders as attacker since protection against outsiders can be achieved by link-to-link protection using the physical layer keys as discussed in [10]. We clearly distinguish between the two types of insiders who may control a certain number of nodes. There are e passive attackers (eavesdroppers) who are nice but curious, i.e., they transmit messages without any modifications but they cooperate to determine the secret end-to-end key $k_{\mathscr{S}\mathscr{R}}$. Additionally, there are a active attackers who are only interested in preventing a successful end-to-end key exchange. We assume, that active attackers usually do not cooperate, but we also evaluate limits if they collaborate. Moreover, if attackers are modifying as well as eavesdropping, we have to add them to both categories.

Even we focus on attacks on the network layer, there is no reason why an active attacker should not try to disturb the prior physical layer key generation. Thus, we want to give a short overview on this problem in the next section.

3 Physical Layer Key Generation

3.1 Key Generation

Key generation on the physical layer is widely discussed in literature nowadays (see [2, 7] for an overview). Mainly, there are two different approaches that need to be distinguished: the source-type model and the channel-type model. For both models, the main idea is the same: Two users, $\mathscr{A}$ and $\mathscr{B}$, want to generate a common secret key by means of the physical layer. For that purpose, they use a certain advantage they have over an eavesdropper $\mathscr{E}$, i.e., a passive attacker. For the source-type model, such an advantage is that $\mathscr{A}$ and $\mathscr{B}$ can observe correlated sequences from a common source of randomness, whereas $\mathscr{E}$ has either no access to this source or can only observe another sequence, which differs from the realizations obtained by $\mathscr{A}$ and $\mathscr{B}$. One example for such a source of common randomness is the current realization of their communication channel, which cannot be observed by an eavesdropper that is located at another position. For the channel-type model, the source of randomness is controlled either by $\mathscr{A}$ or $\mathscr{B}$ and the observation of this source is then transmitted over the wireless channel via a wiretap code. Exploiting their particular advantage over the eavesdropper, $\mathscr{A}$ and $\mathscr{B}$ want to agree on a common key. Ideally, they can communicate over an authenticated and noiseless public channel with unlimited bandwidth in order to exchange some information for the key agreement. The communication strategy has to guarantee that the key is kept secret from $\mathscr{E}$ who has perfect access to this public channel. Finally, both generate an individual key based on the information that is then available to them. The requirements for a secret key agreement are formulated as follows in [2, 7]:

1. The keys that are generated by $\mathscr{A}$ and $\mathscr{B}$ have to be equal with high probability.
2. The generated keys have to be independent of the public communication and the further observations of the eavesdropper.
3. The generated keys have to be approximately uniformly distributed over the key alphabet.

Here, it is assumed that all involved parties are allowed to know the applied codebook as well as the public communication strategy in principle.

3.2 Key Generation Under Active Attacks

As mentioned before, we assume that the two users $\mathscr{A}$ and $\mathscr{B}$ are interested in generating a common key and, therefore, are not malicious. Therefore, our attacker is another node of the system, which shares direct links with $\mathscr{A}$ and $\mathscr{B}$. In order to analyze key generation schemes under active attacks we need to distinguish between the two key generation models.

Active Attacks in the Source-Type Model
An overview on physical layer key generation under active attacks in the source-type model, where the common randomness is given by the random channel between $\mathscr{A}$ and $\mathscr{B}$, is given in [13]. Here, three different attack categories are distinguished:

- Disruptive jamming attacks: The attacker aims to prevent $\mathscr{A}$ and $\mathscr{B}$ from the successful generation of a key, e.g., by jamming during the channel probing phase.
- Manipulative jamming attacks: The attacker sends a signal in order to manipulate the channel measurements and therefore compromises the generated key.
- Channel manipulation attacks: The attacker aims to control the wireless channel between $\mathscr{A}$ and $\mathscr{B}$ such that he can infer the generated key.

For a key generation scheme, which defends against these attacks, we refer the reader to [13].

Active Attacks in the Channel-Type Model
Key generation schemes under active attacks in the channel-type model are not yet studied in the literature. Due to the fact that the channel-type model is similar to the secure communication over the wiretap channel, some of the results may be applied to the key generation, e.g., we can take a look on the results on the arbitrarily varying wiretap channel [3, 9]. Hence, we assume that physical layer keys can be generated even in the presence of active attackers and focus on end-to-end key establishment in the following.

4 End-to-End Key Establishment

4.1 Robust Secret Sharing for Key Establishment

In former studies, we excluded active insiders. Hence, a simple XOR of the sub keys was the basis to establish a common secret key $k_{\mathscr{S}\mathscr{R}}$ between sender and receiver in different hostile multihop scenarios. The attacker knows all sub keys transmitted over paths that contain at least one corrupted node. As long as one sub key is not known to the attacker, he has no chance to derive $k_{\mathscr{S}\mathscr{R}}$.

In this work, we focus on active insiders. Thus, we need another approach that allows for erased and modified messages (sub keys). To deal with erasures, the secret sharing by Shamir [12] looks promising. The idea of this scheme is to split a secret s into u shares in such a way that t of these shares are necessary to reconstruct the secret. The benefit of this scheme is twofold. On the one hand, it secures availability of the system against erasing attackers. Only t of u messages are needed to "decode" the information. On the other hand, it secures confidentiality against eavesdroppers. Less than t messages do not reveal any information about the shared secret.

Technically, a secret sharing scheme $\mathbf{ss}(u, t)$ computes u shares of a secret s by means of a polynomial $f(x) = \alpha_{t-1} \cdot x^{t-1} + \cdots + \alpha_1 \cdot x^1 + \alpha_0 \left(\cdot x^0\right)$ of degree $t - 1$. The coefficient α_0 is set to s, the other $t - 1$ coefficients α_i with $i \geq 1$ are

randomly chosen within a finite field $\mathbb{F}_p$. One share consists of a tuple $(x_i, f(x_i))$, where $x_i \in \mathbb{F}_p$ is a sample point ($x_i \neq 0$) and $f(x_i)$ is the function value. All the u shares should contain a different x_i. By means of Lagrange interpolation, any t tuples allow for recovery of the function $f(x)$. The desired secret is obtained by calculating $f(0) = \alpha_0 = s$.

If we apply this scheme to the key generation, the end-to-end key represents the secret. The sender can compute a number of shares and send these shares as separate messages to the receiver who needs only t of them to successfully reconstruct the key. The problem with secret sharing is, however, that the availability is only ensured when active attackers drop messages. If they modify messages, they distort the obtaining of the secret. At the end, a single modified message prevents the successful decoding. However, there exists a solution to this issue, namely robust secret sharing.

The goal of robust secret sharing is to ensure that the secret can be correctly reconstructed even if up to $u - t$ of the shares are corrupted by attackers. Using such a scheme, sender and receiver can be enabled to securely exchange a secret key even in the presence of active insiders. A number of robust secret sharing schemes has been introduced in the literature. In [8], the author recommend digital signatures. This is not an option for us, since we want to avoid asymmetric cryptography. Otherwise, we could establish a common key, e.g., by a Diffie-Hellman key agreement without using any physical layer keys at all. Another idea for the construction of a robust secret sharing scheme is to use Message Authentication Codes (MACs) for the protection of the shares [1, 5, 11]. The goal is here to enable the reconstructors to check the validity of the shares they want to use for the reconstruction of the share. Generally, the majority of the players need to be honest to ensure the correct reconstruction. In the cites approaches, a number of MACs are computed for each share and the necessary keys need to be distributed as well. Since we want to focus on the discussion whether a robust secret sharing scheme can be helpful for a secure end-to-end key exchange, we will use a simple scheme that requires less overhead. This scheme is introduced in the next section.

4.2 Proposed Basic Scheme

Inspired by the majority scheme introduced in [11], we can define a first simple construction of a robust secret sharing scheme. The key idea is to use two instead of one polynomial and to "hide" the secret s in the first polynomial $f(x) = \sum_{i=0}^{t-1} \alpha_i \cdot x^i$ and the squared secret s^2 in the second polynomial $g(x) = \sum_{i=0}^{t-1} \beta_i \cdot x^i$. Thus, a receiver can test whether the squared result of the first polynomial is equal to the result of the second, otherwise he will discard the result. The reason for squaring s is to avoid linearity between the two functions and, therewith, to prevent forgeability. The probability to forge a share in such a system depends on the size of $\mathbb{F}_p$ and is exactly $\frac{1}{p}$ by guessing, as long as the active attackers have not eavesdropped t or more parts.

In our system, there are ℓ groups with m nodes each. Thus, we can construct m parallel paths for the secret sharing. Hence, we can set $u = m$ for a uniform utilization of the forwarding nodes. Let us denote an **rss**(m, t) as a robust secret sharing scheme that is confidential against $e < t$ eavesdroppers and that stays available against $a \leq m - t$ active attackers. Such an **rss**(m, t) scheme has two functions: **share**(s, m, t) and **assemble**$(\mathbf{y} = [(x_1, f(x_1), g(x_1)), (x_2, f(x_2), g(x_2)), \ldots])$ with $|\mathbf{y}| = t$.

The first function **share**(s, m, t) creates two random polynomials $f(x)$ and $g(x)$ of degree $t - 1$ where $f(0) = \alpha_0 = s$ and $g(0) = \beta_0 = s^2$. For each of the m forwarders, it outputs a triple $(x_i, f(x_i), g(x_i))$ with $x_i \in \{1, 2, \ldots, m\}$. The **assemble** function uses t of those triples to recover $f(x)$ and $g(x)$ and tests whether $f(0)^2 = g(0)$. If this holds, **assemble** returns $s = f(0)$, otherwise it discards the result.

As stated previously, the reason for squaring s is to avoid linearity between the two functions and, therewith, to prevent forgeability. The third value of a triple works like a checksum. If we set the secret of the second function $g(x)$, e.g., to a fixed value, the receiver could recognize any change one the first and the third element of the triple. However, an attacker may just alter the second element of the triple, which would lead to a false assembled secret, but would not be recognizable. If we set the last coefficient β_0 of $g(x)$ equal to s or to a fixed multiple n of s, an attacker could easily add a value Δ to the function value of $f(x)$ and $\Delta \cdot n$ to the value of $g(x)$, which would yield a valid but forged secret.

The reason for this issue is the calculation of the Lagrange interpolation. Let us assume, we want to assemble t shares $(x_1, f(x_1), g(x_1)), \ldots, (x_t, f(x_t), g(x_t))$. To calculate the secret s, we use the Lagrange interpolation for finite fields. It holds:

$$\begin{aligned} s = f(0) &= \sum_{i=1}^{t} \left(f(x_i) \cdot \prod_{i \neq j} (-x_j)(x_i - x_j)^{-1} \right) \\ &= f(x_1) \cdot c_1 + f(x_2) \cdot c_2 + \cdots + f(x_t) \cdot c_t \end{aligned}$$

Since $\prod_{i \neq j} (-x_j)(x_i - x_j)^{-1}$ is equal for both functions $f(x)$ and $g(x)$, we can substitute this by a factor c_i. Moreover, we assume that the secret of the second polynomial $g(x)$ is a product of factor n and secret s of function one. Then it holds:

$$f(x_1) \cdot c_1 + f(x_2) \cdot c_2 + \cdots + f(x_t) \cdot c_t = \frac{g(x_1) \cdot c_1 + g(x_2) \cdot c_2 + \cdots + g(x_t) \cdot c_t}{n}$$

If an active attacker wants to alter a share, he adds Δ to $f(x)$ and $\Delta \cdot n$ to $g(x)$. W.l.o.g., we assume he adds it to the second triple, which leads to $(x_2, f(x_2) + \Delta, g(x_2) + \Delta n)$. We will now show that the equality remains:

$$f(x_1) \cdot c_1 + (f(x_2) + \Delta) \cdot c_2 + \cdots + f(x_t) \cdot c_t =$$

$$f(x_1) \cdot c_1 + f(x_2) \cdot c_2 + \cdots + f(x_t) \cdot c_t + \Delta \cdot c_2 =$$

$$\frac{g(x_1) \cdot c_1 + g(x_2) \cdot c_2 + \cdots + g(x_t) \cdot c_t}{n} + \Delta \cdot c_2 =$$

$$\frac{g(x_1) \cdot c_1 + g(x_2) \cdot c_2 + \cdots + g(x_t) \cdot c_t + \Delta n \cdot c_2}{n} =$$

$$\frac{g(x_1) \cdot c_1 + (g(x_2) + \Delta n) \cdot c_2 + \cdots + g(x_t) \cdot c_t}{n}$$

However, by using s^2 as secret for the second polynomial, an attacker who want to alter the second element of a triple, e.g., adding a fixed value like before, cannot forge the third value without knowledge about s. Moreover, we found out in some simulations that for fixed but arbitrarily chosen first and second values of a triple only one element of $\mathbb{F}_p$ is accepted by the **assemble** function. This third value (which is accepted) depends on s, i.e., there exists a bijective function from $s \in \mathbb{F}_p$ to a valid checksum. Without any knowledge about s, an attacker only could guess a valid third value with probability $\frac{1}{p}$. Thus, using s^2 as coefficient β_0 for the second function $g(x)$ prevents unseen attacks as long as s is secret, which means that an attacker eavesdropped less than t messages.

In comparison to other work, it seems to be contradictory that we want to achieve security even with a majority of active attackers. However, we need to keep in mind, that our attacker model is slightly more differentiated than attacker models in the literature. Thus, our a active attackers just want to corrupt the messages and do not cooperate. In case of clever adaptation to the secret, they need to eavesdrop t parts in the first place, which would make them to cooperating eavesdroppers and yield an increased e. Hence, the requirement in the related work that the majority of nodes have to be honest, i.e., $a < \frac{m}{2}$, is comparable to our assumption $a + e < m$ if we assume that dishonest nodes may eavesdrop and actively attack.

Without the possibility of eavesdropping, active attackers are only able to create valid shares of a wrong secret if they cooperate. Thus, we assume that the collaboration of active attackers is limited. Especially, when $a \geq t$, we need to assure that the largest number of active attackers that collaborate is below t. Put differently, if we construct j distinct sets of collaborating active attackers $\mathbf{A}_1, \ldots, \mathbf{A}_j$, we need to assure $\max_{1 \leq i \leq j} |\mathbf{A}_i| < t$. From this follows that $t > 1$ if $a > 0$. Otherwise, cooperating active attackers could easily construct valid shares for a wrong secret and prevent integrity.

Figure 2 illustrates an example of robust secret sharing. On the one side, the modification of the message by $\mathscr{F}_{1,2}$ is recognized by $\mathscr{R}$ and messages of $\mathscr{F}_{1,1}$ and $\mathscr{F}_{1,3}$ are sufficient to obtain the secret s. One the other side, neither $\mathscr{F}_{1,1}$ nor $\mathscr{F}_{1,3}$ individually can use their knowledge about one sample point to gain any information about s. Thus, confidentiality is preserved.

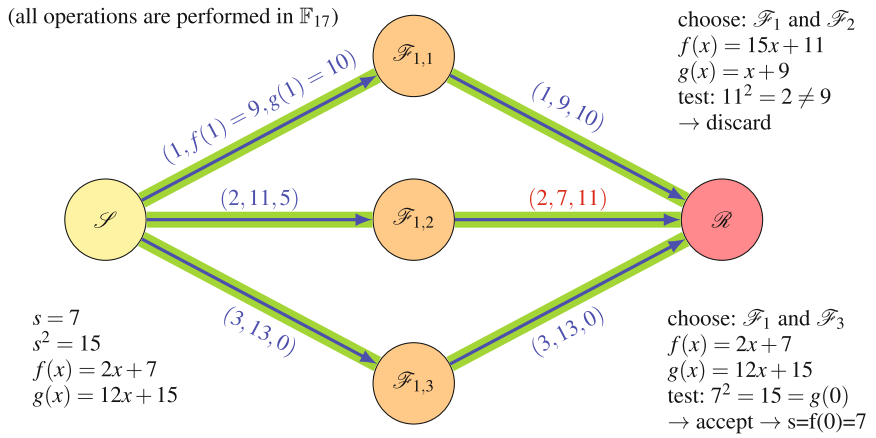


Fig. 2 Example of **rss**$(3, 2)$ scheme. The transmission of the secret $s = 7$ is confidential against $e < t$ eavesdroppers. Furthermore, the availability is ensured against $a \leq m - t$ active attackers

In the following sections, we will discuss how this basic construction can work under the consideration of different attacker models. This discussion necessitates to classify the relevant scenarios and explain why we need a different strategy there.

As a first step, we define the lower and upper limits with respect to the key generation. The lower limit is given by $a = 0$, i.e., there is no active attacker. This case is covered by the solutions introduced in [10]. For the upper limit, we could think of an attacker that controls a complete group of forwarders. Thus, it would be impossible to exchange a key since each message can be modified or dropped by the attacker. Moreover, even a forwarder group with $a + e \geq m$ is not controllable. For confidentiality reasons, it is necessary to fulfill $t > e$; for availability reasons, $m - a \geq t$ is needed. Thus, we get a contradiction, because $m - a > e \Rightarrow m > a + e$ ↯. Hence, the basic requirement for a successful key establishment is that at least one forwarder per group is not under the control of an attacker. Under this assumption, we can distinguish three classes that require a specific solution:

Case 1: $a + e < m$ – **rss** using m distinct paths
Case 2: $a + \text{maxPerGroup}(e) < m$ – XOR of many **rss** using all possible paths
Case 3: $\text{maxPerGroup}(a) + \text{maxPerGroup}(e) < m$ – nested **rss**

With a growing number of attackers, the costs in terms of computational and communication overhead for a certain solution increase. However, our main focus is to present a solution that allows for an end-to-end key establishment for every possible case, even for the extreme case that the ratio of corrupted nodes becomes very high.

The three cases apply a robust secret sharing scheme in different manners, thus, we can also use another robust scheme as basis. For example, we can also think of using the MAC-based solution introduced in [5]. However, this scheme implies a bigger communication overhead. In our scheme, a tuple consists of three values. The tuples to be sent in the MAC-based scheme consist of the function value, $n - 1$

MACs, and $n-1$ keys in case of n players. Hence, the communication overhead is smaller in the solution proposed in this paper. Hence, we will assume the use of this solution for the scenarios discussed in the following sections.

4.3 Case 1: Few Active and Passive Attackers

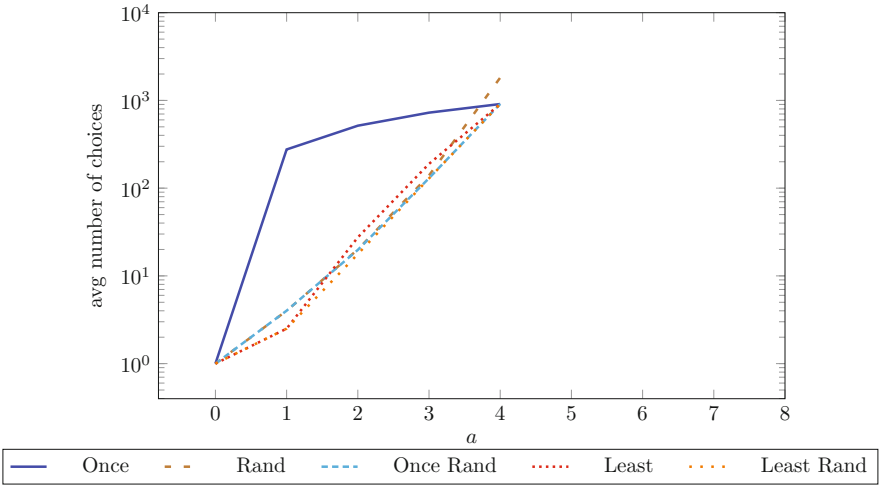
In this section, we assume the sum of active attackers and eavesdroppers $a+e$ of all forwarders is less than the number of nodes per group m. Thus, we can establish a key by means of one $\mathbf{rss}(m, t = m-a)$. Therefore, we send the m tuples via arbitrary but distinct paths to the receiver. For instance, we could use all "horizontal" channels that connect $\mathscr{F}_{i,j}$ with $\mathscr{F}_{i+1,j}$. In the following, we refer to the set of paths selected for the transmission of the m shares as link configuration.

The worst case from point of view of the honest users is given if all eavesdroppers observe distinct paths. Equivalently, the same holds for active attackers—the maximum damage can be caused if all active attackers are placed on different paths used by the sender to transmit the sub keys. Even in these worst cases, the construction is secure, because we assure confidentiality by $e < t = m-a$ and availability by $t = m-a$. However, if all a active attackers modified different tuples, the receiver will get m tuples of which only t are valid. Theoretically, this leads upto $\binom{m}{m-a}$ tests at the assemble function. Hence, we wanted to measure the real costs, especially when the number of active attackers is lower than the maximum a the system is designed for.

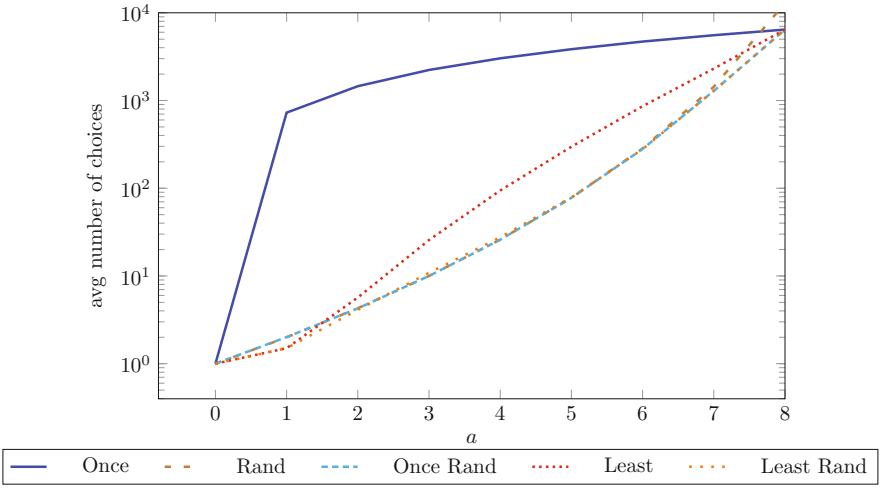
The receiver has different possibilities to select the t tuples:

- *Once*: make choice successively → no repetitions
- *Rand*: make choice at random
- *Once Rand*: make choice at random without repetitions
- *Least*: make choice based on least used items
- *Least Rand*: same as *Least* with random choice if there are multiple possibilities

We did simulations to obtain values for different m, ℓ, and a. In Fig. 3, we show results for $m = 16$, $\ell = 4$ and two different maximum numbers of $a = 4|8$. We performed 100, 000 runs for each strategy and each number of active attackers. The results indicate that *Least Rand* and *Once Rand* are good strategies to choose the t tuples for the assembling. *Least Rand* is even better for one attacker, because it is optimized for most changes within the test set. However, the benefit diminishes for more attackers. *Once* seems to be a bad strategy, when there are fewer active attacks than the system is designed to handle. This behavior was expected since, *Once* mostly replaces only one tuple and reuses other tuples quite long, which might be altered. Thus, it is advisable to use some random function in addition to *Once*. This structural issue effects also *Least*, which is a good option for one attacker but gets worse for more attackers, since some combinations of messages are chosen more frequently. *Rand* performs worst in cases, where a is maximal. The reason certainly is that there are many repetitions of the chosen set of t tuples. Comparing both plots,



(a) $\max(a) = 4$



(b) $\max(a) = 8$

Fig. 3 Averaged number of choices to get unmodified data for $m = 16$, $\ell = 4$

it is better to design the system to withstand more attackers than actually expected to lower the number of choices. However, this is only possible as long as there are few eavesdroppers ($a + e < m$).

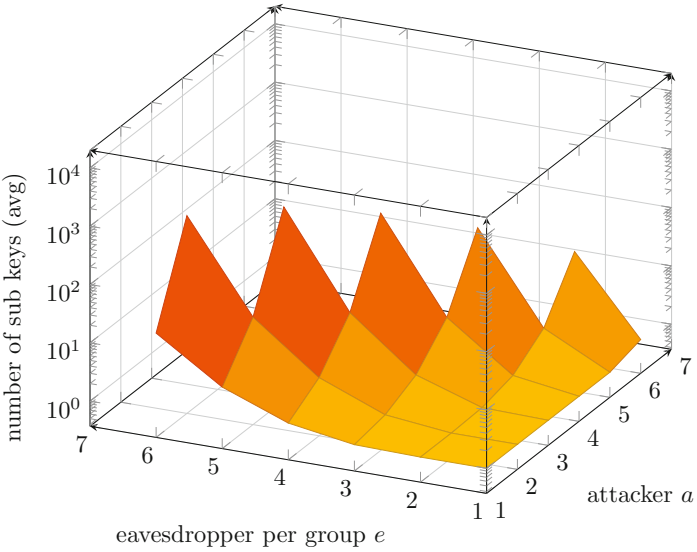
4.4 Case 2: Few Active and Many Passive Attackers

In the previous section, we could show that the **rss** scheme is suitable to protect availability against active attackers and confidentiality against eavesdroppers. However, if we assume more eavesdroppers in the system, a single **rss** scheme is not sufficient anymore. In this section, we assume the case $a + \text{maxPerGroup}(e) < m$. The function maxPerGroup(e) returns the maximum number of eavesdroppers on each of the ℓ groups. Thus, there may exist link configurations where the number of eavesdropped paths $\geq m - a$. Hence, confidentiality is not assured. However, there exist link configurations for m paths from sender to receiver with at most maxPerGroup(e), i.e., at most $m - a - 1$ eavesdropped links.

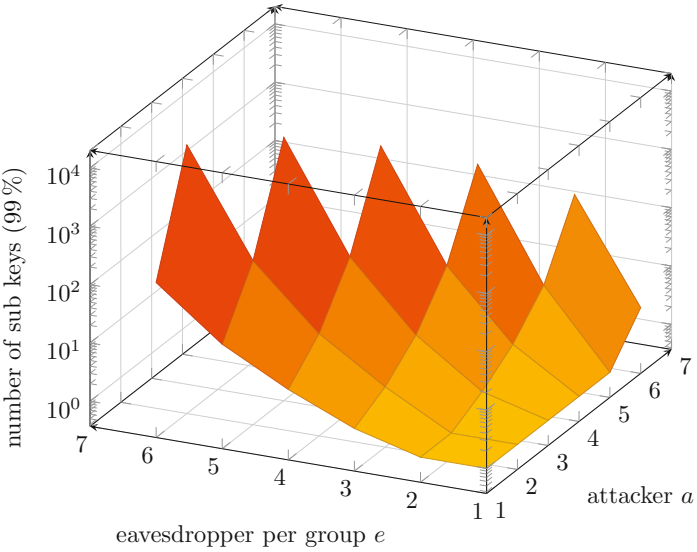
Using one of these "secure" configurations would ensure confidentiality as well as availability of the transmitted sub keys. The idea to ensure the selection of a secure link configuration is the following: We set $t = m - a > \text{maxPerGroup}(e)$ for many sequential **rss**(m, t) schemes. The shares of each scheme are transmitted using different link configurations. Each transmission delivers a sub key. Finally, these sub keys are XORed in order to compute the end-to-end key $k_{\mathscr{SF}}$. As long as at least one of those sub keys remains confidential, the summed secret remains confidential as well. There exists $m!^{\ell-1}$ link configurations of which each could transmit a sub key. However, this would yield in a large amount of sub keys and can be optimized. Since we only need to connect eavesdroppers with each other to limit the eavesdropped path to $m - a - 1$, we simply choose maxPerGroup(e) out of m per group and connect them. Thus, there will be at least one link configuration with the wanted properties. Overall, we need to exchange $\binom{m}{\text{maxPerGroup}(e)}^{\ell}$ sub keys to guarantee a confidential key establishment.

The costs for such an exchange depend on the actual parameters and, thus, may be quite high as well. Hence, we tried to figure out by means of simulation how many sub keys are really necessary, especially if $a + \text{maxPerGroup}(e)$ is clearly smaller than m. For the simulation, we constructed a random network of $m \times \ell$ forwarders and assigned a nodes in the whole network to be active attackers and e nodes per group to be eavesdroppers. Afterwards, we tried to achieve a confidential summed key by using random link configurations for a transmission with the **rss** scheme. We performed 10, 000 runs on different networks and 10 runs each with the same settings regarding the attackers. In this process, we counted the number of sub keys needed and evaluated average as well as maximum numbers.

In Fig. 4, we present the results for the example of a 9×3 forwarder matrix. We see that for any combination of e eavesdroppers per group and a active attackers greater or equal to the number of nodes per group m, there is no possibility to establish a common secret. However, if $a + \text{maxPerGroup}(e) < m$, we get a solution. The number of sub keys needed rapidly decreases with smaller e or a. If m is significantly larger than $a + \text{maxPerGroup}(e)$, then the costs are quite small. However, when using only as much sub keys as shown, we would achieve confidentiality only on average or in 99% of the cases, respectively. Thus, the actual number of sequential **rss** exchanges



(a) Average number of sub keys needed



(b) Number of sub keys sufficient for 99 % of the cases

Fig. 4 The number of sub keys to establish a confidential key for the example of $m = 9$ and $\ell = 3$

highly depends on the security demands. Moreover, in some extreme cases with very high costs, the solution introduced in the next section may be more efficient.

4.5 Case 3: Many Active and Passive Attackers

The last scenario represents the worst case for the establishment of a key. We assume $\text{maxPerGroup}(a) + \text{maxPerGroup}(e) < m$, which means that in the extreme case, there is only one trustworthy node in each group. We cannot simply send sub keys by means of sequential **rss**, because there may be link configurations where the number of actively attacked paths plus the number of eavesdropped links is larger or equal the number of all paths m. Thus, we can guarantee neither the confidentiality, nor the arrival of each sub key at the receiver. However, there also exists a link configuration that connects all eavesdroppers or all active attackers with each other, what leads to a fully trustworthy path and $a + e < m$ attacked paths.

To exploit each trustworthy node in a group, we need to think iteratively. If we assume $\ell = 1$, then it holds $\text{maxPerGroup}(e) = e$ and $\text{maxPerGroup}(a) = a$. Thus, it also holds $a + e < m$, which implies that the first scheme with one $\mathbf{rss}(m, t = m - a)$ scheme would work here. To extend this finding to the case where $\ell = 2$, we assume all forwarders in group 2 are kind of "intermediate" receivers that have to assemble the secret from the messages they got from forwarders of group 1. Each of the m "intermediate" receivers can receive a message (secret) securely by means of an $\mathbf{rss}(m, t = m - a)$ scheme via all predecessor nodes. Hence, a secure communication to all nodes in group 2 is possible. Furthermore, the sender could also send an encapsulated **rss**-tuple instead of a single secret to each "intermediate" receiver. Thus, after all nodes in the group assembled their secret (here: an **rss**-tuple), they can send it to the receiver, who can then assemble the secret of the sender. We denote this method as nested robust secret sharing (**nrss**).

In Fig. 5, we illustrate the procedure for $m = 3$. We need $m = 3$ tuples that are sent to a successor to enable him to assemble the secret. This secret may be an element of a further tuple. For this solution, we need to send a lot of messages that contain many elements. However, we can guarantee the availability and the confidentiality of the system, if $\text{maxPerGroup}(a) + \text{maxPerGroup}(e) < m$ holds and the single **rss** parameters are chosen adequately. In the following, we want to present a proof for the security of the suggested scheme as well as an identification of the communication costs in terms of number of messages to be sent and the lengths of these messages to establish a key with the receiver.

Theorem 1 *For a secure transmission of a key $k_{\mathcal{SR}} \in \mathbb{F}_p$ in an $m \times \ell$ forwarder matrix, it is sufficient to send m^ℓ messages of size 3^ℓ elements $\in \mathbb{F}_p$.*

Proof We will prove the theorem by mathematical induction. For the sake of convenience, we assume that the "x-points" for the secret sharing are smaller than p.

base case: The theorem holds for $\ell = 0$ (no forwarders).

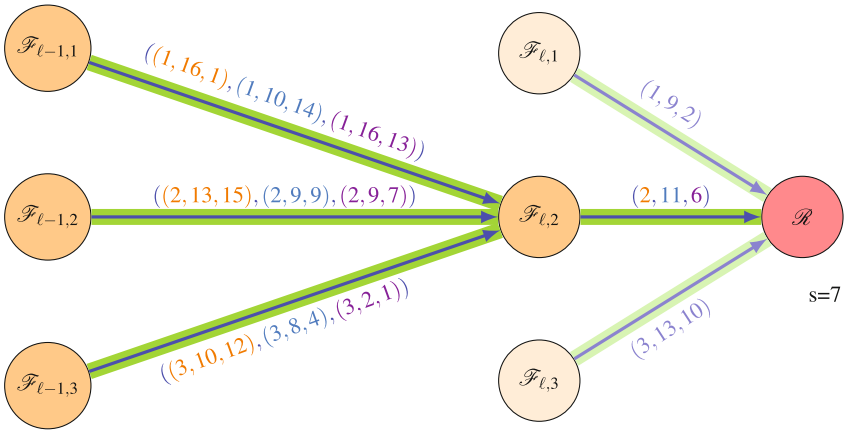


Fig. 5 Example of nested robust secret sharing scheme

messages: Communication is easily possible via a direct physical connection. We need exactly $m^\ell = m^0 = 1$ message from sender to receiver to communicate.
message size: To protect the transmitted key $k_{\mathscr{S}\mathscr{R}}$, we encrypt it by means of the physical layer generated key between sender and receiver. We need a message of length $3^\ell = 3^0 = 1$, since we only send one message and do not split the secret into shares that need to be transported.

induction step: Assuming the theorem holds for $\ell = n$, then we will show it also holds for $\ell = n + 1$.
messages: If we want to communicate with any node in group $n + 1$, we need to send an **rss**$(m, m - a)$ message via any node in group n. We have to use an **rss**$(m, m - a)$ scheme for the transmission. Hence, m messages need to be sent. Thus, we will send $m \cdot m^n = m^{n+1}$ messages. Since we only have a active attackers and e eavesdroppers in group n, the message can be assembled ($t \leq m - a$) and it stays confidential ($t > m - e$).
message size: The message size to a node of group $n + 1$ is tripled in relation to the message size to a node of group n because we have to transmit the x-value and its function values for the two polynomials each of $\mathbb{F}_p$ to enable the **rss** assemble function to get s of size q. Overall, we need $3 \cdot 3^n = 3^{n+1}$ elements per message. □

In Table 1, we illustrate the costs by means of elements we need to transmit in order to securely transmit a secret of size 1. For example, the sender has to transmit $19,683 \cdot 16 = 314,928$ Bytes for the transport of a 128 bit end-to-end key in a 9×3 forwarder matrix. Thus, this scheme is extremely costly, but we can prove the security. Furthermore, we see room for improvement, e.g., implicit x-values or heterogeneous finite fields, but this is not the focus of this paper.

Table 1 Transmitted data (in elements of $\mathbb{F}_p$) for end-to-end key exchange with nested robust secret sharing in dependency of m and ℓ

$m\backslash\ell$	0	1	2	3	4	5
1	1	3	9	27	81	243
4	1	12	144	1,728	20,736	248,832
9	1	27	729	19,683	531,441	14,348,907
16	1	48	2,304	110,592	5,308,416	254,803,968
25	1	75	5,625	421,875	31,640,625	2,373,046,875

5 Conclusion

We have shown that end-to-end key establishment with physical layer keys under active insider attacks is possible. We analyzed limits and presented three different approaches based on a robust secret sharing scheme. Within this work, we only focused on the general possibilities and their security but not on efficiency. However, we estimated the costs for our schemes and demonstrated that they are quite high, depending on the area of control of the attacker. Hence, we want to emphasize that the most important aspect to secure end-to-end key establishment with physical layer keys is to carefully assess the attackers strength and to set the parameters of the presented schemes in a reasonable way.

In future work, we want to optimize the schemes to enable applicability. One direction of future work is to analyze alternatives to the proposed robust secret sharing schemes, especially algebraic manipulation detection codes (AMD) introduced in [3]. AMD codes perform a probabilistic encoding of a source while decoding is deterministic. Undetectable modifications are only possible with a small error probability.

Further, a full implementation would allow for meaningful measurements in terms of energy and speed. Furthermore, we would like to assess the performance in comparison to a common key exchange that incorporates asymmetric cryptography under the consideration of different attacker models.

Acknowledgements This work is supported in part by the German Research Foundation (DFG) in the Collaborative Research Center 912 “Highly Adaptive Energy-Efficient Computing”.

References

1. Bishop A, Pastro V (2016) Robust secret sharing schemes against local adversaries. In: Proceedings of the Public-key cryptography-PKC 2016–19th IACR international conference on practice and theory in public-key cryptography, part II
2. Bloch M, Barros J (2011) Physical-layer security: from information theory to security engineering. Cambridge University Press, Cambridge

3. Boche H, Schaefer RF (2013) Capacity results and super-activation for wiretap channels with active wiretappers. IEEE Trans Inf Forensics Secur
4. Boyd C, Mathuria A (2003) Protocols for authentication and key establishment. Springer, Berlin
5. Cevallos A, Fehr S, Ostrovsky R, Rabani Y (2012) Unconditionally-secure robust secret sharing with compact shares. In: Proceedings of the advances in cryptology—EUROCRYPT 2012–31st annual international conference on the theory and applications of cryptographic techniques
6. Fettweis G, Nagel W, Lehner W (2012) Pathways to servers of the future: highly adaptive energy efficient computing (HAEC). In: Proceedings of the conference on design, automation and test in Europe
7. Liang Y, Poor HV, Shamai S (2009) Information theoretic security. Found Trends Commun Inf Theory 5(4–5):355–580
8. Michael O. Rabin (1983) Randomized byzantine generals. In: Proceedings of the 24th annual symposium on foundations of CS. IEEE Computer Society
9. Nötzel J, Wiese M, H Boche (2016) The arbitrarily varying wiretap channel—secret randomness, stability, and super-activation. IEEE Trans Inf Theory
10. Pfennig S, Franz E, Engelmann S, Wolf A (2016) End-to-End key establishment using physical layer key generation and specific attacker model (Lecture Notes in Electrical Engineering), vol 358, Chap. 6. Springer, Berlin, pp 93–110
11. Rabin T, Ben-Or M (1989) Verifiable secret sharing and multiparty protocols with honest majority. In: Proceedings of the annual ACM symposium on theory of computing. ACM, New York
12. Shamir A (1979) How to share a secret. Commun. ACM 22(11):612–613
13. Zeng K (2015) Physical layer key generation in wireless networks: challenges and opportunities. IEEE Commun Mag 53(6):33–39

http://www.springer.com/978-3-319-59264-0

Proceedings of the 2nd Workshop on Communication Security
Cryptography and Physical Layer Security
Baldi, M.; Quaglia, E.A.; Tomasin, S. (Eds.)
2018, XIII, 141 p. 27 illus., 14 illus. in color., Hardcover
ISBN: 978-3-319-59264-0