A Practical Man-In-The-Middle Attack on Signal-Based Key Generation Protocols

Simon Eberz*, Martin Strohmeier+, Matthias Wilhelm*, Ivan Martinovic+

*University of Kaiserslautern, Germany
+University of Oxford, UK

Abstract. Generating secret keys using physical properties of the wireless channel has recently become a popular research area. The main security assumption of these protocols is that a sufficiently distant adversary is unable to guess a generated secret due to the unpredictable behavior of multipath signal propagation. In this paper, we introduce a practical and efficient man-in-the-middle attack against such protocols. Using this attack, we demonstrate: (i) intentional sabotaging of key generation schemes, which leads to a high key disagreement rate, and (ii) a key recovery that reveals up to 47 % of the generated secret bits. We analyze statistical countermeasures (often proposed in related work) and show that attempting to detect such attacks results in a high false positive rate, questioning the overall benefit of such schemes. We implement and experimentally validate the attacks using off-the-shelf hardware, without assuming any technological advantage for the adversary.

1 Introduction

Communications over the wireless channel are affected by physical wave phenomena such as reflection, diffraction, or scattering, which contribute to a complex multipath behavior of transmitted signals. The measured channel response at the receiver is therefore considered a frequency- and position-dependent random variable that carries a certain amount of information entropy and can serve as a source of randomness. An additional physical property exploited in key generation protocols is channel reciprocity. If the channel response between the two transmitters, Alice and Bob, is sampled over a short time interval (depending on mobility patterns and the transmission frequency), both transmitters generate highly correlated estimates. Since sampling the wireless channel response is inherently given during any wireless message exchange, this approach offers an interesting alternative method to generate symmetric secret keys without relying on asymmetric cryptography. One of the main assumptions is that an eavesdropper (Eve) is unable to guess the generated bits because her view of the channel between Alice and Bob de-correlates rapidly with distance and thus results in inaccurate estimates. Concretely, it is assumed that if Eve is positioned at least half a wavelength λ away from Alice and Bob, then her estimates are de-correlated from those computed by Alice and Bob (for more information, see, e.g. [14]). Similarly, if an active attacker (Mallory) attempts to inject packets

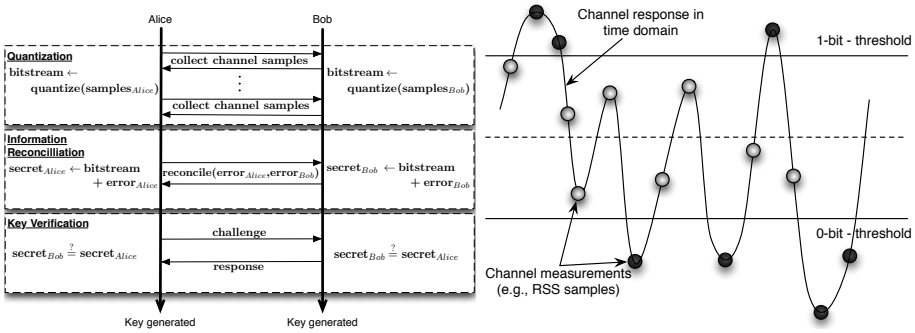
into the channel during key generation, he is unable to control how his signal is received at both sides, which results in a key disagreement. In case of the 2.4 GHz ISM frequency band, $\lambda/2$ is approximately 6.25 cm, which makes physical key generation attractive for WLAN and wireless sensor network applications.

The variety of existing protocols signify the importance of understanding the overall security of signal-based key generation schemes under a realistic adversarial setting. In this work, we assume an active attacker without additional knowledge or technological advantage. His only "toolbox" is the broadcast nature of the wireless channel that allows him to eavesdrop and inject packets at will. The main goal of a MITM attacker is to reveal the secret key generated by Alice and Bob. This is done by injecting his own information during the channel response estimation, which is subsequently used by Alice and Bob as part of their secret key. To avoid key disagreements that may lead to attack detection, he waits for *injection opportunities* that help him to keep the key generation protocol intact and still succeed. We also show that the attacker has an efficient way of forcing Alice and Bob to re-run the key generation protocol in case the number of opportunities for key recovery is too small, or simply to launch a DoS attack (we refer to this as *sabotaging attack*). To quantify the impact of these opportunities, we introduce the *attack efficiency* and *key recovery rate* metrics. As the goal of this work is to offer practical insights, we implement the key generation protocol by Mathur et al. [14] and evaluate our attack against it. Finally, we discuss countermeasures and show that an attempt to statistically detect our attack results in a high false positive rate, i.e., it leads to the rejection of a large number of legitimate packets required by the key generation protocol. Since Alice and Bob cannot be sure how many of Mallory's bits were successfully injected (in our experiments we were successful in revealing up to 47.4 % of the key) and this may be improved further by using better radio hardware, they are left without any reliable method on estimating the correct length of the secret, which questions the general applicability of such protocols.

1.1 Signal-Based Key Generation Protocols

In this subsection, we provide a bird's-eye view on physical key generation schemes (see [11,12] for detailed overviews). The three general phases that are shared by most signal-based key generation protocols are (see Fig. 1a):

Quantization Phase: Alice and Bob create a time series of the wireless channel response by exchanging packets and measuring channel properties. Examples for such properties are the received signal strength indicator (RSSI) and the channel impulse response (CIR). RSSI is often preferred because of its simplicity (it can easily be measured on a per-packet basis with off-the-shelf hardware). To create the initial secret bitstream, the series needs to be quantized by both nodes, i.e., the measurements need to be mapped to symbols. This requires calculating thresholds using a single threshold/multi-threshold approach or dynamic threshold schemes (for more details, see Section 6). Fig. 1b shows how measurements can be converted into bits by using two thresholds.



(a) Timing diagram, illustrating the communication between Alice and Bob, and processing steps during the protocol phases.

(b) A sample quantizer. All measurements above the 1-bit threshold and below the 0-bit threshold are converted into bits, resulting in a 1100100 bit sequence.

Fig. 1: A general overview of the signal-based key generation.

Information Reconciliation Phase: After quantization, the generated sequences at Alice's and Bob's side are likely to disagree because of noise and radio hardware artifacts. Both then apply information reconciliation methods to identify and correct such errors. Error correcting codes are one possibility to achieve this [6]; alternatively, many protocols use an interactive approach and reveal some information about their errors to reconcile their shared secret. If both nodes fail to agree on a common key, the samples are discarded and the protocol needs to be re-run. Some protocols also try to de-correlate their bitstream by using hash functions to extract randomness from the given imperfect input sequence [10], the so-called privacy amplification [4].

Key Verification Phase: Finally, both parties need to cryptographically verify the mutual secret. Usually this is done using a simple challenge-response protocol. An unsuccessful response constitutes a key disagreement and the process starts from the beginning. If protocols use dynamic thresholds, the quantization phase can be adapted by decreasing the number of possible thresholds, i.e., adapting a tradeoff between secrecy (the key length) and a successful key agreement rate.

2 General Idea of the Man-In-The-Middle Attack

The general idea of our attack is to "poison" the quantization phase between Alice and Bob. An active attacker attempts to impersonate both participants and to inject spoofed packets during the quantization phase, which are subsequently used in the key generation. In the best case, Alice and Bob agree on a common key of that Mallory knows a (preferably large) part.

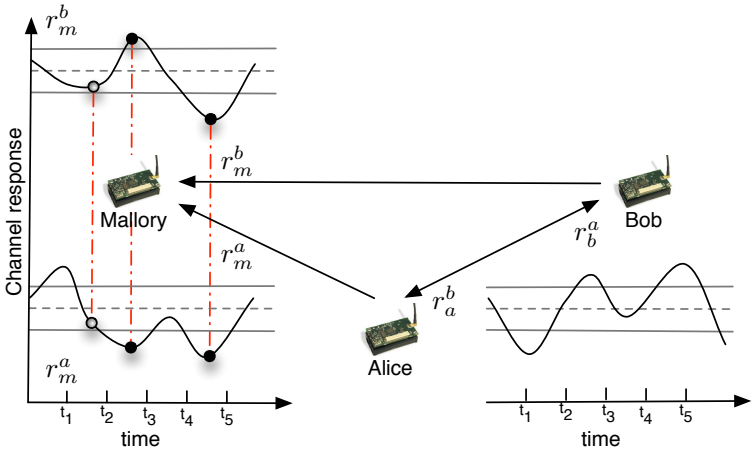


Fig. 2: Overview of the attack principle. Three different cases are depicted from Mallory's view of r_m^a and r_m^b: Case 1 in interval $[t_1 - t_2]$ is discarded as it lies within the thresholds. Case 2 in interval $[t_2 - t_3]$ is a sabotaging opportunity. Case 3 in interval $[t_4 - t_5]$ provides a key recovery opportunity.

2.1 Assumptions

We make the following assumptions about the attacker and the environment:

- The attacker adheres to all given security constraints and assumptions of physical key generation schemes. Specifically, he is not violating any constraints on the physical distance, such as being near legitimate transmitters.
- The attacker is always in transmission range of both Alice and Bob.
- The attacker is able to freely control his own transmission power up to a given (common) hardware limitation.
- The attacker is able to destroy legitimate packets sent by Alice and Bob when required, e.g., by employing reactive jamming as described in [18].

2.2 Injection Opportunities for Sabotage and Key Recovery Attacks

There is a number of challenges when injecting packets during the quantization phase. A naive attacker may send spoofed packets purely at random; however, he would not know how they are received. In consequence, this attack is futile and likely leads to a key disagreement because Alice's estimate of the injected packets differs greatly from Bob's. On the other hand, if the attacker constantly sends with a strong signal to superimpose Alice's and Bob's communication, he might be able to inject some packets but risks easy detection by statistical countermeasures. This means that we need a more sensitive approach to enable efficient control over the outcomes of our injected packets. The key idea of our attack is to find *opportunities* where we exploit the reciprocity of the channel in the same way as Alice and Bob use it to generate the correlated estimates.

We use a notation similar to [14]: r_x^y denotes the channel response received by node x from a probe signal sent by node y. The channel responses of two subsequent probes between Alice and Bob are thus defined as

$$\begin{aligned} r_a^b &= s \cdot h + n_a \\ r_b^a &= s \cdot h + n_b \end{aligned} \tag{1}$$

with s being the probe signal, n_x the independent noise process at node x and h a stochastic process describing the wireless channel between Alice and Bob. Furthermore, Mallory's overheard signals are

$$\begin{aligned} r_m^b &= s \cdot h_{bm} + n_m \\ r_m^a &= s \cdot h_{am} + n_m \end{aligned} \tag{2}$$

with h_{xm} denoting the channel between node x and Mallory. If Mallory is more than $\lambda/2$ away from Alice and Bob, h_{am} and h_{bm} are assumed to be uncorrelated with h.

However, while Mallory does not know *how* exactly his packet is received by Alice or Bob, he does know that the differential in the channel response is correlated. Hence, injected packets received by Alice or Bob preserve this differential. Assuming that n_m is similar and thus negligible at two subsequent measurements, the scenario in Fig. 2 shows two useful cases for Mallory's injections:

1. $r_m^b \ggg r_m^a$ (or vice versa): Mallory measures a large differential as seen in interval $[t_2 - t_3]$. Due to the channel reciprocity, it follows that for a spoofed answer by Mallory the responses are $r_b^m \ggg r_a^m$. Knowing that an injected packet will cause a highly differential channel response at both Alice and Bob, this constitutes an opportunity to produce highly differential estimates for Alice and Bob in the quantization phase ($\rightarrow$ sabotage attack).
2. $r_m^b \approx r_m^a$: Mallory measures a small differential as seen in interval $[t_4 - t_5]$. Here, it follows that for a spoofed answer by Mallory the responses are $r_b^m \approx r_a^m$. Knowing that an injected packet causes a similar channel response at both Alice and Bob, this constitutes an opportunity to generate similar values for Alice and Bob in the quantization phase ($\rightarrow$ key recovery attack).

2.3 Measuring the Success of MITM Attacks

We define several metrics for the two attacks to quantify the success of this approach in attacking physical key-generation protocols:

Sabotage Attack:

1. *Attack interval*: Defines how many probes made by Alice and Bob are sampled on average until a single disagreement bit can be injected. The ratio reflects the time to find opportunities and have a successful spoof showing up in the quantized bits. Obviously, the faster the attack is done, the better.
2. *Required spoof attempts*: This ratio measures how many spoof attempts are necessary to cause a single disagreement bit. Fewer attempts mean a reduced chance of detection for the attacker, thus it should be as low as possible.

Table 1: A summary of the notation used.

Symbol	Meaning
$d/d_{\max}$	(Max.) Perceived RSSI difference by the attacker
q_+/q_-	High/low threshold for excursions
$L/\widetilde{L}$	Messages exchanged for information reconciliation
α	Parameter needed for threshold calculation
m	Number of packets above/below threshold needed for excursion
h_u	Vector of channel estimates of node u
σ	Standard deviation of RSSI

Key Recovery Attack:

1. *Key recovery rate*: The success of the key recovery attack is measured by the number of bits of a secret key that are guessed by Mallory. Importantly, this measure is sensitive to wrong guesses as they rapidly increase the search space (i.e. the duration of the brute-force attacks)[1].
2. *Key recovery efficiency*: Defined as the percentage of spoofing attempts that are successfully injected and form a bit in the key. As the detection probability increases with the attacker's activity, a high efficiency is preferable.

3 Attacking a Concrete Key Generation Protocol

To illustrate the effectiveness of our attack concept in the real world, we apply it in a practical scenario. We consider the protocol described by Mathur et al. [14], the best representative, and implement it on standard off-the-shelf MicaZ hardware.

The measured wireless channel characteristic r of this protocol is the received signal strength indicator (RSSI), taken on a per-packet basis. The quantization phase consists of three separate steps: probing, quantization, and subsequent bit conversion. First, Alice sends a probe to Bob, who then responds with a probe of his own. These exchanges use a pre-defined frequency of 20 Hz (i.e., a 50 ms gap between probes). Both parties save the (highly correlated) received signal strength of the packets. This process is repeated n times, depending on the desired key length. When the probing completes, both Alice and Bob have obtained n estimates of the channel, which are saved as vectors h_a and h_b, respectively. They now independently calculate the thresholds $q_+^u = \text{mean}(h_u) + \alpha \cdot \sigma(h_u)$ and $q_-^u = \text{mean}(h_u) - \alpha \cdot \sigma(h_u)$, where α is a protocol parameter (0.5 in this case) and $\sigma(h_u)$ denotes the standard deviation of h_u. The results are quantized as follows:

$$Q(x) = \begin{cases} 0 & \text{if } x < q_- \\ 1 & \text{if } x > q_+ \end{cases}$$

[1] A bit-string of length ℓ with i errors results in an additional brute-force factor of $\sum_{i=1}^{\ell} \binom{\ell}{i}$.

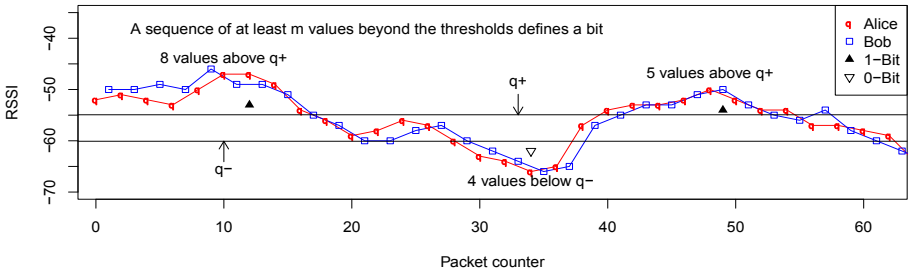


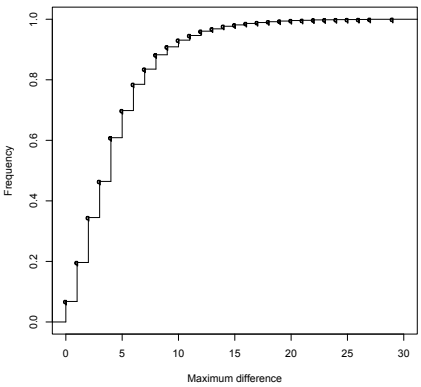
Fig. 3: Illustration of the bit generation process in our implementation of [14], with an excursion being quantized with 4 or more subsequent packets over threshold q_+ or below threshold q_-.

Alice and Bob then parse their measurements to find so-called *excursions*, i.e., m or more consecutive values in h_u that lie above q_+ or below q_- (where $m = 4$ is again a protocol parameter). An excursion above q_+ is converted to a 1-bit, while an excursion below q_- denotes a 0-bit. To reconcile the information, Alice sends a list of k excursions in the form of array indexes $L = \ell_1, \ell_2, \ldots, \ell_k$ to Bob. Bob checks if his measurements h_b contain excursions of length $\geq m-1$ at the locations specified in L. Subsequently, he sends back a list $\widetilde{L}$ that contains the indexes matching with excursions on his side. Excursions in L but not in $\widetilde{L}$ are dropped by both parties. After exchanging the L-messages, the quantizer function is applied to all elements defined by the indexes in $\widetilde{L}$ to form the bit string. Fig. 3 illustrates the process for our choice of $m = 4$. Alice and Bob should now have agreed on an identical key. A disagreement can only occur if m consecutive values lie above q_+ in h_a and below q_- at the same index in h_b or vice-versa. When this is noticed during key verification, the batch of bits is discarded and the protocol is restarted.

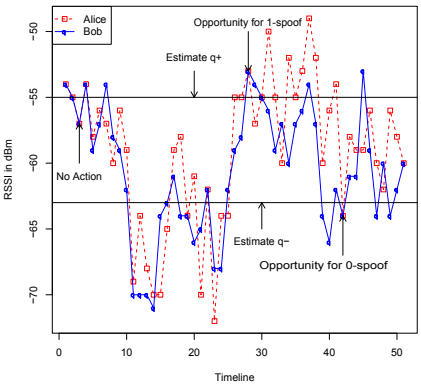
3.1 Implementation of the MITM Attacks

The experimental setup consists of two mobile motes (Alice and Bob) and one stationary attacker, Mallory. In our scenario, the two legitimate nodes and the attacker are in the same room. The distance between Alice/Bob and Mallory is always greater than 15 cm, as required by the security assumptions. Alice and Bob are moved independently within the room to create the necessary uncorrelated measurements. While this scenario does not make unreasonable assumptions, the attacker might not be able to be in the same room. Thus, in a second scenario with Mallory in a different room, we analyze whether the attack still yields satisfying results under these more difficult circumstances.

Detecting Attack Opportunities. The key parameter defining opportunities is the maximum RSSI difference $d_{\max}$ between probes. It is intuitive that the number of spoofing attempts increases when $d_{\max}$ is increased. After the probing



(a) Cumulative distribution of opportunities over RSSI differences as measured in a 2,000-packet run.



(b) Examples of key recovery opportunities as seen by the attacker. An opportunity is found when Alice and Bob's RSSI values are similar and exceed a threshold.

Fig. 4: Injection opportunities.

phase, Mallory creates two arrays h_a and h_b, containing his own view of the two independent channels between him and Alice/Bob, as illustrated in Fig. 4b. The difference d_i at packet counter i is computed as $d_i = |h_a[i] - h_b[i]|$. The optimal opportunity is at $d_i = 0$, but larger values of d are also suitable for the attack because only differences $d \geq \sigma(h_u)$ typically lead to a key disagreement. The results are summarized in Table 2 and Fig. 4a, showing that opportunities occur reasonably often. The number of excursions for the attack is sufficiently high as well, even if there are only a few of length $m \geq 4$ with $d = 0$. This does not constitute a problem, although it might reduce the attack's effectiveness.

Thresholds and their Estimation. Besides finding the perfect attack timing, one needs to estimate values for q_+ and q_-. Exact knowledge of both thresholds is not necessary; if a packet is part of an excursion, the attacker knows that it lies either above q_+ or below q_-. Fig. 5 illustrates this: an estimated threshold only causes a wrong guess if the assumed value of q_+ lies below the actual value of q_- (or vice-versa). With $\alpha = 0.5$, the difference between q_+ and q_- equals the standard deviation σ. Accordingly, any mistake in deriving both thresholds smaller than this standard deviation might result in fewer recovered bits, but does not lead to bit errors. To reduce the probability of a bit error and to increase the attack's robustness, a security margin is added to the estimated thresholds.

One method to estimate thresholds is scenario-based guessing, relying on the fact that average RSSI and standard deviation change only slightly between independent protocol runs. Such data can be collected for several scenarios and

Table 2: Number of opportunities in 8,000 packets and resulting excursions in our implementation of [14] (left). Real (q_+^A) and derived (q_+^M) thresholds (right).

d	Opportunities	Excursions
0	542 (6.8%)	15
1	1030 (12.9%)	77
2	1187 (14.8%)	132
3	955 (11.9%)	182

Run Nr.	q_+^A	q_+^M	$\lvert q_+^A - q_+^M \rvert$	σ
1	-52.7	-51	1.7	7.4
2	-49.5	-49	0.5	6.5
3	-51.3	-50	1.3	8.2
4	-53.1	-52	1.1	7.7

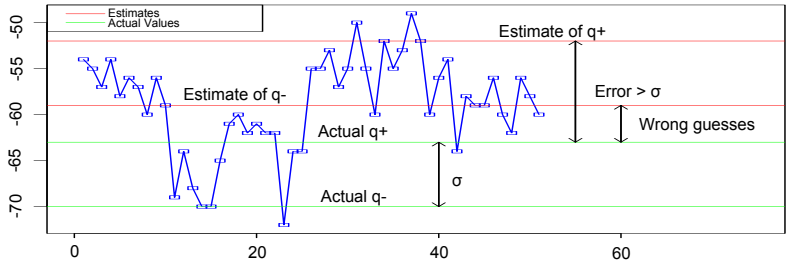


Fig. 5: Effects of inaccurate thresholds. Only the area between the actual q_+ threshold and the estimated q_- threshold is susceptible to wrong bit guesses.

used as reference for an attack. While this method has proven useful in our experiments, it may not be possible to find thresholds suitable for any setup, rendering it unpractical. Another possibility is to manipulate the setup phase of a protocol run. Algorithm 1 exploits the information about excursions that an attacker gains from the L-messages. Mallory waits for opportunities and sends spoofed messages without taking the thresholds into consideration. Afterwards, he checks the L-messages to find his own probes. If the number of spoof attempts was statistically significant then the thresholds should be well reflected in the attacker's spoof trace.

Table 2 shows that this approach yields very accurate approximations of q_+, the error $\lvert q_+^A - q_+^M \rvert$ being considerably lower than σ. However, deriving q_- failed, as too few successful spoofs were detected in the lower RSSI-spectrum. One possibility to deal with this is to ignore the negative threshold and to only use q_+ to detect 1-bits, which slightly reduces the overall key recovery rate. Another method is to simply define a sufficiently large distance x between q_+ and q_- and setting $q_- = q_+ - x$. As explained above, if this distance is greater than the standard deviation σ, this does not lead to bit errors. Considering the values of σ, $x = 10$ is a conservative assumption.

3.2 Sabotaging Attack

In the protocol of Mathur et al., a key disagreement occurs only if m or more packets are received with a difference in signal strength greater than the standard

Algorithm 1 Estimation of thresholds in the setup phase

Input : $d_{\max}$
Output : Estimates of q_+ and q_-
while $i < n$ **do**
  Receive packets i_{Alice}, i_{Bob}
  if ($|\text{RSSI}_{\text{Alice}} - \text{RSSI}_{\text{Bob}}| \leq d_{\max}$ **then**
    Send $m + 2$ spoofed probes to Alice and Bob
    spoofs.add(i, $\text{RSSI}_{\text{Alice}}$)
  end if
end while
Receive $\widetilde{L}$ from Bob
Sort *spoofs* descending by RSSI
$l :=$ *spoofs*.length
Check for longest sequence S in *spoofs*$[0, \ldots, \ell/2]$ with $a \in \widetilde{L}\ \forall a \in S$
$q_+ = \min(S)$
Check for longest sequence S in *spoofs*$[\ell/2 + 1, \ldots, \ell]$ with $a \in \widetilde{L}\ \forall a \in S$ and $|S| > 3$
$q_- = \max(S)$
if $q_- =$ null **then**
  $q_- := q_+ - x$
end if
Return q_+, q_-

deviation σ. Thus, to deliberately cause a bit error, an attacker sends packets when the difference between the RSSI of the last packets *exceeds* a pre-defined threshold $d_{\max}$. As no quantization is needed on the attacker's side, knowing the values for q_+ and q_- is not crucial, although they can help making the attack more precise by reducing the number of necessary packets. In our first implementation, the attacker simply waits until he receives two consecutive packets with a greatly differing RSSI and starts injecting packets. A single bit error is generally enough to force a complete restart of the protocol because no additional error correction schemes are implemented and the location of the error is unknown. In a further refined version, we altered the attacker's own sending strength to make the attack more efficient: every packet sent to the node with the higher RSSI uses the maximum sending strength; packets to the other node are sent with a significantly lower power while still allowing for the correct reception of the packet. This power adaptation ensures a greater difference in the reception of the packets and is more likely to create a disagreement excursion.

3.3 Key Recovery Attack

The attacker monitors the wireless channel and scans the received data for key recovery opportunities. If one is found, he starts to inject messages. To ensure an excursion, Mallory sends $m + 2$ unicast probes to both Alice and Bob. At the same time, Mallory stores whether the opportunity was triggered by a high or low RSSI value to determine the bit afterwards. Algorithm 2 describes this in more detail.

In addition to sending spoofed messages, the attack requires to destroy the legitimate packets sent by Alice and Bob. To simulate such a jamming effect,

Algorithm 2 Key recovery attack

Input : Estimates of q_+ and q_- from Algorithm 1, $d_{\max}$
Output : Known part of secret key
while $i < n$ **do**
  Receive packets i_{Alice}, i_{Bob}
  if ($|\text{RSSI}_{\text{Alice}} - \text{RSSI}_{\text{Bob}}| \leq d_{\max}$) & ($\text{RSSI}_{\text{Alice}} > q_+ \mid \text{RSSI}_{\text{Alice}} < q_-$) **then**
    Send $m + 2$ spoofed probes to Alice and Bob
    spoofs.add(i, $\text{RSSI}_{\text{Alice}}$)
  end if
end while
Receive $\widetilde{L}$ from Bob
for all j in $\widetilde{L}$ **do**
  if $\widetilde{L}[j] \in$ *spoofs* **then**
    $key[j] :=$ quantize($spoofs[j]$.rssi)
  end if
end for
Return *key*

upon receiving a spoofed probe, the motes voluntarily cease their transmission until the attack is over.[2] If the index of a spoofed packet appears in $\widetilde{L}$, the attacker can derive the RSSI of the packet from his own saved measurement and infer the resulting bit.

4 Results

4.1 Sabotaging Attack

For the sabotaging attack, we conducted 9 identical runs comprising 5,000 probes overall. The results in Table 3a show the efficiency of using a fixed transmission strength. While the success depends on the nodes' movement and the erratic nature of the wireless channel, we can assume with 95 % confidence that 142.37 probes are enough to cause one successful disagreement. Likewise, 7.17 spoofing attempts result in one disagreement. Assuming 2,000 probe messages are necessary to generate a key with reasonable length, this leaves roughly 93 % of the setup phase to recover the key while still ensuring a key disagreement with very high probability once the protocol run finishes.

Table 3b reflects the gain in efficiency when employing the adaptive sending power approach. In the previous version, 100 packets are not enough to achieve a reliable key disagreement; however, adjusting the sending strength raises the efficiency significantly. On average, the number of disagreements almost doubles for the same amount of probes or spoofing attempts. Again assuming a 2,000 packets run, the attacker now requires less than 4 % of the protocol's duration to sabotage the complete run with 95 % confidence. This comparatively small number of packets ensures that the distortion effect is kept minimal, preventing detection. In combination with the key recovery attack, the increasing efficiency

[2] Recent work [18] shows that reactive jamming is successful at rates > 99.9 %.

Table 3: Results of the sabotaging attack with 95 % confidence intervals. Both metrics improve significantly when adjusting the attacker's sending strength.

(a) Constant sending strength. (b) Adjusted sending strength.

	Attack interval	**Required spoof attempts**	**Attack interval**	**Required spoof attempts**
Mean	113.58	6.01	62.41	3.30
Variance	1403.25	2.30	333.25	0.61
Error	12.49	0.50	6.09	0.26
Upper limit	84.79	4.84	48.38	2.70
Lower limit	142.37	7.17	76.45	3.90

enables the attacker to start sabotaging at a later point in the setup phase, thus generating more accurate thresholds.

4.2 Key Recovery Attack

The results of the first scenario with all motes in the same room are documented in Table 4. Note that the threshold estimates are close to the actual values, which helps to mitigate bit errors. The most conservative setting $d = 0$ results in about 40 % of the key being revealed (assuming a length of 64 bit, this would speed up a brute force attack by factor 2^{23}) and indeed the highest key recovery efficiency. More than half of the sequences sent by the attacker cause an excursion with both Alice and Bob. Increasing the maximum difference to 1 reduced the efficiency below 50 %, but greatly increased the key recovery rate. Further increase of the tolerance level decreases the efficiency with no benefits to the percentage of the key known to the attacker. Another insight gained from the results is that the revealed bits are almost exclusively 1-bits. This can be explained by the fact that the difference in the reception of spoofed packets at Alice and Bob increases with the distance between Alice/Bob and the attacker. However, this is not a real issue if the overall number of bits is sufficient because the attacker is not interested in specific random keys.

The results of the second scenario with Mallory in a different room show that the attack performs better if the attacker is physically close to the conversation partners. Both key recovery rate and efficiency are about halved. The number of successfully created excursions above q_+ has decreased to near zero and most of the retrieved bits are 0-bits. This is intuitive because the attacker does not increase his sending strength enough to match the weakening caused by the wall. On the other hand, due to the weakened signal strength, the condition of receiving signals below q_- is fulfilled most of the time. This results in a rather poor key recovery efficiency. Yet, the attack is successful independent of the physical proximity of the attacker. The efficiency can easily be improved if the attacker is able to use superior antennas as well as to increase the sending strength without being limited by regulations or power consumption.

Table 4: Key recovery attack results for two scenarios and different $d_{\max}$.

	Same room			Different rooms		
$d_{\max}$	0	1	2	0	1	2
q_+ (actual/assumed)	-57.4/-55	-54.2/-55	-53/-55	-53.8/-52	-53.2/-52	-53/-55
q_- (actual/assumed)	-65/-65	-62/-65	-62/-65	-61/-62	-61.3/-62	-62/-65
Spoof attempts	76	91	130	55	78	130
Bits recovered (0/1)	10/32	12/33	5/27	11/3	14/2	14/4
Resulting key length	108	95	84	64	69	71
Key rec. efficiency [%]	**55.3**	**49.5**	**24.6**	**25.5**	**20.5**	**13.8**
Key rec. rate [%]	**38.9**	**47.4**	**38.1**	**21.9**	**23.1**	**25.3**

5 Possible Countermeasures

There is one obvious countermeasure to spoofing attacks: Alice could recognize that Mallory is impersonating her when he is sending a packet that carries her own MAC address. Yet, the standard setting in most commercial wireless adapters is to discard such packets, not to forward them above the MAC layer. Furthermore, wireless network interfaces are typically not able to send while in monitor mode, rendering this approach impractical for most applications. The authors of [14] propose a scheme that generates radio fingerprints, which are then used to distinguish legitimate from spoofed packets [19]. Obviously, the attacker must not be present at this point of time, which is a strong assumption. Additionally, the fingerprint is bound to the receiver/transmitter pair as well as the environment where it was generated. This makes it impossible to pre-generate fingerprints in a safe environment. In this section, we look at defensive schemes that would allow Alice and Bob to generate identical keys despite Mallory's presence, while ensuring a high level of uncertainty for the attacker. To achieve this, they have to selectively remove a high percentage of Mallory's packets and keep the false positive rate as low as possible. In any case, such an approach would limit the secret key rate because the original messages cannot be recovered. Other countermeasures include the introduction of time into the given protocols. This opens additional statistical detection vectors but is likely to introduce new difficulties on its own and is not currently present in the discussed protocols.

Table 5: Effects of packet-based filtering for three selected thresholds.

(a) Attacker involvement.

Threshold [dB]	Discarded spoofs	Discarded legit
10	216 (61.0 %)	522 (31.7 %)
15	126 (35.6 %)	216 (13.1 %)
20	54 (15.25 %)	78 (4.74 %)

(b) Standard run.

Threshold [dB]	Discards
10	564 (28.2 %)
15	150 (7.5 %)
20	36 (1.8 %)

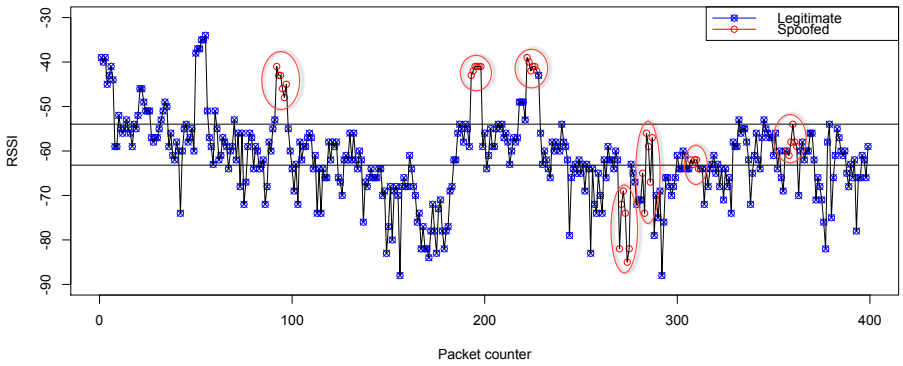


Fig. 6: Packet trace with spoofed and legitimate packets. While there are spoofs that seem like outliers at first, overall they are difficult to distinguish from a large number of legitimate packets with similar or even more extreme values.

5.1 Packet-Based Detection

One approach is to discard a packet when the RSSI value differs from the previous packet by more than a threshold t. However, the signal strength of the following packets in the spoof sequence is relatively consistent. So, when an attack is detected, Alice and Bob must not only discard the packet that caused the detection but also the following $m - 1$ packets (where m is the unknown number of packets sent per spoof attempt). Fig. 6 shows a pattern generated by the attacker's activity, illustrating how some of the spoofed packets might look suspicious. Yet, legitimate packets often follow similar patterns, making false positives likely. Table 5 lists the results of the filtering mechanism in a 2,000 packet run. With a strict threshold of 10 dB, the majority of spoofed packets (61 %) is rejected. However, a large part of the legitimate ones is filtered as well, leaving only 63 % of all packets for quantization. Raising the threshold to 15 dB, only one third of the injected packets is removed and the key length is reduced by 17 %. A threshold higher than 15 dB has negligible effects on both spoofed and legitimate packets. If the same filtering mechanism is used without the presence of an active attacker, the impact is smaller but a threshold of 10 dB still removes more than a quarter of legitimate packets in every run. Thus, to generate the same key length the number of packets must be increased by one third. That is, packet-based detection requires to prolong the run if suspicious packets were discarded. And because attacks are rare events it is not desirable to severely limit the average performance of the system.

5.2 Run-Based Detection

Another approach, conceptually close to [19], is to determine whether an attack occurred after a complete protocol run to potentially discard the run. There

Table 6: Percentage of packets over given distance from median RSSI.

Min. difference from median	Standard [%]	Spoofed [%]	Standard (LOS Break) [%]
10	22.75	25.2	27.4
15	11.6	16.7	14.3
20	3.5	5.2	5.4
25	1.2	1.1	1.1

are several statistics that could be altered predictably by an attack, such as the variance of RSSI values. We also tested this method against our implementation.

Table 6 summarizes the results for different scenarios and shows that reliably accepting the legitimate run also means not to detecting an attacker. Normal occurrences, such as breaks in the line-of-sight between Alice and Bob, render at least this simple implementation of the run-based detection unsuccessful. Even an imperfect reference value, causing few false positives, would require a large amount of training data because its variance strongly depends on the scenario.

6 Related Work

In 1993, Maurer introduced a concept that describes an abstract broadcast channel accessible to three parties. This channel provides strongly correlated information to two parties and weaker correlated information to the third party [15]. Consequently, even if an adversary is able to sample the same channel, secure keys can still be generated by the two legitimate participants.

With the widespread deployment of wireless networks, this idea was recently used to generate secret keys between two parties over a wireless channel by exploiting channel reciprocity (see Table 7 for an overview). In these protocols, several different sources of information are used; the most common one is the received signal strength indicator (RSSI) because it can be easily measured on a per-packet basis on off-the-shelf hardware. The RSSI method is used in several works [3,11,13,14]. While RSSI provides a convenient channel property, there are several others that were proposed as information sources, e.g., the channel impulse response (CIR) [9,14,20,21] that allows fine-grained measurements but requires specialized hardware, or the carrier phase [16]. Most of these protocols generate entropy by random device movements, although frequency-selective fading experienced through frequency hopping can also be used to generate secret keys in stationary scenarios [17].

While most protocols assume a passive attacker, the authors of [14,20] propose countermeasures against active attacks by employing radio fingerprinting [19]. However, despite the tremendous number of different protocols, there is little research on the attacker's side. A side-channel attack on signal-based key generation schemes by exploiting re-radiation is proposed in [7], which requires precise knowledge of the participants' positions. As this information is often hard to obtain and generally not considered public, the practical applicability

Table 7: Overview of recent physical key generation schemes.

Channel property[a]	RSSI [1,2,3,5,14,17,20]	CIR [9,14,20,21]	Phase [16]
Entropy source	Movement [1,3,5,9,13,14,20,21]	Channel-selective fading [17]	Angle of arrival [2]
Hardware	802.15.4 [1,2,5,13,17]	UWB [3,9]	802.11a [14,20]
Quantization	1-threshold [2,3]	2-thresholds [1,9,14,20]	Dynamic multi-threshold [5,13,16,17,21]
Error correction	Block-based parity [1]	Quantization-dependent [2,3,5,9,14,17]	Error correction codes [20,21]
Attacker model	Passive [1,2,3,5,9,16,17,21]	Active [13,14,20]	—

[a] Some protocols use multiple channel properties.

can be difficult. Edman et al. [8] present a passive attack that puts the practical applicability of the theoretical foundations of signal-based key generation protocols in doubt, i.e., the assumption that the RSSI is uncorrelated at distances greater than $\lambda/2$. According to the authors, a relatively high cross-correlation exists even at larger distances (up to 90 cm), enabling passive attackers to guess 50 % of the key or more by pure eavesdropping. Our contribution consists of a flexible active attack in a realistic scenario, requiring only publicly available information and off-the-shelf hardware, and is entirely independent of physical proximity. In order to demonstrate our attack's practicality, we successfully apply it to the protocol described in [14] without violating any security assumptions. In summary, we believe that the attack described in this work is applicable to all protocols that use RSSI-based quantization of the wireless channel.

7 Conclusion

In this paper, we introduced a novel idea for a man-in-the-middle attack based on *injection opportunities* against signal-based key generation schemes. Using this idea, without assuming any advantage for the adversary, we implemented an attack that exploits imperfect error correction and allows to disrupt a protocol run by deliberately forcing a key disagreement. Following the same idea, we designed a more severe *key recovery* attack that is able to reveal large parts of the secret key generated between two legitimate transmitters. We demonstrated its performance by attacking a concrete protocol in different scenarios using off-the-shelf hardware. Typically, between 40 % and 50 % of the secret key were revealed to the attacker. This success rate decreases with larger distances between the attacker and the legitimate nodes. However, this mitigating factor could easily be improved by using superior hardware or increased sending power. In the worst case, we still recovered around 25 % of the key correctly.

Besides evaluating the attack itself, we analyzed potential countermeasures. We examined statistical mechanisms to detect an attacker and filter spoofs on a

per-packet basis or to reject compromised runs entirely (as oftentimes mentioned in related work). However, without a significant amount of training data the approach was shown to cause a prohibitively large number of false positives. Given these practical problems, simply generating longer keys to impede brute-force attacks could be superior. Yet, such a high price to pay might undermine the advantages of current key generation protocols.

References

1. S. T. Ali, V. Sivaraman, and D. Ostry. Secret key generation rate vs. reconciliation cost using wireless channel characteristics in body area networks. In *Proceedings of the IEEE/IFIP 8th International Conference on Embedded and Ubiquitous Computing (EUC '10)*, pages 644–650. IEEE, Dec. 2010.
2. T. Aono, K. Higuchi, T. Ohira, B. Komiyama, and H. Sasaoka. Wireless secret key generation exploiting reactance-domain scalar response of multipath fading channels. *IEEE Transactions on Antennas and Propagation*, 53(11):3776–3784, 2005.
3. B. Azimi-Sadjadi, A. Kiayias, A. Mercado, and B. Yener. Robust key generation from signal envelopes in wireless networks. In P. Ning, S. De Capitani di Vimercati, and P. F. Syverson, editors, *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07)*, pages 401–410. ACM, Oct. 2007.
4. C. Cachin and U. Maurer. Linking information reconciliation and privacy amplification. *Journal of Cryptology*, 10(2):97–110, 1997.
5. J. Croft, N. Patwari, and S. K. Kasera. Robust uncorrelated bit extraction methodologies for wireless sensors. In T. F. Abdelzaher, T. Voigt, and A. Wolisz, editors, *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '10)*, pages 70–81. ACM, Apr. 2010.
6. Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology—EUROCRYPT '04*, volume 3027 of *LNCS*, pages 523–540. Springer, May 2004.
7. N. Döttling, D. Lazich, J. Müller-Quade, and A. de Almeida. Vulnerabilities of wireless key exchange based on channel reciprocity. In Y. Chung and M. Yung, editors, *Proceedings of the 11th International Workshop on Information Security Applications (WISA '11)*, pages 206–220. Springer, Aug. 2011.
8. M. Edman, A. Kiayias, and B. Yener. On passive inference attacks against physical-layer key extraction. In *Proceedings of the 4th European Workshop on System Security (Eurosec '11)*, pages 8–13. ACM, Apr. 2011.
9. S.-B. Hamida, J.-B. Pierrot, and C. Castelluccia. An adaptive quantization algorithm for secret key generation using radio channel measurements. In K. Al Agha, M. Badra, and G. B. Newby, editors, *Proceedings of the 3rd International Conference on New Technologies, Mobility and Security (NTMS '09)*, pages 1–5, Dec. 2009.
10. R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *Proceedings of the 21st annual ACM Symposium on Theory of Computing (STOC '89)*, pages 12–24. ACM, May 1989.
11. S. Jana, S. N. Premnath, M. Clark, S. K. Kasera, N. Patwari, and S. V. Krishnamurthy. On the effectiveness of secret key extraction from wireless signal strength

in real environments. In K. G. Shin, Y. Zhang, R. Bagrodia, and R. Govindan, editors, *Proceedings of the 15th International Conference on Mobile Computing and Networking (MOBICOM '09)*, pages 321–332. ACM, Sept. 2009.

12. Z. Li, W. Xu, R. Miller, and W. Trappe. Securing wireless systems via lower layer enforcements. In R. Poovendran and A. Juels, editors, *Proceedings of the 5th ACM Workshop on Wireless Security (WiSe '06)*, pages 33–42. ACM, Sept. 2006.
13. H. Liu, J. Yang, Y. Wang, and Y. Chen. Collaborative secret key extraction leveraging received signal strength in mobile wireless networks. In A. G. Greenberg and K. Sohraby, editors, *Proceedings of the 31st IEEE International Conference on Computer Communications (INFOCOM '12)*, pages 927–935. ACM, Mar. 2012.
14. S. Mathur, W. Trappe, N. Mandayam, C. Ye, and A. Reznik. Radio-telepathy: extracting a secret key from an unauthenticated wireless channel. In J. J. Garcia-Luna-Aceves, R. Sivakumar, and P. Steenkiste, editors, *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking (MOBICOM '08)*, pages 128–139. ACM, Sept. 2008.
15. U. Maurer. Protocols for secret key agreement by public discussion based on common information. In E. F. Brickell, editor, *Advances in Cryptology—CRYPTO '92*, volume 740 of *LNCS*, pages 461–470. Springer, Aug. 1993.
16. Q. Wang, H. Su, K. Ren, and K. Kim. Fast and scalable secret key generation exploiting channel phase randomness in wireless networks. In *Proceedings of the 30th IEEE International Conference on Computer Communications (INFOCOM '11)*, pages 1422–1430. IEEE, Apr. 2011.
17. M. Wilhelm, I. Martinovic, and J. B. Schmitt. Secret keys from entangled sensor motes: Implementation and analysis. In *Proceedings of the 3rd ACM Conference on Wireless Network Security (WiSec '10)*, pages 139–144. ACM, Mar. 2010.
18. M. Wilhelm, I. Martinovic, J. B. Schmitt, and V. Lenders. Reactive jamming in wireless networks: How realistic is the threat? In *Proceedings of the 4th ACM Conference on Wireless Network Security (WiSec '11)*, pages 47–52, New York, NY, USA, June 2011. ACM.
19. L. Xiao, L. Greenstein, N. Mandayam, and W. Trappe. Fingerprints in the ether: Using the physical layer for wireless authentication. In *Proceedings of the IEEE International Conference on Communications 2007 (ICC '07)*, pages 4646–4651. IEEE, June 2007.
20. C. Ye, S. Mathur, A. Reznik, Y. Shah, W. Trappe, and N. B. Mandayam. Information-theoretically secret key generation for fading wireless channels. *IEEE Transactions on Information Forensics and Security*, 5(2):240–254, June 2010.
21. J. Zhang, S. K. Kasera, and N. Patwari. Mobility assisted secret key generation using wireless link signatures. In *Proceedings of the 29th IEEE International Conference on Computer Communications (INFOCOM '10)*, pages 1–5. IEEE, Mar. 2010.