Masaryk University
Faculty of Informatics

Edu-hoc: Experimental and educational platform for wireless ad-hoc networking

Master's Thesis

Lukáš Němec

Brno, Fall 2016

Replace this page with a copy of the official signed thesis assignment and the copy of the Statement of an Author.

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Lukáš Němec

Advisor: RNDr. Petr Švenda, Ph.D.

Acknowledgement

TODO thanks

Abstract

TODO abstract

Keywords

keyword1, keyword2, …

Contents

1 Introduction

TODO

2 Problem analysis

TODO

2.1 Creating WSN network

In order to create functioning WSN network, one must ask and answer many questions which define the shape and parameters of such network, these questions include following:

1. What is the purpose of this network?
2. Which devices will be used?
3. Will the network be homogenous or heterogenous (only one device type or multiple ones)?
4. Where will the network be placed?
5. What is the expected lifetime of a network?
6. What will be the power source for devices?
7. How shall we upload new code to devices?
8. How the results will be collected from the nodes?

Some of these are independent decisions, some are quite closely related to another. Now we can elaborate on these questions a bit more and provide possible reasoning for each one. Numbers in parentheses refer to the point in the previous enumeration.

2.1.1 Network purpose (1)

The biggest decision and it basically dictates the options for all the others. General purpose or educational network will be completely different than production network with just one goal only. Basically there is either one sole purpose that is given from the beginning, because the network is designed with this goal in mind; let it be forest

fire monitoring, air pollution measurements, whales migration monitoring, or reporting free parking places; and it is perfectly fit for such goal and not so much fo the others.

On the other hand, a network for development purposes or one with education as one of its purposes cannot be application specific and has to be general as much as it can be. Because there might be many different requirements for the network through its lifetime and is should satisfy all of them. Single purpose network will have devices for the specific job only (maybe even manufactured by required specifications), however, a multipurpose network might require different resources for different applications because one does not know what the requirements might be in the future.

2.1.2 Device selection (2) and (3)

Device selection is very much dependent on the network purpose, but there are some general options and directions one can prefer in certain situations.

First and foremost, is there going to be just one type of device or are we going to use multiple types of devices? The homogenous network will definitely be much easier to create and develop applications for, because all devices share the same properties, all devices behave basically the same, we can use the same set of libraries and radio drivers, everything is almost guaranteed to be compatible.

Heterogeneous network, on the other hand, will be much more complex from the beginning, just because the selection of devices that are capable of communication with each other is an issue of its own. The best option might be using some well-known protocol (e.g. Xbee CITE) which is widely supported across many platforms. The most basic requirement is the same radio frequency, but sometimes one might be forced to write everything from scratch, just because the supported libraries might not be cross platform compatible.

The other aspect to the device selection is its price, therefore for single purpose network, we are going to select the cheapest device that fulfils our needs without much of redundancy because there is no need for it. However is we do not know the purpose in advance, or whether ith might change over time, then we would like devices with are a reasonable mixture of resources compared to its price (there is no

point in a device that has the maximum amount of memory, but lacks in computational power). Or we might even consider devices which are a bit expensive right now, but we expect them to be widely used within few years time because it will be interesting to have applications ready for such devices when they come down to the reasonable price.

2.1.3 Network placement and lifespan (4) (5)

Expected life of the network is closely related to the next issue regarding the power supply to the devices. However, we are also interested in the durability of such devices and possibly resistibility to the weather conditions, water, possibly low or high temperatures or even high pressure. There are definitely going to be extreme differences in devices selected for long term outdoor usage and ones placed inside.

Placement of these devices can even give us more options regarding the power supply because in certain conditions can be used for power harvesting either from solar sources or geothermal ones, the options very much depend on expected surroundings of these devices.

On of the factors that would determine the nature of our network is allowing movement of certain nodes. Devices and options for them are inherently going to be different in the case of the static network compared to moving one. Dynamic network will have much greater needs on redundancy and durability of each individual device and also the design of applications and protocols will have to be much more resilient to the possible events which might occur during the movement (neighbour devices will change, the mortality rate will be naturally much larger, environment around the network will change).

In the static topology we can predict and assume many possible events, even prepare for them in advance. Therefore once a node is not active for certain period of time, we assume it never will be active again (which in a dynamic network could just be a temporal change of topology and we might be able to communicate again after some time). Thus once the static network is deployed (deployment can be randomised) we can actually save the current state and derive everything else from it, in the dynamc network we cannot make any assumptions at all from the initial positions and we have to adapt to the current situation during the whole lifetime of such network.

2.1.4 Power source (6)

Two options for the power supply are available, either supply from a battery or by wire, let the wire be only power cable or cable combined with data (e.g. USB cable). Options are usually determined by the device itself, because devices usually have some requirements regarding the power supply and not all devices can, for example, be powered from five volts which are supplied by USB cable.

The least convenient method is direct power from the outlet, which unfortunately might be sometimes the only option if the device demands an extraordinary amount of power which cannot be supplied any other way or only for short time. Such way of power might, for example, be required for a base station which provides connection to the network from the internet, usually a computer or similar device.

For short network with short life span (e.g. few hours to test some scenario in real conditions or throw off devices for short term use) the ideal solution are battery powered devices, because it keeps the invested amount of resources low. Also, battery power is ideal for out of the grid networks, where any other solution would not be applicable. Here the main goal is to optimise the power usage to the absolute minimum, therefore maximise the network lifetime.

For any other network the optimal solution usually is low power supply via cable, that usually also provides a data connection.

Alternative options for outdoor networks usually include some forms of energy harvesting, this might include solar power, geothermal energy or any other form of power harvesting. This option is usually associated with a presence of battery, that will supply power continually and energy harvesting will be used to recharge this battery. This is the usual case because many of these power options are not continuous and gained an amount of power might vary through the day.

2.1.5 Code and data upload and download (7) and (8)

Code upload and download is usually only a problem for long term network, usually for the development of research purposes, because this goal usually requires frequent changes of source code, thus a convenient way of doing so. In any other kind of network software

is usually uploaded only once and prior to the network deployment, therefore there is no need for a long-term solution for the code update.

When the solution for code upload is required for already placed devices, there are two main options, either updated over the connected cable or some FOTA[1] solution, which would generally be preferred for battery powered networks, since it does not require any additional cables.

In a case of power provided through a low power cable, it is usually the best option to use this cable to provide software uploads and updates since the cable is already present and it is the most reliable way to do so.

The same applies to data transfers, cable connection is the most reliable and straightforward solution, when the cable is not present, such functions have to be provided via the radio.

2.2 Testbed design

For the construction of our own testbed, questions from the previous part have been answered respectively:

1. What is the purpose of this network? There is not one specific application, therefore it is generic type network with research and education purpose.

2. Which devices will be used? Arduino based devices with built-in radio 3.2.

3. Will the network be homogenous or heterogenous (only one device type or multiple ones)? Only one type of device, with the educational network we have to keep thing simple.

4. Where will the network be placed? The network will be placed inside a building, across multiple rooms.

5. What is the expected lifetime of a network? There is no set lifetime goal, only as long as it will be required or replaced by another type of network in the future.

1. firmware over the air

6. What will be the power source for devices? All devices are going to be connected via USB cable, therefore this cable will provide the power.

7. How shall we upload new code to devices? Same as previous, through the USB cable.

8. How the results will be collected from the nodes? Either through the USB cable or using radio communication.

2.2.1 Possible challenges

Since the network is going to be permanent and without battery power, there are not that many challenges to solve. The main issue is the placement of the individual nodes together with all the supporting infrastructure in such manner, that everything would work properly, while the nodes should be distributed evenly across all the covered area.

3 TESTBED deployment

TODO nodes placement with map in appendix, central server placement

3.1 JeeTool

Motivation Few nodes can be managed manually without any major problems, most frequently using directly Arduino IDE [1]. This approach is working flawlessly for up to three devices and it is possible to manage up to 10 devices this way, but with great effort and sacrificing the usability.

Either intentionally or not, there is no doubt that the official Arduino IDE is designed with simplicity in mind, therefore it is usable for a person without a solid background in IT [2]. On the other hand, this simplicity comes at the cost of many features, that might have been useful for more advanced use (e.g. management of multiple devices).

MoteTool inspiration To remedy this problem, jeeTool [3] was created, a deployment tool that uses Arduino Makefile project [4] to manage the network of 25 Arduino devices. This tool was inspired by motetool project [5] which was developed by Dusan Klinec in order to manage the simmilar task, only for the different type of devices? The basic functionality is the same, it allows users to detect all connected devices and upload applications to the selected ones. In addition, jeeTool also supports bidirectional communication with individual devices via a serial port.

Implementation details As has been stated, Arduino makefile is used for the actual application upload. In order to make Makefile call for each of the connected devices, jeeTool creates temporary bash script file, which is used to set the environment for the specific node and execute the upload via Makefile. The temporary script itself is then executed with system call directly from jeeTool. If multiple nodes are selected for upload, jeeTool enables either serial deployment (mostly for debugging purposes) and parallel deployment, where all Makefile

calls are made within discrete threads, thus all of them are executed simultaneously.

Communication over serial port In addition to the upload features, jeeTool also enables communication with devices over a serial port. This is done via jSerialComm library [6] at the moment, in the past RxTx library [7] was used, unfortunately, this library is currently not being developed[1] and decision was made to change the library to more up to date one, instead of using problematic old code.

Individual nodes are matched with files, which are named after the nodes (name of the device in the Linux file system) and jeeTool is capable of writing all inbound communication in such file or read such file and write it over to the serial port. For the inbound communication there is an option of delayed termination of the program, thus jeeTool can be left running for specified amount of time before the ports to devices are closed (e.g. collect results after the experiment). For the outbound communication, jeeTool can either send the whole content of the file at once or there is an option of delay between individual lines of the file (can be used for periodical sending of messages etc.).

3.2 Hardware selection

New network was designed with both the research 4 and educational 5 use in mind. Therefore the selection of hardware for individual nodes was limited by these intended use cases. Since there already was experience with the research type network [8] and its pitfalls related to the complicated use of specialised hardware, we opted for the developer-friendly approach; devices based on Arduino platform 3.2, which are designed with simplicity in mind, as Arduino focus group are not only programmers but also artists and generally people without any advanced IT skills.

JeeNode USB and JeeLink Devices selected for the network are Arduino clones JeeLink nodes [9]. In addition to that, also several JeeNode devices [10] were obtained, these two are similar in many

1. last update is from 2011

ways and can interact with each other, JeeNode USB is only a little bit more versatile in its use options.

Both of these devices are based on Arduino Mini [11] with ATmega328P processor. Wireless communication is enabled by the RF12B chip 3.2 [12] which is permanently mounted to the device itself? Both devices can be connected directly to a computer via USB[2]. The other most important difference between these two types is the fact that JeeLink has a plastic cover, therefore there is no additional connectivity. On the other hand, JeeNode is not covered and there are pins available fo additional sensors or any other devices, that one might need to connect. Last, but certainly not least, JeeLink has additional 16 Mbit of flash memory [9] available, therefore this can be used for convenient storage.

For both of these devices, there is accompanying library JeeLib [13] that contains all necessary functions for controlling the radio and any other function that is different from standard Arduino Mini.

RF12B radio module Simple and quite cheap radio module from HopeRF [12] is present on all JeeMote and JeeLink devices. The version sold in Europe is using 868 MHz frequency for the radio[3]. As many other common radio modules, this radio is half-duplex, therefore it is able either to receive or to transmit messages, but cannot do both at one time.

The radio module is equipped with omnidirectional whip antenna ($\approx$ 8 cm long wire) by default. It is possible to modify the properties of such antenna by shaping it in various ways, e.g. straight wire compared to coiled one. There might also be many other versions and shapes of antennae, which might improve radio properties, but this would be subject to whole another research in quite a different field.

2. the USB port type is different
3. other version can use 433 MHz or 915 MHz

4 Research use

One of the intended uses for the created testbed was its use in research which can be of various nature. In this particular case, we would like to make a case study for key generation mechanism without any pre-shared information between individual nodes.

In today's application this would be quite a common problem since a number of devices equipped with some kind of radio module are constantly growing, let it be wifi, Bluetooth, or any other frequency and specification. However, we cannot expect for these devices to have any shared data nor capability to exchange them over an already secure channel since the secure channel is what we are trying to accomplish.

What we would like to show, is a scheme for key derivation without any shared secret, only using properties of radio channel shared between these two devices and the nature of radio wave propagation through such shared channel.

4.1 Keys from radio signal

Due to a nature of a wireless channel which is not perfect, we can observe fluctuations and disturbances in the wave propagation. It might seem difficult at first to measure these since by no means we are able to observe all such events. Despite that there is a way, almost every radio module is able to measure RSSI[1], which can be described as an amount of power in the received signal. There are many arbitrary scales and absolute values do not match on devices from different manufacturers since RSSI is relative index [14].

However, our algorithm can work regardless of the differences in devices, even two devices with the same configuration can produce different absolute values due to antenna position and overall device manufacturing differences, since some of these devices are partially assembled by hand [2].

1. received signal strength indication
2. e.g. JeeNode motes come as a ready to assemble kit

4.1.1 Quantization principle (bits from signal strength)

Quantization enables to extract bits from individual values of signal strength. There are many different approaches to this problem [15] two main approaches here are: lossless quantization [16] and contrary to that we have lossy quantization [17].

Main difference between these is number of generated bits per original signal strength measurements, while lossless quantization produces bit value from every measurement of signal strength, which is useful for high-performance demands, but it requires guaranteed variance in the radio channel (e.g. the nodes are constantly moving, or the environment is changing) during the key establishment phase. Otherwise, the resulting keys could possibly be very weak.

Lossy quantization, on the other hand, does not have guaranteed output length per number of measured values, which can lead to a very limited length of output. However, this kind of quantization is expected to have better results in static environments because its nature is to drop such bits, that fail to differ from others.

Also, there are few different measurements of the channel one can use to produce bits. One would be RSSI measurements on received messages, another measure

Since our network is static and without any moving nodes, we implemented lossy quantizer algorithm designed by Mathur et.al. [17], which showed promising results for the of the shelf wireless devices similar to ours, and also contained experimental results from several different scenarios, where some of these were comparable to our conditions.

4.1.2 RSSI Quantization

Quantization princeple designed by Mathur et.al. [17] works as follows:

1. both nodes send n messages to each other in alternating pattern, both nodes send counter value inside these messages, which is used to synchronise messages on individual nodes. For every one of these messages, signal strength is measured upon reception.

2. when n messages have been successfully exchanged, both nodes can proceed to the computational part.

3. both nodes calculate mean m and standard deviation sd for signal strength values of all received messages.

4. both nodes calculate q^+ and q^- values, which are upper and lower quantizer bin boundaries, as follows:

$$q^+ = m + \alpha \cdot sd$$

$$q^- = m - \alpha \cdot sd$$

5. every signal strength measurement is then processed and it is rejected, if it lies within q^+ and q^- boundaries, values above this range are assigned a bit value of one, values below are assigned a bit value of 0.

6. nodes then synchronise their measurement by exchanging counter values associated with those messages, where signal strength measurements were assigned either one or zero bit values.

7. those counter values that match on both nodes are expected to be excursions in the same direction and are used in the final outcome.

   TODO add some nice picture that would show this

4.2 Improvement strategies

4.2.1 Cooperative jamming (can it improve our situation?)

4.2.2 Other means of disturbances

4.2.3 other (shield)

TODO add shielding description with drawing

4.3 Experiment setting

An application was created, ir order to collect data from real world network. Such data then would be processed by quantization algorithm 4.1.2 and produced bit sequences were analysed regarding the entropy and usability for cryptographic keys 4.4.

This application uses speacially modified JeeNode motes 3.2, because analog RSSI values are not available through provided connections to radio module. Fortunately, analog RSSI can be obtained through short jumper wire from radio module to the Arduino analog pin

In the experiment itself, two motes exchange messages with counter value (counter is updated only on one of these motes) and both motes save measured RSSI values for each message. The counter value is used to guarantee delivery of these messages, mote A sends message with value n and mote B upon reception responds with the same value n. Mote A can increment counter value only when respond message with current counter value is received. Othervise the same counter value n has to be send again and both motes will keep the RSSI measurement only from the most recent message with counter n.

Application is programmed in such way, that 1000 counter values are exchanged, then they are send out via the serial port and application is terminated. Intervals between individual messages should be as short as possible, so that the random disturbances in the radio channel would be captured on both nodes. Even though RSSI measurement are definitely not symetrical, using this approach we can expect them to be highly correlated

The limit of 1000 messages was set due to the memmory constrains of individual devices. Since the bottleneck of our experiment was in the communication via the serial port, we opted to exchange all messages and gather data first and only after that to transfer them out to the server. The application was then uploaded to the nodes again and run from fresh state several times as the experiment settings required. In the optimal conditions, the expected time for single run of application is about 30 seconds; runtime over two minutes is usually a result of some kind of error, either in application itself, placement of devices, or disturbances in the radio channel.

In addition to regular nodes, up to two sniffer motes were active at the time of application runtime. These were uploaded with modified version of application, so that they would only receive messages, but would not send any messages to other devices. We have done several trials on placement of these sniffer motes, in order to see any impact of position in the network on the results.

4.4 Performance Evaluation (results from experiments)

We have made several runs of our application with different settings. Nodes A and B were always on the same places (sniffer possitions were altered), but the conditions of radio channel itself were different. First, as a base level, we had run the experiment in conditions without any disturbances. After we realised, that the plain static network does not produce any usable data, we tryed to introduce other techniques 4.2.1 4.2.2 that would add disturbances to the radio channel, therefore changes in the measured RSSI values.

To analyse the gathered data we used set of R scripts and all data postprocessing was done outside of the motes, while in the real settings, motes would have to do all of the computations on their own. In orde to evaluate produced bit sequences, we used min entropy 4.4 and frequency test 4.4, both on eight and four bit chunks of data. The goal was to obtain several measurements of entropy in the data and then use the most conservative of them to estimate contained randomness. Since we had collected only small amounts of data for individual sources, we decided not to use any of statistical test batteries , because our data either did not match the required amount for input or the results would not be statistically signifcant for such small sample.

Min entropy Min entropy [18] is considered the most conservative estimate of unpredictability in information theory. To make an min entropy estimate of IID[3] source we can use simple calculation as follows, given the dataset with N samples:

3. Independend and identically distributed

1. Find the most frequent value and let C_{MAX} equal number of occurences of this value in dataset.
2. $pmax = \frac{C_{MAX}}{N}$
3. $C_{BOUND} = C_{MAX} + 2.3 \cdot \sqrt{N \cdot pmax \cdot (1 - pmax)}$
4. $H = -\log_2(\frac{C_{BOUND}}{N})$
5. W is the number of bits in each sample of dataset.
6. Min entropy is then equal to $\min(W, H)$

It is important to note, that this calculation of min entropy estimate is only meaningful for IID sources [18], for non-IID sources we have to use other means. Testing sources with dependencies with this min entropy calculation may result in overestimates and generally output without any meanigful use.

There are many test for estimating min entropy contained within output of non IID sources, most notably collision test, partial collection test or the compression test [18]. To us the most meaningfull one was the frequency test[18], because of its ability to work on small amounts of data, larger amount of data only increase the accuracy, but the test can provide meaningful estimate on any amount of data provided.

Frequency test On first sight, the calculation is simmilar to the original min entropy 4.4. The calculation is following, using 0.005 as confidence level α:

1. Count occurences of each value in the dataset as $count_i$ for all $i = 1 \ldots n$ where n means all possible outputs.
2. Calculate ϵ from confidence level α and number of observations in dataset.
$$\epsilon = \sqrt{\frac{\log(\frac{1}{1-\alpha})}{2 \cdot N}}$$
3. $pmax$ is the probability of most likely observation value.
$$pmax = max_{i=1..n}(\frac{count_i}{N})$$

4. Resulting estimate of min entropy is

$$min - entropy = -\log_2(pmax + \epsilon)$$

Using these two approaches we can get the most conservative estimate of entropy contained within our gathered data.

Results from all experiments are sumarised in the table 4.1 and then results for each individual source are discussed in following subsections.

4.4.1 Plain source

As we can see from the table 4.1, there were no bits produced from this experiment. The reason behind is really simple, there were no disturbances in the radio channel, therefore all measurements of RSSI resulted in the same value. Therefore the quantization algorithm 4.1.2 computes sd equal to 0, thus fails to establish bounds q^+ and q^- and there are no bits produced on the output.

Since there are no bits produced, there is no point in measuring the enthropy of the output. This result can be considered as the pitfall of static networks, without any moving parts or changes in the environment, there is not enough entropy in the radio channel as such, that could be used to extract entropy, because there is none. Of course, the greater the area such network covers, disturbances are more probable, but we have to be avare of this problem and be prepared to indentify such enviroment. The main reason being any bits generated in such environment should not be used [15].

4.4.2 Jammer

Friendly jamming can be used as a possible remedy to the problem of static network 4.2.1, creating artificial noise in the network and thus possibly adding some entropy (given that the jamming is not predictable).

As the results in the table 4.1 show, this approach was not succesfull, mostly because of limitations of used radio modules 3.2. While this approach would work with more advanced radios that are capable of substractions on the signal level and allow for more devices to

Sources:	Plain	Jammer	Moving people	Moving shielding
Total runs:	40	10	424	26
Usable runs I. ($sd \neq 0$)	0	0	256	25
Avg. bits produced I.	0	0	189.41	238.48
Avg. errors produced	-	-	0.41	0.00
Avg. errors on sniffer	-	-	51.00	59.88
Usable runs II. (Both 1s and 0s in output)	-	-	97	10
Avg. bits produced II.	-	-	409.06	470.00
Min entropy (8bit chunks)	-	-	25.96	34.21
Min entropy (4bit chunks)	-	-	50.64	68.39
Frequency test (8bit chunks)	-	-	42.45	52.62
Frequency test (4bit chunks)	-	-	72.40	96.01

Table 4.1: table of sources evaluation

broadcast at one time. RF12 radio module and its acompaniing library however does not allow this, mostly because of really simple collision avoidance alghorith. The approach is based on simple principle, not to send messages when there is any other signal present in the wireless medium.

Therefore our primitive jammer, although jammer itself was able to broadcast anytime, failed. The legitimate nodes A and B could not receive nor broadcast their own messages, because of the collision avoidance. Thus the only effect of jammer was prolonget time needed to exchange 1000 messages between devices.

To counter this behaviour, we tried to limit the transmission power used by sniffer, even using it without antena. Once the signal was limited to the required level, that nodes A and B would not be interupted by it, then we observed same results as if the jammer was not present at all.

Although there might be other option how to make this approach work, because we limited ourselves to using stock radio modules and available software for radio management, we were not able to produce any bits using this method. Thus we leave out this option for either more advanced users (who might be albe to control the radio on the most elemental level, therefore defining own collision avoidance rules etc.) or users with more specialised hardware (e.g. software defined radio modules).

4.4.3 Moving people

This option was not one of the original intended sources, but during the experiment with plain source we discovered, that people walking through the monitored area caused much wanted iregularities in radio signal. This option would be a viable for busy places, like already tried cafeteria [17] or town square. Thans to the netwrok placement we utilised our clasroom and made these measurements during one of the lectures. The data clearly show some patterns, mostly we can observe timeframes which produce bit sequencies with high enthropy and also timeframes where no bits were generated.

These differences can by explained by a simple fact, people do not move in a random fashion. And that is exactly what we can observe in our data, during runs, that produced data people were moving, but most of the time they remained stationary, therefore with amost no effect on the network. Although movement of people is not random, it is can be considered not predictable, therefore it might be sufficient source of randomness for our purposes.

The results show promising values and show us, that there is an potential and more research should be done in this direction. We can see clear trend in data, once we filter out unusable runs (either because no bits were produced, or because produced bit sequence contained only zeroes), we can see, that about one quarter of runs is suitable for bit extraction. Of course these runs are not evenly distributed in our sample, they are rather clustered. On average we can generate about 400 bits from one succesfull run. As we can see from the results of min entropy 4.4 and frequency test 4.4, the most conservative estimate is about 26 bits of entropy contained within these data. Thus on average we would like to utilise five consecutive runs, in order to generate usable key.

4.4.4 Moving shielding

In last experiment, motivated by the promissing data from previous one, we tryed to introduce disturbances in the radio channel without the presence of any persons 4.2.3. In the table we can see, that this approach achieved almost the same results as the previous one 4.4.3.

Unfortunatelly, due to an error, we did not manage to collect any larger amount of data, therefore there is not much we can conclude from these results. However, the current results are promising, averages of generated bits and entrophy estimates are slighly better than in other experiments. These values might decrease with more experiments, but given the fact, that these results were produced on network without any interference from outside, this area is definitely worth looking into.

The big question regarding this source is the predicatability of such data, because as we can see on graph of sample run there are clearly visible excursions that do repeat in somewhat regular fashion.

4.4.5 Speed (bits of key per time)

One run (exchange of 1000 messages) takes approximately 30 seconds to complete. This is under condition, that these two nodes are only one to use the wireless medium, otherwise the required time would be different.

Anyway, approximately 30 seconds to exchange values and then some minor time for calculations and exchange of two messages with identified excursions (which are negligible regarding the time). From these 1000 values we can obtain about 25 bits of enthropy on average 4.4.3, therefore we can gather about 50 bits per minute.

This is comparable to the original result 1 bit/sec of Mathur et.al. [17], however we were capable of echanging way less messages per minute than the original proposal (our rate 1000 msgs/30 sec is not comparable to 4000 msgs/sec achieved by Mathur et.al.), probally due to the nature of used devices 3.2.

4.4.6 Possible errors

Question about reliability surely comes to mind. We have no direct guarantee, values and measurement on individual nodes are clearly not symetrical, we only expect them to be highly correlated.

As we can see on the boxplots 4.1 4.2, the errors between nodes A and B are indeed scarce, therefore there is reasonable probbability, that the nodes A and B will be able to establish common key, using the obtained bit sequence from quantisation algorith 4.1.2.

The other important part is how many bits were compromised to the sniffer node and potential attacker. As we can see, on average there is an hamming distance of 50 bits between output bit sequence for nodes A and B and bit sequence derived at sniffer node. Of course there are excursions in both directions, therefore we cannot entirely guarantee that the derived bit sequence is secret.

On the other hand, our bit sequence still needs some post processing in order to extract the randomness 4.5, the most simple one would be hash function. Therefore, if we use correct function, we can expect, that one different bit would cause change in half of the bits in the resulting key thanks to the avalanche effect [19].

We can clearly see, that guessing is also not an option, although individual bit positions in sequence are known, there is no way to estimate whether the identified excursion is over or below the treshold. Guessing which bits are not correct is also not an option, while guessing just one bit change would mean 400 trials on average, two bits would mean about 400^2 trials. Complexity for this case grows

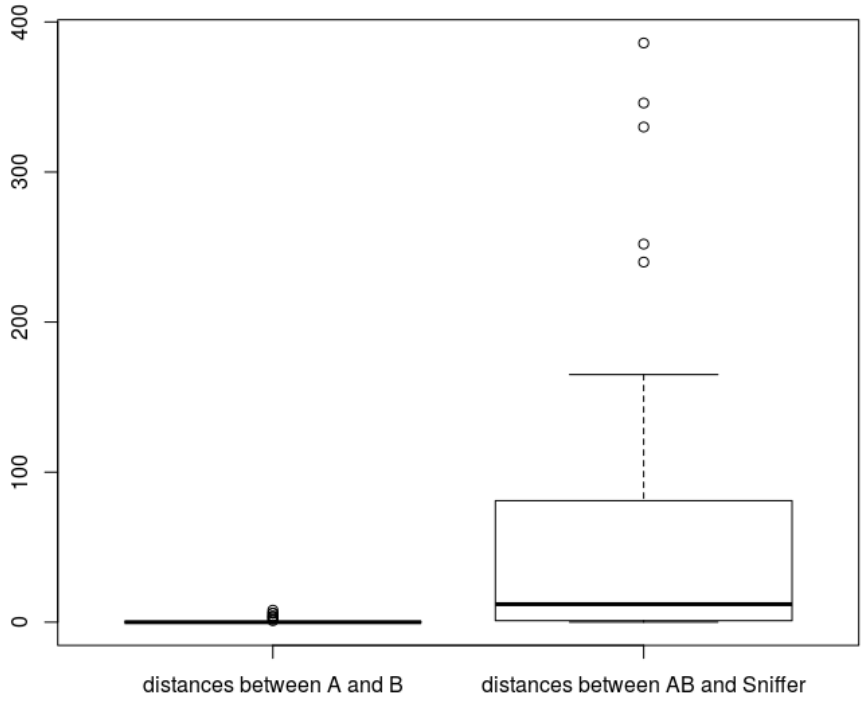

Figure 4.1: Boxplot of Hamming distances between generated bit sequences from nodes A, B, and sniffer node during movement of people 4.4.3

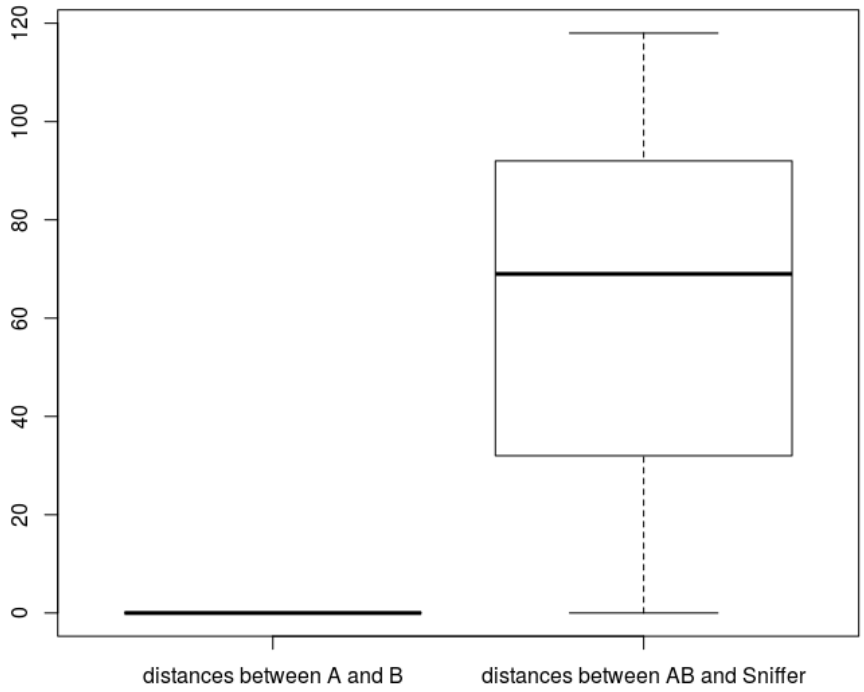

Figure 4.2: Boxplot of Hamming distances between generated bit sequences from nodes A, B, and sniffer node during movement of shielding 4.4.4

exponentialy and the sniffer gets 50 bits wrong on average without any reasonable clue which bits are correct and which are not.

4.5 Discussion and use cases

5 Education use

5.1 motivation for educational WSN network

The current state of the art WSN devices usually uses specialised hardware and software in order to achieve the best performance available. This, unfortunately, is not the ideal prerequisite for an easy to learn matter. In fact, most of WSN devices have rather complicated setup and are quite challenging for novices [20].

Because of such discouragement, it is difficult to teach how to work with WSN's; few hours (at least) are usually required to explain the basics, which is reasonable for a research project or something similar, but for a class exercise, this would turn out to be not the most effective use of time, if it would be achievable at all. And we have not yet mentioned more advanced topics in this area, such as common techniques for encryption or message authentication.

Issue of this nature can be solved in various ways, in the case of Edu-hoc, we decided to sacrifice performance 3.2; which is not that much important for a network with an educational purpose. On the other hand, using hardware that is really easy to comprehend and use is of a great benefit here. Also having less powerful, but relatively cheap devices (in the range of $30 rather than $100 or more) gives the opportunity to lend each the students one of the devices, so that they can try the basics on their own, and also use this device for interactions with the network.

5.2 Scenario approach (attack and repair)

In order to make learning more enjoyable experience and also to add some challenging part to the learning process, we decided to make Edu-hoc scenario based; each scenario being composed of two distinct parts: first part in the role of an attacker (both enjoyable and educational) and the second part as a code reviewer or developer (primarily educational).

attacker part of a scenario Students are presented with network application which has known, or easily detectable, vulnerability. Task is to take advantage of such vulnerability and exploit it using own nodes and carefully executed interactions with the network. How successful these efforts were can be easily evaluated by percentage difference from the expected traffic of the network or by presenting learned secret in the submission of the solution. Exact methods of evaluation are described in 5.3.

reviewer part of a scenario An important part of the educational process is not only to find mistakes, but also to be able to correct them. The task in this part of the scenario is then to take the current source code of an application, which has been deployed on the network during the attacker part 5.2, correctly identify which mistakes were made and propose changes in the code, which would make the application secure against such kind of an attack. Exact evaluation methods are again described in 5.3.

5.2.1 1st scenario - Eavesdropping

The first scenario can be considered really simple and it is by design. Since this scenario is the first encounter with Edu-hoc, we opted for passive attacker approach, without any actual interaction with the network, just listening for any traffic.

Eavesdropping is also the first thing one would do if one would be about to attack some network because it does not compromise the presence of an attacker and intercepted messages may provide many useful pieces of information. This is another important reason why this task was selected as a first one, not only it is rather easy to do, but it provides vital information for the rest of the exercises (e.g. which nodes are present in the network, which frequency is used and what are the settings of the radio)

scenario setting Each node in the network sends messages, after each message, there is a delay of 1000 ms, therefore any receiving node has time to process the transmitted messages. All messages are sent as a broadcast and nodes within the network do not receive nor

process them since it is not needed for successful scenario run. If this would be the case, transmission rates would have been updated accordingly and the scenario would be much more likely to fail on its own, because of single node malfunction, while with current settings, the network can operate without any problems even is several of the nodes would fail.

attack principle The attack is very simple, attacker node only has to listen for any traffic present in the network. For best results application has to be able to process as many messages as possible, however any performance changes will affect the end result only slightly (final percentage will be better, but the bottleneck is in the processor of the radio module, so better optimised application will increase the end result only by approximately 10-20%)

application and securing it The main issue with the application is unencrypted broadcast. This can be fixed quite easily, just by the addition of simple encryption and for simplicity using common shared key. We have to be aware of the fact, that attacker might be able to collect some of the nodes and thus learn such key and compromise it and there are techniques which deal with this problem However, in our scenario, this is sufficient solution because it is adequate to measure against eavesdropping.

5.2.2 2nd scenario - Black hole attack

This scenario presents more advanced concepts, dynamic routing in particular. The task is to attack the routing algorithm and diverse all traffic so that the central node does not receive any messages. This scenario requires active attacker, thus some basic interaction with the network.

scenario setting The scenario is divided into two parts, first parts are shorter and are used to establish routes for this particular run of scenario. Without any attack or interference outside from the network, these routes should be always the same.

In the following part of scenario each node periodically sends messages to the parent node, therefore all messages are routed to the central node. This node then counts all received messages and the final outcome is the number of expected messages compared to the number of actual messages received.

attack principle This attack requires some prior knowledge before execution which can be obtained simply by using eavesdropping technique from the previous exercise. This way an attacker can learn how the routing algorithm works (nodes broadcast their current distance to the central node, where central node has distance of 0, all nodes that hear distance announcement message then update their distance and broadcast this updated distance)

With this knowledge attacker than try to inject message with the same content as the central node does, but earlier than the central node is scheduled to do so. After this, the message from a central node will not improve the position of other nodes and they will not update their distance. Another possible approach is to find an intentional weakness in the algorithm implementation (application allows negative distance values).

The only thing that remains is to make sure to use nonexisting ID within the network and after the routing tree is established, attacker node can simply disappear, because it is no longer needed.

application and securing it In order to fix the application one has to fix two issues; check for negative counter values (which is the easy part and requires usually addition of one condition within the code); and what is more important, implement some kind of authentication mechanism for messages from the central node, or otherwise make sure that the attacker cannot diverse all the traffic only by being a bit faster than the central node.

Many options are available, the easiest one being preshared random authentication tag, which would be added to the messages. This is really simple countermeasure, but it guarantees that the attacker can't send a valid message before the central node sends it.

5.2.3 3rd scenario - Sinkhole attack

Although very similar to the second scenario, this poses far greater challenge than the previous one. The goal is not only to divert the traffic but also modify it on the fly and send it to the original recipient.

The greatest issue here is related to the performance, since all messages from the entire network will be routed through single attacker node, therefore most of the messages are very likely to be dropped somewhere in the process, because single node is not capable of receiving and processing messages from every other node in the network, if they do not transmit on very slow rate.

Task for this scenario is to deliver a modified message to the central node before the original message is delivered; messages with the same identifier but different content are discarded and only the one which was received the earliest is kept at the central node.

The reason why two very similar scenarios exist, while it could have been only this one, is to present scenarios in a manner of slowly increasing difficulty. If we were to skip the second scenario 5.2.2, the task might be too complicated to some, while this way, required task in the third scenario 5.2.3 is only a minor extension to the second one, however, the achieved result is a far greater impact.

scenario setting Runtime of this scenario is the same as for the previous one 5.2.2; initial phase of route establishment and after that messages are routed from nodes to the central node. The expected outcome is to divert ad modify as many messages as possible.

attack principle The first part of the attack can be executed in the same fashion as the previous one 5.2.2 although it is not the ideal way. As the goal is not to prevent all messages from reaching its destination, but to deliver as many modified messages as possible. Therefore it is not a good idea to modify all of the routes in the network, but only as much as the attacker node is capable of handling (i.e. only third of the nodes in the network). This can be done by sending the distance message only to selected nodes instead of broadcasting it.

Another part of the attack is rather simple, attacker node has to receive all messages which are addressed to it, modify these messages as it is required and send these messages to the central node. Modifi-

cation usually requires simple text substitution inside the body of the message or task very similar to this one.

application and securing it Again as many times with this scenario, the fix is partly the same as per second scenario 5.2.2, which deals with the attack on the routing; either preventing it or detecting it.

The another part of this fix is primarily an authentication problem, therefore any kind of MAC will be able to secure this application. Therefore the problem of shared key arises again, but as before in first scenario 5.2.1 one master key will be able to do the job, if we assume, that the attacker does not have access to individual devices, only to radio communication between them.

The point of the exercise is to deal the problem at hand without complicating it with anything else. Of course, we could add any advanced key distribution schemes to the application in order to add countermeasures against master key compromise, but that would make completely different application without any real added value to the original educational purpose.

5.2.4 4th scenario - Jamming

Evey once a while attacker comes around a network, application or protocol which is designed with security in mind, meaning there are no backdoors or weak spots in the design or implementation (really idealistic scenario, however, it is very likely that the attacker does not resource to find or use existing vulnerabilities, thus for such attacker there is no real way how to exploit such application).

Nevertheless, it is very likely that such attacker would like to cause any kind of damage he is capable of doing. Most likely to make the application, protocol, or network unavailable; most likely by DoS attack or something similar. For wireless communication it is quite easy to do, since the wireless medium, air to be specific, can be used by only one device at a time. If an attacker is able to utilise all ate available slot for transmissions, then the legitimate user would not be able to transmit anything. Such technique is called jamming and it is quite easy to deploy.

scenario setting The setting of this scenario is really simple, all nodes send a message via the fixed routing tree to the central node, who the counts the number of delivered messages. There is no phase of route establishment, only the message delivery phase. The assignment is to prevent delivery of as many messages as possible. Since there is no option to modify the routes etc. the only reasonable way is jamming.

attack principle As has been stated before, jamming is the expected way of solving this scenario, although there might be many creative solutions created. With jamming one can either bypass the media access protocol and send data continually, thus preventing everyone else from sending any messages, or the another viable option is to send many valid messages to the central node, thus overloading it and forcing this node to drop many incoming messages, including the valid ones.

application and securing it Since this attack does not exploit any weakness, only the actual property of wireless communication and network design, there is not much one can do to prevent such attack, while only modifying the application source code. However, there are ways of dealing with such kind of an attack. Countermeasures are usually based on multiplication of the devices and decentralisation of the whole network. Therefore give two or even more nodes the ability to become the central node, use multiple frequencies etc.

5.3 Evaluation principle

Measuring the success can be done in various different ways, Eduhoc utilises few different methods to decide, whether the attack was successful and how big was the impact of an attack.

Evaluation for each scenario is a little bit different because there are different objectives. The task of first scenario 5.2.1 is to capture as many messages as possible, therefore the evaluation is done as a simple comparison between the set of captured messages and set of pre-computed messages that are known to be transmitted in the time period of scenario run. Comparing these two inputs line by line we

get intersect of these two sets, that contains all successfully captured messages and is stripped of all the duplicates and possible fraud messages. A number of messages in the resulting is then compared to the expected number of messages in order to receive the final percentage evaluation, the higher the number, the better the result.

The second and the fourth scenario 5.2.2 5.2.4 have the same goal, only differ in the scenario settings, which can be omitted for the evaluation purposes. The common goal is to prevent as many messages as possible from delivery to the central node. The way it is done is rather simple, the central node counts every delivered message and the resulting number is compared with the expected amount of messages.

Unfortunately, this design of scenario lacks in the evaluation one important thing, the credit for the attack cannot be given to any participant, because we have no identifier involved. Messages only vanish, but we do not know, whether it was by joint work of multiple attackers, one really successful one or by a fault in the network itself. Therefore there is not much of use for such scenario in the courses, and because of this, modified versions of scenarios have been used 5.5.

The third scenario 5.2.3 however utilises message modification, which can be used for identification purposes. Therefore messages on the central node are not only counted, but alco compared to the expected ones and differences are noted[1].

Since students are assigned a unique identifier to modify the messages with, we can easily distinguish between participants and evaluate how successful they were in this task.

The last option which can be utilised in evaluation is a submission of secret, which was received from interaction with the network, node either responds to received message with such secret or the secret has to be extracted from the communication in the network. Such secret can be a short word, number or something of similar nature and again such secret can be unique for each participant, which makes te evaluation an easy task.

1. the amount of computation on this level would limit the usability of the node, thus the node only sends received messages over the serial port to the server, where the actual evaluation is done

5.4 Web interface and auto run

Edu-hoc was designed as a long-term exercise, thus single scenarios are expected to run for days, possibly week or longer. However, for some it would be extremely impractical to repeat only once during the whole time because we would like to present fresh scenario to everyone involved, and also we would like to reset the scenario once a while. This will ensure correct initial settings and if we set the intervals right, then everyone will be able to test out several solutions within a reasonably short timeframe.

To achieve such kind of behaviour, we can utilise various approaches, the easiest one being set of bash scripts and setting the cron service accordingly. The other option might be modified JeeTool, which would be running continuously and in set intervals would trigger the corresponding action.

The last option is the most complicated one but offers the most

5.5 PA197 use and results

6 Future and related work

6.1 RSSI

6.2 Edu-hoc

7 Summary

Bibliography

[1] *Arduino*. https://www.arduino.cc/. 2016.

[2] *Arduino Introduction*. https://www.arduino.cc/en/guide/introduction. 2016.

[3] L. Němec. *JeeTool*. https://github.com/crocs-muni/JeeTool. 2016.

[4] *Arduino-Makefile*. https://github.com/weareleka/Arduino-Makefile. 2016.

[5] D. Klinec. *WSNmotelist*. https://github.com/ph4r05/WSNmotelist. 2014.

[6] W. Hedgecock. *jSerialComm library*. http://fazecast.github.io/jSerialComm/. 2016.

[7] *RXTX library*. http://rxtx.qbang.org. 2011.

[8] Vashek Matyáš et al. "WSNProtectLayer: Security Middleware for Wireless Sensor Networks". In: *Securing Cyber-Physical Systems*. CRC Press, Sept. 2015, pp. 119–162. ISBN: 978-1-4987-0098-6. DOI: doi:10.1201/b19311-6. URL: http://dx.doi.org/10.1201/b19311-6.

[9] JeeLabs. *JeeLink product page*. http://jeelabs.net/projects/hardware/wiki/JeeLink. 2016.

[10] JeeLabs. *JeeNode product page*. http://jeelabs.net/projects/hardware/wiki/JeeNode_USB. 2016.

[11] *Arduino Mini*. https://www.arduino.cc/en/Main/ArduinoBoardMini. 2016.

[12] HOPE RF. *RF12B Datasheet*. https://www.sparkfun.com/datasheets/Wireless/General/RF12B-IC.pdf. 2016.

[13] JeeLabs. *JeeLib library*. http://jeelabs.net/projects/jeelib/wiki. 2016.

[14] *Understanding RSSI*. http://www.metageek.com/training/resources/understanding-rssi.html. 2016.

[15] Suman Jana et al. "On the Effectiveness of Secret Key Extraction from Wireless Signal Strength in Real Environments". In: *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking*. MobiCom '09. Beijing, China: ACM, 2009, pp. 321–332. ISBN: 978-1-60558-702-8. DOI: 10.1145/1614320.1614356. URL: http://doi.acm.org/10.1145/1614320.1614356.

[16] B. Azimi-Sadjadi et al. "Secret Communication over Fading Channels". In: *Securing Wireless Communications at the Physical Layer*. Ed. by Ruoheng Liu and Wade Trappe. Boston, MA: Springer US, 2010, pp. 281–309. ISBN: 978-1-4419-1385-2. DOI: 10.1007/978-1-4419-1385-2_12. URL: http://dx.doi.org/10.1007/978-1-4419-1385-2_12.

[17] Suhas Mathur et al. "Radio-telepathy: Extracting a Secret Key from an Unauthenticated Wireless Channel". In: *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*. MobiCom '08. San Francisco, California, USA: ACM, 2008, pp. 128–139. ISBN: 978-1-60558-096-8. DOI: 10.1145/1409944.1409960. URL: http://doi.acm.org/10.1145/1409944.1409960.

[18] Elaine Barker and John Kelsey. "Recommendation for the entropy sources used for random bit generation". In: *Draft NIST Special Publication* (2012).

[19] A. F. Webster and S. E. Tavares. "On the Design of S-Boxes". In: *Advances in Cryptology — CRYPTO '85 Proceedings*. Ed. by Hugh C. Williams. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 523–534. ISBN: 978-3-540-39799-1. DOI: 10.1007/3-540-39799-X_41. URL: http://dx.doi.org/10.1007/3-540-39799-X_41.

[20] Fei Hu and Xiaojun Cao. *Wireless sensor networks: principles and practice*. CRC Press, 2010, pp. 225–226.

A Testbed map

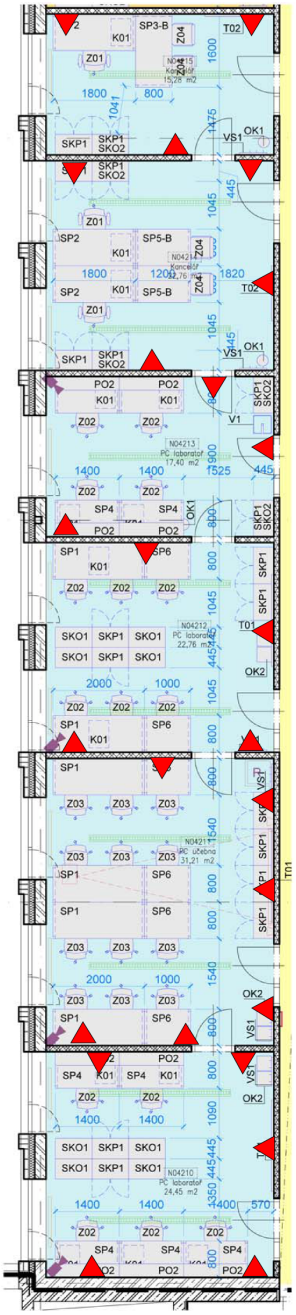

Figure A.1: Map of testbed with placement of individual nodes

B RSSI graphs

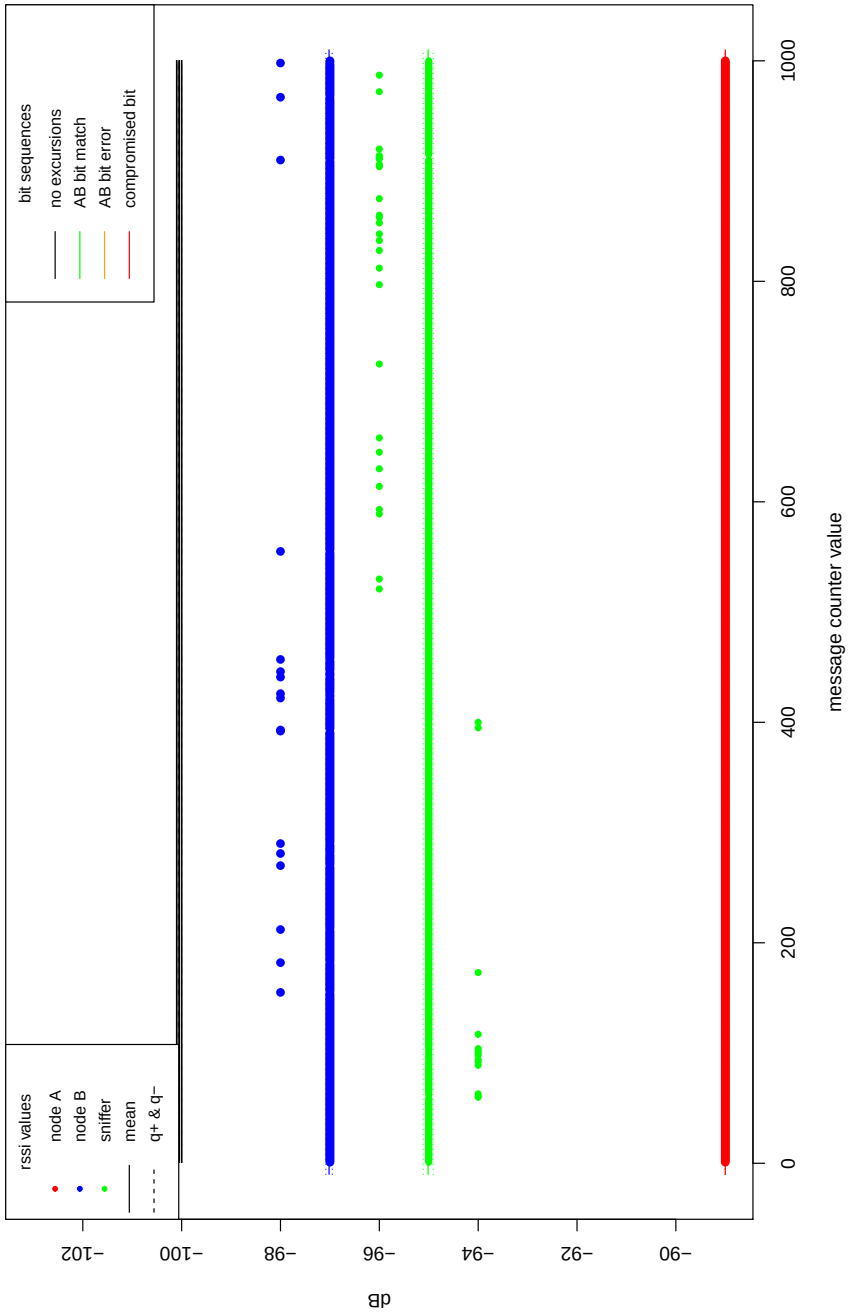

Figure B.1: sd = 0

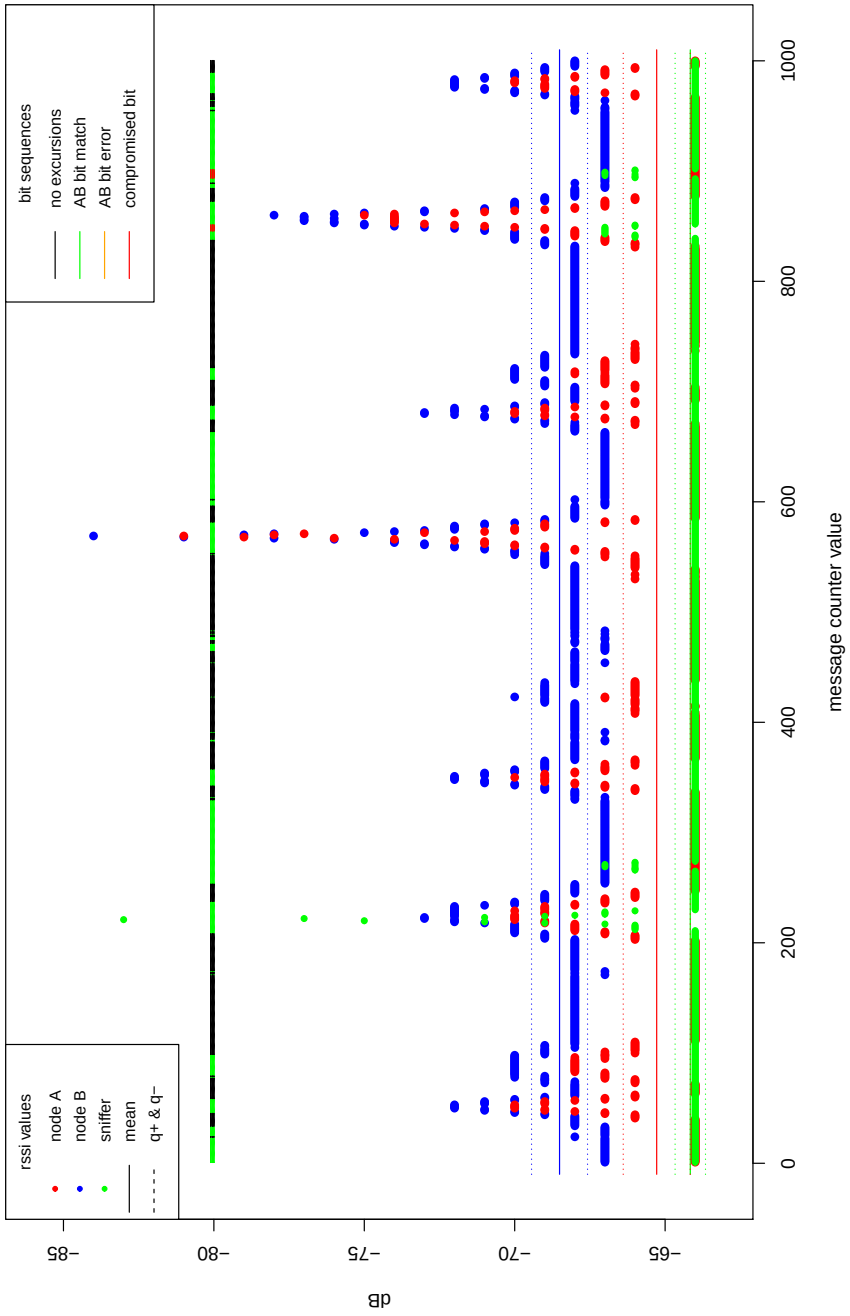

Figure B.2: spikes

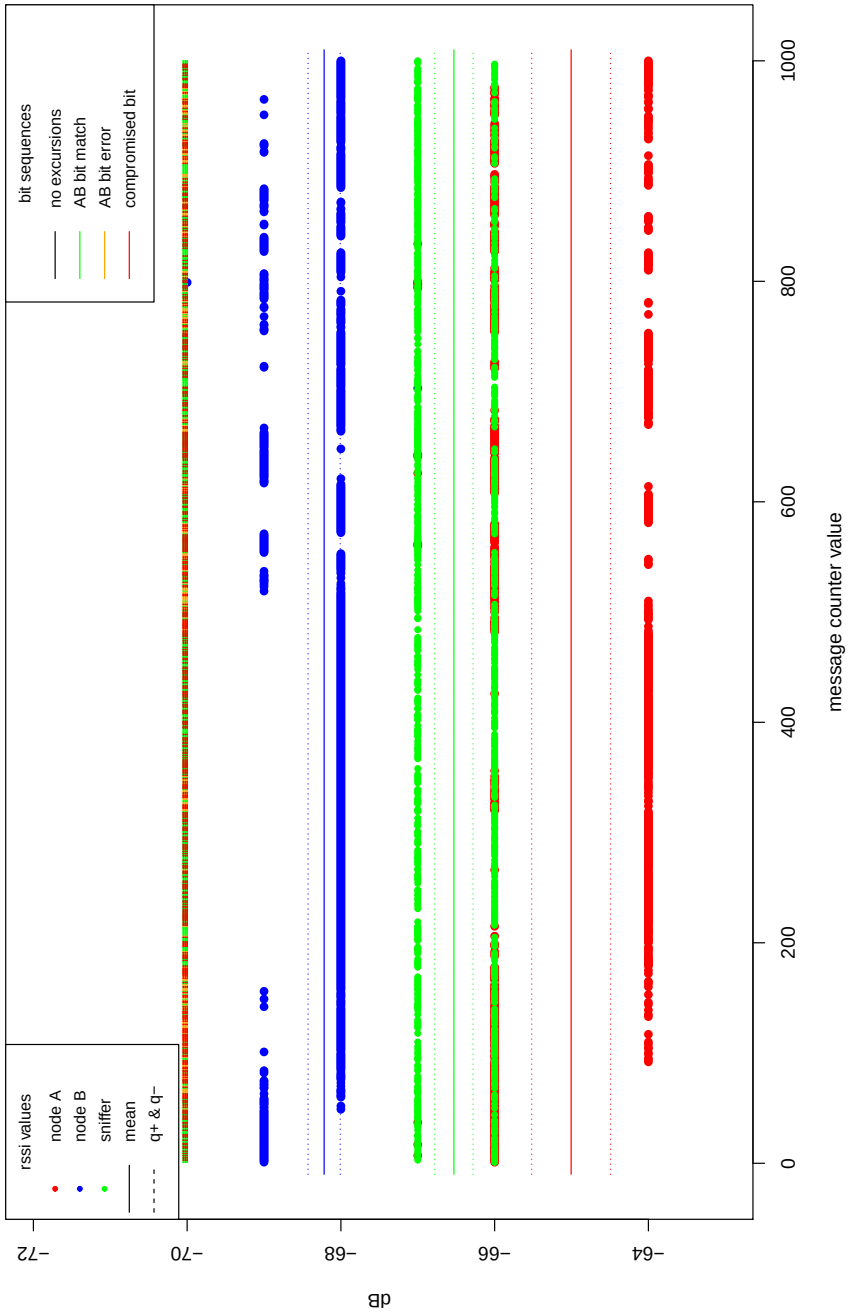

Figure B.3: 1000 bits output