On Wireless Channel Parameters for Key Generation in Industrial Environments

Dan Kreiser, Zoya Dyka, Stephan Kornemann, Christian Wittke, Ievgen Kabin, Oliver Stecklina and Peter Langendoerfer

***Abstract*— The advent of industry 4.0 with its idea of individualized mass production will significantly increase the demand for more flexibility on the production floor. Wireless communication provides this type of flexibility but puts the automation system at risk as potential attackers now can eavesdrop or even manipulate the messages exchanged even without getting access to the premises of the victim. Cryptographic means can prevent such attacks if applied properly. One of their core components is the distribution of keys. The generation of keys from channel parameters seems to be a promising approach in comparison to classical approaches based on public key cryptography as it avoids computing intense operations for exchanging keys. In this paper we investigated key generation approaches using channel parameters recorded in a real industrial environment. Our key results are that the key generation may take unpredictable long and that the resulting keys are of low quality with respect to the test for randomness we applied.**

***Index Terms*— automation systems, cryptography, key exchange, latency, real-time, wireless communication**

I. Introduction

In recent years wireless communication systems have become more and more popular in automation systems. One of the reasons behind this development is that maintenance costs are reduced due to the fact that lesser cables need to be replaced. Another driving factor is that factory floors can be adapted by far quicker than if wired systems are used. The latter will become more important in the near future with the idea of industry 4.0 that postulates individualized mass production, which requires highly flexible production processes. Even though wireless systems are providing obvious benefits for automation systems they open up new challenges with respect to dependability and security. In this article we focus on the latter. Due to the wireless communication potential attackers do not longer need to get access to the premises of the victim to eavesdrop or manipulate the communication of the automation system. The consequences are that competitors may gain know how of the victim, that due to manipulation of the communication the production is disturbed e.g. quality of the products reduced, or the production process is stopped etc. In order to prevent attackers from gaining confidential information and from interfering with the running system security means are essentially needed. The proper use of cryptographic means will help to ensure:

- confidentiality of the data exchanged in the automation system;
- integrity of the messages exchanged;
- authentication of the communication partners.

The security of the cryptographic means depends on the secrecy of the keys used. This means that the distribution of the keys is essential for the effectiveness of the security means. Key distribution is known as one of the most demanding problems in the security area. In automation systems the key distribution becomes even more challenging as at least a part of these systems consists of resource constraint devices and due to the fact that these systems need to fulfill real time requirements and require short latency. Additional time and energy are needed for the generation and agreement of cryptographic keys. Thus, the applied key distribution approaches can significantly influence the latency of the message transmission. Taking into account these constraints the idea of generating keys from wireless channel parameters seems very appealing as it avoids computing intense asymmetric cryptography operations.

We investigated the applicability of wireless channel parameters to the key generation in a real industrial environment and compared this approach with classical key exchange methods.

We run our experiments in the model factory of the Innovation Centre Modern Industry Brandenburg, Chair of Automation Technology at BTU Cottbus-Senftenberg [1]. This model factory is well suited to make realistic measurements in a controllable environment. We defined six different set-ups for our measurements to reflect changing conditions in such an environment. We collected and analyzed about 5000 Received Signal Strength Indicator (RSSI) values

Dr. Dan Kreiser is with IHP, Im Technologiepark 25, 15236 Frankfurt (Oder), Germany (e-mail: kreiser@ihp-microelectronics.com).

Dr. Zoya Dyka is with IHP, Im Technologiepark 25, 15236 Frankfurt (Oder), Germany (e-mail: dyka@ihp-microelectronics.com).

Stephan Kornemann is with IHP, Im Technologiepark 25, 15236 Frankfurt (Oder), Germany (e-mail: kornemann@ihp-microelectronics.com).

Christian Wittke is with IHP, Im Technologiepark 25, 15236 Frankfurt (Oder), Germany (e-mail: wittke@ihp-microelectronics.com).

Ievgen Kabin is with IHP, Im Technologiepark 25, 15236 Frankfurt (Oder), Germany (e-mail: kabin@ihp-microelectronics.com).

Dr. Oliver Stecklina, was with BTU Cottbus-Senftenberg (e-mail: Oliver.Stecklina@b-tu.de].

Prof. Dr. Peter Langendoerfer, is with IHP, Im Technologiepark 25, 15236 Frankfurt (Oder), Germany (e-mail: langendoerfer@ihp-microelectronics.com).

per measurement with the goal to generate keys using different quantization algorithms and to verify the quality of the generated keys. We estimated the time and energy of the key generation (for the best case) to compare this approach with classical key agreement methods. Our major findings are:

- The time needed to collect sufficient RSSI values to generate a key is unpredictable. It may even take days, thus for real time environments this approach is unsuitable.
- The quality of the keys generated with respect to their randomness is low which may simplify attacks.
- The time and energy needed to establish a key using classical methods are deterministic and lower than those of the key generation from channel parameters. This holds even true if we compare the worst case for classical methods with the best case for key generation from channel parameters.

For the last point we admit that we did our calculation based on the assumption that the asymmetric cryptographic operations are accelerated by a special hardware. We are convinced that this is justified since this hardware requires 0.3mm² in a 130 nm technology only, i.e. its cost is in the range of 10-20 cent.

The rest of this paper is structured as follows. In the following section we introduce the key generation approaches used for comparison in this paper. Our experiments are presented in section III. Section IV illustrates quantization algorithms discussed in the literature and used in our experiments to generate keys out of channel parameters. In the following section we analyze the quality of the generated keys in detail. Section VI provides a profound comparison of standard key establishment approaches and the channel based key generation investigated here. This paper concludes with a short summary of our major results and a short outlook on next steps.

II. Key Generation Approaches

Cryptographic approaches are means to guarantee the confidentiality of a (wireless) communication and the (mutual) authentication of the participants. To be confidential messages have to be encrypted using a cryptographic key. If the cryptographic key used for encryption and for decryption is the same, the cryptographic approach is called symmetric. The currently recommended length of symmetric keys is 128 bit using the Advanced Encryption Standard (AES) algorithm [2].

The typical problems of symmetric cryptographic approaches are: how the two (or more) participants can securely exchange a shared secret, i.e. the cryptographic key, and how participants can authenticate each other. These problems can be resolved using asymmetric cryptographic approaches such as RSA or elliptic curve cryptography (ECC). In these approaches each participant has a private key, that is its secret, and a public key that can be known by all other parties. The private key and the public key are a pair. Everybody can use the public key of a party to encrypt a message and only the owner of the corresponding private key can decrypt the received message. This can be used to exchange a newly generated symmetric cryptographic key. The latter can be encrypted and confidentially exchanged, i.e. the symmetric key can be encrypted using the public key of the receiver and sent to him as a cipher text. Furthermore, the sender can use its private key to generate a digital signature that can be used for authentication. The receiver can verify the identity of a sender using the public key of the sender (digital signature verification) and can decrypt the message using its own private key.

ECC is much faster than RSA, but in comparison to the AES ECC is slow and leads to a high computational burden for encryption/decryption of messages. Due to this fact, ECC is usually applied only for (mutual) authentication of participants and for confidential exchange of symmetric cryptography keys. The encryption/decryption of the messages is then done using the securely exchanged symmetric key.

Table I schematically shows some key exchange protocols. The Diffie-Hellmann protocol (DH) was proposed in 1967 and exploits modular exponentiation in finite fields to share a secret between two participants [3]. Both participants use their public knowledge about a prime field *GF(p)* with a generator *g* to exchange the secret key $\boldsymbol{secret_{AB}}$. Using this protocol each participant – Alice and Bob – generates a random number once and performs the modular exponentiation twice (see Table I). The modular exponentiation is a time consuming and processing intense operation, especially if the known numbers *p, g* and the randomly generated numbers *a* and *b* are large. Instead of the modular exponentiation the multiplication of an EC point *P* with a scalar *k*, denoted as *kP*-operation, can be used to share a secret. This protocol is called EC Diffie-Hellmann (ECDH) [4]. The sender and receiver use their public knowledge about an EC over a finite field. Each of the participants generates a random number once and performs the *kP*-operation twice (see Table I). The DH and ECDH protocols work without using public keys of participants and without any authentication of the participants. If (mutual) authentication is required, it has to be done additionally.

A secret key can be shared between Alice and Bob also by using the ECC encryption. The private and public keys of Alice are $\boldsymbol{k_A}$ and $\boldsymbol{Pub_A}$, respectively. These keys are a key pair: $\boldsymbol{(k_A;Pub_A)}$. Bob is the owner of the key pair $\boldsymbol{(k_B;Pub_B)}$. One of the participants, for example Alice, generates the symmetric key $\boldsymbol{secret_{AB}}$ and sends it encrypted to Bob. Alice performs two EC point multiplications to encrypt the message using Bob's public key $\boldsymbol{Pub_B}$; Bob performs only one *kP*-operation to decrypt the message using his own private key $\boldsymbol{k_B}$ (see Table I). Here is assumed: Alice is sure, that the owner of the public key $\boldsymbol{Pub_B}$ is really Bob. Thus, Alice can be sure that only Bob can use the $\boldsymbol{secret_{AB}}$ for the next communication. If Bob requires the authentication of Alice, it has to be done additionally. This way of establishing a common key is denoted as key transport protocol.

Key exchange protocols using the public keys of both participants are computationally more expensive, for example protocols based on the Matsumoto, Takashima, and Imai (MTI) approach. Two types of MTI protocols – A0 and C0 – were proposed in [5]. Alice and Bob know the public keys of each other and assume that the owner of $\boldsymbol{Pub_A}$ is Alice and the owner of $\boldsymbol{Pub_B}$ is Bob. If the verification of key certificates is required, it has to be done additionally. Alice and Bob generate a random number once and perform three *kP* operations according to protocol A0 (see Table I).

Corresponding to protocol C0 each participant generates a random number and then performs two *kP* operations and one division in the finite field.

TABLE I
CLASSICAL KEY EXCHANGE APPROACHES

key exchange approach	***A*** (Alice)	***B*** (Bob)
Diffie-Hellmann	knows $g \in GF(p)$ • generates a random a • sends to ***B***: $y_A = g^a \bmod p$ • receives y_B • $secret_{AB} = (y_B)^a \bmod p$	knows $g \in GF(p)$ • generates a random b • sends to ***A***: $y_B = g^b \bmod p$ • receives y_A • $secret_{AB} = (y_A)^b \bmod p$
EC Diffie-Hellmann	knows point ***G*** of an EC • generates a random a • sends to ***B***: $Q_A = a \cdot G$ • receives Q_B • calculates $Q = a \cdot Q_B$ • $secret_{AB}$=*x-coord of Q*	knows point ***G*** of an EC • generates a random b • sends to ***A***: $Q_B = b \cdot G$ • receives Q_A • calculates $Q = b \cdot Q_A$ • $secret_{AB}$=*x-coord of Q*
based on ECC encryption	**A** is owner of: $(k_A; Pub_A)$ and knows Pub_B • generates $secret_{AB}$ • sends the encrypted secret: m=**Encr**($secret_{AB}$, Pub_B)	**B** is owner of: $(k_B; Pub_B)$ • decrypts the received message using its own k_B: $secret_{AB}$=**Decr**(m, k_B)
MTI based	**A** is owner of: $(k_A; Pub_A)$ and knows point ***G*** of an EC, and knows Pub_B • generates a random a	**B** is owner of: $(k_B; Pub_B)$ and knows point ***G*** of an EC, and knows Pub_A • generates a random b
A0	• sends to ***B***: $R_A = a \cdot G$ • receives R_B • calculates $Q = k_A \cdot R_B + a \cdot Pub_B$ • $secret_{AB}$=*x-coord of Q*	• sends to ***A***: $R_B = b \cdot G$ • receives R_A • calculates $Q = k_B \cdot R_A + b \cdot Pub_A$ • $secret_{AB}$=*x-coord of Q*
C0	• sends to ***B***: $R_A = a \cdot Pub_B$ • receives R_B • calculates $Q = (a/k_A) \cdot R_B$ • $secret_{AB}$=*x-coord of Q*	• sends to ***A***: $R_B = b \cdot Pub_A$ • receives R_A • calculates $Q = (b/k_B) \cdot R_A$ • $secret_{AB}$=*x-coord of Q*

All classical key exchange approaches given in Table I need additional steps for the (mutual) identification/authentication of participants or are vulnerable to some attacks, for example to the man in the middle attack. To increase the resistance of key exchange approaches against these attacks more complex protocols based on MTI were proposed, for example Elliptic Curve Menezes-Qu-Vanstone (ECMQV) approach [6].

A completely different and relatively new kind of key agreement approaches uses a common source of randomness that is identical only for pairs of communication partners, for example the channel parameters of a wireless communication in a room. This method is based on the reciprocity theorem. If Alice sends an ideal *probe* signal to Bob, Bob receives a noised signal that is a result of the multipath propagation of the *probe* signal in the room. If Bob sends the identical *probe* signal to Alice and if the environment is static (i.e. not changing), Alice receives the same noised signal as Bob i.e. a signal with the same distortion due to the multipath propagation in the room.

The multipath propagation in the room depends on the geometry of the room. Real rooms are not symmetric. They can have small walls, tables, devices, etc. causing the wave reflections. This results in a multipath propagation of a signal in the room. Let's assume that Alice and Bob have devices, which can send really identical *probe* signals (request from Bob and answer from Alice) and the environment is static while the *probe* signals are exchanged then the environment causes the same deformation of the *probe* signals only for Alice and Bob. A malicious receiver Eve can listen to the communication of Alice and Bob, but receives the *probe* signal with other noise than Alice or Bob, if the position (coordinates) of Eve differs from coordinates of Alice and Bob. This can be used to generate a common secret for Alice and Bob only. Alice and Bob need to use the same quantization algorithm, i.e. an algorithm to obtain the $secret_{AB}$ from the identically noised *probe* signals. For UWB (Ultra-WideBand) or OFDM (Orthogonal Frequency-Division Multiplexing) communications many key bits can be generated using a single exchange of a *probe* signal [7], [8].

The most researched case is that only one key bit can be generated per *probe* signal exchange, for example using RSSI values or the phase of the signal. In this case Alice and Bob have to exchange the identical *probe* signal at least n times to generate an n-bit long key. Fig. 1 illustrates such a key generation.

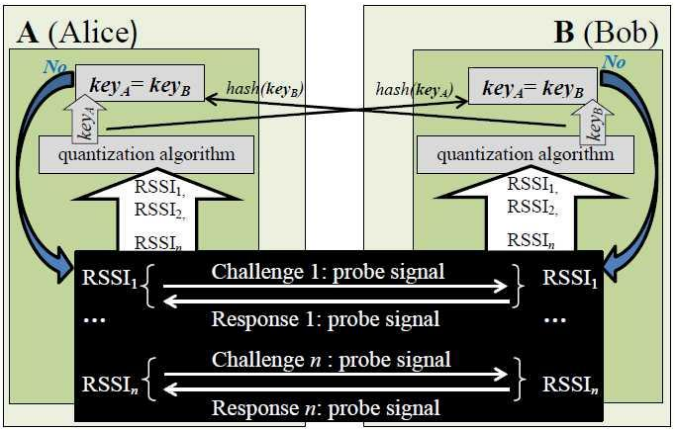


Fig. 1. Generation of a common secret key $secret_{AB} = key_A = key_B$ between two communication partners – Alice and Bob – using channel parameters of a state of the art wireless communication.

In case of key generation using channel parameters the environment has to be:

- static while the *probe* signal is exchanged. Otherwise Alice and Bob can have different RSSI values for the generation of the key bit;
- dynamic after each exchange of the *probe* signal: the geometry of the room or the coordinates of Alice and Bob have to be changed significantly. Otherwise Alice and Bob can have the same RSSI values as for the previous pair of *probe* signals for the generation of the next key bit, so that the sequence of the generated key bits will not be random.

In reality the exchanged *probe* signals are similar but not identical and the environment can be non-static while the *probe* signal is exchanged. These differences can cause differences in the keys generated on Alice's and Bob's sides, which can be compensated by collecting more than n RSSI values and using adequate quantization algorithms. In such a

case an additional step – the verification of the equivalency of Alice's and Bob's generated keys $\boldsymbol{key_A}$ and $\boldsymbol{key_B}$ respectively – is necessary. This can be done for example by sending and comparing the hash values of both keys (see Fig. 1). Only if both hash values are equal $hash(\boldsymbol{key_A})=hash(\boldsymbol{key_B})$ it will be assumed that the generated keys are the same, i.e. $\boldsymbol{secret_{AB}}=\boldsymbol{key_A}=\boldsymbol{key_B}$. The hash can be sent only once, i.e. from Alice to Bob. If the generated keys are not identical, the key generation approach has to be repeated from the beginning, i.e. starting with the collection of new RSSI values. Another disadvantage of this method is the fact that the generated keys can be predictable or even manipulated using the environment. For example, changes in the room geometry (or coordinates of Alice and Bob) can be predictable; the quality of the communication can be manipulated e.g. by adding walls or by jamming.

Thus, the time of the key generation using channel parameters is long and not deterministic. The quality of the generated keys depends on a lot of factors and the problem of the (mutual) authentication of participants is not solved. In spite of these disadvantages the generation of a symmetric cryptographic key using channel parameters is investigated in literature in [9]-[12] as a promising alternative to the classical key exchange approaches (see Table I), because the complex computations in Galois fields can be avoided. Important is that not only the computations but also sending and receiving of the data require energy and time. The energy that is necessary for data transmission can be even higher than the energy for computations. Due to this fact, not only the computational burden but also the amount of transmitted data has to be estimated and taken into account for a fair comparison of different key exchange approaches. Table II gives an estimation of transmitted information and computational burden for selected classical key exchange approaches and for the key generation using channel parameters of a wireless communication.

TABLE II
KEY EXCHANGE APPROACHES: ESTIMATION OF EFFORT

	ECDH		***ECC***		***MTI(A0)***		***Using channel parameters***	
	Alice	**B**ob	**A**lice	**B**ob	**A**lice	**B**ob	**A**lice	**B**ob
generation of a random number	233 bits	233 bits	128 bits	-	233 bits	233 bits	-	-
sent information	234 bits	234 bits	468 bits	-	234 bits	234 bits	$x\cdot$(128α *probe* signals, 256 bits hash)	$x\cdot$(128α *probe* signals)
received information	234 bits	234 bits	-	468 bits	234 bits	234 bits	$x\cdot$(128α *probe* signals)	$x\cdot$(128α *probe* signals, 256 bits hash)
computation burden	2 *kP*	2 *kP*	2 *kP*	1 *kP*	3 *kP*	3 *kP*	$x\cdot$(128α values of channel parameters, quantization algorithm, hash)	$x\cdot$(128α values of channel parameters, quantization algorithm, hash)

The key exchange approach using ECC is the best of the classical methods regarding the computational burden, i.e. it is better than ECDH and MTI based approaches. Only a 128 bit long random number has to be generated that is about 100 bit less than for other methods, and only 3 *kP* operations have to be executed in total i.e. one *kP* less than for ECDH and 3 *kP* less than for MTI. Using our hardware accelerator for NIST EC *B-233* [13] one *kP* operation needs about 13 000 clock cycles to be executed and consumes about 2 µJ energy. Using the classical approaches given in Table II in total 468 bits of information have to be transmitted to exchange a 128 bit long symmetric session key, i.e. 2 packets with up to 234 bits are necessary for a compressed representation of two 233 bit long EC point coordinates. The transmission of 468 bits of data consumes about 360 µJ for sending and 180 µJ for receiving using IHP FeuerWhere sensor nodes [14], [15]. Thus the energy of the data transmission 360 µJ + 180 µJ = 540 µJ is 270 µJ higher than the energy for a *kP* operation.

The assessment of the computational burden and of the data transmission energy for the approaches using the channel parameters is not deterministic. Depending on the applied quantization algorithm a different number of *probe* signals has to be exchanged. Some algorithms need exactly 128 *probe* signal exchanges for the generation of the 128 bit long symmetric secret key but others need α times more exchanges (see coefficient α in Table II). Alice and Bob obtain a channel parameter, for example an RSSI value, for each *probe* signal. With the goal to reduce the data transmission energy packets without a payload, i.e. empty packets can be exchanged and used for calculating the RSSI value. This means that only for the collection of the data, that are inputs for the quantization algorithm, Alice and Bob have to exchange $128\cdot\alpha$ *probe* signals which consumes about $22.4\cdot\alpha$ mJ energy on each side. Energy and time of the calculation of $128\cdot\alpha$ RSSI values from *probe* signals, processing of these values using the quantization algorithm and the calculation of the hash of the generated key on each communication side can be used for the estimation of the minimal computational burden. Even if the calculation burden is negligible in comparison to a *kP* operation, the calculated hash values have to be exchanged, i.e. 2 packets with 256 bits of the data have to be sent, consuming about 579 µJ using IHP FeuerWhere sensor nodes. If the exchanged hash values are different, the key generation approach has to be repeated from its beginning. The variable *x* in Table II represents the number of key generation loops performed until an identical secret key will be generated by both communication partners.

III. COLLECTION OF THE KEY MATERIAL IN AN INDUSTRIAL ENVIRONMENT

A. Measurement Setup

We performed all our measurements in a real industrial environment, which is a model factory with two movable robot arms and a milling machine. The measurement setup is shown in Fig. 2.

We used for our measurements four identical sensor nodes: *Master* ***M***, *Slave* ***S***, *Eve1* (shortly denoted as ***E1***) and *Eve2* (***E2***). The *Master* sends a *probe* signal to the *Slave:* $\boldsymbol{M\rightarrow S}$. The two malicious nodes, *Eve1* and *Eve2*, can also listen to this communication. The distance between the *Slave* and *Eve1* is small, only 10 cm (see Fig. 2) which is about one third of the wave length λ of the communication frequency 868 MHz ($\lambda\approx$35 cm).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2748182, IEEE Access

The distance between *Slave*, *Master* and *Eve2* is in all our measurement by far larger than the wave length λ. The positions of *Slave*, *Eve1* and *Eve2* are the same for all measurements. Only the position of the *Master* was changed from position 1 to position 2 (see Fig. 2).

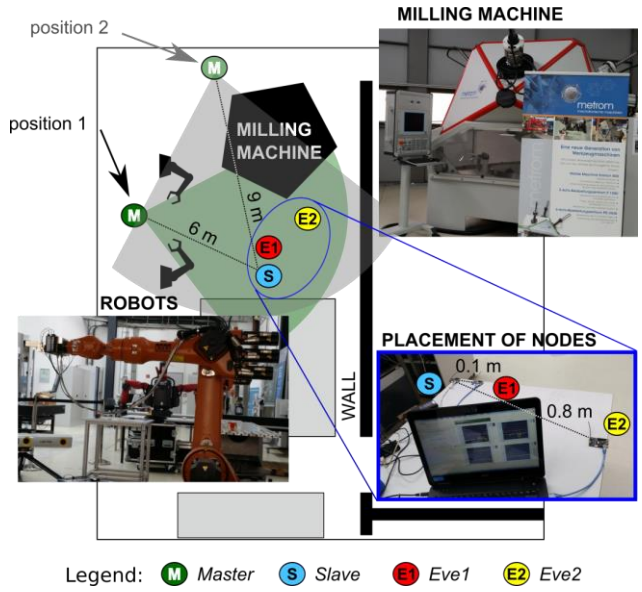


Fig. 2. Measurement setup in a model factory.

The experiments were made with the goal to investigate how the geometry of the room and the activity of industrial instruments such as the robots or the milling machine influence the parameters of the wireless communication channel and in consequence the generated keys. Table III gives an overview of the different measurement cases.

TABLE III
OVERVIEW OF MEASUREMENT CASES

Measurement case	*Master*	static room	milling machine active	robots active
1a	position1	√	-	-
1b	position1	-	√	-
1c	position1	-	-	√
2a	position2	√	-	-
2b	position2	-	√	-
2c	position2	-	-	√

In Fig. 3 the observed communications are shown schematically.

For our measurements we used the IHP FeuerWhere [15] low-power sensor nodes (see Fig. 4) with following features:

- three low-power transceivers: Sub 1 GHz RF transceiver (TI CC1101), 2.4 GHz RF transceiver (TI CC2500), 2.4 GHz IEEE 802.15.4/ZIGBEE® RF transceiver (TI CC2520);
- MMCX antenna connector for each RF transceiver;
- interoperable with other IEEE 802.15.4 devices;
- TI MSP430 microcontroller (16k RAM, 256k Flash);
- µSD card slot.

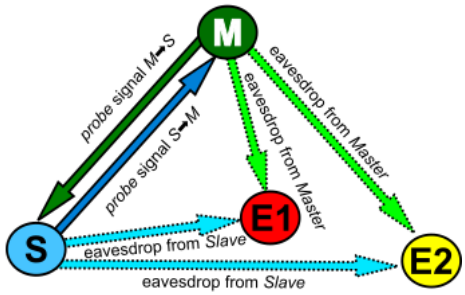


Fig. 3. Observed communications:
The *Master* sends *probe* signals to the *Slave* (this ***M→S*** communication is shown as a dark green arrow). The two malicious nodes, ***E1*** and ***E2*** can listen to this communication. This fact is depicted by the light green dotted arrows (***E1←M***) and (***E2←M***).
The *Slave* responds to the *Master* with *probe* signals ***S→M*** (blue arrow). ***E1*** and ***E2*** can listen to this communication, too. This fact is depicted by the light blue dotted arrows (***E1←M***) and (***E2←M***).

Fig. 4. IHP FeuerWhere sensor node.

In our experiments the sensor nodes are communicating at a frequency of 868 MHz using the TICC1101 radio module. The *Master* sends a *probe* signal to the *Slave* and waits for an answer. The *Slave* sensor node receives this signal and reads its current Received Signal Strength Indicator (RSSI) value from the radio module. Afterwards the *Slave* responses with a new *probe* signal to the *Master*. Now the *Master* obtains its RSSI value of the received signal. The malicious nodes *Eve1* and *Eve2* listen to the communication between *Master* and *Slave* and obtain their own RSSI values for all received *probe* signals. The structure of the packet that we used as a *probe* signal is shown in (see Fig. 5).

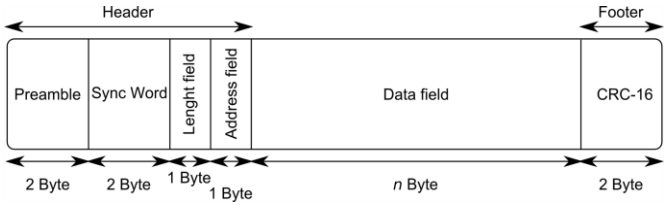


Fig. 5. Structure of packets used as *probe* signals in our experiments:
The RSSI value will be obtained using the preamble. The data field in our experiments consists of *n*=4 byte.

The packet consists of a preamble, a synchronization word, a length, an address, a data and a CRC-field. The RSSI value of the packet is determined using the preamble. This means that the data field in such packets can be empty if the packet is

used as a *probe* signal. In our experiments the data field is always 4 bytes long:

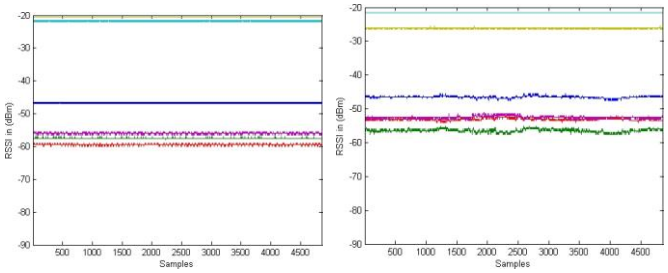
- the 1st byte is the identifier of the sender;
- the 2nd and 3rd bytes contain the sequence number of the exchanged *probe* signal. This is necessary for our experiments to determine pairs of request and answer signals. Only these pairs have to be processed further. Each request without an answer has to be excluded from further processing;
- the 4th byte of the data field is only used for measurement purposes in our experiments. As the *Master* and the *Slave* do not have sufficient memory to store all RSSI values for later processing these values need to be send to a storage device. In simple words the 4th byte is used to piggyback the RSSI value determined from the last *probe* of the *Master* and the *Slave*, respectively. These values are read and stored to a laptop device by ***E1*** and ***E2*** that are attached to the laptop device for later analysis.

Thus, with the goal to keep the communication time between *Master* and *Slave* as small as possible we used nodes *Eve1* and *Eve2* not only for listing to the communication but also for processing all received packets. The information allocated in the 4th byte of packets allows using the nodes *Master* and *Slave* only for the exchange of the *probe* signals and obtaining RSSI values: after sending the request signal the *Master* changes its mode immediately from "send" to "receive" and vice versa. The change of the modes on both nodes occurs quasi parallel in time and takes in our experiments about 40 ms. The duration of the *probe* signal is 1.694 ms. Thus, the minimal time needed for the communication on the *Master* node is 1.7+40.0+1.7=43.4 ms.

We executed 12000 request-answer cycles in our experiments lasting about 20 minutes but we could collect only about 5000 pairs because a lot of packets are lost. For further processing we selected 4864 request-answer pairs. This was sufficient for generating 38 keys with a length of 128 bits. We didn't collect more measurements because a key exchange approach that requires a lot of energy and time is not reasonable for application in industrial environments.

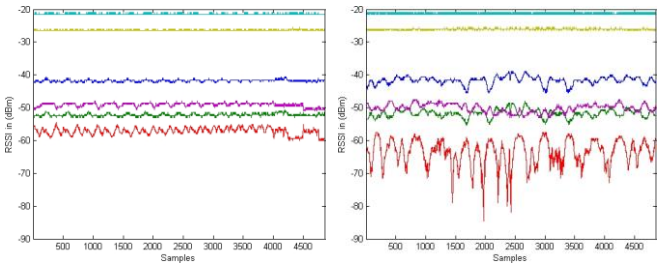
B. Measurement Results

Fig. 6 – Fig. 8 show the RSSI values for each node in each of the measurment cases defined in Table III. The RSSI values collected on the *Slave* node are shown as green line in Fig. 6 – Fig. 8 reflecting that the the communication from the *Master* to the *Slave* (***M→S***) is shown in green color in Fig. 3. The blue line in Fig. 6 – Fig. 8 represents the RSSI values obtained on the *Master* node from the *Slave*'s signals. The RSSI values obtained on node *Eve1* from the ***M→S*** communication are shown with a red line and those from the *Slave* to the *Master* (***S→M***) are shown in light-blue color. The violet and yellow lines show the RSSI values collected on node *Eve2* from the ***M→S*** and ***M→S*** communication respectively.



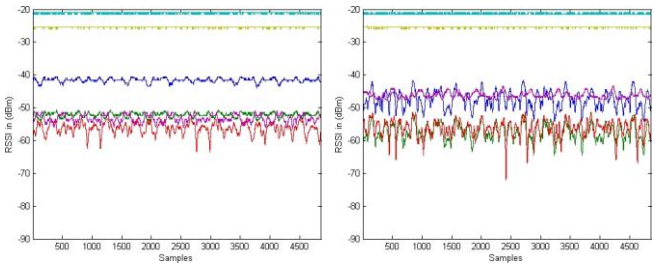
Measurement case 1a *Measurement case 2a*

Fig. 6. Collected RSSI values in a static industrial environment.



Measurement case 1b *Measurement case 2b*

Fig. 7. Collected RSSI values when the milling machine is active.



Measurement case 1c *Measurement case 2c*

Fig. 8. Collected RSSI values when the robots are active.

IV. Quantization Of Measured Data: Generation Of The Keys

To generate a common secret key out of the RSSI values different quantization algorithms can be used. The main objectives of such algorithms are shown below.

1) Decreasing the influence of non-reciprocity of the channel.
   An ideal case is:
   - each new key, generated on the *Master* node is identical to the key, simultaneously generated on the *Slave* node;
   - a malicious node that can listen to the communication obtains a key that differs significantly from the key of the *Master* and the *Slave*.
2) Increasing the quality of the generated keys.
   In an ideal case the agreed keys are a random sequence of bits.
3) Increasing the key generation rate.
   In an ideal case the time for the key agreement using channel parameters is significantly shorter than the one using any other key exchange approach, for example compared to the approaches based on ECC.

In our experiments we generated keys using 5 different quantization algorithms. They are shortly described in the following subsection using a part of our collected RSSI values i.e. the pairs from number 901 up to 940 as an example. The estimation of the quality of the generated keys and the channel reciprocity will be given in the next sections.

A. Quantization algorithms

Algorithm 1

This algorithm uses the mean value of 128 collected RSSI values as threshold in order to transform the RSSI values into a sequence of 128 key bits. This algorithm is one of most referenced ones in the literature which is the reason why we implemented it.

1) The *Master* and the *Slave* calculate the mean of 128 collected RSSI values. The mean values on both nodes can be (and usually are) different.

The *Master* and the *Slave* generate their keys using the same formula (algorithm):

$$k_i = \begin{cases} 1, & if \quad RSSI_i > mean \\ 0, & if \quad RSSI_i \le mean. \end{cases} \tag{1}$$

In formula (1) k_i is the value of the key bit at the position i ($0 \le i \le 127$). Fig. 9 illustrates the generation of the key on the *Master* and the *Slave* nodes using Algorithm 1.

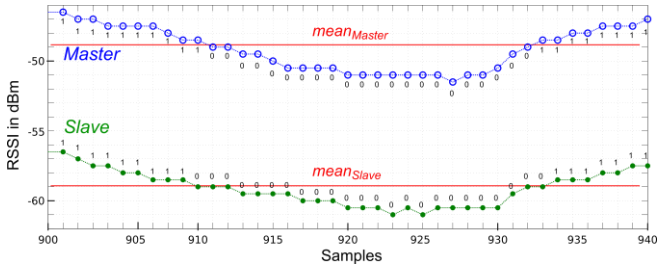


Fig. 9. Illustration of the generation of a part of the key using Algorithm 1 all RSSI values above the *mean* line result in a '1' for the corresponding key bit, all values below or on the *mean* line in a '0'.

Corresponding to formula (1) each RSSI value greater than the *mean* value results in a key bit '1' and each RSSI value smaller or equal to the *mean* value causes the key bit to be '0'. The *mean* value for the *Master* $mean_{Master}$*=-48.94 dBm* was obtained using RSSI values collected on the *Master* node for the generation of the 8th key i.e. using the RSSI numbers 897 up to 1024. In the same way the mean value $mean_{Slave}$*=-58.95 dBm* for the *Slave* was calculated. The obtained key bit sequences are:

Master: 11111111110000000000000000000000011111111
Slave: 11111111100000000000000000000000001111111

The generated bit sequences are not identical: 2 of 40 bits, in our example are different, displayed in red. This fact shows that the reciprocity of the channel is not ideal. Moreover, the generated bit sequences consist of long subsequences of '1' and '0'.

Algorithm 2

This algorithm is based on a comparison of the current and the previous RSSI values as follows:

$$k_i = \begin{cases} 1, & if \quad RSSI_i \ge RSSI_{i-1} \\ 0, & if \quad RSSI_i < RSSI_{i-1}. \end{cases} \tag{2}$$

For generating a 128 bits long key 129 RSSI values are necessary. Fig. 10 illustrates the generation of the key on the *Master* and the *Slave* nodes using formula (2).

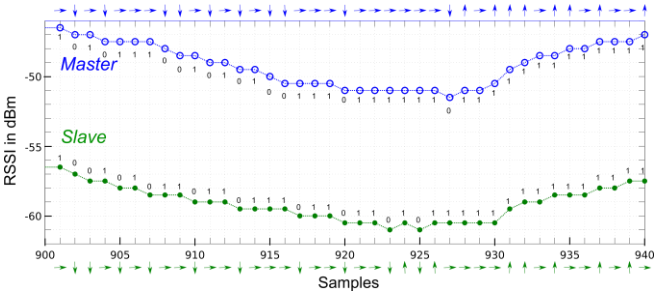


Fig. 10. Illustration of the generation of a part of the key using Algorithm 2; arrows represent the relation between 2 neighbor RSSI values.

The fact that the current RSSI value $RSSI_i$ is equal to the previous RSSI value i.e. to $RSSI_{i-1}$ is shown in Fig. 10 as '→'. The symbol '↑' represents that $RSSI_i > RSSI_{i-1}$ and the symbol '↓' illustrates that $RSSI_i < RSSI_{i-1}$.

The obtained key bit sequences using the RSSI values and this algorithm as shown in Fig. 10 are:

Master: 10101110010101001110111111101111111111111
Slave: 10010101101101110110110101011111111111111

Also here, the generated bit sequences are not identical: 13 of 40 bits are different (see positions marked in red). In comparison to Algorithm 1 the generated bit sequences look more random but their identical parts also consist of a long sequence of '1'. Algorithm 2 is a version of the Adaptive Secret Bit Generation (ASBG) presented in [16].

In [16] the authors claim that this algorithm provides the highest entropy of all algorithms researched by them. We could not verify this claim as important information such as the number of RSSI values, measured in total, are missing in [16]. In addition in [16] there are discrepancies between what the authors report in their tables and in their graphs. [12] reports on an evaluation of the algorithm proposed in [16] using their own implementation und measurement results. The results in [16] also show an insufficient randomness of the generated keys.

Algorithm 3

This algorithm was proposed in [10] and needs 128 RSSI values for generating 128 bits long key.

The *Master* calculates the maximal and minimal values in the set of collected RSSI values, denoted as $RSSI_{max}$ und $RSSI_{min}$. The interval $\Delta = |RSSI_{max}-RSSI_{min}|$ will be divided into n equal intervals, called quantization levels [10]. In our implementation $RSSI_{max}=-42\ dBm$, $RSSI_{min}=-53.5\ dBm$ and $n=8$, i.e. we have 8 quantization levels that correspond to the 1-bit Adaptive Quantization Scheme [10]. Each level is $|53.5-42|/8=1.4375\ dBm$. Each quantization level, numbered from 1 up to 8, can be numbered using a Gray code (see Table IV):

TABLE IV
NUMBERING QUANTIZATION LEVELS USING 3-BIT GRAY CODE

index numbers of quantization levels, in decimal	index numbers of quantization levels, as 3-bit Gray code			Corresponding interval of RSSI values, i.e. quantization levels
	d1	d0	e	
1	0	0	0	$[RSSI_{min};\ RSSI_{min}+\ \Delta/8]$
2	0	0	1	$(RSSI_{min};\ RSSI_{min}+2\cdot\Delta/8]$
3	0	1	1	$(RSSI_{min};\ RSSI_{min}+3\cdot\Delta/8]$
4	0	1	0	$(RSSI_{min};\ RSSI_{min}+4\cdot\Delta/8]$
5	1	1	0	$(RSSI_{min};\ RSSI_{min}+5\cdot\Delta/8]$
6	1	1	1	$(RSSI_{min};\ RSSI_{min}+6\cdot\Delta/8]$
7	1	0	1	$(RSSI_{min};\ RSSI_{min}+7\cdot\Delta/8]$
8	1	0	0	$(RSSI_{min};\ RSSI_{max}]$

Each of the collected RSSI values is assigned to a quantization level, i.e. mapped into one of the intervals listed in the last column of Table IV. Thus, each RSSI value can be represented using its Gray code index number. For example, the *Master*'s $RSSI_{901}$=-46.5 dBm is in quantization level 5 (see Fig. 11) and will be represented by d1d0e=110. Also the RSSI values 902-907 are in quantization level 5 and will be represented by the same 3-bit long number d1d0e=110. The Gray code representation of other RSSI values of the *Master* is shown at the top of Fig. 11.

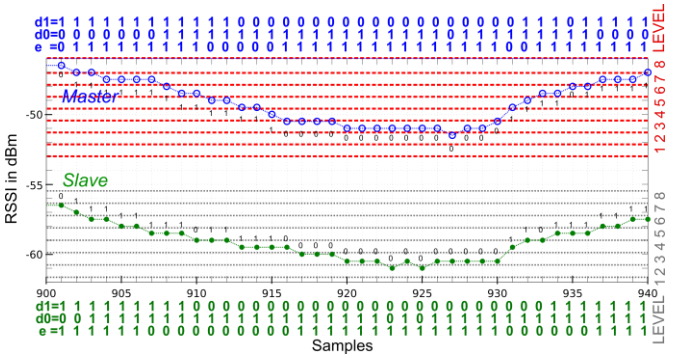


Fig. 11. Illustration of the generation of a part of the key using Algorithm 3.

The *Slave* processes its RSSI values in the same way: it calculates its own interval $\Delta=|RSSI_{max}-RSSI_{min}|=|63.5-52.5|=1.375\ dBm$; divides the interval into 8 quantization levels and represents each RSSI value as a 3-bit Gray code. This representation is shown at the bottom of Fig. 11. For example the *Slave*'s RSSI values 917-930 are in quantization level 3 and are represented as d1d0e=011.

After representing RSSI values as Gray code the *Master* sends all "e"-bit values to the *Slave*, i.e. the *Master* sends its 128 bit long "e-line". The *Slave* replaces its own "e-line" with the one received from the *Master* and uses this modified Gray code representation for obtaining the key. The *Master* and the *Slave* generate the sequence of key bits using their 3-bit Gray code representation of RSSI values as follows:

$$k_i = \begin{cases} d0, & if \quad e_i = 0 \\ d1, & if \quad e_i = 1. \end{cases} \tag{3}$$

For example, the *Master* obtains two key bit values as '1' and '1' using the RSSI values $RSSI_{909}$ and $RSSI_{910}$ respectively. Both RSSI values are located in quantization level 6 (see Fig. 11) and they are represented as d1d0e=111 (see Table IV), i.e. $e=1\rightarrow$ key bit k_i =d1=1.

On the *Slave* node $RSSI_{909}$ is also represented as d1d0e=111. The *Slave*'s own value was d1d0e=110 because $RSSI_{909}$ is located in quantization level 5 as it is shown in Fig. 11, but its own value e=0 was replaced with the e-value of the *Master* i.e. e=1. Thus, the *Slave* obtains the key bit '1' ($e=1\rightarrow$ k_i =d1=1) this is the same value as on the *Master* node. The $RSSI_{910}$ is represented as d1d0e=011 after replacing its own e-value with the one received from the *Master*. The generated key bit is '0' ($e=1\rightarrow$ k_i =d1=0).

The obtained key bit sequences for the RSSI values shown in Fig. 11 are:

Master: 01111111111111100000000000000001111011111
Slave: 01111111101111100000000000000001101111111

The generated bit sequences are not identical: 3 of 40 bits are different (see red marked positions). Moreover, the generated bit sequences contain long sequences of '1' and long sequences of '0'. Another disadvantage is that an additional communication from the *Master* to the *Slave* is required for the e-line exchange after the collection of the RSSI values is finished.

Algorithm 4

This algorithm was presented 2009 in [11] and is an example of quantization methods that try to find a tradeoff between the key generation rate and the probability of errors. It needs more than 128 RSSI values for generating a 128 bit long key. It uses two thresholds q_+ and q_- in order to select RSSI values for generating the key. The thresholds are calculated with the following formula:

$$q_{\pm} = mean \pm \sigma \cdot \alpha. \tag{4}$$

Here *mean* is the mean value of the RSSI values that are used for the key generation. In our experiments we used always 256 RSSI values, i.e. $RSSI_i$ with $0 \le i \le 255$. σ is the standard deviation calculated for this set of RSSI values and α is a factor. In our experiments $\alpha=10\%$ that is similar to experiments in [11] with $\alpha=1/8$.

Corresponding to the algorithm [11] the *Master* selects in this set of $RSSI_i$ values sequences with RSSI values either

above q_+ or below q_-, i.e. those that are not in the interval of [q-; q+], whereby these sequences have to consist of at least m RSSI values. In our experiments $m=4$. The *Master* sends to the *Slave* all indices i of the $RSSI_i$ values which have at least m successors in this sequence. For example, if $RSSI_{904}$ (see Fig. 12) is the first value that is not in the interval [q-; q+] and at least the next 3 RSSI values – i.e. $RSSI_{905}$, $RSSI_{906}$, $RSSI_{907}$ – are also not in the interval [q-; q+], then the index of the first RSSI value in this sequence – here the one of $RSSI_{904}$, i.e. the number 904, will be send to the *Slave*. These values are closely to q-, but they all are smaller than q-, i.e. they are not in interval [q-; q+]. After that the next RSSI value here $RSSI_{905}$ will be examined. The values $RSSI_{906}$, $RSSI_{907}$ and $RSSI_{908}$ are not in the interval [q-; q+]. This means, the index number 905 will be sent to *Slave* too. In the example illustrated in Fig. 12 the *Master* sends all numbers from 904 up to 936, i.e. the set {904, 905, …, 936} to the *Slave*. The $RSSI_{937}$ value does not satisfy the above explained conditions, i.e. its index will not be sent to the *Slave*. In Fig. 12 the RSSI values which indices will not be sent to the *Slave* are shown on the blue background.

The *Slave* receives the set of indices (in our example, the numbers {904, 905, …, 936}) and examines: if m-1 (in our experiments 4-1=3) RSSI values are above q+ or below q- in all sequences of three RSSI values starting with:

{$RSSI_{904}$, $RSSI_{905}$, $RSSI_{906}$},

{$RSSI_{905}$, $RSSI_{906}$, $RSSI_{907}$},

… and so on up to the last sequence

{$RSSI_{936}$, $RSSI_{937}$, $RSSI_{938}$}.

The *Slave* replies to the *Master* with a set of indices i of corresponding sequences. In our experiments the *Slave* sends to the *Master* exactly the same set {904, 905, …, 936}. After this step the *Master* and the *Slave* generate the key from the corresponding RSSI values using the following formula:

$$k_i = \begin{cases} 1, & if \quad RSSI_i > q_+ \\ 0, & if \quad RSSI_i < q_-. \end{cases} \tag{5}$$

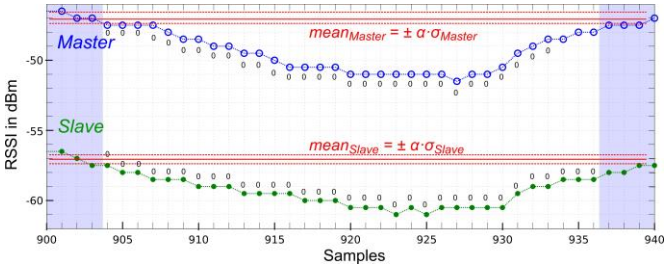


Fig. 12. Illustration of the generation of a part of the key using Algorithm 4 with $m=4$ and $\alpha=0.1$.

For the example shown in Fig. 12 the key bit sequences, obtained by using this algorithm, are 33 bit long:

Master: *000000000000000000000000000000000*
Slave: *000000000000000000000000000000000*

The generated bit sequences are identical but consist of '0' only.

The key generation rate depends significantly on m and α: e.g. if m and α are small, for example $m=4$ and $\alpha=0.1$, a 128 bit long key cannot be generated using even double the number of RSSI values, i.e. 256. Generally, the number of the RSSI values to be collected for generating a key cannot be determined a priori. Algorithm 4 comes with two drawbacks. Firstly, additional information is exchanged to agree on the set of corresponding RSSI values, which requires additional time and energy. Secondly, this exchange is done as plain text which might simplify key guessing attacks.

Algorithm 5

This algorithm was presented 2005 in [9]. To generate an n-bit long key $n+N$ RSSI values are necessary, whereby the number N of additional RSSI values is deterministic. In our experiments we used 256 RSSI values to generate a 128 bit long key, i.e. $n=N=128$. At first the *Master* has to remove $N/2=64$ RSSI values from the set of 256 RSSI values and after that the *Slave* does the same. To do so, the *Master* selects 128/2+64/2 largest and 128/2+64/2 smallest values, i.e. 128+64 "good" values together. The *Master* removes the "bad" RSSI values, i.e. the not selected 64 values and sends their positions (indices) to the *Slave*. Fig. 13 illustrates this algorithm.

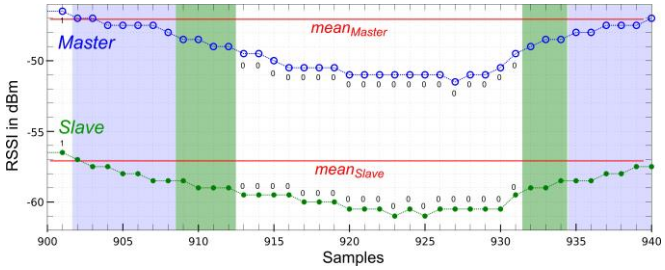


Fig. 13. Illustration of the generation of a part of the key using Algorithm 5.

In the example shown in Fig. 13 the *Master* removes the RSSI values: $RSSI_{902}$, …, $RSSI_{908}$ and $RSSI_{935}$, …, $RSSI_{940}$ and sends the set of their indices: {902, 903, …, 908, 935, 936, …, 940} to the *Slave*. The blue field in Fig. 13 depicts the values which the *Master* selected to be removed.

The *Slave* also removes the RSSI values with the received indices. After this step the *Slave* selects from the remaining RSSI values the 128/2 largest and the 128/2 smallest values, i.e. 128 "good" values. The *Slave* removes the 64 not selected RSSI values and sends their indices to the *Master*. The green background in Fig. 13 depicts the values which the *Slave* selected to be removed. The *Slave* sends the indices 909, 910, 911, 912 and 932, 933, 934. The *Master* also removes the RSSI values with received indices. The RSSI values selected for generating the key are shown in the white field. The *Master* and the *Slave* obtained their keys using the mean value of the remaining RSSI values as threshold (see formula (1)):

Master: *1000000000000000000*
Slave: *1000000000000000000*

These two sequences are identical but two times shorter than the sequences generated by using Algorithm 1, Algorithm 2 or Algorithm 3. Please note that the generated 20 bit long part of the key consists of only one '1' and nineteen '0'. Algorithm 5 requires additional exchange of information about the "bad" RSSI values twice: from the *Master* to the *Slave* and from the *Slave* to the *Master*.

[9] proposed to use the quantization algorithm here described as a part of a new key generation schema. The main contribution of [9] is the idea to increase the fluctuation of the channel characteristics using a beam-forming technique of special antennas. We think, this is a reasonable and promising way to generate the keys with good randomness even in a stationary environment. As we are using standard radio front ends, we cannot evaluate the influence of beamforming on the quality of the generated keys.

V. Evaluation Of The Generated Keys

In this section we discuss two aspects of the generation of the key using channel parameters: the Bit Disagreement Rate (BDR) and the quality of the suitable keys, i.e. of those keys that are identical on both nodes. We also discuss the applicability of the RSSI values measured in an industrial environment for the key generation approaches introduced here.

We generated a relatively small number of keys using our measurement results. The main goal was to evaluate whether or not this approach can be applied in industrial environments. As these settings are requiring real time and short latencies key set-up times of several hours are by default inacceptable. Corresponding to the recent requirements of the automatization industry the time for the agreement of a 128 bit long key has to be in the range of a few milliseconds [17]. Nevertheless we run measurements for about 20 minutes for each of the six cases (case *1a*, *1b* and so on) to get a reasonable number of RSSI values for our evaluation.

Using Algorithm 1, Algorithm 2 or Algorithm 3 and our measured RSSI values it was possible to generate 38 keys per case. Using Algorithm 5 only 19 keys could be generated. The number of the keys which could be generated using Algorithm 4 is not deterministic. In our implementation a maximum of 19 keys could be generated, but for 4 out of 6 measurement cases only a smaller number of keys was generated.

The number of identical keys generated on *Master* and *Slave* nodes in our experiments is really small and depends significantly on the dynamicity of the environment and on the quantization algorithm. For example, using Algorithm 1 only two identical keys in all six measurement cases were generated and no identical keys were generated using Algorithm 2. Algorithm 3-Algorithm 5 give better results, but all the generated identical keys look not random. They have long sequences of '1' or '0'. We assessed their randomness using 5 different tests for randomness monobit, serial, autocorrelation, runs test and compression: (run length encoder). The first four tests are described in NIST [18]. We do not run the complete NIST test [18] because the number of collected RSSI values, required to perform all tests, should exceed 10^6.

A. Quality of generated keys

We analyzed the keys that we obtained for each quantization algorithm. Our proceeding is shown below.

1. For each of the six measurement cases and for each key_l that was generated on the *Master* and the *Slave* nodes, i.e. for all keys not only for identical keys, we calculated the BDR according to the following formula:

$$BDR_l = \frac{length(key_l) - (\#identical\_bits)}{length(key_l)} \cdot 100 \quad \%. \quad (6)$$

   Here $length(key_l)=128$ and *#identical_bits* is the number of identical bits in the key key_i of the *Master* and *Slave* nodes. If the *Master* and the *Slave* generated an identical key key_i, its BDR is 0%.

2. For each of the six measurement cases and for each identical key key_j we calculated its number of '1'-bits as a relative number:

$$\#1_j = \frac{number\_of\_'1'\_in\_key_j}{length(key_j)} \cdot 100 \quad \%. \quad (7)$$

   We use the value $\#1_j$ to demonstrate that the keys generated using channel parameters are not random at least not in our experiments. One of the reasons, why this happens, is the periodical movement of robot arms and of parts of the milling machine. This fact can be a problem for an industrial environment because the manufacturing processes are periodical, i.e. the processing steps are periodically repeated.

3. For each of the six measurement cases we performed 5 randomness tests for the identical keys. The calculated BDR_l and $\#1_j$ are represented graphically in Fig. 16 – Fig. 20.

Table V shows for each investigated quantization algorithm and each measurement case the number of generated keys, their average BDR, and the number of identical keys and the average number of '1' in identical keys.

TABLE V
M*↔*S communication: quality of generated keys assessed by an average BDR and a relative (average) number of '1' bit values.

Measurement case	*Algorithm 1*				*Algorithm 2*				*Algorithm 3*				*Algorithm 4*				*Algorithm 5*			
	generated keys	BDR, %	identical keys	#1, %	generated keys	BDR, %	identical keys	#1, %	generated keys	BDR, %	identical keys	#1, %	generated keys	BDR, %	identical keys	#1, %	generated keys	BDR, %	identical keys	#1, %
1a	38	48	0	-	38	24	0	-	38	0	38	0	-	-	0	-	19	48	0	-
1b	38	26	0	-	38	20	0	-	38	1	32	1	17	3	8	67	19	5	11	50
1c	38	21	0	-	38	21	0	-	38	3	18	27	14	7	5	62	19	2	10	50
2a	38	27	0	-	38	22	0	-	38	5	20	1	8	3	2	36	19	12	7	50
2b	38	13	0	-	38	22	0	-	38	6	12	31	19	2	14	46	19	1	17	50
2c	38	5	2	53	38	24	0	-	38	6	9	26	19	0	18	48	19	0	19	50

Fig. 14 and Fig. 15 illustrate Table V graphically. In Fig. 14 the average BDR is shown for each measurement case and

each quantization algorithm. Fig. 15 shows the number of identical keys.

As it is shown in Table VI we obtained no keys that successfully passed all 5 randomness tests. This means that the generated bit sequences can be not used as cryptographic keys.

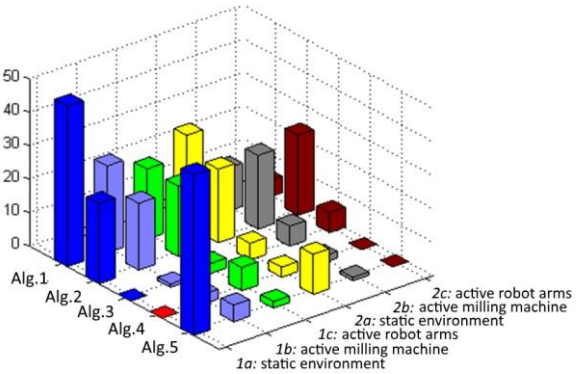


Fig. 14. Average BDR for each case and quantization algorithm. The position marked red for case *1a* and Algorithm 4 represents the fact that no key was generated.

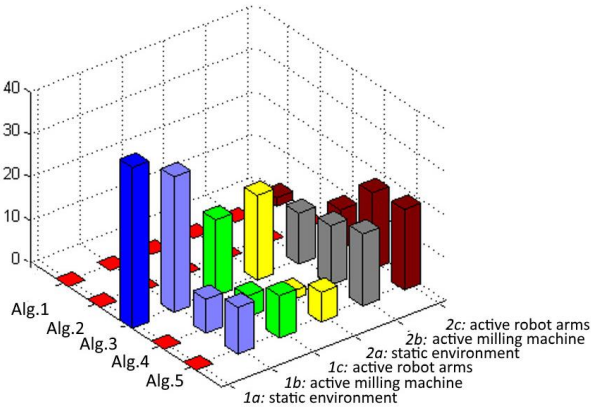
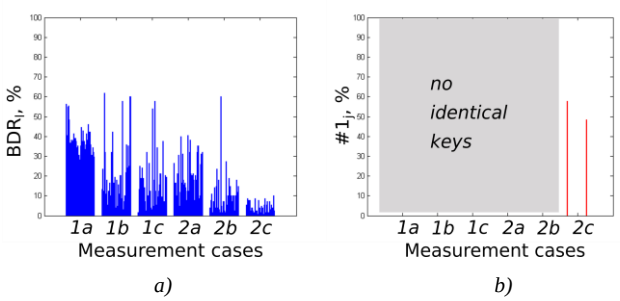


Fig. 15. Number of identical keys for each case and quantization algorithm. The positions marked red represent the fact that no identical keys were generated for: Algorithm 1, measurement cases *1a*, *1b*, *1c*, *2a*, *2b*; Algorithm 2, all measurement cases; Algorithm 4, measurement case *1a* and Algorithm 5, measurement case *1a*.

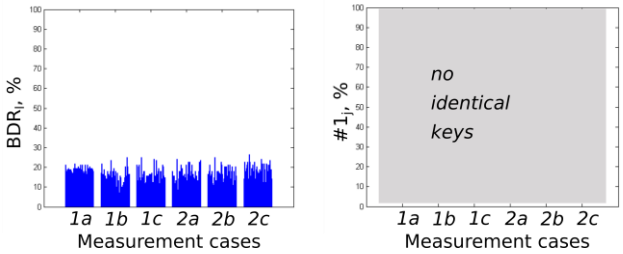
The quality of keys generated using Algorithm 1 is represented in Fig. 16. In each measurement case (i.e. in case *1a*, *1b*, *1c*, *2a*, *2b* and *2c*) we obtained 38 keys on the *Master* and 38 keys on the *Slave* nodes. The BDR for all these keys is shown in Fig. 16-*a)*. Only in case *2c* (robots are active and the *Master* is behind the milling machine) two of the 38 keys were identical, i.e. their bit disagreement rate was zero (BDR=0%). For these identical keys the number of '1' is shown in Fig. 16-*b)*.



a) *b)*

Fig. 16. Quality of keys generated using Algorithm 1 for all six measurement cases. Only 2 keys in case *2c* have a BDR=0%, i.e. they are identical. The number of '1' bit values is close to 50% in both identical keys.

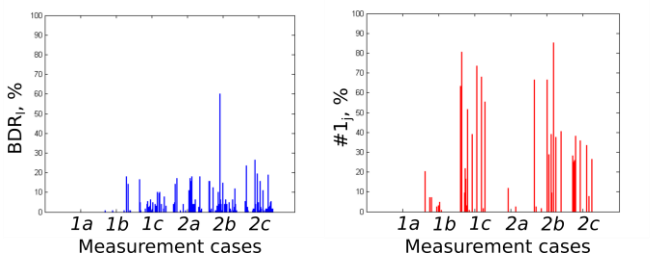
Fig. 17 represents the results of the analysis of the keys generated using Algorithm 2. In each of the measurement cases 38 keys were generated on the *Master* and 38 keys on the *Slave* node. Their BDR is shown in Fig. 17-*a)*. No identical keys are generated using Algorithm 2.



a) *b)*

Fig. 17. Quality of keys generated using Algorithm 2 for all six measurement cases. No identical keys are generated. The average BDR is about 20% in each case.

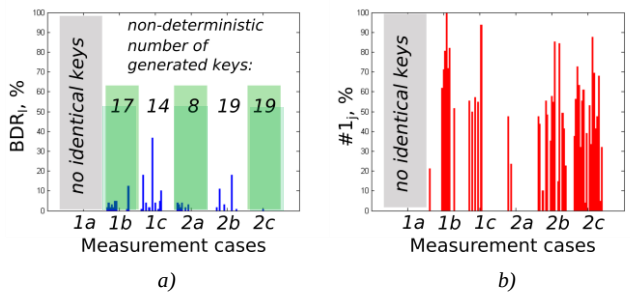
Fig. 18 represents the results of the analysis of the keys generated using Algorithm 3. Also here 38 keys were generated on the *Master* and 38 keys on the *Slave* node in each of our cases. In case *1a* all 38 generated keys are identical, i.e. their BDR is 0% (see Fig. 18-*a*), but they consist of only '0' bit values which can be seen in Fig. 18-*b*. This means, the cryptographic keys generated using Algorithm 3 and RSSI values measured in a static environment are completely unusable. Also in cases *1b* and *2a* many of the generated keys are identical but consist of '0' bits only. For cases *1c*, *2b* and *2c* about half of the generated keys are identical but they contain either a large number of '0' bits or a big number of '1' bits. Only in case *2c* the identical keys consist of about 30% of '1' bit values.



a) *b)*

Fig. 18. Quality of keys generated using Algorithm 3 for all cases. The number of identical keys generated using Algorithm 3 is high for the static environment: 38 keys in case *1a* are identical. Their BDR is 0%, but all these keys consist of '0' bit values only, and in case *2a* 20 keys were identical, but the number of '0' bit values is in average 99%. For non-static cases the BDR increases, but the identical keys still contain either a big number of '0' bits or a big number of '1' bits and are not suitable as cryptographic keys.

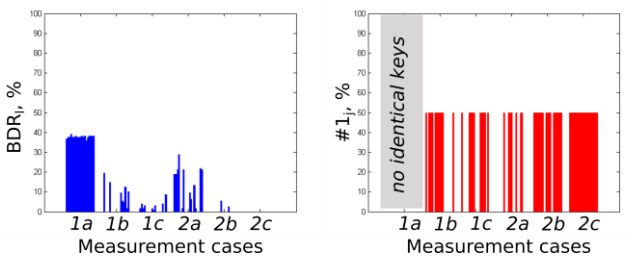
Fig. 19 represents the results of the analysis of the keys generated using Algorithm 4. The number of generated keys is not deterministic and depends significantly on the measurement case. In Fig. 19-*a)* the number of generated keys for each of the measurement case is given. In case *1a*, i.e. in a static environment with the *Master* at position 1 (see Fig. 2) no key was generated even though about 5000 RSSI values were processed. Also in case *2a*, i.e. in a static environment with the *Master* at position 2, only 8 keys were generated and only 2 of these keys are identical. In a mobile environment the BDR of the generated keys is not high and the average number of '1' bit values in those keys is between 46% and 67% (see Table V).



a) *b)*

Fig. 19. Quality of keys generated using Algorithm 4 for all measurement cases. The number of generated keys depends significantly on the environment and is not deterministic.

Fig. 20 represents the results of the analysis of the keys generated using Algorithm 5. For each case 19 keys were generated. The BDR in all cases is not high excepting case *1a*, for that no identical key was generated. The identical keys generated in other cases consist to 50% of '1' bit values.



a) *b)*

BDR of all generated key *The number of '1' in identical keys*

Fig. 20. Quality of keys generated using Algorithm 5 for all cases. The BDR in all cases is not high excepting case *1a*, for that no identical key was generated. The identical keys generated in the other cases consist to 50% of '1' bit values.

Table VI shows the results of different randomness tests that we made to assess the randomness of the generated keys. Only identical keys were examined. The symbol "√" means that for the given measurement case and algorithm the performed randomness test was successful. If a test was not successful it is depicted using '-'. In Table VI cells filled in grey represent the measurement cases, for which no identical keys were generated.

TABLE VI
Tests of randomness for identical keys.

Measurement case	*Algorithm 1*					*Algorithm 3*					*Algorithm 4*					*Algorithm 5*				
	monobit test	serial test	autocorrelation test	runs test	compression test	monobit test	serial test	autocorrelation test	runs test	compression test	monobit test	serial test	autocorrelation test	runs test	compression test	monobit test	serial test	autocorrelation test	runs test	compression test
1a						-	-	-	-	-										
1b						-	-	-	-	-	-	-	-	-	-	√	-	√	-	-
1c						-	-	-	-	-	-	-	-	-	-	√	-	√	-	-
2a						-	-	-	-	-	-	-	-	-	-	√	-	√	-	-
2b						-	-	-	-	-	-	-	√	-	-	√	-	√	-	-
2c	√	-	√	-	-	-	-	-	-	-	-	-	√	-	√	√	-	√	-	-

B. *Randomness of measured channel parameters*

Due to the fact that no algorithm in all our measurement cases succeeded in all randomness tests, we decided to examine the randomness of the measured RSSI values. We performed the same randomness tests for RSSI values collected on the *Master* node and on the *Slave* node. Table VII shows the results of the tests for our six measurement cases. Please note, only in one of these cases the RSSI values pass three out of five randomness tests (see Table VII, measurement case *1a*, RSSI values collected on the *Slave*).

TABLE VII
Tests of randomness for measured channel parameters

Measurement case	RSSI values collected on the *Master*					RSSI values collected on the *Slave*				
	monobit test	serial test	autocorrelation test	runs test	compression test	monobit test	serial test	autocorrelation test	runs test	compression test
1a	-	-	√	-	√	√	-	√	-	√
1b	-	-	-	-	√	-	-	-	-	√
1c	-	-	-	-	√	-	-	√	-	√
2a	-	-	-	-	√	-	-	-	-	√
2b	-	-	-	-	√	-	-	√	-	√
2c	-	-	√	-	√	-	-	√	-	√

C. Quality of listen by malicious nodes

To evaluate the quality of the keys extracted by malicious nodes we analyzed the quality of the keys generated by them based on the RSSI values retrieved from the communications they eavesdropped.

- We generated keys based on the communication from the *Master* to the *Slave* eavesdropped by *Eve1*.
- We compared the keys generated on the *Eve1* and on the *Slave* nodes using their BDR.

$$BDR_{E1,Slave:\ generated\ keys} = \frac{1}{\#k_{E1}} \sum_{i=1}^{\#k_{E1}} BDR_i, \tag{8}$$

where $\#k_{E1}$ is the number generated at Eve1.

- We compared only the keys that are identical on the *Master* and the *Slave* nodes with the corresponding keys extracted on *Eve1*.

$$BDR_{E1,Slave:\ MS\ identical} = \frac{1}{\#k_{MS\_id}} \sum_{i=1}^{\#k_{MS\_id}} BDR_i, \tag{9}$$

where $\#k_{MS\_id}$ is the number of identical keys of *Master* and *Slave* nodes.

The calculated average BDR values are given in Table VIII.

In this experiment it was expected that the correlation of RSSI values measured on the *Slave* and *Eve1* nodes is high because the node *Eve1* was placed relative closely (10 cm that is ≈ *λ/3*) to the *Slave*. Thus, we expected that the keys generated on the *Eve1* and on the *Slave* node are similar and that their BDR is small. Our results show that only for Algorithm 3 and the static case the average BDR is about 4%. Important is that the keys generated using this algorithm either contain a long sequence of '0' bit values or consist of '0' bit values only, see Fig. 18-*b*, case *1a* and case *2a*).

TABLE VIII
BDR of keys generated on malicious nodes, %

Measurement case		Communication *M→S* (comparison with keys of the *Master*)				Communication *S→M* (comparison with keys of the *Slave*)			
		Eve1		*Eve2*		*Eve1*		*Eve2*	
		$BDR_{E1,Slave:\ generated\ keys}$	$BDR_{E1,Slave:\ MS\ identical}$	$BDR_{E2,Slave:\ generated\ keys}$	$BDR_{E2,Slave:\ MS\ identical}$	$BDR_{E1,Slave:\ generated\ keys}$	$BDR_{E1,Slave:\ MS\ identical}$	$BDR_{E2,Slave:\ generated\ keys}$	$BDR_{E2,Slave:\ MS\ identical}$
Algorithm 1	*1a*	50		50		65		2	
	1b	49		37		47		55	
	1c	56		59		53		53	
	2a	49		52		50		50	
	2b	70		68		49		49	
	2c	26	29	49	45	47	40	50	52
Algorithm 2	*1a*	29		32		23		2	
	1b	22		21		18		20	
	1c	22		21		23		16	
	2a	21		21		15		24	
	2b	31		24		27		21	
	2c	27		26		27		20	
Algorithm 3	*1a*	4	4	6	6	0	0	0	0
	1b	36	35	10	11	3	1	3	1
	1c	62	69	52	45	31	27	31	27
	2a	4	3	7	3	5	1	6	1
	2b	67	68	45	42	36	31	35	31
	2c	26	23	51	56	27	26	27	26
Algorithm 4	*1a*	50							
	1b	49	54	58	21	11	47	53	66
	1c	56	54	49	76	72	54	40	54
	2a	49	69	78	73	69	69	64	64
	2b	70	73	79	66	71	51	51	49
	2c	26	25	25	51	53	49	50	52
Algorithm 5	*1a*	50		50		66		3	
	1b	54	63	28	20	49	47	61	60
	1c	59	56	80	78	50	50	53	54
	2a	56	56	58	56	49	50	50	50
	2b	84	87	75	76	49	50	51	51
	2c	14	14	53	53	49	49	49	49

The second lowest BDR of 14% was retrieved for the keys generated using Algorithm 5 and the RSSI values measured in case *2c*. In this case all keys generated on the *Master* and the *Slave* are identical. They could be used as shared secret (see red filled cells in Table VIII) except for the fact, that they are no random bit sequences. Fig. 21 gives an overview of the BDRs for all individual keys for this case (see red bars).

We made the same experiments as described above for the RSSI values measured on the *Eve2* node. The calculated BDR values are shown in Table VIII. In these experiments it was expected that the correlation of RSSI values measured on the *Slave* and *Eve2* nodes is low because the node *Eve2* is placed at a distance of 80cm from the *Slave* that is more than *2λ*. Thus, the expected BDR should be high in comparison to the BDR values of *Eve1* for the same measurement set-up (compare the blue filled cells in Table VIII with the red filled cells). For *Eve1* the BDR values for each key for the measurement case *2c* and quantization Algorithm 5 are shown as red bars and for Eve2 these values are shown as blue bars in Fig. 21.

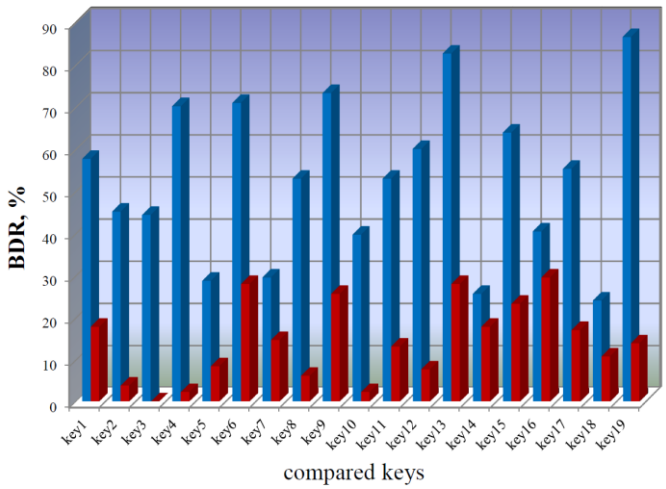


Fig. 21. BDR of keys generated on *Eve1* and *Eve2* in comparison to the *Slave* node for measurement case *2c* using Algorithm 5. The malicious nodes generated the keys using RSSI values of the communication eavesdropped from the *Master* to the *Slave*. The BDR values for keys generated by *Eve1* (compared to the keys that are generated by *Slave*) are shown as blue bars. Their average value is about 53%. The BDR values for keys generated by *Eve2* (compared to the keys that are generated by *Slave*) are shown as red bars. Their average value is 14%.

Table VIII shows also the results of our experiments using the RSSI values from the communication from the *Slave* to the *Master* collected on both malicious nodes. In these experiments it was expected that the keys extracted on *Eve1* and *Eve2* nodes differ significantly from the keys generated on the *Master* node, because the distance between the *Master* and each of the malicious nodes is about 9 m which is are significantly larger than the wave length. For Algorithm 3 in the measurement cases *1a*, *1b* and *2a* we obtained unexpected small BDR values between 0% and 6% (see green filled cells in Table VIII).

VI. A Comparison Of Key Agreement Methods

In this section we compare the key generation approaches using channel parameters with a classical key agreement method based on ECC and discuss their applicability in industrial environments.

For the comparison we selected the most efficient version of the channel parameter based key generation i.e. Algorithm 5 in measurement case *2c*. For the rest of this evaluation we do not consider the quality of the keys used by the different key establishment approaches. In other words we neglect the fact that the randomness keys generated from channel parameters is low and focus on the efficiency of the key setup methods only.

From the ECC based approaches we selected an MTI-based method because it leads to the highest computational burden. Our goal is to estimate the maximum time and energy needed by the ECC based approaches.

The time needed for the generation of a key using channel parameters and Algorithm 5 for the quantization of the collected RSSI values can be estimated based on our experiments as follows: we sent 12000 request-answer signals, but only 5000 request-answer pairs were collected and 19 identical 128 bit long keys were generated. As we mentioned above, we collected the RSSI values for about 20 minutes. Thus, the time for generating one key can be estimated as $time > time_{measurements}/19 \approx 20min/19 \approx 1\ min$. Because this key generation time estimated using real measurements is to long to be applicable for automation industry, we use instead a best case assumption for the evaluations. Some steps of the key generation approach using channel parameters and the ECC-based approach need calculation time and energy for sending and receiving packets, determined as follows:

$$time = t_{overhead} + n \cdot 26\mu s$$
$$energy = P \cdot time. \quad (10)$$

Here $t_{overhead} = 1694\mu s$; n is the number of bits in the data field of a packet. The sending power is $P = 46.2mW$ and the receiving power is $P = 23.1mW$ corresponding to [14].

The calculation of the time and energy consumption of both approaches is shown step by step in Table IX. Also the result of our estimations is summarized in Table IX: the ECC based approach needs 0.04 s only to exchange keys whereas the approach using channel parameters needs 2.7 s which is about 70 times slower.

The analysis presented in this section shows that even in the optimal case the key generation cannot meet the timing requirements for key exchange or key set-up as required at least for a part of the automation cases. Even worse, the key generation based on channel parameters may take unpredictable long times even up to several days [19]. This fact is inacceptable in automation systems that require real time.

Our comparison of the key distribution approaches clearly indicates that standard key agreement protocols such as the ECC based ones outperform the key generation from channel parameters. This holds not only true for the effort i.e. time and energy needed to agree on a shared key but also for the quality of the keys. The keys we generated were not random which might allow a potential attacker to guess the keys and even can relatively easy manipulate the keys e.g. by changing the environment.

The ECC based approaches provide in addition to the key exchange a means to properly authenticate the sender of a certain message which is infeasible if the keys are generated from channel parameters without any additional means.

TABLE IX
Comparison of the time and energy needed to establish a common key

Required steps		*MTI(A0)*		*Algorithm 5 for quantization of RSSI values*	
		Alice	**B**ob	**A**lice	**B**ob
generation of a random[1] number		233 bits	233 bits	not required	not required
	time	3.7 ms	3.7 ms	0 ms	0 ms
	energy	20 µJ	20 µJ	0 µJ	0 µJ
sent information		234 bits	234 bits	1)256 *probe* signals, 2)Set of 64 indices; each index is a .1 byte long 3)256 bits hash	1)256 *probe* signals, 2)Set of 64 indices; each index is 1 byte long
	time	7.8 ms	7.8 ms	646.7+15.0+8.3=670 ms	1646.7+15.0=661.7 ms
	energy	360 µJ	360 µJ	46.2mW·670ms=30954µJ	46.2 mW·661.7 ms= 30572 µJ
received information		234 bits	234 bits	1)256 *probe* signals 2)Set of 64 indices; each index is 1 byte long	1)256 *probe* signals, 2)Set of 64 indices; each index is 1 byte long 3)256 bits hash
	time	7.8 ms	7.8 ms	646.7+15.0=661.7ms	646.7+15.0+8.3=670 ms
	energy	23.1mW 7.8ms = 180 µJ	180 µJ	23.1mW·661.7ms=15285µJ	23.1 mW·670 ms= 15477 µJ
computation burden		3 *kP*	3 *kP*	1)Processing of 256 RSSI values corresponding to Algorithm 5 2) calculating hash	1) Processing of 256 RSSI values corresponding to Algorithm 5 2) calculating hash 3)comparison of the received and the calculated hash
	time	3·0.3ms=0.9ms	0.9 ms	Assumed: 0ms	Assumed: 0ms
	energy	3·2 µJ=6 µJ	6 µJ	0 µJ	0 µJ
Summed up	time	20.2 ms	20.2 ms	1331.7 ms	1331.7 ms
	energy	566 µJ	566 µJ	46239 µJ	46049 µJ
	on both nodes together	40.4 ms≈0.04 s 1132 µJ≈1.1 mJ		2663.4 ms≈2.7 s 92288 µJ≈92.3 mJ	

VII. CONCLUSION

The generation of a common secret key out of channel parameters like RSSI is a relatively new field of research and gets quite a lot of attention as it promises to avoid heavy processing as it is required when asymmetric cryptographic approaches are applied for key exchange. This is very appealing whenever resource constraint devices are used. In this paper we investigated the applicability of this new approach for the field of automation systems using channel parameters i.e. RSSI values recorded in a real industrial environment.

One of the lessons we learned is that the duration of the key generation based on channel parameters is not deterministic and may take a long time. This also means that it requires a significant amount of energy as messages need to be exchanged to have channel parameters. So, this approach is not suitable for resource constraint devices. It is also not applicable in real time systems in which the duration of a certain operation needs to be deterministic and even very fast sometimes. Based on this experience we started a comparison of the key generation approaches with classical key establishment approaches. The result is that the latter outperform the key generation approaches even if we consider the worst case for classical approaches and the best case for the key generation approaches using channel parameters.

The second lesson we learned is that none of algorithms tested here was able to generate truly random common keys, which makes the keys more vulnerable against attacks like brute-forcing. In contrast to that classical key exchange approaches allow the use of well tested random number generators which in turn lead to high quality keys.

We admit that we applied execution times of special hardware providing cryptographic operations when we compared the energy consumption of the different approaches, but as this hardware is as small as 0.3 mm^2 and by that will cost most probably less than 20 cent we are sure that for automation systems such a solution is feasible.

We also like to point out that our findings are based on measurements done at a frequency of 868 MHz and with standard RSSI values. This means that at a different frequency and or when using other or more channel parameters the key generation approaches may provide better results. We will run the experiments described here again using a radio operating at 5.4 GHz and proving us with 5-8 values from its preamble.

[1] The time and energy consumed while generating random numbers on a MSP430F5438A [20] at 8 MHz. The operations needed to generate these numbers are described in [21]. The energy consumption of the operations was calculated according to the procedure described in [22]. The exact values used for this calculation were taken from [20].

ACKNOWLEDGEMENT

The work presented here was partly supported by the German Ministry of Research and Education (BMBF) within the ParSec project, grant agreement no. 16KIS0219K.

REFERENCES

[1] Model Factory of the Innovation Centre Modern Industry Brandenburg, Chair of Automation Technology BTU Cottbus-Senftenberg ,Oliver Stecklina, http://www.imi4bb.de/TransferI4/#transf-i4-functional.area

[2] Federal Information Processing Standards Publication 197, "Announcing the ADVANCED ENCRYPTION STANDARD (AES)", (November 2001), http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

[3] W. Diffie and M. Hellman, "New directions in cryptography", IEEE Trans. Inf. Theor. 22, Issue 6, 644-654, 1976. http://dx.doi.org/10.1109/TIT.1976.1055638

[4] NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography", http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.pdf

[5] T. Matsumoto, Y. Takashima and H. Imai " On Seeking Smart Public-key Distribution Systems". In Transactions of the IECE of Japan, E69, pp. 99- 106, 1986.

[6] L. Law, A. Menzenes, M. Qu, J. Solinas, and S. Vanstone "An Efficient Protocol for Authenticated Key Agreement". In Designs, Codes and Cryptography, 2003, 28, pages 119-134.

[7] H. Liu, Y. Wang, J. Yang, and Y. Chen, "Fast and Practical Secret Key Extraction by Exploiting Channel Response" In Proc. INFOCOM 2013, Turin, Italy, pages 3048-3056, 2013.

[8] J. Huang, and T. Jiang "Dynamic Secret Key Generation Exploiting Ultra-wideband Wireless Channel Characteristics" in Proc. WCNC 2015, New Orleans, USA, pages 1701-1706, 2015.

[9] T. Aono and K. Higuchi and M. Taromaru and T. Ohira and H. Sasaoka}, "Wireless secret key generation exploiting the reactance-domain scalar response of multipath fading channels", in Proc. EuMA, Paris, France, pp. 173-176, 2005.

[10] N. Patwari, J. Croft, S. Jana, and S. K. Kasera, "High rate uncorrelated bit extraction for shared secret key generation from channel measurements," IEEE Transactions on Mobile Computing, vol. 9, no. 1, pp. 17–30, Jan. 2010. Published, 2010.

[11] Mathur, Suhas and Trappe, Wade and Mandayam, Narayan and Ye, Chunxuan and Reznik, Alex, "Radio-telepathy: Extracting a Secret Key from an Unauthenticated Wireless Channel", in Proc. MobiCom 2008, San Francisco, USA, pp. 128-139, 2008.

[12] Zenger, Christian and Zimmer, Jan and Paar, Christof, "Security Analysis of Quantization Schemes for Channel-based Key Extraction", Proc. MOBIQUITOUS, Coimbra, Portugal, pp. 267-272, 2015.
[13] E. Alpirez Bock, master's thesis "SCA resistent implementation of the Montgomery kP-algorithm", BTU Cottbus, 2015
[14] Manual: "Low-power Sub-1 GHz RF Transceiver", http://www.ti.com/lit/ds/symlink/cc1101.pdf
[15] Krzysztof Piotrowski, Anna Sojka-Piotrowska, "IHPNode – the experimental platform for wireless sensor networks and Internet of Things," In Proc: SP 2016, Zielona Góra , Poland, 2016, pp. 121—124, ISBN:9788378422440.
[16] Suman Jana, Sriram Nandha Premnath, Mike Clark, Sneha K. Kasera, Neal Patwari, and Srikanth V. Krishnamurthy, "On the effectiveness of secret key extraction from wireless signal strength in real environments.", in Proc. MobiCom '09, New York, USA, pp. 321-332, 2009.
[17] PROFIBUS & PROFINET International (PI): http://www.profibus.org.pl/index.php?option=com_docman&task=doc_view&gid=28
[18] Barker, E., Kelsey, J.: NIST Draft Special Publication 800-90 B: Recommendation for the Entropy Sources Used for Random Bit Generation (August 2012), http://csrc.nist.gov/publications/drafts/800-90/draft-sp800-90b.pdf
[19] C. T. Zenger, A. A. Ambekar, F. Winzer, T. Pöppelmann, H. D. Schotten, and C. Paar, " Preventing Scaling of Successful Attacks: A Cross-Layer Security Architecture for Resource-Constrained Platforms" in Proc. BalkanCryptSec 2014, pp. 103-120, 2015.
[20] Texas Instruments, MSP430F5438A Mixed Signal Microcontroller Datasheet, available at: http://www.ti.com/lit/gpn/msp430f5438a
[21] A. Soijka-Piotrowska, PhD Thesis "On the applicability of short key asymmetric cryptography in low power wireless sensor networks", BTU Cottbus, 2016.
[22] Krzysztof Piotrowski, Peter Langendoerfer, and Steffen Peter., "How public key cryptography influences wireless sensor node lifetime", in Proc. SASN '06, New York, USA, pp. 169-176, 2006.
[23] ParSec: BMBF project „IKT 2020 – Forschung für Innovationen", „Ein paralleles (Par) zuverlässiges und sicheres (Sec) Funksystem zur latenz-optimierten Fabrikautomatisierung, http://www.parsec-projekt.de

Dr. Dan Kreiser received his Diploma in computer science from the Humboldt University of Berlin in 2008 and the Dr.-Ing. degree in computer science from the Brandenburg University of Technology Cottbus - Senftenberg in 2016. Since 2008 he is with the sensor networks and mobile middleware group at IHP in Frankfurt (Oder).
His research interests include the development of hardware implementations for UWB-systems (especially localization and communication) and for elliptic curve cryptography (key exchange and key generation).

Dr. Zoya Dyka holds a diploma in radio-physics from the Taras Shevchenko University Kiew in 1996 and received her doctorate degree from the BTU Cottbus in 2012. Since 2013 Zoya is leading a young researchers group in the field of tamper resistant crypto ICs. She has published more than 15 refereed technical articles and filed four patents in the security area of which two are granted. She has served as reviewer for highly selective journals such as IEEE Transactions on Computers, and was organization chair of the IEEE ISSSE and IEEE ASYNC conferences. Her research interests include design of efficient hardware accelerators for crypto-operations and anti-tampering means.

Stephan Kornemann received his M.Sc. degree in information and media technology from the Brandenburg University of Technology Cottbus - Senftenberg, in 2012. From 2010 to 2012 he worked as software tester at Philotech and received his software testing qualification certificate from ISTQB. Since 2012 he is member of the sensor networks and mobile middleware group, at IHP in Frankfurt (Oder). His current research focuses on intrusion detection systems for wireless sensor networks.

Christian Wittke received his M.Sc. degree in information and media technology from the Brandenburg University of Technology Cottbus - Senftenberg, in 2013. Since 2013 he is member of the sensor networks and mobile middleware group at IHP in Frankfurt (Oder), Germany. He is currently pursuing the Ph.D. degree in computer science. His research interests include the development of hardware implementations for elliptic curve cryptography and their resistance to side channel analysis attacks.

Ievgen Kabin received his B.Sc. degree in electronics engineering in 2007 and his Specialist degree in "Electronic Systems" from the National Technical University of Ukraine "Kyiv Polytechnic Institute", Kyiv Ukraine in 2009.
From 2009 to 2010, he was a Leading Engineer in the state-owned enterprise "Scientific Production Center of Energy-efficient Designs and Technologies "Tehnoluch"; from 2010 to 2015 - Junior Researcher in E.O. Paton Electric Welding Institute of the National Academy of Sciences of Ukraine. Since 2015 he is member of the sensor networks and mobile middleware group, at IHP Frankfurt-Oder (Germany). His research interests include investigations in the field of elliptic curve cryptography and magnetron sputtering of carbon thin films. He holds two patents.

Prof. Dr. Oliver Stecklina received his Diploma in computer science from the Brandenburg University of Technology Cottbus in 2003 and his Dr. rer. nat. degree in computer science from the Brandenburg University of Technology Cottbus - Senftenberg in 2016. From 2003 to 2009, he was a Consultant and Senior Consultant with the secunet Security Networks AG. From 2009 to 2015, he was a member of the sensor networks and mobile middleware group, at IHP Frankfurt-Oder (Germany). From 2015 till 2017, he was a Research Assistant with the Brandenburg University of

Technology Cottbus - Senftenberg. Since 2017 he is a Professor at the University of Applied Sciences at Lübeck. He is author of more the 20 articles. His research interests include security in deeply embedded systems and low power system architectures. He published the langOS operation system for MSP430 based platforms and the tiny scale soft-core processor tinyVLIW8. He was recipient of the Best Student Paper Award of the 8th BSI Security Congress in 2003.

Prof. Dr. Peter Langendörfer holds a diploma and a doctorate degree in computer science. Since 2000 he is with the IHP in Frankfurt (Oder). There, he is leading the sensor networks and mobile middleware group. Since 2012 he has his own chair for security in pervasive systems at the Technical University of Cottbus. He has published more than 110 refereed technical articles, filed ten patents in the security/privacy area of which 5 have been granted already. He worked as guest editor for many renowned journals e.g. Wireless Communications and Mobile Computing (Wiley). Peter is highly involved in EU projects especially in the following security related EU projects UbiSecSens (WP lead), WSAN4CIP (tech mngr), TAMPRES (coord.), SMARTIE (coord.). He was chairing international conferences such as WWIC and has served in many TPCs for example at Globecom, VTC, ICC, SECON and Trusted. His research interests include wireless sensor networks and cyber physical systems, especially privacy and security issues.