Innovative Key Generation Approach to Encrypt Wireless Communication in Personal Area Networks

Daniel Bichler and Guido Stromberg
Infineon Technologies AG
Germany
Email: daniel.bichler@infineon.com
guido.stromberg@infineon.com

Mario Huemer
University of Klagenfurt
Austria
Email: mario.huemer@uni-klu.ac.at

Abstract—In this paper we present a signal processing methodology for sharing symmetric keys in personal area networks. Symmetric encryption and decryption are commonly used because of limitations in computing power and energy consumption. However, key sharing still imposes challenges regarding usability, computational complexity of algebraic key exchange algorithms, and security. Our approach is that keys are generated locally on devices by shaking them, and that the keys are equal if and only if the devices are shaken together. Based on practical assessments, we show that the key generation algorithm is able to generate keys from acceleration data with an average entropy of 13bit/key in 70% of the cases.

I. INTRODUCTION

Security and privacy are key issues in wireless network environments. Personal Area Networks (PAN) however, impose further requirements on the security mechanisms than to keep personal information private. Formost, the security mechanisms need to be implemented without confusing the usability of the networked devices. A further restriction arises because most devices in such applications are small, battery powered, and have little computational power, making complex algorithms for encryption or key exchange like public and private key methods [1] unfeasible. Thus, we constrain our discussion on systems that rely on symmetric encryption methods.

However, the level of security and privacy in PANs depends highly on the application [2]. In systems utilizing symmetric encoding and decoding, usually one of the biggest challenges is to make sure the devices that are allowed to communicate securely have the same symmetric key. In state-of-the-art systems, this is conducted by key exchange methods, which are either manual (e.g. typing in the key in a keypad) or which exploit algebraic key-exchange algorithms [3].

In this paper, we present and evaluate an alternative to exchanging keys by generating a cryptographic key locally from exposing devices to common physical environmental conditions. In particular, we consider two devices to be shaken together and use the recorded acceleration samples to generate a local key on both devices. Shaking devices together enhances the usability of small devices which we are able to carry in one hand. Furthermore, external hardware components (e.g. a wire to physically connect two devices together) are needless to establish a secure communication between the devices.

As the Bluetooth application space is quite typical in respect of the required level of security for PANs, our goal is to develop a symmetric key which is equivalently strong as the Bluetooth PIN code. Typically, the Bluetooth PIN consists of three to four digits which range from 0 to 9 [4]. This corresponds to a key-strength of about 10 to 13 bit.

Although we believe that the local generation of encryption keys from acceleration measurements is novel, there exist several approaches for key exchange in symmetric key encryption systems. Most commonly known, Diffie and Hellman (DH) proposed an algorithm to securely distribute a symmetric key between two parties [5]. The vulnerabilities of the so-called DH key exchange algorithm are carefully described in [6]. However, especially due to its computational effort, the DH algorithm is not suitable for pervasive devices. Alternatively, the additional computational effort of complex distribution algorithms can be reduced by pre-distributing keys during an initialization phase. Afterwards, these keys can be used to encrypt and decrypt subsequent communication or to securely exchange additional symmetric keys [7]. This method however, increases the initial configuration effort and complicates the usability of each device.

The prior work that is probably closest related to ours is conducted within the framework of the Smart-Its project, in which devices in their direct surrounding are grouped using their context of proximity [8], e.g. by measuring the acceleration while moving the device. The acceleration data is broadcasted to all devices inside the wireless range. If the similarity between the received acceleration data and the measured acceleration reaches a certain threshold, the devices assume that they have been moved together and hence will accept a connection. However, our prime objective is not to exchange the acceleration characteristics, but to generate unique keys locally which are kept secret on each device. Therefore, the signal processing methods developed in the Smart-Its project are not appropriate for key generation.

Thus, the system must be designed so that the devices create exactly the same symmetric key by their own if and only if they are shaken together. To this end, a 3D acceleration sensor is used to record the motion of the devices in each direction during a shaking process. This allows us to position the sensors held in the hand arbitrarily on each device. The symmetric
key is generated out of the recorded shaking process of the acceleration sensor using signal processing methods which we will disclose during the course of this paper. Shaking devices together is very user friendly and practical especial for small, mobile, battery powered personal devices.

This paper is organized as follows: In Sec. II we introduce our hardware prototypes for recording the shaking processes and for transmitting the acceleration data to the personal computer for off-line signal processing. In Sec. III, we propose and assess the key generation algorithm which aims at generating exactly the same key on both devices if and only if they are shaken together. Sec. IV characterizes the synchronization algorithm which is used to minimize the time displacement between the starting point of the shaking processes. We conclude the paper in Sec. V and summarize the essential results of the key generation algorithm and the synchronization.

II. DATA ACQUISITION

Let us start with considering two devices shaken in one hand. Typically, the shaking process consists of fast up-and-down movements in the three dimensional space. To monitor motions, we have built prototypes with 3D acceleration sensors. The value of the acceleration sensors are 10bit A/D converted with a sampling rate of 200Hz using an off-the-shelf 16 bit micro controller. The sampled data is transmitted via serial line to a personal computer, on which we perform off-line signal processing using Matlab. At a later stage, the Matlab algorithm will be ported onto the embedded controller, so that the prototypes operate truly independent.

All 3-D data is filtered with a first order low-pass filter with cut-off frequency of 100Hz. In order to reduce the influence of the relative alignment of the acceleration sensors of the different devices, we compute the absolute values of the acceleration vectors. The shaking is usually along a fixed axis as we have validated in some initial experiments, so that we loose only a small fraction of information about the shaking process. Thus, we can actually view the shaking process as a one-dimensional oscillation.

Our generated test data consists of the ensemble \( \tilde{E} = (\tilde{A}, \tilde{B}) \) of \( S = 88 \) recorded shaking processes, where each shaking process consists of two acceleration sequences; \( \tilde{A} \) includes all shaking sequences from the prototype device A and \( \tilde{B} \) from the prototype device B. Note that the two prototypes are unsynchronized, and that none of the sequences are aligned. We denote \( \tilde{a}_n^A \in \tilde{A} \) and \( \tilde{a}_n^B \in \tilde{B} \) as the unsynchronized sequences of prototype A and B, where \( n \) is the index of our shaking experiment. In the following section, we will present a key generation algorithm and analyze its performance considering arbitrary displacements between the sequences as a result of imperfections of the synchronization algorithm. We thus define \( E = (A, B) \) as the synchronized shaking processes; \( A \) includes all synchronized shaking sequences from the prototype device A and \( B \) from the prototype device B. Correspondingly, we denote \( a_n^A \in A \) and \( b_n^B \in B \) as the synchronized sequences of prototype A and B. We limit the duration of the synchronized shaking process to 5 seconds which yields sequences of 1000 samples. \( \hat{a}_n^A \in \tilde{A} \) and \( \hat{b}_n^B \in \tilde{B} \) denote the zero mean and of unit energy versions of \( a_n^A \) and \( b_n^B \), respectively.

III. Key Generation

Our objective is to generate exactly the same cryptographic key from two shaking sequences obtained by two independent hardware prototypes if and only if they are shaken together. Due to the fact that both sequences are not identical the key generation algorithm must have the ability to map similar sequences to the same key and sequences with less similarity to different keys. We further assume that the synchronization between the shaken devices is non-ideal. In this section, we therefore present a key generation algorithm and estimate its quality as a function of the displacement between two shaking sequences. The same key generation algorithm is applied on both shaking sequences of the hardware prototypes without any interaction. At the beginning of this section, we concentrate on explaining the key generation in detail on the basis of the shaking sequences from prototype A.

Due to the high similarity of the frequency spectra between the shaking processes, key generation based on the time domain is assumed to be more beneficial [9]. Thus, our approach for generating a cryptographic key \( k^A_{n,i} \) is that we start with splitting the shaking sequence \( \hat{a}_n^A \) into \( I \) segments \( \hat{a}_{n,i}^A \) of constant length \( L \) to avoid processing the complete sequence as a whole. In this section \( \hat{a}_{n,i}^A \) represents the \( i \)-th segment of the \( n \)-th shaking experiment of \( A \). Then, we calculate from the segment \( \hat{a}_{n,i}^A \) a fragment of the cryptographic key \( k_{n,i}^A \). At the end, the key \( k^A_n \) is constructed by the concatenation of \( I \) fragments.

The calculation of the keys’ fragments is done in two steps as shown in Fig. 1. First, we reduce the dimensionality of the segments \( \hat{a}_{n,i}^A \). The objective herein is to focus on the main attributes of all segments to the key generation algorithm, to remove outlier components of our measurement, and to reduce memory resources for the implementation of the key generation algorithm. Commonly, the segments can be represented by a weighted sum of patterns which consist of common components of all segments from the test data. The weights indicate the similarity between the segment \( \hat{a}_{n,i}^A \) and

![Fig. 1: Correlation of segments with patterns and subsequent hash-function](image-url)
used for the representation of more than 95% of its signal energy. Note that the patterns $\hat{r}_{n,q} \in \hat{A}$ correspond to the same key fragments $\hat{A}_{n,q}$, which results in a relatively small half-space. Additionally, the graph of the Eigenvalues indicates that 5 corresponding Eigenvalues of the test set when the shaking vectors $\hat{A}_{n,i}$ are assigned to their mean vectors until reached. 

Second, in the quantization phase, the $\hat{d}^{A}_{n,i}$ are assigned to the closest $\hat{r}_{n,q}$ and the corresponding key fragment

$$k_{n,i}^{A} = \arg\min_{q=1,...,Q} ||\hat{d}^{A}_{n,i} - \hat{r}_{n,q}^{A}||$$ (1)

is generated. Thus, the index $q$ of the $\hat{r}_{n,q}^{A}$ which is closest to the $\hat{d}^{A}_{n,i}$ is assigned to the $i$th fragment of the key $k_{n,i}^{A}$. Similar $\hat{d}^{A}_{n,i}$ are therefore assigned to the same $\hat{r}_{n,q}^{A}$ and result in the same key fragment. Note that both the training step as well as the quantization step are performed on the same data sequences $\hat{g}_{n,i}^{A}$. Thus, in contrast to the patterns $\hat{r}_{m}$, which are fixed for all our experiments, the $\hat{r}_{n,q}^{A}$ are computed individually for each sequence $\hat{g}_{n}$. 

The same procedure explained for prototype A is also applied to the prototype B. To generate exactly the same cryptographic key fragments $k_{n,i}^{A}, k_{n,i}^{B}$, from the weight vectors of the two shaking sequences $\hat{d}^{A}_{n,i}, \hat{d}^{B}_{n,i}$, the PNN algorithm must assign the $\hat{d}^{A}_{n,i}$ to the same key fragments $k_{n,i}^{A}$ and $k_{n,i}^{B}$. Otherwise, we define that the key generation has failed because not all key fragments $k_{n,i}^{A}$ and $k_{n,i}^{B}$ are equal.

We express the quality of the key generation algorithm by the fraction of successful cases for which $k_{n,i}^{A} = k_{n,i}^{B}$ in relation to the total number of experiments, which we display as a function of the average entropy of the successfully created keys. Additionally, to analyze the influence of the time displacement between the shaking sequences on the quality of the key generation algorithm, we intentionally delay the starting point of the shaking sequences $\hat{g}_{n}$ by $	au = 0,\ldots,5$ samples ($\tau = 0$ indicates genie aided synchronization obtained from the peak of the cross-correlation function of the two sequences). This information is used to formulate a metric for the synchronization algorithm that we will present in Sec. IV.

For estimating the quality of the cryptographic key we assume that the duration of the shaking processes is fixed to 5 seconds. Hence, the key generation algorithm is sensitive on the three parameters $L, M,$ and $Q$. We examine the influence of each parameter on the quality of the key generation algorithm by a systematic exhaustive search ($L = 20, 25, \ldots, 150, M = 2, \ldots, 15,$ and $Q = 2, \ldots, 15$). The objective of the exhaustive search is to find combinations of the three parameters which concurrently maximize the ratio of successfully generated equal keys and the average entropy of these keys.

The results of the exhaustive search are illustrated in the Pareto chart, shown in Fig. 3, where the different envelope functions illustrate the boundary of the maximum average entropy per key which can be achieved for a certain ratio of successfully generated keys depending on the time displacement $\tau$. Generally, the higher the entropy of a key, the lower is the relative number of generated equal keys. The Pareto chart also illustrates the influence of the displacement between the sequences on the quality of the key generation algorithm. Increasing the displacement between the sequences reduces the quality of the cryptographic key. Assuming genie aided synchronization ($\tau = 0$), the maximum entropy we can

---

**Fig. 2:** The 5 most important Eigenvectors and Eigenvalues.

As a second step follows a hash-function, which maps similar weight vectors $\hat{d}^{A}_{n,i}$ exactly to the same key segment and different $\hat{d}^{A}_{n,j}$ to different key segments. To this end, we build a predefined number $Q$ of groups of similar $\hat{d}^{A}_{n,i}$ using the Pairwise Nearest Neighbor (PNN) quantization algorithm [10], [11]. The PNN algorithm is based on two phases: First, the training phase is used to find groups of similar $\hat{d}^{A}_{n,i}$ where the centers of the groups are defined by so-called representation vectors $\hat{r}_{n,q}^{A}, q = 1,\ldots,Q$. At the beginning of calculating the representation vectors all $\hat{d}^{A}_{n,i}$ are considered to be representation vectors. Then iteratively, the two nearest representation vectors are replaced by their mean vectors until the predefined number $Q$ of representation vectors $\hat{r}_{n,q}^{A}$ is reached.
achieve with our proposed key generation algorithm is around 140 bits. Unfortunately, for this set-up, only one successful key could be generated from our test data. The Pareto chart illustrates that we reach our intended objective of generating exactly the same cryptographic keys on both devices with the same entropy as the Bluetooth PIN code (13 bit/key) for about 80% of all shaking processes.

Fig. 3 shows that the number of successfully generated keys is above 80% within a displacement of $\tau \leq 3$ samples. Consequently, the synchronization algorithm must synchronize the sequences within a displacement of $\tau \leq 3$ to successfully generate a reliable number of keys.

The Pareto chart also shows that there are parameters for which our key generation algorithm does not work properly, i.e. it always generates the same key regardless of the input sequence, which results in an entropy of 0 bits. In the majority of cases, however, there exists a set-up of parameters which allows a robust and reliable key generation with reasonable entropy. Note that different shaking processes yielding to the same key do not happen, practically.

IV. SYNCHRONIZATION

In the previous section we have shown that the quality of the key generation algorithm depends on the displacement between the shaking sequences. In this section, we propose an algorithm to synchronize both shaken devices independently of each other. The objective of the synchronization is to keep the displacement between the synchronized sequences as small as possible. We define the displacement between the synchronized sequences as the index of the peak of the cross-covariance function of the sequences.

In order to conduct the synchronization, we need to define a common valid condition to determine the starting point of the shaking processes. Basically, this is achieved by searching for specific characteristics defined by the shaking process inside the acceleration sequence provided by the hardware prototypes. We separate the synchronization algorithm into two phases:

First, a coarse approximation of the supposed starting point of the shaking process is produced. This approximation is kept simple to avoid computing power intensive methods. The first observed characteristic is the energy $E_n^A(t) = |(s_{n,t}, \ldots, s_{n,t+W})|^2$, which is computed for a window of $W$ acceleration samples of the sequence $s_n^A$. We take the smallest $t_1$ for which $E_n^A(t_1) > E_{th}$ as a first estimate of the starting point of sequence $s_n^A$, where $E_{th}$ denotes an appropriate, predefined threshold.

Mainly due to the tolerance of the acceleration sensors, the recorded acceleration sequence of two devices when shaken together is only similar. Consequently, the energy slope differs between the shaken devices. Thus, the energy of both sequences $E_n^A(t)$ and $E_n^B(t)$ exceeds the threshold $E_{th}$ at different times $t$. Hence, the time displacement between the two shaking sequences does not fulfill the quality on the displacement which is required by the key generation
algorithm.

In Sec. III, we have shown that segments of the shaking sequences can be represented by a small number of patterns $v_n$, where $v_n$ is the strongest pronounced component. Thus, in the second phase of the synchronization algorithm, we refine the starting point of the shaking process by detecting $v_1$ in the acceleration sequence $w_{n,t}^A$, $t \geq t_1$, where $w_{n,t}^A$ denotes the zero mean and the unit energy version of $\{s_{n,t}, \ldots, s_{n,t+L}\}$. More precisely, we define the second starting time $t_2$ as the time index of the sequence where the first local maximum of the correlation of the first pattern $v_1$ with the sequence of $w_{n,t}^A$, $t \geq t_1$ reaches a predefined threshold. Note that both $v_1$ and $w_{n,t}^A$, $t \geq t_1$ have the same length, are zero mean and of unit energy.

To determine the performance of the synchronization algorithm, we apply the same synchronization algorithm on the acceleration sequences $x_n^A$ and $x_n^B$ provided by prototype A and B, respectively. Once we have found the starting point of the shaking processes, we consider the following 1000 acceleration samples as the synchronized shaking sequences $\alpha_n$ and $\beta_n$. As a reference, we calculate the index of the peak of the cross-covariance function between $\alpha_n$ and $\beta_n$, which yields optimal synchronization. Fig. 4 illustrates the distribution function of the absolute values of the displacement $\tau$ of the synchronized sequences according to the algorithm we have described above to the ideal reference for 0, 1, ..., 10 samples. Only a few outliers have a displacement of more than 10 samples and are therefore not shown. More than 83% of all shaking sequences can be synchronized within a displacement of $\tau \leq 5$ samples which fulfills closely the criterion required from the key generation algorithm defined in the previous section.

Fig. 3 also contains the results for the key generation algorithm combined with the synchronization explained above. The resultant envelope function is between the envelope function using genie-aided synchronization and the envelope function with a constant displacement of 5 samples of the shaking sequences. We reach our intended objective of generating cryptographic keys with the same entropy as the Bluetooth PIN code (13bit/key) in about 70% of the autonomously synchronized shaking sequences. Consequently, the quality of the key generation algorithm is only sparsely sensitive to relatively small time displacements.

V. CONCLUSION

In this paper, we have assessed the idea of generating a cryptographic key by measuring acceleration data on small hand-held devices. The key is to be used during the pairing process and enables a secure connection between the devices. To this end, the two devices that shall initiate a secure connection are shaken together, so that both experience the same acceleration over time, from which the cryptographic key is generated locally without any communication between the devices.

We have introduced a key generation algorithm which is based on pairwise nearest neighbor quantization. Furthermore, we have presented a synchronization algorithm to independently synchronize the shaking sequences, where 83% of the shaking sequences can be synchronized with a displacement $\leq 5$ samples. Practical experiments assuming off-line computation have shown that with a success rate of about 70%, a key with an average entropy of 13 bits can be generated completely autonomously. This corresponds e.g. to the cryptographic strength of the Bluetooth PIN code.

Additionally, we have shown that the impact of the presented synchronization algorithm to the performance of the key generation algorithm is quite low. Thus, future work will be dedicated to optimizing the key generation algorithm by utilizing more sophisticated quantization algorithms.

REFERENCES