Key Generation Based on Acceleration Data of Shaking Processes

Daniel Bichler[1], Guido Stromberg[1], Mario Huemer[2], and Manuel Löw[1]

[1] Infineon Technologies AG, Germany
firstname.surname@infineon.com
[2] University of Klagenfurt, Austria
mario.huemer@uni-klu.ac.at

Abstract. Hard restrictions in computing power and energy consumption favour symmetric key methods to encrypt the communication in wireless body area networks which in term impose questions on effective and user-friendly unobtrusive ways for key distribution. In this paper, we present a novel approach to establish a secure connection between two devices by shaking them together. Instead of distributing or exchanging a key, the devices independently generate a key from the measured acceleration data by appropriate signal processing methods. Exhaustive practical experiments based on acceleration data gathered from real hardware prototypes have shown that in about 80% of the cases, a common key can be successfully generated. The average entropy of these generated keys exceed 13bits.

1 Introduction

Security and privacy are key issues in pervasive network environments. Despite their importance, security and privacy need to be implemented without confining the usability of the networked devices. This is particularly true, since at least conceptually, the number of devices is high, and many applications require a secure connection between dedicated devices instead of trusted zones in which many devices share a common key. However, the level of security and privacy in pervasive applications varies largely and depends highly on the application [1]. A further restriction arises because most devices in such applications are small, battery powered, and have little computational power so that complex algorithms for encryption or key exchange such as public and private key methods are infeasible [2]. Thus, we constrain our discussion to systems that rely on symmetric encryption methods. In these systems, 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 done by key exchange methods, which are either manual (e.g. typing in the key in a keypad) or exploit key-exchange algorithms.

Let us consider creating a secure wireless communication between dedicated personal devices, for example, a mobile phone and a headset. The most popular communication technology for personal area networks is Bluetooth, which

J. Krumm et al. (Eds.): UbiComp 2007, LNCS 4717, pp. 304–317, 2007.

already provides a security mechanism to encrypt the wireless communication between dedicated devices. The security mechanism is based on a two-stepped approach. In the first step, the so-called pairing phase, the master of the local Bluetooth cell generates a random key which it transports to the slave device. In order to prevent sniffing the key, it is encrypted with the so-called personal identification number (PIN). In the subsequent second step, this transmitted key is used to encrypt and decrypt the user data. Thus, both the slave and the master Bluetooth devices operate on the same encryption key. The possible key length used in Bluetooth applications ranges from 8 bit to 128 bit depending on the security level of the participating Bluetooth devices [3,4]. Typically, the Bluetooth PIN consists of three to four digits which range from 0 to 9 [5]. This corresponds to a maximum key-strength of about 10 to 13 bits.

There are three ways in which the PIN code can be shared between devices. The first way, commonly used for devices with limited user interface (e.g. head phones), is using a fixed PIN that is factory-installed and cannot be changed. This method thus limits the level of security significantly. Alternatively, a small number of factory-default PINs can be selected using a limited user interface such as DIP switches. The third method, which is offered by cell phones, is typing in the key using a keypad. This method offers a reasonably secure way of establishing a common key, but is also the least user-friendly. In any case however, pairing Bluetooth devices is often a tedious task.

In this paper, we present and evaluate an alternative to exchanging keys using computing intensive methods or to typing the key in manually. The basic idea is to generate a key or PIN locally from exposing devices to common physical environmental conditions. In particular, we consider two devices being shaken together and to use the recorded acceleration samples to generate local keys on both devices. This is especially practical for hand held devices. As we believe that the Bluetooth application space is quite typical in respect of the required level of security for personal area networks, our goal is to develop a symmetric key or PIN which is equivalently strong as the Bluetooth PIN.

Thus, the system must be designed so the devices create exactly the same symmetric key on 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, especially for small, mobile, battery powered personal devices.

Although we believe that the local generation of encryption keys from acceleration measurements without exchanging any acceleration information between the shaken devices is novel, there exist several approaches for key exchange in symmetric key encryption systems. Commonly, key distribution algorithms are time and computing power intensive because secure key distribution is based on complex mathematic algorithms. Diffie and Hellman (DH) proposed an algorithm

for securely distributing a symmetric key between two parties [6]. The vulnerabilities of the so-called DH key agreement protocol are carefully described in [7]. 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 the subsequent communication or to securely exchange additional symmetric keys [8]. This method increases the initial configuration effort and complicates the usability of each device, but it ensures security and privacy.

In ubiquitous computing, accelerometers have already been used in several ways and for several applications. In context awareness applications, sensor data is jointly processed in order to estimate certain conditions of the surrounding. For example, in the case of activity recognition, people carry several sensors, such as accelerometers integrated into their clothes, which decide autonomously on particular events [9]. Finally, the acceleration information, recorded while moving, can be used to detect physical activities and to capture the local dynamics of people [10]. Additionally, acceleration sensors can be applicable to establishing connections between devices by bumping objects together [11]. Another relevant work which uses acceleration sensors is given in [12] where the sensors are used to recognize whether they are carried by the same person.

The prior work that probably has the closest relation to ours is conducted within the framework of the *Smart-Its* project, in which devices in their direct surroundings are grouped by considering their proximity [13], e.g. by measuring the acceleration while shaking the devices. The acceleration data is then broadcasted to all devices within 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. This work has been extended by using a secure transmission protocol to exchange the acceleration information between the shaken devices [14]. They use exponentially quantized FFT coefficients of the acceleration sequence [15] to derive several key parts locally on each device. Again, depending on the similarity of the received and the local key parts, the devices determine if they are shaken together or not. Finally, only the equal key parts are used to derive a cryptographic key. One drawback of this method is that both devices have to exchange their own derived key parts.

However, our prime objective is not to exchange the acceleration characteristics, but to locally generate unique keys which are kept secret on each device. In our work, both devices should work completely autonomously without exchanging any acceleration information between the dedicated devices. The methods noted above are not appropriate for our independent key generation algorithm.

This paper is organized as follows: The feasibility of our approach is generally examined in Sec. 2. In particular, we assess the similarity of the acceleration sequence between devices of the shaking processes and determine the maximum entropy of the acceleration sequences to validate that ideally, a certain randomness of the generated keys can be guaranteed. In Sec. 3, we introduce a key generation algorithm which aims at providing exactly the same key on both devices

if and only if they are shaken together. The quality of the generated keys and other results are illustrated in Sec. 4. In Sec. 5, we summarize the results of our work and provide an outlook on further optimization of the key generation algorithm.

2 Signal Analysis

Let us start with considering shaking two devices in one hand. Typically, the shaking process consists of fast up and down movements in the three dimensional space, while bar-mixer type rotations seldomly occur.

In our experimental setup, we first record the shaking process as a sequence of three-dimensional vectors, each component representing the force in the x-, y-, and z-directions, respectively. We further assume that the relative orientation of the devices is not known a priori. For our subsequent calculations, we always compute and consider only the absolute values of the acceleration vectors. Firstly, this reduces the influence of the relative alignment of the acceleration sensors inside the hand. Secondly, since the shaking is usually on one fixed axis, as we have validated in some initial experiments, we lose only a small fraction of information about the shaking process. Thus, we can actually view the shaking process similar to a one-dimensional oscillation.

Since both devices are unsynchronized by default, the devices must agree on a common starting point of the shaking process. However, in order to evaluate different key generation algorithms independent of the time shift between both sequences, in this paper, we assume genie aided synchronization, for which we assume no time displacement between both sequences. In our setup, we have first completely recorded the two shaking sequences, and then off-line computed and adjusted the time displacement of the sequences by considering the peak of their cross-correlation function. In a more practical way, it would also have been possible to synchronize the start of recording the shaking process by explicit RF communication between two devices. As we will see during the course of this paper, in our current system, only the information of the starting point of the shaking process might need to be exchanged between the shaken devices.

Before we investigate how keys can be generated from shaking processes, let us first check the feasibility of our approach. There are two criteria that we consider as prerequisite for our effort: First, the devices shaken together must exhibit a much more similar sequence of the acceleration data than all other sequences, and second, the randomness of the sequence must exceed that of the desired key.

To this end, we have built prototypes with 3D acceleration sensors. The value of the acceleration sensors are 10bit A/D converted with a sampling rate of $f_s = 200$Hz using a 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. All 3-D data is low-pass filtered with a first order filter with cut-off frequency of 100Hz. As discussed above, the absolute values are computed. In this paper, we consider one absolute value of the acceleration vector as one sample.

Our test data consists of 88 shaking experiments recorded from 10 individuals. All persons were asked to shake two devices together in one hand for at least 5 seconds. Our test data thus generated consists of the ensemble $\mathcal{E} = (\mathcal{A}, \mathcal{B})$ of $S = 88$ shaking experiments, where each shaking experiment consists of two sequences; $\mathcal{A}$ includes all shaking sequences from prototype device A and $\mathcal{B}$ these from prototype device B. We denote the synchronized sequences of prototype A and B by $\underline{a}_n \in \mathcal{A}$ and $\underline{b}_m \in \mathcal{B}$, where n and m are the index of our shaking experiment. We limit the duration of each shaking process to 5 seconds, which yields sequences of exactly 1000 acceleration samples. $\hat{\mathcal{E}} = (\hat{\mathcal{A}}, \hat{\mathcal{B}})$ with $\underline{\hat{a}}_n \in \hat{\mathcal{A}}$ and $\underline{\hat{b}}_m \in \hat{\mathcal{B}}$ are the zero mean and the unit energy versions of $\underline{a}_n, \underline{b}_m$, respectively.

2.1 Similarity Measure Between Shaking Processes

As we want two devices shaken together to generate exactly the same cryptographic key, the resulting sequences of the acceleration vectors shall be as similar as possible. Accordingly, two sequences of different shaking processes shall be as different as possible to avoid generating the same key when the devices are not shaken together.

The degree of similarity of two sequences is typically expressed by the cross-covariance [16]. We consider both time and frequency analyses, for which the results are summarized in Tab. 1. If devices are shaken together, the maximum value of the cross-covariance of $\underline{\hat{a}}_n, \underline{\hat{b}}_n$ is 99.8% in the time domain. The cross-covariance of the frequency spectrum is generally even higher because we ignore the time information when the respective frequency component occurs. Ideally, the maximum cross-covariance of any $\underline{\hat{a}}_n, \underline{\hat{b}}_n$ would be 1. However, the measurements are affected by tolerances of the acceleration sensors. Additionally, the spatial distance between the acceleration sensors during the shaking process has a small influence on the acceleration measurements due to receiving different centrifugal forces in case of a circular motion. These influences have an impact on the cross-covariance in both the time and the frequency domain. Furthermore, the missing timing synchronization between the ADC of the hardware prototypes results in a sub-sample displacement of the sequence in the time domain. This could be reduced by oversampling or by interpolation.

Analyzing the cross-covariance of different shaking processes, i.e. of $\underline{\hat{a}}_m, \underline{\hat{b}}_n$ for $m \neq n$, we see fundamentally different behavior of the cross-covariance measures in the time and in the frequency domain. While the maximum value of the cross-covariance is only 56.4% in the time domain, the cross-covariance of the frequency spectrum is 92.3%. Due to the high cross-covariance of the frequency spectrum, sequences from the same shaking processes are not distinguishable from others. Thus, the same key would rather likely be generated for different shaking processes if the key generation was based on a frequency-based technique. For this reason, the frequency domain is not suitable for key generation and we will continuing analyzing the sequences only in the time domain.

Table 1. Characteristics of the maxima of the cross-covariance of all shaking sequences of $\mathcal{E}$ in the time and frequency domain

	characteristics of cross-covariance			
	$cov(\underline{\hat{a}}_m, \underline{\hat{b}}_m), m = 1, \ldots, \lvert S\rvert$ (same shaking experiments)		$cov(\underline{\hat{a}}_m, \underline{\hat{b}}_n), m \neq n, m, n = 1, \ldots, \lvert S\rvert$ (different shaking experiments)	
	time domain	frequency domain	time domain	frequency domain
minima	0.871	0.938	0.04	0.094
first 25%	0.871 to 0.973	0.938 to 0.982	0.04 to 0.128	0.094 to 0.166
50%	0.973 to 0.991	0.982 to 0.996	0.128 to 0.258	0.166 to 0.461
last 25%	0.991 to 0.998	0.996 to 0.9996	0.258 to 0.564	0.461 to 0.923
median	0.985	0.993	0.18	0.300
maxima	0.998	0.9996	0.564	0.923

2.2 Estimating the Quality of the Cryptographic Key

The quality of a cryptographic key is given by its randomness, which can be expressed by entropy [17]. In this section we will estimate the highest possible entropy of the keys we can generate by calculating the entropy of the shaking process. This is valid since the entropy of the generated keys will never exceed the entropy of the shaking process, regardless of which mathematical transformation is used to calculate the cryptographic key. The entropy of the shaking process can only be increased by extending the duration of the shaking process.

To estimate the entropy correctly, we must consider the dependencies between the samples of the sequence. To this end, the entropy must be calculated based on the conditional probability of correlated lag vectors, i.e. of segments which contain correlated samples [18]. The length of these lag vectors is indicated by the length of the autocorrelation function [19]. If the length of the autocorrelation function is one, the samples of the sequence are completely uncorrelated and the conditional probability equals the probability of each individual sample. Unfortunately, in our case the sequence is highly correlated, see Fig. 1, and the autocorrelation function is almost as long as the complete sequence. To correctly estimate the conditional probability, we would need a huge statistical basis to cover all possible lag vectors. First calculations have shown that due to the limited length of our sequences, the statistical basis is too weak to reliably estimate the conditional probabilities of the shaking sequences in this straightforward way.

Thus, we employ a different approach to calculate the entropy of the sequences by using a linear forward prediction filter. The prediction filter predicts the correlated part of the sequence according to the dependencies given by the autocorrelation function [20,21]. As we know, the entropy of a predictable sequence is zero, thus the entropy of the error of the prediction filter equals the entropy of the original sequence. The forward prediction error of the prediction filter yields a quite uncorrelated sequence, see Fig. 2. Consequently, the entropy is close to the unconditional entropy of the individual samples of the forward prediction error.

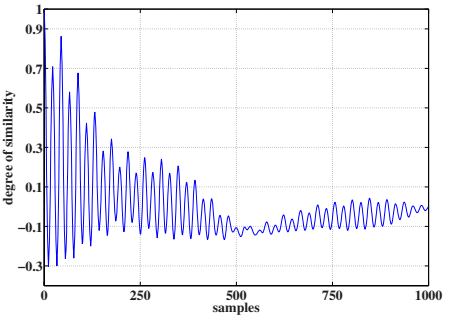

Fig. 1. Autocorrelation function of a shaking sequence

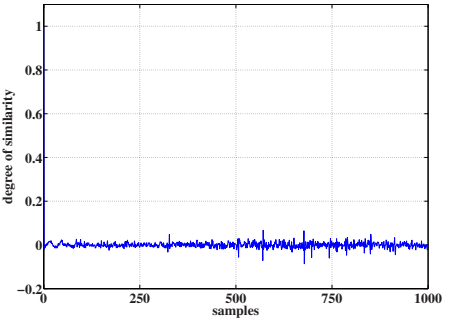

Fig. 2. Autocorrelation function of the forward prediction error

To calculate the entropy of the shaking processes, we have concatenated all shaking sequences from the prototype A of our test data, described in Sec. 2.1, to a sequence $\underline{s}^A = (\underline{a}_1, \ldots, \underline{a}_{88}), \underline{a}_n \in \mathcal{A}$. Afterwards, we have constructed a linear forward prediction filter, which exhibits the same length as the autocorrelation function of $\underline{s}^A$. Then, we have estimated the entropy of the sequence resulting from the forward prediction error of $\underline{s}^A$, which yields an average entropy of 3800bit/sequence. Note that the use of such prediction filter is infeasible for key generation and is just used for the assessment of the entropy of the shaking process.

Reflecting the presented results in this section, we assume that the cross-covariance between sequences from the same shaking processes are high enough to independently generate exactly the same key from both sequences. Additionally, the cross-covariance of sequences from different shaking processes is sufficiently small to guarantee different keys from different shaking processes. Furthermore, we have demonstrated that the shaking process of 5 seconds length exhibits a significant amount of entropy in the range of 3800 bits. This largely exceeds the entropy of the Bluetooth PIN code. Although we thus believe that the shaking process can be used for key generation, the evidence did not yield any hint on the key generation algorithm itself. The construction of such algorithm will be discussed in the following section.

3 Key Generation

Our objective is to generate exactly the same cryptographic key out of two shaking sequences obtained by two independent hardware prototypes if and only if they are shaken together and without exchanging any key parts or acceleration data. Due to the fact that both sequences are not identical, see Sec. 2.1, the key generation algorithm must have the ability to map even similar sequences to the same key and sequences with less similarity to different keys. The key generation algorithm is applied separately on both shaking sequences of the

hardware prototypes without any interaction. As the algorithm is the same on both devices, let us now consider just one prototype, say device A.

In the previous section we have shown that key generation based on time domain analysis is most beneficial. To this end, our approach for generating a cryptographic key $\underline{k}_n^A$ is that we start with splitting the shaking sequence $\underline{a}_n$ into I segments

$$\underline{a}_{n,i} = (a_{n,i\cdot L}, \ldots, a_{n,(i+1)\cdot L}). \tag{1}$$

$\underline{a}_{n,i}$ indicates the ith segment ($i = 0, \ldots, I-1$), each consisting of L subsequent samples of measured absolute samples. Note that the length of each shaking sequence is limited to 1000 acceleration samples. Thus, the number of segments $I = \lfloor 1000/L \rfloor$ only depends on the segment length L. $a_{n,i\cdot L}$ represents the $(i\cdot L)$th sample value of the shaking sequence $\underline{a}_n$. We further normalize each $\underline{a}_{n,i}$ to

$$\hat{\underline{a}}_{n,i} = \frac{\underline{a}_{n,i} - \bar{\underline{a}}_{n,i}}{|\underline{a}_{n,i}|}, \tag{2}$$

where $\bar{\underline{a}}_{n,i}$ represents the mean value of $\underline{a}_{n,i}$. Now, all segments are zero mean and have unit energy. Then, we calculate from each segment $\hat{\underline{a}}_{n,i}$ a fragment of the cryptographic key $k_{n,i}^A$. At the end, the key $\underline{k}_n^A$ consists of the concatenation of I fragments.

The calculation of the key's fragments is done in two steps as shown in Fig. 3. First, we reduce the dimensionality of the segments $\hat{\underline{a}}_{n,i}$. The objectives herein are 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. 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 $\hat{\underline{a}}_{n,i}$ and the patterns $\underline{v}_m, m = 1, \ldots, M$ and are summarized in a weight vector

$$d_{n,i,j}^A = \sum_{j=0}^{L-1} \hat{a}_{n,i,j} v_{m,j}, \tag{3}$$

which constitute the outcome of the correlator bank as shown in Fig. 3. Thus the length of the vectors $\hat{\underline{a}}_{n,i}$ equals the length of the patterns $\underline{v}_m$. M indicates the number of patterns that are used for the representation of the segments $\hat{\underline{a}}_{n,i}$.

The patterns are computed from a separate training set $\mathcal{E}'$ using the principal component analysis (PCA) [22]. To this end, we have additionally recorded 15 shaking processes $\mathcal{E}' = (\mathcal{A}', \mathcal{B}')$ to calculate the Eigenvalues. The results of the PCA are the Eigenvectors and Eigenvalues of the covariance matrix of all segments with length L from the acceleration sequences in $\mathcal{E}'$. The Eigenvectors represent the main components and the Eigenvalues indicate how strong the respective Eigenvectors are pronounced in the segments of $\mathcal{E}'$. The segments can be completely reassembled by a linear combination of the Eigenvectors. To reduce the dimensionality of the weight vectors $\underline{d}_{n,i}^A$, obtained when correlating

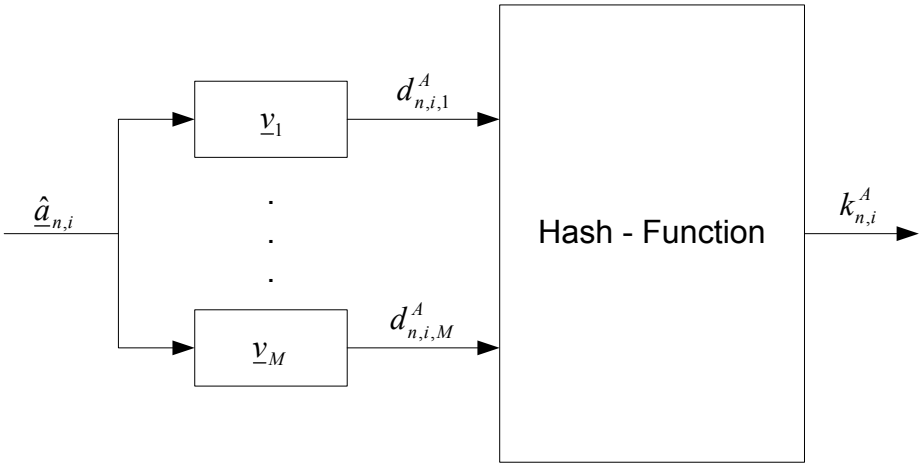

Fig. 3. Correlation of Segments with Patterns and subsequent Hash Function

the segments $\hat{\underline{a}}_{n,i}$ with the Eigenvectors, we neglect the Eigenvectors with the smallest corresponding Eigenvalues.

Fig. 4 shows the five Eigenvectors with the highest corresponding Eigenvalues of the training set $\mathcal{E}'$, where the shaking sequences are divided into I segments of $L = 40$ samples. Additionally, the graph of the Eigenvalues indicates that 5 Eigenvectors are sufficient for representing the data set $\mathcal{E}'$ to more than 95% of its signal energy. Note that the patterns $\underline{v}_m$ are only computed once and then kept fixed for all our experiments. The patterns are calculated again only if the segment length changes because both the segment length and the pattern length must be equal for the correlation.

As a second step follows a hash-function, which maps similar weight vectors $\underline{d}^A_{n,i}$ exactly to the same key segment and different $\underline{d}^A_{n,i}$ to different key segments. To this end, we establish a predefined number Q of groups. This is achieved in two phases: First, in the training phase, we use an agglomerative hierarchical clustering algorithm to find the center of Q groups. Each group is represented by a so-called representation vector $\underline{r}_{n,q}, q = 1, \ldots, Q$. To estimate the $\underline{r}_{n,q}$ we use the same method as building a dendrogram. At the beginning, all $\underline{d}^A_{n,i}$ are considered to be representation vectors. Then, iteratively, the two nearest representation vectors are replaced by their mean vector until the predefined number Q of representation vectors $\underline{r}^A_{n,q}$ is reached [22].

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

$$k^A_{n,i} = \underset{q=1,\ldots,Q}{argmin} |\underline{d}^A_{n,i} - \underline{r}^A_{n,q}| \tag{4}$$

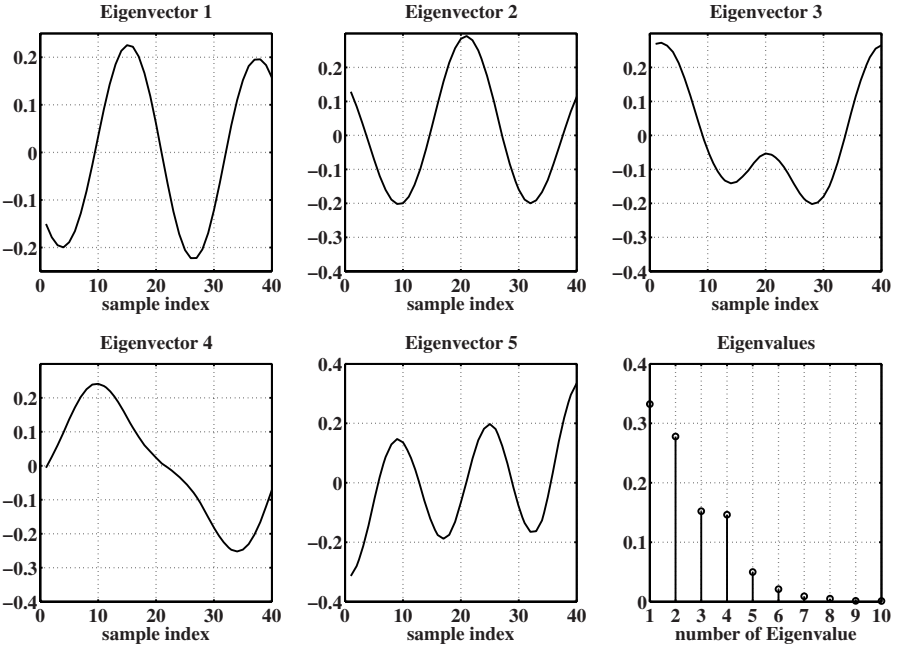

Fig. 4. The 5 most important Eigenvectors and Eigenvalues

is selected. Thus, the index q of the representation vector $\underline{r}^A_{n,q}$, to which $\underline{d}^A_{n,i}$ is closest, represents the ith fragment of the key $k^A_{n,i}$. Similar $\underline{d}^A_{n,i}$ are therefore assigned to the same representation vector and result in the same key fragment.

Therefore, the key generation is a two-step process. After the accelerometer data has been sampled and the $\underline{d}^A_{n,i}$ have been computed based on fixed Eigenvector patterns $\underline{v}_m$, first the representation vectors $\underline{r}^A_{n,q}$ are computed in the so-called training step. In the following classification phase, the key fragments $k^A_{n,i}$ are calculated using these representation vectors. The eigenvector projection as well as the classification step are therefore conducted individually for each shaking process and on each device.

As indicated above, the same procedure is applied on both prototype devices A and B. The keys for the nth shaking experiment of prototypes A and B

$$\underline{k}^A_n = (k^A_{n,0}, \ldots, k^A_{n,I-1}) \tag{5}$$

$$\underline{k}^B_n = (k^B_{n,0}, \ldots, k^B_{n,I-1}) \tag{6}$$

are composed by concatenating the key fragments $k^A_{n,i}$ and $k^B_{n,i}$, respectively.

As an example, Fig. 5 shows the result of a correct classification of the weight vectors of a shaking experiment $\underline{d}^A_{n,i}$ and $\underline{d}^B_{n,j}$ with $Q = 4$ regions and $M = 3$ patterns. The fundamental condition to generate independently the same key from $\underline{d}^A_{n,i}$ and $\underline{d}^B_{n,j}$ is that the weight vectors with the same index $i = j$ are

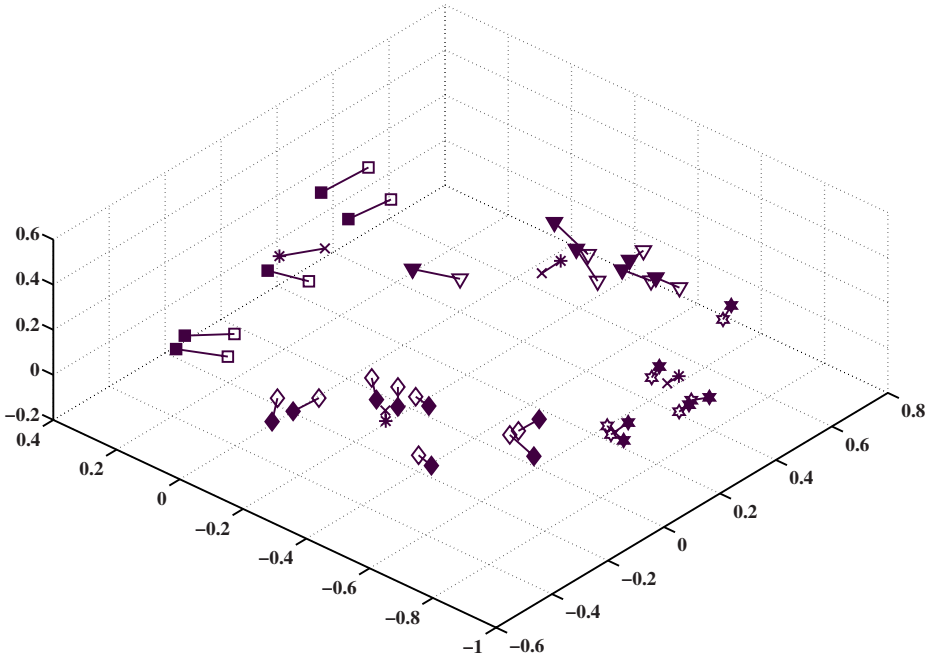

Fig. 5. Grouping of weight vectors from two shaking sequences of the first shaking experiment: The solid markers represent the $\underline{d}^A_{n,i}$ and the blank markers the $\underline{d}^B_{n,j}$. $\underline{d}^A_{n,i}$ and $\underline{d}^B_{n,j}$ with $i = j$ are connected via a line. The $\underline{d}^A_{n,i}$ and $\underline{d}^B_{n,j}$ with the same symbol are assigned to the same $\underline{r}^A_{n,q}$ and $\underline{r}^B_{n,q}$, respectively. The $*$-markers represent the $\underline{r}^A_{n,q}$ and x-markers the $\underline{r}^B_{n,q}$ of $Q = 4$ groups.

assigned to the representation vectors $\underline{r}^A_{n,q}$ and $\underline{r}^B_{n,q}$ with the same index q. Note that the representation vectors $\underline{r}^A_{n,q}$ and $\underline{r}^B_{n,q}$ are usually different.

4 Results

In this section we report on the quality of the previously explained key generation algorithm applied to the test data $\mathcal{E}$ which contains 88 shaking sequences of the prototypes A and B when shaken together. Ideally, all generated keys between prototypes A and B from the same shaking processes should be equal, thus $\underline{k}^A_n = \underline{k}^B_n$. Practically, however, we will see that it is not always possible to generate exactly the same cryptographic key on both prototypes when they are shaken together.

Especially important for estimating the quality of the key generation algorithm is the number of successful cases for which $\underline{k}^A_n = \underline{k}^B_n$ in relation to the total number of experiments. We are further interested in the average entropy of the equal keys. To this end, we calculate the entropy of the successfully

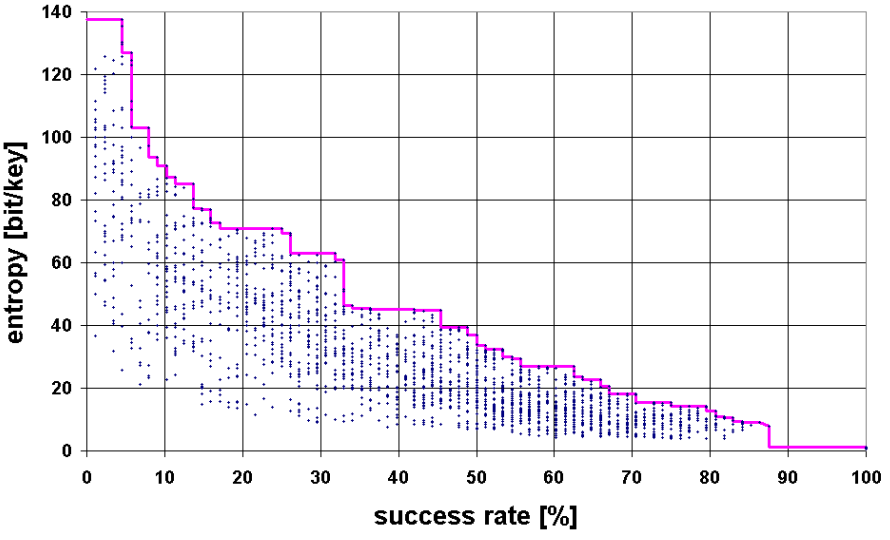

Fig. 6. Pareto chart of the results of the exhaustive search

generated keys considering the conditional probability of correlated lag vectors as explained in Sec. 2.2.

During this assessment, we assume that the duration of the shaking processes is fixed to 5 seconds which corresponds to 1000 samples. Hence, the key generation algorithm is sensitive on only three parameters:

1. **Segment length** L**:** The segment length $L = |\underline{\hat{a}}_{n,i}| = |\underline{\hat{b}}_{n,i}|$ affects the number of segments of each sequence and thus determines the number of key fragments ($I = \lfloor 1000/L \rfloor$).
2. **Number of patterns** M**:** M determines the accuracy of the segment representation by weight vectors and their dimensionality of the quantization space.
3. **Number of representation vectors** Q**:** Q defines the number of groups of weight vectors and the number of different symbols of which the cryptographic key consists.

We examine the influence of these parameters on the quality of the key generation algorithm by a systematic exhaustive search. We vary the respective parameters within a dedicated range, i.e. $L = 20, 25, \ldots, 150$, $M = 2, \ldots, 15$, 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. 6. The envelope function illustrates the boundary of the maximum average entropy per key (y-axis) which can be achieved for a certain ratio of successfully generated keys

(x-axis). Each dot represents the outcome of the key generation algorithm for one specific setting of the three given parameters L, M, and Q.

Generally, the higher the entropy of a key, the lower the relative number of generated equal keys. The maximum entropy we can achieve with the proposed key generation algorithm is around 140 bits. Unfortunately, only one successful key could be generated from our test data for this selection of parameters. The Pareto chart also shows that there are parameters for which our key generation algorithm does not work properly, i.e. the algorithm 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 is a set-up of parameters which allows a robust and reliable key generation with reasonable entropy. In particular, we reach our intended objective of generating cryptographic keys with the same entropy as the Bluetooth PIN code (13bit/key) in about 80% of the cases with the parameter setting of $L = 75, M = 9$ and $Q = 12$. In this example, only 1.3% of the experiments from different shaking processes result in the same key.

5 Conclusion and Outlook

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. Practical experiments assuming genie-aided synchronization and off-line computation have shown that with a success rate of about 80%, a key with an average entropy of 13 bits can be generated. This corresponds e.g. to the cryptographic strength of the Bluetooth PIN code.

Further work will be dedicated to synchronization and improved quantization algorithms.

References

1. Stajano, F.: Security for ubiquitous computing. John Wiley & Sons, Chichester (2002)
2. Cam, H., Özdemir, S., Muthuavinashiappan, D., Nair, P.: Energy-efficient security protocol for wireless sensor networks. In: IEEE VTC Fall Conference, vol. 5, pp. 2981–2984. IEEE, Los Alamitos (2003)
3. Bluetooth Spezial Interest Group: Specification of the Bluetooth System. Bluetooth, Version 1.1, vol. 1 (2001)
4. Bluetooth Spezial Interest Group: Specification of the Bluetooth System. Bluetooth, vol. 2, Version 1.1 (2001)
5. Muller, T.: Bluetooth security architecture. White Paper Version 1.0, Bluetooth (1999)

6. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory 22, 644–654 (1976)
7. Raymond, J.F., Stiglic, A.: Security issues in the Diffie-Hellman key agreement protocol. IEEE Transactions on Information Theory 22, 1–17 (2000)
8. Boyd, C., Mathuria, A.: Key establishment protocols for secure mobile communications: A selective survey. In: Boyd, C., Dawson, E. (eds.) ACISP 1998. LNCS, vol. 1438, pp. 3–540. Springer, Heidelberg (1998)
9. Bao, L.: Physical activity recognition from acceleration data under semi-naturalistic conditions. Master's thesis, Massachusetts institute of technology (2003)
10. Bao, L., Intille, S.S.: Activity recognition from user-annotated acceleration data. In: Ferscha, A., Mattern, F. (eds.) PERVASIVE 2004. LNCS, vol. 3001, pp. 1–17. Springer, Heidelberg (2004)
11. Hinkley, K.: Bumping objects together as a semantically rich way of forming connections between ubiquitous devices. In: Dey, A.K., Schmidt, A., McCarthy, J.F. (eds.) UbiComp 2003. LNCS, vol. 2864, Springer, Heidelberg (2003)
12. Lester, J., Hannaford, B., Borriello, G.: "Are you with me?" - Using accelerometers to determine if two devices are carried by the same person. In: Ferscha, A., Mattern, F. (eds.) PERVASIVE 2004. LNCS, vol. 3001, pp. 33–50. Springer, Heidelberg (2004)
13. Holmquist, L.E., Mattern, F., Schiele, B., Alahuhta, P., Beigl, M., Gellersen, H.W.: Smart-its friends: A technique for users to easily establish connections between smart artefacts. In: Abowd, G.D., Brumitt, B., Shafer, S. (eds.) Ubicomp 2001: Ubiquitous Computing. LNCS, vol. 2201, pp. 273–291. Springer, Heidelberg (2001)
14. Mayrhofer, R., Gellersen, H.: Shake well before use: Authentication based on acceleration data. In: Pervasive 2007: 5th International Conference on Pervasive Computing. LNCS, vol. 4480, pp. 144–161. Springer, Heidelberg (2007)
15. Huynh, T., Schiele, B.: Analyzing features for activity recognition. In: SocEUSAI, vol. 121, pp. 159–163 (2005)
16. Aylward, R., Lovell, S.D., Paradiso, J.A.: A compact, wireless, wearable sensor network for interactive dance ensembles. In: Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks (BSN'06), vol. 00, pp. 65–70. IEEE Computer Society Press, Los Alamitos (2006)
17. Shannon, C.: A mathematical theory of communication. Bell System Technical Journal 27, 379–423, 623–656 (1948)
18. Weigend, A.S., Gershenfeld, N.A.: Time series prediction: Forecasting the future and understanding the past, vol. 15. Addison-Wesley Publishing Company (1994)
19. Molgedey, L., Ebeling, W.: Local order, entropy and predictability of financial time series. In: The European Physical Journal B - Condensed Matter and Complex Systems, vol. 15, pp. 733–737. Springer, Heidelberg (2004)
20. Proakis, J.G., Manolakis, D.G.: Digital Signal Processing: Principles, Algorithms, and Applications, 2nd edn. Macmillan Publishing Company (1992) ISBN 0-02-396815-X
21. Haykin, S.: Adaptive filter theory, 4th edn. Prentice-Hall, Englewood Cliffs (2002)
22. Patterson, D.: Artificial neural networks, theroy and applications. Prentice Hall Inc., Englewood Cliffs (1996)