Shake Well Before Use: Authentication Based on Accelerometer Data

Rene Mayrhofer and Hans Gellersen

Lancaster University, Computing Department, South Drive, Lancaster LA1 4WA, UK
{rene,hwg}@comp.lancs.ac.uk

Abstract. Small, mobile devices without user interfaces, such as Bluetooth headsets, often need to communicate securely over wireless networks. Active attacks can only be prevented by authenticating wireless communication, which is problematic when devices do not have any a priori information about each other. We introduce a new method for device-to-device authentication by shaking devices together. This paper describes two protocols for combining cryptographic authentication techniques with known methods of accelerometer data analysis to the effect of generating authenticated, secret keys. The protocols differ in their design, one being more conservative from a security point of view, while the other allows more dynamic interactions. Three experiments are used to optimize and validate our proposed authentication method.

1 Introduction

Applications envisioned for ubiquitous computing build upon spontaneous interaction of devices, such that a device can make serendipitous use of the services provided by peer devices that may not be known a priori. In many scenarios, it will be desirable to verify and secure spontaneous interactions in order to ascertain that devices become paired as intended and protected against attacks on their wireless link. In a managed network environment, device-to-device authentication would be based on prior knowledge of each other or access to a trusted third party, but neither can be assumed to be available in wireless ad hoc networks for ubiquitous computing. As a consequence, secure device pairing requires the user to be in the loop, for example to enter a shared secret such as a PIN code into both devices. A challenge is to find mechanisms for users to pair devices that are not only secure but also scale well for use in ubiquitous computing. Specific challenges are that devices will, in many cases, be too small to reasonably include key pads and displays, and that required user attention must be minimal to be acceptable for spontaneous and short-lived interactions.

Pairing of a mobile phone with a headset for interaction over a wireless channel is a familiar example: we would like to achieve such interaction in a spontaneous manner (i.e. not requiring pre-configuration of phone and headset for each other) but also ensure that it is secure. The wireless communication channel between the devices is susceptible to attacks ranging from eavesdropping to

A. LaMarca et al. (Eds.): Pervasive 2007, LNCS 4480, pp. 144–161, 2007.

man-in-the-middle (MITM). If an attacker were successful in establishing themselves between, in this case, phone and headset, during the pairing process, then they would obtain complete control over all phone calls. To safeguard against such attacks, a so-called *out-of-band channel* is used during pairing in order to authenticate communication over the primary channel. The out-of-band channel must be limited such that it is user-controllable that only the intended devices can communicate over it for the purposes of authentication. Note that authentication and the subsequent pairing can be anonymous or "ephemeral" [1], i.e. based on information only shared over the out-of-band-channel rather than actual device identities.

In this paper we contribute a method for device-to-device authentication that is based on shared movement patterns which a user can simply generate by shaking devices together. Using embedded accelerometers, devices can recognize correlation of their movement and use movement patterns for authentication. From a user perspective, jointly shaking is a simple technique for associating devices [2]. In our method, it simultaneously serves as out-of-band mechanism. Shaking has a number of characteristics on which we can build for our purposes:

- It is *intuitive*. People are familiar with shaking objects as manual interaction that does not require learning, for instance from shaking of medicine, or musical instruments. This means that shaking is unobtrusive in the sense that it does not require the user's full attention while being performed.
- It is *vigorous*. While there are many motion patterns that could be performed with two devices, shaking tends to produce the highest continuous acceleration values. While bouncing will produce larger accelerations, they only occur as short spikes. Shaking provides acceleration larger than most activities – and can thus be detected by simple thresholding – for as long as necessary to pair devices (and as long as the user will not get tired).
- It is *varying*. As we will show below in our first experiment (in section 7.1), the activity of shaking can be surprisingly different for different people. We do not use shaking patterns as identification, but still benefit from large differences in acceleration values, because this generates high entropy from an attacker's point of view.

It is important to note that users do not have to follow a particular pattern of shaking but that they can shake as they like; we do not attempt to identify people by their shaking patterns, but use it as a source of shared device movement.

We contribute two protocols that combine cryptographic primitives with accelerometer data analysis to establish secure wireless channels by creating authenticated secret keys. The two protocols achieve this aim differently: the first is based on Diffie-Hellman key agreement and authentication of this key, uses a conservative and better known design, provides better security and allows more flexibility in comparing accelerometer time series; the second generates cryptographic key material directly out of accelerometer data streams, is computationally less expensive and thus easier to implement on resource limited devices, and allows more dynamic interactions and group authentication.

Both protocols use standard techniques of sensor data processing and time series analysis: sampling, alignment, and feature extraction. After extracting appropriate features, our cryptographic protocols ensure that authentication is only possible if both devices have access to the same feature values. Specifically, they protect against MITM attacks on the wireless communication channel by using additional information gathered from the extracted features. This approach is general, so that other sensors than accelerometers can be used with similar methods, apart from changes in domain-specific heuristics. Sensor-based authentication offers potential benefits to small, mobile devices that communicate wirelessly and do not have traditional user interfaces. Examples are mobile phones, smarts cards, key fobs, and generally accessories like headsets, watches, or glasses.

2 Related Work

First concepts on secure device pairing suggested direct electrical contact [3], while other suggestions to implement an out-of-band channel include a "physical interlock" and the "Harmony" protocol [4], ultrasound [5], visual markers and cameras [6], audio messages [7], the GSM short message service (SMS) [8], key comparison, distance bounding and integrity codes [9], or manual input [10,1]. The DH-DB protocol proposed in [9] might also be applicable to an interactive challenge-response scheme based on sensor data such as accelerometer data. These approaches, with the exception of using camera phones, have in common that they scale poorly from a user point of view. That is, they tend to be obtrusive and require the user's attention. In our approach, we implement a low bandwidth private channel over similar accelerometer readings, and use it for authenticating a device pairing.

The idea of shaking two (or multiple) devices together to pair them has first been described as "Smart-Its Friends" [2]. We use the same interaction technique but extend it to include secure authentication. Castelluccia and Mutaf presented a protocol for pairing CPU-constrained wireless devices under the assumption of anonymous broadcast channels [11]. To achieve this property of source indistinguishability, they argue that devices engaging in this authentication protocol should be shaken and rotated randomly around each other. This shaking serves to prevent signal strength analysis, but is, in contrast to our work, not used directly as input to the authentication protocol. Hinckley presented an implementation of "synchronous gestures" [12] as a means of user interaction. By correlating accelerometer time series on devices connected via WLAN, bumping them together or tilting them can be detected and used as user input. Bumping is one possible user interaction for starting the pairing process, i.e. a trigger for our authentication method. Another closely related work was presented by Lester et al. [13] and describes how to determine if two devices are carried by the same person.

3 Design of the Acceleration-Based Pairing Method

Figure 1 shows our architecture for authenticating device pairings with shaking patterns. Both protocols make use of the same three pre-processing tasks 1 to 3. They are executed locally on each device and result in "active" time series segments of equidistant samples. Our two protocols differ in tasks 4 and 5, which can both be interactive, i.e. communicate with the remote device to which the pairing is in process.

For protocol 1, tasks 4.1 and 5.1 are actually executed in parallel: after generating a secret key with standard Diffie-Hellman (DH) key agreement (which is the first phase of task 5.1), the devices exchange their time series segments via an interlock protocol. Then they compare their locally generated segment with the one received from the remote device to check if they are similar enough. If they pass this check, the second phase of task 5.1 derives the secret session key that will be used for consecutive secure communication. This design is conservative from a security point of view and, due to the non-interactive feature extraction and comparison, allows the devices to use different means of verification. The disadvantage of splitting task 5.1 into two phases is potentially a larger delay for authentication, and the disadvantage of using DH is higher computational load.

Protocol 2 executes its tasks 4.2 and 5.2 in order: discrete (in contrast to the real-valued samples) feature vectors are extracted in task 4.2, which act as input to the interactive key agreement in task 5.2. This is an iterative process. In each time step, feature vectors generated by 4.2 are checked for matches in task 5.2. After sufficient iterations, a secret shared key can be generated out of the collected matching feature vectors in task 5.2. This design has the advantages of more dynamic key agreement, with devices being able to "tune into" other device's key streams, and of being less computationally expensive. On the other hand, it does not provide forward secrecy and protection against offline attacks as protocol 1 does, and is more unconventional and thus less well studied from a security point of view.

For both protocols, there is a trade-off between usability and security that can be exploited by applications and users depending on their requirements. Tasks 4 and 5 are described in more detail in sections 5 and 6, respectively.

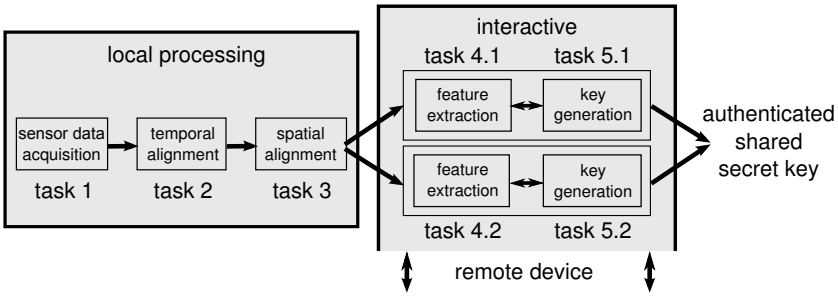


Fig. 1. Architecture for both authentication protocols

4 Pre-processing of Accelerometer Data

The three pre-processing tasks, executed as consecutive steps, are used to sample and segment the sensor data so that feature extraction can build on normalized time series.

Task 1: Sensor data acquisition. This first task is conceptually straight forward, but requires careful implementation. Sensor data is assumed to be available in the form of time series of acceleration values in all three dimensions, sampled at equidistant time steps. These must be taken locally and not be communicated wirelessly — for security purposes, it is critical not to leak any of this raw data, which can be difficult considering the possibility of powerful side-channel attacks (see e.g. [14]). Our practical experience shows a sample rate between 100 and 600 Hz to be appropriate.

Task 2: Temporal alignment. As the two devices sample accelerometer time series independently in task 1, we require temporal synchronization for comparison. We assume that devices are equipped with sufficiently accurate real-time clocks, so that differences in sampling rates and drift will not be issues. This reduces temporal alignment from an arbitrarily complex problem to *triggering* the authentication procedure and to *synchronizing* the starting points for time series comparison.

Triggering can be *explicit* by direct user input, e.g. pressing an "authenticate now" button on both devices within a short time frame or bumping both devices against the table or each other, or *implicit*, simply by starting to shake both devices. We prefer the second protocol due to its ease of use, although it is more difficult to implement. Synchronization can be at a *sample level*, i.e. within less than half the sample width, or at an *event level*, i.e. based on the onset of detected (explicit or implicit) events with the respective device. We use the latter, because it does not require time synchronization between the devices — shaking events can be detected locally at each device without communication, which is beneficial from a security point of view.

For both triggering and synchronization, we detect motion and align those parts of the time series where shaking is detected, which we call *active segments*, by their start times. Segments are considered active when the variance of a sliding window exceeds a threshold. Practical experiments show good results at a sample rate between $f = [128; 512]$ Hz with a sliding window of $v = f/2$ samples, i.e. 1/2 second, and a variance threshold around $T_\sigma = 750$.

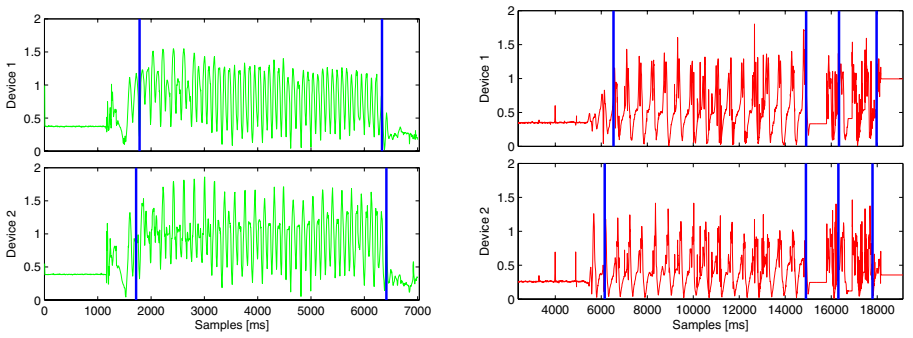
Task 3: Spatial alignment. Shaking is inherently a three-dimensional movement. In addition to the need to capture all three dimensions, the alignment between the two devices is unknown. This means that the three dimensions recorded by the two devices will not be aligned, which is a hard problem in itself. Lukowicz et al. describe how to calibrate three-dimensional accelerometers without user interaction during stable periods [15]. However, since we are interested in the active phases and

have to assume that the alignment of the devices changes during the transition,[1] we can not directly apply this result. Instead, we reduce the three dimensions to a single: by taking only the magnitude over all normalized dimensions, i.e. the length of the vector, we solve the alignment problem. This approach requires considerably less resources than other methods such as principal component analysis (PCA) or modeling using domain-specific knowledge.

The result of these steps is that, when shaken together, both devices will extract active segments of one-dimensional acceleration magnitude vectors. Even without synchronized clocks, the start times of these independent time series are typically synchronized within a few samples (on the event level).

5 Feature Extraction for Authentication Purposes

Two devices that are shaken together will experience similar, but not exactly the same movement patterns. Even assuming noise-free sampling of accelerations, the two accelerometers must have physically separate centers. Whenever rotation is part of the movement, these separate centers will necessarily experience different accelerations, thus causing different sensor time series even if the devices remain fixed in relation to each other. The problem of verifying that two devices are shaken, or more generally, moved together therefore becomes a classification problem. Figure 2 shows examples of spatially aligned sensor time series used as input to feature extraction with detected borders of active segments.



(a) Two devices shaken by one person in the same hand

(b) Two devices shaken by two people, one each

Fig. 2. Example time series after spatial alignment with detected active segments

In deciding if time series are similar enough for authentication, the aim of the feature extraction task is twofold: a) to extract feature values that are robust to small variations in the shaking patterns and to sampling noise and b) to extract

[1] When a user picks up the two devices to shake them, they will most probably be aligned differently in their hand than they were before picking them up.

a sufficiently large feature vector for use in the authentication protocol. In our approach, the feature vector will be used to authenticate a key or to directly generate a key, and thus it needs to be of high entropy from an attacker's point of view, i.e. involve a large amount of uncertainty.[2] As indicated in section 1, we argue that shaking is an appropriate movement for creating entropy: it creates varying sensor readings, because it is one of the human movement patterns that includes the highest frequency components. Slower movements will intuitively not generate as much entropy.

There is an extensive body of literature on feature extraction from accelerometer data. Particularly relevant to our problem are the described uses of the coherence measure by Lester et al. [13] and cross-covariance by Aylward et al. [16]. Both suggest sliding, windowed variances on each device for activity detection, as used in our current implementation for task 2. Huynh and Schiele compare different features for activity recognition and suggest the use of quantized FFT coefficients [17]. For task 4, we select the most promising of the recently suggested features: coherence and quantized FFT coefficients.

5.1 Coherence

We adopt the approach that was previously used by Lester et al. to distinguish between two devices worn by the same person (on different parts of the body) and two devices worn by two people walking in-step. They used coherence averages and showed that simple, non-calibrated, cheap accelerometers are suitable for analyzing human motion. Coherence is approximated by the magnitude squared coherence (MSC) as

$$C_{xy}(f) = \frac{P_{xy}(f)}{P_{xx}(f) \cdot P_{yy}(f)}$$

with (cross-) power spectra

$$P_{xy}(f) = \frac{1}{n} \sum_{k=0}^{n-1} x_k(f) \cdot \bar{y_k}(f)$$

computed over FFT coefficients $x_k(f) = FFT(a_k(t) \cdot h(t))$ and $y_k(f) = FFT(b_k(t) \cdot h(t))$ using the standard von-Hann window $h(t) = \frac{1-\cos(2\pi t/w)}{2}$ That is, it is computed as the power spectrum correlation between two signals split into n (optionally overlapping) averaged slices a_k and b_k of the signals a and b, respectively, normalized by the signal power spectra. Note that, although the signals a and b in time domain are real, their FFT coefficients x and y are complex. By using squared magnitudes, C_{xy} is also real-valued. By $\bar{x}$ we refer to the conjugate complex of x. Because the significance of coherence values depends on the number of averaged slices n – the more slices, the lower the coherence

[2] The authentication protocol is said to be computationally secure if an attacker's entropy of the key approaches the key length, which is typically 128 bits.

values are for the same signals –, we reduce longer time series to a maximum length of 3 seconds. This is a compromise between sufficient variability for robust classification and quick user interaction. The final value is computed simply by averaging up to a cut-off frequency f_{max}

$$C_{xy} = \frac{1}{f_{max}} \int_0^{f_{max}} C_{xy}(f)\, df$$

With this heuristic, we threshold C_{xy} to create a binary decision of similarity for our authentication protocol. As explained below, our experiments have shown that, with a sampling rate of $r = 256\,\text{Hz}$ and windows of $w = 256$ samples with an overlap of 7/8 and a cut-off frequency of $f_{max} = 40\,\text{Hz}$, coherence provides good distinction between two devices being shaken by one person from two devices being shaken by two people, one each.

5.2 Quantized FFT Coefficients

Coherence is a powerful measure of similarity, but, due to its use of continuous values, does not lend itself to directly creating cryptographic key material out of its results. Keys must be bit-for-bit equal, and thus be based on discrete instead of continuous values. By retaining basic features of the coherence measure and condensing them into discrete feature vectors, we can use those for a different way of comparing two accelerometer time series. Coherence is based on FFT coefficients, so it seems logical to quantize them into discrete values.

Huynh and Schiele compared different features with different window sizes and found that pairwise adding of neighboring FFT coefficients and grouping into exponential bands performed best in recognizing activities with moderate to high intensity levels, while other features like pairwise correlation or spectral energy were worse [17]. They also reported that the highest FFT peaks could generally be found up to the tenth coefficient, which backs our own findings that coefficients above 20 Hz do not contribute significantly.

We compared four variants of FFT-based feature vectors: linearly or exponentially quantized coefficients used either directly of added pairwise. Our experiments have shown that pairwise added, exponentially quantized FFT coefficients performed best, as also suggested in [17]. When aiming for equivalence of feature vectors, there is however an additional complication: small differences of values near the boundaries of quantization bands can lead to different feature values, although the FFT coefficients are only marginally different. Our solution is to quantize each FFT vector into multiple *candidate* feature vectors with different offsets. These offsets range from 0 to the value of the smallest quantization band. The similarity criteria in this case is simply the percentage of matching candidate feature vectors out of all vectors sent to another device. Thresholding this percentage produces a binary decision for the authentication protocol. We achieved best results for distinguishing shaking by one person from shaking by two people, one device each, with $b = 6$ exponentially scaled bands for quantization, $k = 4$ candidates, and a cut-off frequency of $f_{max} = 20\,\text{Hz}$ at a sampling rate of $r = 512\,\text{Hz}$ with FFT windows of $w = 512$ samples, overlapping by 50%.

6 Authentication Protocols

The two feature vectors generated in task 4 constitute, if equivalent, a shared secret password. This shared string is not directly suitable to act as a secret key for cryptographic primitives, because it is neither of defined length (e.g. 128 bits) nor distributed uniformly. But it is possible to create a cryptographically secure secret key via interactive protocols, authenticated by the feature vectors.

The choice of features directly influences requirements on the cryptographic protocols. To compute the coherence measure, both vectors need to be available completely to both devices.[3] Therefore, the time series must be exchanged during the interactive protocol — in a way that does not reveal them to an attacker. Our first authentication protocol uses asymmetric cryptography to achieve this.

Feature vectors composed of quantized FFT coefficients, on the other hand, do not allow for additional differences — authentication should only proceed if both vectors are bit-for-bit equal. The advantage is that cryptographic key material can be created using only symmetric cryptography, which is more suitable for embedded devices.

For the formal descriptions of our protocols, we use the following notation: $c = E(K, m)$ describes the encryption of plain text m under key K with a symmetric cipher, $m = D(K, c)$ the corresponding decryption, $H(m)$ describes the hashing of message m with some secure hash, and $m|n$ the concatenation of strings m and n. The notation $M[a : b]$ is used to describe the substring of a message M starting at bit a and ending at bit b. The symbol $\oplus$ describes bit-wise XOR and $|S|$ the number of elements in a set S. If a message M is transmitted over an insecure channel, we denote the received message $\widetilde{M}$ to point out that it may have been modified in transit, by noise or attack. C refers to some publicly known constant. We use AES as a block cipher for E and D and SHA_{DBL}-256 as a secure hash for H, which is a double execution of the standard SHA-256 message digest to safeguard against length extension and partial-message collision attacks [18] and is defined as $\text{SHA}_{\text{DBL}}\text{-256} = \text{SHA-256}\left(\left(\text{SHA-256}\left(m\right)\right)|m\right)$.

6.1 Protocol 1: Diffie-Hellman and Interlock*

Fig. 3 shows our first authentication protocol, which is based on a standard Diffie-Hellman (DH) key agreement (introduced in their seminal article [19]) followed by an exchange of the condensed time series and comparison locally at each device.

Using DH key agreement, devices A and B generate two – supposedly – shared keys K^{Auth} and K^{Sess}, where it is impossible to infer one from the other (under the assumption that the hash function does not allow to find a pre-image). Creating two keys, one for authentication, one as session key, provides forward secrecy. Because DH is susceptible to MITM, the devices need to verify that their keys are equivalent. The unique key property of DH guarantees with a

[3] For security reasons, both devices should independently decide if authentication was successful, and thus both need to compute the coherence.

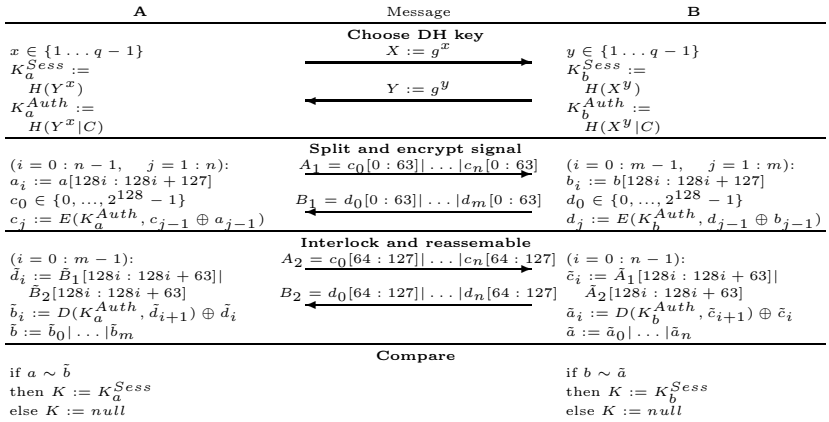


Fig. 3. Protocol 1: Diffie-Hellman key agreement followed by exchange of the complete time series via interlock*

very high probability, that, if $K_a^{Auth} = K_b^{Auth}$, there can be no attacker E with $K_{e1}^{Auth} = K_a^{Auth}$ and $K_{e2}^{Auth} = K_b^{Auth}$, and subsequently, no $K_{e1}^{Sess} = K_a^{Sess}$ and $K_{e2}^{Sess} = K_b^{Sess}$.

This verification is done with an extended *interlock* protocol. Interlock [20] is not used widely, but is an efficient (in terms of message length) method to verify that two parties share the same key. By using this key as an input to a block cipher and splitting packets in halves, a MITM can only decrypt these packets after having received both halves. The interlock protocol then demands that A and B will only send their second halves after they have received the first halves from the respective other side. This has the effect that both sides must commit themselves to their values, by sending the first halves of the encrypted blocks, before they can receive, and subsequently decrypt, the other side's message. Thus, interlock can be seen as a commitment scheme (see e.g. [21] for a definition) based on block ciphers. An attacker E is now left with only two options: either to forward the original packets, or to create packets on its own. In the former case, A and B will be unable to decrypt the messages properly, because they do not share the same key. In the latter case, E must guess the contents of the messages, and encrypt them with the appropriate keys, before it has access to the actual messages. When the messages sent by A and B have an entropy of e bits. this leaves E with a single 2^{-e} chance of remaining undetected.

The original version of interlock is suitable for messages the size of the cipher block length. Because in our case the vectors of the accelerometer sensor data, condensed into a time series of magnitudes, have arbitrary length, we introduce a slightly extended protocol that we call *interlock**. In this variant, A and B encrypt their complete messages, i.e. the (zero-padded) vectors a and b with lengths of n and m blocks, respectively, with any of the well-known block cipher modes. For our motion authentication protocol, we simply use the cipher block chaining (CBC) mode with a random initialization vector (IV). The resulting

cipher texts c and d with lengths of $n+1$ and $m+1$ blocks are then split into two messages by concatenating the first halves of all cipher blocks into the first messages A_1 and B_1 and the second halves of all cipher blocks into the second messages A_2 and B_2. This ensures that E can not decrypt any of the blocks, and can therefore not even learn parts of the plain text messages.

After exchanging their messages a and b, A and B verify that $a \sim b$, that is, that they are similar enough under their chosen criteria. We use coherence as described in section 5, but other suitable features can be used without changes to the protocol. Because of this possibility, we do not try to minimize the message lengths as e.g. suggested in [13]. In fact, A and B could use completely different similarity criteria, and could still authenticate using the same protocol. This is important for practical implementations, because different generations of devices will need to be compatible with each other.

The MANA III scheme described in [10] serves a similar purpose as this protocol, but using different cryptographic primitives. While we employ a block cipher, the MANA III scheme uses a MAC. Both constructions build on a mutual commitment to an authenticator string before transmitting parts of it.

6.2 Protocol 2: Candidate Key Protocol

In our second protocol, which we call the *candidate key protocol* (CKP), the shared secret key is generated from sensor data instead of by DH. As depicted in Fig. 4, feature vectors v are hashed to generate *candidate key parts* h. If the feature extraction task produces multiple "parallel" feature vectors v^i for each time window, as suggested above in section 5, then these yield multiple candidate

A	Message	**B**
$s_a^{r_a} \in \{0,\ldots,2^{128}-1\}$	$C_a^{r_a} :=< r_a, s_a^{r_a}, h_a^1, h_a^2, \ldots >$ →	$s_b^{r_b} \in \{0,\ldots,2^{128}-1\}$
For each vector v_a^i:		**For each vector** v_b^j:
$h_a^i := H(s_a \vert v_a^i)$	$C_b^{r_b} :=< r_b, s_b^{r_b}, h_b^1, h_b^2, \ldots >$ ←	$h_b^j := H(s_b \vert v_b^j)$
add $< r_a, v_a^i >$ to LH_a		add $< r_b, v_b^j >$ to LH_b
$r_a := r_a + 1$		$r_b := r_b + 1$
For each hash $\tilde{h}_b^j$:		**For each hash** $\tilde{h}_a^i$:
if $\exists < \hat{r}_a, \hat{v}_a^i > \in LH_a$		if $\exists < \hat{r}_b, \hat{v}_b^j > \in LH_b$
s.t. $\tilde{h}_b^j = H(\tilde{s}_b^{r_b} \vert \hat{v}_a^i)$		s.t. $\tilde{h}_a^i = H(\tilde{s}_a^{r_a} \vert \hat{v}_b^j)$
then add $< \hat{r}_a, \hat{v}_a^i >$ to $MC_{a,b}$		then add $< \hat{r}_b, \hat{v}_b^j >$ to $MC_{b,a}$
if $\vert MC_{a,b} \vert >= N_a$	$K_a :=< N_a, s_a^{r_a}, kh_a >$ →	if $\vert MC_{b,a} \vert >= N_b$
then $kh_a := H(s_a^{r_a} \vert last_{N_a}(MC_{a,b}))$		then $kh_b := H(s_b^{r_b} \vert last_{N_b}(MC_{b,a}))$
and $k_a := H(s_a^{r_a} \vert last_{N_a}(MC_{a,b}) \vert C)$	$K_b :=< N_b, s_b^{r_b}, kh_b >$ ←	and $k_b := H(s_b^{r_b} \vert last_{N_b}(MC_{b,a}) \vert C)$
if $\exists \hat{c}_a \subseteq MC_{a,b}$	ACK kh_b →	if $\exists \hat{c}_b \subseteq MC_{b,a}$
s.t. $H(\tilde{s}_b^{r_b} \vert \hat{c}_a) = \widetilde{kh}_b$		s.t. $H(\tilde{s}_a^{r_a} \vert \hat{c}_b) = \widetilde{kh}_a$
then $k'_b := H(\tilde{s}_b^{r_b} \vert \hat{c}_a \vert C)$	ACK kh_a ←	then $k'_a := H(\tilde{s}_a^{r_a} \vert \hat{c}_b \vert C)$
if k_a is not set $k := k'_b$		if k_b is not set $k := k'_a$
if k'_b is not set $k := k_a$		if k'_a is not set $k := k_b$
if $kh_a = \widetilde{kh}_b$		if $kh_b = \widetilde{kh}_a$
then $k := k_a$		then $k := k_b$
else $k := k_a \oplus k'_b$		else $k := k_b \oplus k'_a$

Fig. 4. Protocol 2: candidate key protocol for directly creating a secret key from common feature vector hashes

key parts h^i. The one-way hashes are a simple way to communicate that a device has generated a certain feature vector without revealing it. To make dictionary attacks harder, we use the standard method of prepending random salt values s before hashing. When B receives such candidate key parts from A, it can check its own history of recently generated feature vectors LH to check for equals. When B has generated the same feature vector, it is stored in a list of *matching key parts* MC specific to each communication partner. As soon as enough entropy has been collected in this list, B concatenates all feature vectors, appends C, hashes the resulting string, and sends a *candidate key* K to A. If no messages have been lost in transit, A should be able to generate a key with the same hash, and thus the same secret key, which it acknowledges to B. If messages have been lost, A can simply ignore a candidate key and create its own later on.

CKP is again a general protocol and can be used with any feature vectors. Here we apply it to quantized FFT coefficients, which work well for accelerometer data. A more thorough analysis of CKP itself will be provided separately.

7 Experimental Evaluation

We conducted three experiments, two to optimize parameters for the feature extraction tasks described in section 5, and one to validate our assumption of ease of use. All three experiments used four simple ADXL202JE accelerometers, two on each device, mounted at an angle of 90° so that all three dimensions could be measured with a maximum acceleration of 2 g. The accelerometers are fixed with compressed foam inside ping-pong balls (see Fig 5), and sampled at roughly 600 Hz. By choosing balls as "device" shapes and orienting the accelerometers randomly inside the balls, each data set has different orientations. The subjects were also asked to pick the devices up at the start of each sample, so that orientations change between samples. Although the accelerometers were wired to enable higher sampling rates, the attached cables were lightweight, flexible, and long enough so as not to disturb movements of subjects.

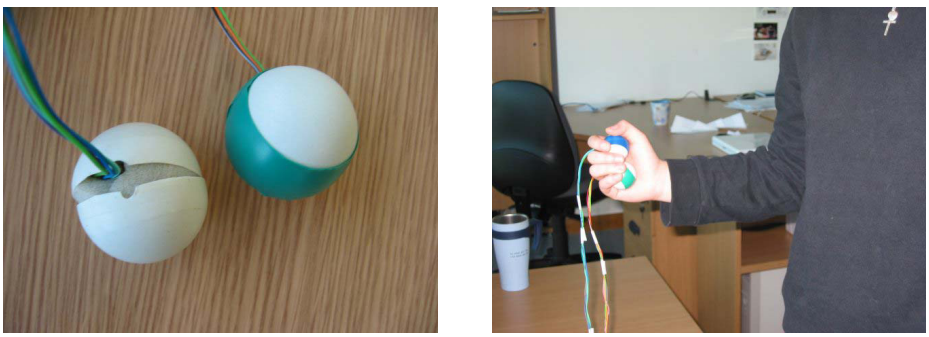


Fig. 5. Experimental setup: devices with accelerometers and subject during data collection

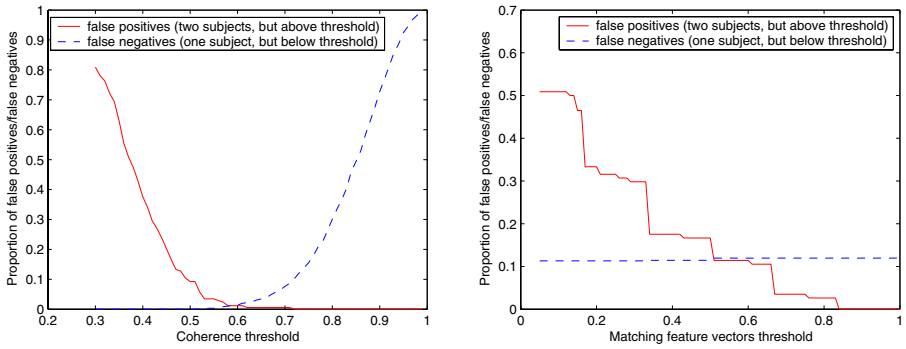
7.1 Experiment 1: Single Subjects Data Collection

The first experiment was explorative and aimed to discover how people typically shake small, lightweight objects. 51 people, 19 female aged between 20 and 55, 32 male aged between 20 and 58, of different professions, including cafeteria staff and other non-office workers, were asked to shake both ping-pong balls, explicitly without further instructions. For each subject, 30 samples of roughly 5 seconds were taken: 5 each with both balls in the left, both in the right hand, one ball in each hand, and while either standing or sitting. This extensive data set of 1530 samples shows surprisingly large differences in style, frequency, and vigor of the shaking patterns. Samples with both balls in one hand serve as our "positive" data set where authentication should be successful. The cases where one ball was shaken in each hand are "neutral": because a single person is performing the motion, authentication could, but does not have to succeed.

7.2 Experiment 2: Pairs Trying to "hack" Authentication

The second experiment served to establish our "negative" data set of cases where authentication should not be successful. It was organized as a competition with a small prize to motivate participants to try harder. The goal was for a pair of subjects to produce shaking patterns as similar as possible to each other. 8 different pairs contributed 8 complete data sets of 20 samples each and 4 incomplete sets with less samples: 5 samples each for both subjects using their left hands, both their right, one subject left, the other right, and vice versa. Each sample has roughly 15 seconds, because some time was allowed for starting the motion and synchronization. For more flexibility in moving together, the pairs were only standing but not sitting. Immediate feedback after each sample was provided to the pairs in the form of the similarity values for both protocols, so that they could adapt their shaking patterns appropriately for highest values.

Data from these two experiments was used to find parameters for detecting active segments for the temporal alignment task, and to optimize parameter combinations for the feature extraction task. The parameters for feature extraction reported in section 5 have been found by a full parameter search using this extensive data set. For coherence, we use the parameter combination that generates the maximum difference of coherence averages between all positive and negative samples. Due to the larger parameter search space with higher dimensionality, for the second protocol we use the combination that minimizes $4e_P + e_N$. e_P is the percentage of false positives, i.e. the number of successful authentications for pairs, and e_N is the percentage of false negatives, i.e. the number of authentication errors for both balls shaken in one hand. That is, false positives were weighted higher than false negatives. The values listed above in the respective sections produced optimal results on this data set. An explorative analysis of the results depending on these parameters shows that most of them are robust w.r.t. the difference in coherence averages. This suggests that even with suboptimal parameter combinations, which may be the case when using these values with different data sets, results should not deteriorate significantly.



(a) Protocol 1 (b) Protocol 2

Fig. 6. Thresholds for coherence and the number of matching FFT slices control the trade-off between false positives and false negatives w.r.t. all positive/negative samples

Figure 6 shows the trade-off between false positives and false negatives, depending on the thresholds. Unsurprisingly, the percentage of false positives decreases with increasing thresholds for both protocols. For protocol 1, shown in Fig. 6a, false negatives begin to increase noticeably at a threshold of around 0.6, while for protocol 2, shown in Fig. 6b, they remain nearly constant. The threshold, either for coherence or for the percentage of matching candidate feature vectors, can be set by the application, or possibly even by the user. From a security point of view, we obviously prefer to restrict the number of false positives to zero. With a coherence threshold of 0.72 and a threshold of 84% matching parts, we achieve false negatives rates of 10.24% and 11.96%, respectively, with no false positives. These false negatives are sufficiently low to provide user friendly interaction, as also shown by our third experiment. The feedback of a failed authentication is immediate, and users just need to shake the devices again.

There is room for improving the results for our first protocol using coherence. As explained in section 5, we only use 3 seconds for comparing the time series. If active segments are longer than this, we can choose freely which parts to use. Figure 7 shows the average coherence values for our "negative" data sets, depending on the offset of the compared time series parts. Number 1 corresponds to the first 3 seconds, number 2 to the time series between 3 and 6 seconds, etc. The graph shows that two people tend to loose synchronization the longer the common movement needs to be sustained. We could exploit this fact by skipping the first few seconds and comparing later parts, at the expense of forcing users to shake devices longer. The results given above were generated by taking the beginning of the active segments, and thus with the most difficult parts.

Data from the first experiment was also used to estimate the entropy of feature vectors used for our second protocol. Using the parameters found with the first two data sets but 256 Hz instead of 512 Hz sample rate, quantized FFT coefficient vectors were computed over all 1530 samples. This parameter combination generates feature vectors of 21 discrete values from 0 to 5. Each subject

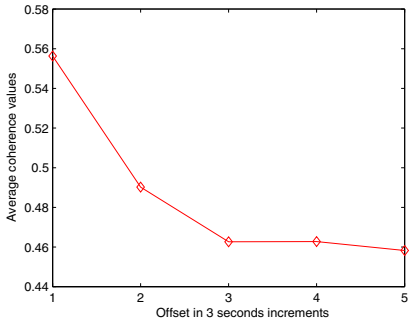


Fig. 7. Average coherence values depending on the segment offset show that people tend to synchronize their movements better at the beginning of their coordinated motions

generated on average 526.86 different feature vectors, with a minimum of 140 and a maximum of 1037. Aggregated over all subjects, there were 5595 different vectors for the left hands, 4883 for the right, and 7988 and 7770 for device 1 and 2, respectively. Overall, 12220 different feature vectors were generated during the first experiment, corresponding to an entropy of 13.58 bits per feature vector. If we assume an attacker to know which device, person, and hand are involved in a protocol run, this entropy decreases to around 7 to 10 bits, depending on the person. Overlapping feature vectors will have even less entropy, but we can still assume to generate at least 7 bits entropy per second using our second protocol.

7.3 Experiment 3: Single Subjects Live Usability Validation

The third experiment was run in "live" mode instead of data collection with batch processing, and used the same parameters. 30 subjects were asked to shake both devices in their dominant hand, with the aim of achieving successful authentication for both protocols. A simple GUI showed the status of both devices (active/quiescent) and the similarity values for both protocols, with green background if it was higher than the respective threshold and red for lower values. Subjects were asked to read a short list of tips for improving the similarity values (to align the devices roughly along the movement axis, to keep the wrist stiff, to shake quickly and vigorously, and to keep the elbow steady) and then to use interactive trial&error for achieving successful authentication. 8 of the subjects could immediately and reproducibly achieve this for both protocols starting with their first try, 8 subjects after at most 5, and 2 subjects after at most 10 tries. The remaining 12 subjects had more difficulty, but 7 could reproducibly achieve authentication after being shown once how others did it, and then within at most 3 further tries. 5 subjects only achieved authentication with either of the protocols, but not with both at the same time. This experiment shows that, even though the average rate of false negatives is low for the extensive data set from the first experiment, some people have more trouble to generate strong but similar movement patterns than others. Nonetheless, it also shows that the

method is easy enough to learn within a few minutes from printed instructions and trial&error, and that it can be used intuitively after this brief learning period. The fact that a few subjects performed significantly better after being shown suggests that the printed instructions need to be improved.

8 Conclusions

We have proposed two protocols for authentication based on accelerometer data that generate secret keys between two devices when they are shaken together. Although using similar techniques for accelerometer data analysis, it is evident that the protocols achieve their aim very differently, from a security as well as from a protocol point of view. We consider the first protocol more secure, but the second to be more scalable. That is, if a large number of devices are in range of the wireless network, a device using protocol 1 may need to run Diffie-Hellman key agreement with a considerable number of other devices to find that which it is shaken together with. For the second protocol, it only needs to broadcast its candidate key parts stream, and the matching device can "tune in", i.e. synchronize, to this key stream. On the other hand, the security level of our CKP-based protocol 2 is limited to the entropy of the feature vectors, and is susceptible to offline attacks. When (pessimistically) estimating the entropy rate at around 7 bits per second, 20 seconds of shaking should be sufficient to achieve a security level of 128 bits. Users or applications may choose lower security levels.

Another potential issue in terms of security of protocol 2 is that secure hash functions, the cornerstone of our design, have been subjected to considerably less theoretical analysis than the DH construction or block ciphers which are used in protocol 1. New attacks on hash functions are being discovered [22], although the SHA-256 family of hashes, including the even more conservative $\mathrm{SHA}_{\mathrm{DBL}}$-256, is still considered secure. Additionally, protocol 1 utilizes these well-studied cryptographic primitives within a conservative design. An attacker has a one-off chance for an online attack – to guess the whole time series – and is thus significantly less likely to be successful than an offline attack on protocol 2. Although we can not currently quantify the security level against such unlikely online attacks, the security level of protocol 1 against offline attacks is 128 bits even after only 3 seconds of shaking (assuming DH to be secure). By introducing two protocols with different design, application developers can decide on this well-known trade-off between security and performance according to their requirements. Protocol 2 offers benefits for devices with limited resources, large wireless networks, and quick interaction, while we recommend using protocol 1 for higher security demands.

Feature extraction and cryptographic protocols are mostly independent of each other. Improvements in feature extraction to generate higher entropy and/or be more robust against off-center rotational effects in the movements can be used without modifying the cryptographic protocols, with the potential to significantly increase the entropy rate and thus decrease shaking time. For protocol 1, such improvements can even be distributed independently while remaining

compatible to older devices. We note that our cryptographic protocols are also suitable for use with other types of sensors, while pre-processing and feature extraction tasks would most likely need to be modified.

Potential applications for our pairing protocols are manifold; coupling a mobile phone with a Bluetooth headset, establishing a transient secure connection between two smart cards for exchanging digital money, or passing access rights between key chains are prominent examples. 3D accelerometers are now being embedded into off-the-shelf mobile devices like the "Nokia 5550 Sport" and can immediately be used for authentication with our protocols. In our experiments described in section 7, we intentionally used simple, cheap accelerometers that are suitable for mass deployment.

The user interaction for authenticating devices is limited to just shaking them together for a few seconds, and is thus unobtrusive. By combining the explicit user interaction – taking two devices into one hand and shaking them as an indication that they should pair – with implicit authentication, we limit the burden placed on users. Connections are secured by default, not only as an option.

Full source code of our implementation including a demonstration application as well as our data sets are available as open source at `http://www.openuat.org`.

Acknowledgments

We gratefully acknowledge support by the Commission of the European Union under the FP6 Marie Curie Intra-European Fellowship program contract MEIF-CT-2006-042194 "CAPER", and especially thank Kristof van Laerhoven for insightful discussions on analyzing accelerometer data and providing sensor boards.

References

1. Hoepman, J.H.: The emphemeral pairing problem. In: Proc. 8th Int. Conf. Financial Cryptography, Springer-Verlag (2004) 212–226
2. Holmquist, L.E., Mattern, F., Schiele, B., A., P., Beigl, M., Gellersen, H.W.: Smart-its friends: A technique for users to easily establish connections between smart artefacts. In: Proc. UbiComp 2001, Springer-Verlag (2001) 116–122
3. Stajano, F., Anderson, R.: The resurrecting duckling: Security issues for ad-hoc wireless networks. In: Proc. 7th Int. Workshop on Security Protocols, Springer-Verlag (1999) 172–194
4. Kindberg, T., Zhang, K., Im, S.H.: Evidently secure device associations. Technical Report HPL-2005-40, HP Laboratories Bristol (2005)
5. Kindberg, T., Zhang, K.: Validating and securing spontaneous associations between wireless devices. In: Proc. ISC'03: 6th Information Security Conf., Springer-Verlag (2003) 44–53
6. McCune, J.M., Perrig, A., Reiter, M.K.: Seeing-is-believing: Using camera phones for human-verifiable authentication. In: Proc. IEEE Symp. on Security and Privacy, IEEE CS Press (2005) 110–124
7. Goodrich, M.T., Sirivianos, M., Solis, J., Tsudik, G., Uzun, E.: Loud and clear: Human verifiable authentication based on audio. In: Proc. ICDCS 2006: 26th Conf. on Distributed Computing Systems, IEEE CS Press (2006) 10

8. Nicholson, A.J., Smith, I.E., Hughes, J., Noble, B.D.: LoKey: Leveraging the sms network in decentralized, end-to-end trust establishment. In: Proc. Pervasive 2006, Springer-Verlag (2006) 202–219
9. Čagalj, M., Čapkun, S., Hubaux, J.P.: Key agreement in peer-to-peer wireless networks. IEEE (Special Issue on Cryptography and Security) **94** (2006) 467–478
10. Gehrmann, C., Mitchell, C.J., Nyberg, K.: Manual authentication for wireless devices. RSA Cryptobytes **7** (2004) 29–37
11. Castelluccia, C., Mutaf, P.: Shake them up! In: Proc. MobiSys 2005: 3rd Int. Conf. on Mobile Systems, Applications, and Services, ACM Press (2005) 51–64
12. Hinckley, K.: Synchronous gestures for multiple persons and computers. In: Proc. UIST '03: 16th ACM Symp. on User Interface Software and Technology, ACM Press (2003) 149–158
13. Lester, J., Hannaford, B., Borriello, G.: "Are you with me?" – Using accelerometers to determine if two devices are carried by the same person. In: Proc. Pervasive 2004, Springer-Verlag (2004) 33–50
14. Batina, L., Mentens, N., Verbauwhede, I.: Side-channel issues for designing secure hardware implementations. In: Proc. IOLTS: IEEE Online Testing Symp. (2005)
15. Lukowicz, P., Junker, H., Tröster, G.: Automatic calibration of body worn acceleration sensors. In: Proc. Pervasive 2004, Springer-Verlag (2004) 176–181
16. Aylward, R., Lovell, S.D., Paradiso, J.A.: A compact, wireless, wearable sensor network for interactive dance ensembles. In: Proc. BSN 2006: Int. Workshop on Wearable and Implantable Body Sensor Networks, IEEE CS Press (2006) 65–68
17. Huynh, T., Schiele, B.: Analyzing features for activity recognition. In: Proc. Soc-EUSAI 2005. ACM Int. Conf. Proceeding Series, ACM Press (2005) 159–163
18. Ferguson, N., Schneier, B.: Practical Cryptography. Wiley Publishing (2003)
19. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. on Information Theory **IT-22** (1976) 644–654
20. Rivest, R.L., Shamir, A.: How to expose an eavesdropper. Commununications of ACM **27** (1984) 393–394
21. Vaudenay, S.: Secure communications over insecure channels based on short authenticated strings. In: Proc. CRYPTO 2005, Springer-Verlag (2005)
22. Wang, X., Yin, Y., Yu, H.: Finding collisions in the full SHA-1. In: Proc. CRYPTO 2005, Springer-Verlag (2005)