ShakeCast: Using Handshake Detection for Automated, Setup-Free Exchange of Contact Data

Tim Weißker, Erdan Genc,
Andreas Berst, Frederik David
Schreiber, Florian Echtler
Bauhaus-Universität Weimar
Weimar, Germany
firstname.lastname@uni-weimar.de

Figure 1: Exchanging contact data during handshake gesture. Alignment of gravity vector along y-axis of accelerometer is used to detect initiation of handshake.

MobileHCI'17,, September 04–17, 2017, Vienna, Austria
ACM 978-1-4503-5075-4/17/09.
https://doi.org/10.1145/3098279.3122131

Abstract

We present ShakeCast, a system for automatic peer-to-peer exchange of contact information between two persons who just shook hands. The accelerometer in a smartwatch is used to detect the physical handshake and implicitly triggers a setup-free information transfer between the users' personal smartphones using Bluetooth LE broadcasts. An abstract representation of the handshake motion data is used to disambiguate between multiple simultaneous transmissions and to prevent accidental data leakage.

To evaluate our system, we collected individual wrist acceleration data from 130 handshakes, performed by varying combinations of 20 volunteers. We present a systematic analysis of possible data features which can be used for disambiguation, and we validate our approach using the most salient features. Our analysis shows an expected match rate between corresponding handshakes of 92.3%.

ACM Classification Keywords

H.5.m. [Information Interfaces and Presentation (e.g. HCI)]: Miscellaneous

Author Keywords

handshake; smartwatch; accelerometer; Bluetooth; contacts; contact data; smartphone; mobile; data exchange; peer-to-peer; P2P

Introduction

Exchanging contact information is a mundane task that people have attempted to simplify for centuries, first through physical items such as business cards and, more recently, through technological means such as data transfer via smartphones. For example, many modern mobile devices are capable of directly exchanging contact data when held back-to-back through an integrated NFC transceiver. Nevertheless, all these methods still require active participation of the user, which may become burdensome when many contacts are made in a context such as a business fair or conference.

However, it is customary in many cultures to briefly shake hands with the other person upon meeting. Based on this custom, we present ShakeCast, a system which uses the handshake as an implicit trigger to broadcast a small, user-selected information snippet containing contact information. An off-the-shelf smartwatch is used as a wrist-worn sensor to detect the motion made by the users' hands, and a regular smartphone then broadcasts the contact snippet to other devices within reception range while at the same time listening for the peer device's transmission. Note that in a scenario such as a business meeting, several handshakes may take place simultaneously within close proximity to each other. To ensure that contact data is only exchanged with those persons the user actually shook hands with, the broadcast data is hashed with a feature vector extracted from the motion data itself.

Related Work

The growing popularity of smartwatches allows for a variety of novel system interaction modalities like the recognition of and response to motion gestures. The *Office Smartwatch* [3], for example, recognizes hand turns to lock and unlock doors or knocking gestures to virtually announce the presence of a person in front of the door. Wilkinson et al. use a smartwatch to augment touch gestures on a tablet with additional degrees of freedom [11]. Xu et al. [13] have shown that gesture recognition with watches works even for finger gestures, while WatchMI [14] uses IMU data to augment touches on the watch with additional information such as pressure. Motion data can also be maliciously exploited to reconstruct text written on a whiteboard [1], typed PINs on the phone [9] or typed text on a keyboard [10].

The concept of gesture-based exchange of contact information was proposed by Hinckley [6] and first demonstrated with iBand [7], a custom bracelet-like device that is able to detect up-and-down movements and to exchange information via built-in infrared sensors. The ShakeOnit system [4] is capable of detecting various two-sided greeting gestures by using data gloves. Strategies for detecting handshakes using smartwatches combined with a subsequent data exchange have been shown by Augimeri et al. [2] and Wu et al. [12]. However, these strategies need the full accelerometer data of the other participating device to make a decision and hence require a pre-existing network connection between the devices. SyncTap by Rekimoto [8] purely relies on timing information of a simultaneous button press to pair two devices on the same network.

A promising approach to handshake-triggered data exchange was presented by Aimee Ferouge [5], who introduced peakmaps to resolve ambiguities when more than a single pair of people shake hands in close proximity. Although this doesn't require the full accelerometer data of the other side, there still is the need for an initial data exchange before decisions can be made. In our system, we aim at making all decisions about detection and disambiguation locally, i.e. without any accelerometer data from other devices.

ShakeCast

Our system was designed to combine and improve the characteristics of the works presented above. In particular, ShakeCast should be able to: a) detect handshakes using commodity smartwatches without any custom hardware modifications, b) correlate matching handshakes based on locally available data only (i.e. no prior data exchange), c) exchange contact information when a matching handshake was successfully detected - without the need for any additional infrastructures (peer-to-peer), and d) disambiguate between multiple arriving contact data from people in close proximity.

We used two LG G Watches running Android Wear 1.5 as wearable sensors and paired each to a corresponding smartphone. In our test setup, we used two Motorola Moto E (2nd generation) running Android 6.0 as peer devices. The smartphone and the smartwatch were connected via Bluetooth LE using the Android Wear framework. The smartphones exchange contact data via BTLE advertisements.

Handshake Detection

During an initial video analysis, we found that a typical handshake consists of three consecutive phases. In the *invitation* phase, one of the participants raises the forearm to a horizontal orientation with the hand being open. Accepting the invitation, the partner performs the same arm movement and grabs the inviting hand. Then, the *oscillation* phase follows, where both partners move their hands up and down. Eventually, in the *finalization* phase, the hands are released and the forearms of both participants move back to their initial positions.

In our system, we use these distinct phases to automatically detect a handshake using the accelerometer data of a commodity smartwatch worn at the wrist of the shaking hand. As also illustrated in figure 1, the gravity component of the accelerometer data shifts mostly to the y-axis in the invitation phase and back out in the finalization phase. These events serve as start and end triggers for the deeper analysis of accelerometer data recorded between the trigger events, which will be referred to as a data packet. The exact gravity force threshold for the y-axis was set to $7m/s^2$. This approach allowed us to reduce the battery consumption by discarding irrelevant gestures directly on the smartwatch. A lower sample rate could also be used before initiation to further conserve power.

To detect the characteristic handshake oscillation movement on the y-axis, we move a window of width w through the data values and check for acceleration values of a minimum intensity i_{min}; if the longest streak of these positive windows exceeds a threshold of n, a handshake was detected in this data packet. Using the aforementioned hardware operating at 100 Hz, we empirically found $w = 20$, $i_{min} = 15m/s^2$ and $n = 6$ to be suitable values. In order to reduce the analysis effort for data packets triggered accidentally by non-shake motions, we require an incoming packet to be between a minimum ($l_{min} = 100$) and maximum sample length ($l_{max} = 700$), i.e. between 1 and 7 seconds.

For the handshake matching mechanism explained below, we require a reliable labelling of the oscillation region's start within the analyzed data packet. To this end, we use the first minimum or maximum after the start of the first positive window. Figure 2 shows an exemplary handshake plot, in which the longest positive window streak and the first local minimum within it is visualized.

Peer-to-Peer Data Exchange

When a handshake was detected, the system exchanges contact data between the two smartphones using Bluetooth Low Energy due to its ability to broadcast small

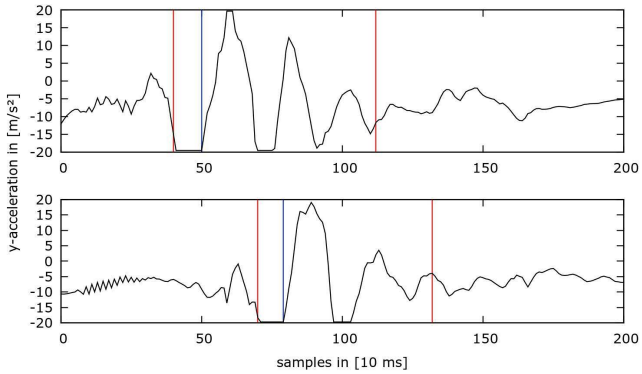

Figure 2: An exemplary y-acceleration plot of a handshake from both participants. The red (outer) lines indicate start and end of the longest positive window streak; the blue (inner) line is the first local extremum within this extracted range.

advertisement packets to all nearby devices without any prior setup. A user-configured URL (e.g. pointing to the user's homepage or business profile) is shortened using the bit.ly service to account for the highly restricted payload size of at most 31 bytes in the advertisement broadcast, and is then sent out to peer devices as manufacturer-specific data with a custom payload ID. At the same time, each device listens for incoming broadcasts with matching payload ID and saves the received URL into a list of recent contacts while indicating successful reception through a brief vibration.

However, when multiple pairs of people are shaking hands in close proximity, a single mobile device may receive several broadcasts, i.e. also contacts of people that the owner did not actually shake hands with. To address this issue, we use the features described in the next section to derive a code with which the payload is hashed (using XOR) before being sent to the other devices. Ideally, only the device with the matching handshake data can reconstruct this code and therefore decode the payload. Please note that this is just a disambiguation mechanism and must not be considered an encryption in the cryptographic sense. However, since the intended usage of our system is to only transfer publicly-available data such as the user's homepage URL, this does not pose a major privacy issue (although some form of contact network analysis may be possible nevertheless).

We briefly evaluated wireless signal strength as an alternative disambiguation mechanism. However, even preliminary tests showed that the physical distance itself has far less influence on the weak BTLE signal than other confounding factors, such as placement of the phone in the front or back pocket, body orientation etc. We therefore focused on the feature-based approach described below.

Handshake Matching

As a data basis for our analysis, we collected data from 130 handshakes among varying combinations of 20 volunteers. Both participants wore the smartwatch on their dominant hand, and all data between the start and end labels were logged on the smartphone, resulting in 260 unique acceleration traces. In order to generate the same code for hashing on both devices out of slightly different acceleration curves, we need to determine features which behave similar, given two related sets of accelerometer data (x- ,y-, and z-axis over a data packet identified by the algorithm described above). We considered the Fast Fourier Transformation (FFT) of the signal as most important set of features because of its time invariant and noise filtering properties.

The features we take into consideration are a variety of FFT values, the mean value of the signal and the amount of zero crossings for the data obtained in the handshake window. For an exemplary FFT window size of 65, we denote the resulting

32 distinct FFT values for the seven channel combinations as FFT_{i0} to FFT_{i31} for $i \in \{X, Y, Z, XY, XZ, YZ, XYZ\}$, where XY is the magnitude of the (combined) accelerometer channels: $XY_j = \sqrt{x_j^2 + y_j^2}$.

Due to inherent noise, the data packets for two matching handshakes will have different length. We address this by using a fixed window size for feature extraction. An optimal window size can be estimated by iterating through variable window sizes and looking for maximum correlation in our sample data. We evaluated window sizes in the range between 50 and 100 samples with a step size of 5. The lower bound was chosen to get a representative sample of the oscillation phase and not only one peak, the upper bound so that not too much non-handshake data will be taken into account. The latter would happen if the oscillation phase were shorter than the window size.

To analyze the data, we generated virtual instances of handshakes by combining all the individual data recordings in pairs and looking at the difference in feature values. 260 individual traces combine into $\frac{n(n-1)}{2} = 33670$ instances, 130 matching and 33540 non-matching. If multiple non-matching instances were generated where the handshake partners were the same persons, we removed all but one instance to avoid biasing the data towards specific persons. This information was only partly available, resulting in a total number of 16589 non-matching instances.

The goal was now to find the features that are most suitable to correctly classify the instances regarding their target classes. A perfect feature will have a minimal difference within in the set of matching and maximal difference within the set of non-matching instances. To measure this quality, we used the Pearson product-moment correlation coefficient, also called Pearson's r, between the individual feature and the target vector. After running our analysis on the sample data, we found the highest importance for the features of FFT_{Y2}, FFT_{Y3} and FFT_{XYZ0} with a window size of 65 (i.e. 0.65 seconds), as illustrated in figure 3.

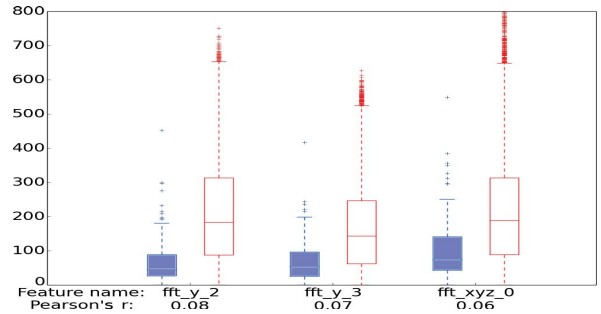

Figure 3: The three most important features, ordered by Pearson's r with their distribution of differences. Blue filled boxes show the difference distribution in the set of matching instances, red empty boxes in the set of non-matching instances.

Using these features, we created a simple encoding scheme that takes an incoming contact broadcast and a recent local handshake data packet. A successful decoding step shows that these are in fact data from one physical handshake, while a failure to decode shows they are part of different handshakes. We divide the value range of a feature into 7 equally sized successive regions (bins) and assign a bit sequence to each bin, which is simply the binary representation of the bin number. The concatenation of all bit sequences resulting from each feature value's bin is then used as a hash value to encode the data in the contact broadcast.

However, due to the large amount of noise present in the data, the local representation of the handshake may have sufficiently different feature values so that they fall into

different bins than those used in the broadcast, even if they have been computed from the same physical handshake. In order to allow for some margin of error, multiple hashes are generated by also considering one or more adjacent bins for each feature. If any hash succeeds in decoding the broadcast data (i.e. the result is a valid HTTP URL), it is accepted as a match. Consequently, the decoding can always take up to $Dec_{max} = (2a + 1)^{N_f}$ steps, where a denotes the accepted bin difference and N_f the amount of features we select. Fortunately, the computational cost of even multiple decoding attempts is negligible. Of course, this approach raises the false-positive rate, as discussed in the following section.

Evaluation

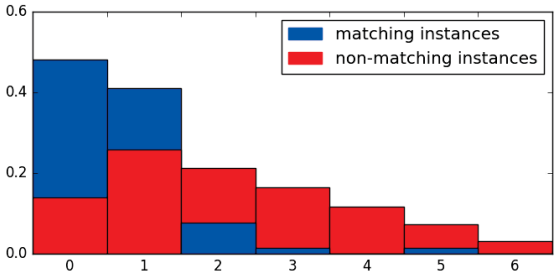

Figure 4: Histograms showing the bin distance of the matching instances in blue and the non-matching instances in red for feature FFT_{Y2}.

As can be seen in figure 4, for the strongest feature FFT_{Y2} only about 50% of all actually matching shakes fall into the same bin, but the vast majority of the other matching pairs fall in an adjacent bin.Therefore, an accepted bin difference of at least 1 should be chosen. From a usability point-of-view, the worst case for the classifier would be a false negative, as the users would have to shake hands again to complete the exchange. False positives, on the other hand, could be reduced using additional non-feature-based approaches such as evaluating Bluetooth signal strength.

		Predicted	
N_f, a	Actual	Positive	Negative
$N_f = 2, a = 1$	Positive	76.15%	23.85%
	Negative	21.17%	78.83%
$N_f = 3, a = 1$	Positive	57.69%	42.31%
	Negative	11.23%	88.77%
$N_f = 2, a = 2$	Positive	96.92%	3.08%
	Negative	46.95%	53.05%
$N_f = 3, a = 2$	Positive	92.31%	7.69%
	Negative	33.68%	66.32%

Table 1: Confusion matrices illustrating the proportions of (in-)correctly matched handshakes for varying amounts of selected features N_f and accepted bin difference a.

Table 1 shows us the different accuracies we can achieve for different a and N_f values where N_f follows the order of the most valuable attributes established in Figure 3. Using only the two most salient feature FFT_{Y2} and FFT_{Y3} provides nearly 77% true positive and 79% true negative matches with $Dec_{max} = 9$ decoding steps (2 features with 3 possible bins each). Adding the third feature FFT_{XYZ0} further improves the true negative rate but reduces the true positive rate by roughly 20%, an undesirable result ($Dec_{max} = 27$). Allowing a bin difference of two instead vastly reduces the false negatives rate. While this allows us to accept nearly all matches, we have more than doubled our false positive rate ($Dec_{max} = 25$).

Using both bin difference $a = 2$ and including the third feature gives us better values in true positives and in true

negatives than both variants with $a = 1$ and far better true negative values than with two features for 4% worse true positive values. This approach provides the best-balanced solution, although the decoding process for every received handshake will now take up to $Dec_{max} = 125$ steps. However, even in the worst case with up to 125 iterations, the decoding process still takes only a negligible amount of time due to the tiny amount of encoded data.

Discussion & Future Work

We have presented ShakeCast, a method to automatically exchange contact data between two persons shaking hands. We contribute a peer-to-peer approach to identify the correct handshake among multiple candidates, a method to encode contact data into space-constrained BTLE broadcasts, and an evaluation of our approach on a dataset of real-world handshakes.

During our data collection process, we observed that the motions for the same handshake between two persons can vary widely (e.g. "fishy" vs. "firm" handshake). This often leads to huge differences in the accelerometer output. Therefore, the generation of an unique and robust identifier based just on the accelerometer data is a hard problem. Our best-balanced solution will not recognize about 7.7% of handshakes in a scenario with multiple simultaneous transmissions, while mistakenly collecting about 33.7% of non-matching transmissions. In the future, we will use more advanced analysis methods such as PCA, in combination with a k-fold validation approach, to improve our matching algorithm.

Another possible solution might be to use additional sensors to get more accurate information about the actual handshake. For example, an approach we consider promising would be the use of a smart ring. A finger-worn sensor would provide the opportunity to exclude the additional movement in the wrist joint from the accelerometer data.

One drawback of our approach is the fact that the usage of a smartwatch is not ideal for this application as the watch has to be worn on the shaking hand. This conflicts with the habit that most people usually wear their watch on the non-dominant hand while using the dominant one for shaking hands. However, the increased popularity of fitness trackers which are also worn on the dominant hand may provide an opportunity to detect handshakes without forcing the users to change their habits.

REFERENCES

1. L. Ardüser, P. Bissig, P. Brandes, and R. Wattenhofer. 2016. Recognizing text using motion data from a smartwatch. In *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. 1–6. DOI: http://dx.doi.org/10.1109/PERCOMW.2016.7457172
2. A. Augimeri, G. Fortino, M. R. Rege, V. Handziski, and A. Wolisz. 2010. A cooperative approach for handshake detection based on body sensor networks. In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*. 281–288. DOI: http://dx.doi.org/10.1109/ICSMC.2010.5641696
3. Yannick Bernaerts, Matthias Druwé, Sebastiaan Steensels, Jo Vermeulen, and Johannes Schöning. 2014. The Office Smartwatch: Development and Design of a Smartwatch App to Digitally Augment Interactions in an Office Environment. In *Proceedings of the 2014 Companion Publication on Designing Interactive Systems (DIS Companion '14)*. ACM, New York, NY, USA, 41–44. DOI: http://dx.doi.org/10.1145/2598784.2602777

4. David Cranor, Amanda Peyton, Arlene Persaud, Rajiv Bhatia, Sinbae Kim, and V. Michael Bove. 2011. ShakeOnit: An Exploration into Leveraging Social Rituals for Information Access. In *Proceedings of the Fifth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '11)*. ACM, New York, NY, USA, 277–278. DOI: http://dx.doi.org/10.1145/1935701.1935761

5. Aimee Ferouge. 2015. *Handshake Recognition Applied to Wireless Data Exchange in Smartbands*. Master's thesis. Delft University of Technology, The Netherlands.

6. Ken Hinckley. 2003. Synchronous Gestures for Multiple Persons and Computers. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology (UIST '03)*. ACM, New York, NY, USA, 149–158. DOI: http://dx.doi.org/10.1145/964696.964713

7. Marije Kanis, Niall Winters, Stefan Agamanolis, Anna Gavin, and Cian Cullinan. 2005. Toward Wearable Social Networking with iBand. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems (CHI EA '05)*. ACM, New York, NY, USA, 1521–1524. DOI: http://dx.doi.org/10.1145/1056808.1056956

8. Jun Rekimoto. 2004. SyncTap: synchronous user operation for spontaneous network connection. *Personal and Ubiquitous Computing* 8, 2 (2004), 126–134.

9. A. Sarkisyan, R. Debbiny, and A. Nahapetian. 2015. WristSnoop: Smartphone PINs prediction using smartwatch motion sensors. In *Information Forensics and Security (WIFS), 2015 IEEE International Workshop on*. 1–6. DOI: http://dx.doi.org/10.1109/WIFS.2015.7368569

10. He Wang, Ted Tsung-Te Lai, and Romit Roy Choudhury. 2015. MoLe: Motion Leaks Through Smartwatch Sensors. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (MobiCom '15)*. ACM, New York, NY, USA, 155–166. DOI:http://dx.doi.org/10.1145/2789168.2790121

11. Gerard Wilkinson, Ahmed Kharrufa, Jonathan Hook, Bradley Pursglove, Gavin Wood, Hendrik Haeuser, Nils Y. Hammerla, Steve Hodges, and Patrick Olivier. 2016. Expressy: Using a Wrist-worn Inertial Measurement Unit to Add Expressiveness to Touch-based Interactions. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 2832–2844. DOI: http://dx.doi.org/10.1145/2858036.2858223

12. Fang-Jing Wu, Feng-I Chu, and Yu-Chee Tseng. 2011. Cyber-physical Handshake. In *Proceedings of the ACM SIGCOMM 2011 Conference (SIGCOMM '11)*. ACM, New York, NY, USA, 472–473. DOI: http://dx.doi.org/10.1145/2018436.2018527

13. Chao Xu, Parth H. Pathak, and Prasant Mohapatra. 2015. Finger-writing with Smartwatch: A Case for Finger and Hand Gesture Recognition Using Smartwatch. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications (HotMobile '15)*. ACM, New York, NY, USA, 9–14. DOI: http://dx.doi.org/10.1145/2699343.2699350

14. Hui-Shyong Yeo, Juyoung Lee, Andrea Bianchi, and Aaron Quigley. 2016. WatchMI: Pressure Touch, Twist and Pan Gesture Input on Unmodified Smartwatches. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '16)*. ACM, New York, NY, USA, 394–399. DOI: http://dx.doi.org/10.1145/2935334.2935375