# Gesture Recognition Implemented on a Personal Limited Device.

Benoit Ducray*, Sheila Cobourne*, Keith Mayes* and Konstantinos Markantonakis*
*Smart Card Centre, Information Security Group (SCC-ISG)
Royal Holloway, University of London, Egham, Surrey, TW20 0EX
Email: {Benoit.Ducray.2013, Sheila.Cobourne.2008, Keith.Mayes, K.Markantonakis}@rhul.ac.uk

***Abstract*—For a biometrics system, one of the principal challenges is to protect the biometric reference template, as if a malicious individual is able to obtain this template, the genuine user would not be able to reuse the biometric for any application. A solution may be to use a new form of authentication based on gesture recognition. This type of authentication has the added advantage that in the case of compromise, the gesture can be changed yet still retain the advantages of the biometric input. In this paper, we investigate whether it is feasible to implement a Gesture Recognition system on a personal limited device such as a smart card. To do this, we set out an experiment using sample gestures based on practical results of gesture authentication trials and an optimised version of Dynamic Time Warping (DTW) algorithm to analyse the data captured. We implemented them on both a contact Smart Card (SC) and the more powerful Samsung Galaxy S4 mobile phone, using Host Card Emulation (HCE). The result of this experiment was that it would take around a minute for the SC and a second for HCE.**

## I. Introduction

One of biometrics systems' principal challenges is to protect the biometric reference template. If a malicious individual is able to obtain the template, that would mean the genuine user would not be able to reuse this biometric for any application. A proposed solution used a cancelable biometric [1], but this method does not protect when the malicious individual gets access to the original biometric template. To address this issue, a new form of authentication based on gesture recognition has been proposed [2], [3], [4]. Gestures are not usually aimed at high security applications, but as convenient alternatives to simple PIN or password entry. However, depending on the method and precision of capture, gestures can include some biometric related characteristics as well as the something-you-know, making them more like two-factor authentication inputs. This type of authentication has the added advantage that in the case of compromise, the gesture can be changed yet still retain the advantages of the biometric input. A prime location for the related reference template would be on a security evaluated Smart Card (SC), as it is tamper resistant, easy to carry and if used with Gesture Recognition, the system can provide a three-factor authentication method. If the matching processes could also be carried out on-card (Match-on-Card), then this provides additional protection, as the template does not need to leave the card during an authentication. It also protects against attacks on the implementation due to the tamper-resistance of the smart card chip.

Gesture Recognition is demanding in terms of computation power and memory storage, so this paper sets out to investigate whether it is feasible to implement a Gesture Recognition system on a personal device of limited capability, such as an SC. To do this, an experiment was performed using sample gestures based on practical results of gesture authentication trials which used depth cameras as sensors (i.e. the Kinect™ [5] and Leap Motion [6]) and the Dynamic Time Warping algorithm (DTW) [7], to analyse the captured data. We chose DTW because unlike other classifiers such as Hidden Markov Models or Neural Networks, DTW requires little or no training. [1] We varied the data length, number of frames and tracking points of the sample gestures, and implemented them on both a contact SC and the much more powerful Samsung Galaxy S4 mobile phone. The latter used Host Card Emulation (HCE) [9] to emulate the SC, and the DTW algorithm was optimised to minimise memory usage on both platforms.

The experiment showed that the implementation on an SC was slow, (in a excess of minute) and the HCE version was much faster (around 1s or 2s), although the overall processing time depended on the gesture data length. It should be noted that the test applications were implemented at the platform level, rather than in low-level native code which would have been much faster.

This paper is structured as follows: Section 2 presents some background information about Smart Cards, Host Card Emulation, Gesture Recognition and the comparator we use i.e. the DTW algorithm. In Section 3, we present the details of the experiment, gestures, hardware and the optimisation of the gesture comparator used as well as the results. In Section 4 we discuss the feasibility of using gesture recognition on a personal device of limited capabilities. The conclusion and future work appear in Section 5.

## II. Background

### A. Smart Card

A modern Smart Card (SC) consists of an integrated circuit incorporating various types of attack and tamper resistance, packaged and embedded within a card carrier [10]. The attack resistant capabilities can be formally certified by independent evaluation. An SC is able to store and protect modest amounts of data, carry out on-card processing (e.g encryption and

[1] A comparison of classification algorithms can be seen in [8].

mutual authentication) and can communicate with an SC reader, all via the embedded microcontroller chip [11]. An SC requires a reader to which it connects either with direct physical contacts (‘contact card’) or via very short range wireless (‘contactless card’). Since an SC can store secret identifiers securely and engage in cryptographically protected (challenge-response) protocols, SCs play a very useful role in secure authentication [12]. An SC facilitates high security protocols and processes in a very user friendly way that is both easy-to-use and offers tamper-resistant security.

Many research papers have proposed how to store biometric information on smart cards, for use in two factor authentication (e.g. [13]) or three factor authentication (e.g. [14]).

### B. Host Card Emulation

Host Card Emulation (HCE) is a technology which emulates an SC on mobile equipment using only software [9], and can be used with Near Field Communication (NFC) to emulate a contactless SC. Before HCE, all messages from a card terminal were routed to a hardware Secure Element (SE) in the mobile handset. HCE communicates directly with the mobile operating system, which decides if messages should be handled by a physical SE or a software application [15].

### C. Gesture recognition

In this paper, when we refer to a "gesture" we mean a set of frames and tracking points produced by our gesture capture devices. These elements are organised such that every frame contains the same number of tracking points.

- Frames: they represent the length of a gesture in time. The frequency generation of a new frame depends on the sensor: the Kinect™ software generates 30 frames per second [5], the Leap Motion software can produce frames up to a rate exceeding 100 frames-per-second [16].
- Tracking Points: they represent the features of the gesture. Movement is usually tracked in three dimensions, which means that for each point we have information on the horizontal, vertical and depth (x, y, z). Depending on the sensor used, there may be support for point tracking, for example Kinect™ is able to track points from the head to toe including the head, hands, hips, feet, etc [17], but it is also possible to use the raw image from the sensor to track other points. The Leap Motion device, which tracks and records hand movements, could use raw data to track more information about the hands, such as finger thickness.

Accelerometers installed in mobile devices can also be used to capture movement in three dimensions. This has led to research into authenticating an individual based on gestures performed while holding a mobile device [18], [19]. One of the drawbacks of the accelerometer is that it captures only one point and not physical information about the user. Depth cameras, on the other hand, can capture several points. Depth cameras include the Kinect™, which provides us with 20 3D skeleton tracking points; these points were used by [20] with the Nearest Neighbor Algorithm to identify walking subjects.

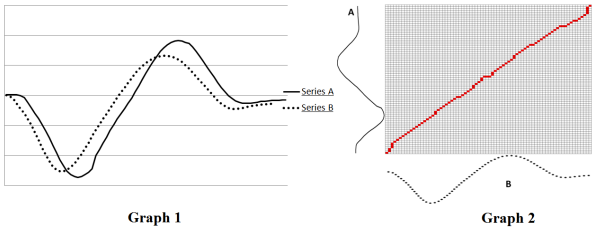


Figure 1. Graph 1 Two time series (A and B): Graph 2 The warping path between A and B obtained using the DTW algorithm

In [3], upper body parts recorded in the skeleton generated by the Kinect™ were used for authentication based on gesture recognition: using 6 of the available 20 skeleton tracking points, gave a True Positive Rate (TPR) of 93%, with 0% False Positive Rate (FPR) if the attacker [2] did not know the gesture and 1.7% of FPR once the attacker had seen the gesture. Other authors chose to use all 20 skeleton tracking points from the Kinect™ [4] which gave them a TPR of 98.11% for 1.89% of FPR. The Kinect™ was used with the DTW algorithm for analysis and recognition of 3D signatures [21]: here TPR was 99% and 1% FPR.

Hand gesture authentication, accuracy and attack resistance against shoulder surfing, were explored in [22]. In this experiment, reference hand gestures were recorded using a depth camera, filmed, and shown to a group of ‘attackers’: they were then asked to copy the gestures [22]; here the TPR was 96.6% and FPR was 2.2%.

### D. Comparator : Dynamic Time Warping

There are several algorithms that can be used for gesture recognition: for example, Dynamic Time Warping (DTW); Neural Network; or Hidden Markov Model. The Dynamic Time Warping (DTW) algorithm is frequently used to do the comparison, but some systems may instead use a mix of Bayes, Neural Network or Random Decision Forest [2]. In this paper we will focus on the DTW algorithm, because it requires little or no training. Other classification algorithms such as the Neural Network or the Hidden Markov Model need several examples from the user in order to get an accurate authentication rate. The DTW algorithm looks for an optimal alignment between two time-bound sequences, independently of the variation of speed or time between both sequences. Originally, this algorithm was used in speech recognition [23]. The interested reader is referred to other works that have used this method e.g. [24], [25], [26].

In practice, the principle of DTW is to define a warping path with the minimal cost. This cost is given by the cost function (or distance function) which is the distance (or the error) between the two sequences, as shown in Figure 1. DTW is reviewed in [7] and can be summarised as follows:

[2]“attacker”, in this paper, means an unauthorised person who copies a gesture with the aim to be authenticated

Table I
INFORMATION ON THE GESTURES AND APDU SENT.

| | Gesture size in bytes | Number of frames | Number of tracking points | Number of frames sent per APDU | Size of the APDU data | Number of APDU sent |
|---|---|---|---|---|---|---|
| Gesture 1 | 3240 | 90 | 6 | 1 | 36 | 90 |
| | | | | 6 | 216 | 15 |
| Gesture 2 | 1764 | 49 | 6 | 1 | 36 | 49 |
| | | | | 6 | 216 | 9 |
| Gesture 3 | 5940 | 90 | 11 | 1 | 66 | 90 |
| | | | | 3 | 198 | 30 |
| Gesture 4 | 3234 | 49 | 11 | 1 | 66 | 49 |
| | | | | 3 | 198 | 17 |

To use DTW to align two sequences A and B, where $A = (a_1, a_2, ..., a_N)$ of length $N \in \mathbb{N}$ (i.e a positive integer) and $B = (b_1, b_2, ..., b_M)$ of length $M \in \mathbb{N}$, we construct an N-by-M matrix where the $(i^{th}, j^{th})$ element of the matrix contains the distance $d\,(x_i, y_j)$ between the two points $x_i$ and $y_j$, using a distance function, generally the Euclidean distance, $d(x_i, y_j) = (x_i - y_j)^2$. Each element $(i, j)$ of the matrix corresponds to a hypothetical alignment between the points $x_i$ and $y_j$. From this matrix we can determine a warping path W where the $k^{th}$ element of $W$ is defined as $w_k = (i, j)_k$ so we have:

$$\begin{aligned} W =& w_1, w_2, \ldots, w_k, \ldots, w_K \\ & max(m, n) \leq K < m + n - 1 \end{aligned} \tag{1}$$

The warping path is typically subject to constraints on boundary conditions, continuity and monotonicity.

- **Boundary conditions**: $w_1 = (1, 1)$ and $w_k = (m, n)$, the warping path must start and finish in diagonally opposite corner cells of the matrix.
- **Continuity**: Given $w_k = (x, y)$ then $w_{k-1} = (x', y')$ where $x - x' \leq 1$ and $y - y' \leq 1$. Allowable steps in the warping path are restricted to adjacent cells (including diagonally adjacent cells).
- **Monotonicity**: Given $w_k = (x, y)$ then $w_{k-1} = (x', y')$ where $x - x' \geq 0$ and $y - y' \geq 0$. The points in W are forced to be monotonically spaced in time.

We are interested only in the path which minimises the warping cost:

$$DTW(AB) = min(\sqrt{\sum_{k=1}^{K} w_k}) \tag{2}$$

We can find this path using dynamic programming to evaluate the following recurrence which defines the cumulative distance $\gamma(i, j)$ as the distance $d(i, j)$ found in the current cell and the minimum of the cumulative distances of the adjacent elements:

$$\begin{aligned} \gamma\ (m; n) =& d(m; n) + min(\gamma\ (m - 1; n - 1); \\ & \gamma(m - 1; n); \gamma(m; n - 1)) \end{aligned} \tag{3}$$

Where: $\gamma\ (m; n)$ is an $(M + 1) \times (N + 1)$ matrix; $\gamma\ (0; n)$ and $\gamma\ (m; 0)$ are initialized with a large number representing infinity, or zero, depending on the application; $\gamma\ (0; 0)$ with zero; $d(m; n)$ is the cost function.

The cost of the minimal path between both sequences is contained at $\gamma(M; N)$.

The next section describes the performance evaluation experiments that were conducted.

## III. EXPERIMENT

In the work of [3], six of the available Kinect™ skeleton tracking points were used in a gesture authentication system with promising results, giving an Equal Error Rate (EER) of 2.8%. We devised a proof-of concept authentication experiment using a Leap Motion device, which tracks and records hand movements in three dimensions. For more technical information concerning the Leap Motion device please see [6]. Preliminary results from a small sample of volunteers indicated that it is feasible to use this device in gesture authentication systems although the EER is 11.88%.

For the performance evaluation in this paper, we used the Leap Motion to record a gesture of 90 frames with 11 tracking points (the five finger tips and roots and the palm centre) from which we truncated the floating numbers and encoded them all into two bytes.

We chose to emulate gestures from these two capture devices, setting the number of frames and tracking points in our sample gestures accordingly to reflect the different characteristics of the sensors. The DTW algorithm was used to analyse the gestures.

We are not assessing the performance of any cryptographic protocols because they would be the same for both SC and HCE.

### A. Gesture Data and Hardware

*1) **The gestures**:* We created four different sample gestures with varying memory requirements and processing time, described as follows:

- **Gesture 1**: This gesture represents the capture of six points in three dimensions and is composed of 90 frames. The total amount of data of this gesture is 3240 bytes. This gesture represents a three second gesture obtained with a device capturing at 30 frames per second. The

number of tracking points represents either the five fingertips and the palm centre (if performing hand gesture recognition), or both hands, elbows and shoulders for upper body gesture recognition.

- **Gesture 2**: This gesture captures six points in three dimensions and is composed of 49 frames. The total amount of data of this gesture is 1764 bytes. This gesture may represent a 1.63s gesture obtained with a device capturing at 30 frames per second. The tracking points are the same as in Gesture 1.
- **Gesture 3**: This gesture captures 11 points in three dimensions and is composed of 90 frames. The total amount of data of this gesture is 5940 bytes. Again, this gesture may represent a three second gesture obtained with a device capturing at 30 frames per second. The tracking points can represent the five fingertips, five finger roots and the palm centre for hand gesture recognition, or both feet, knees, hands, elbows, shoulders and the head for body gesture recognition.
- **Gesture 4**: This gesture captures 11 points in three dimensions and is composed of 49 frames. The total amount of data of this gesture is 3234 bytes. This gesture may represent a 1.63s gesture obtained with a device capturing at 30 frames per second. The tracking points are the same as in Gesture 3.

*2) **The hardware**:* The devices used for the experiment were: an ACR1281U reader which can be used with both SC and NFC devices as contactless reader, attached to a PC running Windows 7 with 2 GB of RAM and a processor of 1.86 GHz. As an SC, we used a Java Card 2.2.2 with 16 bits processor, and a HCE equivalent application running on a Samsung Galaxy S4 with Android 5.0.1, 2 GB RAM, Quad-core (4x1.6 GHz Cortex-A15 and 4x1.2 GHz Cortex-A7).

*3) **The experiment protocol**:* Firstly, we needed to decide how to send the gesture information from the terminal to the card. A normal Application Protocol Data Unit (APDU), which is how we communicate with a card, can send up to 256 bytes. We tested two methods for sending the gestures information:

- Sending all the information frame by frame: in this way the APDU data size is 36 bytes for Gesture 1 and 2 and 66 bytes for Gesture 3 and 4
- Sending the maximum number of frames that an APDU can handle: for Gesture 1 and 2, it is six frames which gives an APDU data size of 216 bytes and for Gesture 3 and 4, it is three frames, so the APDU data size is 198 bytes

All the information about the gestures and the APDUs sent are summarized in Table I.

We measured the communication time for the APDUs described above for both SC and HCE, in order to know how this decision may affect the performance evaluation. We carried out 100 time measurements, to assess if the measured response time was stable.

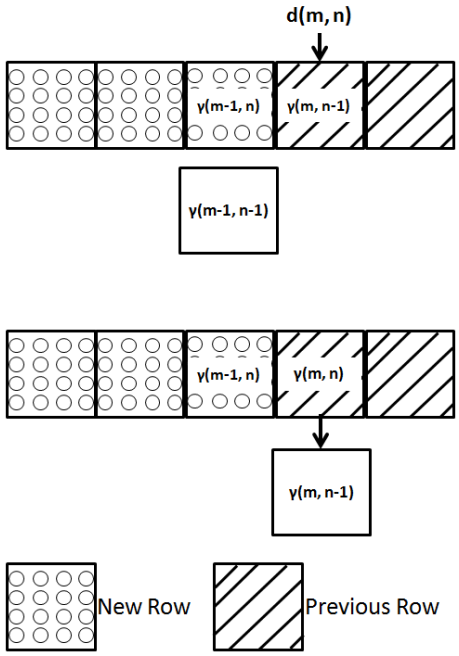


Figure 2. Application of DTW with only one row in memory

We then performed the Gesture Recognition application with DTW. First, we captured 100 time measurements for each of the four gestures, when running the application by sending the gesture frame by frame to the SC. We repeated the experiment packing the maximum number of frames into the APDU. We then repeated these two steps using HCE.

### *B. Dynamic Time warping: Memory optimisation*

The main drawback of the DTW algorithm is that, for a gesture A of M frames and a gesture B of N frames, it needs to fill an M x N matrix where the cell (i,j) represents the score between frame i of gesture A and frame j of gesture B plus the cumulative score.

Some works try to optimise the DTW either in calculation, memory or both. In [27], they reduced the amount of calculation and memory needed by focussing on a part of the DTW matrix, which may contain the warping path. But they force the warping path of any comparison to be in this calculated section which may imply more false positive results. The same comment can be made if we do not calculate the full matrix as the warping path will be altered.

Equation 3 shows we only need three elements, $\gamma(m-1;n-1), \gamma(m-1;n), \gamma(m;n-1)$. Thus we only need to have in memory two rows, either the row m and row m-1, or the row n and row n-1. Let us say that we have in memory the row m and row m-1: this method then reduces the memory cost from M x N to 2M, although the number of calculations remain unchanged.

We found that it is possible to implement the DTW algorithm by storing only one row of size M plus a temporary variable of the size of one element of M. This method

Table II
TIMES MEASURED IN MILLISECOND

| | Size of the APDU | Number of APDU sent | Communication time per APDU | | Average time for the full application | | Estimate processing time | |
|---|---|---|---|---|---|---|---|---|
| | | | SC | HCE | SC | HCE | SC | HCE |
| Gesture 1 | 36 | 90 | 6.71 | 23.65 | 92982.15 | 2228.48 | 92378.09 | 99.48 |
| | 216 | 15 | 24.72 | 77.54 | 76029.02 | 1196.35 | 75658.16 | 33.12 |
| Gesture 2 | 36 | 49 | 6.71 | 23.65 | 50622.60 | 1214.52 | 50293.73 | 55.40 |
| | 216 | 9 | 24.72 | 77.54 | 43760.30 | 667.18 | 43555.80 | 23.14 |
| Gesture 3 | 66 | 90 | 9.79 | 31.81 | 108674 | 3084.76 | 107792 | 221.60 |
| | 198 | 30 | 23.76 | 71.80 | 94083.19 | 2253.42 | 93370.15 | 99.30 |
| Gesture 4 | 66 | 49 | 9.79 | 31.81 | 32448.10 | 1684.36 | 31968.07 | 125.52 |
| | 198 | 17 | 23.76 | 71.80 | 28092.53 | 1237.19 | 27702.44 | 64.68 |

overwrote the row at each iteration and saved the temporary variable in the last overwritten cell.

So if we are looking for $\gamma(m;n)$ which will be saved in the cell c(a), we can find $\gamma(m;n-1)$ in the cell c(a-1); $\gamma(m-1;n)$ is the current value of c(a), and $\gamma\ (m-1;n-1)$ is saved in the temporary variable. Once the cost $\gamma(m;n)$ is calculated, we have to save the value in c(a) in the temporary variable then overwrite c(a) with the value of $\gamma(m;n)$. Figure 2 is illustrating this method.

### C. Results

For the communication, the SC is always more than three times quicker than the HCE (the average time for each kind of APDU can be seen in the Table II in the communication time per APDU column) and is more stable as its average standard deviation is 0.29ms against 5.65ms for HCE.

We then measured the time for the full application (communication time plus processing time) as described earlier in III-A3. The average time for each gesture can be seen in Table II under the column Average time for the full application. Knowing the communication time, the number of APDUs sent and the full time for the application, we can calculate the time needed by the devices to process the Gesture Recognition. This time can be seen in Table II under the column Estimate processing time.

We observed that the SC needed a lot of time to process a gesture; more than 27s for the quickest. Even if we used SC technology with a quicker communication interface, the SC processing would still be a bottleneck. On the other hand, the HCE had a slow communication time, but its processing time was much quicker than the SC, rarely exceeding 100ms duration.

## IV. DISCUSSION

The experiment has shown that it is possible to implement authentication based on Gesture Recognition on an SC at platform level, however the performance (one minute duration for a three second gesture) was far too slow to be practical. A solution to this problem could be to develop the application on a lower level, either in hardware or in native code. Based on the work of [28] who implemented signature recognition on an SC, on both application level and native level, using a algorithm of similar complexity to the one we used, we can estimate that the process time would be three times faster.

An HCE application would be more feasible for a real application as it takes around one second. However, an HCE application does not provide the attack resistance offered by an SC. The HCE application could be protected at least from phone malware, by running within a Trusted Execution Environment (TEE). A TEE offers a more restricted/protective environment for running sensitive code, compared to normal phone applications [29], although does not offer the tamper resistance of an SC.

Using a device with faster communication speed, or devices supporting extended APDUs (an extended APDU is able to support up to $2^{16}$ data bytes [30]) will reduce the time needed for the full Gesture Recognition application. The application installed on a personal, limited device, will still remain slower than an equivalent application installed on a secure server, but it will be more versatile, supporting both on-line and off-line transactions.

An example application that could use this kind of authentication would be for building access. If a sensor and a reader are installed at a restricted area entry point, possession of the SC (or the phone) plus knowledge of the correct gesture performed in the correct manner would be needed to enter. This three factor authentication then reduces the likelihood that the system could be hacked.

## V. CONCLUSION

In this paper we introduced gesture recognition as a means of authentication that included some biometric content, and that implementation required a secure means to store and process the reference template. We investigated whether an SC

or an HCE implementation would be feasible for a Gesture Recognition application, when using the DTW algorithm to compare gestures. Thus, we measured communication and processing time for both SC and HCE.

Although it is possible to run a Gesture Recognition application on an SC at platform level, it is not feasible for a real application as our implementation took around a minute. There is scope for improving performance by implementing the application at a lower level, either in hardware or with native code.

Our HCE application, was far more practical for a real world application, although it did not provide the attack resistance that an SC offers. The use of a TEE may enhance security and resist logical and malware attacks, although a TEE does not offer the tamper-resistance of an SC.

Future work includes the implementation of Gesture Recognition in hardware and/or native SC code, and frame optimisation research.The latter may determine the minimal number of gesture frames to authenticate genuine users whilst still rejecting attackers.

## REFERENCES


[1] N. Ratha, J. Connell, R. M. Bolle, and S. Chikkerur, "Cancelable biometrics: A case study in fingerprints," in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 4. IEEE, 2006, pp. 370–373.

[2] A. Chahar, S. Yadav, I. Nigam, R. Singh, and M. Vatsa, "A leap password based verification system," in *Biometrics Theory, Applications and Systems (BTAS), 2015 IEEE 7th International Conference on*. IEEE, 2015, pp. 1–6.

[3] B. Ducray, S. Cobourne, K. Mayes, and K. Markantonakis, "Authentication based on a changeable biometric using gesture recognition with the Kinect™," in *2015 International Conference on Biometrics (ICB)*. IEEE, 2015, pp. 38–45.

[4] J. Wu, J. Konrad, and P. Ishwar, "Dynamic Time Warping for gesture-based user identification and authentication with Kinect," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2371–2375.

[5] Microsoft, "Kinect for Windows Sensor Components and Specifications," http://msdn.microsoft.com/en-us/library/jj131033.aspx, 2014, [Online; accessed 16 November 2016].

[6] "Leap Motion," https://developer.leapmotion.com/getting-started/javascript/developer-guide, 2016, [Online; accessed 25 February 2016].

[7] E. Keogh and C. A. Ratanamahatana, "Exact indexing of Dynamic Time Warping," *Knowledge and information systems*, vol. 7, no. 3, pp. 358–386, 2005.

[8] G. D. Clark and J. Lindqvist, "Engineering gesture-based authentication systems," *IEEE Pervasive Computing*, vol. 14, no. 1, pp. 18–25, 2015.

[9] N. Prakash, "Host card emulation," *International Journal of Scientific and Research Publications*, vol. 5, no. 8, pp. 1–3, 2015.

[10] K. Eagles, K. Markantonakis, and K. Mayes, "A comparative analysis of common threats, vulnerabilities, attacks and countermeasures within smart card and wireless sensor network node technologies," in *Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems*. Springer, 2007, pp. 161–174.

[11] X. Leng, "Smart card applications and security," *information security technical report*, vol. 14, no. 2, pp. 36–45, 2009.

[12] K. Mayes, F. Piper, and K. Markantonakis, "Smart card based authentication: any future," *Computers & Security*, vol. 24, pp. 188–191, 2005.

[13] C.-T. Li and M.-S. Hwang, "An efficient biometrics-based remote user authentication scheme using smart cards," *Journal of Network and computer applications*, vol. 33, no. 1, pp. 1–5, 2010.

[14] X. Huang, Y. Xiang, A. Chonka, J. Zhou, and R. H. Deng, "A generic framework for three-factor authentication: preserving security and privacy in distributed systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 8, pp. 1390–1397, 2011.

[15] A. Umar, K. Mayes, and K. Markantonakis, "Performance variation in host-based card emulation compared to a hardware security element," in *Mobile and Secure Services (MOBISECSERV), 2015 First Conference on*. IEEE, 2015, pp. 1–6.

[16] "Leap Motion," https://developer.leapmotion.com/documentation/cpp/unreal/Leap_Unreal_Cpp_Tutorial.html, 2016, [Online; accessed 16 November 2016].

[17] Microsoft, "Human Interface Guidelines v1.8," http://go.microsoft.com/fwlink/?LinkID=247735, 2013, [Online; accessed 16 November 2016].

[18] F. Hong, M. Wei, S. You, Y. Feng, and Z. Guo, "Waving authentication: your smartphone authenticate you on motion gesture," in *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 2015, pp. 263–266.

[19] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan, "UWave: Accelerometer-based personalized gesture recognition and its applications," *Pervasive and Mobile Computing*, vol. 5, no. 6, pp. 657–675, 2009.

[20] R. M. Araujo, G. Grana, and V. Andersson, "Towards skeleton biometric identification using the Microsoft Kinect sensor," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. ACM, 2013, pp. 21–26.

[21] J. Tian, C. Qu, W. Xu, and S. Wang, "Kinwrite: Handwriting-based authentication using Kinect." in *NDSS*, 2013.

[22] M. T. I. Aumi and S. Kratz, "AirAuth: evaluating in-air hand gestures for authentication," in *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services*. ACM, 2014, pp. 309–318.

[23] V. Velichko and N. Zagoruyko, "Automatic recognition of 200 words," *International Journal of Man-Machine Studies*, vol. 2, no. 3, pp. 223–234, 1970.

[24] D. Gavrila and L. Davis, "Towards 3-d model-based tracking and recognition of human movement: a multi-view approach," in *International workshop on automatic face-and gesture-recognition*. Citeseer, 1995, pp. 272–277.

[25] G. Ten Holt, M. Reinders, and E. Hendriks, "Multi-dimensional dynamic time warping for gesture recognition," in *Thirteenth annual conference of the Advanced School for Computing and Imaging*, vol. 300, 2007.

[26] J. F. Lichtenauer, E. A. Hendriks, and M. Reinders, "Sign language recognition by combining statistical DTW and independent classification," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 11, pp. 2040–2046, 2008.

[27] J. Putz-Leszczyńska and M. Kudelski, "Hidden signature for DTW signature verification in authorizing payment transactions," *Journal of telecommunications and information technology*, pp. 59–67, 2010.

[28] O. Henniger and K. Franke, "Biometric user authentication on smart cards by means of handwritten signatures," in *Biometric Authentication*. Springer, 2004, pp. 547–554.

[29] J.-E. Ekberg, K. Kostiainen, and N. Asokan, "Trusted execution environments on mobile devices," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 1497–1498.

[30] "ISO 7816-4," http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816-4.aspx, 2013, [Online; accessed 11 November 2016].