Seeing is Not Believing: Visual Verifications through Liveness Analysis using Mobile Devices

Mahmudur Rahman
Florida International University
Miami, FL
mrahm004@cs.fiu.edu

Umut Topkara
IBM Research
Yorktown Heights, NY
umut@us.ibm.com

Bogdan Carbunar
Florida International University
Miami, FL
carbunar@cs.fiu.edu

ABSTRACT

The visual information captured with camera-equipped mobile devices has greatly appreciated in value and importance as a result of their ubiquitous and connected nature. Today, banking customers expect to be able to deposit checks using mobile devices, and broadcasting videos from camera phones uploaded by unknown users is admissible on news networks. We present Movee, a system that addresses the fundamental question of whether the visual stream coming into a mobile app from the camera of the device can be trusted to be untampered with, live data, before it can be used for a variety of purposes.

Movee is a novel approach to video liveness analysis for mobile devices. It is based on measuring the consistency between the data from the accelerometer sensor and the inferred motion from the captured video. Contrary to existing algorithms, Movee has the unique strength of not depending on the audio track. Our experiments on real user data have shown that Movee achieves 8% Equal Error Rate.

1. INTRODUCTION

In response to the ubiquitous and connected nature of mobile devices, industries such as utilities, insurance, banking, retail, and broadcast news have started to trust visual information gleaned from or created using mobile devices. Mobile apps utilize mobile device cameras for purposes varying from authentication to location verification, to tracking and witnessing. Today, one can deposit a check using a mobile phone, and videos from mobile phones uploaded by unknown users are shown on broadcast news to a national audience.

We address the fundamental question of whether the visual stream that a mobile app receives from the camera of the device can be trusted and has not been tampered with by a malicious user attempting to game the system. We refer to this problem as video "liveness" verification. The practical attacks we consider are i) feeding a previously recorded video through man in the middle software, ii) pointing the camera to a replay of a video, and iii) pointing and moving the camera over a static photo. This problem is a cornerstone in a variety of practical applications which use the mobile device camera as a trusted witness, including citizen journalism, smart cities, mobile authentication, and product condition verification for online sales (see Section 5).

In this paper, we propose to consult the motion sensors of mobile devices in order to verify the "liveness" of the video streams. We introduce Movee, a system that exploits the inherent movement of the user's hand when shooting a video. The verification relies on the consistency between the inferred motion from captured video and inertial sensor signals. Movee uses the intuition that being *simultaneously* captured, these signals will necessarily bear certain relations, that are difficult to fabricate and emulate. In this case, the movement of the scene in the video stream should have similarities with the movement of the device that registers at the motion sensors.

In essence, Movee provides CAPTCHA [42] like verifications, by including the user, through her mobile device, into the verification process. However, instead of using the cognitive strength of humans to interpret visual information, we rely on their innately flawed ability to hold a camera still. Movee can also be viewed as a visual public notary that stamps an untrusted video stream, with data simultaneously captured from a trusted sensor. This data can later be used to verify the liveness of the video.

Previously, [25, 19, 18] have proposed to use audio streams in captured videos as a means to protect against spoofing in biometric authentication, by using static and dynamic relations between voice and face information from speaking faces. Others proposed video-based anti-spoofing methods [15], [14], [35] for biometric authentication. Instead, we propose to use the previously unexplored combination of video and accelerometer data to verify the liveness of the video capture process.

Movee has four main modules. The Video Motion Analysis (VMA) module processes the video stream as it is captured by the camera. It uses video processing techniques to infer the motion of the camera, producing a time-dependent motion vector. VMA is inspired by the process used in image stabilization capable cameras. Meanwhile, the Inertial Sensor Motion Analysis (IMA) module converts the data signal from the inertial sensors into another time-dependent motion vector. When the video capture is completed, the motion vectors from the VMA and IMA modules are compared in the Similarity Computation (SC) module. SC relies on a flavor of the Dynamic Time Warping (DTW) algorithm from speech and pattern recognition to compute the "simi-

ACSAC '13 Dec. 9-13, 2013, New Orleans, Louisiana USA

larity" of the two motion vectors. The SC module also produces a set of features which summarize the nature of the similarity. The features are used by the Classification module, which runs trained classifiers to decide whether the two motion sequences corroborate each other. If they do, Movee concludes the video is genuine.

The contributions of this work are the following.

- Introduce the "liveness" analysis problem to videos captured from mobile devices. Propose a suite of attacks that enable the perpetrator to tamper with and claim ownership of plagiarized media.
- Devise Movee, a lightweight liveness analysis solution that verifies the similarity of movement as inferred from simultaneously captured video and inertial sensor streams.
- Collect datasets of genuine and fraudulent video/inertial sensor samples. Provide a full-fledged implementation of Movee, consisting of a mobile client and a server component.

We have implemented Movee using a combination of Android, for the mobile app, and C++/PHP for the processing server. We have collected 100 genuine video/inertial sensor samples from 10 different users. We have used these samples to create two test datasets, each containing video/inertial sensor samples fabricated according to attacks against Movee. Our cross-validation tests conducted on these test datasets show that the accuracy of Movee in differentiating fraudulent and genuine videos is 92% for one attack and 84% for the other. Moreover, our implementation shows that the liveness analysis of Movee is efficient. The server, running on a slightly outdated Dell laptop, takes an average of 1.3s to analyze a 6s video.

2. SYSTEM MODEL

The system consists of a service provider (e.g., Vine [10]), that offers an interface for subscribers to upload videos they shot on their mobile devices. We assume subscribers own mobile devices equipped with a camera and inertial sensors (i.e., accelerometers). Devices also have Internet connectivity, which, for the purpose of this work may be intermittent. Each user is required to install an application on her mobile device, which we henceforth denote as the "client".

The client is used to capture videos that are later posted to the provider hosted user account. The client simultaneously captures and uploads the video and the inertial sensor streams from the device. The provider verifies the authenticity of the video by checking the consistency of the two streams. The verification is performed using limited information: the two streams are from independent sources, but have been captured at the same time on the same device.

In the remainder of the paper we use accelerometer and inertial sensor interchangeably.

2.1 Attacker Model

We assume that the service provider is honest. Users however can be malicious. As shown in the following, the user can tamper with/copy video streams and inertial sensor data. We assume that the inertial sensor and the data extracted from the inertial sensor are genuine and have not been tampered with. The goal of attackers is to fraudulently claim ownership/creation of videos they upload. A malicious user can launch several types of attacks on the video stream:

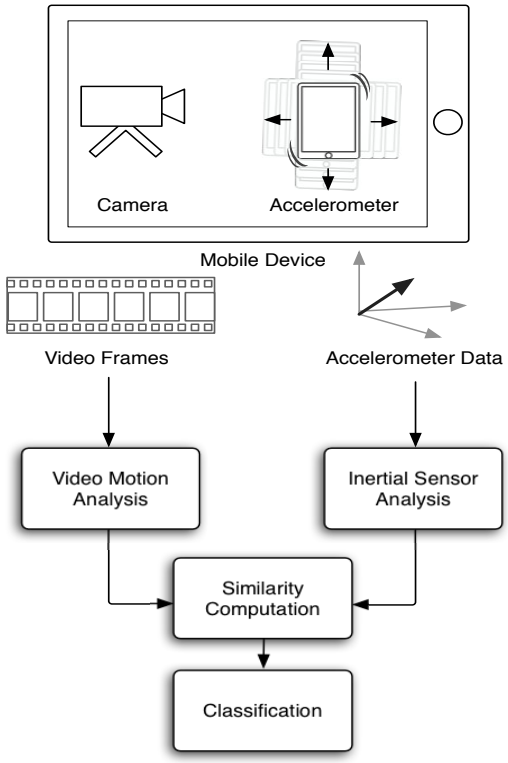


Figure 1: Movee uses four modules to verify a video stream: Both i) Video Motion Analysis (VMA), and ii) Inertial Sensor Motion Analysis (IMA), produce movement estimations during capture, iii) Similarity Computation extracts features, which iv) Classification uses to make the final decision.

Copy-Paste attack. The attacker copies a video taken by another user and uploads it as her own.
Replay attack. The attacker points the camera of the device to a replay of a video.
Projection attack. The attacker points and moves the camera of the device over a static photo or a projected image.
Random movement attack. Copy an existing video, then move the device in random directions, allowing the capture of inertial sensor data. Associate the video with the captured sensor stream and upload to the provider.
Direction sync attack. This is a sophisticated attack that improves on the random movement attack. Specifically, the attacker uses the device to emulate the movement observed in the video, e.g., if the image moves to the right, the user moves the device to the right.

3. MOVEE: SYSTEM OVERVIEW

We introduce Movee, a system that verifies the authenticity of a video taken with a mobile device. Movee performs a liveness analysis based on the consistency of the inferred motion from the simultaneously and independently captured streams from the camera and the inertial sensors. If the data from the inertial sensor corroborates the data from the camera, the system concludes that the video was genuine: it has been taken by the user pointing the camera to a real scene.

The Movee client is intended to be installed in mobile devices as part of special purpose video capture apps. When

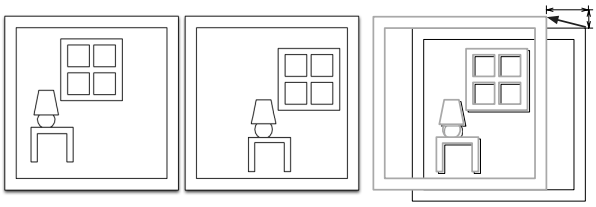


Figure 2: The Video Motion Analysis module processes each consecutive video frame and finds the motion vector by computing the amount of displacement that common image components have shifted between two frames.

the user wants to capture a visual, be it a video or a photo, two things happen simultaneously: i) the camera turns on and starts capturing video frames, ii) the inertial sensors are turned on and start collecting a stream of data concurrent with the camera.

Figure 1 shows a diagram of Movee. Movee infers the direction of motion and the magnitude of motion from two different types of sensor data. The data from the camera sensor is stored in periodically captured image frames. The data from the inertial sensor in most mobile devices comes in the form of periodically captured acceleration magnitudes on 3 main axes as measured by the accelerometer. The *Video Motion Analysis* (VMA) module uses an efficient image processing method to infer a motion vector over the timeline of the video from frame-by-frame progress. The *Inertial Sensor Motion Analysis* (IMA) module, converts the raw inertial sensor readings into a motion vector over the same timeline. Subsequently, the *Similarity Computation* (SC) module extracts features which represent agreements and differences between the two motion data (from the VMA and IMA modules) for the same time period. The final decision of whether the captured video is genuine is made in the *Classification* module. The Classification module uses trained classifiers on the data produced by the SC module to find out whether the inertial sensor data corroborates the video sensor data.

In the rest of this section, we describe the details of the four main modules of Movee.

3.1 Video Motion Analysis (VMA)

The Video Motion Analysis (VMA) module takes as input the captured video stream and outputs an estimate for the direction and magnitude of movement of the camera. The output of VMA is then used by the Similarity Computation module of Movee (see Figure 1).

It is possible to manually find the movement of the camera when given two consecutive photos taken with it: print the photos on transparency films and then *shift* one sheet on the other and keep *comparing* the two prints until they line up with minimal difference. The amount that the edges of one sheet overhang the other represents the offset between the photos (see Figure 2). The common optical mice [6] use this simple principle to determine pointer movement from a stream of images taken with a low resolution optical sensor mounted to their bottom side. The movement inferred from this analysis will be limited to only two axes, i) horizontal along the X axis, and ii) vertical along the Y axis.

VMA uses the *shift* and *compare* principle as well, by applying it on all consecutive frames of the video. The result is a frame-by-frame displacement vector. However, it would have been prohibitively expensive to compute the differences between two frames for all possible pixel shifts, especially considering how large each frame is. *Phase Correlation* [23] allows us to find the shift that minimizes the difference by carrying the computation into the frequency domain.

Phase correlation is an image processing technique that computes the spatial shift between two similar images (or sub-images). It is based on the Fourier shift property: a shift in the spatial domain of two images results in a linear phase difference in the frequency domain of the Fourier Transform (FT) [24]. It performs an element-wise multiplication of the transform images, then computes the inverse Fourier transform (IFT) of the result, and then finds the shift that corresponds to the maximum amplitude, to yield the resultant displacement. The maximum amplitude can be defined in the two-dimensional surface with delta functions (colloquially referred to as *peaks*) at the positions corresponding to spatial shifts between the two images. Phase correlation enables us to avoid the exhaustive search among all possible pixel shifts of one of the video frames over the next frame in order to find the one shift amount that results in minimal difference between the two frames. Instead, we find the location of the peak point in the Phase Correlation.

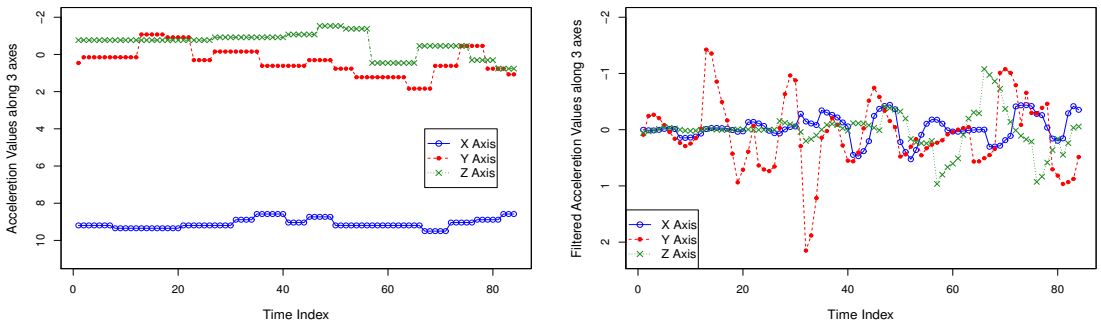
VMA first retrieves the frame per second (fps) rate of the stream and each available frame. In a pre-processing step, it applies a Hamming window [40] filter to eliminate noise from each frame. For each pair of consecutive frames, VMA applies the phase correlation method to obtain linear shifts between images in both X and Y directions. It then computes the *cumulative shift* along the X and Y axes by adding up the linear shifts for all consecutive frames retrieved from that video. Let $VS_{x,i}$ and $VS_{y,i}$ denote the cumulative video shifts of the i-th frame on the X and Y axes. We use (Section 3.4) $VS_{x,i}$ and $VS_{y,i}$ as feature descriptors.

3.2 Inertial Sensor Motion Analysis (IMA)

The Inertial Sensor Motion Analysis (IMA) module (see Figure 1) relies on the accelerometer sensor widely available in mobile devices. The IMA processes the data from the accelerometer in order to produce a motion direction and magnitude which is then compared in the Similarity Computation module with the output from the VMA module.

The inertial sensor coordinate system is defined relative to the screen of the phone in its default orientation. The X axis is horizontal and points to the right, the Y axis is vertical and points up and the Z axis points towards the outside of the front face of the screen (coordinates behind the screen have negative Z values). Let $\{(A_{x,i}, A_{y,i}, A_{z,i})|i = 1..m\}$ denote the accelerometer trace, recorded every T seconds, where $(A_{x,i}, A_{y,i}, A_{z,i})$ is the i-th sample, containing accelerometer readings on the three axes.

Filtering. In a pre-processing step, IMA uses a combination of low-pass and high-pass filters to remove the effects of gravity from the recorded raw acceleration values. Specifically, for the low-pass filter, let $G_{a,i}$ be the filtered gravity value on the a axis ($a \in \{X, Y, Z\}$) in the i-th sample and let $G_{a,i+1}$ be the gravity value to be filtered in the current, i+1-th sample. $A_{a,i+1}$ is the acceleration reading on the a axis for the i+1-th sample. Then, $G_{a,i+1} = \alpha G_{a,i} + (1-\alpha)A_{a,i+1}$, $\forall a \in \{x, y, z\}$. We have experimented with values of α ranging between 0.6 and 0.95. In our experiments we have used the value we found to perform best, $\alpha = 0.8$.



(a) (b)

Figure 3: (a) Raw Accelerometer Data. (b) Filtered Accelerometer Data. The Y axis is the dominant axis here for the direction and orientation of the mobile device.

Subsequently, IMA passes the result through a high-pass filter, $FA_{a,i+1} = A_{a,i+1} - G_{a,i+1}$, where $FA_{a,i+1}$ denotes the filtered acceleration value on the a axis, $\forall a \in \{x, y, z\}$, for the $i+1$-th sample. Figure 3(b) shows the effects of filtering for the sample raw acceleration of Figure 3(a)).

Inferring distance from acceleration data. Given acceleration data on each axis, $A_{a,1}, ..A_{a,m}$, $a \in \{X, Y, Z\}$, captured every T seconds, IMA computes the position (relative to the starting point) using a double integral. We adopt the trapezoidal rule [27], used for approximating the definite integral $\int_c^d f(x)dx$, representing the area below the curve where c,d are end points of integration. The integration step is first applied to obtain velocity ($vel_{a,i} = vel_{a,i-1} + \frac{A_{a,i}+A_{a,i-1}}{2} * T$). In a second application, the integration retrieves the position ($pos_{a,i} = pos_{a,i-1} + \frac{vel_{a,i}+vel_{a,i-1}}{2} * T$). $vel_{a,i}$ and $pos_{a,i}$, $i = 1..m$, denote the velocity and position at the i-th sample on the axis a. The resulting position shifts are combined to obtain the cumulative shift, $AS_{x,i}, AS_{y,i}, AS_{z,i}$, along each axis. $AS_{x,i}, AS_{y,i}, AS_{z,i}$ are then used as feature descriptors (see Section 3.4).

3.3 Similarity Computation (SC)

The Similarity Computation (SC) module compares the two motion sequences computed by the VMA and the IMA modules. It returns a set of features that summarize the nature of the similarity between the two sequences. The features are then used by the Classification module (see Section 3.4) to decide whether the two motion sequences corroborate each other, thereby concluding whether the video is genuine or not. The video motion and inertial sensor streams encode the same user hand movement, which are processed by the VMA and IMA modules respectively (see Figure 1) to each yield a motion stream.

To compute their similarity, we use a well-known sequence similarity measurement method from speech and pattern recognition, called Dynamic Time Warping (DTW). Similar to the well-known string edit distance, DTW is a dynamic programming solution to find the minimum cost set of operations that converts one sequence to the other.

In this subsection, we describe how we adapted the DTW algorithm to the practical issues in comparing the two motion sequences from the VMA and IMA modules. The two sequences differ in their number of samples, and have different magnitudes due to the nature of their source sensors. The VMA sequence length is proportional to the number of video frames, whereas the IMA sequence length is proportional to the product of the sample rate of the inertial sensor and the length of the recording interval. We perform a *stretching* step to make sure that the VMA and IMA sequences are of same length.

Furthermore, the motion sequence that the VMA infers from the video stream does not take into account the distance of objects into the camera. This may result in the same motion being registered as faster when the objects are close to the camera, and slower when the objects are far. We perform a *calibration* step to compute a coefficient to match the average speed of the motion the video stream to that of the inertial sensor stream.

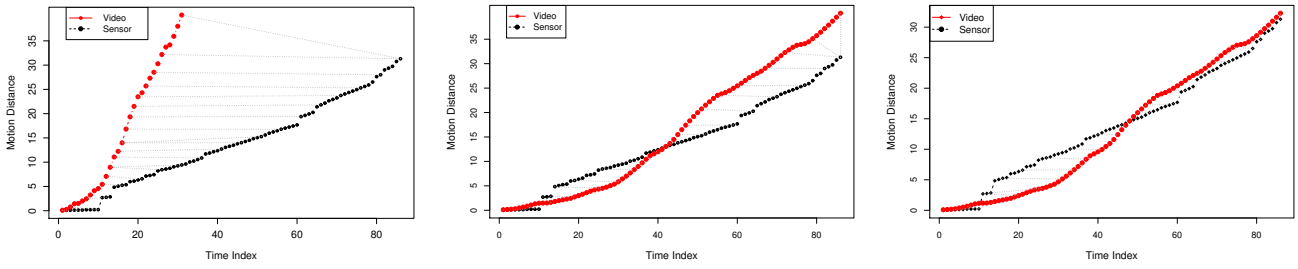
In the rest of this subsection, we first briefly detail the DTW algorithm, then present the stretching and calibration processes. We provide justification to the use of these methods with observed improvements in the resulting accuracy that the system gains after processing the features in the Classification module.

3.3.1 Dynamic Time Warping (DTW)

Given two time-dependent vectors, Dynamic Time Warping (DTW) [34] is dynamic programming algorithm for finding an optimal set of operations that minimize the cost of converting one vector to the other.

Let $\mathcal{F}$ be a feature space. Let $X = (x_1, x_2, .., x_n)$ and $Y = (y_1, y_2, .., y_m)$, $n, m \in \mathbb{N}$, be time-dependent vectors, $x_i, y_j \in \mathcal{F}$, $i = 1..n, j = 1..m$. An (n, m)-warping path of X and Y is a sequence $P(X, Y) = (p_1, .., p_L)$, where $p_l = (i, j) \in [1 : n] \times [1 : m]$, $\forall l \in [1 : L]$. A warping path satisfies (i) *boundary conditions*, $p_1 = (1, 1)$ and $p_L = (N, M)$, (ii) a *monotonicity condition*, $n_i \leq n_{i+1}$ and $m_j \leq m_{j+1}$, $i = 1..n-1$, $j = 1..m-1$, and (iii) a *step size condition*, $p_{l+1} - p_l \in (1, 0), (0, 1), (1, 1)$ for $l \in [1 : L-1]$.

DTW computes the (n, m)-warping path of X and Y as follows. Start with an empty path $P(X, Y)$. Assume it has already aligned X and Y up to the x_i and y_j in Y, $i < n, j < m$. To align x_{i+1} and y_{j+1}, DTW has the option to perform one of the following three *moves*, illustrated in Figure 4. First, a *diagonal move*, where it matches x_{i+1} to y_{j+1}. It then adds $(i+1, j+1)$ to $P(X, Y)$. In the next step



(a) (b) (c)

Figure 5: Example alignment of video and inertial motion streams extracted from the same experiment: (a) when using only DTW. (b) when stretching the shorter vector and applying DTW. (c) after stretching *and* calibration and applying DTW. Stretching helps achieve a significant alignment improvement.

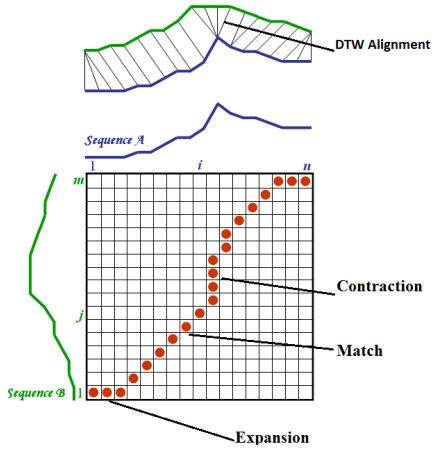


Figure 4: Illustration of DTW alignment for two time-dependent sequences. The red dots show the optimal warping path. A *diagonal move* is a match between the two sequences. An *expansion* duplicates one point of one sequence and a *contraction* eliminates one of the points.

it aligns x_{i+2} with y_{j+2}. Second, an *expansion move*, where it repeats x_i to match y_{j+1}. It adds $(i, j+1)$ to $P(X,Y)$ and continues the next for x_{i+1} and y_{j+2}, and (iii) a *contraction move*, where it drops x_{i+1}, and continues with the next step, to align x_{i+2} with y_{j+1}. Given a cost function for each move type, $c(i,j)$, $i = 1..n$, $j = 1..m$, the cost of a warping path $P(X,Y)$ is defined as $c_p(X,Y) = \sum_{l=1}^{L} c(p_l)$. The goal of DTW is to find a warping path p^*, of minimal cost among all possible warping paths.

Movee uses a variation of the DTW algorithm: the Variable Penalty Dynamic Time Warping (VPdtw) [20]. The process of expanding and contracting the time axis of a sensor stream can produce a very high quality alignment to a video stream. However, excessive numbers of expansions and/or contractions can often result in matches at random parts of the streams and appear artificial rather than catching the genuine common movement patterns. Penalized dynamic time warping uses a penalty to constrain the use of expansions and/or contractions. This penalty is incurred whenever a non-diagonal (i.e., expansion or contraction) move is taken (see Figure 4).

Let L denote the length of the longer sequence between the video and inertial sensor sequences for each sample. We extract several characteristics of the computed DTW alignment as feature descriptors, to be used by the Classification module (see Section 3.4). First, the *normalized penalty cost*, defined as the penalty cost divided by L. Second, the *ratio of overlap points*, which is the number of overlap points between the two streams, divided by L. Third, the *ratio of diagonal moves*, the number of diagonal moves divided by L. Fourth, the *ratio of expansion moves*, the number of expansion moves divided by L. Finally, the *ratio of contractions moves*, the number of contraction moves divided by L. The normalization to L ensures that the values are independent of the sample length.

3.3.2 Stretching

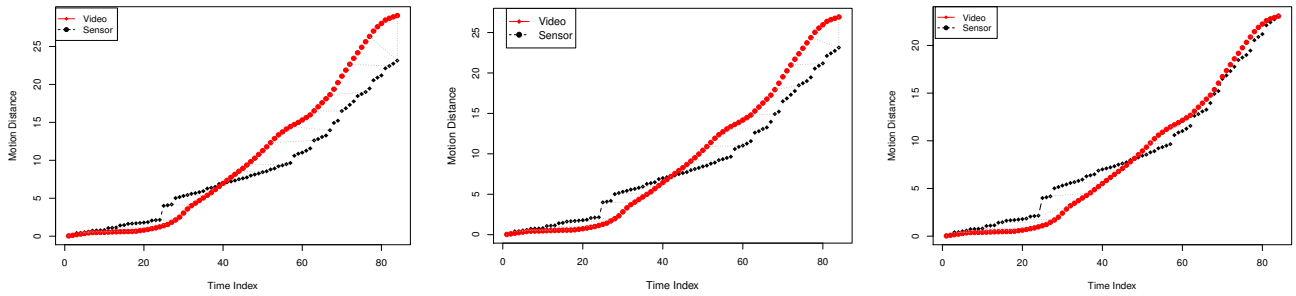
The sensor and video streams are sampled at different rates, thus the two vectors are of different length. The stretching step extends the shorter sequence (length s) to the length of the longer sequence (l). We use linear interpolation to compute $l - s$ new points for the shorter sequence. In Section 6.4 we show that depending on the attack type, the use of stretching improves the accuracy of Movee in differentiation fraudulent from genuine videos by a rate of 8-10%. This result is illustrated in Figure 5(b), where the use of stretching significantly improves the ability of the DTW procedure to align the video and inertial sensor movement streams when compared to Figure 5(a).

3.3.3 Calibration

An artifact of the method used in the Video Motion Analysis module is that the same motion pattern can be registered as faster when the objects in the view are close to the camera, and slower when the objects are far. In order to compensate for this artifact, we calibrate the speed of the video motion vector with a coefficient to match that of the speed of the inertial sensor motion vector.

The goal is to compute a calibration factor *CF*, that is used to multiply all the points in the video stream. We have explored several calibration methods, including mean based and linear curve fitting. We provide details however only on the two methods that performed the best in our experiments, *truncated mean* and *polynomial curve fitting*.

Truncated mean. The truncated mean computes the mean after discarding the high and low ends of the probability distribution (see Figure 6(b)). We apply this concept as follows:



(a) (b) (c)

Figure 6: The Video Motion Analysis module has a limited ability to determine the average speed of the device motion. For this reason, we use the average speed calculated from the inertial sensor motion vector to calibrate that of the video motion vector. This figure shows the effect of calibration in the similarity computation. (a) No calibration. (b) Truncated Mean Calibration. (c) Curve Fitting Calibration.

For each pair of points in the sensor and video vectors, compute their ratio and add it to a *ratio vector*. Compute the truncated mean of the ratio vector, discarding 12.5% from both the low and the high ends of the distribution.

Polynomial curve fitting. Polynomial curve fitting [21] constructs the polynomial that has the best fit to a series of data points (see Figure 6(c)). To compute the coefficients that best fit the curve to the given data, the least squares method [21] is used which minimizes the error between the data and the fitted polynomial [21]. Let SP_s denote the average value over the points on the fitted curve for the sensor stream and let SP_v denote the average value of the points on the curve of the video stream. Compute the calibration factor as $CF = \frac{SP_s}{SP_v}$.

Figures 6(b) and 6(c) show sample calibration outputs for these two methods, when compared to the uncalibrated version shown in Figure 6(a).

3.3.4 *Example Alignment*

To illustrate the need for the DTW, stretch and calibration steps previously described, we provide here experimental results of their use on a genuine sample of video and inertial sensor streams, captured using Movee (see Section 4 for implementation details). Figure 5(a) shows the alignment between the video and inertial sensor streams when only DTW is used. Figure 5(b) shows the resulting alignment when DTW and stretching are applied. Finally, Figure 5(c) shows the alignment achieved when DTW is applied along with stretching and calibration. The experiment shows that stretching is vital to achieve a good alignment, while calibration further improves the quality of the alignment.

3.4 Classification

The Similarity Computation module produces 14 features that represent the nature of the similarity between the motion information inferred from the video stream and the one observed from the inertial sensor data. The features are: (1) the movement direction of the target from the center of the screen (see Section 4), (2-5) the cumulative shift of the video and accelerometer on the x and (y) axes (4 descriptors), (6) the video motion direction, (7) the sensor motion direction, (8) the DTW distance after stretching and calibration steps, (9) the calibration factor, CF, (10) the normalized penalty cost, (11) the ratio of overlap points, (12) the ratio of diagonal moves, (13) the ratio of expansion moves and (14) the ratio of contractions moves.

The Classification module runs trained classifiers over these features to determine whether there is sufficient evidence that the video stream is genuine. Section 6.2 describes the classifiers used in our experiments.

Figure 7: Movee in action: Target icon (bullseye) at the bottom of the screen shows the direction in which the user needs to move the camera.

4. MOVEE IMPLEMENTATION

We have implemented a Movee client using Android and a server component using C++ and PHP. We used the OpenCV (Open Source Computer Vision) library [5] for the video motion analysis. The client allows users to capture movies and simultaneously provide proofs of liveness. Figure 7 shows a snapshot of Movee. When the user starts the client, she is presented with an initial screen that instruct her to hold the device firmly before pressing the start button. This is done to prevent initial accelerometer reading errors. Furthermore, once the user presses the start button, a target appears (bullseye). The user is instructed to move the camera in the direction of the target. Once the camera superimposes on the target, the target will change to a new place on the screen, and the user needs to continue to follow it. A progress bar indicates the elapsed time.

The target disappears after the first 6 seconds. This denotes the *verification* step. During the verification step, the Movee client captures the video stream and logs the accelerometer data. Following the verification period, the user can continue capturing the intended scenes. The Movee client only captures the data during the verification interval, which it sends to our server. We were inspired by Vine [10] to choose the verification interval to be 6s. Vine is an application that allows users to create and post (on Twitter, Facebook) video clips. This choice has the additional advantage that it keeps the size of the video file small (around 150 KB in the Samsung Admire Phone), thus reducing communication overheads. While the verification step can be performed on the client, we chose to impose a communication overhead for the improved liveness analysis performance of a more powerful server.

We have used a Samsung Admire smartphone running Android OS Gingerbread 2.3 with an 800MHz CPU to test the client side and a Dell laptop equipped with a 2.4GHz Intel Core i5 processor and 4GB of RAM for the server.

5. APPLICATIONS

Citizen journalism. The recent emergence of video sharing sites, e.g., [13, 10] has paved the way toward citizen journalism: people that witness events of public importance (e.g., public protests, natural and man-made disasters, meteorite landings) are now able to post their records of the events and share them with the community at large. The popularity of such sites, the rapid interest in certain videos and the intrinsic fame they bring to their creators raise important authenticity questions. People may upload fraudulent videos (e.g., created from old footage or from movies), in order to deceive, bias public opinion or simply cheat their way to fame. Movee can be used in conjunction with trusted location and time verification solutions (e.g., [17]) to verify claims made by video uploaders.

Smarter cities, Mobile 311. Mobile 311 apps by municipalities and metropolitan governments [4, 2] tap into crowdsourced reporting of potholes and open manholes for city maintenance, and to avoid possible hazards. To gauge the correctness and severity of a case, the systems require multiple users to report the same case before dispatching a crew. The shortcoming is that, the system is set to wait for multiple complaints to come in before action, and may even be accused of malpractice due to inaction even in the presence of information. Movee can act as the required witness to the genuine-ness of the reported case, and can eliminate the need to wait for multiple reports.

Prototype verification. Kickstarter [3] requires that real prototypes are used in the promotion videos of campaigns. Movee can be used by participants to verify the liveness of footage provided as evidence.

6. EVALUATION

We first describe our data collection process and the experimental setup. Second, we evaluate the overhead of the liveness analysis on the server and then study the ability of Movee to detect the attacks introduced in Section 2.1.

6.1 Data Collection

We have used the implemented Movee application to collect video and accelerometer samples. We used the 3D accelerometers, available in most recent smartphones and tablets, to acquire motion acceleration data. The Samsung Admire smartphones (Android OS Gingerbread 2.3 with 800MHz CPU), on which we collected the data, sample accelerometer readings at 16.67Hz [8] mode.

Motion direction inference. Given the video shift values computed by the VMA module, $VS_{x,i}$ and $VS_{y,i}$ the motion direction of the video is determined. For instance, if $VS_{x,i} < 0$ and $|VS_{x,i}| \gg |VS_{y,i}|$, the motion direction is to the right. A similar process is used to determine movement in the other directions. Furthermore, the IMA module maps the accelerometer values on its coordinate system. We exemplify the sensor motion direction decision using the following example (that we extended for all other directions and device orientation combinations). If the device is in landscape orientation and $AS_{y,i} < 0$, $AS_{y,i} \gg AS_{x,i}$ and $AS_{y,i} \gg AS_{z,i}$.

Data collection. We have collected data from 10 users [1]. Each user was asked to use Movee, following the instructions shown on the screen: move the device in one direction, for 6 seconds. We have collected 10 well defined (6s long) samples from each user; the total of 100 samples are stored in a "genuine" dataset. We have created two test datasets. The first dataset, that we call the "random" dataset, contains 50 genuine samples and 50 fraudulent samples created according to the Random attack. Each fraudulent "random" sample is created from one genuine sample, by coupling its video with the inertial sensor data of another, randomly chosen sample. The second dataset, called the "direction sync" set, contains the other 50 genuine samples and 50 fraudulent samples created according to the Direction Sync attack: Each fraudulent sample couples the video of one genuine sample with the inertial sensor data of another genuine sample, with the same direction of movement. This is effectively modeling the scenario of one user taking the video and another user (the attacker) emulating the movement in the video.

6.2 Experiment Setup

The Classification module (see Section 3.4) runs trained classifiers to determine whether there is sufficient evidence that a video stream is genuine. We have used three classifiers, Multilayer Perceptron (MLP) [26], Decision Tree (C4.5) and Random Forest (RF) [16].

We have applied 10-fold cross-validation tests [30] to assess how the results of the statistical analysis will generalize to an independent data set. We have used the Weka version 3.7.9 data mining suite [11] to perform the experiments, with default settings: For the backpropagation algorithm of the MLP classifier, we set the learning rate to 0.3 and the momentum rate to 0.2.

6.3 Metrics

We briefly define the metrics we use to evaluate the accuracy of Movee. We borrow several metrics from biometrics. The Receiver Operating Characteristic (ROC) curve [43] is a visual characterization of the trade-off between the False Accept Rate (FAR) and the False Reject Rate (FRR). The Equal Error Rate (EER) [41] is the rate at which both accept and reject errors are equal. A lower EER denotes a more accurate solution. The area under the ROC curve (AUC)

[1]Of the 10 users, 7 are males and 3 females, aged 23-32, occupation ranging from biology to fashion design, housewife and software, civil and electrical engineering

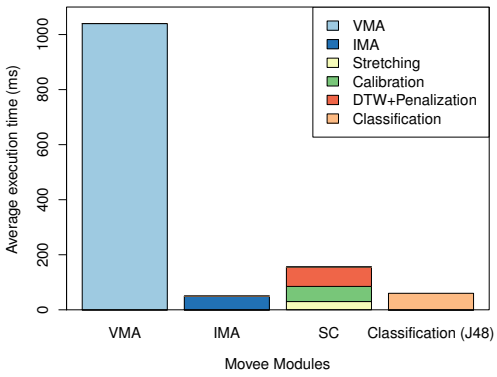


Figure 8: Movee (per-module) server side overhead: video processing is the most expensive. The total cost is however under 1.3s.

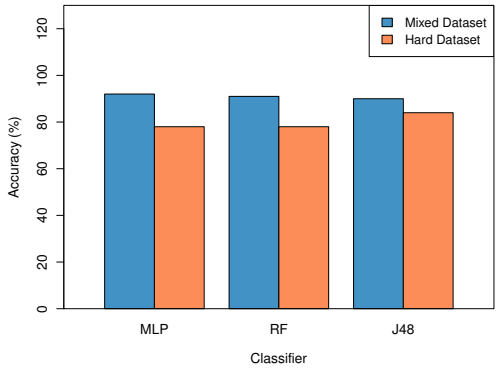


Figure 9: Movee accuracy, on random and direction sync data sets, when different classifiers are used. MLP outperforms the random forest and J48 classifiers on the random dataset (92%) while J48 outperforms the random forest and MLP on the direction sync dataset (84%).

is equal to the probability that a classifier will rank a randomly chosen genuine sample higher than a randomly chosen fraudulent one. An area of 1 represents a perfect test; an area of 0.5 represents a worthless test.

6.4 Experimental Results

6.4.1 Movee server overhead

Figure 8 shows the overhead (divided into modules) of the liveness analysis on the server, running on the above described Dell laptop, for 6s videos. The values are an average over 10 experiment runs. It shows that the VMA is the most time consuming module, slightly exceeding 1s. The IMA and Classification components (running the J48 classifier) impose the smallest overheads, together being 110ms. MLP takes an average of 940 ms and Random Forest an average of 140 ms. The overhead of the SC module is around 150ms, with the smallest cost imposed by the stretching step and the highest cost by the penalty based DTW.

6.4.2 Attack detection analysis

It is straightforward to see that Movee prevents the "Copy-Paste" and "Replay" attacks of Section 2.1: no sensor stream exists. Movee does not detect the "Projection", as the video and sensor streams are indeed captured in the same user hand movement. However, a human observer can immediately detect that the movie is of a poster. In the following, we study the ability of Movee to detect the last, but more complex "Random" and "Direction Sync" attacks. For this, we explore the accuracy of Movee in detecting fraudulent samples, on both random and direction sync data sets, using the above mentioned classifiers. The results are shown in Figure 9. For the random data set, the multilayer perceptron neural networks (MLP) provides the highest accuracy, 92% whereas the Random Forest (RF) and the C4.5/J48 Decision Tree exhibit accuracy of 91% and 90% consequently.

Figure 10(a) shows the ROC curve and the computed EER value for MLP and the random dataset. The EER value of MLP is as small as 0.08.

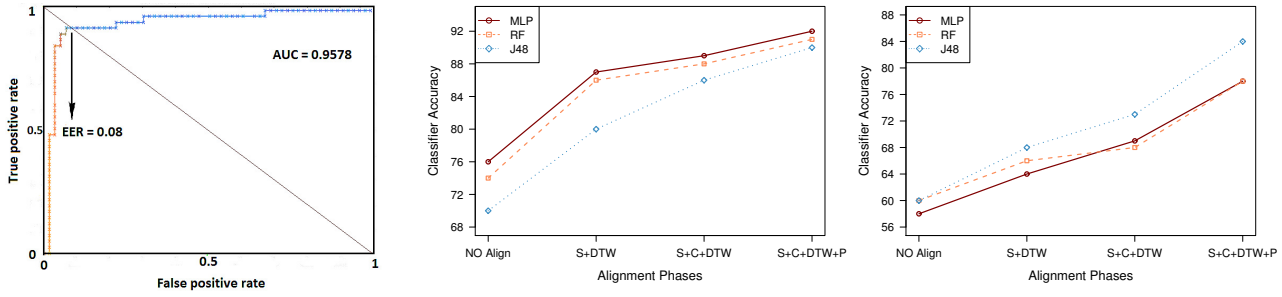
Finally, we evaluated the impact of each step of the SC module on the accuracy of Movee, for both test datasets. For each dataset, we measured the accuracy of the three classifiers when (i) no alignment phase was applied, (ii) when stretching and DTW were applied, (iii) for stretching, calibration and DTW, and (iv) for stretching, calibration and penalty based DTW. Figure 10(b) shows the results for the random dataset, and Figure 10(c) shows the results for the direction sync dataset. The stretching and DTW steps contribute the most to the accuracy of Movee: almost 12% for all three classifiers on the random attack and almost 8% for the direction sync attack. For the direction sync attack, the Penalization step brings the most accuracy improvement, almost 11% for all three classifiers. Note that MLP exhibits the best performance for the random attack; C4.5 achieves the best performance for the direction sync attack. When all processing steps are applied, C4.5 should be chosen: it outperforms MLP and RF's accuracy by 6% for the direction sync attack, while it lags only 2% behind MLP's accuracy for the random attack.

6.5 Limitations

We have not experimented with very short videos (less than 6s) or with videos shot in unusual circumstances: involving very high accelerometer activity, e.g., running, or when the user is in a moving vehicle. Due to the lack of gyroscope sensors in the Samsung Admire device, we have not integrated gyroscope readings to verify camera rotation movements.

Furthermore, we have not experimented with doctored video and accelerometer streams. For instance, given an input video, the attacker can use the work of Davison et al. [22] to recover the 3D trajectory of the camera. Then, given root access (e.g., using [9, 7], create a corresponding accelerometer sample and feed it to Movee, e.g., using a solution similar to [12]. We defer the task of providing trust for the integrity of the mobile app, as well as the trust for the integrity of the device's connection to its camera and accelerometer sensors to the providers of Movee. Establishing the integrity of a mobile platform and mobile apps is currently an active area of research [38, 1, 39].

Finally, we have not experimented with "green screen" attacks, where the attacker captures a video with a portion



(a) (b) (c)

Figure 10: (a) ROC curve on random dataset for Movee when the MLP classifier is used. The EER of MLP is as low as 0.08. (b) The impact of the SC steps on the Movee accuracy, for the three classifiers, for the random attack, and (c) for the direction sync attack. Stretching+DTW provide the highest improvement (10%) for the random attack. The penalization brings an 11% improvement for the direction sync attack.

of the scene being a green screen. Following the video capture, the attacker overlays additional video footage or static images on the green section. We note however that Movee raises the bar here: an attacker needs to invest in additional equipment to thwart the defenses of Movee. The quality of the equipment determines the (in)ability of a human observer to detect the attack.

7. RELATED WORK

The combination of video and accelerometer data has been studied by Hong et. al. [28] in order to improve the compute-intense motion estimation in video encoding. They have shown that the use of accelerometer data improves the speed of the encoding process by a factor of 2-3. Moiz et. al. [33] introduced and developed a wearable, multi-modality, motion capture platform, and used its inertial and ultrasonic sensors to estimate position. The focus of our work is different, on verifying liveness of a video through the consistency of its video and accelerometer data.

Indyk et al. [29] studied the problem of finding pirated video on the Internet. They propose to extract a small number of pertinent features (temporal fingerprints) based on the shot boundaries of a video sequence, and match them against a database of videos. We note our work is on an orthogonal problem, of verifying the liveness of a video claimed to have been taken by a mobile device user. As such, these two problems can complement each other.

A flavor of liveness analysis similar to the one we proposed, is used to verify biometric liveness. Kollreider et al. [31] study the problem of verifying the actual presence of a live face in contrast to a photograph (playback attack) for face recognition based biometrics. They introduce a lightweight optical flow approach that estimates face motion estimation on the structure tensor and a few input frames. Park et. al. [36] introduce a liveness detection method for distinguishing a two-dimensional object from a three-dimensional object. The approach proposed uses video sequence images and does not require additional hardware or user interaction. Their work has direct application to face recognition biometrics: it can identify the use of a flat picture. Further work is needed to understand the vulnerability of this approach to photo movement and photo bending/3D printing attacks.

Multi-modal approaches relying on different sensor sets [25, 19, 18] have been proposed, to exploit the static and dynamic relationship between voice and face information from speaking faces for biometric authentication. Chetty [18] proposed liveness checking techniques for multimodal biometric authentication systems. Their techniques fuse acoustic and visual speech features and measure the degree of synchronization between the lips and the voice extracted from speaking face video sequences.

Accelerometers have been used to provide biometric information, in the form of gait or gesture recognition. Mantyjarvi et al. [32] proposed several solutions that achieve low EER (equal error rates) for identifying users of mobile devices from gait signal acquired with three-dimensional accelerometers, when the device was worn on the belt, at the back. Pylvänäinen [37] used 3D accelerometers and hidden Markov models to identify gestures performed using a mobile device.

Summary. Our work introduces novel techniques for combining video and inertial sensor data to verify the liveness of a video stream. Movee verifies that the video has indeed been shot as claimed by the user, using her mobile device. We note that Movee does not require additional equipment, but requires the user to install and shoot the video using Movee's client application.

8. CONCLUSIONS

In this paper we have introduced the concept of "liveness" analysis, of verifying that a video has been shot on a claimed mobile device. We have proposed Movee, a system that relies on the accelerometer sensors ubiquitously deployed on most recent mobile devices to verify the ownership of a simultaneously captured video stream. We have implemented Movee, and, through extensive experiments, we have shown that (i) it is efficient in differentiating fraudulent and genuine videos and (ii) imposes reasonable overheads on the server. In future work we intend to integrate more sensors (e.g., gyroscope), as well as the use of MonoSLAM [22] as an alternative VMA implementation to improve accuracy.

9. ACKNOWLEDGMENTS

We thank the shepherd and the anonymous reviewers for their excellent feedback.

10. REFERENCES

[1] Arxan: Protecting the App Economy. http://www.arxan.com/.

[2] Chicago Works. http://www.chicagoworksapp.com/.

[3] Kickstarter. http://www.kickstarter.com/.

[4] NYC 311: Pothole or Other Street Surface Complaint. http://www.nyc.gov/apps/311/allServices.htm?requestType=topService&serviceName=Pothole+or+Other+Street+Surface+Complaint.

[5] Open Source Computer Vision. http://opencv.org/.

[6] Optical mouse. https://en.wikipedia.org/wiki/Optical_mouse.

[7] Root and Me. https://play.google.com/store/apps/details?id=com.iamjake.root&hl=en.

[8] Sensor Delay. http://developer.android.com/reference/android/hardware/SensorManager.html.

[9] Unlock Root. http://www.unlockroot.com/.

[10] Vine. http://vine.co/.

[11] Weka. http://www.cs.waikato.ac.nz/ml/weka/.

[12] XPrivacy 1.9.5: The ultimate privacy manager. http://forum.xda-developers.com/showthread.php?t=2320783.

[13] YouTube. http://www.youtube.com.

[14] A. Ali, F. Deravi, and S. Hoque. Liveness detection using gaze collinearity. In *Emerging Security Technologies (EST)*, pages 62–65, 2012.

[15] A. Anjos and S. Marcel. Counter-measures to photo attacks in face recognition: A public database and a baseline. In *Biometrics (IJCB)*, pages 1–7, 2011.

[16] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[17] S. Capkun, K. B. Rasmussen, M. Cagalj, and M. B. Srivastava. Secure location verification with hidden and mobile base stations. *IEEE Trans. Mob. Comput.*, 7(4):470–483, 2008.

[18] G. Chetty. Biometric liveness detection based on cross modal fusion. In *Information Fusion, 2009. FUSION '09. 12th International Conference on*, pages 2255–2262, July.

[19] G. Chetty and M. Wagner. Multi-level liveness verification for face-voice biometric authentication. In *Biometric Symposium*, 2006.

[20] D. Clifford and G. Stone. Variable penalty dynamic time warping code for aligning mass spectrometry chromatograms in r. *Journal of Statistical Software*, 47(8):1–17, 4 2012.

[21] I. Coope. Circle fitting by linear and nonlinear least squares. *Journal of Optimization Theory and Applications*, 76:381–388, 1993.

[22] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6):1052–1067, June 2007.

[23] E. De Castro and C. Morandi. Registration of translated and rotated images using finite fourier transforms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(5):700–703, May 1987.

[24] J. B. J. Fourier and A. Freeman. *The Analytical Theory of Heat*. Cambridge University Press, 2009.

[25] R. Frischholz and U. Dieckmann. Bioid: A multimodal biometric identification system. *IEEE Computer*, 33(2):64–68, 2000.

[26] S. I. Gallant. Perceptron-based learning algorithms. *Trans. Neur. Netw.*, 1(2):179–191, June 1990.

[27] B. F. Hildebrand. *Introduction to numerical analysis: 2nd edition*. Dover Publications, Inc., 1987.

[28] G. Hong, A. Rahmati, Y. Wang, and L. Zhong. Sensecoding: accelerometer-assisted motion estimation for efficient video encoding. MM '08, pages 749–752. ACM, 2008.

[29] P. Indyk, G. Iyengar, , and N. Shivakumar. Finding pirated video sequences on the internet. Technical report, Stanford University, 1999.

[30] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. pages 1137–1143, 1995.

[31] K. Kollreider, H. Fronthaler, and J. Bigün. Non-intrusive liveness detection by face images. *Image Vision Comput.*, 27(3):233–244, 2009.

[32] J. Mantyjarvi, M. Lindholm, E. Vildjiounaite, S.-M. Makela, and H. Ailisto. Identifying users of portable devices from gait pattern with accelerometers. In *ICASSP '05*, volume 2, pages ii/973–ii/976 Vol. 2, March.

[33] F. Moiz, D. Leon-Salas, and Y. Lee. A wearable motion tracker. BodyNets '10, pages 214–219.

[34] M. MÃijller. Dynamic time warping. In *Information Retrieval for Music and Motion*, pages 69–84. Springer Berlin Heidelberg, 2007.

[35] G. Pan, L. Sun, Z. Wu, and S. Lao. Eyeblink-based anti-spoofing in face recognition from a generic webcamera. In *ICCV 2007.*, pages 1–8.

[36] G.-t. Park, H. Wang, and Y.-s. Moon. Liveness detection method and apparatus of video image, August 2007.

[37] T. Pylvänäinen. Accelerometer based gesture recognition using continuous hmms. IbPRIA'05, pages 639–646. Springer-Verlag, 2005.

[38] A. Shabtai, Y. Fledel, and Y. Elovici. Securing android-powered mobile devices using selinux. *Security Privacy, IEEE*, 8(3):36–44, 2010.

[39] J. Six. *Application Security for the Android Platform: Processes, Permissions, and Other Safeguards*. O'Reilly, 2011.

[40] J. O. Smith. *Spectral Audio Signal Processing*. W3K Publishing, 2011. online book.

[41] N. P. H. Thian and S. Bengio. Evidences of equal error rate reduction in biometric authentication fusion, 2004.

[42] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. Captcha: Using hard ai problems for security. In *In Proceedings of EUROCRYPT*, pages 294–311. Springer-Verlag, 2003.

[43] Wikipedia. http://en.wikipedia.org/wiki/Receiver_operating_characteristic.