Secure Pairing of Mobile Devices

Megan Hacker	Mark Crovella	Leonid Reyzin
Boston University	Boston University	Boston University
mhacker@bu.edu	crovella@bu.edu	reyzin@cs.bu.edu

May 16, 2012

Abstract

As mobile devices become increasingly popular, the necessity for both user-friendly and secure pairing methods for these devices also rises. One natural approach to pairing devices is to match them based on a shared experience. In this work, we define a shared experience as the act of physically holding two devices together and shaking them for a short period. The common movement data collected during the shaking process can subsequently be used to verify the authenticity of a secret key established via a key exchange protocol. This paper explores the process of key verification using two different measures: a coherence measure derived through time series analysis and a measure based on Hamming distance. Using ROC curves, we show that both of these measures robustly distinguish between the case where two devices have been shaken together and the case where two devices have been shaken separately.

1 Introduction

The ability to pair two or more mobile devices is desirable for a number of reasons. Consider two users meeting for the first time who wish to share their contact information with the other; performing this task using their phones can make it easy, quick, and free from human error (such as accidentally saving the wrong information). Of course, these users can connect their devices through a wireless network to perform such an exchange, but this approach makes the users' information vulnerable to adversarial attacks called man-in-the-middle attacks. Man-in-the-middle attacks can be passive attacks where the adversary simply listens in on the conversation between the two users, or they can be active where the adversary takes control of the entire interaction, posing as user A from user B's perspective and user B from user A's perspective, while the users remain unaware that they are not in fact communicating with the intended party. Although passive attacks are easily countered using encryption, authentication is needed to protect against active attacks.

A natural way to authenticate a mobile device is to require the user to provide secret information as proof of their identity, and for obvious reasons, it is preferable to do so with minimal user overhead. Using a password requires users both to remember information and to enter it correctly when prompted. This overhead can be reduced by allowing the user to create a one-time password based on data collected by the device; this way, users do not have to remember or enter any information themselves. It is important that this data is not collected in a predictable way. Given a predictable data collection method, it would be very easy for an adversary to fabricate a set of desired data. A logical solution to this problem is to collect the data randomly. Shaking a device and collecting the movement data using the device's internal accelerometer is a simple and intuitive way of creating an unpredictable, one-time use secret. Two mobile devices that are physically held together and shaken are expected to come up with very similar measurements. These can then be compared, with the intent of pairing the corresponding devices only if the measurements are determined to be similar.

In order to evaluate the similarity between two sets of measurements, we explore two different similarity measures. The first measure uses time series analysis techniques on the data to produce a final value between 0 and 1. This value is large for two sets that are similar and small otherwise. The second measure compresses the data from three dimensions into one and projects

the resulting vectors onto a predetermined number of randomly generated uniform vectors. Two final vectors are constructed based on the directions of the projections, and the Hamming distance between these two final vectors is used as the similarity measure between the two original measurement sets.

The remainder of the paper is structured as follows: Section 2 discusses related work as well as some key differences between that work and this work. Section 3 introduces our new method that uses Hamming distance. Section 4 describes the iOS application (which will be referred to as the 'app' from here on) used to collect movement data. Sections 5 and 6 go in depth into the implementation of the app, how the two different measures analyze the collected data, the results from experiments performed by various users, and our interpretation of these results. Finally, the paper concludes with future work in Section 7 and the conclusion in Section 8.

2 Related Work

The main inspiration for this project came from work done by Mayrhofer and Gellersen who created a key verification method and a key construction method, both based on simultaneous shaking of mobile devices, called ShaVe and ShaCK respectively [8].

ShaVe works in the following way. First, a secret key is generated using the Diffie-Hellman key agreement over a wireless channel [3]. Next, two devices are shaken together, and the collected acceleration data is exchanged between the devices using an Interlock protocol [9]. This data is then analyzed using a coherence measure which determines whether or not the previously generated secret key will be adopted by the two devices as the secret key for a secure communication session.

The method to compute the coherence measure used in the implementation of ShaVe was adopted from work done by Lester et al. [7]. Given two $m \times 3$ matrices A and B as input, where A and B are two different time series, the method returns a coherence value, C_{xy}, that is a measure of the similarity between A and B in the frequency domain. A and B represent the data collected during the shaking session undergone by device A and device B respectively, and m is the number of samples collected by each device. It is likely that the two devices will be imperfectly aligned during a given shaking session; to account for this, Eqn. 1 is used on both A and B to obtain

rotation invariant vector representations of each matrix.

$$\vec{A} = \sqrt{(A_X)^2 + (A_Y)^2 + (A_Z)^2} \tag{1}$$

$\vec{A}$ and $\vec{B}$ are split into n averaged slices where slice k is denoted by $\vec{A}_k$. Mayrhofer and Gellersen use overlapping slices in their implementation, but we did not. For a particular frequency f, Discrete Fourier Transform coefficients $x_k(f)$ and $y_k(f)$ (Eqn. 2) are computed to be used in the cross-power spectra equation $P_{xy}(f)$ (Eqn. 3), which is in turn used in the magnitude squared coherence equation $C_{xy}(f)$ (Eqn. 4) [1].

$$x_k(f) = FFT(\vec{A}_k(t) \cdot h(t)), \qquad y_k(f) = FFT(\vec{B}_k(t) \cdot h(t)) \tag{2}$$

$$P_{xy}(f) = \frac{1}{n} \sum_{k=0}^{n-1} x_k(f) \cdot \bar{y}_k(f) \tag{3}$$

$$C_{xy}(f) = \frac{P_{xy}(f)^2}{P_{xx}(f) \cdot P_{yy}(f)} \tag{4}$$

Note that $h(t) = \frac{1-\cos(2\pi t/w)}{2}$ is the standard von-Hann window, and $\bar{y}_k(f)$ denotes the complex conjugate of $y_k(f)$. If f is varied from 0 to a maximum value called f_{max}, the various values of $C_{xy}(f)$ computed for the different f's can be averaged to come up with the final coherence value C_{xy}. Mayrhofer and Gellersen do this with the following equation:

$$C_{xy} = \frac{1}{f_{max}} \int_0^{f_{max}} C_{xy}(f)\, df \tag{5}$$

But since there is a finite number of values in A and B (that is, the data is discrete), it makes more sense for our purposes to replace the integral with a summation:

$$C_{xy} = \frac{1}{f_{max}} \sum_0^{f_{max}} C_{xy}(f) \tag{6}$$

If the C_{xy} computed using the above method on a given A and B is above a threshold θ, we conclude that device A and device B were shaken together. Otherwise, we conclude that device A and device B were not shaken together.

[1] This is a corrected version of the equation published in [8].

ShaCK, on the other hand, starts with the shaking of two devices held together and works in real time to extract feature vectors from the collected acceleration data. These feature vectors are exchanged via an interactive cryptographic protocol, and as soon as a sufficient number of matching vectors are exchanged, a shared secret key can be constructed using the matching vectors. Since the secret key is assembled over the duration of the shaking session, it is clear that ShaCK allows users to undertake shorter shaking sessions at the expense of constructing a weaker key than would be constructed if the users were to shake for a longer duration. Under the assumptions that (1) the average user does not want to shake devices for any longer than absolutely necessary, and (2) the average user does not understand the security implications of a weaker key (and thus is willing to trade security for faster key construction), it can be further assumed that the average user is likely to take full advantage of the flexibility of this protocol. This is potentially a large problem because even the strongest security guarantees for a given protocol will not protect a user that does not use the protocol as it is intended to be used. ShaCK is assumed to generate 7 bits of entropy per second regardless of the duration of the shaking session, but this is a very questionable assumption. Consider a shaking session that lasts s seconds. There is no way to guarantee the shaking that occurs between second $i-1$ and second i is independent from the shaking that occurs between second i and second $i+1$ for any i up to $s-1$. Therefore, it would be more reasonable to assume that ShaCK generates 7 bits of entropy during the first second of shaking and e bits of entropy during each subsequent second of shaking where e is inversely proportional to s. Future work on this project might determine an actual value for e and/or a relationship between the shaking that occurs during neighboring time segments in the shaking session as a whole; either of these discoveries would certainly help to provide better entropy guarantees.

Mayrhofer and Gellersen conclude their work by discussing a user study they performed that shows their authentication methods robustly differentiate between devices shaken together and shaken independently. Additionally, this user study demonstrated that their methods are easy to learn, feasible to implement on mobile devices, and secure against adversaries with full control over a wireless channel.

3 New Method: Random Projections and Hamming Distance

The high level idea behind this new approach is the following. Consider two devices that have been shaken together. The resulting $x_k(f)$ and $y_k(f)$ vectors from running the coherence method (i.e. the data sets after the FFT function has been applied, but before either the cross-power spectra or magnitude squared coherence functions have been applied) should be very close together, and thus, the probability of generating a random uniform vector that falls somewhere between them in space is very low. Another way to see this is by considering the projections of $x_k(f)$ and $y_k(f)$ onto a random uniform vector. If this random uniform vector is between $x_k(f)$ and $y_k(f)$, then the two projections will have opposite signs, but if this random uniform vector is either above or below both $x_k(f)$ and $y_k(f)$, then the two projections will have the same sign. Given $x_k(f)$ and $y_k(f)$ that are close, the probability that the two projections will have the same sign is very high. If two devices have not been shaken together, $x_k(f)$ and $y_k(f)$ will not be close, so it is much more likely that a random uniform vector will fall between the corresponding $x_k(f)$ and $y_k(f)$ vectors, and the probability that the two projections will have the same sign is low.

For obvious reasons, using only one random uniform vector will not yield strong results, so we generate many vectors and project both $x_k(f)$ and $y_k(f)$ onto each one. The goal is to use the fraction of projection pairs with the same sign over the total number of projection pairs to determine whether two devices have been shaken together or separately. By representing the sign of a given projection with a binary value, we create two binary vectors, one storing the signs of the projections of $x_k(f)$ and one storing the signs of the projections of $y_k(f)$. We then compute the Hamming distance between these two vectors in order to compare the signs of the projections. The fraction of same sign projection pairs over the total number of projection pairs will be large for devices that have been shaken together, and the fraction of same sign projection pairs over the total number of projection pairs will be small for devices that have been shaken separately. This technique is of interest because there are protocols that can perform key agreement based on two parties each holding one of two strings that are close in Hamming distance [4, 5, 1, 2].

We can also think about Hamming distance geometrically. Let the angle

between the $x_k(f)$ (or $y_k(f)$ without loss of generality) vector be ϕ, and let the arbitrary random uniform vector we wish to project onto be $\vec{d}$. Using the dot product representation

$$x_k(f) \cdot \vec{d} = \|x_k(f)\| \|\vec{d}\| \cos\phi$$

one can see that the sign of $\cos\phi$ is the sign of the projection of $x_k(f)$ onto $\vec{d}$. Furthermore, the relative Hamming distance between the two binary vectors discussed above can be represented by $\frac{\phi}{\pi}$. This is true based on the following. Consider two vectors in $\mathbf{R}^2$ with angle ψ between them; for simplicity, let one vector be the x unit vector, and let the other vector be some vector in the positive portion of the y-plane. A random line somewhere in the positive portion of the y-plane and that travels through the origin has π equally likely positions it could take, so the probability that a random line falls between these two vectors is $\frac{\psi}{\pi}$. This argument may not seem relevant to our purposes since we are clearly not working in a two dimensional space, but we can extend this argument from two dimensions to higher dimensions by applying a rotation to any higher dimensional space that brings one vector to the x axis and the other vector to the xy-plane. The resulting zeros in all dimensions except the first and second will have no impact on the first two dimensions.

4 The Application

The app used to collect movement data is currently only compatible with the iOS platform and can only successfully collect data using devices with a built-in accelerometer. Thus, the app is limited to running on the iPhone, iPod Touch, and fourth and fifth generation iPod Nanos at present.

Immediately after starting the app, the user is prompted to either create a new Dropbox account or log in to an existing one; data collected during a shaking session is written to a file which can then be uploaded to Dropbox, a website that enables users to store and share files. Once linked with a Dropbox account, the user must specify the shaking duration (in seconds), user posture while shaking, and sampling rate (in hertz) that will be used in an upcoming shaking session. The default settings are ten seconds, sitting posture, and 50 hertz, but users can select a longer shaking duration of twenty seconds, a standing posture, or either a slower sampling rate of twenty-five hertz or a faster sampling rate of 100 hertz. The user can begin a shaking

session with another device as soon as identical settings have been selected on both devices.

Once a shaking session has been completed, each of the two participating devices contains a text file of approximately *selected_frequency* $\times$ *selected_duration* samples, where one sample consists of an acceleration measurement in the X dimension, an acceleration measurement in the Y dimension, and an acceleration measurement in the Z dimension. Each measurement is somewhere in the range of $(-2.1, 2.1)$. Each text file is named with the device's unique ID, the user's name (as entered by the user before beginning a shaking session), the user's selected session settings, the date, and the time. Finally, the user is given the option to upload the file in his/her device (the file is automatically deleted after the upload) or to start a new shaking session with the same settings (the file is automatically deleted before the new shaking session begins). Alternatively, the user can go back to the previous screen using the button on the Navigation Bar if he/she wishes to select new settings.

5 Implementation

Using MATLAB, we implemented the coherence measure described in Section 2 and the Hamming distance measure described in Section 3. Once the user has performed a shaking session, uploaded the collected data to Dropbox from both devices, and downloaded the corresponding text files from Dropbox to a computer, the user can then run the MATLAB program, DataEvaluation.m. Let a *data set* be defined as one $m \times 3$ matrix representing the recorded movement of a given device in an arbitrary shaking session, and let an *experiment* be defined as p consecutive shaking sessions all using the same settings; thus, an experiment results in a total of $2p$ data sets where p data sets come from device A, and the remaining p data sets come from device B.

5.1 MATLAB Code

The program begins by asking the user to specify how they would like to evaluate the available data sets. The current output options include the following:

- A set of receiver operating characteristic (ROC) curves [6] created by applying the coherence measure to multiple experiments performed by multiple users.
- One ROC curve created by applying the coherence measure to one experiment performed by one user.
- One ROC curve created by applying the Hamming distance measure to the same individual experiment mentioned in the bullet above.
- A graphical display of an application of MATLAB's built-in Procrustes transformation to the same individual experiment above.

The user is also asked to provide some values that will be used by the different measures such as the number of slices to split the time series into (used in both coherence and Hamming distance evaluations) and the maximum frequency used as the upper bound in the summation that computes the final coherence value (coherence only).

The code reads in text files from a specified directory under the assumption that the files are sorted in alphabetical order within the directory. Sorting the files in this manner ensures that for a given experiment, all the data sets from device A are above all the data sets from device B (without loss of generality), and all the data sets from device A (B) are further sorted by the time of day that the data was collected. In other words, as long as the files are sorted alphabetically, data set i's correct match is data set $i + p$ (where i goes from 1 to p). Consider the following illustrative example with $p = 3$:

DataSet1: DeviceA_12:01PM

DataSet2: DeviceA_12:05PM

DataSet3: DeviceA_12:13PM

DataSet4: DeviceB_12:01PM

DataSet5: DeviceB_12:05PM

DataSet6: DeviceB_12:13PM

DataSet1's correct match is DataSet4, DataSet2's correct match is DataSet5, and DataSet3's correct match is DataSet6. Note that the actual file names

consist of more data than the file names in the above example, but these were shortened for simplicity. The term *correct match* denotes the pairing of two data sets such that one data set came from device A, one data set came from device B, and the two data sets were collected during the same shaking session where device A was shaken with device B. It is very important to pay attention to this assumption because both the creation of ROC curves and the implementation of the Procrustes transformation rely on the proper matching of data sets. It is possible to properly match data sets based on the time at which they are collected (similar to how the Bump pairing protocol works [10]), and this approach could be explored in future work on this project in order to eliminate the need for this assumption.

5.2 ROC Curves from Coherence

An ROC curve depicting the application of the coherence measure to an arbitrary experiment is created in the following way. The program iterates through the data sets in a specified directory, and it matches each data set with every other data set exactly once where order does not matter (that is, after data set i has been matched with data set j, data set j will not get matched with data set i later in the iteration). Thus, for p shaking sessions (i.e. $2p$ data sets), there will be $p(2p-1)$ total pairs, of which p are correctly matched and $2p(p-1)$ are incorrectly matched. For each pairing of files, the coherence measure is applied to the files' corresponding matrices and results in a final coherence value of C_{xy} which is stored in the following vector:

$$\left[\begin{array}{cc} C_{xy} & b \end{array}\right], \qquad b = \begin{cases} 0 & \text{if the files are incorrectly matched} \\ 1 & \text{if the files are correctly matched} \end{cases}$$

All of the vectors are appended together to create a $p(2p-1) \times 2$ matrix that is used to calculate the false positive and true positive rates for that experiment; these will be defined shortly. A threshold θ is varied from a minimum value of θ_{min} to a maximum value of θ_{max}. For each value of θ, the program iterates through the coherence matrix and compares each C_{xy} to θ. If $C_{xy} \geq \theta$, then it is asserted that the two matrices associated with that particular C_{xy} came from two devices that were shaken together, and it is asserted that the two matrices associated with that particular C_{xy} came from two devices that were not shaken together otherwise. Next, the corresponding b is considered in order to determine whether this assertion is

a true or false result. Let a *true positive* be defined as an assertion that two devices were shaken together when a particular C_{xy}'s corresponding $b = 1$; that is, an assertion that two devices were shaken together when they were actually shaken together. Let a *false positive* be defined as an assertion that two devices were shaken together when $b = 0$, a *true negative* be defined as an assertion that two devices were not shaken together when $b = 0$, and a *false negative* be defined as an assertion that two devices were not shaken together when $b = 1$. Thus, the false positive rate is the fraction of false positives out of the total negatives, and the true positive rate is the fraction of true positives out of the total positives. Finally, the ROC curve is created by graphing the true positive rate versus the false positive rate for each θ.

5.3 ROC Curves from Hamming Distance

There are some key differences between the creation of an ROC curve depicting the application of the Hamming distance measure to an arbitrary experiment and the creation of an ROC curve depicting coherence analysis. The most obvious difference is the lack of final coherence values to compare to the varying values of θ. Instead, this approach uses the coherence method to obtain the $x_k(f)$ and $y_k(f)$ vectors (as described in Sections 2 and 3), and after multiplying these complex vectors by their respective complex conjugates to get real-valued vectors, projects these vectors onto each of r random uniform vectors where r is specified by the user. Let a *projection pair* be defined as the projections of $x_k(f)$ and $y_k(f)$ onto the same random uniform vector. Since these r vectors are generated randomly, the final ROC curve looks slightly different each time this function is called. To account for this randomness, the code performs r projection pairs a total of r times and averages the results. We chose to perform the r projection pairs r times for the simple reason that we did not wish to introduce another parameter (certainly, anyone implementing this work in the future may adjust this number as they see fit for their own purposes).

Throughout a single iteration of r projection pairs, two binary vectors, $\vec{b_1}$ and $\vec{b_2}$, are built up where each entry in each respective vector represents whether the projection of $x_k(f)$ ($y_k(f)$) onto each random vector is positive or negative. Once all of the r projections pairs for that iteration have been completed, and thus $\vec{b_1}$ and $\vec{b_2}$ have been completed, the program computes the Hamming distance between $\vec{b_1}$ and $\vec{b_2}$ to determine how many projection pairs had the same sign and how many projection pairs had opposite signs.

In other words, consider row i in $\vec{b_1}$ and $\vec{b_2}$. If $\vec{b_1}$ and $\vec{b_2}$ both have either a 0 or a 1 (where 0 represents a negative projection, and 1 represents a positive projection) on this row, the number of same sign projection pairs is incremented. However, if $\vec{b_1}$ has a 1 (0) and $\vec{b_2}$ has a 0 (1) on this row, the number of opposite sign projection pairs is incremented. The final value that gets compared to the varying values of θ is the fraction of same sign projection pairs over the total number of projection pairs.

5.4 The Procrustes Transformation

The Procrustes transformation is useful because it finds a linear function that optimally brings two sets of data points in $\mathbf{R}^n$ into alignment. MATLAB's built-in Procrustes transformation takes two matrices A and B, determines a linear transformation to apply to the values in B in order to conform them to the values in A, and returns both the transformation and the resulting matrix Z from applying the linear transformation to B. Currently, DataEvaluation.m reads in the text files from a specified directory, matches the data sets based on user input declaring whether to match the data sets "correctly" or "incorrectly," and outputs a set of graphs that display A, B, and Z. The correct matching case pairs each data set with its proper match while the incorrect matching case pairs each data set with the data set one index below it in the directory. In other words, under the assumption described earlier in this section, the incorrect matching case pairs each data set with another data set collected by the same device but during a different shaking session. While this code is not currently used with either the coherence or Hamming distance measures, it could be used to obtain rotation invariant matrices that could later be compared using variations of those two methods. This will be discussed further in the Future Work section.

6 Testing and Analysis

In order to test the ability of the coherence and Hamming distance measures to successfully classify devices as shaken together or shaken separately, we performed an initial experiment with one user, $p = 10$ (number of shaking sessions in the experiment), and using the default app settings mentioned in Section 3; we call these files the Original Test Files and will refer to this initial experiment as the Original Experiment. This experiment, as well as

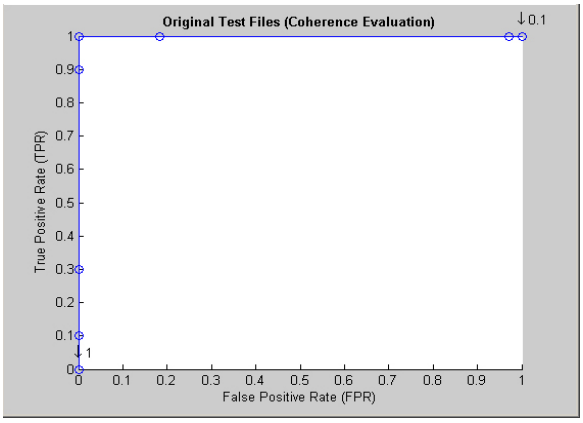

Figure 1: *ROC Curve using* $n = 20$ *and* $f_{max} = 5$.

all subsequent experiments discussed in this section, were performed using an iPhone 4 and an iPod Touch, both running iOS 5.1.

6.1 Initial Testing of the Coherence Measure

After running the coherence analysis code using the Original Test Files many times with varying the values of both n and f_{max}, we found some of the strongest results, which are depicted in Figure 1. This figure shows the resulting ROC curve from applying the coherence measure to each of the possible data set pairs in the Original Experiment. According to the default settings, a sampling rate of 50 Hz and a shaking duration of ten seconds implies each of the twenty data sets have approximately 500 samples. The ROC curve in this figure was created by utilizing the coherence measure with $n = 20$ slices (i.e. a window size of approximately 25 samples) and a maximum frequency of $f_{max} = 5$ Hz. Each point on the graph represents the true positive rate (TPR) vs. the false positive rate (FPR) as calculated for a particular threshold value θ where θ is varied in increments of .1 over the range of .1 to 1 to make up the whole graph. As expected, larger values of θ yield a smaller FPR, but this comes at the expense of a smaller TPR as well. In Figure 1, $\theta = .4, \theta = .5$, and $\theta = .6$ all yield a perfect $1 : 0$ TPR to FPR ratio, or the maximum area under the curve (AUC). For $n = 10$, the best θ's (that is, the θ's with the largest AUC's) occurred between .3 and .5 for both small and large f_{max}'s, and for $n = 5$, the best θ's occurred between .4 and .5 for large f_{max}'s and between .8 and .9 for small f_{max}'s. From this data, one

	Sampling Rate (Hz)	User Posture	Shaking Duration (sec)
Experiment 1	100	Sitting	10
Experiment 2	50	Sitting	10
Experiment 3	50	Sitting	20
Experiment 4	50	Standing	10
Experiment 5	25	Sitting	10

Table 1: *Experiment settings used in robustness testing*

can conclude that smaller values of f_{max} result in the initial decrease of TPR occurring at larger values of θ. In other words, smaller values of f_{max} result in smaller overall coherence values. The largest possible θ with the largest possible AUC is most desirable; the larger the threshold, the less likely it is to classify two truly dissimilar data sets as similar, but the threshold cannot be too large otherwise even the truly similar data sets will be classified as dissimilar.

6.2 Robustness Testing of the Coherence Measure

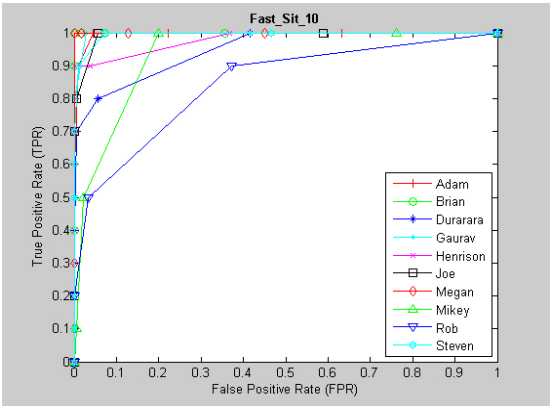

Figure 2: *Exp. 1*

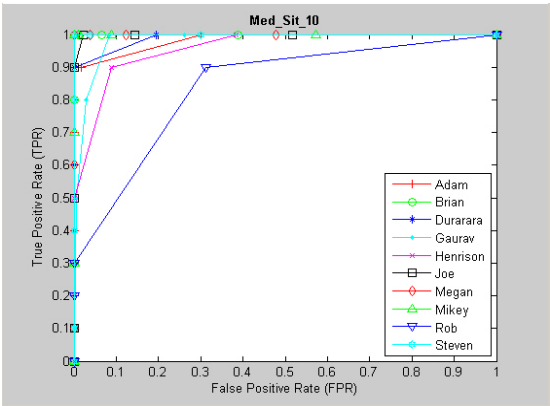

Figure 3: *Exp. 2*

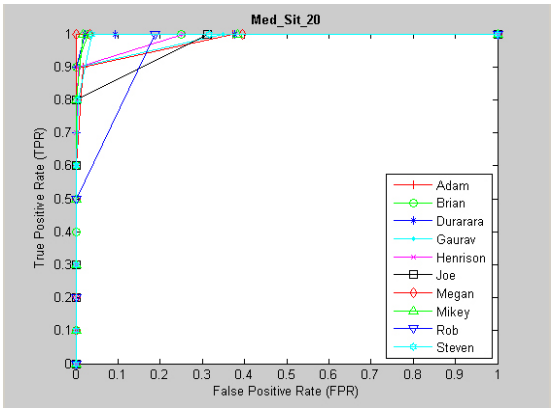

Figure 4: *Exp. 3*

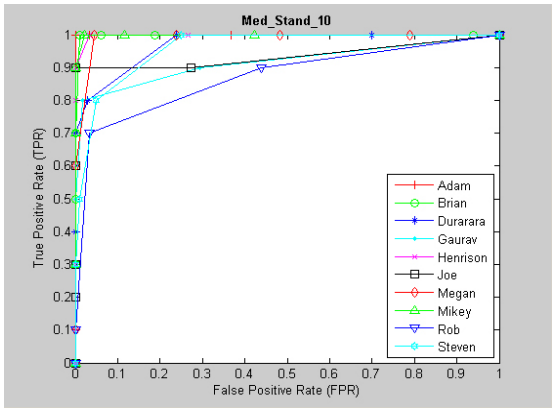

Figure 5: *Exp. 4*

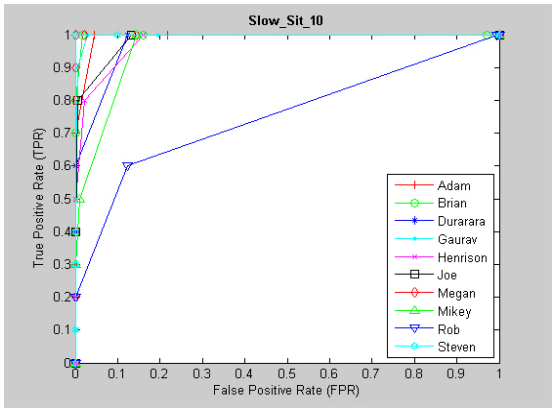

Figure 6: *Exp. 5*

We performed further robustness testing by asking ten different users to each perform five different experiments, each with $p = 10$ shaking sessions. The various experiment settings are highlighted in Table 1. Each of the users found the app easy to use and were able to successfully perform the experiments with little to no instruction. The following discussion encompasses ROC curves created with $n = 20$ slices and $f_{max} = 5$ Hz, the same values as used above with the ROC curve for the Original Test Files.

When comparing the effect of user posture in the ROC curves of Experiment 2 and Experiment 4, one can easily see that Experiment 4 has faster decreasing TPR's than Experiment 2. One observation made while watching users perform the various experiments is that when shaking while sitting, most users tend to rest their shaking arm on a nearby surface or in their lap.

This unconscious action could potentially be causing users to unknowingly limit both their shaking range and motion. In contrast, when shaking while standing, users tend to create a larger shaking range with a more vigorous motion due to the lack of physical restriction in the shaking vicinity. It is possible that the difference in the shaking ranges and/or motions could be responsible for the more rapidly decreasing TPR's in the standing case.

When comparing the effect of sampling rate in the ROC curves of Experiments 1, 2, and 5, we noted that the AUC's all start decreasing around the same (small) FPR values in Experiment 5 but that the AUC's start decreasing at different FPR values in Experiments 1 and 2; that is, the FPR's drop from 1 to values $< .2$ for small values of θ in the slowest sampling rate experiment while the FPR's decrease from 1 much more steadily in the medium and fastest sampling rate experiments (one can easily see from the graphs that the FPR's touch values between 1 and .2 in Experiments 1 and 2 but not in Experiment 5). This is the expected result because a slower sampling rate means fewer samples, which in turn implies smaller values of f_{max}, and as discussed previously, smaller values of f_{max} result in smaller overall coherence values. Therefore, the probability of falsely classifying dissimilar data sets (with a slow sampling rate) as similar becomes small at small values of θ, whereas the probability of falsely classifying dissimilar data sets (with a medium or fast sampling rate) as similar becomes small at medium to large values of θ.

When comparing the effect of shaking duration in the ROC curves of Experiments 2 and 3, it is clear that a longer shaking duration produces larger AUC's. In fact, Experiment 3 yielded the best results out of all five experiments. This is the expected result since larger data sets are more difficult for an adversary to successfully replicate. Unfortunately, a longer shaking duration is not a user-friendly option for real-world applications of this work; most people would not want to shake two devices for twenty seconds or more to simply initialize a transaction. From observing Figures 2 to 6, almost all users manage to obtain at least one large AUC for some value of θ in each of the experiments. This indicates that the coherence measure can robustly differentiate between devices shaken together and devices shaken independently.

6.3 Testing the Hamming Distance Measure

To test the performance of the Hamming distance measure, we go back to using the Original Test Files. However, we now vary θ in increments of .01 from .85 to 1. Using the same $n = 20$ as before and projecting onto $r = 50$ random vectors, we found that there is a much more gradual initial decrease in TPR here than in any of the ROC curves for coherence (see Figure 7). Note that we experimented with larger values of r, but they made little to no difference in the results. The largest AUC's are smaller in this case than the largest AUC's in the coherence analysis case, and as evidenced by the new range of values for θ, they occur for very high values of θ. The high values of θ can be attributed to the fact that the number of same sign projection pairs is overwhelmingly larger than the number of opposite sign projection pairs regardless of whether a data set pair is a true match. This is a surprising result since the expectation was that data sets coming from separate shaking sessions would yield $x_k(f)$ and $y_k(f)$ vectors that were very far apart from each other. Since a greater distance between arbitrary $x_k(f)$ and $y_k(f)$ vectors would result in a high probability of generating random vectors between them, more projections should go in opposite directions for improperly matched data set pairs. Currently, we are generating random vectors with all entries chosen uniformly at random from the interval $[-1, 1]$, but it is possible that improvement in the generation of random vectors could improve the overall AUC's. Regardless of whether r is increased or decreased

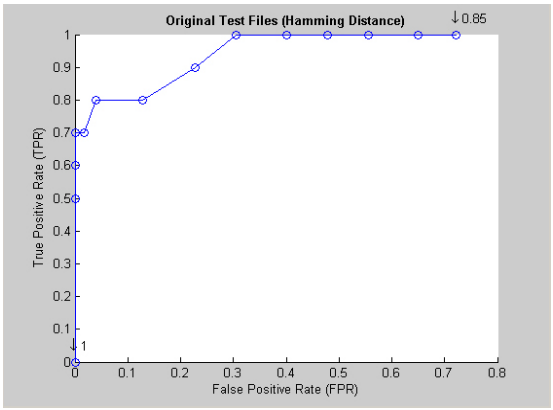

Figure 7: *ROC Curve using* $n = 20$ *and* $r = 50$.

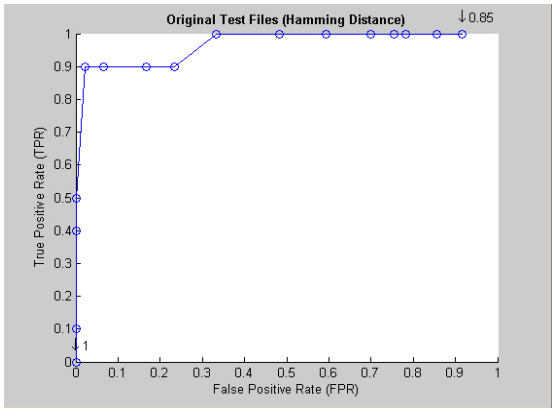

Figure 8: *ROC Curve using* $n = 10$ *and* $r = 50$.

while n stays the same, the AUC's follow the same general pattern where they begin to decrease from 1 at $\theta = .9$ (and where the FPR is around .3). If n is decreased while r remains static, the AUC's increase significantly, achieving areas close to 1 for $n = 10$ (see Figure 8). However, if n is increased while r remains the same, the AUC's worsen such that they begin to decrease from 1 at $\theta = .86$ (and where the FPR is around .4) for $n = 30$. It is also worth noting that for smaller values of n, the FPR achieves higher values for the starting value of $\theta = .85$ (for $n = 10$, the FPR is close to .9, for $n = 20$, the FPR is close to .7, and for $n = 30$, the FPR is close to .5). This makes sense because the more slices an arbitrary time series is split into, the fewer samples fit into a given slice, and that implies the FFT function is applied more frequently but to smaller segments since it is applied to each individual slice. Splitting a time series into fewer slices should naturally result in more false positives since bigger slices provide less accuracy. Overall, since it is possible to adjust the parameters of the Hamming distance evaluation to obtain AUC's very close to 1, it can be concluded that this measure robustly differentiates between devices shaken together and devices shaken separately.

7 Future Work

This project can be taken in many different directions in the future. The most obvious, but completely separate, direction is to investigate ShaCK and either improve it or come up with a better alternative to constructing a secure key. A more relevant direction is exploring the possible extension of coherence analysis from $\mathbf{R}$ to $\mathbf{R}^3$. By taking two data sets and immediately transforming them into rotation invariant vectors, a lot of entropy is lost. The Procrustes transformation can help here; after applying this transformation, time series analysis could be performed on the individual X, Y, and Z vectors of the two now rotation invariant matrices resulting in a final coherence value for each dimension.

In terms of future robustness testing, it would be interesting to see if data sets differ based on the hand in which the user shakes the devices; shaking movements may vary depending on whether the user is shaking the devices in his or her dominant versus non-dominant hand. To improve the usability, this work could be extended to include devices that are not compatible with iOS. One very important open question to consider is the following: how is this work affected on a device where the accelerometer is not performing as

expected? Perhaps the device has been unintentionally damaged somehow, or it is simply old. It may also be possible for an adversary to open a device and alter the accelerometer to behave in a certain way. This is a legitimate concern because there may not be any outward signs that a given accelerometer is behaving erratically, and without proper warning, the average user would have no reason to suspect anything was wrong.

8 Conclusion

In this work, we addressed one potential solution to the problem of how to securely pair mobile devices that wish to communicate with each other. In our approach, the mobile devices simultaneously undergo a shared experience and subsequently use data collected during this shared experience to verify a previously exchanged session key. We defined a shared experience to be the act of physically holding two devices together and shaking them for a specified amount of time. We introduced and explained the iOS app used to collect data during a shaking session using the accelerometer in an iOS compatible device. We then examined two separate methods that we claim can robustly classify devices as shaken together or shaken independently. Both of these methods, when applied to a pair of data sets collected using the app, return a positive, real value that gets compared to a threshold value to determine the classification of the devices in question. The act of holding two devices together and shaking them is both easy to learn and easy put into practice, as shaking is a natural human motion. Users would not be required to learn any complex movements, remember any type of password or secret information, or buy any additional equipment beyond the mobile device itself. This ease of use combined with the positive results discussed in this paper make it plausible to consider an extension of this work into real-world scenarios. While the current results of this work are promising, there is a lot more to be done to improve and/or extend these findings.

References

[1] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith. Secure remote authentication using biometric data. In Cramer [2], pages 147–163.

[2] R. Cramer, editor. *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*. Springer, 2005.

[3] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

[4] Y. Dodis, B. Kanukurthi, J. Katz, L. Reyzin, and A. Smith. Robust fuzzy extractors and authenticated key agreement from close secrets. *IACR Cryptology ePrint Archive*, 2010:456, 2010.

[5] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[6] T. Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, June 2006.

[7] J. Lester, B. Hannaford, and G. Borriello. Are You with Me? - Using Accelerometers to Determine If Two Devices Are Carried by the Same Person. In A. Ferscha and F. Mattern, editors, *Pervasive Computing, Second International Conference, PERVASIVE 2004, Vienna, Austria, April 21-23, 2004, Proceedings*, volume 3001 of *Lecture Notes in Computer Science*, pages 33–50. Springer, 2004.

[8] R. Mayrhofer and H. Gellersen. Shake well before use: Intuitive and secure pairing of mobile devices. *IEEE Transactions on Mobile Computing*, 8(6):792–806, June 2009.

[9] R. L. Rivest and A. Shamir. How to expose an eavesdropper. *Commun. ACM*, 27(4):393–394, Apr. 1984.

[10] A. Studer, T. Passaro, and L. Bauer. Don't bump, shake on it: the exploitation of a popular accelerometer-based smart phone exchange and

its secure replacement. In *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC '11, pages 333–342, New York, NY, USA, 2011. ACM.