Evaluation of Apple iDevice Sensors as a Potential Relay Attack Countermeasure for Apple Pay

Gareth Haken
Royal Holloway, University of London, Egham, UK
gareth.haken.2013@live.rhul.ac.uk

Konstantinos Markantonakis
Royal Holloway University of London, Egham, UK
K.Markantonakis@rhul.ac.uk

Iakovos Gurulian
Royal Holloway University of London, Egham, UK
Iakovos.Gurulian.2014@live.rhul.ac.uk

Carlton Shepherd
Royal Holloway University of London, Egham, UK
Carlton.Shepherd.2014@live.rhul.ac.uk

Raja Naeem Akram
Royal Holloway University of London, Egham, UK
RajaNaeem.Akram.2008@live.rhul.ac.uk

ABSTRACT

Traditional countermeasures to relay attacks are difficult to implement on mobile devices due to hardware limitations. Establishing proximity of a payment device and terminal is the central notion of most relay attack countermeasures, and mobile devices offer new and exciting possibilities in this area of research. One such possibility is the use of on-board sensors to measure ambient data at both the payment device and terminal, with a comparison made to ascertain whether the device and terminal are in close proximity. This project focuses on the iPhone, specifically the iPhone 6S, and the potential use of its sensors to both establish proximity to a payment terminal and protect Apple Pay against relay attacks. The iPhone contains 12 sensors in total, but constraints introduced by payment schemes mean only 5 were deemed suitable to be used for this study. A series of mock transactions and relay attack attempts are enacted using an iOS application written specifically for this study. Sensor data is recorded, and then analysed to ascertain its accuracy and suitability for both proximity detection and relay attack countermeasures.

Keywords

Relay Attacks; Apple Pay; Ambient Sensors

1. INTRODUCTION

Contactless smart cards are now used extensively in transportation and banking systems, evidenced by the increase of contactless transactions. The technology was first introduced by UK banks in 2007 [1] and for the first half of 2015 £2.5bn worth of transactions were made using a contactless card, prompting authorities to increase the maximum spend for a single transaction from £20 to £30 [2]. This growth in contactless card transactions, especially over the last year, seems to indicate that such systems will continue to grow as awareness increases and fears over security issues wane.

Communication between contactless smart cards and readers is facilitated with Radio Frequency IDentification (RFID) technology, which uses radio waves to encode information. RFID systems contain two components, a transponder, found on the object to be identified, and the interrogator, which is able to interpret signals from the transponder [3].

ISO 14443 is a standard detailing technical parameters for RFID systems used in contactless smart cards. EMV contactless payment cards are therefore RFID systems which comply with the ISO 14443 standard. Near Field Communication (NFC) is another wireless data interface which complies with this standard.

NFC is a wireless communication technology that achieves data transmission using high frequency alternating magnetic fields operating at 13.56 MHz [3], meaning NFC interfaces can communicate with ISO/IEC 14443 compliant readers and transponders [3]. NFC can be viewed as a variant of High Frequency RFID which operates in the RFID High Frequency range of 13.56 MHz. Android and iOS both utilise NFC for contactless payment schemes. The devices run in Card Emulation Mode to emulate Debit/Credit cards which can be used with appropriate RFID readers to pay for goods, in exactly the same way as contactless payment cards.

Whilst NFC enables mobile devices to interact with RFID readers there must also exist some mechanism which provides the functionality and security afforded by the chip on contactless cards. In the iPhone 6S this functionality is provided by the Secure Element, which is defined by [4] to be "a tamper-resistant platform (typically a one chip secure microcontroller) capable of securely hosting applications and their confidential and cryptographic data (e.g. key management) in accordance with the rules and security requirements set forth by a set of well-identified trusted authorities".

The particular variant present in Apple devices is defined by [5] as an industry-standard, certified chip running the Java Card platform, that is compliant with financial industry requirements for electronic payments. In the case of iOS

CPSS'17, April 02 2017, Abu Dhabi, United Arab Emirates

DOI: http://dx.doi.org/10.1145/3055186.3055201

devices it takes the form of a dedicated chip [6] designed to store payment information securely. The NFC chip and Secure Element (SE) enable the payment device to emulate a payment card, conforming to the EMV standards [7].

This paper begins with a discussion of current theoretical attacks against EMV contactless transactions, including the relay attack and its potential threat to Apple Pay. Existing relay attack countermeasures are then discussed with reasons why they are unsuitable for mobile devices. Ambient sensors are then suggested as a means for establishing proximity and a full investigation is conducted into the sensors available on the iPhone 6S, with an evaluation of their suitability for deployment in an EMV contactless environment. One of the main considerations here is whether the sensor records enough data within the 500 ms limit imposed by the EMV contactless standards [8]. Transportation is another key area in which contactless technology has been adopted, where limits as low as 300 ms are imposed [9]. Consequently the 500 ms limit for this study is based on the upper limit from two key industries utilising contactless technology. The sensors chosen for further study are then discussed, along with test cases and how data will be processed to evaluate the sensor as a potential relay attack countermeasure. Results are then considered, followed by an evaluation, conclusion and areas for further work.

The objectives of this paper are:

- The creation of a test bed able to collect sensor data for both legitimate and illegitimate transactions. The question of legitimacy is discussed in 6.4
- Ascertain the suitability of the iPhone's ambient sensors to provide data which might be used in a proximity detection mechanism. Suitability is calculated on the refresh rate of the sensors and whether this allows accurate data to be recorded within the 500 ms limit imposed on EMV contactless transactions.
- Use the data recorded to calculate the effectiveness of an ambient sensor in detecting a relay attack whilst allowing genuine transactions to complete without issue.

2. ATTACKS TARGETING EMV CONTACTLESS TRANSACTIONS

This section examines some previous work that has been conducted around vulnerabilities with EMV contactless transactions and how these might be exploited.

2.1 Potential Attacks

A simple attack against contactless payment cards is theft of the card. No PIN is required during a contactless transaction so if an attacker steals a card they are able to use it for contactless payments without any verification [10].

Another attack reads data off the card just by passing a suitable reader within range, the read data is what would have been embossed on the old magstripe cards [11, 12, 13].

The attack outlined in [14] targets devices which are compliant with the EMV Kernel 2 specification [15], specifically its magstripe mode. The paper identified a weakness which meant a Unpredictable Number used to generate the cryptogram was limited to a value in the range 0 to 999. Through skimming and cloning a victims contactless card, with precomputed values for all potential Unpredictable Numbers, transactions could be fraudulently enacted.

2.2 Relay Attack

A relay attack is a variation on the man in the middle attack, it is conceptualised with the assistance of a Chess analogy in the Grand Master Chess problem [16], which uses a game of postal Chess to illustrate the process.

2.2.1 *Mafia fraud*

One of the first documented applications of this concept in an attack scenario was in [17] and was called the "Mafia Fraud". It involves 4 participants A, B, C and D. The attackers are B and C, where B is a Mafia restaurant owner and C is a collaborator who wishes to purchase an expensive diamond from D, who is a jeweller; A is an innocent customer in B's restaurant. The attack is explained in [17].

2.2.2 *Practical implementations*

The theoretical relay attacks discussed in [17] are translated into a meaningful physical implementation in [18], this demonstrated that relay attacks could be leveraged against the UK's Chip and Pin payment system. The practical implementation developed in [18] proved that the "Mafia Fraud" idea discussed in [17] could be realised, with modified hardware running custom software. The attack targeted EMV chip and pin payment systems and demonstrated the security of this system could be subverted using relay attack techniques [18].

Another practical implementation of a relay attack is detailed in [19], which targets contactless payment schemes. It seeks to challenge the assertion that RFID credit cards offer a higher level of security when measured against their magnetic stripe counterparts, a view offered by both [20] and [21]. The paper also seeks to prove that custom hardware and advanced skill is not required in order to implement an effective practical relay attack, by considering mobile smart phones as a cost effective attack platform. Modern mobile phones have flexible execution environments and easy to use application development kits allowing complex programs to be written and executed on the device. This means one device can be programmed to act as a reader, which steals a genuine token, and the other as a token which utilises the stolen data. Advanced communication mediums found on smartphone allow the two devices to communicate.

The major advantages of this implementation is summed up by the quote "The attack implementation requires no unlocking of devices or secure elements, no hardware or software modification to the phone platform, and minimal knowledge of the data that is to be relayed" [19]. The attack is relevant as contactless payment schemes increase in popularity [2, 22, 23] and NFC enabled mobile phones are now able to emulate these payment cards. Applying the principles of a relay attack to one of the mobile payment schemes removes some of the obstacles identified in the chip and pin attack. The main one being the need for modified hardware and the use of wires to connect the malicious token to the communication medium. All the necessary technology is encapsulated in the smart phone and its usage raises no suspicion. The paper concludes by stating that the relay attack implementation proved effective and easy to implement. It also states that an attack of this sort could threaten various

mobile payment schemes and urges that action be taken to protect such platforms against relay attack techniques[19].

2.2.3 Modes of Operation

Relay attacks can be executed in two distinct modes, passive and active [24]. When executing in passive mode packets of data are captured and transmitted to the genuine token so that a response can be calculated and relayed back to the genuine reader. A passive attack is able to bypass any application layer security protocols, even if they are based on strong cryptographic principles [19]. It does this by relaying whole packets of data to the genuine token, where any cryptographic protections are navigated and a response calculated. The token may then re-apply cryptographic protections and transmit the response back to the reader, via the apparatus used for the relay attack. Any cryptographic protections applied are then unpacked by the reader and the response accepted. An active attack differs in that data is not just relayed but can also be modified by exploiting known vulnerabilities. Relay attacks are therefore very dangerous with serious security implications [19]. Even in their most basic form a relay attack can circumvent any cryptographic protections applied to the data, in fact the attacker needs no knowledge of the data as merely relaying it is enough to circumvent RFID and NFC protection mechanisms.

3. APPLE PAY

Apple Pay is a mobile payment scheme [6] available on compatible Apple devices to enact transactions, in a similar way to contactless cards. The NFC [25] chip and Secure Element (SE) [25, 26] enable the payment device to emulate a payment card, conforming to the EMV standards [7].

3.1 Traditional Attacks and Apple Pay

Any attack that targets contactless cards lack of a verification method is generally defeated by Apple Pay. Touch ID [25] is Apple's proprietary fingerprint scanner, used for verification during payments. It would therefore be highly improbable for a thief to use Apple Pay if the device were stolen. There have been some successful attacks against Touch ID [27] but these are very specialist techniques and not easily applicable.

The attack investigated by Channel 4 news [13] relied on data being stolen from a contactless card, while still in the victim's pocket. This is another attack counteracted by Touch ID, in fact any attack that relies on RFID skimming is defeated by Touch ID. This includes relay attacks where the victims card is in a place outside direct control of the victim, such as a pocket.

One attack that Apple Pay fares no better protecting against than a contactless card, is the Mafia fraud relay attack. Touch ID has no effect here as the victim believes the malicious terminal to be a genuine one, and as such verifies using Touch ID.

3.2 Relay Attack against Apple Pay

We found no existing relay attacks which target Apple Pay, however an Android Application called "Spot.me" [28], provides insight into how probable such an attack is. This application allows users to capture an Apple Pay transaction token from a friend's iPhone, and use this token at a later date to purchase goods via a contactless terminal [29]. The user transmitting the token must still authenticate using Touch ID. This app has been removed from the Google Play store in the UK, potentially for security reasons.

4. COUNTERMEASURES FOR RELAY ATTACKS

Verification mechanisms on smart phones, such as Touch ID [30, 6], are used in mobile payment schemes to solve some of the vulnerabilities discussed. These do not however protect against relay attacks as described in the Mafia Fraud, so further countermeasures are required.

4.1 Distance Bounding Protocol

Relay attacks attempt to extend the distance between a legitimate user and reader. Countermeasures therefore attempt to verify the distance between the authorised token and the reader [31], ascertaining whether the authorised token is in range of the reader. One such technique is called distance bounding, and was first introduced by Brands and Chaum [32] for the prevention of Mafia Fraud attacks on Automated Teller Machine (ATM) cashpoints.

Distance bounding protocols for NFC exchanges are harder to implement as the short communication distances require high precision clocks and specialised hardware, which are expensive to implement [33]. In addition "conventional channels and, in particular the ISO/IEC 14443 communication channel are too slow for accurate distance bounding" [34].

4.2 Proximity Detection

Further countermeasures are discussed in [19] with only those not degrading user experience considered. One suggestion involves restricting the allowed time for a response, which was deemed unsuitable as it requires dedicated monitoring of the RF channel, adding overhead. Other suggestions centred on establishing the location of the two devices, thus establishing proximity [19]. Global Positioning System (GPS) readings are considered but hard to implement indoors. Other methods for establishing location are discussed in [19], with the aim being to establish proximity of the token in respect to the reader.

Mobile phones possess an array of sensors [35, 36, 37] allowing them to measure data from their environment. One concept involves leveraging these ambient sensors to establish a mobile payment device's position, in relation to a reader. Both devices record data, a comparison made and, if within some tolerance, proximity established.

4.3 Related Work

A solution to the Mafia Fraud is proposed in [38], which uses captured audio and ambient light as a basis for comparison. Audio yielded better results but the 30 second measurement time is far longer than the limit of 500 ms imposed on EMV transactions [8]. Paper [39] investigates whether a shared radio environment could be used for proximity detection. The paper demonstrates that measuring radio environments could be an effective way of establishing proximity, but collection times are above the 500 ms limit. Accelerometer readings are considered in [40] as a basis for comparison. Users tap the payment device against the reader twice, and accelerometers measure the vibrations generated. If they match then proximity is established. One problem we foresee is persuading users to tap their £700 smart phone against a reader. Saxena et al [41] discuss the use of GPS to provide

location information with which to calculate proximity. This method took around 10 seconds to collect data, which is too long, and does not function well inside. The final piece of related work, Akram et al [36], is directly related to the concept of using ambient sensors for proximity detection. The study involves two android devices which measure ambient data for a 500 ms period [8]. The paper found ambient sensors on Android devices were not suitable as an anti-relay mechanism, with the limited collection time of 500 ms [8] being an issue. This limit did not allow enough accurate data to be measured in order that a valid comparison could be made. All these pieces have focused on Android as the mobile platform, and hence the sensors available on Android devices. For these reasons this paper will focus on Apple Pay and the sensors available on Apple devices, specifically whether they are suitable for the implementation of a proximity detection and anti-relay mechanism.

5. AMBIENT SENSORS

With the growth in usage of Apple Pay, its potential susceptibility to relay attacks and a lack of suitable countermeasures for mobile devices, the investigation into whether ambient sensors can provide a countermeasure are valid. There also exists no current work into iPhone sensors in particular and as such they will be the focus of this paper.

5.1 Sensors in Mobile Devices

Mobile devices and mobile payment schemes provide new opportunities for attackers and security professionals alike. The plethora of on-board sensors found in most smart phones [35, 36, 37] could be utilised to measure ambient conditions, with the data captured used in a comparison mechanism to help establish proximity.

5.2 iPhone Sensors

The iPhone used in this study is the iPhone 6S running iOS 9.3.2, the software used to write the test application was xCode 7.3. Choosing the sensors to test involved answering two questions; does the sensor return adequate data in under 500 ms, and does it measure appropriate data whilst the device is static. Some of the sensors only measure data when the phone is in motion, such as the Gyroscope discussed below. Given the nature of contactless transactions any method which required either payment terminal or device to be in motion could seriously degrade user experience. Appropriate data here is defined as data that measures either ambient surroundings, location or orientation of the device. A third party iOS app [42] and various research sources were used to answer these questions.

5.2.1 iOS Frameworks

In iOS a framework is a dynamic library linked to at compile time, they are either public or private [43]. Public frameworks are permitted in iOS apps seeking inclusion in the app store, they are well documented and easy to access [43]. Private Frameworks are not permitted for use in apps for the app store. They are not documented or easy to access, and are unstable against firmware changes [43].

5.2.2 Accelerometer

The accelerometer measures changes in velocity along a linear path. The device has three accelerometers, one for each axis meaning device movement can be measured in any direction, giving device orientation. The update rate can be set to 100 Hz [44], which was tested using a third party app [42]. Data collected shows that substantial volumes of sensor data are recorded within 500 ms.

5.2.3 Attitude

The Attitude sensor measures the Roll, Pitch and Yaw of the device. Use of [42] proved this measurement indicated orientation with an acceptable update rate of 30 Hz.

5.2.4 Bluetooth

Because the NFC chip in Apple devices can only operate in card emulation mode, and no developer access is allowed to the chip [45], building card reader functionality into the app is not possible. This means the use of NFC for communication between the two test devices is not possible and bluetooth was selected in its place. Bluetooth was not used as a sensor in the study itself as previous work in this field has indicated Bluetooth is not capable of returning sufficient data within the 500 ms limit [36].

5.2.5 Gyroscope

Use of [42] proved that this sensor only captures data whilst the device is in motion, and as such the sensor was not considered any further.

5.2.6 Light

The iPhone ambient light sensor is not easily accessible. It can be accessed through the public IOKit.framework, however this is discouraged by Apple [1] and as such the header file, IOKit.h, is not provided. There are various Github repositories which provide private iOS header files [46, 47], however neither repository has an entry for IOKit.h, making it difficult to access the ambient light sensor directly. Other techniques suggested involve using both the camera and screen brightness to establish ambient lighting. Both of these techniques have their own complications [48, 49]

5.2.7 Location

The iPhone 6S uses a combination of GPS and network statistics to establish location. Experiments with the update rate shows its about 1 Hz after the first second, however four readings are recorded in the first 300 ms and is therefore considered acceptable.

5.2.8 Magnetometer

The magnetometer in the iPhone 6S measures the magnetic field of the earth. It can be used to give device orientation as proven by experiments using [42]. The update rate of 40 Hz is also within the bounds required for this study.

5.2.9 Pressure

This sensor was not available on [42] and so a tailored application was written to test its suitability. The pressure sensor was deemed unsuitable due to its very slow update rate of 0.33 Hz.

5.2.10 Proximity

The proximity sensor is also unavailable on [42] and another tailored application was written to test it. Unfortunately the proximity sensor only returns a Boolean value indicating whether there is something in proximity of the

[1] http://iphonedevwiki.net/index.php/IOKit.framework

screen or not. This behavior was witnessed whilst running the app and corroborated with developer references [50].

5.2.11 Sound

A tailored app was written to investigate the recording and storage of sound. Both are straightforward and an interval for the recording can be specified. The sample rate for sound can be set to 44.1 kHz and the granularity of recording time can be set to 0.5 seconds so the sensor captures data in an acceptable time frame and format for this study.

5.2.12 WiFi

Obtaining a list of SSIDs in range of an iPhone 6S is not possible using standard frameworks in iOS 9. The only existing capability is the measuring of signal strength for the connected SSID. There is a private framework that can access WiFi strengths and data [2], and although not encouraged, access to Private frameworks was still possible prior to iOS 9. Unfortunately since the release of Xcode 7.3 the private frameworks have been removed entirely [51]. Given the complexity of capturing WiFi data, and average read times of 3873.5 ms [36], I have decided to consider it no further.

6. EXPERIMENTAL SETUP

Using the data for sensor read frequency gathered in Section 5, and considering the requirements discussed, five sensors were chosen for this study. Each one will be tested as to whether the data they collect is accurate enough to be used as the basis for a proximity detection system. These five sensors are now discussed below.

6.1 Sensors Chosen

The sensors were organised into two groups, those that measured ambient data and those that measured device orientation. Each one is now considered separately.

6.1.1 Accelerometer

Accelerometer data can be obtained from three locations within the same framework. The Core Motion framework's CMotionManager class includes a callback function that executes every time a new accelerometer reading has been recorded. This function includes a CMAccelerometerData instance as a parameter, which can be interrogated for the accelerometer readings. The readings are recorded in CMAcceleration objects, which each have three properties of type double, representing the readings along the X, Y and Z axes. The measurements are in units of gravity (G), with 1 G being equal to $9.806\,65\,\mathrm{m\,s^{-2}}$.

The second method for reading accelerometer data within the Core Motion framework is the deviceMotion property of the CMotionManager class. This property is of type CDeviceMotion and contains a property called userAcceleration which is of type CMAcceleration.

The third method of reading accelerometer data is the gravity property of CDeviceMotion. This operates in exactly the same way as the userAcceleration property but the data recorded is the total acceleration of the device, which is equal to gravity plus the acceleration the user imparts on the device [52]. All three of these methods will be used to gather accelerometer readings

[2]http://iphonedevwiki.net/index.php/MobileWiFi.framework

6.1.2 Attitude

The CDeviceMotion object is also the source for all data related to the attitude of the device. The attitude property of CDeviceMotion is of type CMAttitude, and this class contains three properties of type double for Roll, Pitch and Yaw readings. There are also two other representations of attitude data encapsulated in the CMAttitude class, these are attitude data as a rotation matrix and attitude data as a quarternion [52]. A rotation matrix in linear algebra describes the rotation of a body in three-dimensional Euclidean space [52]. It is represented by a CMRotationMatrix object which contains nine properties, each of type double, giving a 3 x 3 matrix. The data contained within the matrix can be used to calculate Euler angles [53], which describe the orientation of a body in three dimensional space. The Quaternion is represented by a CMQuarternion object which has four properties of type double. These values can also be resolved to Euler angles [54]. All three representations will be collected and used as part of the analysis, further details on how they will be used will be presented in Section 7.

6.1.3 Location

Location data is delivered via the CLLocationManager object and the didUpdateLocation delegate callback function of that object. This object has a desiredAccuracy property, which for these tests has been set to kCLLocationAccuracyBestForNavigation, this is the highest level of accuracy the iPhone can deliver and is ordinarily used for navigation. One of the parameters to this callback function is an array of CLLocation object, the last item being the latest location data. From this object latitude and longitude data can be obtained and this is the data which is recorded.

6.1.4 Magnetometer

There are three ways to measure magnetic fields in iOS9 on an iPhone 6S. The first two are part of the Core Motion Framework and utilise the CMMotionManager class of that framework. CMMotionManager has a property named magnetometerData of type CMMagnetometerData, which in turn has a property named magneticField of type CMMagneticField. This class has three properties of type double, which hold measurements for the X, Y and Z axes in µT. The values stored here represent raw magnetic field data, which is in contrast to the calibrated value of the magneticField property of the CDeviceMotion class [52].

The CDeviceMotion class is the second source of magnetic field data in the Core Motion framework. The CMotionManager class has a property called deviceMotion of type CMDeviceMotion. This in turn has a property called magneticField which is of type CMCalibratedMagneticField. This class also has three properties of type double which measure magnetic fields along the X, Y and Z axes, again in µT. The difference with these values is that they have been adjusted for any bias the device itself may have introduced.

The third method of measuring magnetic field data involves the Core Location framework and the CLLocationManager of that framework. The CLLocationManager contains a property named heading, which is of type CLHeading. The magnetic field values are stored in three double properties as for the other two examples. Each measurement represents an axis and is the deviation from the magnetic field lines of the device along that axis, measured in µT [52].

6.1.5 Sound

Sound recording is delivered using the AVAudio Recorder class. This can be initialised using an array of recordSetting objects to configure the quality and format of the recording. For these tests the quality of the audio has been set to the highest level and the format chosen is the .wav file format. This format was chosen as it does not compress the audio and has a good level of compatibility with python libraries used in the analysis phase in Section 7. Other formats, such as Apple's Lossless format, recorded more data for the given time period, but were much harder to analyse using standard scientific libraries, such as the python module SciPy.

6.2 iOS Test Application

There is only one use case for the test application, which is to record sensor data and store it in an appropriate format. Two devices running the application will need to ensure sensor data collection is synchronised. This gives the following requirements:

- Application should coordinate sensor data collection.
- Application should sensor data to be stored on the device in an appropriate format.
- Data should be easy to extract from the device.
- Application should allow two copies of the application on different devices, to synchronise with each other.

One major design problem was achieveing synchronisation across the two devices. NFC could not be used as Apple does not allow developers full access to the NFC chip architecture [45], so Bluetooth was chosen in its place. It is not thought this will have any negative impacts on the study as the aim is to compare sensor readings from two devices in close proximity, the choice of communication medium is not thought to be integral to this.

The collection of sensor data is achieved through the app, and stored in the form of comma seperated value files in the documents directory of the application. A directory will be created with the name of the test input by the user, and then saved to the documents directory. This directory is accessible from iTunes if the correct entries are set in the applications plist. This means once all the tests are complete, the device can be synced with iTunes and all the files downloaded. All the App's Objective C source code[3] and python analysis code[4] is available on github.

6.3 Test Design

The iOS application will be installed on two iPhone 6S devices, for tests at close range one will act as a payment Device and the other as the Terminal. Once paired, pushing start on the payment Device (Bluetooth master) executes the sensor measuring routine on both devices simultaneously. If non-Bluetooth mode is selected then tests are started on the two devices independently, with manual synchronisation between the people in command of the devices.

Once the tests have been started each sensor is tested sequentially for a period of time, these are discussed in Section 7 and changed a few times during development to accommodate some of the analysis features introduced. For each sensor, data is collected at the maximum rate allowed by iOS, except the microphone which collects samples continuously. All data is saved in comma seperated value file format with an epoch time stamp for each entry, again the microphone is an exception to this as sound files are saved individually in the .wav file format.

[3]https://github.com/CPSSAP/BLTE_Transfer_Clean
[4]https://github.com/CPSSAP/Python-Analysis-Code

6.4 Test Cases and Environments

For both the Orientation and Ambient sensor groups a series of test cases were designed to allow for large quantities of sensor data to be recorded. For both sensor groups 400 legitimate and 400 illegitimate transactions were enacted, this is 40 runs of the test software which records 10 transactions for each run. Legitimate transactions are those that represent a genuine contactless transaction, illegitimate transactions represent simulated relay attack attempts. For Orientation sensors the device and terminal have the same orientation for legitimate transactions and different orientations of a gradually increasing magnitude, for illegitimate transactions. Ambient sensor transaction test cases are simpler to describe as the demarcation between legitimate and illegitimate transactions is the distance apart of the Device and Terminal. The ranges for illegitimate transactions are $\approx 10m$, $\approx 25m$, $\approx 50m$, $\approx 100m$ and $> 1000m$. At each of these ranges eight tests are carried out, giving 80 transactions in total, and with 5 distinct ranges this gives a total of 400 illegitimate transactions. All transactions are conducted with the Device and Terminal inside a building as this is where most real world transactions will take place.

7. PERFORMANCE ANALYSIS

Each series of transactions were analysed in a certain way with different mathematical formulae used to illicit a valid comparison metric and subsequently a value for Equal Error Rate (EER). Each process is discussed in the coming sections.

7.1 Transaction Analysis

At this point in the study we have a series of comma seperated value files representing sensor readings and times at which they were taken. Each transaction consists of sensor readings from a Device and Terminal at roughly the same time. Some of these transactions are legitimate and some illegitimate, as described in the previous section. The 400 transactions were reduced to 380 once corrupted results and user error had been taken into account. The corrupted results were caused mainly by the device being moved accidentally whilst recording sensor data and a strict time frame meant these transactions could not be repeated.

The basic premise of transaction analysis was to iterate through all the transactions, and compare readings by calculating a similarity metric. This is then used to calculate the False Positive Rate (FPR) and False Negative Rate (FNR) for a series of thresholds, and subsequently the EER which is the point at which the plots intercept. A series of Python scripts were written to achieve this using various mathematical formulae which are detailed in Section 7.2 below.

7.1.1 Basic Overview of Analytical Process

Two comma seperated value files representing a transaction for a particular sensor are loaded and the timestamp of the first sensor reading for each is extracted. Whichever of the times is later is then used to synchronise the two data streams, as described in Section 7.3 below. The Device file

is then processed so that sensor readings for each entry are reduced to one value and all those values then collected and interpolated, this is done to assist in calculating the similarity metric. The same is done with the Terminal file and allows a common 500 ms period between the two samples to be found, values at 10 ms intervals can then be extracted, up to the 500 ms maximum. The two sets of interpolated sensor readings from both sides of the transaction are then used to calculate a single value representing the similarity of the two samples. For most sensors the similarity metrics calculated were the Mean Absolute Error (MAE) and Correlation Coefficient, but some sensors calculated other variants of these and are discussed individually below, with the relevant equations.

7.2 Sensor Specific Analysis Process

7.2.1 Accelerometer

The Accelerometer sensor delivers data in three distinct ways. Each one of these results in a vector, with components representing acceleration along the X, Y and Z axes. Before further analysis these values were reduced to one single value using Eq.1

$$M = \sqrt{x^2 + y^2 + z^2} \tag{1}$$

These values are then synchronised and interpolated, as described in Section 7.3, and used to calculate a measure of similarity between the two samples of a transaction. The first measure calculated is the MAE using Eq.2

$$MAE(D,T) = \frac{1}{N}\sum_{i=0}^{N} \mid D_i - T_i \mid \tag{2}$$

Where D and T represent sets of sensor readings taken from the Device and Terminal respectively, and which taken together make up a single transaction. The sets D & T must be of equal size in order for the calculation to be successful. The MAE represents the mean of all the absolute differences between the corresponding sensor reading pairs, for a specific time. The second measure of similarity used is the Pearson Correlation Coefficient and is given by Eq.3 below. As for the MAE D & T Represent sets of sensor readings from a single transaction.

$$\begin{aligned} a &= \left(\sum_{i=0}^{N} D_i T_i\right) - \left(\sum_{i=0}^{N} D_i\right)\left(\sum_{i=0}^{N} T_i\right) \\ b &= \left[\sum_{i=0}^{N} D_i^2 - \left(\sum_{i=0}^{N} D_i\right)^2\right] \\ c &= \left[\sum_{i=0}^{N} T_i^2 - \left(\sum_{i=0}^{N} T_i\right)^2\right] \\ PCC(D,T) &= \frac{a}{\sqrt{bc}} \end{aligned} \tag{3}$$

This measure returns a value in the range -1 to 1, with 1 indicating a very strong positive correlation, -1 a very strong negative correlation and 0 no correlation. There are different equations for calculating this measure so another implementation, Eq.4, was used to corroborate the values returned from Eq.3. A further implementation can be found in the scipy python library, this function was also used to corroborate results.

$$COR(D,T) = \frac{cov(D,T)}{\sigma_D \cdot \sigma_T} \tag{4}$$

Where cov(D, T) is given in Eq.5 and the standard deviation equation is at Eq.6

$$cov(D,T) = \frac{1}{N-1}\sum_{i=0}^{N}(D_i - \mu_D)(T_i - \mu_T) \tag{5}$$

Where μ_D & μ_T are the the mean values of D and T respectively

$$\sigma_x = \sqrt{\frac{\sum_{i=0}^{N}(x_i - \mu_x)^2}{N-1}} \tag{6}$$

Where μ_x is the mean value for the set x.

7.2.2 Attitude

The attitude sensor also records data in three different ways but, unlike the accelerometer, the vectors returned are of varying sizes. The first is a vector of size three which can be processed using the equations and methods detailed in the accelerometer section above. The second vector is known as a Quaternion and, as the name suggests, is a vector with four components. This vector can be converted into Euler angles using the python function below, which is a python conversion of Java code from [54]

```
from __future__ import division
import math

def quaternionToEuler(q1):
    x, y, z, w = q1[0:4]
    test = (x * y) + (z * w)
    heading = 2 * math.atan2(x, w)
    attitude = math.pi/2
    bank = 0

    if (test < -0.499):
        heading = -2 * math.atan2(x, w)
        attitude = math.pi/2
        bank = 0
        return

    sqx, sqy, sqz = x * x, y * y, z * z

    heading = math.atan2((2 * y * w) -
        (2 * x * z), 1 - (2 * sqy) - (2 * sqz))
    headingDeg = heading * (180/math.pi)
    attitude = math.asin(2 * test)
    attitudeDeg = attitude * (180/math.pi)
    bank = math.atan2(2 * x * w - 2 * y * z,
        1 - 2 * sqx - 2 * sqz)
    bankDeg = bank * (180/math.pi)

    return (headingDeg, attitudeDeg, bankDeg)
```

The convenient thing about this conversion is that there are now three values and they can be processed and analysed in the same way as those collected by the accelerometer. For this and further equations using Quaternions to be successful the Quaternions must be Unit Quaternions, which those collected by Apple devices are [52].

Another way of processing pairs of Quaternions involves obtaining the dot product. This measure gives a value in the range -1 to 1, similar to the Correlation Coefficient above, but for pairs of quaternions rather than whole sets. Subsequently the mean value is then calculated from the set of dot products to get a single value for similarity. The dot product is calculated using Eq.7, where D_q & T_q represent a pair of Quaternions recorded at the same time by the Device and Terminal respectively.

$$\begin{aligned} DOT(D_q, T_q) = (D_q x * T_q x) + (D_q y * T_q y) \\ + (D_q z * T_q z) + (D_q w * T_q w) \end{aligned} \tag{7}$$

One further method for processing quaternions was used and it involves calculating the difference in angles between two Quaternions, it involves using the dot product as described in Eq.7 and is calculated using Eq.8 below

$$\theta = acos((2 * dot(q1, q2)^2) - 1) \tag{8}$$

The mean was then taken of all the angles for each transaction, to give a single measurement of similarity. One final data structure used to record attitude data is a 3 x 3 Rotation Matrix. This was reduced to three Euler Angles which can then be processed using the equations and methods detailed in the accelerometer section above. The reduction to three Euler Angles is achieved using the method described in [53].

7.2.3 Magnetometer

The Magnetometer sensor is similar to the Accelerometer in that it returns sensor readings in vectors of size three, and can subsequently be processed in the same way as the Accelerometer.

7.2.4 Location

The location sensor returns data in the form of Latitude and Longitude measurements, which are processed using the Haversine Formula. This formula is able to calculate a distance between two points on a sphere given its radius. The sphere in this case is the Earth, the radius of which is different depending on where you measure but for this project the value 6372.8 km was chosen[5]. The equation, which is Eq.9 below, was implemented in python[5]. The formula gives D which is an approximate distance between the two pairs of lat lons, in km.

$$\begin{aligned} a =& sin((\phi_2 - \phi_1)/2)^2 \\ b =& sin((\lambda_2 - \lambda_1)/2)^2 \\ D =& R * \left(2sin\left(\sqrt{a + cos(\phi_1) * cos(\phi_1) * b}\right)\right) \end{aligned} \tag{9}$$

7.2.5 Sound

Sound was recorded in the .wav format as this was easier to manipulate using the scipy python library, and in mono as the microphone on the iPhone 6S is not stereo. Both sound files were opened using the read function of the scipy.io.wavfile module[6], which reads the recorded data into Numpy.Array data structures[7]. The two files can then be synchronised, which is discussed below, and a common 500 ms period identified. This results in two array structures of equal length and containing recording data from the two .wav files. These arrays are used as parameters to the Numpy.corrcoef function[8], which returns the Pearson product-moment correlation coefficient, this is similar to that used for other sensors above, giving a number in the range -1 to 1.

7.3 Data Collection Challenges

7.3.1 Time Taken to Collect Samples

To make detailed analyses vast amounts of transactions were recorded. To expedite this process every run of the iOS app resulted in 10 transactions being enacted. This introduced a problem with drift and it was observed that by the tenth transaction the drift had reached approximately 150 ms, which introduced problems with identifying a common 500 ms window. To solve this problem the period for sensor data collection was raised from 500 ms to 1 s, this allowed for a drift of up to 500 ms on legitimate transactions. For illegitimate transactions the collection period was set to 2 s to take into account both drift and any delay introduced by human operators.

7.3.2 Data Synchronisation

It was observed that Bluetooth communication introduced a 15ms delay on average, which grows to about 150 ms by the tenth transaction. To counteract this, the start time of both data streams was compared, the larger taken (known as time X) and then subtracted from the start time of both streams. This resulted in one stream having a time of zero and the other having a minus value. Time X is then subtracted from the time of the next item in the data streams, until both streams have a positive value for time. Whichever of the streams had the minus time values now has these records removed, and the remaining entries are shifted up to fill any gaps left.

7.3.3 Data Synchronisation Sound

Sound was harder to synchronise because it did not have any time recorded for each sensor reading. It did have a sampling rate which remained consistently 44 100 Hz throughout. Using the start time of two recordings the difference could be calculated (later time - earlier time) in seconds, and then multiplied by 44100 to get a figure for how many samples were taken in that period. The earlier recording can then have that number of samples subtracted from its start to bring it in line with the later recording. Both samples then have the first 22050 samples extracted, which represents 500 ms, with $44100/2 = 22050$.

7.3.4 Consistent Analytical Period

The limit on time for sensor data collection was 500 ms, as discussed previously. To implement this in the iOS app a timer was used in the Objective C code to stop each sensor after this time. Whilst the timer was very accurate there were varying numbers of sensor readings taken in this time, and sometimes at irregular intervals. A further problem was introduced by the time synchronisation method described above in that some records were removed in order to achieve

[5]https://rosettacode.org/wiki/Haversine_formula
[6]http://docs.scipy.org/doc/scipy/reference/io.html\#module-scipy.io.wavfile
[7]http://docs.scipy.org/doc/numpy/reference/generated/numpy.array.html
[8]http://docs.scipy.org/doc/numpy/reference/generated/numpy.corrcoef.html

synchronisation. A further problem was introduced by the multiple transactions for each execution of the software. To overcome these issues interpolation and longer sensor recording times were used. Following synchronisation the sensor readings for each time entry are reduced to one value, the method for which varies from sensor to sensor and is described above. This results in two data streams, each with a time and single value representing the sensor readings. Each separate data stream is then interpolated and values for 0 - 500 ms, at 10 ms intervals, are recorded. This results in two data streams with sensor readings taken over a 500 ms period at 10 ms intervals. Both data stream arrays are now of the same length and contain data from both Device and Terminal sampled at the same time.

The synchronisation method does introduce a new problem in that it removes some entries. If readings are taken over 500 ms this means one of the two data streams will be shorter. This problem was easy to solve as it just meant sampling over a slightly longer period, 600 ms was chosen for legitimate transactions and then increased to 1 s following addition of the multiple transactions feature, 2 s was chosen for illegitimate transactions as described above.

7.3.5 Zero Readings

The Magnetometer sensor in its Device Motion variant measures data at around a 10 Hz frequency. Whilst this is more than fast enough for effective data collection the first 100 - 150 ms returns zeros. Because of the shifting mechanism used to synchronise collection timings there can exist a large number of zeros on one side of the transaction and none on the other, which could skew the analysis. To negate this both sides of the transaction are analysed and all zeros on one side are changed to zeros on the other.

7.4 Calculating Equal Error Rate

As discussed above the EER is the point on a graph of FPR and FNR values for various thresholds, where the two lines intercept. At this point the rate for False Negatives and False Positives is equal. In the context of this analysis a False Positive is any illegitimate transaction which produces an MAE, Correlation Coefficient, or other similarity metric, which falls under the threshold value. In contrast a False negative is any legitimate transaction producing an MAE .etc which falls over the threshold value. One hundred threshold values are created, and range from the smallest to the largest MAE or correlation coefficient observed for each sensor. An ideal threshold value would have a very low rate of False Positives and Negatives, with all legitimate transactions falling beneath it and illegitimate transactions above it.

The True Positive Rate and True Negative Rate are then calculated using the equations Eq.10 and Eq.11. These values are then used to calculate the FPR and FNR using Eq.12 and Eq.13.

$$TruePositiveRate = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{10}$$

$$TrueNegativeRate = \frac{TrueNegatives}{TrueNegatives + FalsePositives} \tag{11}$$

$$FPR = 1 - TrueNegativeRate \tag{12}$$

$$FNR = 1 - TruePositiveRate \tag{13}$$

The FPR and FNR are calculated for each threshold and then plotted against each other to calculate the EER. In addition to using MAE to calculate the FPR and FNR, the correlation coefficient is also used, with thresholds ranging from -1 to 1.

7.5 Results

To summarise the results the tables below contain the EER and the threshold at which the EER occurs. As a general analysis any sensor with a low EER might be deemed a suitable sensor for establishing proximity and the threshold at which this EER sits would be the value used to differentiate between legitimate and illegitimate transactions. Table 1 below displays data for the orientation group of sensors, and Table 2 the data from the ambient group of sensors.

Table 1: Table showing optimal threshold (T) and Equal Error Rate for Orientation Sensor data

Sensor	MAE		Correlation	
	T	EER	T	EER
Acc1	0.0093	0.4539	-0.0022	0.4934
Acc2	0.0036	0.5316	0.0099	0.5184
Acc3	2.7×10^{-6}	0.4803	0.0249	0.4947
Att1	0.0214	0.0487	0.1091	0.5184
Att2	1.2103	0.0487	0.0967	0.5079
Att3	1.7120	0.025	0.0574	0.5105
Att4	(DIFF)		(DOT)	
	0.0645	0.0	0.9975	0.9632
Mag1	670.69	0.5	0.0078	0.4816
Mag2	9.2719	0.3224	0.0967	0.6079
Mag3	119.50	0.5526	0.0197	0.5145

Table 2: Table showing optimal threshold (T) and Equal Error Rate for Ambient Sensor data

Sensor	MAE		Correlation	
	T	EER	T	EER
Location	-	-	0.00214	(HAV) 0.54737
Sound	-	-	0.10764	0.49079
Mag1	649.269	0.631579	0.01830	0.49605
Mag2	8.48862	0.440789	0.98324	0.47763
Mag3	79.4930	0.468421	-0.01657	0.5

7.6 Analyses

7.6.1 Accelerometer

Each sensor variant was tested for similarity using the Mean Absolute Error and Correlation Coefficient, which leads to 6 different EERs. The best EER observed was 45%, this was from the Core Motion's CoreMotion Manager object using the MAE as a similarity comparator. For such a high risk environment as contactless payment infrastructures this would not be an acceptable EER as it would mean 45% of illegitimate transactions getting through and the same number of legitimate transactions being refused. This would

have a major impact on the usability and security of the system.

7.6.2 Attitude

The attitude sensors in their different variants delivered some very low EERs when using MAE as a comparison metric. Values of 4% were observed for both the attitude property of the Core Device Motion Manager and the Quaternion property when converted to Euler angles. A value of 2% was also observed for Rotation Matrices converted to Euler Angles. All these sets of readings only registered two positive values for FPR with the rest of the values for each threshold being 0. This could be down to the fact only gradients at 9 degree intervals were tested which may have some bearing on the EERs observed and consequently skewed the results. For the difference between Euler angles using Quaternions, as described in Eq.8 above, the EER observed was 0%. There were only four values observed for FPR though and this may have skewed results, as above. The Dot Product of Quaternions, as described in Eq.7, gave very unusual results in that the FPR was 1 for every threshold tested, except one. This led to a very high EER of 96%. One point of note for the various representations of attitude data is that they all produce very similar results, except for the Dot product calculations for Quaternions.

7.6.3 Magnetometer - Orientation

Magnetometer data can also be generated and recorded in three distinct ways, leading to three values for the MAE and three for the Correlation Coefficient. The best EER observed was from the deviceMotion property of the Core Motion Manager object, using the MAE as a similarity metric. The value observed was 32%, which for such a high risk environment as contactless payment infrastructures would still not be an acceptable level of false positives and negatives.

7.6.4 Location

The iPhone 6S does has a GPS chip but also uses other sensors to aggregate data from other sources and uses this to ascertain location. This means the iPhone is not solely reliant on GPS for location and this explains why it was able to deliver location readings whilst indoors and within the 500 ms constraint. Although data was obtained within the parameters required this data was not accurate enough to be used for both proximity detection and subsequently as a relay attack countermeasure. The EER observed for this sensor was 54% and not suitable for the purposes of relay attack detection. Given the specific formula used to process locastion data, the haversine formula as discussed, only one EER was calculated. This is in the Correlation column of table 2 but does not represent the Correlation value as this was not used as a comparison metric. Neither was Mean Absolute Error which accounts for the lack of data in this column.

7.6.5 Magnetometer - Ambient

This was the only sensor that was tested in both the orientation and the ambient groups and it fared better in the ambient data tests. As described above magnetic field data can be ascertained in one of three ways and the best EER reading observed came from the deviceMotion property of the Core Motion Manager object. This reading was 47% however and was was also deemed unsuitable for relay attack detection.

7.6.6 Sound

Ambient noise provided an EER of 49% and as such was also deemed unsuitable. Similar to location the data from this sensor was processed using a specific formula which only gave a value for correlation and not MAE, hence the lack of data in that column.

8. CONCLUSIONS AND FUTURE WORK

The aim of this project was to ascertain whether sensors found on the iPhone 6S could be used to establish proximity as a countermeasure to relay attacks directed against Apple Pay. The iPhone 6S carries 12 sensors of varying types, these include sensors which measure ambient data and orientation of the device. Any sensor used as a relay attack countermeasure must collect enough data within 500 ms, which is the maximum allowable time for contactless transactions. Because of this constraint these sensors were screened to create a shortlist of those that were capable of collecting data within this time frame. There were 5 sensors deemed suitable for analysis, the location sensor, accelerometer sensor, microphone, magnetometer sensor and attitude sensor.

For sensor readings to be a reliable countermeasure to relay attacks the EERs produced must be sufficiently low so that the amounts of false positives (relay attack attempt) and false negatives (genuine transactions denied) are kept to an absolute minimum. From the EERs calculated there were some down at 4% and even zero, this would indicate these sensors would be suitable for relay attack countermeasures. The underlying data for these sensors however, revealed very few values for the FPR which could have skewed the data towards a low EER. These low error rates do however mean that using the orientation of the device might have some potential for an effective relay attack countermeasure, although further work would be needed at finer granularity to prove this assertion. The remaining sensors all gave EERs at around the 50% mark and as such none of those tested would be suitable as a relay attack countermeasure. All source code from both the iPhone App and python test scripts is publicly available for scrutiny as described in Section 6.2.

A further interesting area of research is the use of Wi-Fi to create indoor GPS systems capable of locating devices to within 65 cm of each other, the concept was developed by Massachusetts Institute of Technology (MIT) and is explained at [55]. One problem with using GPS for proximity detection indoors is the lack of both a valid signal and the accuracy it offers, and as most contactless transactions take place inside this is a serious weakness. If devices are able to be located accurately within 65 cm of each other whilst inside, this would make an effective relay attack countermeasure as any attempted relay attack within this range should be easily detected by the legitimate actors in the transaction. If this location could be ascertained within a 500 ms window then this would potentially make an ideal method for both establishing proximity, and in turn detecting potential relay attacks. Should this technique be developed further then using it to implement relay attack countermeasures would be a worthy area of future research.

9. REFERENCES

[1] UK CARDS ASSOCIATION. History of cards: Timeline and milestones. [online] Available at: http://www.theukcardsassociation.org.uk/history_of_cards/index.asp. [Accessed 30 May 2016].

[2] BBC. Contactless card limit rises to 30 as card use surges. [online] Available at: http://www.bbc.co.uk/news/business-34110348, 2015. [Accessed 30 May 2016].

[3] K Finkenzeller. *RFID Handbook*. Wiley, third edition, 2010.

[4] GlobalPlatform. GlobalPlatform made simple guide: Secure Element. [online] Available at: http://www.globalplatform.org/mediaguideSE.asp, 2016. [Accessed 11 Jun 2016].

[5] Apple. iOS security, iOS 9.0 or later. [online] Available at: https://www.apple.com/business/docs/iOS_Security_Guide.pdf, 2015. [Accessed 09 Jun 2016].

[6] Apple. The safer way to pay. With your fingerprint. [online] Available at: https://www.apple.com/apple-pay/, 2016. [Accessed 09 Jun 2016].

[7] Apple. About EMV and Apple Pay for merchants. [online] Available at: https://support.apple.com/en-gb/HT205645, 2016. [Accessed 08 Jun 2016].

[8] EMVCo. *EMV Contactless Book A: Architecture & General Requirements*. EMVCo, version 2.6 edition, 2016.

[9] Smart Card Alliance. Transit and Contactless Open Payments: An Emerging Approach for Fare Collection. [online] Available at: http://www.smartcardalliance.org/resources/pdf/Open_Payments_WP_110811.pdf. [Accessed 22 February 2017].

[10] UK CARDS ASSOCIATION. What is Contactless? [online] Available at: http://www.theukcardsassociation.org.uk/individual/what-is-contactless.asp. [Accessed 16 January 2017].

[11] NFC Admin. How to read a contactless credit card such as Visa paywave or MasterCard paypass. [online] Available at: http://www.nfc.cc/2012/04/02/android-app-reads-paypass-and-paywave-creditcards/, 2012. [Accessed 04 Jun 2016].

[12] saush. Getting information from an EMV chip card with Java. [online] Available at: https://blog.saush.com/2006/09/08/getting-information-from-an-emv-chip-card/, 2006. [Accessed 04 Jun 2016].

[13] Channel 4. Millions of Barclays card users exposed to fraud. [online] Available at: http://www.channel4.com/news/millions-of-barclays-card-users-exposed-to-fraud, 2012. [Accessed 04 Jun 2016].

[14] M Roland and J Langer. Cloning Credit Cards: A combined pre-play and downgrade attack on EMV Contactless. Technical report, NFC Research Lab Hagenberg University of Applied Sciences Upper Austria, 2013.

[15] EMVCo. *EMV Contactless Specifications for Payment Systems: Kernel 2 Specification*. EMVCo, version 2.6 edition, 2016.

[16] J. H. Conway. *On Numbers and Games*. Academic Press, 1976.

[17] Desmedt Y, Goutier C, and Bengio S. Special Uses and Abuses of the Fiat-Shamir Passport Protocol. Advances in Cryptology. In *Advances in Cryptology CRYPTO 87*, pages 21–39. Springer-Verlag, 1988.

[18] Drimer S and Murdoch SJ. Keep Your Enemies Close: Distance Bounding Against Smartcard Relay Attacks. Technical report, Computer Laboratory, University of Cambridge, 2007.

[19] Francis L, Hancke G, Mayes K, and Markantonakis K. Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones. Technical report, Royal Holloway, Information Security Group, Smart Card Centre, 2012.

[20] info security. RFID credit cards are more secure than magnetic strip cards, says ITRC. [online] Available at: http://www.infosecurity-magazine.com/news/rfid-credit-cards-are-more-secure-than-magnetic/, 2011. [Accessed 06 Jun 2016].

[21] Roberti M. Are RFID-Enabled Credit Cards Safer Than Magstripe Cards? [online] Available at: http://www.rfidjournal.com/blogs/rfid-journal/entry?7870, 2010. [Accessed 06 Jun 2016].

[22] UK CARDS ASSOCIATION. One in 10 card transactions now contactless: Uk card expenditure Statistics: October 2015. [online] Available at: http://www.theukcardsassociation.org.uk/news/CESOct2015news.asp, 2015. [Accessed 30 May 2016].

[23] UK CARDS ASSOCIATION. Monthly contactless spending reaches 1bn : UK Card Expenditure Statistics: November 2015. [online] Available at: http://www.theukcardsassociation.org.uk/news/CardExpenditureStatisticsNovember2015.asp, 2016. [Accessed 30 May 2016].

[24] Hancke G, Mayes K, and Markantonakis K. Confidence in Smart Token Proximity: Relay Attacks Revisited. Technical report, Royal Holloway, Information Security Group, Smart Card Centre, 2008.

[25] I Kar. Here's How the Security Behind Apple Pay Will Really Work. [online] Available at: http://bankinnovation.net/2014/09/heres-how-the-security-behind-apple-pay-will-really-work/, 2014. [Accessed 08 Jun 2016].

[26] Y Heisler. Apple Pay: An in-depth look at what's behind the secure payment system. [online] Available at: https://www.engadget.com, 2014. [Accessed 09 Jun 2016].

[27] J Leyden. Apple's new iPhone 6 vulnerable to last year's TouchID fingerprint hack. [online] Available at: http://www.theregister.co.uk/2014/09/23/iphone_6_still_vulnerable_to_touchid_fingerprint_hack/, 2014. [Accessed 12 Jun 2016].

[28] Google. Spot.Me. [online] Available at: https://play.google.com/store/apps/details?id=me.spot&hl=en_GB, 2016. [Accessed 01 Mar 2016].

[29] M Clark. SpotMe app lets Android users spend Apple Pay Cash. [online] Available at: http://www.nfcworld.com/2015/03/03/334455/spotme-app-lets-android-users-spend-apple-pay-cash/, 2015. [Accessed 06 Jul 2016].

[30] Apple. About Touch ID security on iPhone and iPad.

[online] Available at: https://support.apple.com/en-gb/HT204587. [Accessed 16 January 2017].

[31] Rasmussen K.B and Capkun S. Realization of RF Distance Bounding. Technical report, ETH Zurich, Department of Computer Science.

[32] Brands S and Chaum D. Distance-Bounding Protocols. In *Advances in Cryptology EUROCRYPTO 93*, pages 344–359. Springer-Verlag, 1994.

[33] Toegl R. Tagging the Turtle: Local Attestation for Kiosk Computing. In *Advances in Information Security and Assurance*, pages 60–69. Springer, 1988.

[34] Roland M. *Security Issues in Mobile NFC Devices*. Springer International Publishing, 2015.

[35] Apple. Compare iPhone models. [online] Available at: http://www.apple.com/uk/iphone/compare/. [Accessed 16 January 2017].

[36] R.N Akram, I Gurulian, C Shepherd, K Markantonakis, and K Mayes. Empirical Evaluation of Ambient Sensors as Proximity Detection Mechanism for Mobile Payments. Technical report, Royal Holloway, Information Security Group, 2016.

[37] ifixit. iPhone 6s Teardown. [online] Available at: https://www.ifixit.com/Teardown/iPhone+6s+ Teardown/48170. [Accessed 16 January 2017].

[38] T Halevi, D Ma, N Saxena, and T Xiang. Secure Proximity Detection for NFC Devices Based on Ambient Sensor Data. In *Computer Security ESORICS 2012: 17TH European Symposium on Research in Computer Security, Pisa, Italy, September 10-12*, pages 379–396. Springer Berlin Heidelberg, 2012.

[39] A Varshavsky, A Scannell, A LaMarca, and E De Lara. Amigo: proximity-based authentication of mobile devices. In *UbiComp '07 Proceedings of the 9th international conference on Ubiquitous computing*, pages 253–270. Springer-Verlag Berlin, 2007.

[40] M Mehrnezhad, F Hao, and S. F Shahandashti. Tap-Tap and pay (TTP): Preventing the Mafia Attack in NFC Payment. Technical report, School of Computing Science, Newcastle University, 2015.

[41] N Saxena, T Xiang, and Y Zhu. Location-Aware and Safer Cards: Enhancing RFID Security and Privacy via Location Sensing. In *IEEE Transactions on Dependable and Secure Computing (Volume:10 , Issue: 2)*, pages 27–69. IEEE, 2013.

[42] INNOVENTIONS. Sensor Kinetics Pro. [online] Available at: https://itunes.apple.com/gb/app/ sensor-kinetics-pro/id623633248?mt=8, 2013. [Accessed 15 Jun 2016].

[43] theiphonewiki. /System/Library/Frameworks. [online] Available at: https://www.theiphonewiki.com/wiki/ /System/Library/Frameworks, 2015. [Accessed 22 Jun 2016].

[44] Apple. Event Handling Guide for iOS. [online] Available at: https://developer.apple.com/library/ content/documentation/EventHandling/Conceptual/ EventHandlingiPhoneOS/Introduction/Introduction. html, 2016. [Accessed 22 Jun 2016].

[45] C Tadlock. Will Apple support NFC tags in iOS for the iPhone 7/. [online] Available at: https://gototags.com/blog/ will-apple-finally-support-nfc-tag-reading-ios-10-iphone-7/, 2016. [Accessed 07 Jul 2016].

[46] Seriot N. iOS-Runtime-Headers. [online] Available at: https://github.com/nst/iOS-Runtime-Headers/tree/ master/PrivateFrameworks, 2016. [Accessed 15 Jun 2016].

[47] Soto J. iOS9-Runtime-Headers. [online] Available at: https://github.com/JaviSoto/iOS9-Runtime-Headers/ tree/master/PrivateFrameworks, 2015. [Accessed 15 Jun 2016].

[48] b2cloud. Obtaining Luminosity from an iOS Camera. [online] Available at: http://www.b2cloud.com.au/ tutorial/obtaining-luminosity-from-an-ios-camera/, 2011. [Accessed 22 Jun 2016].

[49] iphonedevwiki. Graphicsservices.framework. [online] Available at: http://iphonedevwiki.net/index.php/ GraphicsServices.framework, 2010. [Accessed 22 Jun 2016].

[50] Apple. UIDevice Class Reference. [online] Available at: https://developer.apple.com/reference/uikit/uidevice, 2016. [Accessed 22 Jun 2016].

[51] Apple. Xcode 7 Release Notes. [online] Available at: https://developer.apple.com/library/content/ documentation/Xcode/Conceptual/ RN-Xcode-Archive/Chapters/xc7_release_notes.html, 2016. [Accessed 23 Jun 2016].

[52] Apple. API Reference. [online] Available at: https://developer.apple.com/reference/. [Accessed 22 February 2017].

[53] G Slabaugh. Computing Euler angles from a rotation matrix. Technical report, City University of London, 2013.

[54] M Baker. Maths - Conversion Quaternion to Euler. [online] Available at: http://www.euclideanspace.com/maths/geometry/ rotations/conversions/quaternionToEuler/, 2016. [Accessed 16 Jul 2016].

[55] M Harris. MIT turns Wi-Fi Into Indoor GPS. [online] Available at: http://spectrum.ieee.org/tech-talk/ telecom/wireless/mit-turns-wifi-into-indoor-gps, 2016. [Accessed 01 Aug 2016].