Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Fraud Prevention and Detection in Mobile Payments

Master Thesis

CONFIDENTIAL

Cedric Waldburger

wcedric@ee.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zurich

Supervisors:
Christian Decker
Conor Wogan (SumUp)
Prof. Dr. Roger Wattenhofer

April 15, 2013

Acknowledgements

I would like to thank my advisors Prof. Roger Wattenhofer and Christian Decker from the Institute for Distributed Computing at ETH Zurich for their help and support. I also wish to thank Stefan Jeschonnek, Conor Wogan, Matti Biskup and the whole Software Development Team at SumUp for their continuous support and help.

Abstract

Mobile payments have seen a lot of traction recently and a set of companies have emerged around the opportunity of payments processed from mobile devices. We've conducted this work in collaboration with a company that allows merchants to process credit card transactions on their smartphone by using a mobile application and hardware dongle.

The innovative technology has opened up many new beneficial opportunities for both, merchants and clients. It has also opened new opportunities for fraud in the area of credit card transactions. Being historically prone to fraud, the credit card industry is highly regulated. But as existing anti-fraud regulations are specified for the traditional credit card business, the advances in technology call for new anti-fraud measures.

We identify two separate but complementary methods against credit card fraud: **preventive**, in the form of an automatic, in-depth check of all users during the sign-up process against a database of individuals with high risk status, and **reactive** by providing mechanisms to verify every transaction's signature in real time with the card holder's previous signatures.

Our work has shown that the optimization of the check during sign-up is able to reduce the number of unnecessary matches by up to 90 percent which results in a reduction of 8 - 16 hours of the manual work done by the operations team each month. At the same time, we were able to make the system more reliable by reducing the number of false positives by about 50 percent.

We found that the characteristics of a signature captured by finger on a mobile device are much less stable than those of signatures captured with a pen. To account for the resulting false positives, we propose to use a feedback loop to ensure no transactions are lost. We propose a real time signature verification algorithm that fuses the score of a DTW algorithm, an HMM and a score base on global features.

Keywords: Fraud Detection, Signature Verification, Mobile Payments, Mobile Applications, Dynamic Time Warping, Hidden Markov Models

Contents

Chapter 1

Introduction

As smartphones become ubiquitous, they're being used in more and more industries and fields. An application that has seen a lot of traction recently is mobile payments. The usage of the smartphone in this field includes the utilization as an electronic wallet, a virtual bank account and as a payment processing terminal to accept credit cards and other payment methods. This work focuses on an application of the latter. The density of smartphones per capita and the computational power of current mobile phones have reached a level where it becomes practical to use phones electronic cash registers. iZettle[1], Square[2], SumUp[3] are just some of the companies which make use of this situation and are enabling merchants to receive payments via their smartphone. This thesis was written in collaboration with SumUp, a mobile payment company located in Berlin.

Financial businesses have a higher risk of being the target of fraud attacks than businesses in other industries due to the direct financial gain for fraudsters. On mobile devices, it becomes even more important to build automatic processes to protect the system from fraud as the mobile payments are usually a huge amount of small transactions which makes it unfeasible to monitor all transactions manually. While some security measures are imposed by financial authorities, those rules were made for traditional financial institutions. The new mobile, connected environment not only opens up a lot of beneficial opportunities but also more attack vectors for fraudsters. Therefore it is important to not only respect given guidelines but pro-actively develop new ways and adapt existing guidelines to the specifics of the mobile environment to prevent attacks.

We focus our work on two separate, complementary approaches to reduce the fraud risk for a mobile payment company like SumUp. We implement in-depth checks of new users during the sign-up process, which allows us to reduce the risk of signing up a high risk user. After screening a new user, suspicious applications are flagged for manual inspection of the information that caused the flagging.

[1] http://www.izettle.com
[2] http://www.squareup.com
[3] http://www.sumup.com

As this reduced the amount of accounts that have to be checked manually, we were able to reduce the amount of manual work per sign-up substantially.

The second approach focuses on reactive fraud detection and makes use of the most popular method to authorize credit card transactions in mobile payments: by signature. Authorization of credit card transactions can be done in various ways. They differ in terms of technical complexity and thus cost, speed at which a transaction can be processed and acceptance by card schemes. The most popular card schemes in Europe are MasterCard, VISA, American Express, Diner's Club and others. Widely used are the three following methods:

- Swipe And Signature (SAS): The card holder swipes his card through a card reader, the information on the magnetic stripe is read and the transaction is confirmed by signature on the mobile device. This approach has the lowest technical complexity of all three but is easier to attack as the data on the magnetic stripe is not encrypted. Main problem: cards can be copied while the card data is read. Widely deployed in the USA due to high acceptance by card schemes.
- Chip And Signature (CAS): The card holder inserts his card into a card reader and the encrypted information on the card's chip is used to validate the card against the card terminal. The user authorizes the transaction by signing on the smartphone screen. This is currently the most popular method to authorize payments in Europe on mobile devices. The biggest drawback of using CAS to authorize a transaction is that Visa Europe, even though the technical complexity for such readers is much higher, does not consider this method secure enough and therefore requires an extended flow that requires the client to confirm the transaction with a text message. Except from Visa Europe, all other card schemes consider CAS to be sufficiently secure.
- Chip And Pin (CAP): A card holder inserts the card into a card reader and enters his Personal Identification Number (PIN) into the reader's pin pad to authorize the transaction. Only after the correct PIN has been entered, the phone is able to read the data required to process the transaction. The main drawback of this authorization method is its high technical complexity, mainly because the reader must be equipped with a pin pad. This incurs higher costs on both, the hardware and software. Although this process allows to authorize transactions with any card scheme in Europe, the high cost make a widespread use in the short term unlikely.

Although authorization with CAS is a lengthy process with VISA cards in Europe as each transaction needs to be confirmed via text message, it is currently the best compromise between speed, cost and an almost universal acceptance by card schemes. SumUp provides merchants with SAS or CAS readers based on

the merchant's profile. This means that currently, all transactions are authorized by signature. Also with other providers of the same service, authorization by signature remains the predominant way to authorize transactions. It is therefore of paramount importance that the authorization by signature process is as secure as possible which motivates our research.

We propose an automatic signature verification system based on a set of global features, Dynamic Time Warping (DTW) and Hidden Markov Models (HMM) to make the authorization process more accurate. As the signatures, which are captured by finger, tend to have a high variability and unstable characteristics, our system has to account for false positive matches. To account for that, we propose a system involving a feedback loop. With a feedback loop, transactions are still possible even if a card holder's signature could not be matched to previous signatures while allowing the signature verification to be trained for future authorizations.

Going forward, we refer to the preventive part of our work as *fraud prevention* and the reactive part as *fraud detection*. We use the term *fraud protection* for findings that apply to both, fraud prevention and detection.

1.1 SumUp - Card Payments for Everyone

Credit cards enjoy a huge popularity among consumers which is illustrated by the number of credit card holders in the United States of America (USA): In 2008 over 176.8 million people owned a credit card with average of 3.5 cards per card holder. About 60 percent of consumers own a rewards credit card and approximately 51 percent of the USA's population owns at least two credit cards [1].

While credit cards are also very popular with consumers in Europe, far fewer businesses in Europe allow customers to pay by credit card than in the USA. This is mostly due to the following reasons:

- A European business initially pays between 200 EUR and 500 EUR for a credit card terminal
- There is a monthly subscription fee from 20 EUR to 50 EUR
- A percentage of each transaction goes to the credit card company. Usually, merchants pay between 2.75 and 5 percent to the card payment terminal provider. Often, a minimal fee is charged for small transactions.

It is likely that these costs are higher due to more extensive regulations which require more operational effort. Also, the CAS and CAP devices are more complex than SAS devices and the hardware costs therefore higher in Europe.

SumUp's goal is to lower the barrier for merchants to accept credit card payments by providing a professional yet inexpensive solution for anyone to accept card payments. The company was founded in the fall of 2011 and has enjoyed a rapid growth since. Today, SumUp's services are used by thousands of merchants in more than ten European countries. The business has been successful because it removed two out of three of the previously mentioned obstacles to a more widespread adoption: anyone who signs up with SumUp receives a free card reader for use with a smartphone and there is no monthly subscription fee. As the merchant already owns the expensive hardware in form of a smartphone and most people already pay for a data subscription, the cost on both ends is much lower. This way, SumUp is able to finance itself through the 2.75 percent transaction fee. At its core, it competes with traditional Credit Card Terminal companies who require their users to pay a monthly fee for their terminal and the expensive initial charge for the device.

Instead of building and selling expensive hardware, the only hardware required — the card reader — is shipped at no charge and the software is distributed for free via the Apple AppStore and Google Play Store. Payment through SumUp gives taxi drivers, market traders and other small stores, who couldn't afford one of the traditional payment terminals, an enormous economic

Figure 1.1: In-App Flow during a Transaction from step 1 to step 5 (from left to right)

advantage, in that mobile payments can now be processed right on the spot and without an initial financial investment.

Processing a transaction with SumUp is illustrated in five simple steps in Fig 1.1.

0. Once the merchant has registered a SumUp Account and provided identification documents, he receives a free card reader and can accept payments.
1. The purchase amount can be entered manually into the mobile application or via previously created products
2. Debit cards and credit cards like Visa and MasterCard are supported and can be chosen by clicking on the respective logo.
3. After reading the card, the mobile application shows a confirmation of the card type and number.
4. The customer confirms the transaction with a signature written onto the screen of the smartphone or tablet.
5. After successful completion, the customer can have the receipt sent to their email account or via SMS to their phone.

Depending on the authorization method, a number of requests are made towards the SumUp servers and from there to other components of the transaction authorization chain, including acquirers, issuing banks and credit card institutions. Internally, a request is sent to SumUp's fraud server which performs a variety of checks and analysis to decide whether or not the transaction is accepted or declined. There's currently no established third party provider for fraud detection in this area and all rules are custom made by each company.

C2C	Repay a friend
C2B	Buy Groceries
B2C	Pay for train to work with company account
B2B	Pay for Business Lunch

Figure 1.2: Examples of mobile payment applications as listed by the EPC

The signature is part of the data that is exchanged with the fraud server and the signature verification process as described in Chapter 3 is an integral part of the authorization process.

Along with simplicity, quick setup time and low cost, SumUp's advantage over its competitors is that it deducts a relatively low fee per transaction of 2.75%. From that fee, it also pays the other parties in the value chain. To build a sustainable business model, SumUp will enable customers to also process other types of mobile payments, alongside credit card transactions. The Consumer-To-Business credit card transactions will only be one part of all mobile payment transactions. A long term strategy is to implement all payment schemes as listed by the European Payments Council (EPC). Mobile payments may at one point replace all current payment methods for Business-to-Consumer (B2C) transactions as well as Business-to-Business (B2B), Consumer-to-Business (C2B) and Consumer-to-Consumer (C2C) transactions as listed with examples in Table 1.2. The EPC predicts this will happen due to the availability and convenience of mobile devices paired with the user's perception of having full control over it. This creates an environment of trust and convenience for conducting payments. [2]

1.2 Fraud

Fraud, as defined by Phua et al. [3], refers to the abuse of a profit organization's system without necessarily leading to direct legal consequences. Fraud detection, as part of the overall fraud control, has become one of the most established applications of data mining.

The large volume of transactions processed each day by SumUp make a manual verification of each transaction impossible. As such, SumUp has to rely on automated systems to process and validate transactions. At time of this research, there is no established provider of anti-fraud software in this field. However, companies like Sift Science[4], a company that specializes in providing a fraud control service as a third party, has recently raised more than four millions USD from leading venture capital firms.

At the time of this work, SumUp has internal research to implement, train and tweak its own fraud rules. An internal assessment has shown that the four most popular fraud scenarios are the following:

- Copied or stolen cards
- Money laundry
- Impersonators transferring money under someone else's name
- Illegal money being transferred through SumUp's system

We will mainly focus on the first fraud case and concentrate on identifying a card holder by signature. After a card has been used a certain amount of times, the goal is to build a reliable signature model to verify it's the same person signing the next time the card is used.

Not only the physical cards are at risk to be stolen, also the digital copy of the credit card data needs to be protected. This is one of the reasons SumUp only saves encrypted card information and is obliged to do so according to the Payment Card Industry Data Security Standard (PCI DSS) and in an environment defined by the standard. The standard defines a set of rules to reach these control objectives:

- Build and maintain a secure network
- Protect card holder data
- Maintain a vulnerability management program
- Implement strong access control measures

[4] http://siftscience.com

- Regularily monitor and test networks
- Maintain an information security policy

This has an impact on our work as the regulations connected to protecting card holder data only allows storing card numbers and not names of card holders. The goal of this requirement is that even if someone would gain access to SumUp's database, it would not be possible to retrieve all information needed to process a transaction. This has one disadvantage though: we cannot collude information of multiple cards used by a single customer and can therefore only build a signature model based on the signatures we collect per card, not per customer. As the majority of all US card holders holds at least two credit cards [1], being able to merge the signature databases from different cards of one card holder, could significantly increase the accurateness of the signature model.

It is impossible to be absolutely certain about the legitimacy of a transaction. In reality, the goal must be to filter out possible evidences of fraud from the available data using cost-effective algorithms in real time.

1.3 Signature Verification

Most previous work has been done on signature verification with signatures captured with a pen on paper or on a digital tablet. When a signature was captured on paper, it was afterwards digitalized using a scanner. The dynamic characteristics of a signature are lost in that process and the algorithms that can be applied to the signature information are very different. Therefore it is common to divide signature verification methods into two methods:

- **Offline signature verification** performs recognition algorithms based on static features of a signature, mainly shape and length.
- **Online signature verification** performs algorithms on the dynamic features of a signature as well. These analyze the speed, the acceleration, the angular acceleration, the pressure and many other local properties of a signature.

Since digital tablets, touchscreens interfaces and smartphones became affordable and widely deployed, it became feasible to capture dynamic features of a signature. In Chapter 3 we look at the common techniques in both areas, offline and online, to verify signatures.

The high accuracy and resolution of current digitizing tablets and smartphone screens enables to capture signature data in high resolution and precision on relatively cheap devices. Traditionally, digital signatures were captured on a digital tablet with a pen. Signatures captured by SumUp however, are collected on a variety of mobile devices and have fundamentally different characteristics than signatures captured on a digital tables. The main reasons for these differences are:

- The signatures are captured by finger instead of a pen which is not how people are accustomed to sign. This leads to higher variability and less stable signature models.
- Most smartphones don't have a fixed sampling rate but an event-based sampling. Whereas the sampling rate on digital tablets is constant, the rate in signatures captured on smartphones can vary a lot within one signature.
- The signatures are captured on a lot of different devices with different screen sizes, resolutions, sensor densities and other device specific characteristics.

We have to verify signatures within a fraction of a second to give instant feedback whether or not a transaction will be authorized. This limits not only the

number of algorithms we can use in parallel but also the complexity of our models and our dataset. We discuss our strategies to overcome the listed difficulties in Chapter 3.

Chapter 2

Fraud Prevention PEP & HRA Detection

The fraud prevention is applied during the sign-up process of a new user and with the goal to keep high risk users from gaining access to the system. Any financial institution has a high interest to implement as many anti-fraud mechanisms as possible. With the financial industry being a strictly regulated industry, many precautions are mandatory and described by the governing financial authority. Yet, more mechanisms are added on top of the mandated ones as they provide a competitive advantage.

The financial authority governing SumUp's operations is the United Kingdom (UK) Financial Services Authority (FSA) and the regulatory environment is specified in the Money Laundry Regulations (MLR) from 2007 [4]. Chapter 2 of the MLR specifies the due diligence that has to be done for every client. As part of the due diligence process, the financial institutions are required to check that the customer does not fall into one of the two following categories:

- **Politically Exposed Persons** (PEP) are people who hold a prominent political function. In the United Kingdom this is defined as a national position. Also a PEP's spouse and children are included in this category.
- **High Risk Accounts** (HRA) are people who have previously committed a financial crime, been involved in a money-laundering related crime (e.g. dealing with narcotics) or are listed on a government watch list.

We will refer to the combination of PEP and HRA as "high risk clients". Any new client needs to be checked against both lists at sign-up and regularily after the initial sign-up. If the algorithm returns a positive match with either a PEP or an HRA, a flag is raised in the SumUp Operations Admin Panel (SOAP) and the match has to be confirmed or falsified manually. If the match can be falsified, nothing more needs to be done and the user will continue the sign-up

flow. However, if there is a positive PEP or HRA match, different procedures have to be followed.

For an identified PEP match, additional due diligence needs to be done to onboard the client. Additional documents need to be collected that give detail information about the wealth and assets of said person. After performing the additional due diligence, a client can be onboarded but settlements need to be blocked. A positive match with the HRA list means that the subject cannot be onboarded and a Suspicious Act Report (SAR) needs to be filed with the Serious Organized Crime Agency (SOCA). All HRAs are banned from conducting business with any financial institution.

Previous to our work, a test based on only first and last name was already in place but produced a lot of false positives. This required a lot of manual work to falsify matches. Our goal was to create a fast, reliable check requiring as little manual input as possible. Chapter 4 lists our results.

In the next section we'll present the data of the two databases and their origin before talking about the way we structured the lookup and implementation of our solution.

Field	Value
uid	unique identifier within the World-Check database
last_name	subject's primarily used last name
first_name	subject's primarily used firs first name
aliases	other first/last name combinations that the subject has used
alternative_spelling	alternative spelling of the subject's name
category	used to define if subject is a PEP or HRA
sub_category	used to define if subject is a PEP or HRA
age	age of the subject
dob	the subject's date of birth
deceased	information about the subject's death
locations	the cities the subject was associated with
countries	the countries the subject was associated with
further_information	description and references about the subject
external_sources	information about from which sources the information in the database was extracted

Table 2.1: Fields of the World-Check database CSV file

2.1 PEP & HRA Database

World-Check [5] is one of the providers of the PEP & HRA database. It consolidates the list by polling many different lists - including the FBI's Most Wanted LIist, Interpol and others.

The data is supplied in a text file as Comma Separated Values (CSV) and contains information in 26 fields per person. The most important fields are listed in table 2.1. Unfortunately, the records are far from complete. A lot of the records are missing some of the important fields like birthday or locations, which makes it hard to falsify a match automatically and sometimes a match falls back on just first and last name correlation. World-Check normalizes the data so that all dates have the same format, city and country names are always spelled the same name and data in one field is always represented the same way.

The full database file is about three gigabytes in size and contains some 1.8 million records of PEPs and HRAs combined. As the database is constantly changing, there's also a daily, weekly and monthly incremental update and delete file, with which the database is kept updated without having to download the full database file every so often. World-Check is a paid service and the data is retrieved over encrypted HTTPS connection.

2.2 Algorithm

Clearly, parsing three gigabytes of raw data and rebuilding the database each time a new user processes through the sign-up flow is slow and not practical. We therefore chose to parse the needed data into a relational database, allowing us to perform queries within split seconds. The algorithm we used to make the lookup faster, consists of two parts - parsing and lookup.

Parsing the data

The raw data contains only one record per individual but often lists not only one pair of first and last name but many. The additional pairs are retrieved from the aliases and alternative_spellings fields. For faster lookup, each record from the original CSV file is parsed into multiple records in the database such that there exists one database record for each pair of first and last name combination. We create a record for each permutation of first and last name retrieved from those fields to cover all identities a high risk user might possibly use. Listing A.1 shows the algorithm used to create the cross product of all name pairs.

Lookup

To improve the duration of a lookup, the database has an index on first and last name on the PEP/HRA table to speed up the lookup. An index was also generated on the other fields used by the lookup algorithm: Date Of Birth (DOB), deceased, locations (cities) and countries.

To reduce the number of false positives, the lookup uses additional information besides the first and last name to confirm or falsify a match. For each lookup, at least the following information needs to be provided to the algorithm:

- First and last name
- Date of birth
- City
- Country

The algorithm is outlined in Listing 2.1. It's output is a set of Know Your Customer (KYC) action and statuses which flag the customer in the system and may restrict him from certain actions, e.g. processing transactions with her account.

Listing 2.1: The Lookup algorithm

```
def lookup fname, lname, dob, city, country

    # collect all hits on first and last name
    all_hits = verified_hits = []
```

```
    all_hits << PEP.find_by_first_name_and_last_name(fname, lname)
    all_hits << HRA.find_by_first_name_and_last_name(fname, lname)

    for record in all_hits

        # skip if peson is already deceased
        next if record["deceased"]

        # parse dob. if the dob in the db is incomplete (eg
            1971/02/00)
        # only check the complete parts
        y = record["dob"].year  != 0 ? record["dob"]["year"]  :
            null
        m = record["dob"].month != 0 ? record["dob"]["month"] :
            null
        d = record["dob"].day   != 0 ? record["dob"]["day"]   :
            null

        # now skip if one of the values exists but doesn't match
        next if y && y != dob["year"]
        next if m && m != dob["month"]
        next if d && d != dob["day"]

        # at this point, dob matches or wasn't given
        # now, check city
        next unless record["locations"].contains? city

        # and country
        next unless record["countries"].contains? country

        # all checks passed, add this record to the result set

        verified_hits << record

    end

    # now set KYC status for the proven records
    for r in verified_hits
        set_kyc_status r
    end
end
```

2.3 Implementation & Architecture

The PEP & HAR check was implemented as a Sinatra[1] Ruby Applicaton, accessible through a REST interface. It fulfills multiple purposes: It parses the initial or incremental database file, it continuously rechecks all users in the database versus the updated database and also answers requests from the sign-up component.

It uses Ruby ActiveRecord (AR)[2] to connect to a relational database. The Ruby Sinatra Application runs within a CentOS[3] Linux environment and the linux Crontab Tool[4] is used to run the first two tasks in a constant interval:

- **daily:** Each day, the incremental update and delete file are downloaded from World-Check to keep the database up to date.
- **weekly:** On a weekly base, each user is screened against the updated database and new hits raise a flag in SOAP.

The third task, answering requests from the sign-up flow is done via a REST interface. Chapter 4 goes into greater detail of how the requests are structured. The system topology is shown in Figure 2.1.

A request is triggered and answered with the following steps:

1. A new user signs up from one of the mobile apps or the website
2. The sign-up component sends a request for the new user to the fraud detection component
3. The component retrieves the necessary data from the database
4. The database delivers all matches based on first and last name
5. The Lookup algorithm is performed within the fraud detection component
6. The result is returned to the sign-up component with information of whether or not the user can continue the sign-up flow
7. The respective KYC actions and statuses are set in the database.

TODO: Add performance improvement/performance data in Experiments chapter

[1] http://www.sinatrarb.com/
[2] http://guides.rubyonrails.org/active_record_querying.html
[3] http://www.centos.org/
[4] http://unixhelp.ed.ac.uk/CGI/man-cgi?crontab+5/

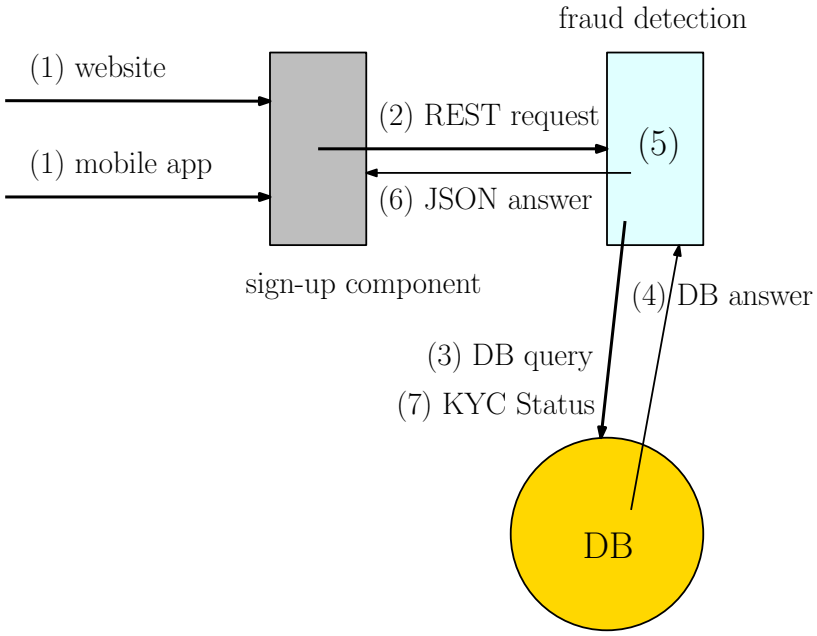

Figure 2.1: The simplified architecture and strucutre of a PEP/HRA check request

Chapter 3

Fraud Detection Signature Verification

For centuries, being able to authenticate someone based on biometrics, has been important to conduct business, identify each other and connect written information with an author. In order to guarantee someone's identity, it has always been important to work towards more exact, more reliable means to measure someone's biometric properties. The characteristic properties of a handwritten signature help to prove a signer's identity. A signature has four legal properties: [6]

- **Authentication**: Signature verification allows to confirm a signer's identification
- **Acceptance**: By signing a document, the writer conveys a willful intent and acceptance of the document's terms and contents
- **Integrity**: By signing a document, the signer establishes the integrity of the signed document and that it has not been altered
- **Non-repudiation**: The above three factors make it impossible for the signer to deny having signed the document

Signature verifications is a particularly important biometric identification process as the signature is a widely accepted method for endorsing financial transactions [7]. As the signature is recorded in the SAS and CAS authorization process, it makes sense for SumUp to use this information to improve the security of both authorization methods.

We present existing methods and related work before describing the signature verification methods used in our work and the peculiarities for signature detection in mobile payments. Traditionally, detection methods can be assigned to either feature- and function-based methods. We describe both approaches in Section

3.1 and Section 3.2. As a combination of feature- and function-based approaches has been providing better results than the individual techniques [8], we combine both approaches in our method to verify signatures.

3.1 Global Systems

Global systems, also called feature-based systems, are characterized by the fact that the feature vector consists of measurements that are based on the whole signature. Velocity, acceleration and position are usually looked at either combined or per dimension (usually x/y). Popular global features include:

- Signature length
- Total time to sign
- Maximum and average velocity
- Maximum and average acceleration
- Total dots recorded
- Number of segments
- Signature Heght (H), Width (W) and W to H-Ratio
- Number of points with positive x (y) velocity

Global features are derived from the signature as a whole. A lot of research has been done in this area and the features can be simple measurements as those listed in Table **??** or more complicated characteristics obtained through techniques like the discrete Wavelet Transform [9], the Hough Transform [10], horizontal and vertical projections [11] or smoothness features [12].

3.2 Local Systems

Function-based systems, also called local systems, are characterized by the fact that the feature vector consists of measurements on single points or groups of points of the signature.

The most popular methods are Dynamic Time Warping (DTW) and Hidden Markov Models (HMM).

- Pressure
- Horizontal (x) and vertical (y) position
- Path tangent angle
- Velocity and acceleration in a particular dot
- Log radius of curvature
- Pen elevation and pen azimuth (not available in our case as signatures are captured by finger)

Typically, a function vector has less or equal many elements as the signature vector. Matching two signatures means to find an algorithm to match

- two feature vectors
- or to match the vector of the test signature to a reference vector or model

Researchers have tried various techniques two match the two feature vectors. Among these techniques are: Dynamic Time Warping [13] [14], Hidden Markov Models [15], directional pdf [16], stroke extraction [17], synthetic discriminant functions [18], granulometric size distributions [19], neural classifiers [20], wavelets[21], grid features[22] and elastic matching[23] to name a few. [24]

Before trying one of these advanced techniques, one might try to compare two feature vectors in a simpler manner. The simplest approach of comparing two feature vectors element by element is to use linear correlation [25] and calculating the distance between each pair.

This approach has two major drawbacks:

- It only works if the two vectors have equal length. As signatures always vary a bit, it is rarely the case that two signatures and thus the two feature vectors have equal length.

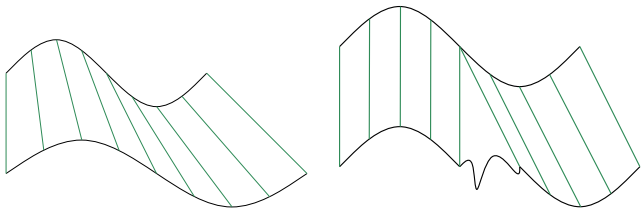

Figure 3.1: Two conceptual drawing of how DTW deals with the two main problems in signature recognition. The time series on the left are stretched but still matched thanks to DTW's ability to account for the scaling of segments or the signature as a whole. The drawing on the right is a representation of how DTW is still able to match two sequences even if one is distorted to a certain degree.

- Even if the overall path of the signature is very similar, very often there are parts that are distorted or additional in one of two signatures. Linear correlation isn't able to account for local distortions and would thus create a bad score even if the beginning and end of the signature's path were a perfect match.

As both phenomenon are very characteristics for signature verification, more advanced techniques, including those mentioned, are developed. Our work concentrates on Dynamic Time Warping (DTW) and Hidden Markov Models (HMM) as these two models have shown most success — especially combined — in recent work.

3.2.1 Dynamic Time Warping

Dynamic Time Warping (DTW) is a dynamic programming algorithm to measure the similarity between two time series which may vary in time or speed. This has been used for speech recognition and can also be used for signature detection to cope with the non-linear time distortions which one might see in the signals because a signer does not always sign with the same speed. It has shown to be a much more robust distance measure than the Euclidean distance [26], [27], [7] due to it's ability to match similar shapes even if they are out of phase in the time axis.

Koegh et al. [28] showed that the mean error rate average over 1000 runs for DTW was an order of magnitude lower than the error rate for the Euclidean distance. However, the DTW algorithm also took approximately 230 times longer to evaluate than the Euclidean distance. It has first been applied to signatures in 1977 by Yasuhara and Oka [29] who concluded that is a very useful approach for online real-time signature verification. Yasuhara and Oka used an adaption of the algorithm that was originally proposed by Sakoe and Chiba [30] and tuned the algorithm for the use on signature data.

As Figure 3.1 shows, DTW accounts for the two main difficulties when comparing two time series - stretching and distortion. This is of particular importance for our work, as we collect signatures on different devices with different sampling rates and even if a signer signs with the same speed but on two different devices, the two time series have very different length. Additionally, the different devices have different touchscreen sizes and surfaces and it is expected that people will not always complete the signature within the same speed due to different friction forces between the finger and touchscreen and length of the signature path due to a larger or smaller screen.

Classification is done by computing the distance DTW distance $dtw[s][t]$ between the model signature s and a test signature t which leads to a DTW score. If the score is below a certain threshold, we will consider the two signatures to match.

Training is done by computing the distance measure $dtw[n][m]$ for all signatures n, m in the set of signatures for a certain user and selecting the signature s with the smallest distance to all other signatures. While there are other techniques, including HMM, which allow to build and train a signature model based on all the reference signatures, DTW only allows to compare two time series at once. Our strategy is to run new test signatures against the signature that has the lowest average DTW score to each other signature in the training set.

Algorithm: We have two signature vectors X, Y containing data for each point of the signatures:

$$X = x_1, x_2, ..., x_i, ..., x_I$$
$$Y = y_1, y_2, ..., y_j, ..., y_J$$

And the distance between two vectors i, j defined as the 2-norm:

$$d(i, j) = ||x_i - y_i||$$

We define a warping path C as

$$C = c_1, c_2, ..., c_k, ..., c_K$$

where each element c_k correspondes to a combination (x_i, y_i).

The algorithm spans an $I \times J$ matrix G between the two signature vectors. The matrix is initialized with

$$g_1 = g(1, 1) = d(1, 1) * w(1)$$

where g_k is the accumulated distance after k steps and $w(k)$ is a weighting function that has to be defined. In each iteration, g_k is computed as

$$g_k = g(i, j) = \min_{c_{k-1}}[g_{k-1} + d(c_k) * w(k)]$$

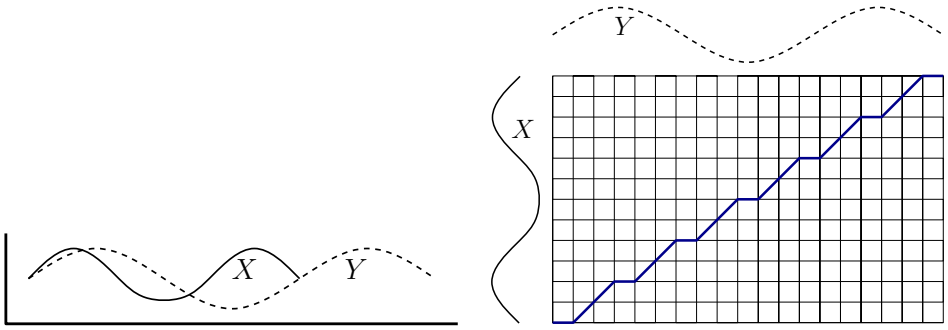

Figure 3.2: Put a good, extensive caption here

until both Signatures X, Y have been traversed.

The normalized distance of the two signatures is therefore:

$$D(X,Y) = \frac{g_K}{\sum_{k=1}^{K} w(k)}$$

where $\sum w(k)$ compensates the effect of the length of the sequences.

The definitionen of weighting factors w_k defines the matching between the two signatures. The most common definitionen in literature is one where three types of transitions - deletion, match and instertion - are allowed. The resulting g_k becomes:

$$g_k = g(i,j) = \min \begin{bmatrix} g(i,j-1) + d(i,j) \\ g(i-1,j-1) + 2d(i,j) \\ g(i-1,j) + d(i,j) \end{bmatrix}$$

The first case corresponds to the case of insertion, the second to a match and the third to an insertion. If in each step, there is a match, the path will go along the matrix' diagonal. Generally, the if the path is close to the diagonal, the signature's are similar and the score thus small.

Even though the DTW algorithm has been outperformed by more powerful algorithms like HMMs or SVMs in speech detection, it remains very effective in Signature detection as it deals well with small amounts of training data, which is typical for signature verification problems.

In general, DTW is known to have two drawbacks in signature verification:

- heavy computational load
- warping of forgeries

DTW causes heavy computational load because it does not obey the triangular inequality and thus indexing a set of signatures takes a lot of time. As soon as the pool of signatures for a signer get bigger, the computation costs raise because the test signature has to be compared to each of the signatures in the pool of confirmed signatures. Eamonn Keogh et al. [28] presented a lower bounding method to index all samples without comparing each of them to each other.

The second drawback can be addressed by looking at how straight or bended the warping path is. A straight warping path indicates that a genuine signature is more likely whereas a curvy warping path indicates a forgery. Work on this has been done by Y. Sato et al. [31] but made comparison between different signatures more difficult because it introduces another dimension and thus made computation harder and has hence not found wide spread use.

Hao Feng et al. [32] proposed another extension of the DTW algorithm, called extreme points warping (EPW) which proved to be more adaptive in the field of signature verification than DTW and reduced the computation time by a factor of 11. Instead of warping the signature as a whole, they only warp so called Extreme Points that are characteristic to a signer's signature and match the curves between those points linearly.

3.2.2 Hidden Markov models

A Hidden Markov Model (HMM) is a stochastic process with an underlying Markov Model. Unlike with Markov Models (MM), the states of a Hidden Markov Model can not directly be observed. Only the symbols emitted from Model's states may be observed. As each symbol may have been emitted by any state moving from state to state is hidden from the observer. If the signature is interpreted as a succession of observed symbols then we can, given an HMM, calculate the probability of each path through the states in the model. By training the HMM to math the signatures that are known to be genuine we can thus reconstruct a probability of the user matching.

An HMM as illustrated in Figure 3.3 is defined by:

- N hidden states $w_1, w_2, ..., w_N$
- the alphabet of M of symbols $v_1, v_2, ...$
- the $N \times N$ transition matrix A, which fulfills

$$\forall i, \qquad \sum_{j=1}^{N} a_{ij} = 1$$

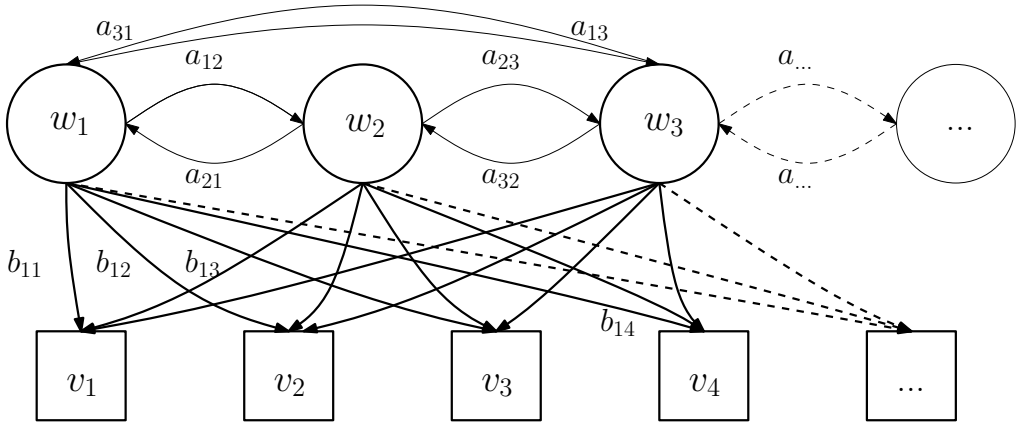

Figure 3.3: General representation of an HMM with states $w_1, w_2, w_3, ...$, observations $v_1, v_2, v_3, ...$, the state transition probabilities a_{ij} and the observation emission probabilities b_{ij}

and which elements are defined by

$$a_{ij} \geq 0, \qquad a_{ij} = P(w_j(t+1)|w_i(t)), \qquad 1 \leq i, j \leq N$$

- the $N \times M$ emission matrix B, which fulfills

$$\forall i, \qquad \sum_k b_j(k) = 1$$

and which elements are defined by

$$b_j(v(t)) = P(v(t)|w_j(t))$$

- the initial distribution π

The underlying Markov Model of an HMM is modeled according to the application. The most common types are shown in Figure 3.4:

- Left-To-Right HMM (LTR HMM) are characterized by the fact that from each state only the same state or more right state can be reached. Once a state is left, it is never reached again. Depending on the application, no skipping of states (top left), skipping a single step (top right) or skipping an arbitrary number of states may be allowed (mid left). Which states can be reached from which state is defined by the initial transition matrix. TODO: Reference to our transition matrix which is a non-skipping HMM
- Parallel HMMs have parallel paths, only the states belonging to the same path can be reached once a path is chosen. They have similar properties to LTR Markov Models with the exception that there are multiple paths.

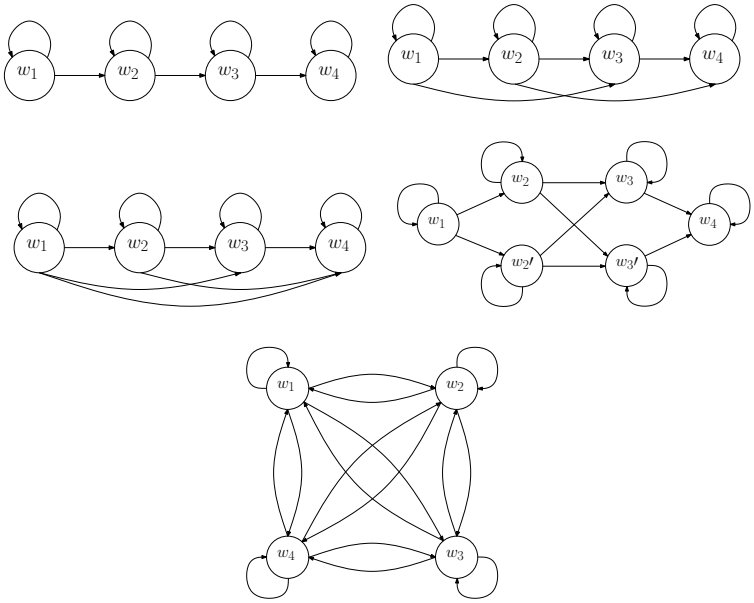

Figure 3.4: Different types of HMMs (from top left to bottom right): LTR HMM where skipping of states is not allowed, LRM with skipping of one step, LRM with allowed skipping, Parallel HMM, Ergodic HMM

- Ergodic HMMs are the most generic form of Markov Models where a transition is possible from any state to any other state or the same state. The transition matrix of an ergodic Markov Model has only non-zero elements.

In speech and signature recognition, a left-to-right Markov Model has proven to be a good choice. This can be interpreted as that a signer will never go back in his signature, which corresponds to the left-to-right analogy in the time series that corresponds to the signature.

While HMMs with a small set of states and observations perform bad because they are too simple, too many states and observations make the model computational heavy and accuracy is reduced because of overfitting.

The following three problems and their solution are what make HMMs so useful for real world applications:

1. Evaluation: Given the model parameters and observed data, estimate the optimal sequence of hidden states. This problem is solved by applying the Forward-Backward algorithm to the model.

2. Decoding: Given the model parameters and observed data, calculate the likelihood of the data. This is often solved with the Viterbi algorithm.

3. Training: Given just the observed data, estimate the model parameters. The Baum-Welch algorithm is used to estimate the model parameters.

We're interested in the Evaluation of an observed sequence to match a test signature to one of the signature models and we are interested in the training to train the signature models with the signatures we collect from a card holder. We follow previous authors who used the discretized angle between two points of a signature as the symbols. Based on those symbols, we train our model using the Baum-Welch algorithm. After training signature models, we calculate the likelihood of a signature belonging to a certain card holder by using the Forward-Backward algorithm to compare the signature to this card holder's signature model. A detailed description of all three problems and their solutions is given by Rabiner et al. in **??**.

In our case, we chose to work with an LTR HMM that doesn't allow to skip any states which leads to a transition matrix A of the form:

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & \dots & 0 \\ 0 & a_{22} & a_{23} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & a_{(N-1)(N-1)} & a_{(N-1)N} \\ 0 & 0 & 0 & 0 & a_{NN} \end{bmatrix}$$

We also chose to work with a LTR HMM that doesn't skip any states and we chose π such that the system starts in the first state:

$$\pi = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}$$

The symbols are calculated as the discretized angle between two points in a signature as illustrated in Figure 3.5. We use an alphabet of 4, 8 and 16 symbols in our experiment as described in TODO: ref to experiments section.

Evaluation: The Forward-Backward algorithm is a recursive solution. Before looking at the algorithm in detail, we discuss a direct approach to evaluate the probability. A first approach to evaluate which model has produced a certain set of observations might be the following:

1. For all possible observation sequences, calculate the probability of going from the first to the last state while emitting that sequence
2. These probabilities sum up to the total probability $P(V^T)$

$$P(V^T) = \sum_{r=1}^{r_{\max}} P(V^T|w_r^T)P(w_r^T)$$

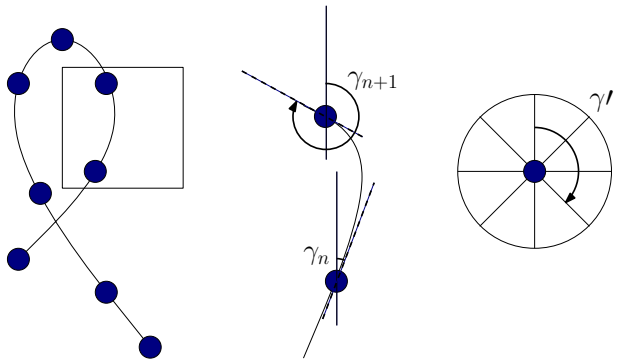

Figure 3.5: Our symbols are defined as the discretized tangular angle. The left image represents a close up look of a signature and the dots that define it. The drawing in the middle shows a zoomed version of the boxed segment on the right and the tangular angle for the two dots in this area. The right drawing shows the discretized $\gamma\prime$ for an alphabet of 8 symbols.

3. For all possible sequences in $w_r^T = w(1), w(2), ..., w(T)$ within T time intervals we get

$$P(V^T) = \sum_{r=1}^{r_{\max}} \prod_{t=1}^{T} P(v(t)|w_(t))P(w(t)|w(t-1))$$

The problem with this approach is that it is of complexity $\mathcal{O}(N^T T)$ and therefore unfeasible even for small values of N and T. E.g. for $N = 5$ States and $T = 100$ observations, there are on the order of $100 * 5^100 \approx 10^{72}$ computations.

Clearly, a more efficient way is required. Fortunately, there's The Forward-Backward algorithm is a recursive approach to approximate the solution.

1. We define the forward-backward variable which is the probability that a model is currently in state i and has already produced the first t elements of V^T

$$\alpha_i(t) = \begin{cases} \pi_i b_i(v(0)) & t = 0 \\ \sum_{j=1}^{N} \alpha_j(t-1) a_{ij} b_j(v(t)) & t \neq 0 \end{cases}$$

2. For $t = 0$ we initialize a

$$\alpha_i(0) = \pi_i b_i(v(0))$$

3. We compute α for all states i at all times t

4. The total probability that a model M generated the sequence V^T results from the sum of all $\alpha_i(T)$

$$P(V^T|M) = \sum_{i=1}^{N} \alpha_i(T)$$

This algorithm is only of complexity $\prime(N^2T)$ which results in our example to $5^2 * 100 = 2500$ computations. Compared to the 10^72, a saving of about 69 orders of magnitude.

In a similar way, we can consider a backward variable $\beta_i(t)$ defined as

$$\beta_i(t) = \begin{cases} 1 & t = T \\ \sum_{j=1}^{N} \beta_j(t+1) a_{ij} b_j(v(t+1)) & t \neq T \end{cases}$$

The backward variable β defines the probability of the partial observation sequence from $t+1$ to the end. In analogy to α, we get the same computational complexity to compute β.

Training: The Baum-Welch algorithm is used to train signature model. The problem to adjust the model parameters (A, B, π) to maximize the probability of the observed sequences is by by far the most difficult of the three. There is no analytical way to solve for the model which maximizes the probability. We can, however, find a local maximum with an iterative procedure such as the Baaum-Welch algorithm. The algorithm's output is the transition matrix A that is most likely to generate the given sequence of observations. The following needs to be known about the model M to start the algorithm:

- the number of hidden states N
- one or multiple observations sequences $V_1, V_2, ...$
- the initial transition Matrix A, that defines the HMM's structure
- the initial distribution π

In order to get to an iterative solution, we proceed as follows:

1. We first define the probability of being in state w_i at time $t-1$ and being in state w_j at time t as

$$\xi_{ij}(t) = P(w_i(t), w_j(t+1)|V^T, M) = \frac{\alpha_i(t) a_{ij} b_j(v(t+1)) \beta_j(t+1)}{P(V^T|M)}$$

$$= \frac{\alpha_i(t) a_{ij} b_j(v(t+1)) \beta_j(t+1)}{\sum_k = 1^N \sum_l = 1^N \alpha_k(t) a_{kl} b_l(v(t+1)) \beta_l(t+1)}$$

2. By summing over j we get the probability to be in state w_i at time t as

$$\gamma_i(t) = \sum_{j=1}^{N} \xi_{ij}(t)$$

3. With that, we can calculate the number of transitions from w_i to w_j as

$$\sum_{t=1}^{T-1} \xi_{ij}(t)$$

and the expected number of transitions to any other state as

$$\sum_{t=1}^{T-1} \gamma i(t)$$

4. And we get the expected number of times in one state from

$$\sum_{t=1}^{T} \gamma_i(t)$$

5. With that we can calculate:

- Expected Frequency (number of times) in state w_i at time $t = 0$ $\overline{\pi_i} = \gamma_i(0)$
- $\overline{a_{ij}} = \frac{\text{expected number of transitions from state } w_i \text{ to state } w_j}{\text{expected number of transitions from state } w_i} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$
- $\overline{b_j(k)} = \frac{\text{expected number of times in state } w_i \text{ and observing symbol } k}{\text{expected number of times in state } j} = \frac{\sum_{t=1}^{T} \gamma_i(t) b_i(k)}{\sum_{t=1}^{T} \gamma_i(t)}$

6. Through repeated execution of this algorithm, a local maximum will be found.

3.3 Challenges in Signature Detection Specific to Mobile Payments

A lot of work has been done on signature detection on offline and online signatures. However, almost all work on dynamic signatures has been done on signatures that were captured using a pen on a digital tablet. In our case, signatures were captured on a wide range of different devices and with a user's finger instead of a pen.

3.3.1 Signature are written with Finger

Our experience shows that people are not used to write their signature with their bare finger and the first few times they sign, their signature differs a lot. However, after just 10-15 times, the signature's shape stabilizes.

This means that it will be a lot harder to detect signatures based on the first few signatures than on later signatures, once a user got used to signing with her finger.

It also means that we should prefer later signatures to earlier ones as in later signatures the signer's signature might have stabilized.

3.3.2 Sparse Initial Dataset

Although SumUp is live in over 10 countries as of today, it is still only used in a relatively small set of locations and we are therefore unlikely to gather a lot of data about a certain user until the concept becomes more often deployed and used.

This means that we have to try and find a solution that works reasonably well with a small amount of initial data per user.

3.3.3 Device & Software Fragmentation

Unlike signatures captured on a digital tablet, our database of signatures is captured on a variety of devices with different properties. There are various factors that have an influence on the digital representation of the signature:

- Different Manufacturer: Both, iOS and Android devices use a variety of manufacturers for their handsets and the touch screens used. Recent studies have shown the differences of how signals are captured on different screens (TODO: link reference)

- Screen Size: the screen diagonal of current smartphones typically ranges between X and X cm, those of tablets typically ranges between X and X cm. A consequence is that the user might not only sign slightly differently but also that the signature will consist of more or less data points and it will take users a longer or shorter amount of time to sign.
- ...

We will consider these factors when applying our algorithm in Chapter X (TODO)

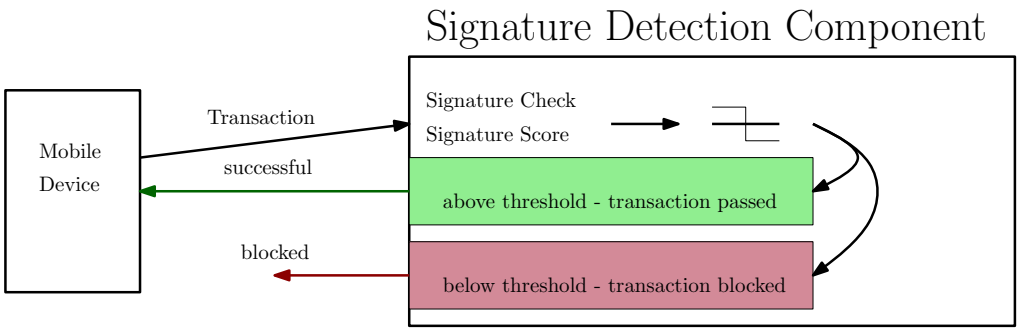

Figure 3.6: Simplified illustration of processing a transaction without the feedback loop. When the transaction object is received, the signature is extracted and analyzed. The score is compared with a threshold and if too low, the transaction is blocked. In case of a false positive match, this transaction is lost which we need to avoid.

3.4 Feedback Loop

Due to the mentioned challenges, our systems are likely not to perform as well as the same systems on traditional signatures captured by pen on a digital tablet. We are likely to get results with a higher Equal Error Rate (EER) which would hurt SumUp's business if we declined transactions because of false positive matches.

To eliminate the risk of blocking transactions because of a false positive match, we propose to use a feedback loop along with our signature verification system.

Traditionally, the system would work as shown in Figure TODO

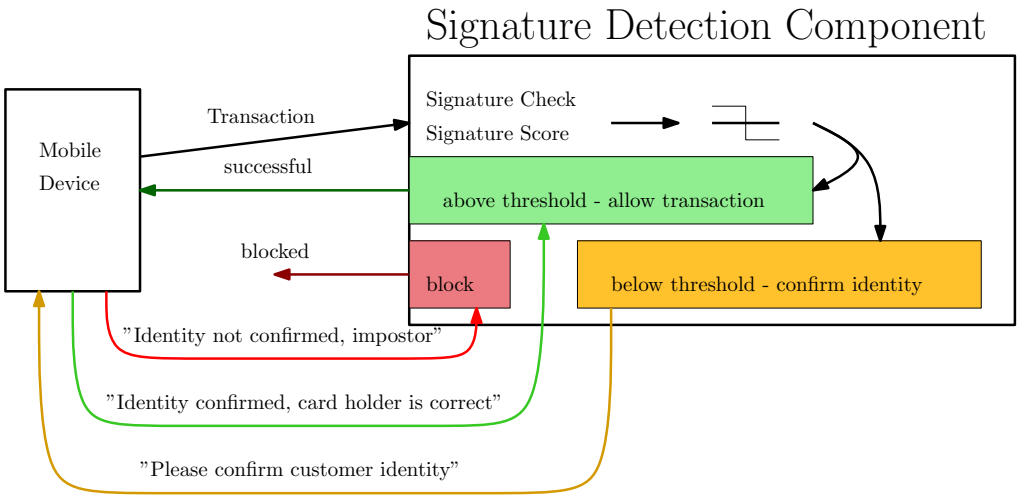

Figure 3.7: Simplified illustration of processing a transaction with the feedback loop. If the signature check does not reach the acceptance level, a request is sent to the merchant to confirm the card holder's signature. After a positive confirmation, the transaction is processed, otherwised blocked.

Chapter 4

Experiments

Section 4.1 will give insight into how our work has affected the number of false positives and thus the manual inputs from Operations have changed since the new component was released.

Section 4.2 will first introduce the database of signatures we collected to test our signature verification algorithms and afterwards show the equal error rate (EER) achieved in different scenarios.

Method	Path	Parameter
POST	/person/check	{"params": {"user_id": "1"}}
POST	/person/check	{"params": {"business_owner_id": "1"}}
POST	/person/check	{"params": {"business_officer_id": "1"}}

Table 4.1: The requests received by the PEP/HAR component

4.1 Preventive Fraud Detection

The algorithms described in Listing A.1 and Listing 2.1 were implemented in a Ruby Sinatra app with a REST interface and released to production about 2 months prior to the submission of this thesis.

4.1.1 Setup

The full database was parsed on a testing environment and the resulting data was copied over into production. The daily updates are received on the production machine and directly parsed into the production database through the parsing script.

The PEP/HRA check is performed either automatically or manually:

- **Automatically** for each new user during sign-up
- **Automatically** for each existing user once a week to re-check each account after incorporating the changes of that week
- **Manually** from the Operations Admin Panel (OAP) via a button

Each request must be triggered via a POST request. Depending on the user object that needs to be checked, the body of the request changes. Table **??** lists the different request types.

4.1.2 Results

The new component was released to production on March 4th, 2013, at beginning of calendar week 10. At the same time, the old component was discontinued. Every new user was automatically checked during the sign-up process. In addition to that, Business Officers and Owners were checked manually through SOAP by the Operations Team.

As SumUp is still growing rapidly, all numbers were normalized in respect to 1000 sign ups per week. The number of manual work could be greatly reduced.

week	Matches	False Positives	Ratio
13-15	3,29	1,39	42%
13-14	25,36	1,40	6%
13-13	1,77	0.98	56%
13-12	2,84	0,95	33%
13.11	2,20	2,42	110%
13-10	2,64	6,71	255%
13-09	64,33	58,91	92%
13-08	60,71	53,99	89%
13-07	55,89	55,89	100%
13-06	77,83	77,83	100%
13-05	65,10	36,86	57%
13-04	71,00	68,73	97%

Table 4.2: The requests received by the PEP/HAR component

Chart 4.1 shows how the number of matches reduced in calendar week 10 after the new component was released. Due to high amount of manual work that was required to reject or approve a match, the false positives were often only rejected one or two week after the initial match, which explains the shift between the curves in Figure 4.1.

Typically, about 50 to 100 matches were generated per 1000 new users with the old component. Almost all of those were false positives and therefore cause manual work that wouldn't have been necessary. With the new component, we were able to drastically reduce the amount of manual work required. After introduction of the new check to the system, the typical amount of matches per 1000 new sign ups after the release of the new component is below 10. The peak in calendar week 14 was caused by a system migration where a lot of accounts were checked again. However, as these accounts had been previously rejected, they did not cause any additional manual work.

It is too early to say whether the ratio of false positives per matches was also significantly reduced due to the backlog of false positives that were cleared within the first weeks after the new component's release. However, the numbers in Table **??** indicate a trend towards a lower amount of false positives per total amount of matches.

The operation department reports that the number of man hours was reduced since the release of the new component. It is still early but the numbers in Table **??** indicate that one to two full days of manual work were saved by introducing the new component.

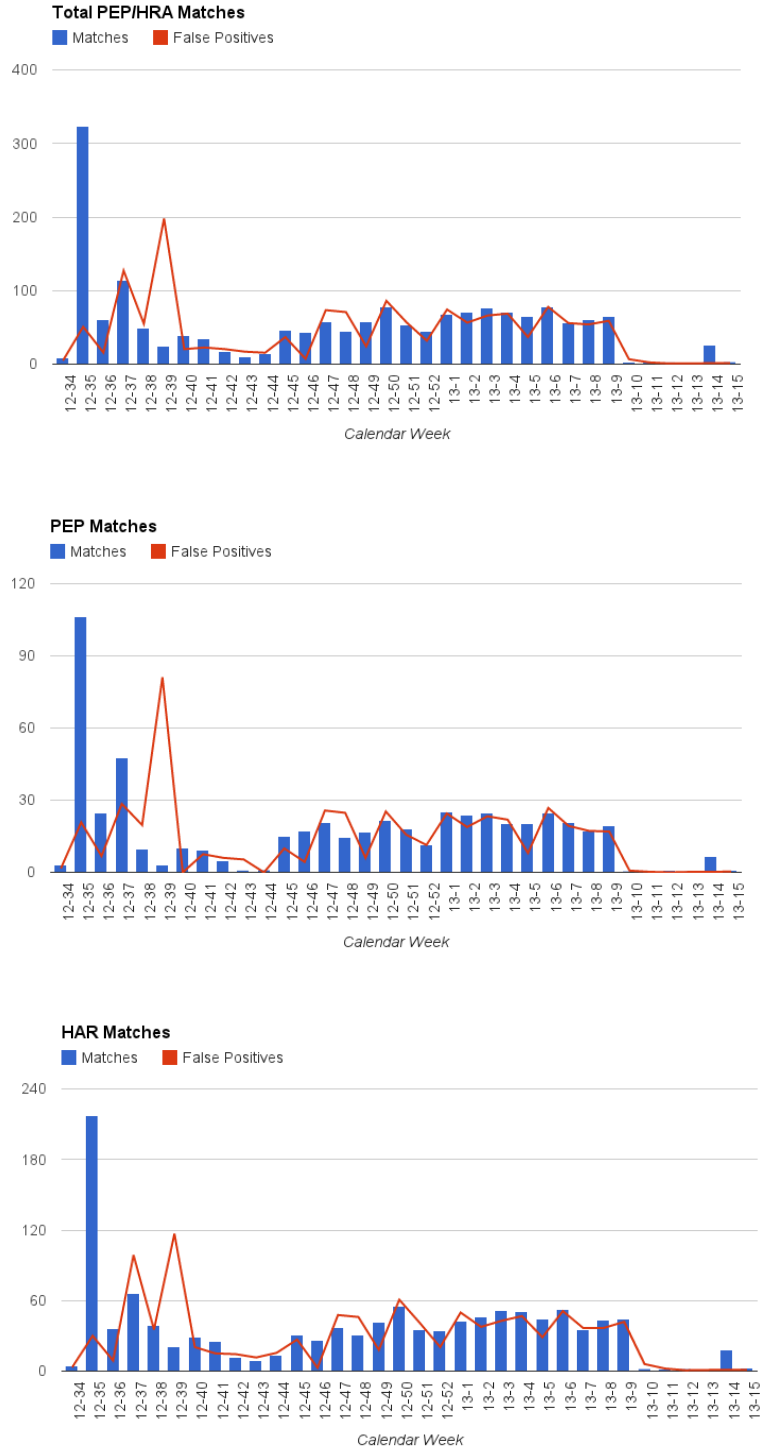

Figure 4.1: Evolution of the number of matches generated by the automatic PEP/HRA check. The top graph shows the evolution of the overall matches and false positives and the bottom and bottom graph the evolution for PEP and HRA separately. The new check was released to production in week 13-10 which is clearly visible as a drop in numbers.

Month	Matches (absolute numbers)	Man Hours
December	283	10 hours
January	463	16 hours
February	853	29 hours
March	125	4 hours
April (month to date)	9	0.3 hours

Table 4.3: The requests received by the PEP/HAR component

Device	Software Version	Screen Size [cm]	Screen Resolution [px]	Pixel Density [ppi]
Apple iPhone 4s	iOS 6.1.2	8.9	640x960	326
Apple iPhone 5	iOS 6.1.2	10	640x1136	326
Samsung Galaxy Note II	Android 4.1.1	14.1	720x1280	267
Apple iPad mini	iOS 6.1.2	20	768x1024	163

Table 4.4: The four devices used to collect signatures, ordered by screen size

4.2 Reactive Fraud Detection

4.2.1 Database

As there don't seem to be any publicly available databases of signatures that were collected with a finger on a mobile device, we collected our own database and forgeries.

Between 8 and 80 Signatures per person were collected from 11 people on 4 different days on four different devices. In total, a set of 487 signatures was collected.

The devices used to collect the signatures are listed in Table ??.

We also collected three forgeries for each person. The forgeries were created by an untrained person in the following modi.

- **Spontaneous Forgery**: A random signature from the set of one person's signatures was shown the imposter for 5 secs. The imposter was only allowed to forge the signature after the five seconds.
- **Sample Forgery**: A random signature from the set of one person's signatures was shown to the imposter for an indefinite amount of time and the imposter was able to look at the signature while forging the signature.
- **Knowledgable Forgery**: The imposter had unlimited access to all of one person's signatures to forge the signature.

Pre-Alignment and Normalization

4.2.2 Forgeries

4.2.3 Experiment Setup

Feature Extraction

Dynamic Time Warping

Hidden Markov Models

list different packages that are available and why we chose to work with matlab

Implementations: We ...

4.2.4 Computational Requirements

4.3 Evaluation

4.3.1 Modi

Chapter 5

Conclusions

5.1 Fraud Prevention

5.2 Fraud Detection

5.3 Outlook

- use lower bound proposed by Keogh et al.[28] to make DTW faster on large datasets

Bibliography

[1] Woolsey, B., Schulz, M.: Credit card statistics, industry facts, debt statistics. Google Search Engine (2010)

[2] Council, E.P.: White paper mobile payments. EPC492-09 (October 2012)

[3] Wang, S.: A comprehensive survey of data mining-based accounting-fraud detection research. In: Intelligent Computation Technology and Automation (ICICTA), 2010 International Conference on. Volume 1. (2010) 50–53

[4] Authority, F.S.: The money laundring regulations 2007 (2013)

[5] world check.com: Aml/kyc/pep/cft solution — world-check marketing portal (2013)

[6] Hanmandlu, M., Yusof, M., Madasu, V.K.: Off-line signature verification and forgery detection using fuzzy modeling. In: Pattern Recognition. (March 2005)

[7] Shafiei, M., Rabiee, H.: A new online signature verification algorithm using variable length segmentation and hidden markov models. In: Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on. (2003) 443–446 vol.1

[8] Fierrez-Aguilar, J., Nanni, L., Lopez-Peñalba, J., Ortega-Garcia, J., Maltoni, D.: An on-line signature verification system based on fusion of local and global information. In: Audio-and video-based biometric person authentication, Springer (2005) 523–532

[9] Ji, H.W., Quan, Z.H.: Signature verification using wavelet transform and support vector machine. In: Advances in Intelligent Computing. Springer (2005) 671–678

[10] Kaewkongka, T., Chamnongthai, K., Thipakorn, B.: Off-line signature recognition using parameterized hough transform. In: Signal Processing and Its Applications, 1999. ISSPA'99. Proceedings of the Fifth International Symposium on. Volume 1., IEEE (1999) 451–454

[11] Fang, B., Leung, C., Tang, Y., Tse, K., Kwok, P., Wong, Y.: Off-line signature verification by the tracking of feature and stroke positions. Pattern recognition **36**(1) (2003) 91–101

[12] Fang, B., Wang, Y., Leung, C., Tse, K., Tang, Y.Y., Kwok, P.C.K., Wong, Y.: Offline signature verification by the analysis of cursive strokes. International Journal of Pattern Recognition and Artificial Intelligence **15**(04) (2001) 659–673

[13] Herbst, B., Coetzer, H.: On an offline signature verification system. In: Proceedings of the 9th annual South African Workshop on Pattern Recognition. (1998) 600–604

[14] Yoshimural, M., Yoshimura, I. In: An application of the sequential dynamic programming matching method to off-line signature verification. Volume 1339 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (1997) 299–310

[15] Justino, E.J.R., Yacoubi, A.E., Bortolozzi, F., Sabourin, R.: An off-line signature verification system using hmm and graphometric features (2000)

[16] Drouhard, J., Sabourin, R., Godbout, M.: A neural network approach to off-line signature verification using directional PDF. Pattern Recognition **29**(3) (March 1996) 415–424

[17] Lau, K.K., Yuen, P., Tang, Y.Y.: Stroke extraction and stroke sequence estimation on signatures. In: Pattern Recognition, 2002. Proceedings. 16th International Conference on. Volume 3. (2002) 119–122 vol.3

[18] Wilkinson, T.S., Pender, D.A., Goodman, J.W.: Use of synthetic discriminant functions for handwritten-signature verification. Appl. Opt. **30**(23) (Aug 1991) 3345–3353

[19] Sabourin, R., Genest, G., Preteux, F.: Off-line signature verification by local granulometric size distributions. Pattern Analysis and Machine Intelligence, IEEE Transactions on **19**(9) (1997) 976–988

[20] Bajaj, R., Chaudhury, S.: Signature verification using multiple neural classifiers. Pattern Recognition **30**(1) (1997) 1 – 7

[21] Ramesh, V., Murty, M.N.: Off-line signature verification using genetically optimized weighted features. Pattern Recognition **32**(2) (1999) 217 – 233

[22] Qi, Y., Hunt, B.R.: Signature verification using global and grid features. Pattern Recognition **27**(12) (1994) 1621 – 1629

[23] de Bruyne, P., Forre, R.: Signature verification with elastic image matching. In: Proceedings - 1986 International Carnahan Conference on Security Technology: Electronic Crime Countermeasures., Chalmers Univ of Technology, Gothenburg, Swed; Univ of Kentucky, Coll of Engineering, Office of Engineering Continuing Education, Lexington, KY, USA; IEEE, New York, NY, USA; GTE Service Corp, Evaluation & Support Dep (August 1986) 113–118

[24] Piyush Shanker, A., Rajagopalan, A.N.: Off-line signature verification using dtw. Pattern Recogn. Lett. **28**(12) (September 2007) 1407–1414

[25] Plamondon, R., Lorette, G.: Automatic signature verification and writer identification - the state of the art. Pattern Recognition **22**(2) (1989) 107 – 131

[26] Keogh, E.J., Pazzani, M.J.: Scaling up dynamic time warping for datamining applications. In: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining. KDD '00, New York, NY, USA, ACM (2000) 285–289

[27] Fuentes, M., Garcia-Salicetti, S., Dorizzi, B.: On line signature verification: Fusion of a hidden markov model and a neural network via a support vector machine. In: Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on. (2002) 253–258

[28] Keogh, E.: Exact indexing of dynamic time warping. In: Proceedings of the 28th international conference on Very Large Data Bases. VLDB '02, VLDB Endowment (2002) 406–417

[29] Yasuhara, M., Oka, M.: Signature verification experiment based on nonlinear time alignment: a feasibility study. IEEE Trans. on Systems, Man and Cybernetics, part C **12**(3) (1977) 212–216

[30] Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. Acoustics, Speech and Signal Processing, IEEE Transactions on **26**(1) (1978) 43–49

[31] Sato, Y., Kogure, K.: Online signature verification based on shape motion and writing pressure. Proceedings of the 6th ICPR (1982) 823–826

[32] Feng, H., Wah, C.C.: Online signature verification using a new extreme points warping technique. Pattern Recogn. Lett. **24**(16) (December 2003) 2943–2951

Appendix A

Appendix Chapter

Listing A.1: The Algorithm that parses the csv file creates a separate DB record for each first/last name pair

```
for record in csv-file

    # Create Arrays to store all first & last names
    # and initialize with the most common first/last name
    first_names = []
    first_names << record["first_name"]

    last_names = []
    last_names << record["last_name"]

    # Get all other pairs from alternative_spellings ...
    for pair in record["alternative_spellings"]
        first_names << pair[0]
        last_names << pair[1]
    end

    # ... and from aliases
    for pair in record["aliases"]
        first_names << pair[0]
        last_names << pair[1]
    end

    # Now create the DB records
    for fname in first_names
        for lname in last_names
            if record["category"] == "PEP"
                PEP.create_in_db fname, lname, record
            else
                HRA.create_in_db fname, lname, record
            end
        end
    end
end
```