User Discrimination Through Structured Writing on PDAs

Rachel R. M. Roberts, Roy A. Maxion, Kevin S. Killourhy, and Fahd Arshad
{rroberts, maxion, ksk, fahd}@cs.cmu.edu

Dependable Systems Laboratory
Computer Science Department
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213 / USA

Abstract

This paper explores whether features of structured writing can serve to discriminate users of handheld devices such as Palm PDAs. Biometric authentication would obviate the need to remember a password or to keep it secret, requiring only that a user's manner of writing confirm his or her identity. Presumably, a user's dynamic and invisible writing style would be difficult for an imposter to imitate.

We show how handwritten, multi-character strings can serve as personalized, non-secret passwords. A prototype system employing support vector machine classifiers was built to discriminate 52 users in a closed-world scenario. On high-quality data, strings as short as four letters achieved a false-match rate of 0.04%, at a corresponding false non-match rate of 0.64%. Strings of at least 8 to 16 letters in length delivered perfect results—a 0% equal-error rate. Very similar results were obtained upon decreasing the data quality or upon increasing the data quantity.

1. Introduction

Passwords are standard in computer access control, but they can be forgotten, compromised without detection, and denied as having been used (repudiation) [14]. A proposed alternative is *biometrics*, or measurements of the human body. *Biological biometrics* are physical traits such as the iris, fingerprint, and face; *behavioral biometrics* are activities such as handwriting, keystrokes, and gait. Handwriting is an *alterable* biometric: it changes in response to varying conditions, e.g., text. Alterable biometrics are suitable for challenge-response protocols, whose dynamic nature resists forgery and replay, thereby providing stronger non-repudiation [14].

Personal digital assistants (PDAs) can potentially provide effective and low-cost biometric authentication. Some allow structured writing that may facilitate handwriting verification (because it enforces writing consistency), while also enabling automatic letter recognition. A PDA—or its input technology—could be integrated into a kiosk to capture and relay biometric data cheaply, while enjoying protection from theft and tampering.

To test whether structured handwriting on PDAs has promise as a biometric, we devised an evaluation of mild difficulty. Phillips et al. [16] recommend that evaluations be not too hard nor too easy; there are three stages of evaluation protocols: technology, scenario, and operational. We take a first step in examining the potential of PDAs to convey biometric-based security, by conducting a preliminary technology evaluation on a laboratory algorithm.

2. Problem and approach

We address whether it is possible to discriminate enrolled users on the basis of their handwriting characteristics in Graffiti [15], the original structured language of Palm handhelds. Specifically, we seek to build a system that can confirm or deny a claimed identity within a closed set of enrolled users; it is assumed that no outsiders can access the biometric system. Our example application is designed to catch insider attacks [21] and to provide traceability of human actions.

Our approach is to devise a non-secret challenge string, one per enrolled user, to distinguish that user from all the others. Challenge strings are pre-computed, based on errors and successes observed in preliminary testing on enrollment templates. In a hypothetical transaction, a user claims an identity and receives a personalized challenge string, which he or she writes on a Palm PDA. To confirm or deny the identity claim, the biometric system determines which enrolled user most likely wrote the sample, and reports whether or not the predicted identity matches the claimed one. We collect biometric data to build a corpus; data from the corpus is used to simulate user participation in the biometric system.

0-7695-2855-4/07 $25.00

3. Background and related work

To our knowledge, no previous research has attempted to differentiate users on the basis of their handwriting idiosyncrasies in a structured input language. Research on a related problem, discriminating users on the basis of their natural handwriting, is summarized in [11, 12, 17, 18, 20, 23]. This prior work can be categorized into signature verification tasks (determining whether a particular person wrote a signature) vs. writer identification tasks (determining which one of N known people wrote a document), and into off-line methods (writing as a static image) vs. on-line methods (writing as a dynamic process). The so-called off-line problems are more difficult than on-line ones, because timing, direction, pressure, and pen-orientation data are not available, and because recovering writing from a background document may be hard. Writer identification is more challenging than signature verification, due to difficulties in automation, character segmentation, and letter recognition. Off-line signature verification, the most typical application, usually achieves false-match and false non-match rates of a few percentage points each, although these may be optimistic due to the small size of signature databases [18]. The best team at the First International Signature Verification Competition achieved equal-error rates of 2.84% and 2.89%, respectively, on two tasks [25].

A wide variety of features is used to characterize natural handwriting. Features studied by manual examiners include the form of the writing as well as spelling and the type of pen used [3]. On-line signature verification may utilize functions of time. In off-line writer identification, features may be text-independent (using global statistical features) or text-dependent (using features computed on characters resolved from the image). Because Graffiti writing differs greatly from natural handwriting, the study of Graffiti letters motivated our features.

The original input language of Palm handhelds, Graffiti, is a stylized version of printed English (see Figure 1). The latest version, slightly modified, is called Graffiti 2. Graffiti is more constrained than natural handwriting; this makes letter recognition easier. The Palm PDA's screen digitizes information about pen pressure (a binary judgment, either up or down) and position (in Cartesian coordinates), in order to recognize letters.

In the framework of research on handwriting biometrics, our problem is a kind of writer verification task, using dynamic information, text-dependent features, and an automated decision process. Aspects of signature verification and writer identification apply to our work, although to a limited extent because of the differences between natural and constrained handwriting.

Most biometric systems perform either *positive identification* (verifying positive claims of enrollment), or *negative identification* (verifying claims of no enrollment) [24]. Our proposed system has characteristics of both, and additionally focuses on differentiating enrollees rather than distinguishing enrollees from outsiders. The insider-detection task we pursue assumes that information about all possible attackers is available.

Figure 1. Graffiti letters "A" to "G" [15]

4. Overview of the three experiments

The aim of this research is to test whether enrolled users can be discriminated on the basis of their handwriting characteristics in a constrained input language. To fulfill this goal, we recruited 52 subjects to write 1417 letters each, and we derived features from those letters to constitute a corpus. Next, twenty-six classifiers, one per alphabet letter, were trained using half of the data. A separate portion of data was set aside to test the classifiers; tests generated user- and letter-specific information about classification errors. For each user, this information was used to order and group letters into challenge strings, which were employed in simulated authentication transactions. A reserved portion of the data produced the transactions (genuine and unpracticed impostor) to examine how challenge-string length affects system accuracy.

The biometric system was trained and tested anew in three distinct experiments, each using a different version of the feature data, to explore the effects of data quality and quantity on results. (1) *High-quality Data* contains features from letters judged to be highly representative of user handwriting. The purpose of the High-quality Data experiment is to learn whether users can be discriminated on the basis of their handwriting alone (and not on their handwriting plus data-capture artifacts). (2) *Reduced Data* is of the same size and proportions as *High-quality Data*, but its features come from letters selected at random, instead of on the basis of quality. The purpose of the Reduced Data experiment is to gauge whether lower data quality might decrease accuracy, in comparison to the High-quality Data experiment. (3) *All Data* contains features from all valid letters we asked subjects to write. The purpose of the All Data experiment is to see whether an additional quantity of data might improve accuracy, in comparison to the Reduced Data experiment.

5. Data collection and preparation

Preparation of the three versions of the data (mentioned in Section 4) was identical, except where noted.

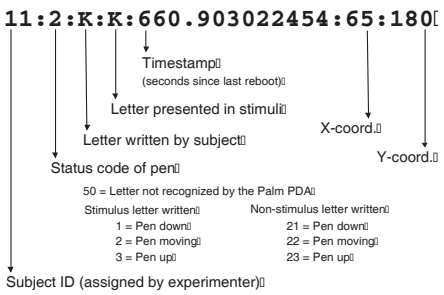

Figure 2. Example raw data for "K"

5.1. Instrumentation

Three Palm m105 handhelds, each running Palm OS 3.5 and having a maximum screen resolution of 160×160 pixels (with 2-bit color support, or 4 gray levels) were used to collect data. For each letter stroke drawn on the screen, a sequence of timestamped (x, y) coordinates, corresponding to the sampled positions of the stylus, was produced internally. For example, the first two consecutive points of a letter "I" might have $<timestamp, x, y>$ values of <93.993284, 54, 176> and <94.013007, 54, 182>. We captured the information using a program installed directly on the Palm PDA and written in the C language, with the help of the Palm Software Development Kit (SDK 3.5). The program noted whether each written letter matched the one expected according to the stimuli order (see Figure 2). The median interval between successive timestamps was 0.02 seconds; the median letter stroke took 0.38 seconds to write.

Approximate granularity of spatial coordinates can be gleaned from the following information. The dimensions of the Palm PDA's writing box are 1.60 cm (width) by 1.85 cm (height), although the entire screen (about 5 cm by 6.75 cm) is sensitive to input. In internal coordinate units, the range of horizontal coordinates in the data was 143; that for vertical coordinates was 218. Assuming the entire writing box received input at some point, we estimate an internal coordinate unit to be about 0.1 mm. The average letter width (in internal units) was 21.4, while the average letter height was 30.5.

5.2. Stimulus materials

Stimuli were chosen to ensure adequate instances of each letter, while discouraging subject boredom or frustration. Stimuli consisted of 5-letter nonsense strings and pangram sentences. Nonsense strings contained a bigram (two-letter combination) followed by a trigram (three-letter combination). Bigram and trigram motifs were visually separated by a space on the screen, which the subject did not write. Motifs were repeated and recombined to facilitate visual processing; two examples of nonsense strings are "BC ZAT" and "YZ CKS". A pangram includes every letter of the alphabet, e.g., "the five boxing wizards jump quickly". Ten nonsense strings were followed by one pangram sentence to form one set; there were 15 sets, resulting in 150 unique nonsense strings (750 letters) and 15 unique pangrams (667 letters). In total, each subject wrote 1417 requested letters. The letter type having the fewest instances in the stimuli had 44, while the one having the most instances had 94; the median number of instances of a given letter type was 52.

Stimuli were presented on the screen of the Palm PDA; subjects were asked to copy each sentence or string that appeared. A separate chart of the 26 Graffiti letters was displayed for reference, if needed. If a written letter was not recognized by the PDA, or if a letter did not match the one the user was expected to write, a beep sounded to prod the user to write it over again. The canonical Graffiti form for the letter "X" requires two strokes, but to simplify analysis we asked subjects to use an alternative single-stroke form (looking like a backwards α); compliance was confirmed. Other alternative strokes (all single-stroke forms) exist for a handful of other letters in Graffiti; these were allowed. For example, the alternative stroke for "Y" looks like a γ.

5.3. Subjects and sessions

Fifty-two subjects participated; each subject wrote in a single sitting lasting about 45 minutes. To increase task manageability, we did not introduce a time lag between enrollment and testing sessions (template ageing), although this is recommended [13]. Roughly half (25) of the subjects reported that they could write Graffiti letters without thinking about how to do it. Eleven others reported that they had learned Graffiti once before; only nine subjects claimed no prior exposure to Graffiti. Five subjects were left-handed and 47 were right-handed.

5.4. Data conditioning and version generation

User errors were excluded before analysis, i.e., when a subject drew a stroke not recognized as a letter, or wrote a letter other than the expected one. This seldom happened, because auditory feedback (upon errors) alerted subjects to pay closer attention. One might include such instances in a failure-to-acquire rate, but we reserve that distinction for letters excluded in the High-quality Data. The three versions of the data, one for each of the experiments described in Section 4, were prepared as follows.

High-quality Data. In the high-quality version of the corpus, data judged to be unrepresentative of actual user handwriting were excluded. Figure 3 shows four "Y"s, the leftmost pair written by one subject, the rightmost by another. Within each pair, the left letter not only looks unrealistic but also contains unrealistic timing information, due to an apparent instrumentation anomaly.

Data were passed through a filter to exclude letters whose first few sampled points contained timestamp intervals shorter than 0.01 seconds (half the typical time lag). The filter's results corresponded well with two human-rater judgments of which letters looked unrealistic, out of a sample of 1166 "Y"s. Although the filter removed 22.33% (16,455 out of 73,684) of the letters, sufficient samples remained for analysis. Little data (as little as 0%) was excluded for many subjects, while much data (as much as 72%) was excluded for some subjects; the median percent data loss was 20%. From the perspective of letters, the smallest percent data loss for a given letter was 8% while the greatest was 32%; the median was 22%. The smallest number of instances of a given letter in the High-quality Data, for any single user, was 13.

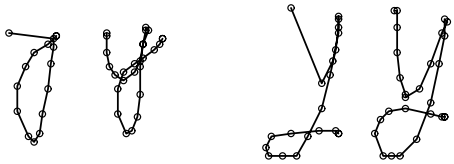

Figure 3. Unrealistic "Y"s (left within each pair); realistic "Y"s (right within each pair)

The data-capture irregularities do not appear to greatly harm letter recognition on the Palm PDA, since gross letter shapes remain. Anecdotally, one subject reflected that a carefully-written letter was occasionally not recognized. It may be that instrumentation errors occur sporadically, with some causing letter-recognition to fail, and others not. Newer or more sophisticated Palm handhelds might ameliorate this problem; future work should keep data quality in mind.

Reduced Data. This version of the data corpus was built to match exactly all proportions of the high-quality version (on a per-letter and per-subject basis). The only difference was that letter instances were included without regard to quality; they were selected at random.

All Data. This version used the data corpus in its entirety. The fewest instances of a given letter, for all users, was 44.

5.5. Feature extraction

Thirteen quantitative features were extracted from each letter stroke, which is represented natively in the Palm PDA as a sequence of $<timestamp, x, y>$ values. Features appear in Table 1 and are elaborated below.

Each feature attempts to capture a salient characteristic of Graffiti handwriting. *Time to Write* informs about the writer's speed. The four extreme coordinates (*Horizontal* and *Vertical Minimum* and *Maximum*) describe the location of the letter in the writing box, which varies among users by handedness or habit. *Horizontal* and *Vertical Start-to-End Distance (Signed)* help inform whether a canonical or alternative stroke is used, and describe an aspect of letter shape. *Horizontal* and *Vertical Distance Travelled* measure how much the stylus moves along a particular axis, whereas *Letter Length* measures the total amount of writing in a letter; all encode how simple or intricate a letter is. *Direction Changes* indicates how straight or shaky a stroke is. *Mean Slope* and *Standard Deviation of the Slope* represent a composite slant quality, and its consistency. Only finite pairwise slopes were included in the latter calculations. Nearly 2% of "I" strokes were perfectly vertical, which resulted in all their pairwise slopes being infinite. We imputed slope features in those cases by assuming that each "I" was not purely vertical, but rather perfectly slanted halfway between vertical and the smallest detectable positive slope.

Table 1. Features of Graffiti letter strokes

Time to Write	Elapsed time to write letter.
Horiz Minimum	Min. coord. value along x-axis.
Horiz Maximum	Max. coord. value along x-axis.
Vert Minimum	Min. coord. value along y-axis.
Vert Maximum	Max. coord. value along y-axis.
H.Start-End Dist (±)	$(x_{Last.coord.} - x_{First.coord.})$.
V.Start-End Dist (±)	$(y_{Last.coord.} - y_{First.coord.})$.
Horiz Dist Travelled	Sum of horizontal distances between successive points.
Vert Dist Travelled	Sum of vertical distances between successive points.
Letter Length	Sum of line-segment lengths between successive points.
Direction Changes	Count of changes between left↔right or up↔down motion.
Mean Slope	Avg. value of pairwise slopes.
Std Dev of the Slope	Std. dev. of pairwise slopes.

5.6. Feature transformation and scaling

Classifiers often perform better when data are normally distributed, or when a normalizing transform is applied to the data. We transformed the values of each individual feature (across all subjects at once) using the versatile Box-Cox power transformation [2], to effect greater symmetry on each feature's distribution. A program written in the statistical language R [19] semi-automatically searched for a good value of the Box-Cox parameter λ, one for each feature. The 13 transformed feature distributions were plotted to inspect their symmetry visually and to verify that an appropriate transformation had been found. After transformation, the feature data were scaled (within each feature, across all subjects at once), such that each feature mean became 0 and each feature standard deviation became 1. Data scaling is recommended before SVM classification [9].

5.7. Training, testing, and evaluation data sets

To train and evaluate the biometric system described in Section 6, three distinct sets of feature data were used: (1) *SVM-train*, to train SVM letter classifiers; (2) *SVM-test*, to test the classifiers and inform challenge-string creation; and (3) *Evaluate-test*, to evaluate the entire system. We split the feature data into these three sets according to a 50%/25%/25% rule, in the following way. Before splitting, we grouped the feature vectors by subject (making 52 groups), and then within each subject, by letter (making 26 groups for each subject). Next, for each subject, we randomized the order of feature vectors within each of the 26 letter groups. This made the data more closely resemble the writing of practiced users, thereby mitigating potential learning effects among novice subjects. After splitting, out of each subject's "A"s, SVM-train had 50%; SVM-test had 25%; and Evaluate-test had 25%; the same percentages held for other letters. During system evaluation (see Section 7), SVM-train and SVM-test are combined to form *Evaluate-train*, which contains 75% of each subject's letter "A"s, et cetera. Evaluate-train holds enrollment data and Evaluate-test holds evaluation data.

6. Biometric system construction

In our example application, an enrolled user approaches a biometric system and claims an identity. The system issues a challenge string tailored to that identity, and the user responds by writing the string in constrained handwriting, using a stylus on a digitized screen. Next, the biometric system decides which enrolled user most likely wrote the sample. Our system prototype incorporates 26 SVM letter classifiers, each trained on enrollment data for its respective letter. When a given letter is present in a writing sample, the corresponding classifier is invoked; decision logic is used to combine the outputs of the classifiers involved, to predict the writer's identity.

The following sections describe (1) how the SVM letter classifiers were built; (2) how the classifiers were tested to produce user- and letter-specific information about errors; (3) how potential challenge strings were devised using this error information; and (4) how decision logic yields writer predictions by the biometric system.

6.1. Letter classifiers

The purpose of a letter classifier is to determine the probability that each subject wrote a given letter, and to choose the most likely writer. We employed support vector machine (SVM) classifiers, because they achieve state-of-the-art performance on handwritten character recognition [6], a problem similar to the one we address.

About SVMs. Support vector machines [4, 22] use supervised learning for classification and regression; they are closely related to neural networks. SVM classifiers transform data into n-dimensional space ($\mathbb{R}^n$) such that an n-dimensional hyperplane can be found to optimally separate the data into classes. These classifiers maximize the margin (the margin is the distance between classes in n-dimensional space) as well as minimize empirical classification errors. SVMs are kernel-based methods; common options include linear, polynomial, radial basis function (RBF), and sigmoid kernels. RBF kernels are reasonable choices for studies in new domains, because they have fewer parameters, and because they avoid numerical difficulties that other kernels can encounter [9].

The two parameters in RBF kernels are called γ and *cost*; γ determines the width of the RBF, while *cost* determines the trade-off between reducing errors and creating a wider margin. Wider margins generalize better, so permitting more errors on training data may prove advantageous. RBF kernel parameters must be tuned before use, to find their ideal values on the data at hand.

Using SVMs to build letter classifiers. Individual letter classifiers, as well as the biometric system as a whole, were built with tools from the R statistical computing project [19], the LIBSVM library of tools for support vector machines [5], and the R package e1071 [7] that provides an interface between the R programming environment and LIBSVM. Under LIBSVM, multi-class classification employs the *one-against-one* approach [10]. Given k classes, one for each subject identity, $k(k-1)/2$ binary classifiers are constructed, one for each pair of classes; the appropriate class label is found through voting. We used unweighted SVMs, along with the option in LIBSVM to report probability estimates. The subject who is assigned the highest probability becomes the classifier's predicted writer.

Twenty-six SVM classifiers, one for each Graffiti letter, were trained using SVM-train data. Each classifier employed an RBF kernel, whose parameters were tuned as follows. Five-fold cross-validation [8] was used to select the best values of *cost* and γ for each letter classifier. Fifteen *cost* values and 13 γ values were tried in every pairwise combination (195 total). The search space formed a two-dimensional grid of powers of 2; exponents for *cost* ranged from -2 to 26 (in 15 steps of two), whereas those for γ ranged from -22 to 2 (in 13 steps of two). For each letter, average accuracy rates over the five folds were recorded for the 195 trained classifiers. The highest score determined the best parameter pair, which was then fixed for the relevant letter classifier. We verified that a local maximum occurred in the grid area, not on the boundary.

6.2. Confusion matrices and charts

To determine which letters are superior at distinguishing one subject from all the rest, classification errors must be studied. Accordingly, each letter classifier was re-trained on the entire SVM-train data set (using the best

Table 2. Letter-classifier accuracies, high-quality data (50% training, 25% test data)

Y	76.09%	**V**	60.79%	**U**	57.51%
B	72.11%	**Q**	59.88%	**O**	57.02%
D	70.12%	**Z**	59.70%	**C**	55.74%
P	69.45%	**K**	59.66%	**F**	55.26%
E	68.34%	**S**	59.43%	**J**	53.90%
M	67.12%	**X**	58.61%	**L**	52.02%
W	65.36%	**A**	58.13%	**T**	51.02%
R	63.26%	**N**	57.79%	**I**	36.95%
G	62.96%	**H**	57.53%		

parameter values), and then tested using SVM-test data, without cross-validation. The SVM-test data informs the challenge strings, while the Evaluate-test data estimates system accuracy; no data used in system creation was reused in its evaluation. In the High-quality Data experiment, accuracy rates for letter classifiers ranged from 37% to 76% (see Table 2). Performing worst was "I", simplest of all Graffiti letters, while "Y", rich in detail and having an alternative stroke, performed best.

Using the results of the tests, 26 confusion matrices were built, one per letter classifier, each having dimensions of 52 users by 52 users. A confusion matrix shows how often one user was (mis)classified as each of the others. Rows are associated with true writer identities, and columns with predicted writer identities. All scores in a row were normalized to sum to 1. For the "A" matrix, the cell in row 1, column 2 contains the fraction of "A"s written by User1, but erroneously predicted by the "A" classifier as having been written by User2.

Next, the rows of the 26 confusion matrices were rearranged to make 52 *confusion charts*, one for each subject. We extract User1's row from the "A" confusion matrix, to become row 1 of User1's chart; this operation is repeated on letters "B"–"Z", for successive rows. Charts for other subjects were constructed similarly. User1's confusion chart is a 26 letters by 52 users matrix in which the cell in row 1, column 2 shows the fraction of User1's "A"s incorrectly classified as belonging to User2. Of special note is the genuine subject's column, which contains *letter-accuracy scores*. These scores show how frequently the genuine writer was correctly identified as him/herself on a given letter. The rows of each confusion chart were sorted in decreasing order by the letter-accuracy scores; ties were broken at random. Now, row 1 of User1's chart shows the highest-scoring letter for that user, in terms of correct classifications; it might be tied in score with some other letter(s). Table 5 shows an abbreviated example confusion chart.

The higher a letter-accuracy score, the better this letter should be (in isolation) at distinguishing the genuine subject from all others in the group. Letters having a score of 1 are *perfect*; all others are *non-perfect*. In Table 5, only "E" is perfect. Two letters are *complementary* if they have no errors in common, with respect to the same other subject. Three or more letters are complementary if the group is pairwise complementary. In Table 5, "D" and "N" both have errors with respect to User3, so they are not complementary. In contrast, "D", "B", and "G" each have errors with respect to a different user, so they are all complementary with respect to each other.

6.3. Letter lists and challenge strings

A challenge string is a set of letters selected to discriminate one user from every other member of the group. Longer strings provide greater discrimination ability and security, while shorter strings are more convenient to write. A *letter list* is an ordered list of elements (called *units*) that are either single letters or complementary letter pairs. A letter list holds all possible lengths of challenge strings for a given user. A challenge string is realized by including letters from the first u units on a letter list. Increasing the number of units, u, should generally maintain—and may enhance—accuracy (and security).

An abbreviated example of a letter list appears in the bottom row of Table 3; it contains the letters "E DB G NR". Challenge strings are formed by including letters from the leftmost cells, up to a desired stopping point. "E" is the shortest possible challenge string; others are "EDB", "EDBG", and "EDBGNR".

Table 3. User1's letter list (example)

Perfect	**Run 1**		**Run 2**
Unit 1	Unit 2	Unit 3	Unit 4
E	D B	G	N R

To form successive units on a letter list, an algorithm selects a single letter or letter pair from the subject's confusion chart. Units are added without backtracking; letters are selected according to the heuristics in Table 4. The terms *perfect*, *non-perfect*, and *complementary* were defined in Section 6.2; a *run* is a group of complementary non-perfect letters. For convenience, we will call letters that have not yet been added to the letter list *unused*. Ties in letter-accuracy scores were decided randomly during chart sorting, so the topmost letter in the chart is favored during ties. We created one letter list per subject.

Example. A scenario involving four users and seven letters will be used to illustrate the process of creating a letter list. Table 5 shows a confusion chart for User1, whose identity is underlined. The "User1" column contains sorted letter-accuracy scores (shown in boldface).

The algorithm first looks for perfect letters, placing each within its own unit. In Table 5, "E" is perfect; it forms the first unit on User1's letter list (see Table 3).

After exhausting perfect letters, the algorithm looks for the highest-scoring non-perfect letter, which is "D". This letter has foreseeable deficiencies, namely its errors with respect to User3. Before adding "D" to the letter list, the algorithm seeks to pair it with a complementary letter (whose errors do not involve User3). The highest-scoring letter complementary to "D" is "B", whose errors involve User2. "D" and "B" are paired together into the same unit, which starts the first run of the letter list.

Next, the algorithm seeks to complete the current run, by successively adding single letters complementary to all those currently in the run. "G" (whose errors involve User4) is the only remaining unused letter complementary to both "D" and "B". "G" is added alone to a new unit in the current run, which is then terminated. No memory of errors is retained from one run to the next.

Now a new non-perfect pair is sought, starting with the highest-scoring unused letter, "N". The only other unused letter complementary to "N" is "R", so those two form a unit and start the second run on the list. The remaining unused letter, "H", is not complementary to "N" and "R"; thus, the second run is finished. "H" is left off the letter list, since it cannot be paired. The completed letter list appears in Table 3; possible challenge strings are "E", "EDB", "EDBG", and "EDBGNR".

The reason that non-perfect letters are only added in the context of a pair (or a run) is to prevent a succession of letters being added whose errors all involve the same user. Such an occurrence would raise the chances of the biometric system erroneously predicting the writer of a challenge string to be that user.

Procedure. A summary of the algorithm follows.

Table 4. Heuristics for creating letter lists

1. A given letter appears at most once on the letter list.
2. Perfect letters are considered equally useful, and superior to non-perfect letters. Perfect letters are added to the letter list before non-perfect letters.
3. A non-perfect letter is considered superior to another non-perfect letter, if its letter-accuracy score is higher. All other things being equal, a superior non-perfect letter is added before an inferior one.
4. A non-perfect letter can only be added to the letter list if a complementary letter accompanies it.
5. After a non-perfect letter pair is added to the list, other complementary letters are sought immediately to complete a run. The purpose of this is to further remedy deficiencies in the non-perfect letters.
6. Units are kept as short as possible, to provide granularity in security levels.

Table 5. User1's confusion chart (example)

Letter	User1	User2	User3	User4
E	**1**	0	0	0
D	**0.9**	0	0.1	0
N	**0.8**	0	0.2	0
B	**0.8**	0.2	0	0
R	**0.7**	0.1	0	0.2
G	**0.7**	0	0	0.3
H	**0.6**	0.1	0.2	0.1

ALGORITHM TO CREATE A LETTER LIST:
(1) Add perfect letters one at a time, each within its own unit, to the subject's letter list.
(2) Add non-perfect letters that fit into runs, one run at a time, until no more runs can be formed.

HOW TO MAKE A RUN:
(A) Find a complementary letter pair,[1] if one exists. If so, add both letters to a unit, the first unit of this run. If not, then no more runs can be formed.
(B) Add single complementary letters to the run, one at a time and each within its own unit, until none are left; then, the run is finished, and its units are added to the letter list. Each complementary letter must be complementary to all letters already in that run. If more than one letter choice exists in an iteration, prefer the one with the highest letter-accuracy score.

6.4. Decision logic to combine classifier outputs

When an SVM classifier is tested on a letter instance, it outputs a writer prediction. For a multi-letter challenge string, multiple classifiers are involved, so decision logic is required to combine the outputs into a single prediction. When a writing sample is submitted to the biometric system, appropriate SVM letter classifiers are invoked to process the relevant letters. A single classifier determines the probability that each of the 52 users wrote one letter instance. If a challenge string has s unique letters, then s classifiers are invoked, each producing 52 probability scores. Next, the s probability scores for User1 are multiplied together under the independence assumption, to produce a joint probability that User1 wrote all the letters. Joint probabilities are then found for User2, and for all the other users. The single user having the highest joint probability is deemed the most likely writer. Because the probabilities are assumed independent, the order of the letters does not affect the calculation; challenge strings could be permuted to thwart replay attacks.

[1] Choose a complementary pair in the following way. Take the highest-scoring unused letter as the provisional first member of the pair. For the second member, take the next-highest scoring letter complementary to the first. If none can be found, try a different provisional first member of the pair (having the next-highest score). Start the search again, and continue until a complementary pair is found, or until there are no more candidates to be the first provisional member of the pair.

7. Evaluation method

The evaluation aims to discover whether users can be discriminated on the basis of handwriting samples, and if so, to find out how long the samples must be. We varied the number of units appearing in challenge strings (recall Table 3), and observed system accuracy; extra units should not decrease accuracy (unlike solitary letters).

Three experiments were conducted, each on a different version of the data, to discover how data quality and quantity affect accuracy: (1) High-quality Data, (2) Reduced Data, and (3) All Data (see Sections 4 and 5.4). We performed an off-line evaluation, in which subject participation was simulated by data in the corpus. Individual letters written by a subject were assembled into a sequence meant to imitate how that subject would respond to a challenge string. The previously unseen Evaluate-test data was used for system evaluation.

7.1. Transaction design and error definitions

At transaction time, a user (the *claimant*) claims an identity; the system responds by displaying the challenge string tailored to that identity. After the user writes the string, the identity claim is accepted or rejected.

Genuine transactions. In genuine transactions, a subject claims his or her true identity. To simulate these events, we used the challenge string assigned to that subject, and called up instances of his or her letters from the Evaluate-test data set. The biometric system predicted the most likely writer. If the predicted writer identity matched the genuine one, no error occurred; otherwise, there was an error. Errors made on genuine transactions contributed to the false non-match rate (the rate at which genuine claimants are denied entry by the system).

Imposter transactions. In imposter transactions, a subject claims an identity other than his or her own. To simulate these events, we used the challenge string assigned to the victim, along with letter instances written by the impostor (from the Evaluate-test data set). The biometric system determined the most likely writer. If the predicted writer identity matched the claimed one, there was an error, because the imposter posed successfully as another subject. Otherwise there was no error, because the imposter failed. Errors made on imposter transactions contributed to the false-match rate (the rate at which imposter claimants are allowed entry by the system).

7.2. Challenge-string evaluation

The 26 SVM classifiers were newly re-trained on all previously seen data, namely SVM-train and SVM-test (collectively called Evaluate-train). All best parameter values from Section 6.1 were kept; 75% of the data was used for re-training. The remaining 25% of the data held in Evaluate-test, previously unseen, was used to evaluate challenge strings. Due to the numerous genuine and imposter transactions, some reuse of data within in Evaluate-test occurred, as specified below.

Within each of the three experiments (see Section 4), the number of units, u, was varied from one to the largest possible number (equaling the number of units in the shortest letter list). In a *round*, all challenge strings have a particular value of u. Between rounds, data were reused, without memory of prior usage.

Within a round, multiple repetitions of transactions were performed, and the error rates were averaged to increase the stability of results. We repeated each transaction as many times as unused data remained, but held constant the number of repetitions for all subjects. In the High-quality Data and Reduced Data experiments, the limiting number of instances per letter was 13; it was 44 in the All Data experiment. Since Evaluate-test contained 25% (or slightly less[2]) of each subject's letter instances, 3 instances of that letter appeared in Evaluate-test for the High-quality Data and Reduced Data experiments ($\lfloor 13/4 \rfloor$=3), whereas 11 appeared in Evaluate-test for the All Data experiment ($\lfloor 44/4 \rfloor$=11). In the respective experiments, repetitions were limited to those numbers, for all subjects. Within a round, the same letter instance was never re-used within genuine transactions, or within imposter transactions involving the same victim identity. Otherwise, instances were selected at random.

For the genuine transactions within a round, 52 challenge strings were evaluated using 3 (or 11) repetitions, reusing no letter instances. Errors were counted, and averaged over the repetitions to produce an average false non-match rate (FNMR). In the High-quality Data and Reduced Data experiments, 156 (or 52×3) genuine transactions were carried out per round, while the All Data experiment had 572 (or 52×11).

For the imposter transactions within a round, 52 subjects each masqueraded as 51 other users. Given a fixed victim, the letters from 51 different imposters were called up to satisfy the victim's challenge string; multiple repetitions (3 or 11) of this arrangement did not reuse letter instances. Errors were counted, and averaged over the repetitions to produce an average false-match rate (FMR). In the High-quality Data and Reduced Data experiments, 7,956 (or $52\times51\times3$) imposter transactions were performed per round, while the All Data experiment had 29,172 (or $52\times51\times11$).

In biometric applications providing positive or negative identification [24], special evaluation procedures should be followed for imposter transactions, whenever *dependent templates* are present [13]. Templates are dependent whenever the enrollment of a new user affects other templates in the system; otherwise, they are in-

[2] Any extra letters padded SVM-train, such that 50% (or slightly more) of the data were assigned to SVM-train, and 25% (or slightly less) were assigned to SVM-test and to Evaluate-test, respectively.

Table 6. Average error rates (in %) for challenge strings, high-quality data

Letters	FMR	FNMR	Letters	FMR	FNMR
1–2	0.35	11.54	7–14	0.01	0
2–4	0.04	0.64	8–16	0	0
3–6	0	0.64	9–18	0	0
4–8	0	0	10–20	0	0
5–10	0	0.64	11–22	0	0
6–12	0	0			

dependent. Our SVM method employs pairwise binary classifiers, and thus the templates are dependent. However, because our target task is insider detection and it assumes no outsiders, our use of dependent templates without special evaluation provisions is reasonable.

8. Results

Table 6 shows the results of the challenge-string evaluation for High-quality Data; these data most closely reflect actual user handwriting. The number of units, u, was varied from 1 to 11, because the subject whose letter list contained the fewest units had only 11. Each unit contains one or two letters, so the number of letters used in a round varies from u to $2u$, depending on the subject. Average false-match rates (FMR) and average false non-match rates (FNMR) both decreased eventually to 0%, as the number of units increased. With only 2–4 letters, FMR=0.04% while FNMR=0.64%. When 8–16 letters were used (or even more), FMR=FNMR=0%.

The results for the Reduced Data and All Data experiments were very similar to the High-quality Data ones. It appears that neither increasing the data quality (while holding quantity constant), nor increasing the data quantity (while holding quality constant), has a clear benefit. Due to the long running time of the evaluations, we were unable to repeat each experiment several times using different random samples, thus preventing estimates of variance and tests of statistical significance. Nevertheless, to probe the stability of the results heuristically, we replicated the High-quality Data experiment once; these results very closely resembled those of the original experiment, as well as those of the Reduced Data and All Data experiments. Note that in the Reduced Data experiment, the subject having the fewest units in his or her letter list had 9; for the All Data experiment it was 8; and for the replication of the High-quality Data experiment, it was 10. The three experiments and the one replication all achieved consistently perfect results when at least 8–16 letters were used. Excepting one chance event (in the seventh round of the High-quality Data experiment), this also happened with only 6–12 letters. The All Data experiment used the most transaction repetitions, and should have the most stable results; it achieved consistently perfect results with only 5–10 letters.

9. Discussion

Although error rates in Table 6 do not decrease monotonically as the number of units increases, deviations from the trend are slight and short-lived. Possible explanations for these small deviations are (1) limited data quantities (or rare atypical instances) in SVM-test and/or Evaluate-test, and (2) sub-optimal challenge strings (because the algorithm for creating letter lists was greedy and heuristic). It just so happened that the trends of the other experiments were more consistent. In general, it appears that adding units to a challenge string should not be harmful. Also, since results under the three different conditions of data quality and quantity did not differ greatly, it seems that (a) data-capture anomalies were not calamitous, and (b) similar performance might be achieved using even less data.

One may ask how it was that such promising results were achieved. Because data from potential attackers was available, template ageing was absent, and ample training data was used, our results may be slightly optimistic. Despite such factors, we feel that system success hailed from the selective grouping of letters into challenge strings. The SVM classifier results were unexceptional (recall Table 2), but combining classifier outputs on letters tailored to each user brought greater success. Some theory supports this rationale; our method is reminiscent of machine learning techniques such as boosting [8] and co-training [1] that conjoin several medium-quality classifiers to produce a very good one.

10. Summary

The aim of this work was to determine whether personalized challenge strings might discriminate enrolled users writing on PDAs. A secondary aim was to determine what approximate length a challenge string must be. Our work suggests that using approximately password-length challenge strings (of at least 5-10 characters, depending on the user) results in a very low equal-error rate, approaching 0%. Employing even longer strings seems to bring consistently perfect results, so asking for a few more seconds of user time (a letter stroke takes less than 0.5 seconds to write, on average) could translate into a very high level of security. These results are a first step towards exploring the promise of structured writing on PDAs to deliver biometric access control.

11. Future work

There are several ways to develop the technology of this work. Using the same data corpus, one could try new features, different splits of training and testing data, other classifiers, new ways to combine classifier outputs, and alternative algorithms to create challenge

strings. Variant challenge strings could be generated non-deterministically, to discourage replay attacks. Effects of writing experience on accuracy could be studied. Other biometric tasks could be attempted, such as positive or negative identification [24], possibly using independent templates [13]. Moreover, causes of system failure could be elicited; multiple replications of experiments could be conducted, to estimate the variance of the results. If new data is collected, its quality should be tested to ensure that it closely represents user handwriting. Higher data granularity, accuracy, and reliability (in time, space, and pressure if available) could be beneficial. Template ageing [13] could be introduced to make the task more difficult. Larger numbers of subjects could be recruited, and the relationship between the size of the user pool and system accuracy could be explored. Finally, one could study and test hypotheses about which kinds of structured letters discriminate writers most effectively.

12. Acknowledgements

The authors are grateful for helpful comments from Patricia Loring and from anonymous reviewers. Marcus Louie implemented the stimulus generator. Sebastian Scherer implemented the data-capture program on the Palm m105. This work was supported by National Science Foundation grant number CNS-0430474 and by the Pennsylvania Infrastructure Technology Alliance.

References

[1] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 92–100, New York, 1998. ACM Press.

[2] G. E. P. Box and D. R. Cox. An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, 26(2):211–243, 1964.

[3] R. R. Bradford and R. B. Bradford. *Introduction to Handwriting Examination and Identification*. Nelson-Hall Publishers, Chicago, 1992.

[4] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[5] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines, 2001. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.

[6] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge Univ. Press, Cambridge, 2000.

[7] E. Dimitriadou, K. Hornik, F. Leisch, D. Meyer, and A. Weingessel. The e1071 package, 2005. Software available at <http://cran.r-project.org/src/contrib/Descriptions/e1071.html>.

[8] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, 2001.

[9] C.-W. Hsu, C.-C. Chang, and C.-J. Lin. A practical guide to support vector classification. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2003.

[10] C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, March 2002.

[11] IJPRAI. Special issue on automatic signature verification. *International Journal of Pattern Recognition and Artificial Intelligence*, 8(3), June 1994.

[12] F. Leclerc and R. Plamondon. Automatic signature verification: The state of the art, 1989–1993. *International Journal of Pattern Recognition and Artificial Intelligence*, 8(3):643–660, June 1994.

[13] A. J. Mansfield and J. L. Wayman. Best practices in testing and reporting performance of biometric devices (version 2.01). Technical report, Centre for Mathematics and Scientific Computing, National Physical Laboratory, Teddington, Middlesex, UK, August 2002.

[14] L. O'Gorman. Comparing passwords, tokens, and biometrics for user authentication. *Proceedings of the IEEE*, 91(12):2021–2040, December 2003.

[15] Palm Inc. Graffiti Alphabet, 2007. Available at <http://www.palm.com/us/products/input/>.

[16] P. J. Phillips, A. Martin, C. L. Wilson, and M. Przybocki. An introduction to evaluating biometric systems. *Computer*, 33(2):56–63, February 2000.

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

[18] R. Plamondon and S. N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63–84, January 2000.

[19] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, 2006.

[20] R. Sabourin, R. Plamondon, and G. Lorette. Off-line identification with handwritten signature images: Survey and perspectives. In H. S. Baird, H. Bunke, and K. Yamamoto, editors, *Structured Document Image Analysis*, pages 219–234. Springer, Berlin, 1992.

[21] E. Shaw, K. G. Ruby, and J. M. Post. The insider threat to information systems: The psychology of the dangerous insider. *Security Awareness Bulletin*, 2–98, September 1998. Department of Defense Security Institute, Richmond, Virginia, USA.

[22] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 2nd edition, 2000.

[23] C. Vielhauer. *Biometric User Authentication for IT Security: From Fundamentals to Handwriting*. Springer, New York, 2006.

[24] J. Wayman, A. Jain, D. Maltoni, and D. Maio. An introduction to biometric authentication systems. In J. Wayman, A. Jain, D. Maltoni, and D. Maio, editors, *Biometric Systems: Technology, Design and Performance Evaluation*, chapter 1, pages 1–20. Springer, London, 2005.

[25] D.-Y. Yeung, H. Chang, Y. Xiong, S. George, R. Kashi, T. Matsumoto, and G. Rigoll. SVC2004: First international signature verification competition. In D. Zhang and A. K. Jain, editors, *Biometric Authentication: Proceedings of the First International Conference, ICBA 2004 (LNCS 3072)*, pages 16–22, Berlin, 2004. Springer.