You are the Key: Generating Cryptographic Keys from Voice Biometrics

Brent Carrara
University of Ottawa
bcarr092@uottawa.ca

Carlisle Adams
University of Ottawa
cadams@site.uottawa.ca

***Abstract*—In this work we apply randomized biometric templates (RBTs) to voice biometrics by performing an experiment using speech samples from the TI46 database. Additionally, we present a novel algorithm for extracting *reliable features* from voice biometrics and analyze the resulting entropy of the cryptographic keys generated by the RBT algorithm. We evaluate our implementation by analyzing the number of guesses required by a powerful adversary to generate a user's cryptographic key when given access to the user's decrypted template and population statistics. Furthermore, we compare our results to the results of prior work. We demonstrate that RBTs are able to generate cryptographic keys with at least 30 bits of entropy for 36% of the population and at least 40 bits of entropy for 7% of the population, while keys generated using prior work only contain at least 20 bits of entropy for 19% of the population. We also demonstrate that RBT generated keys are able to achieve a maximum entropy of 51 bits, while keys generated using prior work are only able to achieve a maximum entropy of 26 bits.**

***Index Terms*—Biometrics, Voice, Cryptographic Keys**

I. Introduction

Biometric key generation is an active area of research due to the permanence, non-repudiation, and portability of an individuals biometric signal. In general, individuals have shown difficulty in generating strong secrets. This has been exemplified by their tendency to choose insecure passwords, share their password with someone else, write their password down, or completely forget their password [1], [2]. These problems can be mitigated by generating strong cryptographic keys from a user's biometric signal [3], [4], [5], [6], [7], [8]. This alternative relies on the user's ability to reliably recreate the same biometric signal when prompted to do so. Not only is the user no longer required to memorize their secret, but the cryptographic keys that they generate are much stronger than the text based secrets they traditionally choose. In order to measure the inherent entropy in biometric signals Biometric Key Generator (BKG) algorithms have been designed. In previous research, BKGs have been applied to many different physical applications. It has been well documented that biometrics can be used as a replacement for password based authentication as well as key management schemes [3], [4], [5], [6], [7], [8]. However, biometrics can also be used in anonymous credential systems [11], [12] as pointed out by Adams in [9]. By using biometrics, Adams shows how it is possible to link a user's digital identity with their physical identity in a privacy-preserving way.

In this paper we present the results of applying a modified version of a BKG algorithm [7] traditionally used to generate cryptographic keys from handwriting biometrics to voice biometrics. We also empirically evaluate the ability of an adversary to recreate a user's biometric key when given access to the user's decrypted template as well as feature statistics from the entire population. The adversary's ability to generate cryptographic keys is evaluated using a probabilistic search algorithm based on the *Guessing Distance* metric [20]. Additionally, we show that cryptographic keys at least 2^{10} times stronger than keys generated from voice biometrics using previous work [4], [5], [6] can be generated for 70% of the population. Our novel contribution in this work is our algorithm for extracting *reliable features* from voice biometrics. We show that it is possible to extract features that demonstrate a high inter-user variation and a low intra-user variation across our population.

Our work has application in many different physical settings. We envision our BKG algorithm being run on either a user's handheld device [4], [5], [6], or on a tamper-proof third-party device [9]. Additionally, our work has applicability in a digital identity management setting as well [33], [34], [35], where a trusted identity manager is responsible for accepting a user's voice signal and producing a cryptographic key from their template. In each of these deployment scenarios, it is conceivable that an adversary could gain access to the user's biometric template, either through gaining access to the user's handle device, by surreptitiously gaining access to the digital identity manager's database, or through gaining access to the tamper-proof device which stores the template. Under this assumption, we evaluate the security of our implementation by calculating the number of guesses it would take an adversary to generate a user's cryptographic key when given access to the user's biometric template.

A. Related Work

The research in the area of privacy-preserving biometric authentication is split into four categories: biometric salting, biometric key generation, fuzzy schemes, and non-invertible transforms [15]. Of particular interest to us in this paper is the work done in the area of biometric key generation.

A number of biometric key generator algorithms have been proposed for different biometrics modalities. In [3], Monrose, et al. proposed hardening an individual's typed password by

978-1-4244-7550-6/10/$26.00 ©2010 IEEE

adding the entropy from their typing dynamics to the contents of their typed password. By observing an individual's key-press duration and inter-key-press delays, their algorithm is able to determine an individual's distinguishing, repeatable features. Once calculated, the distinguishing features are used to reconstruct Shamir secret-shares, which are ultimately combined with the individual's password to add entropy. Furthermore, their scheme is able to adapt to an individual's changing typing patterns over time. The false accept rate (FAR), false reject rate (FRR), added entropy and the number of distinguishing features are determined by a sensitivity parameter, k. By varying the sensitivity parameter, varying levels of FAR, FRR, and entropy can be achieved.

Monrose, et al. further applied their technique of generating hardened typed passwords to voice biometrics in [4], [5], [6]. In [4], [5], Monrose, et al. described a novel technique for generating cryptographic keys that produced a theoretical maximum of 46 bits of entropy. The authors used traditional speech algorithms to extract cepstral feature vectors from each frame of speech. The sequence of feature vectors was then segmented using the *segment vector quantization* algorithm [13]. Once the speech samples were segmented, each segment was mapped to the bit 1, or 0. The resulting bit-string was then used to look up the individual's Shamir secret shares stored in their template. The shares output by the algorithm were used as the individual's cryptographic key. Similar to the result in [3], the resulting metrics of Monrose, et al.'s BKG voice algorithm are dictated by the sensitivity parameter, k. Finding the optimal value of k is of paramount importance to their algorithm. In [6], Monrose, et al. furthered their research by implementing their BKG on handheld PDAs. In their work, the authors discussed generating keys with 60 bits of entropy, however, they did not conclude that extracting the additional bits was possible for every individual. Of paramount importance to our work is the attacker model used by Monrose, et al. In [3], [4], [5], [6], Monrose, et al. were primarily concerned with an attacker having full access to the stored biometric template of an individual; a model we will adopt going forward in this paper.

In [10], Ballard, et al. looked at enumerating and evaluating the ways in which handwriting biometric templates could be attacked in an off-line model. The authors were able to show that trained forgers who were given access to many handwriting samples were able to achieve a very high FAR. In [8], Vielhauer et al. proposed a hashing algorithm for handwritten signatures. In their proposal, 24 features were identified and used in conjunction with user specific statistics to construct their hash. Although they demonstrated an excellent FAR of 0% and a FRR of 7.05%, they provided no analysis on the amount of entropy their algorithm extracted. In [18], however, Vielhauer et al. analyzed their previous work and evaluated their intra-personal feature deviation, inter-personal entropy, and the correlation between the two.

In [7], Ballard, et al. proposed a novel handwriting BKG called randomized biometric templates (RBTs), which produces biometric templates that are unverifiable to an attacker. In [7], Ballard, et al. were concerned with increasing the amount of entropy extracted from handwritten signatures and argued that their RBT algorithm could be applied to many different biometric modalities. Through their novel algorithm, Ballard, et al. were able to demonstrate that a large portion of the population they studied were able to produce cryptographic keys with 44 bits of entropy while a smaller portion of the population was able to produce cryptographic keys with 80 bits of entropy. In [7], Ballard, et al. compared the results of their new RBTs to the biometric hashing scheme of Vielhauer et al. [8], [18]. In their analysis, Ballard, et al. demonstrated that while Vielhauer et al.'s algorithm has a maximum entropy of 40 to 45 bits, theirs has a maximum entropy of 80 bits.

II. Randomized biometric templates (RBTs)

In our work, we modified Ballard, et al.'s proposal [7] to be compatible with voice biometrics. We start by reviewing the RBT algorithm.

The first step in generating a cryptographic key from a biometric signal is to extract features from the signal. The RBT algorithm introduced in this section takes as input a sequence of biometric features and outputs a cryptographic key. In [7], Ballard, et al. hypothesized that their RBT algorithm could be applied to biometric modalities other than handwriting as long as proper features could be extracted. In this paper we show that the RBT algorithm can be modified to in fact generate strong cryptographic keys from voice biometrics.

RBTs require features that are able to effectively differentiate between individuals in the population (i.e. have a large inter-user variation), yet reliably extract similar values from samples of the same individual (i.e. have a low intra-user variation). Once features have been extracted, the RBT algorithm is able to distill entropy from the features by capturing the inter-user variation inherent in the biometric signal. In order to reliably recreate the same cryptographic key, RBTs use quantization as a means to correct small perturbations in subsequent readings from the same individual. Ballard, et al.'s algorithm consists of two main algorithms: an enrollment algorithm, *Enroll*, and a key generation algorithm, *KeyGen*.

1) Enroll ($\beta_1, \ldots, \beta_l, \pi$): The *Enroll* algorithm is a four step process and is shown in **Appendix A**. The algorithm assigns features to a user, computes the necessary error correction information for each of the features, creates a cryptographic key for the user, and finally encodes their secure template, which is used by *KeyGen* to recreate the newly created key.

2) KeyGen (β, π, T): The KeyGen algorithm simply decrypts a user's template, T, with the user's password, π, and attempts to recreate the user's key, K, with their biometric sample, β. The algorithm iteratively checks to see if a computed hash value is equal to the verification hash, v, encoded in the user's template. Once the hashes match, the true key is computed using the hash, H_{key}, and the key is returned by the algorithm. If the key cannot be recreated, the value $\perp$ is returned. In this work, we modified Ballard, et al.'s *KeyGen*

algorithm to be able to reliably recreate cryptographic keys generated from voice biometrics.

III. Algorithms

In this section we introduce our novel feature extraction algorithm, which we will use to convert speech signals into usable features for the RBT algorithm. We also discuss our algorithm for segmenting a sequence of n 24-dimensional feature vectors into a sequence of k (x, y) coordinates. Additionally, we discuss our method for mapping each segment to a set of usable features.

The feature extraction algorithm we outline in this section to extract *reliable features* is dependent on an implicit ordering of the underlying signal being processed. In our case, this ordering is time, since we are dealing with voice signals. Our approach splits the speech signal up into time quanta and extracts features from each quantum for evaluation. It is important to note however, that we do not necessarily extract features from the components of the speech signal, i.e. phonemes. Continuing with this argument, we make the contention that our algorithm could be applied to other signals that also have a time ordering to them. With this in mind, we hypothesize that our algorithm could be used to extract reliable features from other biometric signals, such as typing, and handwriting, since they too can be observed in such a way that they vary in time in a repeatable way.

The data structure chosen to model the population's speech in our study was the self-organizing map (SOM) of Kohonen [21]. Self-organizing maps can effectively be used to project high-dimensional spaces onto a lower, two-dimensional space. In our experiment, vectors of 24-dimensions were mapped to a much more manageable two-dimensions through the use of a SOM. In addition to converting high-dimensional data into a two-dimensional space, the SOM also preserves the "ordering" of the high-dimensional data. This property of SOMs allows for complex vectors to be analyzed and visualized in a much simpler fashion.

A. Feature extraction algorithm

Now that we have outlined the RBT algorithm and discussed self-organizing maps, we introduce our novel algorithm for converting speech signals into *reliable features*. As previously mentioned, Ballard, et al.'s original implementation of RBTs was applied to handwriting samples. Features were extracted from each handwriting sample and used to create RBTs and cryptographic keys for each of the users in their study. During their study, Ballard, et al. extracted a number of different features, e.g. the velocity of the pen during the signature, the start and end coordinates of the signature, etc. Although effective measures for handwriting biometrics, they have no analogous metric in speech processing.

The steps of our feature extraction algorithm are shown in **Figure 1**. Our algorithm takes as input a speech sample, β, and outputs a sequence of features, $\phi_1, \phi_2, \ldots, \phi_{2k}$. Our algorithm starts by capturing a speech sample, β_i, (Step 1) and performs perceptual linear predictive (PLP) analysis [22] as well as delta analysis [14] on each frame (Step 2). From the PLP and delta analysis, the frames are converted into a sequence of n 24-dimensional vectors consisting of PLP coefficients and delta coefficients (Step 3). We then use a SOM to quantize each 24-dimensional vector. Each 24-dimensional vector is quantized to its associated two-dimensional best matching unit (BMU) as determined by our SOM. The BMU, M_{x_i,y_i}, for each vector in the sequence is calculated from our SOM to form the sequence $M_{x_1,y_1}, M_{x_2,y_2}, \ldots, M_{x_n,y_n}$ (Step 4). Each BMUs' coordinates are then extracted to form the sequence of coordinates $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ (Step 5). The sequence of coordinates is then segmented into k segments using our segmentation algorithm (Step 6). Each segment is then mapped to the coordinates, (x_{b_i}, y_{b_i}), which best minimizes the total cumulative distance from each of the coordinates (x_i, y_i) in the segment, to the coordinates (x_{b_i}, y_{b_i}), resulting in a sequence of k coordinates $(x_{b_1}, y_{b_1}), (x_{b_2}, y_{b_2}), \ldots, (x_{b_k}, y_{b_k})$ (Step 7). From the sequence of k coordinates, each of their x_{b_i} and y_{b_i} components are then mapped to the feature values ϕ_i and ϕ_{i+1} respectively, to create the sequence of features $\phi_1, \phi_2, \ldots, \phi_{2k}$ (Step 8). The sequence of features $\phi_1, \phi_2, \ldots, \phi_{2k}$ is then fed into the RBT algorithm to generate a user's cryptographic key and biometric template.

A detailed description of how we segment each sequence of coordinates as well as how we map each segment to usable features is presented in the next two sections.

B. Segmenting the frames

Our segmentation algorithm chooses the set of k nodes from our SOM which best describe a sequence of n frames. Let us assume that we have extracted our 24-dimensional vectors from a given speech sample and have found each vector's associated BMU. Let us also assume that we have created a sequence of coordinates from the BMUs, which we represent by $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. Let $R_1 \ldots R_k$ be disjoint, nonempty ranges over the natural numbers $\cup_{i=1}^{k} R_i = [1, n]$, where the i-th segment is $(x_j, y_j), \ldots, (x_{j'}, y_{j'})$ if $R_i = [j, j']$. Our goal in segmenting this sequence of feature vectors is to reduce the sequence of n coordinates into a list of k disjoint ranges over $[1, n]$ by reducing the error, E_i, for each segment:

$$E_i(R_i, (x_{b_i}, y_{b_i})) = \sum_{k=j}^{j'} d((x_k, y_k), (x_{b_i}, y_{b_i})) \quad (1)$$

Here (x_{b_i}, y_{b_i}) denotes the coordinates of the node in our SOM found to best minimize **Equation 1**, and the function $d(X, Y)$ represents the Euclidean distance between two vectors, X, and Y.

The goal of our segmentation algorithm is therefore to minimize the total error, E, for all the segments:

$$E = \sum_{i=1}^{k} E(R_i) \quad (2)$$

The problem of segmenting a sequence of n two-dimensional values into k segments has been deemed the

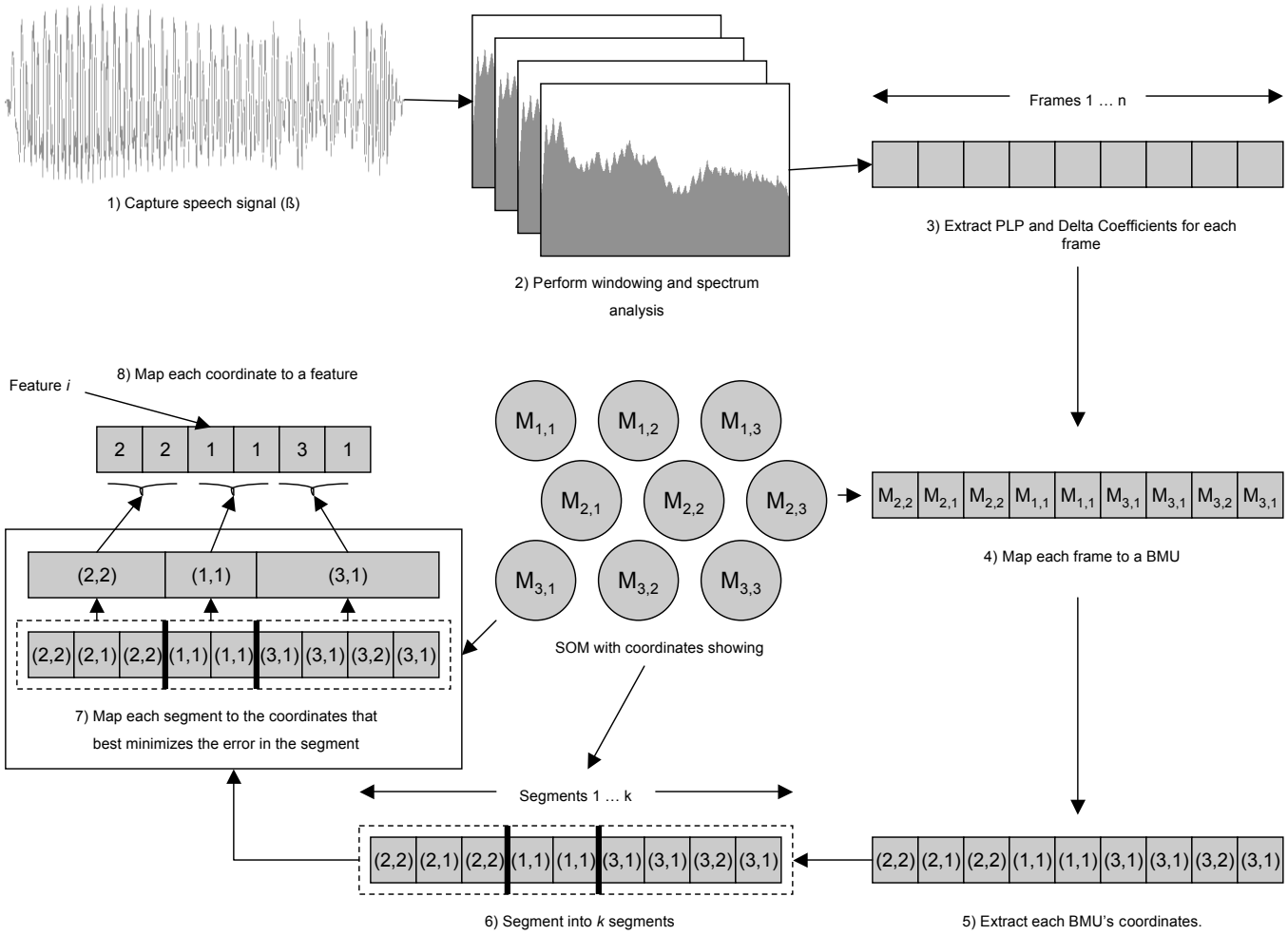

Fig. 1. Flow diagram of our feature extraction algorithm.

k-segmentation problem. The *k-segmentation* problem is optimally solved by the dynamic programming algorithm of Bellman [25]. The main drawback of the traditional dynamic programming approach however, is that Bellman's algorithm, although optimal, has a run-time of $O(n^2k)$. In order to speed up the run-time of Bellman's algorithm, we designed a heuristics based algorithm to segment each of our speech samples. Our segmentation algorithm is split into two stages, *Grouping* and *Segmenting*.

1) Grouping: During the *grouping* stage, coordinates that are "close" to one another are grouped together. The algorithm iteratively places contiguous coordinates into the same group if and only if they are within a certain distance of the component-wise mean of the group. The pseudo-code for the algorithm is shown in **Appendix B**. In our experiments a $THRESHOLD$ value of 10 was found to work best.

2) Segmenting: The next stage of the algorithm segments the n coordinates into k coordinates. Depending on how many groups were created during the *Grouping* stage, groups are either split into two groups if $|R| < k$, or two consecutive groups are merged together if $|R| > k$. Groups were split by searching for a group that when split reduced the total error, E, by the greatest amount. Groups were merged by searching for two consecutive sub-groups that when merged raised the total error by the least amount. In both cases this stage proceeded iteratively either reducing or increasing the cardinality of R until $|R| = k$. Once $|R| = k$, we represented each of the groups in R by the coordinates, (x_{b_i}, y_{b_i}), of the BMU found to best minimize **Equation 1**. Each of the coordinates (x_{b_i}, y_{b_i}) were then mapped to features.

After the *grouping* stage, groups in R that had a cardinality of one were removed prior to the *segmentation* stage of the algorithm. Our heuristics based segmentation algorithm, described in this section, reduced the time it took to segment our speech samples from a number of days to a couple of hours.

C. Mapping segments to features

Once the frames of speech were segmented into their (x_{b_i}, y_{b_i}) coordinates, the coordinates were used to create the

features required by the RBT algorithm. In our study, each of the x and y components that represented a segment of speech were individually mapped to a feature. The x and y coordinates were mapped to ϕ_i and ϕ_{i+1} respectively, thus resulting in two features being mapped to each segment.

During the creation of our technique, other mappings were also tried. The most promising mapping (other than the one we ultimately used) mapped each node directly to a feature by setting $\phi_i = (x_{b_i}, y_{b_i})$. This required mapping each feature to the two-dimensional coordinates representing each segment. This however, greatly reduced the number of *reliable features* in our study (by half), resulting in lower entropy keys. In this study we report the results based on mapping each segment to two distinct features.

Both the SOM and our segmentation algorithm provided us with a way to efficiently convert a sequence of 24-dimensional vectors into a set of usable features. We chose to represent our speech samples as a sequence of two-dimensional coordinates to take advantage of the underlying structure of our SOM. By reducing the dimensionality of our feature vectors from 24 to two, we were better able to create features that relatively made sense. A segment coordinate $(10, 12)$, for instance, is relatively close to the coordinate $(11, 14)$. By extracting the x-component of both these coordinates as a feature we were able to create a feature that exhibits a low intra-user variation (assuming both these coordinates are from the same user's speech samples). A segment coordinate $(10, 12)$ and $(35, 42)$ however, are not relatively close to one another. By extracting the x-component of both these coordinates and comparing them to one another we can discard the x-component as a feature because it exhibits a relatively high intra-user variation (again assuming these coordinates are from the same user's speech samples). This simple property of mapping high dimensional data to an ordered two-dimensional space, allowed us to extract very strong and reliable features.

Our algorithm for mapping speech samples to *reliable features* is the novel contribution of this paper. In our implementation of applying RBTs to voice biometrics our biggest challenge was extracting features that allowed the RBT algorithm to extract a high amount of entropy while minimizing both our FAR and FRR. In handwriting, features can easily be extracted by observing the mechanical movements of a user's hand while they execute their signature, however, extracting features from voice biometrics required extracting features from the raw input signal of the voice samples in a reliable way.

IV. Evaluation

In this section we describe the metrics we used to evaluate our implementation and discuss how we modified the original RBT algorithms for our purposes. We also discuss the setup of our experiment and provide empirical results for the important evaluation metrics.

A. Entropy

Traditionally, the measure *Guessing Entropy* has been used to quantify the entropy in cryptographic keys generated by BKGs. Most notable to us in this study is the entropy reported by Monrose, et al. in [4], [5], [6], where they applied their BKG to voice biometrics. In their analysis, the authors concluded that the theoretical maximum amount of entropy in their keys is summarized by the equation $guesses = min\{2^m, (|A| + 1)/2\}$, where m is the number of features used to generate a user's key and A is the set of users in their experiment. What this metric says is that given the set of keys generated by the their BKG it would take $guesses$ on average to map a key to a certain individual. There are a number of problems with Monrose, et al.'s use of this metric however. One of the problems is that the authors assume keys are generated in a uniform fashion and they do not account for the fact that some users in the system generate keys that are easier to guess than others. Instead, they attempt to summarize the average number of guesses required for the population as a whole.

In this study, we use the metric *Guessing Distance* to address this issue [16], [20]. *Guessing Distance* measures the number of guesses required by an adversary who assumes population statistics can be used to determine an individual's feature values. The metric measures how many guesses are made by an adversary before they correctly guess that a specific user generates the feature value, w, for a given feature. As oppose to being a theoretical metric, *Guessing Distance* is based on empirical data, and requires feature value probability distributions in order to produce an accurate estimation. The *Guessing Distance* estimation algorithm, although originally used to analyze the entropy in keys generated by a handwriting BKG, was designed to be agnostic of the underlying biometric signal being measured. *Guessing Distance* was chosen over *Guessing Entropy* because it allowed us to model a logical adversary's attack on a given user's biometric template using empirical data.

B. False rejection rates and false acceptance rates

Two other metrics we report in this paper are the false rejection rate (FRR) and the false acceptance rate (FAR) of our implementation. The FRR was calculated by taking the percentage of authentic speech samples that were rejected as not being able to generate the correct cryptographic key from an authentic user's template, while the FAR was calculated by taking the percentage of *imposter* samples that were accepted as having generated the correct biometric key from an authentic user's template.

Before quantifying the FAR and FRR of our implementation, we first present our modifications to the *KeyGen* algorithm. We modified the original *KeyGen* algorithm to correct a specific type of error commonly found in our implementation. RBTs, on their own, correct errors through quantization, while this provides a good starting point for error correction it was found that the observed FRR was still very high. Deeper inspection of the problem revealed that quite often the same

Sample Id	s_1	s_2	s_3	s_4	s_5	s_6
Sample 1	01,40	**14,15**	31,16	18,33	27,00	26,24
Sample 2	11,27	01,41	**15,14**	30,16	27,04	27,25
Sample 3	01,41	**15,13**	31,18	19,33	27,01	29,23
Sample 4	02,41	**14,14**	30,17	18,33	27,00	29,23

TABLE I
SEGMENTED SAMPLES FOR USER $M1$ AND UTTERANCE ZERO.

feature value in subsequent speech samples from the same individual were found to be misaligned. An example of the feature values for user $M1$ are shown in **Table I**.

Let us examine the values for segment s_2. We can easily see that the expected value, (**15,14**), is misaligned and appears in segment s_3 in *Sample 2*. This type of error was extremely common in our implementation. Fortunately, the errors were quite easy to correct. We corrected the errors as follows. Let us assume we are generating a cryptographic key from *Sample 2*. Let us also assume that the x-component of s_2 has been encoded in user $M1$'s template as a feature. To correct the error found in *Sample 2* we attempt to generate the user's true cryptographic key by creating three keys. Instead of simply generating one cryptographic key using the x-component of s_2, we generated three cryptographic keys each with the values from the x-component of s_1, s_2, and s_3 respectively. In our example, the keys generated from *Sample 2* using the x-component of s_1 and s_2 would result in keys that do not match the template's verification hash. However, the key generated using the x-component of s_3 would result in a key that does match the verification hash in the user's template. We evaluated the FAR and FRR of our implementation after applying this new algorithm to KeyGen.

C. Experimental methodology

Our experiment was setup as follows. We used a pre-existing data set, the TI46 [27] database, to simulate a group of users uttering the sequence of numbers zero through nine as their passphrase. The TI46 database contained speech samples from each user uttering all the letters of the alphabet, the numbers zero through nine, and ten other words. In our study, however, we only used each of the users' 26 utterances of the numbers zero through nine, resulting in 26 different versions of the passphrase for each user. We partitioned the speech samples into enrollment samples and test samples in a 4:1 ratio. The enrollment samples were used to create our SOM, as well as each of the user's templates, while the test samples were used to test the FRR of our implementation. Additionally, the FAR of our implementation was tested with both the enrollment samples and test samples.

The original TI46 data set consisted of 16 users, eight women and eight men, however, through the use of pitch/time modifications the set of 16 users was grown to 64 [32] in order to increase our sample space. Adobe® Audition® 3 [28] was used to perform the pitch/time modifications. Three different scaling factors were used for the group of men and women and were chosen to ensure the resulting voices still sounded human. The pitch/time modifications performed on our original data set however, are independent of our implementation and are not a required step in our algorithm. To ensure that the we created new users that had human sounding voices we listened to the synthetic speech samples we created to confirm that they sounded human and that they were audibly different from the original speaker.

As part of our algorithm we used traditional signal processing techniques to clean up the samples from the TI46 database. Each speech sample in the data set was amplified and had frames of silence removed from it. Again, Adobe® Audition® 3 was used to perform the signal processing tasks because of its batch scripting capability. Removing silence is a required step in our algorithm. It ensures the speech signals being processed by our feature extraction algorithm contain the most feature rich segments. We found that this step was integral to the success of applying RBTs to voice. Without silence detection a large number of segments were mapped to the same value because no distinguishing features could be extracted from frames containing no speech.

After processing the speech samples with Adobe® Audition®, the speech samples were further processed individually by the Hidden Markov Model Toolkit (HTK) [29] to extract PLP and delta coefficients. HTK was used to generate each of the 24-dimensional vectors that were subsequently used as input to the training phase of our SOM. In addition to using the HTK library to perform both PLP and delta analysis, the HTK library was configured to perform *Cepstral mean normalization* (CMN). CMN is a technique used to compensate for undesirable spectral effects caused by imperfections in audio recording. Although traditionally used when dealing with lengthy samples of speech it was employed in this study.

As previously described, a SOM was used to model the speech samples of our population. All the speech samples were processed by HTK and the enrollment samples were used to train our SOM. The SOM_PAK [24] toolkit was used to train a 48x48 SOM, organized in a hexagon topology. The SOM was initialized with the eigenvectors of the enrollment data to better seed the initial weight vectors of the nodes in the SOM. The SOM was then trained using a learning rate of 0.1 and a radius of 2 while 10,000,000 iterations were used to allow the nodes to converge to an acceptable cumulative error. A number of varying radii and learning rates were also tested in our study by first creating the SOM with the varying parameters and then testing the cumulative error. The cumulative error was calculated by taking the sum of the cumulative distances between each enrollment vector and its associated BMU. It was found that a lower learning rate and lower radius performed best with our data set.

Once the SOM was created, each sequence of feature vectors was segmented into six segments by the segmentation algorithm discussed in **Section III-B**. Of the six segments, the first and the last segments were not used as *reliable features*. These two segments were discarded because they showed a very high variance in the nodes they matched. This was primarily due to the imperfections in the silence

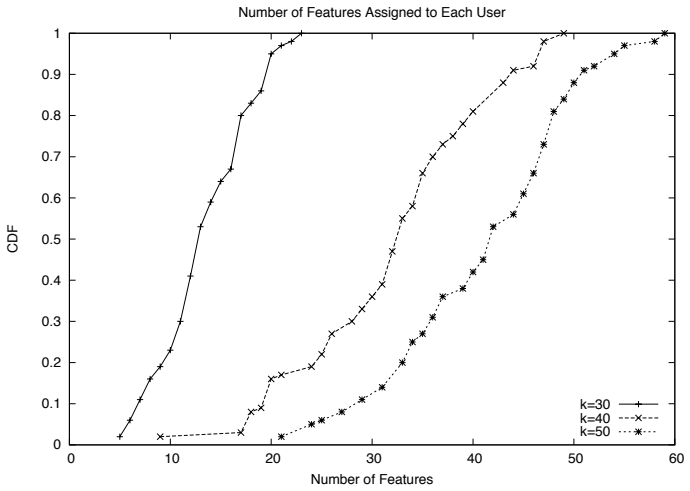

Fig. 2. CDF of the number of *reliable features* in each user's template, i.e. the size of Ψ, for varying levels of k

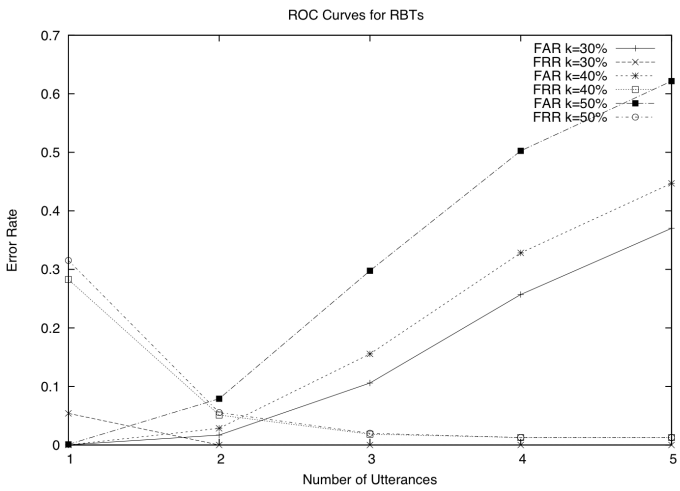

Fig. 3. ROC for varying levels of k.

detection tool used during the signal processing stage of our experiment. Since the first and last segments were discarded, however, only four segments were used as possible features for each utterance. Since each passphrase contained the utterances zero through nine, a total of ten utterances were used. Each utterance was segmented into four segments, and each segment was mapped to two features, resulting in 80 possible features per passphrase. It should also be noted that although the first and last segments were discarded as *reliable features*, they were used to align features in our modified *KeyGen* algorithm.

D. Experimental results

An important algorithm used within the *Enroll* algorithm has not yet been discussed. In **Appendix A**, on line 1, the function **Select** is called by *Enroll*. While the details of **Select** are somewhat complicated they must be explained in some detail in order to properly understand the results of our experiment. Further details about the **Select** algorithm can be found in [7]. The **Select** algorithm performs two distinct tasks. The first is to calculate the individual feature quantization widths, δ_i, by taking into account statistics across the entire population. For each feature value, ϕ_i, each user, u, is tested to see how much error tolerance, $d_{u,i}$, they require to replicate the feature ϕ_i reliably. Each of the $d_{u,i}$ values are ordered such that $d_{u,i} \leq d_{u',i}$. The $d_{u,i}$ value at the k^{th} percentile is then taken to be the value for δ_i. After the **Select** algorithm has determined all the δ_i values, it assigns features to users. To do so, each user's $d_{u,i}$ value is tested to see if $d_{u,i} \leq \delta_i$. If their value falls below δ_i the feature i is added to their set of *reliable features*. As expected, as we increased the value for k, more and more features were added to the RBT generated templates. **Figure 2** shows the cumulative distribution function (CDF) of the number of features encoded in each user's template, i.e. the number of *reliable features*, $|\Psi|$, for varying levels of k.

In order to calculate the FAR and FRR of our implementation a repeated leave-out-k cross validation algorithm was used. Given v samples, we randomly chose $v - k$ samples to create the RBTs and k samples to test the FRR of our implementation. $v - k$ and k were set to be in the ratio 4:1. We tested the FAR by using all v samples from each *imposter* to try and generate an authentic key, given the authentic user's template. Additionally, each template created had the maximum number of *reliable features* in it capped at $|\Psi| = 25$, $|\Psi| = 50$, and $|\Psi| = 60$ for $k = 30$, $k = 40$, and $k = 50$ respectively. This ensured that the strongest possible templates were created for each user for varying levels of k. The FRR and FAR of our implementation are shown in **Figure 3**. It can be seen that if a user utters their passphrase twice a FRR of less than 10% can be achieved. Another very interesting observation is that the FAR at one and two utterances is very low.

The most important metric reported in this study is the amount of entropy in the keys generated by our implementation. In order to compare our implementation against a baseline, we implemented the algorithm of Monrose, et al. [5] as a means to compare our implementation against prior work. Monrose, et al.'s algorithm was implemented using PLP and delta coefficients as their features to ensure we made a fair comparison between the two implementations. In our implementation of Monrose, et al.'s algorithm, we were able to achieve FRR and FAR similar to their documented results. Additionally, in order to quantify the entropy in both our keys as well as the keys of Monrose, et al. we implemented the *Guessing Distance* estimation algorithm as detailed in Ballard's dissertation [20]. The results are shown in **Figure 4**, where the baseline is the observed entropy in our implementation of Monrose, et al.'s BKG as documented in [5].

It should be noted that the entropy reported in this analysis assumes an attacker has access to the **decrypted** version of a user's template and therefore does not account for the amount of entropy in the user's password, π. The entropy reported

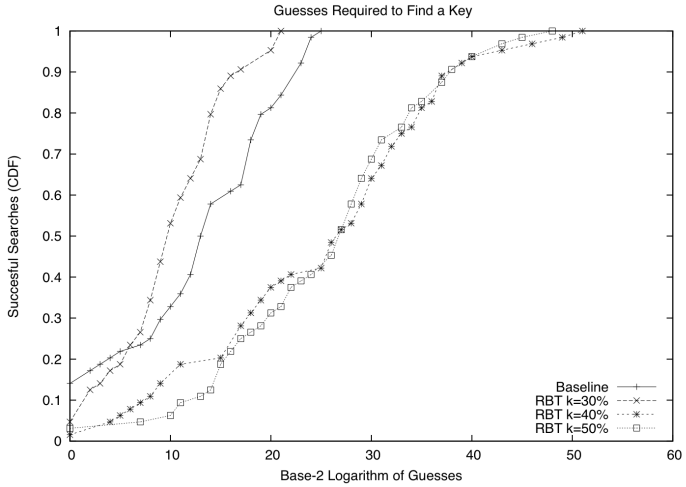

Fig. 4. CDF of the number of guesses required by Ballard, et al.'s search algorithm before finding a user's RBT-derived key.

in this study is therefore an absolute lower bound. Using an encrypted template as the input to the entropy calculation would have increased the resulting amount of entropy by the amount of entropy in the user's password. However, we chose to ignore the entropy in π and focus on the entropy generated by the voice biometric in isolation.

In **Figure 4** it can be observed that at $k = 30$, the templates created by the RBT algorithm produced weaker keys than that of previous work. This was due to the fact that at $k = 30$ the templates contained a very low number of *reliable features*, with 50% of the population assigned $|\Psi| = 12$ or fewer features. At higher levels of k however, users were assigned a much larger number of features with 50% of the users being assigned over $|\Psi| = 30$ and $|\Psi| = 40$ features for $k = 40$ and $k = 50$ respectively. The added number of features increased the entropy of the keys generated by the RBT algorithm. Our results show that RBTs outperformed the BKG of Monrose, et al. by assigning keys that required at least 2^{30} guesses to 36% of the population. The algorithm of Monrose, et al. however, assigned keys that required at least 2^{20} guesses to only 19% of the population. Even more encouraging was the fact that for 7% of the population, keys requiring at least 2^{40} guesses were created using the RBT algorithm. The RBT algorithm was also able to generate keys with a maximum entropy of 51 bits compared to a maximum entropy of 26 bits for keys generated by Monrose, et al.'s algorithm. Our results show that the keys generated using RBTs are able to achieve a much more practical amount of entropy than the keys generated from the algorithms of prior work.

V. Future work

The results from this study, although preliminary, are definitely a step forward in creating cryptographic keys from voice biometrics. However, there are a number of areas that require further research, specifically to increase the entropy of the generated cryptographic keys. In this work, we perform an experiment and present our empirical results. Going forward, further analysis is required to determine the exact mapping of usable features to speech components, i.e. phonemes. By mapping usable features to phonemes it is possible that the passphrase uttered by a user could be varied as long as the underlying phoneme structure of the speech used to generate the biometric keys remains constant.

In order to achieve more entropy, there are two primary ways forward. The first is to gather speech samples from more users and apply our technique. As more diverse samples are gathered, the feature values extracted using RBTs will hopefully start to approach a more uniform distribution, thus reducing the advantage an adversary has in guessing a user's feature values. By choosing a diverse array of users, from varying backgrounds and both sexes it is believed that better feature value distributions can be achieved. In that same vein, more feature rich speech samples are required to increase the number of *reliable features*. In our test, users simply uttered the numerical values zero through nine as their passphrase. In order to generate keys with more entropy, more features are required and thus more feature-rich utterances are needed. A more comprehensive test would have a diverse array of users, preferably in the hundreds, being prompted to say feature-rich words in order to achieve a more practical amount of entropy.

VI. Conclusion

In this work, we show that higher entropy keys are achievable by applying a modified version of randomized biometric templates to voice biometrics. We introduce a novel feature extraction algorithm to extract *reliable features* from voice biometrics and make the argument that the algorithm can be applied to any signal that has an underlying ordering to it. By using our novel feature extraction algorithm we show that it is possible to apply RBTs to voice biometrics in order to achieve practical results. We also show that an adversary given access to *auxiliary information* is still required to perform an increased number of guesses for a high percentage of the population before guessing a correct key. Additionally, we compare the entropy in our RBT generated keys to that of previous work and show that our system is able to generate keys with at least 30 and 40 bits of entropy for 36% and 7% of the population respectively, while those of previous work are only able to achieve 20 bits of entropy for 19% of the population. We also show that RBT generated keys are able to achieve a maximum entropy of 51 bits, while keys generated from the algorithms of previous work are only able to achieve a maximum entropy of 26 bits. Lastly, we show that acceptable levels of FRR and FAR are achievable and describe, in detail, how the amount of entropy can be improved by gathering more feature rich utterances from a more diverse population of users.

Acknowledgment

The authors are grateful to Fabian Monrose and Lucas Ballard for their very helpful feedback on an earlier version of this paper.

REFERENCES

[1] A. Alvare. *How Crackers Crack Passwords or what Passwords to Avoid*, In Proceedings of the Second USENIX Security Workshop, pg. 103-112, August 1990.

[2] D. Fieldmeier, and P. Karn. *UNIX Password Security - Ten Years Later*. In Proceedings of Advances in Cryptology - CRYPTO, pg. 44-63, 1990.

[3] F. Monrose, M. K. Reiter, and S. Wetzel. *Password hardening based on keystroke dynamics*. International Journal of Information Security, pg. 69-83, 2002.

[4] F. Monrose, M. K. Reiter, Q. Li, and S. Wetzel. *Using voice to generate cryptographic keys*. ODYSSEY-2001, pg. 237-242, 2001.

[5] F. Monrose, M. K. Reiter, Q. Li, and S. Wetzel. *Cryptographic key generation from voice*. IEEE Symposium on Security and Privacy, pg. 202-213, 2001.

[6] F. Monrose, M. K. Reiter, Q. Li, D. P. Lopresti, C. Shin. *Toward speech-generated cryptographic keys on resource constrained devices*, Proceedings of the 11th USENIX Security Symposium, pg. 283-296, 2002.

[7] L. Ballard, S. Kamara, F. Monrose, and M. K. Reiter. *Towards practical biometric key generation with randomized biometric templates*. Proceedings of the 15th ACM Conference on Computer and Communications Security, pg. 235-244, 2008.

[8] C. Vielhauer, R. Steinmetz, and A. Mayerhofer. *Biometric Hash based on Statistical Features of Online Signatures*. 16th International Conference on Pattern Recognition, vol. 1, pg. 123-126, 2002.

[9] C. Adams. *Achieving Non-Transferability in Credential Systems Using Hidden Biometrics*. Security and Communication Networks, 2009.

[10] L. Ballard, F. Monrose, and D. Lopresti. *Biometric authentication revisited: Understanding the impact of wolves in sheeps clothing*. Proceedings of the 15TH USENIX Security Symposium, pg. 29-41, 2006.

[11] S. Brands. *Rethinking Public Key Infrastructures and Digital Certificates*. MIT Press, 2000.

[12] J. Camenisch, and A. Lysyanskaya. *An efficient system for non-transferable anonymous credentials with optional anonymity revocation*. Advances in Cryptology Eurocrypt 2001, vol. 2045, pg. 93.118, 2001.

[13] L. Rabiner, B-H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[14] J. R. Deller Jr., J. H. L. Hansen, and J. G. Proakis. *Discrte-Time Processing of Speech Signals*. Macmillan Publishing Co., 1993.

[15] N. K. Ratha, S. Chikkerur, J. H. Connell, and R. H. Bolle. *Generating Cancelable Fingerprint Templates*. IEEE Transactinos on Pattern Analysis and Machine Intelligence, vol. 29, no. 4, pg. 561-572, 2007.

[16] L. Ballard, S. Kamara, F. Monrose, and M. Reiter. *On the Requirements of Biometric Key Generators*. Technical report, Whiting School of Engineering, John Hopkins University, 2007.

[17] H. Feng, and C. C. Wah. *Private key generation from on-line handwritten signatures*. Information Management and Computer Security, vol. 10, no. 4, pg. 159-164, 2002.

[18] C. Vielhauer, and R. Steinmetz. *Handwriting: Feature correlation analysis for biometric hashes*. EURASIP Journal of Applied Signal Processing, vol. 2004, pg. 542-558, 2004.

[19] L. Ballard, S. Kamara, M. K. Reiter. *The Practical Subtleties of Biometric Key Generation*. In Proceedings of the 17th annual USENIX Security Symposium, pg. 29-41, 2006.

[20] L. Ballard. *Robust Techniques for Evaluating Biometric Cryptographic Key Generators*. PhD thesis, The Johns Hopkins University, 2008. Available at http://www.cs.jhu.edu/ lucas/papers/thesis.html.

[21] T. Kohonen. *Self-organizing maps Third Edition*. Springer, 2001.

[22] H. Hermansky. *Perceptual Linear Predictive (PLP) Analysis of Speech*. Journal of Accoustic Society of America, vol. 87, issue 4, pg. 1738-1752, 1990.

[23] J. W. Sammon. *A Nonlinear Mapping for Data Structure Analysis*. IEEE Transactions on Computers, vol. 18, issue 5, pg. 401-409, 1969.

[24] T. Kohonen, J. Hynninen, J. Kangas, and J. Laaksonen. *SOM_PAK: The Self-Organizing Map Program Package*. Technical Report A31, Helsinki University of Technology, Laboratory of Computer and Information Science, 1996.

[25] R. Bellman. *On the approximation of curves by line segments using dynamic programming*. Communications of the ACM, vol. 4, issue 6, pg. 284, 1961.

[26] J. L. Massey. *Guessing and Entropy*. In Proceedings of the 1994 IEEE International Symposium on Information Theory, pg. 204, 1994.

[27] M. Liberman, et al., *TI 46-Word*. Linguistic Data Consortium, 1993.

[28] Adobe, *Adobe Audition 3*. http://www.adobe.com/products/audition/, 2009.

[29] S. J. Young et al. *HTK: Hidden Markov Model Toolkit V3.4*. Cambridge University Engineering Department, 2009.

[30] S. J. Young et al. *The HTK Book*. http://htk.eng.cam.ac.uk/prot-docs/htk_book.shtml, 2009.

[31] J. A. Rice. *Mathematical Statistics and Data Analysis*. Wadsworth, Inc, 1988.

[32] M. Kahrs, and K. Brandenburg, *Applications of Digital Signal Processing to Audio and Acoustics*. Kluwer Academic Publishers, 1998

[33] D. Recordon, and D. Reed, *OpenID 2.0: A Platform for User-Centric Identity Management*. Proceedings of the 2nd ACM Workshop on Digital Identity Management, 2006

[34] R. L. Morgan, S. Cantor, S. Carmody, W. Hoehn, and K. Klingenstein, *Federated Security: The Shibboleth Approach*. http://www.educause.edu/EDUCAUSE+Quarterly/EDUCAUSEQuarterlyMagazineVolum/FederatedSecurityTheShibboleth/157315, 2004

[35] D. Chappelle, *Introducing Windows CardSpace*. http://msdn.microsoft.com/en-us/library/aa480189(loband).aspx, 2006

APPENDIX A
ENROLL ALGORITHM [7]

Input: The password, $\pi \in \Pi$, and biometric samples $\beta_1, \ldots, \beta_l$.
Input: (Global value): The set of all features Φ
Input: (Global value): Quantization widths $\delta_0, \ldots, \delta_N$.
Output: The key K, and template T..

$(\Psi, \widetilde{\Psi}) \leftarrow$ Select($\beta_1, \ldots, \beta_l$) // Select biometric features
$L \leftarrow$ Permute(Ψ)||Permute($\widetilde{\Psi}$)
$k_0 \leftarrow H_{pass,0}(\pi), k_1 \leftarrow H_{pass,1}(\pi)$
for $j = 0$ to $|L| - 1$ **do**
  $i \leftarrow L[j]$
  $\mu_i \leftarrow$ Median($\phi_i(\beta_1), \ldots, \phi_i(\beta_l)$)
  if $\mu_i \geq \frac{\delta_i}{2}$ **then**
    $\alpha_i \leftarrow \lfloor \mu_i - \frac{\delta_i}{2} \rfloor \bmod \delta_i$
  else
    $\alpha_i \leftarrow \lfloor \mu_i + \frac{\delta_i}{2} \rfloor$
  end if
  $x_i \leftarrow \max(0, \lfloor \mu_i - \frac{\delta_i}{2} \rfloor)$ // Quantize feature outputs
  $\gamma_i \xleftarrow{R} [\alpha_i, \Delta]_{\delta_i}$ // Select random quantisation offset
  $C_j = (E^N_{k_0}(i), E^{\Delta}_{k_1}(\gamma_i))$ // Encrypt values
  $K_j = i||x_i$
end for
$K \leftarrow H_{key}(\pi||K_0||K_1||\ldots||K_{|\Psi|-1})$ // Derive key
$C \leftarrow (C_0||C_1||\ldots||C_{|L|-1})$
$v \leftarrow H_{ver}(\pi||K_0||K_1||\ldots||K_{|\Psi|-1})$
return $K, T = (C, v)$

APPENDIX B
GROUPING ALGORITHM USED TO SEGMENT A SEQUENCE OF COORDINATES

Input: The sequence of coordinates $(x_1, y_1), \ldots, (x_n, y_n)$.
Output: The segmented groups of coordinates, R.

$G \leftarrow ()$
$R \leftarrow ()$
for $i = 1$ to n **do**
  if G is empty **then**
    G.append((x_i, y_i))
  else
    $\mu_i \leftarrow$ Mean(G)

```
8:        if distance( (x_i, y_i), μ_i ) ≤ THRESHOLD then
9:            G.append((x_i, y_i))
10:       else
11:           R.append(G)
12:           G ← ()
13:           G ← (x_i, y_i)
14:       end if
15:   end if
16: end for
17: if G is not empty then
18:   R.append(G)
19: end if
20: return R
```