

# Using Voice to Generate Cryptographic Keys

*Fabian Monrose, Michael K. Reiter, Qi Li, Susanne Wetzels*

Bell Labs, Lucent Technologies  
Murray Hill, NJ, USA

## Abstract

In this position paper, we motivate and summarize our work on repeatably generating cryptographic keys from spoken user input. The goal of this work is to enable a device to generate a key (e.g., for encrypting files) upon its user speaking a chosen password (or passphrase) to it. An attacker who captures the device and extracts all information it contains, however, should be unable to determine this key. We outline our approach for achieving this goal and present preliminary empirical results for it. We also describe several directions for future work.

## 1. Introduction

In computer security parlance, a *reference monitor* is an abstract machine that mediates all accesses to objects by subjects in a computer system (e.g., [5]). The speech processing community has made significant progress in implementing speech-based reference monitors to mediate access by humans to computer resources. These reference monitors are typically implemented using speaker authentication technologies including automatic speaker verification (SV) [1] and verbal information verification (VIV) [8]. SV is the process of verifying whether an unknown speaker is the person as claimed. VIV is the process of verifying spoken utterances against the information stored in a given personal data profile. Though there are many differences in the performance of these approaches in various scenarios, to a first approximation, all of these can be viewed as a replacement for typed passwords to authenticate a user and determine whether the user should be allowed to access a resource. They are particularly convenient for use in applications where speech is the natural human-device interface.

There are several scenarios, however, in which reliance on a non-bypassable reference monitor to protect data is simply not possible. One common example is when data is stored on a portable device (e.g., laptop computer) that may be stolen. Once stolen, the thief may examine the disk byte-by-byte, bypassing any reference monitor that was intended to protect it (which is typically implemented as part of the file system).

For such cases, *cryptology* is often used to remove the need for a non-bypassable reference monitor. Cryptology can be used to encode data so that it is rendered

unintelligible to other than the intended recipients of it (a property referred to as *data confidentiality*). Similarly, it can be used to encode data so that any modification of it by unintended parties can be detected (*data authenticity*). In this way, an attacker who captures the encoded data—which is presumed possible since a non-bypassable reference monitor cannot be implemented—can nevertheless not make sense of the data or modify it in an undetectable way. (The attacker can, however, destroy the data, and for this reason at least, cryptography is not a fully adequate replacement for a reference monitor.)

The ability to decode encrypted data to make it intelligible, or to encode data in a way that makes it appear authentic, requires the possession of a secret value called a *key* that must be given as input to the cryptographic algorithm. In order for a cryptographic algorithm to accomplish its goals, keys must have at least the following properties:

1. Keys must be *unguessable*, in the sense that any effort that would enable the key to be discovered must be deemed outside the abilities of the presumed attacker. This implies that a key must be drawn from a distribution with sufficient entropy to render it computationally infeasible to exhaustively search (or otherwise to prevent the attacker from confirming a guess at the key, though this is typically much harder). In addition, the key must remain computationally infeasible to find in light of all the information the attacker is able to gather about it.
2. Keys must be *reproducible* by intended parties when needed to perform cryptographic operations. Since each cryptographic encoding has a corresponding decoding action, and since at least one of these two (and often both) require possession of the secret key, typically the key will need to be reproduced spatially or temporally. For example, a key may need to exist at two different computers simultaneously (a spatial reproduction) if they are interacting in a cryptographic communication protocol. A key may need to exist in the same computer at different times (a temporal reproduction), but not in the intervening period, if the key is used to encrypt and decrypt files on the computer.

## 2. Voice-generated cryptographic keys

In this paper, we advocate research into the generation of cryptographic keys from voice input. We are primarily interested in keys that can be temporally reproduced on the same device from the same user’s voice, and that are unguessable to an attacker who captures that device. This appears to be a harder problem than building a speech-based reference monitor, since a solution to our problem can be used to build a reference monitor directly: the reference monitor would take the cryptographic key derived from the voice signal as input, and compare it to what the key was supposed to be (just as a password-based login program does).

The goal of unguessability precludes perhaps the most natural approach to deriving a repeatable key from a spoken utterance: i.e., apply automatic speech recognition to recognize the password spoken, and then simply use the password as a cryptographic key. Specifically, modern ASR tools suitable for our goals can reliably recognize a vocabulary of at most about  $10^4$  words under the best circumstances. A key drawn from such a small space can, of course, be easily searched by an automated attack. Moreover, requiring the user to create her password by appending several words from the vocabulary taxes the user’s memory and yields marginal improvement in entropy, as experience with PINs has demonstrated. In contrast, to achieve unguessability, it is necessary to draw entropy from *how* the user speaks a password.

To be entirely clear, a solution to our problem would look as follows. The user would utter a password (or passphrase) to her device when prompted to do so. Using the voice input, the device would generate a sequence of bits (the key). Repeated utterance of the same password by the same user should regenerate the same key. However, an attacker who captures and dissects the device should be posed with a computationally infeasible task to recover that key. Ideally, the key should resist discovery even if the attacker knows the password, but even a solution relying on the secrecy of the password would have significant practical utility.

It is important to note that text dependent or speaker dependent models used for speaker verification—as are typical in the implementation of speech-based reference monitors—may leak significant information to an attacker who captures and dissects the device on which the model is stored. In the case of a password that is secret, a text dependent model can obviously leak information about what that password is. And, a speaker dependent model will generally leak information about the user’s relevant voice features. Consequently, the device must generate a speaker dependent cryptographic key from a speaker’s voice characteristics without referring to any text dependent or speaker dependent model. To our knowledge, this is a new challenge to speaker recognition research.

Based on this description, it should be clear that while

*false negatives* is an appropriate measure for key reproducibility, *false positives* is perhaps not the most relevant measure for the security (i.e., unguessability) of such a system. False positives does not fully capture the difficulty of the problem posed to the attacker who captures the device. Rather, measures typical in the cryptographic literature are more suitable to reasoning about the security of such a system. We describe one such measure and its use in our empirical evaluation in Section 3.3, though for completeness, we also report false positives.

## 3. Initial research

There have been a few prior efforts to generate cryptographic keys from various types of biometric data [15, 3, 7, 12], though none with particular attention to voice. In [11], we describe our initial efforts at adapting the approach of [12] to the problem of generating a cryptographic key from a vocalized password. The work of [12] described a way of generating a *hardened password*, the entropy of which was drawn from both the secrecy of the typed password and the user’s keystroke patterns (durations of keystrokes, latencies between keystrokes) while typing it. In Section 3.1 we outline the pertinent aspects of this preceding work, and then we summarize our efforts to extend this work to the voice case in Section 3.2. A brief empirical evaluation of these efforts is presented in Section 3.3.

### 3.1. Cryptographic key generation from biometrics

A biometric measurement can be summarized as a collection of *features*  $\phi_1, \dots, \phi_n$ . For example, if keystroke timings are the biometric of interest,  $\phi_1$  might denote the duration of the first keystroke,  $\phi_2$  the latency between the first and second keystrokes, and so forth. The approach of [12] for generating a cryptographic key from biometrics requires that there be a way of mapping  $\phi_1, \dots, \phi_n$  to an  $m$ -bit *feature descriptor*. Continuing with the keystroke example, the  $i$ -th bit  $b(i)$  of feature descriptor  $b$  might be obtained by comparing  $\phi_1$  to a fixed threshold and assigning  $b(i)$  to be 0 or 1 depending on whether  $\phi_1$  was less than or greater than the threshold.

Ideally, feature descriptors should separate users in the sense that descriptors produced by the same user are “sufficiently similar” (i.e., small intra-user variation), but ones produced by different users are “sufficiently different” (i.e., large inter-user variation). Indeed, if this property were satisfied, and if feature descriptors were reliably repeatable, then the feature descriptor could be a candidate for use as a cryptographic key. However, since users are generally not consistent in all features (and in fact may be consistent in relatively few), the challenge in generating cryptographic keys from biometrics lies in accommodating variations in those features in which a user is inconsistent while still generating the same key

each time. In addition, for schemes measuring biometrics during the entry of a secret password, it is generally necessary to hide *which* feature descriptor bits are consistent for the user, since this information could conceivably leak information to an attacker as to what the user’s password is. This, in turn, could be used to attack the scheme directly (as in [12]) or possibly to better predict the values of those feature descriptor bits in which the user is consistent.

The approach of [12] for achieving these goals is to hide information about which bits of a user’s feature descriptors are consistently repeatable within an  $m \times 2$  table  $T$  stored on the key generating device. During each “login”, the induced feature descriptor  $b \in \{0, 1\}^m$  is used to retrieve the  $m$  elements  $\{T(i, b(i))\}_{1 \leq i \leq m}$ . Initially, the table  $T$  is populated using a *generalized secret sharing scheme* (see [13, Chapter 12] for an introduction) so that these  $m$  elements can be used to construct the same cryptographic key, regardless of the feature descriptor  $b$ . However, as the login program identifies a bit  $b(i)$  of the user’s feature descriptors that is consistent from one login to the next—bit  $b(i)$  is said to be *distinguishing*—the  $i$ -th row of the table is perturbed so that the retrieval of  $T(i, b(i))$  is necessary to construct that cryptographic key. (A key constructed from the table can then be tested for correctness against stored information, e.g., a file encrypted under the correct key.) So, an imposter that yields a feature descriptor  $b'$  such that  $b'(i) = 1 - b(i)$  will cause the key generation to fail. Moreover, the table  $T$  is populated using a secret sharing scheme that renders it computationally costly (and ideally infeasible) for an attacker who captures the device to determine which rows have been perturbed. So, in the limit the attacker must simply guess feature descriptors  $b'$  until it finds one that reconstructs the key. Since there are  $2^m$  such feature descriptors, this becomes infeasible to search as  $m$  grows.

Due to the possibility of transient errors in the feature descriptor of the valid user (e.g., due to noise in her acoustic environment), it is generally necessary to attempt to reconstruction of the cryptographic key not only using the induced feature descriptor  $b$ , but also feature descriptors  $b'$  that are “close” to  $b$ , i.e., such that  $b'$  differs from  $b$  in a limited number of positions. We will elaborate more on this later.

### 3.2. Generating feature descriptors from voice

The steps of key generation that voice considerations impact most immediately are the identification of features  $\phi_1, \dots, \phi_n$  and the means for mapping them to feature descriptors  $b \in \{0, 1\}^m$ . Here we outline the approach that we have studied so far to achieve this; a more detailed treatment is given in [11]. While many of the component algorithms described here have been borrowed from prior work in speech and speaker recognition, to our knowledge our overall algorithm is novel.

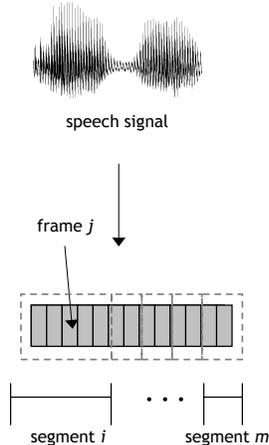


Figure 1: Segmentation

Our initial efforts into identifying suitable feature sets have utilized a representation of the user’s utterance as a sequence of frames, each of which is a 12-dimensional vector of cepstral coefficients characterizing a 30 millisecond window of the utterance (with successive windows overlapping by 10 milliseconds). We first apply endpoint detection, silence removal, and cepstrum mean subtraction [9] to the sequence, and then we use a text independent, speaker independent acoustic model (built from many different utterances from many different speakers) to segment this sequence of frames into  $m$  segments (see Figure 1). This algorithm, which is similar to *segmental vector quantization* [14, p. 382], begins from a segmentation of the frame sequence into  $m$ , roughly equal-length segments. It then iterates the following two steps until they converge on a segmentation:

1. For each segment, find the centroid in the acoustic model that yields the highest likelihood score for the segment.
2. Use the Viterbi algorithm [16, 4] to compute a new segmentation with  $m$  segments that maximizes the product of segment likelihoods relative to the centroids chosen in Step 1.

Given the segmentation so produced, we have explored three different types of features of this segmentation to generate a feature descriptor. Here we describe only the one for which we have gathered significant empirical data. To describe these features, recall that to each segment, say the  $i$ -th, is associated a “closest” *centroid*  $c_i$  in the acoustic model. Moreover, let  $\mu_i$  denote the segment mean. Then, the  $i$ -th feature  $\phi_i$  is the position of  $\mu_i$  relative to a fixed plane translated to a coordinate system with  $c_i$  at the origin (see Figure 2). That is, if  $\alpha$  is a 12-dimensional vector of coefficients specifying the plane  $\alpha \cdot x = 0$  (where  $\cdot$  denotes the dot product), then the

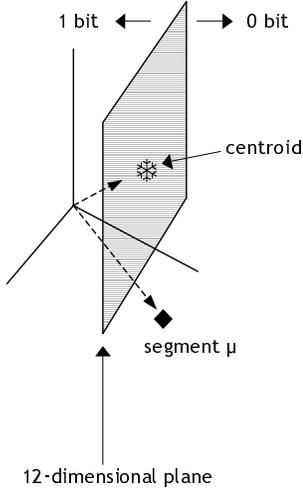


Figure 2: Creation of feature descriptor bit

$i$ -th feature is the value of  $\alpha \cdot (\mu_i - c_i)$ . If this value is less than a specified threshold (we use 0), then  $b(i) = 0$ , and otherwise  $b(i) = 1$ .

### 3.3. Empirical evaluation

In this section we briefly describe the performance of the algorithm described in Section 3.2 in selected empirical tests. Our empirical tests are based on a data set containing 90 users, each recorded during a single phone call saying the same phone number five times. The phone calls were sampled at a rate of 8 kHz. In the tests described here, feature descriptors were of length  $m = 46$ . Though  $m = 46$  is too small for even moderately secure applications—a table  $T$  with 46 rows can readily be exhaustively searched for the key by a modern personal computer—our results suggest that keys of at least this length can be achieved. Our current work is focused on demonstrating good results for larger keys (i.e., 60 bits or more).

#### 3.3.1. Measures

Traditional measures of biometric quality are false negative and false positive rates. In our context, false negatives are attempts in which the key generating device fails to regenerate the cryptographic key despite the valid user uttering her password. Similarly, false positives are instances in which the cryptographic key is regenerated even though the user providing the input is not the valid user. False positive tests can further be differentiated on the basis of whether the imposter is uttering the user’s password or something else. While good false positives in both cases is obviously desirable, we believe there is practical utility even if the password is required to be secret from the attacker.

Both of the above measures are influenced by the error correction strategy in use. As alluded in Section 3.1, the key generating device will generally attempt reconstruction of the key not only with the feature descriptor  $b$  induced by the user’s utterance, but also with feature descriptors  $b'$  that are “close” to  $b$ . In the tests described here, alternative feature descriptors  $b'$  were determined by first eliminating a single bit from  $b$  and “shifting” the remaining bits forward, and then correcting for a limited number  $d$  of additional bit errors in the shifted feature descriptor. For example, if  $m = 5$  and  $b = 01101$  is the feature descriptor induced by the user’s utterance, then some alternative feature descriptors that the login program attempts are obtained by eliminating  $b(2)$  to yield  $0\perp 101$ ; shifting the remaining bits forward to yield  $0101\perp$ ; and then generating feature descriptors of Hamming distance at most  $d$  from  $01010$  or  $01011$ . In general, the login program thus searches  $2m \sum_{i=0}^d \binom{m}{d}$  feature descriptors before returning a negative result to the user. The value of  $d$  that can be accommodated is dictated primarily by the computation time the device is allowed to spend before returning a negative answer to the user. Thus, this is dependent on both the application and the cost of performing reconstructions from the table  $T$  on that device. Here we describe results using  $d = 3$  and  $d = 4$ .

As we alluded previously, however, false positives are not the best measure of security for our schemes, since an attacker who captures the key generating device has significantly more avenues to attempt to recover the key than merely speaking the password (or potential passwords) to the device. As a result, we have focused on a different measure of security for our schemes, which we believe conservatively estimates the effort required for an attacker to recover the key even if given direct access to the table  $T$ . This measure, called *guessing entropy* [10, 2], is a measure intuitively defined as follows. (For a precise definition in our context, see [12].) Consider a game in which an attacker is presented with a table  $T$  of an unknown user selected at random from a finite population  $A$  of users. The goal of the attacker is to find the cryptographic key of this user by selecting feature descriptors  $b$ , retrieving elements  $\{T(i, b(i))\}_{1 \leq i \leq m}$  from the table, and reconstructing a value from these elements, which the attacker may then test to see if this is the right value (e.g., by using it to decrypt a file). Guessing entropy is the expected number of feature descriptors that the attacker must examine in this way before finding the key. To make this game as advantageous for the attacker as possible, we allow the attacker to have perfect knowledge of the population  $A$ , i.e., precise knowledge for each user of which of the user’s feature descriptor bits are distinguishing and, for those that are, what the typical value of that bit is. The attacker can then select feature descriptors for the candidate table in an optimal order to minimize its expected number of tries.

### 3.3.2. Results

We have evaluated the guessing entropy, false negative rate, and false positive rate of the algorithm described in Sections 3.1–3.2. These measures are illustrated in Figures 3–5 for tests involving the previously described recordings of 90 users (in “windows” of size 10; see below). Each test is parameterized by a value  $k$  (defined in [11]) that intuitively indicates the sensitivity of our scheme; i.e., smaller values of  $k$  indicate a more sensitive scheme that generally yields more distinguishing features, and larger values of  $k$  indicate a less sensitive scheme. For each value of  $k$ , planes (i.e., vectors  $\alpha$ ; see Section 3.2) were randomly sampled from among those with coefficients in  $\{-1, 0, 1\}$  and with up to five nonzero coefficients each. For each such plane, four of each user’s utterances were used to train a table  $T$  for that user (see Section 3.1), and then the guessing entropy was computed for those tables. The one remaining utterance for each user was used as a test utterance to compute a false negative rate. Twenty different utterances chosen at random from other users (i.e., an “open set” experiment) were tested against each table  $T$  to compute false positive rates. In one false positive test, denoted “FP – same” in Figures 4 and 5, these other users said the same phone number as the user who trained the table. In another test, denoted “FP – diff”, they said different phone numbers, which happen to be highly correlated to one another but not to the number used to train the table.

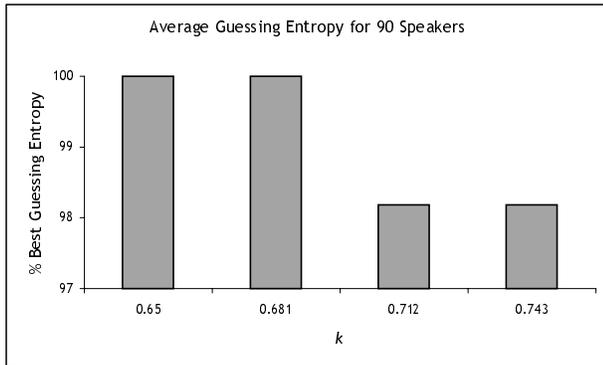


Figure 3: Guessing entropy

Due to computational cost of computing guessing entropy for a collection of tables (which grows rapidly with the number of users), we divided the 90 users in our test into windows of size 10 and performed the described test on each window of 10. The plotted points are then the average of the results for the 9 windows.

The results shown in Figures 3–5 provide evidence that our approach to key generation from voice may yield good security and reliability in practice, once properly tuned. In particular, the point  $k = 0.712$  with  $d = 3$  provides an example achieving false negatives under 9%,

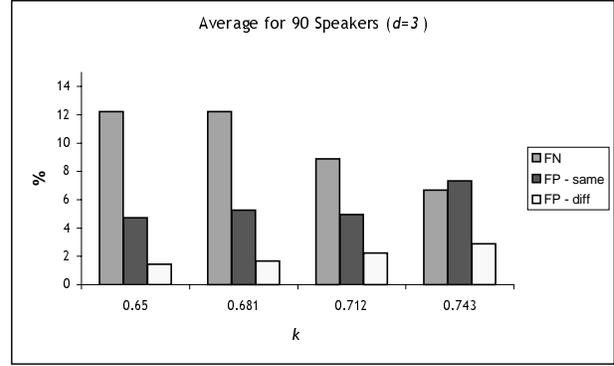


Figure 4: False negatives and positives,  $d = 3$

false positives of roughly 2% when the imposter says a different password, and entropy of roughly 98% of maximum. Since guessing entropy is already a conservative measure of security, a value of 98% of maximum still may provide a strong degree of confidence. Figure 5 demonstrates an even better data point: the point  $k = 0.65$  with  $d = 4$  yields false negatives under 2%, false positives of roughly 2% when the imposter says a different password, and 100% of the maximum entropy. Note, however, that setting  $d = 4$  expends significantly more computation than  $d = 3$  to attempt to reconstruct the key, and so may not be suitable for use with resource constrained devices.

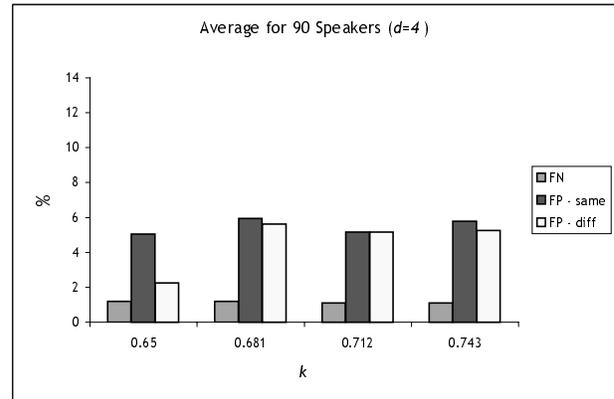


Figure 5: False negatives and positives,  $d = 4$

## 4. Future work

The initial research described in Section 3 has exposed numerous opportunities for future research in the generation of cryptographic keys from biometrics and specifically voice. An immediate direction for future work is more comprehensive testing of our present approach, using larger data sets and utterances recorded in multiple

sessions. Other directions include exploiting other voice features for our purposes. Even within the limited framework we have explored, there are other features that might be used, such as the likelihood scores of the segments into which the user's utterance is divided (see Section 3.2).

For our own work and likely for many alternatives, error correction (see Section 3.3) plays an important role in achieving a reasonable false negative rate. Since error correction requires attempting numerous alternative feature descriptors and corresponding reconstructions from the table  $T$ , it is necessary for reconstructions to be very efficient. One direction of our continuing research is the development of secret sharing schemes for which reconstruction is very efficient, thereby enabling broader error correction (but possibly also more efficient attacks).

Finally, another topic for future work is to tune the particular parameters of our approach with an eye toward deployment in practice. In order to make use of our approach, particular values of  $\alpha$  and  $k$  must be chosen for our scheme. Principal component analysis (e.g., see [6]) may be of use in identifying vectors  $\alpha$  that best separate users, but we have thus far had little success in applying these techniques where guessing entropy is the measure to optimize. We hope to explore this more closely in the future.

## 5. References

- [1] B. S. Atal. Automatic recognition of speakers from their voices. *Proceedings of the IEEE* 64:460–475, 1976.
- [2] C. Cachin. *Entropy Measures and Unconditional Security in Cryptography*. ETH Series in Information Security and Cryptography, Volume 1, Hartung-Gore Verlag Konstanz, 1997.
- [3] G. I. Davida, Y. Frankel, and B. J. Matt. On enabling secure applications through off-line biometric identification. In *Proceedings of the 1998 IEEE Symposium on Security and Privacy*, pages 148–157, May 1998.
- [4] G. D. Forney, Jr. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, March 1973.
- [5] D. Gollman. *Computer Security*. John Wiley & Sons, 1999.
- [6] R. A. Johson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Fourth edition, Prentice Hall, 1998.
- [7] A. Juels and M. Wattenberg. A fuzzy commitment scheme. In *Proceedings of the 6<sup>th</sup> ACM Conference on Computer and Communication Security*, pages 28–36, November 1999.
- [8] Q. Li, B.-H. Juang, Q. Zhou and C.-H. Lee. Automatic verbal information verification for user authentication. *IEEE Transactions on Speech and Audio Processing* 8(5):585–596, September 2000.
- [9] Q. Li and A. Tsai. A matched filter approach to endpoint detection for robust speaker verification. In *Proceedings of IEEE Workshop on Automatic Identification Advanced Technologies (AutoID'99)*, pages 35–38, October 1999.
- [10] J. L. Massey. Guessing and entropy. In *Proceedings of the 1994 IEEE International Symposium on Information Theory*, page 204, 1994.
- [11] F. Monrose, M. K. Reiter, Q. Li and S. Wetzel. Cryptographic key generation from voice. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, May 2001.
- [12] F. Monrose, M. K. Reiter, and S. Wetzel. Password hardening based on keystroke dynamics. In *Proceedings of the 6<sup>th</sup> ACM Conference on Computer and Communications Security*, pages 73–82, November 1999.
- [13] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
- [14] L. Rabiner, B. Juang, and B. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.
- [15] C. Soutar and G. J. Tomko. Secure private key generation using a fingerprint. In *Cardtech/Securetech Conference Proceedings*, Volume 1, pages 245–252, May 1996.
- [16] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13:260–269, April 1967.