Cryptographic Key Generation from Voice
(Extended Abstract)

Fabian Monrose Michael K. Reiter Qi Li Susanne Wetzel

Bell Labs, Lucent Technologies, Murray Hill, New Jersey, USA
{fabian,reiter,qli,sgwetzel}@research.bell-labs.com

Abstract

We propose a technique to reliably generate a cryptographic key from a user's voice while speaking a password. The key resists cryptanalysis even against an attacker who captures all system information related to generating or verifying the cryptographic key. Moreover, the technique is sufficiently robust to enable the user to reliably regenerate the key by uttering her password again. We describe an empirical evaluation of this technique using 250 *utterances recorded from* 50 *users.*

1. Introduction

The inability of human users to remember strong cryptographic keys has been a factor limiting the security of systems for decades. History has proved that users can remember only short passwords, and even then tend to choose passwords that are easily guessed by dictionary attacks (e.g., see [16, 10, 7, 22, 27]). This limitation could be addressed in a wide range of applications by generating strong cryptographic keys from biometric data, possibly in conjunction with the entry of a password [24, 3, 9, 15]. This approach is attractive since it imposes no additional memorization on the user and yet can yield keys significantly stronger than passwords. These keys could then be used in a wide variety of applications, including file encryption, access to virtual private networks, and user authentication. The primary difficulty in this approach lies in accommodating the variations inherent in measuring biometrics, or in the biometrics themselves, while repeatedly generating the same key. In general, this requires a technique designed for the particular biometric being used.

In this extended abstract, we describe the first scheme (to our knowledge) for generating cryptographic keys from a spoken password, i.e., for generating a cryptographic key based on *the voice characteristics* of the user's spoken password. We approach the problem starting from the scheme of Monrose, *et al.* [15], which was initially developed with the goal of utilizing keystroke timings in the generation of a strong cryptographic key from a password. Consideration of a *spoken* password, however, introduces complexities not found in the keystroke case. First, a password typed is unambiguous, whereas a password spoken is not: automatic speech recognition (ASR) remains an active research area despite over 40 years of investigation. Second, modern keyboards offer only a single dimension on which keystrokes can be measured, namely time. In contrast, numerous characterizations of voice features have been explored in speech processing for ASR, speaker verification, and related fields. It is not obvious how to best characterize voice for the task of generating a cryptographic key, and indeed identifying one effective characterization is a contribution of this work.

Despite these challenges, in this extended abstract we adapt the approach of [15] to the task of generating a key from a spoken password; we call this key a *derived key*. In our construction we retain the initial goals of [15], namely that the derived key remain difficult to break even against an attacker that captures all stored information related to generating or verifying it. This maximizes the number of applications in which our techniques can be applied.

1.1. Why voice?

Numerous biometrics have been proposed for user authentication (see `www.biometrics.org`), and conceivably many are candidates for generating cryptographic keys using recently proposed techniques. We choose to study voice for several reasons. First, it is a familiar way of communicating, which makes it ideal for many applications. For example, voice activated phones, dictation devices, and voice dialing are becoming increasingly prominent in the marketplace, and such devices are natural candidates for applying the work described here. Second, work in speaker verification has shown voice to be effective for dis-

tinguishing between users (e.g., [11]). Third, when a person changes her password, she unavoidably changes the vocalization of her password. Thus, unlike with static biometrics (e.g., fingerprints, irises), it is conceivable that a user could have arbitrarily many derived keys over her lifetime.

Voice does come with limitations, however. Probably the most cogent is the risk that a user could be recorded while speaking her password, thereby effectively leaking her keying material. Most other biometrics are vulnerable to a similar risk just as voice is: a camera can photograph an iris from across the room, and fingerprints left on surfaces can be lifted hours later.[1] Nevertheless, recognizing this limitation, we emphasize that the primary protection our method offers is against an attacker who does not have the opportunity to record the user speaking her password. This categorizes many attacks, such as when the attacker steals the device utilizing our techniques from an empty hotel room, or files encrypted under our derived keys over a network.

1.2. Criteria

The high level criteria we adopt for the generation of a derived key incorporates both security and usability requirements. Our usability requirement is simply that the device, upon receiving its user's utterance of a password (this password can be different per user), successfully generate the user's derived key. This criterion need not be absolute, since the key generation process will be affected by the user's acoustic environment; e.g., clearly a blow horn sounding in the background can destroy any possibility that the device might successfully generate the key. However, when the user utters the password close to the device in an office environment—i.e., in an environment yielding a reasonable signal-to-noise ratio—the device should reliably regenerate the user's derived key without excessive delay.

As indicated previously, our security goals anticipate the physical capture of the key generating device as the primary attack with which we are concerned. Since we make no assumptions regarding tamper-resistance of the device, we assume that an attacker, upon capturing the device, has full access to any data stored within the device. Despite this, the key should still resist cryptanalysis by the attacker. This requirement renders prior literature on speaker verification inadequate to achieve our goals: speaker verification technology generally utilizes plaintext models of the user's speech that, in our setting, would leak information about the user's vocal patterns (i.e., the user's keying material). In contrast, here we adopt the requirement that *no* plaintext information regarding the user's prior utterances of her password be recorded in the device. This makes the repeated generation of the derived key particularly challenging.

Of course, an attacker who captures the device can always cryptanalyze the key by a brute force attack, given sufficient resources. As this work is only an initial investigation into the generation of keys from voice, here we empirically demonstrate the generation of 46-bit keys from a roughly two second spoken password. These 46-bit keys can be easily lengthened to, e.g., 56-bit DES keys using known salting techniques, with a moderate additional cost in login delay [13]. Future work will focus on the generation of longer keys. However, the degree to which a derived key resists a brute force attack depends primarily on the entropy in user speech as captured by the ways we measure it. Here we give evidence using empirical data that the measures we choose plausibly yield significant entropy in the derived key, even among users speaking the same password. However, since our empirical trials involve limited populations of users, we cannot quantify the entropy of derived keys from a large population.

Our goal of extracting entropy from how a user speaks a password, as opposed to only the choice of password itself, differentiates our work from perhaps the most natural approach to deriving a cryptographic key from a spoken utterance: i.e., apply ASR to recognize the password spoken, and then simply use the password as a cryptographic key. We have eliminated this approach from consideration for several reasons, however. First and foremost, modern ASR tools suitable for our goals can reliably recognize a vocabulary of at most about 10^4 words under the best circumstances. This quantity can, of course, be easily searched by an attacker. 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, our work strives to draw entropy from *how* the user speaks a password, which does not impose greater memorization on the user. Second, modern ASR techniques tend to require greater volatile storage from the device than the techniques proposed here, which renders ASR less attractive for some of our target platforms.

1.3. Applications

The most immediate application for a strong key that can be generated from a spoken password is for secure telephony. In one scenario, the user would speak a password to generate her derived key, which would serve as the seed to a pseudorandom process to generate the private key of her public key pair. This private key would be used to de-

[1] In user authentication applications, techniques exist to make the presentation of such "stolen" biometrics to biometric readers more difficult. For example, fingerprint readers typically verify indications of blood flow through the object in contact with the reader. For our goals, however, these defenses are useless: once a biometric measurement is stolen and so the keying material within that biometric is available to the attacker, the attacker can derive the key and attack, e.g., encrypted files to which he has access—completely bypassing the biometric reader.

crypt incoming voice (or more likely, a symmetric key under which the incoming voice is encrypted). This is similar to voice protections as offered by secure phones such as the STU-III, though our techniques offer the advantage that the user need not carry a physical token with keying material to enable the device; rather, the keying material is the user's voice. Similarly, applications like PGPfone (`web.mit.edu/network/pgpfone/`) and Speak Freely (`www.speakfreely.org/`) require entry of a typed password to activate a stored key; this is inconvenient on a phone keypad and will typically be more susceptible to a dictionary attack if the device is captured.

In a variation of secure telephony, our techniques enable the decryption of encrypted *email* in a phone. Several text-to-speech systems exist today—see `morph.ldc.upenn.edu/ltts/` for a list—and a central application of these is to enable users to check their email over the phone (e.g., [23]). Our technique would enable the "reading" of *encrypted* email over the phone: again, the derived key could be used to generate the user's private key, which would be used to decrypt an email. The email could then be rendered to the user by a text-to-speech tool. As in secure telephony, the generation of her private key, and the decryption and rendering of her email, would be performed completely within the telephone. In this way, the plaintext of the email would never be exposed outside the phone, and even an attacker who captured the phone would be unable to decrypt messages intended for the legitimate user.

In a similar way, a dictation device could save its stored dictations encrypted under the derived key. In order to decrypt a dictation, a user would utter her password, thereby enabling the creation of the derived key and decryption of the dictation. Again, even an attacker who stole and disassembled the dictation device would be unable to decrypt the dictations.

We note that at the time of this writing, the algorithms in this extended abstract stretch the computational limitations of current cellular phones and dictation devices. However, voice-enabled personal digital assistants (PDAs) with powerful processors (e.g., 400–500 MHz Intel StrongArm 2) will become available soon [21]. With such rapid advances in user devices, the computation required by the algorithms we propose will soon be within reach for PDAs and mobile phones.

2. Background

In this section we review the techniques in cryptographic key generation from biometrics and in automatic speech processing that are necessary to understand the balance of this extended abstract.

2.1. Cryptographic key generation from biometrics

For the purposes of this extended abstract, known methods for generating cryptographic keys from biometric measurements can be characterized as having two "stages". In the first stage, certain *features* $\phi_1, \phi_2, \ldots$ of raw input from a biometric-measuring device are examined and used to compute an m-bit string called a *feature descriptor*. Feature descriptors should separate users in the sense that descriptors produced by the same user are "sufficiently similar" (i.e., small intra-user variation), whereas ones produced by different users are "sufficiently different" (i.e., large inter-user variation). This separating property is then magnified in the second stage, which develops a cryptographic key from the feature descriptor and stored cryptographic data in the device. If two descriptors are sufficiently similar, then the same cryptographic key will be generated from them. Whereas the second stage is largely independent of the biometric being used, the first is not.

An example of such a mechanism is described in [15]. This work develops a cryptographic key, called a *hardened password*, from a text password and the keystroke dynamics of the user while typing it. In the first stage, the features of interest are the duration of each keystroke and the latency between each pair of keystrokes, yielding features $\phi_1, \ldots, \phi_{15}$ (8 durations, 7 latencies) for an 8-character password. These features are processed to obtain a feature descriptor $b \in \{0,1\}^m$, $m = 15$, by comparing each measured duration or latency to a fixed threshold: if the measured time ϕ_i for the i-th feature is less than the threshold, then the i-th bit $b(i)$ of the feature descriptor b is set to $b(i) = 0$, else it is set to $b(i) = 1$.

In the second stage, elements $\{T(i, b(i))\}_{1 \le i \le m}$ are retrieved from a $m \times 2$ table T on disk. Initially, each entry of this table contains a *share* of the hardened password as produced by a secret sharing scheme. When retrieved, these shares are used to reconstruct the hardened password. As the timing measurements encountered during successful logins are gathered over time (in a file encrypted under the hardened password), patterns in them are distilled and used to "harden" the table by perturbing entries that the legitimate user does not typically use. More precisely, let the mean and standard deviation of ϕ_i over the last h successful logins (for some parameter h) be μ_i and σ_i, respectively. Consider the *partial* feature descriptor B defined by:

$$B(i) = \begin{cases} 0 & \text{if } \mu_i + k\sigma_i < t \\ 1 & \text{if } \mu_i - k\sigma_i > t \\ \perp & \text{otherwise} \end{cases}$$

for some threshold t. That is, $B(i)$ is 0 (resp., 1) if μ_i is sufficiently less than (resp., greater than) t; otherwise, $B(i)$ is undefined. Then, the table T is constructed so that $T(i, j)$ contains a valid share of the hardened password if

and only if $B(i) = j$ or $B(i) = \perp$. Entries $T(i, j)$ where $B(i) = 1 - j$ are filled with random elements. These entries should have no effect on the correct user, since her typing will typically select a valid share of the hardened password from each row, and the login program can then reconstruct the hardened password (correcting for a few errors if necessary). Security is enhanced in this scheme by encrypting the table T under the text password. By its design, the table T appears indistinguishable from a table that would result from decrypting it with the wrong password. Thus, even if an attacker captures the device, any attack against the table (i.e., the hardened password) appears to require at least the same resources as a dictionary attack against the text password and possibly up to 2^{15} times as much, due to the presence of random elements even in the properly decrypted table.

Other techniques fitting this "two stage" structure have been proposed for generating cryptographic keys from biometrics. Davida, *et al.* [3] describe a different second-stage strategy using error-correcting codes and how it could be conjoined with first-stage approaches for generating feature descriptors from iris scans [4, 26]. Juels and Wattenburg [9] subsequently improved the second-stage approach of Davida, *et al.* significantly. Soutar and Tomko [24] describe a different approach for generating cryptographic keys from fingerprints using optical computing techniques.

The focus in this extended abstract is on the generation of appropriate feature descriptors from speech; i.e., we focus here on the first stage of the key generation process. In principle, our techniques could be conjoined with any of several second-stage techniques to generate cryptographic keys. However, we evaluate our scheme primarily with the table-based technique of [15] in mind.

2.2. Automatic speech processing

Automatic speech recognition (ASR) and speaker verification (SV) have been topics of active research since the 1950s. Here we introduce, in a very simplistic manner, only what is necessary for understanding this extended abstract. The reader interested in learning more is referred to any of numerous surveys in the area (e.g., [1, 6, 17, 20]). A particularly accessible text is due to Rodman [19], from which our presentation borrows liberally.

We begin with a brief introduction into how sound is received and represented by a computer. A microphone contains a diaphragm that vibrates in concert with any sound whose frequencies are within its range of operation. The microphone converts these vibrations into an electrical signal. This signal is sampled periodically (8 KHz in our experiments described in Section 4) by an analog-to-digital converter, to produce a sequence of values $A_{\mathsf{time}}(1)$, $A_{\mathsf{time}}(2)$, ..., $A_{\mathsf{time}}(k)$ representing amplitudes. This sequence represents amplitude as a function of time, and is often referred to as a *time domain* representation of sound. Speech can be approximated in the time domain as a Fourier series, i.e., as sums and differences of sine and cosine waves of varying periods, amplitudes, and phases. These sine and cosine waves represent the various frequency components of the signal. Plotting the average amplitude of each of these component frequencies in a small window of time gives the *frequency domain* representation A_{freq}, or *spectrum*, of the signal in that window. Transforming from the time domain to the frequency domain and back are accomplished by the *discrete Fourier transform* (DFT) and *inverse discrete Fourier transform* (IDFT), respectively; see [5] for details.

A spectrum A_{freq} of speech can be viewed as the product of a quickly varying mapping A_1 and a slowly varying one A_2, i.e., $|A_{\mathsf{freq}}(f)| = |A_1(f)| \times |A_2(f)|$. A_1 results from vocal cord vibration; A_2 results from the movement of articulators (the teeth, alveolar ridge, hard palate, and velum). To see this effect in the time domain, the IDFT is applied to $\log(|A_{\mathsf{freq}}(f)|) = \log(|A_1(f)|) + \log(|A_2(f)|)$. This yields a "spectrum like" representation $\psi(t)$, called a *cepstrum*, in which the slowly varying part (the effect of articulators) is represented by lower positions on the time axis and the quickly varying part (vocal cord excitation) is represented by higher positions. The separation in the cepstrum of the effects of articulators from vocal cord excitation is useful because the corresponding values can be used to model the vocal tract of the user. In particular, $\psi(1), \ldots, \psi(12)$, called *cepstral coefficients*, have been shown to be useful for this purpose. Specifically, in our experiments, cepstral analysis is performed on overlapping 30 msec windows of an utterance, each shifted by 10 msec from the previous, to yield a vector $\langle \psi(1), \ldots \psi(12) \rangle$ for that 30 msec window. This vector is called a *frame*, and an utterance is represented by a sequence of frames, one per window.

Many explorations in ASR and SV involve comparing the frames of an utterance to some *acoustic model* of speech, itself derived from frames of spoken utterances. An acoustic model can be constructed from the speech of few users (typically one), in which case the model is *speaker dependent*, or it may be *speaker independent* in that it incorporates speech from a large population. Orthogonally, the model may be based on utterances of the same word(s) against which the model will later be compared (a *text dependent* model) or based on utterances of words unrelated to those against which it will be compared (a *text independent* model).

Our goals necessitate the use of a speaker independent, text independent model; storing a speaker dependent or text dependent model could conceivably leak keying information to an attacker who captures the device that generates the derived key. Thus, in our experiments we em-

ploy a large model derived from a speech database containing various utterances from many different users. Clearly any model that is suitable for storage on a small device must significantly compress this data, and the means by which we do this is *vector quantization* (VQ). VQ is a general technique that, given a collection of vectors (in our case, frames), identifies clusters of vectors and partitions the acoustic space according to these clusters, i.e., with each block of the partition containing one cluster. The vectors in each cluster are then used to generate a multivariate normal distribution for that cluster, parameterized by the vector c of component-wise means (called a *centroid*) and the covariance matrix Σ for the vectors in the cluster.[2] The density function for this distribution is

$$P_{c,\Sigma}(x) = \frac{1}{(2\pi)^{\delta/2}\sqrt{\det(\Sigma)}} e^{-(x-c)^T\Sigma^{-1}(x-c)/2}$$

where δ is the dimension of the vectors, which again is $\delta = 12$ in our setting. For notational simplicity, we abbreviate $P_{c,\Sigma}(x)$ by $P(x \mid c)$, in particular omitting Σ since it does not play an explicit role in the balance of this extended abstract. Moreover, we denote the set of centroids by C. A collection of frames can be compressed in this way to any chosen number of such distributions and associated centroids. In Section 4, we present results based on the use of 20 centroids (i.e., $|C| = 20$).

3. Algorithms

Equipped with the background described in Section 2, here we give an overview of the methodology we use to derive a cryptographic key from a spoken password. The process is illustrated in Figure 1. Our approach begins with the steps described in Section 2.2 for capturing the speaker's utterance (1.a), dividing the voice samples of the utterance into 30 msec windows overlapping by 10 msec each (1.b), and deriving a frame of 12 cepstral coefficients from each window (1.c). We further discard frames corresponding to silence before, within, and after the utterance (e.g., using [12]). In our experience, this step is critical to achieving good results.

The high level goal after this processing is to construct an m-bit feature descriptor (1.h) from this sequence of frames. As described in Section 2.1, the resulting feature descriptor will be used to index into a table (1.i) to retrieve elements of that table. These elements can be used to reconstruct the derived key (1.j) provided that the feature descriptor is sufficiently close to feature descriptors generated in previous successful attempts. The remaining challenge is how to generate a feature descriptor (1.h) from frames in a way that will reliably reconstruct the derived key, possibly after correcting a few bits in the feature descriptor.

Our proposals for constructing a feature descriptor from the frame sequence begin by *segmenting* the sequence of frames into contiguous subsequences, or *segments*. Intuitively, segmenting the sequence best divides the sequence into component sounds. To determine what constitutes a "component sound", the frame sequence is matched against a speaker-independent, text-independent model of speech (1.d), quantized into 20 different centroids and associate multivariate normal distributions as described in Section 2.2. Our segmentation is an iterative process (1.e), converging to a near-optimal segmentation of the user's utterance (1.f). We then consider various features of the resulting segmentation, and generate one feature descriptor bit from each. The feature that we empirically evaluate in Section 4, which is depicted in Figure 1, is the relationship of the point determined by averaging the frames in a segment to a plane through the closest matching centroid. In the following subsections, we describe the details of these steps.

3.1. Segmenting the frames

All of the approaches we discuss for generating a feature descriptor begin with *segmenting* the utterance using the acoustic model. More precisely, let $f(1) \ldots f(n)$ be the sequence of frames from the utterance, and let $R_1 \ldots R_s$ be disjoint, nonempty ranges over the natural numbers such that $\bigcup_{i=1}^{s} R_i = [1, n]$. $R_1 \ldots R_s$ naturally segment $f(1) \ldots f(n)$ into s segments, where the i-th segment is $f(j) \ldots f(j')$ if $R_i = [j, j']$; for simplicity, we denote the i-th segment by $F(R_i)$. Intuitively, a segmentation is a partition of the sequence of frames $f(1) \ldots f(n)$ into subsequences $F(R_1) \ldots F(R_s)$ where each $F(R_i)$ corresponds to one "component sound" of the user's utterance.

Segmentation is performed using the acoustic model described in Section 2.2. For a segment $F(R)$, let

$$L(F(R) \mid c) = \prod_{i \in R} P(f(i) \mid c).$$

This is called the *likelihood* of the segment relative to the centroid c. For a segment $F(R)$, the *matching centroid* for that segment is

$$c(R) = \arg\max_{c \in C} \{L(F(R) \mid c)\}, \quad (1)$$

i.e., the centroid c for which $L(F(R) \mid c)$ is the largest. Our goal, then, is to generate $R_1 \ldots R_s$ that maximizes

$$\prod_{i=1}^{s} L(F(R_i) \mid c(R_i)). \quad (2)$$

[2] More precisely, Σ is derived from all frames in the model, not only those in the cluster. This is standard practice in cases such as ours, in which we do not have enough data to generate a reliable covariance matrix per cluster.

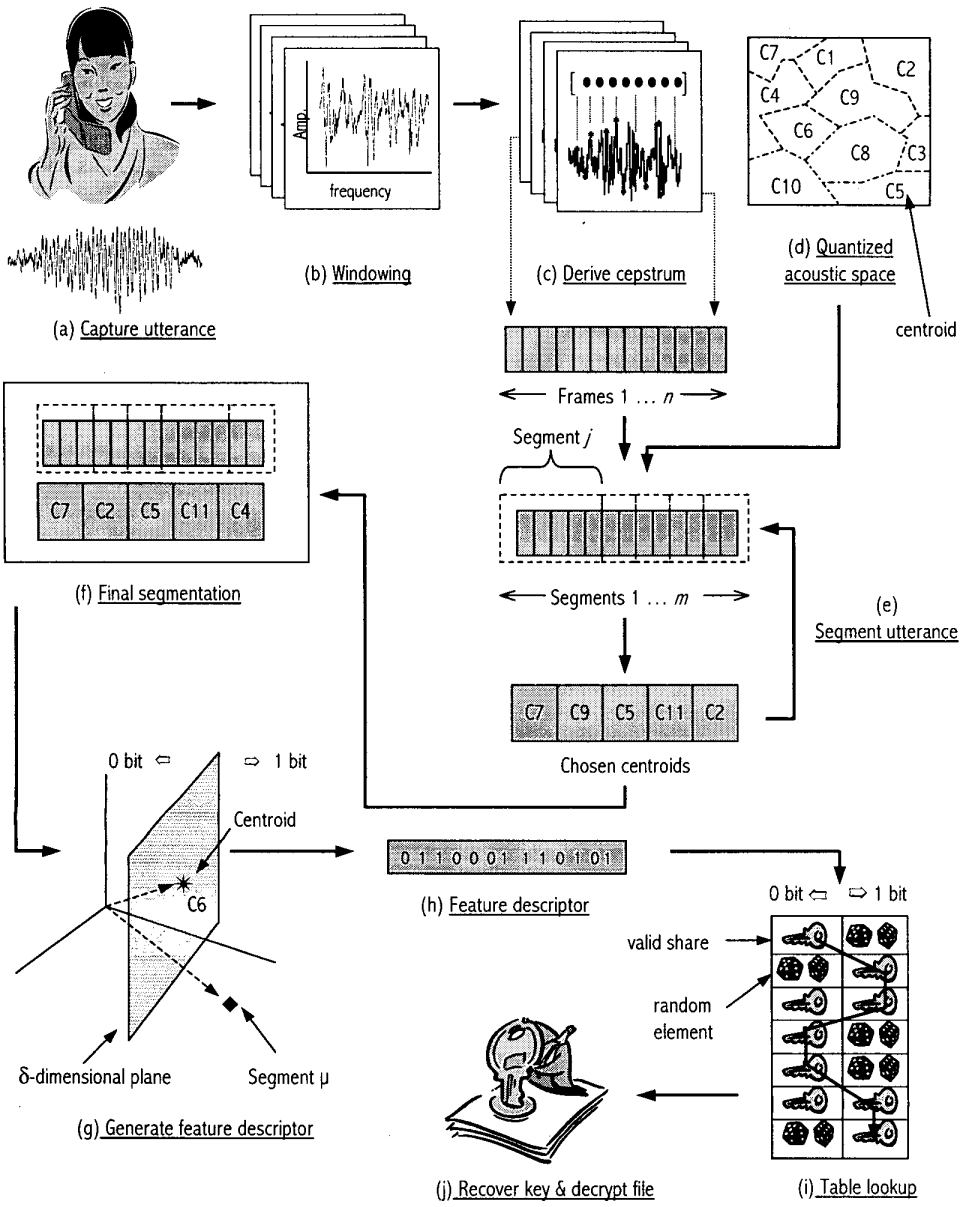

Figure 1. Key generation from voice features

Intuitively, $F(R_1)\ldots F(R_s)$ is an optimal segmentation in that it best maps the input utterance $f(1)\ldots f(n)$ into s component centroids.

The algorithm we employ for optimally segmenting the utterance employs an iterative approach similar to that of "segmental VQ" algorithms (e.g., see [18, p. 382]). The algorithm initializes ranges $R_1 \ldots R_s$ to partition $[1, n]$ into roughly equal-length ranges. The algorithm then repeatedly refines these ranges until subsequent refinements yield small improvement (less than some parameter Δ) in the segmentation they induce (i.e., in (2)). More precisely, the algorithm executes as follows:

i. (Initialization) Assign score $\leftarrow 0$. For $i = 1, \ldots, s$: assign $R_i \leftarrow \left(\lfloor \frac{(i-1)n}{s} \rfloor, \lfloor \frac{in}{s} \rfloor\right]$.

ii. For $i = 1, \ldots, s$: determine $c(R_i)$ as the matching centroid for $F(R_i)$ using (1).

iii. Compute new nonempty, disjoint ranges $R'_1 \ldots R'_s$, $\bigcup_{i=1}^{s} R'_i = [1, n]$, that maximize

$$\text{score}' = \prod_{i=1}^{s} L(F(R'_i) \mid c(R_i)).$$

This step can be performed efficiently using the Viterbi algorithm [25, 8].

iv. Assign $R_i \leftarrow R'_i$ for $i = 1 \ldots s$. If score$'$ − score $> \Delta$, then assign score $\leftarrow$ score$'$ and go to step (ii).

The segmentation $F(R_1)\ldots F(R_s)$ when this algorithm completes is the result, which is input to the algorithms in Section 3.2. In practice, the above algorithm typically terminates within 3 iterations for reasonable values of Δ. We also use heuristics within this algorithm to improve the resulting segmentation. For example, when this algorithm converges, if one of the resulting segments $F(R_i)$ contains only a single frame, then that frame is discarded from the frame sequence (since it presumably resulted from noise in the user's acoustic environment) and the optimal segmentation is recalculated from scratch.

3.2. Mapping segments to features

Having derived a segmentation with s segments, our goal now is to define features of this segmentation that are typically the same when the same user speaks the same utterance. From these features, an m-bit *feature descriptor* is then derived. The approaches introduced here isolate one feature in each segment and generate one descriptor bit from each feature, i.e., $s = m$, though in general this is not necessary; e.g., multiple bits could be derived from each segment so that $m > s$. Note that in principle, this algorithm must not exploit plaintext information derived from the user's previous utterances. Otherwise, this information might also be exploited by an attacker who captures the device performing this computation.

There are several features that one might (and we did) try to generate a feature descriptor b from the segmentation $F(R_1)\ldots F(R_s)$. The first approach was possibly the most straightforward:

I. *Centroid parity.* In this approach, the i-th feature ϕ_i is the "parity" of centroid $c(R_i)$. Specifically, half the centroids in the acoustic space are chosen at random and assigned a "parity" of 0; the other half are assigned a parity of 1. Then, $b(i)$ is the parity of $c(R_i)$.

The second and third features we mention in this extended abstract compare the *segment average* to the matching centroid for the segment. For a segment $F(R_i)$, the segment average is the vector $\mu(R_i)$ of component-wise means of the frames in $F(R_i)$.

II. *Distance from centroid.* The i-th feature ϕ_i is the likelihood $L(\mu(R_i) \mid c(R_i))$. To map these features to a feature descriptor, we fix a threshold t and define

$$b(i) = \begin{cases} 0 & \text{if } L(\mu(R_i) \mid c(R_i)) < t \\ 1 & \text{otherwise} \end{cases}$$

III. *Position relative to a plane through the centroid.* Given a fixed vector α, the i-th feature is $\phi_i = \alpha \cdot (\mu(R_i) - c(R_i))$. That is, normalize $\mu(R_i)$ to a coordinate system with $c(R_i)$ at the origin (i.e., $\mu(R_i) - c(R_i)$), and then let ϕ_i be the linear combination of components in $\mu(R_i) - c(R_i)$ as specified by α. To map these features to a feature descriptor, we simply test whether each feature is positive or negative.

$$b(i) = \begin{cases} 0 & \text{if } \alpha \cdot (\mu(R_i) - c(R_i)) < 0 \\ 1 & \text{otherwise} \end{cases}$$

The value $b(i)$ represents the "position" of $\mu(R_i)$ relative to the plane $\alpha \cdot x = 0$ translated to a coordinate space whose origin is $c(R_i)$: $b(i) = 0$ can be interpreted as $\mu(R_i)$ falling on one side of this plane, while $b(i) = 1$ indicates that it falls on the other (or on the plane itself).

In this extended abstract, we report results only for Algorithm III. We choose to report results for this algorithm for two reasons. First, we have found that it is competitive with the other approaches in terms of the results it yields. Second, since it has the most degrees of freedom in its design (the 12 elements of α), it is the most challenging of these three approaches to analyze to draw conclusions as to its efficacy.

4. Evaluation

In this section we describe our empirical evaluation of Algorithm III described in Section 3.2. In Sections 4.1 and 4.2, we define the measures we use to evaluate our proposal. We describe our experimental methodology in Section 4.3, and our results in Section 4.4.

4.1. Guessing entropy

In order to evaluate the security of these proposals, we first revisit the scheme of Monrose, *et al.* [15] to see in more detail how feature descriptors are used. As described in Section 2.1, the scheme maintains an $m \times 2$ table T such that $T(i, j)$ is a valid share of the hardened password (in our case, the derived key), if and only if $B(i) = j$ or $B(i) = \perp$. An attacker who captures T can reconstruct the derived key if he can find a feature descriptor b that is consistent with B, i.e., so that if $B(i) \neq \perp$ then $b(i) = B(i)$. The attacker can then reconstruct the derived key from shares $T(i, b(i))$, $1 \leq i \leq m$.

Let $\{B_a\}_{a \in A}$ be partial feature descriptors for the users in some population A, and let $\{T_a\}_{a \in A}$ be resulting tables for these users. The security of this scheme is characterized by *guessing entropy* [14, 2]. Intuitively, consider a game in which an attacker is given a table T_a for an unknown account a drawn uniformly at random from A. Guessing entropy captures the expected number of (total) feature descriptors that the attacker guesses until he finds a feature descriptor b such that $\{T(i, b(i))\}_{1 \leq i \leq m}$ successfully reconstructs the key for the account (i.e., such that $\{T(i, b(i))\}_{1 \leq i \leq m}$ consists only of valid shares).[3] To make the game as advantageous for the attacker as possible, the attacker is presumed to have perfect knowledge of the set $\{B_a\}_{a \in A}$. Note that the best possible guessing entropy is thus $\min\{2^m, (|A| + 1)/2\}$, but may be considerably less if, say, a single feature descriptor b is consistent with B_a for many accounts a.

To precisely capture the attacker's knowledge, we define a *cover* to be a function $\mathcal{C}$ from A to total feature descriptors such that $\mathcal{C}(a)$ is consistent with B_a. Let $\mathrm{Img}(\mathcal{C}) = \{b \mid \exists a \in A : \mathcal{C}(a) = b\}$, and $w_{\mathcal{C}}(b) = |\{a \in A \mid \mathcal{C}(a) = b\}|/|A|$. If we denote $\mathrm{Img}(\mathcal{C}) = \{b_1, \ldots, b_\ell\}$ such that $w_{\mathcal{C}}(b_1) \geq w_{\mathcal{C}}(b_2) \geq \ldots \geq w_{\mathcal{C}}(b_\ell)$, then the guessing entropy of the cover $\mathcal{C}$ is

$$E_{\mathcal{C}} = \sum_{i=1}^{|\mathrm{Img}(\mathcal{C})|} (i \cdot w_{\mathcal{C}}(b_i)) \qquad (3)$$

Then, assuming "perfect knowledge" for the attacker means that we compute the guessing entropy using a cover that minimizes (3).

[3] As discussed in [15], guessing entropy is a more accurate measure of the security of this scheme than Shannon entropy, due to the fact that each B_a may have undefined values. For example, suppose that $|A| = m$, that each B_a has exactly one defined value, and that $B_a(i) \neq \perp$ implies $B_{a'}(i) = \perp$ for any $a' \neq a$. Then, the Shannon entropy of a randomly chosen account a's feature descriptor would seem to be at least $\log m$, due to the uncertainty in the position i such that $B_a(i) \neq \perp$. Nevertheless, an attacker need only attempt to reconstruct using at most two different (total) feature descriptors, e.g., b such that $b(i) = 0$ for each $1 \leq i \leq m$, and b such that $b(i) = 1$ for each $1 \leq i \leq m$. Thus, the guessing entropy is at most 1.5.

4.2. False negative rate

The other measure that we use to evaluate our scheme is the *false negative* rate. The false negative rate is the percentage of failures among attempts by the legitimate user to generate her derived key, averaged over the users in a population A. For a given account, consider the total feature descriptor b derived from the legitimate user's utterance of her (correct) password. The login program will first attempt to reconstruct her derived key using $\{T(i, b(i))\}_{1 \leq i \leq m}$, but if that is unsuccessful, it may also try other feature descriptors b' that are "close" to b. More precisely, the alternative feature descriptors b' attempted by the login program are those derived by various error correcting strategies. So far, the best strategies we have identified involve 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 $0\perp101$; 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 $2m\binom{m}{d}$ 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 login program 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.

The method of generating alternative feature descriptors b' described above is motivated by the intuition that a sound in the user's acoustic environment could cause the introduction of a segment of frames dominated by that sound. By eliminating the corresponding bit and shifting the remaining bits forward, we recover a feature descriptor that is not skewed by that sound. Even with this correction, however, false negatives can result from sources such as poor microphone quality, the user's distance to the microphone, inconsistency in pronunciation, or other background noise.

A noteworthy omission from our evaluation is the computation of a *false positive* rate. Interpreted in our setting,

this rate would be the percentage of utterances (possibly of the correct password) by users other than the correct user that nevertheless yield a proper reconstruction of the user's derived key. Like guessing entropy, a false positive rate measures the degree to which our techniques differentiate users. However, guessing entropy more directly characterizes the difficulty faced by the attacker with which we are concerned, i.e., who has direct access to the device on which the table T_a is stored but who has not heard or recorded the user's utterance of her password. We thus believe it provides a more meaningful (albeit pessimistic) measure of the security of our scheme.

4.3. Experimental methodology

The methodology adopted for our experiment was as follows. We employed a preexisting dataset consisting of recordings of five utterances of the phone number "1 800 882 3606" by different users of varying nationalities and both sexes. The recording device recorded each user's utterances over a phone call, where each user called from a different telephone device (and of course over a different phone connection). These recordings are thus of poorer quality, and of more varying qualities, than we might expect if users were to speak directly to the recording device; this data should thus paint a pessimistic picture for our techniques. Nevertheless, as this work is only an initial exploration into the generation of cryptographic keys from voice features, this dataset was adequate to obtain initial evidence supporting the viability of the technique.

For each user $a \in A$ for a chosen population A of users, two of her five utterances were designated as "training" utterances: i.e., B_a is computed from these two utterances. In these experiments, $m = 46$, so that each B_a was generated to be 46-bits in length. The remaining three utterances were used as "testing" utterances to compute a false negative rate. Specifically, each testing utterance by user a yielded a total feature descriptor b; b and a fixed number—as defined by the parameter d—of feature descriptors b' close to b as described in Section 4.2 were examined to see if any are consistent with B_a. If all examined feature descriptors were inconsistent with B_a, then the utterance was counted as a false negative.

Two further parameters require elaboration to specify our experiments. First, for all approaches introduced in Section 3.2 for generating bits of a feature descriptor from the segmentation produced in Section 3.1, an important parameter is the value k used to define B_a for a user a. To recall from Section 2.1:

$$B_a(i) = \begin{cases} 0 & \text{if } \mu_i + k\sigma_i < t \\ 1 & \text{if } \mu_i - k\sigma_i > t \\ \perp & \text{otherwise} \end{cases}$$

where μ_i and σ_i are the mean and standard deviations of ϕ_i as measured in the most recent h (in our case, two) successful logins, and where t is a threshold that we take to be $t = 0$ for Algorithm III. Moreover, due to our evaluation here using features $\phi_i = \alpha \cdot (\mu(R_i) - c(R_i))$ for some vector α, the value of α is also an important parameter to our evaluation.

In order to demonstrate the potential viability of our technique, our evaluation of Section 4.4 strives to show that for a population A, there exist choices of α and k that yield both good guessing entropy and reasonable false negative rates. In order to show this, for a population A we searched a range of values for k, and for each k we computed both the entropy and false negative rate for various α. Due to the infinite number of possible α vectors, we searched only a small constant number of them (again, yielding a pessimistic picture): specifically, vectors α with components in the set $\{-1, 0, 1\}$, and then only vectors with up to 4 nonzero coefficients. We then plotted entropy and false negative rates as a function of k. Each plotted point is the entropy or false negative rate that resulted from the vector α maximizing

$$v \cdot p_{\mathsf{ent}}(\alpha) + (1 - v) \cdot (100 - p_{\mathsf{fneg}}(\alpha)) \qquad (4)$$

where $p_{\mathsf{ent}}(\alpha)$ is the percent of maximum possible entropy and $p_{\mathsf{fneg}}(\alpha)$ is the false negative percentage yielded by α and k. Here, v is a number between 0 and 1.

In total, utterances from 50 users were used in our experiments. However, since the only way we know to compute guessing entropy grows quickly in the number of users, we did not have sufficient resources to perform our evaluation for all 50 users at once. Rather, we divided these users into disjoint populations A of size 10, computed guessing entropies and false negative rates for these populations, and then averaged these measures across populations. That is, every data point presented in Section 4.4 is a data point averaged over these populations. A consequence of this approach is the best possible guessing entropy we could report would be 5.5, since $|A| = 10$ in all cases.

4.4. Experimental results

The results of the experiments described in Section 4.3 are shown in Figures 2 and 3. These figures show the percentage of maximum possible guessing entropy and false negatives for the vector α that maximizes (4), for each k when $d = 4$. The choice of $d = 4$ as the target for our experiments was influenced by two competing factors: as d is increased, the false negative rate generally declines, but the computational overhead of the login program (and thus the login delay experienced by the user) increases. The choice $d = 4$ best balanced these competing factors for our data set. However, given the limited scope of our tests (e.g., only a small fraction of possible α's were searched) and the poor

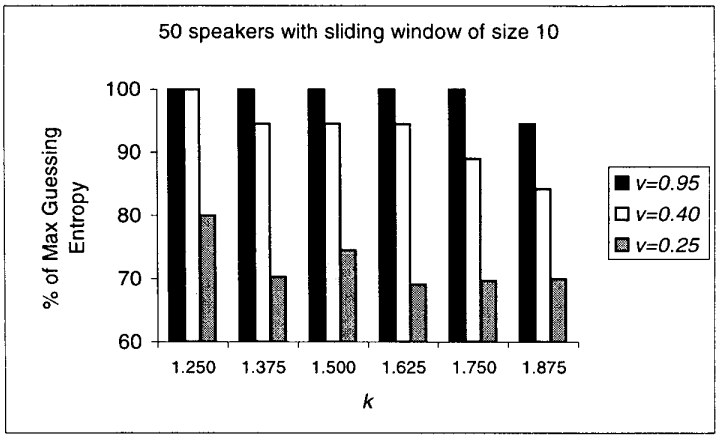

Figure 2. Guessing entropy percentages

quality of our data set (e.g., utterances were recorded over a phone call), we conjecture that Figures 2 and 3 present a pessimistic picture of our scheme and that $d = 3$ may suffice in practice.

Figures 2 and 3 demonstrate expected trends, in particular that both the guessing entropy and false negative rate generally decrease as k increases. Moreover, several of the points in these figures give evidence that our approaches may yield good security and reliability in practice. For example, the experiments with $v = 0.4$ illustrate a point ($k = 1.875$) at which the entropy is roughly 85% of maximum and the false negative rate is approximately 6%. We remind the reader that guessing entropy is itself a pessimistic measure, in that it presumes an attacker with perfect knowledge of $\{B_a\}_{a \in A}$. Thus, a guessing entropy of 85% of maximum may be reasonable in conjunction with a low false alarm rate. Alternatively, these figures also demonstrate that there are choices for α yielding maximum entropy with false negative rates still below 20% (e.g., at $k = 1.625$).

5. Future work

As this work is a first investigation into the repeatable generation of cryptographic keys from vocal utterances, it leaves many directions for future work. First and foremost, our immediate future work is targeting the generation of feature descriptors of length greater than $m = 46$ bits, since this is the effective length of the resulting derived key to an attacker who captures the key-generating device. Our goal is to reach a feature descriptor length that eliminates any need to encrypt the table T (c.f., [15]), since encrypting T using the text of the spoken utterance will provide only marginal additional strength with greater computational cost (see Section 1.2). Second, there is obviously room for investigation of alternatives and improvements to the algorithms described in Section 3, such as using likelihood more aggressively and, more generally, drawing from the vast bodies of literature on speaker verification and speaker recognition. Third, there is a need for more thorough empirical analysis of our proposals (and any other subsequent proposals). In particular, experiments that employ more appropriate data sets and larger numbers of users and utterances are needed. In addition, experiments targeted at identifying the best parameter values to choose in practice are very important. For example, such parameters include which vector α produces the best results for Algorithm III across a wide range of populations and passwords, or which cepstral coefficients and features offer the best performance for any particular algorithm.

An additional focus of our own work is the implementation of the applications described in Section 1.3. Our current target platform is a Psion netBook^{TM}, a portable, subnotebook-format device with a 190 MHz StrongArm processor. Use of the EPOC operating system makes this device compatible with next-generation cellular phones (e.g., the Ericsson R380 and communicator platforms, and more generally, forthcoming 3G phones), which are our eventual target for these techniques as they become available. This implementation should also enable us to collect data directly as to the usability of our techniques.

6. Conclusion

In this extended abstract we presented what are, to our knowledge, the first algorithms for reliably generating a cryptographic key from a user's utterance of a password. This key is constructed to resist cryptanalysis even against

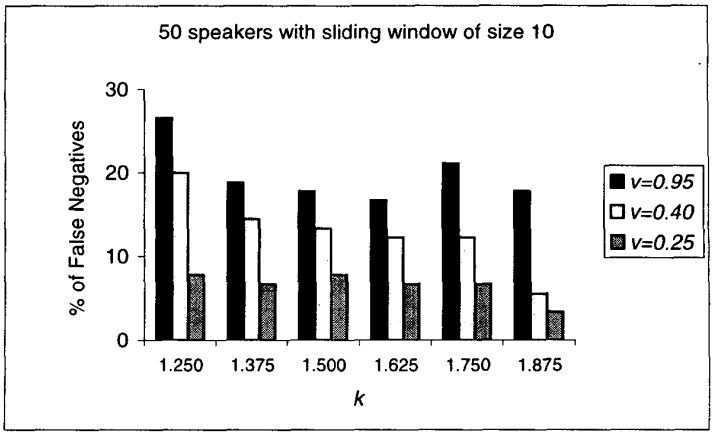

Figure 3. False negative percentages

an attacker who captures the key generating device. An empirical evaluation suggests that for well-chosen parameters, both the reliability of key generation and the key entropy can be high. We further outlined several problems for future work.

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] J. Daugman. High confidence personal identification by rapid video analysis of iris texture. In *Proceedings of the 1992 IEEE International Carnahan Conference on Security Technology*, pages 50–60, 1992.

[5] J. R. Deller Jr., J. G. Proakis, and J. H. L. Hansen. *Discrete-Time Processing of Speech Signals*. Macmillan Publishing Company, New York, 1993.

[6] G. R. Doddington. Speaker recognition—Identifying people by their voices. *Proceedings of the IEEE*, 73(11):1651–1664, 1985.

[7] D. Feldmeier and P. Karn. UNIX password security—Ten years later. In *Advances in Cryptology—CRYPTO '89 Proceedings* (Lecture Notes in Computer Science 435), 1990.

[8] G. D. Forney, Jr. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, March 1973.

[9] A. Juels and M. Wattenberg. A fuzzy commitment scheme. In *Proceedings of the 6th ACM Conference on Computer and Communication Security*, pages 28–36, November 1999.

[10] D. Klein. Foiling the cracker: A survey of, and improvements to, password security. In *Proceedings of the 2nd USENIX Security Workshop*, August 1990.

[11] Q. Li, B.-H. Juang, C.-H. Lee, Q. Zhou, and F. K. Soong. Recent advancements in automatic speaker authentication. *IEEE Robotics & Automation*, 6(1):24–34, March 1999.

[12] 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.

[13] U. Manber. A simple scheme to make passwords based on one-way functions much harder to crack. *Computers & Security*, 15(2):171–176, 1996.

[14] J. L. Massey. Guessing and entropy. In *Proceedings of the 1994 IEEE International Symposium on Information Theory*, page 204, 1994.

[15] F. Monrose, M. K. Reiter, and S. Wetzel. Password hardening based on keystroke dynamics. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, pages 73–82, November 1999.

[16] R. Morris and K. Thompson. Password security: A case history. *Communications of the ACM*, 22(11):594–597, November 1979.

[17] J. Naik. Speaker verification: A tutorial. *IEEE Communications Magazine*, pages 42–48, January 1990.

[18] L. Rabiner, B. Juang, and B. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[19] R. D. Rodman. *Computer Speech Technology*. Artech House, Norwood, MA, 1999.

[20] A. E. Rosenberg and F. K. Soong. Recent research in automatic speaker recognition. In *Advances in Speech Signal Processing*, pages 701–738, New York: Marcel Dekker, 1992.

[21] E. Schwartz. PDAs learn to listen up. *Inforworld.com*, February 4, 2000.

[22] E. Spafford. Observations on reusable password choices. In *Proceedings of the 3rd USENIX Security Symposium*, September 1992.

[23] R. Sproat, J. Hu and H. Chen. EMU: an e-mail preprocessor for text-to-speech. In *Proceedings of the 1998 IEEE Workshop on Multimedia Signal Processing*, December 1998.

[24] C. Soutar and G. J. Tomko. Secure private key generation using a fingerprint. In *Cardtech/Securetech Conference Proceedings*, vol. 1, pages 245–252, May 1996.

[25] 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.

[26] G. O. Williams. Iris recognition technology. In *Proceedings of the 1996 IEEE International Carnahan Conference on Security Technology*, pages 46–59, 1996.

[27] T. Wu. A real-world analysis of Kerberos password security. In *Proceedings of the 1999 Network and Distributed System Security Symposium*, February 1999.