PRIVACY ENHANCED REMOTE VOICE VERIFICATION

by

RAOUL CHRISTOPHER JOHNSON
B.S., Northern Michigan University, 2004

A Dissertation submitted to the Graduate Faculty of the
University of Colorado at Colorado Springs
in partial fulfillment of the
requirement for the degree of
Doctor of Philosophy
Department of Computer Science
2014
This dissertation for Doctor of Philosophy degree by

RAOUL CHRISTOPHER JOHNSON

has been approved for the

Department of Computer Science

by

__________________________
Terrance Boult, Chair

__________________________
Walter J Scheirer

__________________________
Albert Chamilard

__________________________
Rory Lewis

__________________________
Walter Andrews
This research explores creating a secure remote verification scheme using the voice as a biometric identifier, Vaulted Voice Verification. This research allows for remote biometric verification, focusing on mobile devices, in which the biometric data never leaves the user’s control. This is achieved by integrating theories from the computer vision research community with those of the voice research community. This research also contributes to the voice research community with a novel mixing of text-dependent and text-independent voice into a challenge-response protocol.

The Vaulted Voice Verification protocol takes authentic biometric input data, as well as data from other sources, and generates different models. With these models, the challenge-response protocol generates a scrambled challenge, which consists of real and chaff data, to be sent in such a way that only the intended recipient can send the desired response. This research also explores key exchange via challenge-response between remote parties using biometric identifiers, such as voice, to verify the identity of the parties during the exchange.

The major contributions of this research include: adaptation of Vaulted Verification
to voice, extension of the challenge-response protocol to include different types of challenges, mixing text-dependent and text-independent models into a protocol to enhance the overall robustness of speaker verification, creation of an index-based challenge-response protocol in which no biometric data leaves the remote device, a privacy-enhanced biometrically authenticated remote key exchange protocol, a newly created voice dataset, and a security and privacy evaluation of these technologies using a newly created dataset.
To my parents and my family.
Acknowledgements

The journey I’ve taken to make this thesis possible is unique and unlike any other. Numerous people have helped me travel this path; I could not have done it on my own. In fact, there are so many people that deserve acknowledgement that I would need an entire chapter just to scratch the surface. I hope this will suffice.

First and foremost I would like to acknowledge Terry for putting up with me during this effort and never giving up on me. He is someone I can always count on to give me some of his precious time, along with a thousand new ideas and directions, even though he has a million other things to do. Without him, this thesis would not be possible.

I also want to acknowledge the US Olympic Committee, Olympic Training Center Staff, USA Wrestling, my coaches throughout the years, and my Greco-Roman wrestling teammates. Without my Olympic pursuits, I never would’ve been able to travel the world, wrestle on the world stage, and meet so many amazing people, all while working towards my dissertation. The physical outlet that my Olympic endeavors afforded me have given me the perfect balance for the mental demands on my academic pursuits. My athletic career has taken many twists and turns and ups and downs over the years, but I know that I’m surely better for it in the long run and wouldn’t be where I am today without it.

I want to acknowledge the crew (past and present) at Securics, Inc., namely Walter S., Brian H., Mike W., Ethan R., and Demetrio C., that have patiently listened to me drone
on and on about Vaulted Voice Verification and all the ins and outs of index-tables and remote verification. Even without all the other support and encouragement I’ve received from them throughout this process, they deserve acknowledgement for putting up with my rants. The past and present members of the Vision and Security Technology (V.A.S.T.) lab have been crucial in the completion of this dissertation, and therefore also deserve acknowledgement. Also, the financial support from both Securics, Inc and the V.A.S.T. lab have been crucial in the completion of this degree.
# Table of Contents

1 Introduction

1.1 Introduction To The Problem ........................................ 4
1.2 Contributions of this Research .................................... 11

2 Background on Voice as a Biometric ............................... 14

2.1 Authentication Technology ......................................... 14
  2.1.1 Issues of Security and Authentication ....................... 15
  2.1.2 Issues of Remote Authentication .............................. 21
  2.1.3 Trade-offs: Privacy and Security ............................ 25
2.2 Recognition .......................................................... 27
  2.2.1 Speech .......................................................... 27
  2.2.2 Speaker ........................................................ 33
2.3 Voice Modeling ...................................................... 38
  2.3.1 Gaussian Mixture Models .................................... 39
### 2.3.2 Joint-Factor Analysis and I-vectors

### 3 Background on Protection: The Eternal Quest for Security and Privacy

#### 3.1 Biometric Templates

- 3.1.1 Encrypted Templates
- 3.1.2 Fuzzy Commitment
- 3.1.3 Fuzzy Vaults
- 3.1.4 Fuzzy Extractors
- 3.1.5 Biohashing
- 3.1.6 Non-Invertible Transforms
- 3.1.7 Cryptographic Key Generation
- 3.1.8 Secure Biometric Templates

#### 3.2 Template Protection Analysis

- 3.2.1 Replay Attacks
- 3.2.2 Attacks via Record Multiplicity
- 3.2.3 Surreptitious Key-Inversion Attack
- 3.2.4 Blendend Substitution
- 3.2.5 Decoding Attack

#### 3.3 Privacy in Perspective
4.1 Introduction to Remote Verification ........................................... 69

4.2 Challenge-Response ............................................................. 71

4.2.1 Basic Challenge-Response .................................................. 72

4.2.2 Biometric Challenge-Response .......................................... 76

4.3 Vaulted Verification ............................................................. 78

4.4 Combined Approach ........................................................... 81

5 My Voice Is My Key ............................................................... 83

5.1 Vaulted Voice Verification .................................................... 85

5.1.1 Enrollment Process .......................................................... 87

5.1.2 Verification Process .......................................................... 90

5.1.3 Initial Analysis ................................................................. 95

5.2 Mixing Text-dependent and Text-independent Models ................. 98

5.2.1 Enrollment ................................................................. 101

5.2.2 Verification ................................................................. 102

5.2.3 Improving Security and Usability ....................................... 104

5.2.4 Text-dependent and Text-independent Models ...................... 104

5.2.5 Increased Information from Responses ................................ 107

5.3 Index-Based Vaulted Voice Verification .................................. 109

5.3.1 Index-based Enrollment .................................................. 110

5.3.2 Index-based Verification ................................................ 112
5.3.3 Index-based Vaulted Voice Verification - A walkthrough \hspace{1cm} 115
5.3.4 Communication Overhead \hspace{1cm} 123
5.3.5 Storage Requirements \hspace{1cm} 125
5.3.6 Key Management and challenge strength \hspace{1cm} 126
5.4 Security Analysis \hspace{1cm} 129
  5.4.1 Security in “Bits” \hspace{1cm} 129
  5.4.2 Overall Security \hspace{1cm} 134

6 Can You Hear Me Now? Testing Testing 1-2-3 \hspace{1cm} 167
  6.1 Expectations and Resources \hspace{1cm} 167
  6.2 Vaulted Voice Verification Implementation \hspace{1cm} 168
    6.2.1 Server \hspace{1cm} 169
    6.2.2 Client \hspace{1cm} 170
  6.3 Datasets \hspace{1cm} 172
    6.3.1 MIT Mobile Dataset \hspace{1cm} 173
    6.3.2 MOBIO \hspace{1cm} 174
    6.3.3 UCCS Voices \hspace{1cm} 175
  6.4 Analysis \hspace{1cm} 179
    6.4.1 Initial Text-dependent Results \hspace{1cm} 179
    6.4.2 Initial Mixed Text-independent and Text-dependent Results \hspace{1cm} 182
    6.4.3 MOBIO Analysis \hspace{1cm} 186
List of Tables

5.1 Symbol definitions ................................................. 84
5.2 Probability and security bits for distribution ..................... 143
5.3 Guessing entropy for a question .................................... 149
5.4 Example knowledge infused biometric question ................... 159

6.1 The metadata collected by the VVV Data Collection Android app. 177
6.2 Effects of number of utterances on matching accuracy ........... 193
List of Figures

1.1 Three verification scenarios. ............................... 5

2.1 Example of a secure token ................................. 17

2.2 Authentication with a secure token. ...................... 18

3.1 Visual example of blended substitution .................. 62

4.1 Centralized Verification Scenario .......................... 70

4.2 Challenge-Response question and answer dialog ........ 75

4.3 Vaulted Verification enrollment .......................... 78

4.4 Vaulted Verification authentication/verification .......... 79

5.1 Purpose of Vaulted Voice Verification ................... 88

5.2 Initial Vaulted Voice Verification - Enrollment Process. 88

5.3 Initial Vaulted Voice Verification - Verification Process. 91

5.4 Text-independent and text-dependent Vaulted Voice Verification enrollment 100
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.6</td>
<td>VVV Vault Construction</td>
<td>188</td>
</tr>
<tr>
<td>6.7</td>
<td>DET plot text-independent on UCCS</td>
<td>189</td>
</tr>
<tr>
<td>6.8</td>
<td>Example image description for knowledge-infused biometric prompt</td>
<td>191</td>
</tr>
<tr>
<td>A.1</td>
<td>Prompt from Data Collection App</td>
<td>228</td>
</tr>
<tr>
<td>A.2</td>
<td>Data Collection App</td>
<td>229</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

According to [ITU, 2013], there are billions of smartphones currently in use worldwide, and the number continues to grow rapidly. [ITU, 2013] estimates that "by end 2013 there will be 6.8 billion total mobile-cellular subscriptions - almost as many as there are people on the planet." According to [Google, 2013], adaptation of smartphones in the United States grew over 10% from 2012 to 2013. It is estimated that in 2013 over 56% of the population of the United States owns a smartphone. As the smartphone market grows, so does the percentage of that market that use their phone for financial transactions, like mobile banking. A 2013 survey from [Google, 2013] suggests that over 53% of smartphone owners have used their phone for financial related transactions.

The smartphone market growth presents an increasing need for remote ID verification
solutions. Remote ID verification solutions are defined as solutions in which the processing of the verification information takes place on the user’s remote device, “remote” referring to the perspective of the server (i.e., to the server, a mobile device is remote). This differs from centralized verification solutions where the bulk of the verification processing occurs on a server, which inherently raises privacy concerns. Mobile may be the most important class of remote device, but from the view of the server, a laptop is also considered a remote device.

In general, verification solutions first use some form of authentication to establish an identity, then later verify a new identity against the previously established identity. In general, there are three factors available for authentication:

1. Something you have. (e.g., an ID card or a device or a physical token.)

2. Something you knows (e.g., a password or answers to a question.)

3. Something you are. (e.g., a biological trait.)

Biometric authentication, identification using biological traits, is required for a three-factor authentication. With biometric authentication, one can establish their identity (something they are) using biological traits that are relatively unique to them. With its natural use case in the mobile environment, the main biometric identifier discussed in this work is voice. A raw biometric identifier can not be directly revoked; therefore, verification solutions must protect the collected raw biometric data. Remote verification solutions
have the additional task of ensuring that the biometric data is protected while still using that data to verify an identity with a remote server.

There are a number of important questions that need asking:

1. Is there a way that voice can be used in a privacy preserving remote security protocol on a remote device?

2. If the verification occurs on a remote device, how is the server able to trust the authenticity of the transaction?

3. What happens if the remote device is compromised?

4. Even if the user verifies with the remote device, how can a remote server confirm the identity of the user?

5. How can privacy be maintained for the user while security is maintained for the server?

These questions will lead to other questions that will be discussed in various chapters.

This thesis answers these questions by way of a novel challenge-response protocol, Vaulted Voice Verification, resulting in a number of publications [Johnson et al., 2013b] [Johnson et al., 2013a] [Johnson and Boult, 2013] . This work builds on the general idea of Vaulted Verification [Wilber and Boult, 2012] [Wilber et al., 2012] in which a server challenges a remote device, expecting a response generated from a biometric. In this
thesis, we introduce Vaulted Voice Verification where a server challenge can verify the identity of the user using voice [Johnson et al., 2013b].

The concept of Vaulted Voice Verification is further investigated in [Johnson et al., 2013a], which explores expanding the challenge-response protocol and examining security issues for remote ID verification. While [Johnson et al., 2013a] answered many of the problems, it was somewhat impractical because of the large communication overhead. In [Johnson and Boult, 2013], this problem, as well as several others, is addressed by greatly reducing communication overhead and allowing for biometrically authenticated remote key exchange using voice.

1.1 Introduction To The Problem

The problems of verification via voice are not new. Different groups have defined different subproblems and worked to solve different aspects of this problem over the last 30+ years, each attempting to solve it a different way. While others have made progress on different aspects of voice verification, which we will examine in Chapters 2 and 3, they have operated on a centralized matching model. The contributions made in this research encompass the first real privacy preserving remote verification solution for voice.

To better illustrate the problem this research solves, we break the problem into individual pieces and examine the individual issues and concerns that exist within the realm of remote verification. Let us consider two events, each with multiple scenarios involving
voice verification. In the first event, we have a user, Alice, who is accessing her bank account. For the second event we introduce another user, Bob, who needs to communicate with Alice. We first focus on Alice as she accesses her bank account.

Alice has several choices on how to access her bank account. As shown in Figure 1.1, she can go into the bank, call the bank, use a computer, or use an app on her mobile device. These choices break down to three basic modes: in-person, centralized, and remote. No matter how she access her account, the first thing she must do is verify that she is who she says she is. If she chooses to walk into the bank, she must show her driver’s license, swipe her debit card (show documentation of her account), and enter her pin number.
If she decides to call the bank, she must tell the call center employee her pass phrases and may be asked a series of questions based on information she gave to the bank when she set up her account. If she uses her computer to access her account, she will have to enter a username and password from a known IP address. If she is coming from a previously unknown IP address, she will be required to answer an additional series of questions similar to those asked if she calls in. If she uses her mobile device, she must enter a username and a password.

Each access method requires Alice to verify her identity. However, each requires a different type of authentication; driver’s license, debit card, pin number, account questions, pass phrases, and username/password combinations. This highlights the difference between authentication and verification. Authentication, defined and discussed in greater detail in Section 2.1, is the process of making someone else aware of your identity and giving them confidence in their awareness. Verification, as defined by the International Organization of Standards (ISO), is a process defined as a “confirmation, through the provision of objective evidence, that specified requirements have been fulfilled” [ISO/TC, 2005].

Alice walking into the bank means she must show her ID, swipe her debit card and enter her pin. We consider this type of verification to be an in-person verification. It is in-person because Alice is physically located at the bank when verification is taking place. In-person verification is illustrated in Figure 1.1.
If Alice is unable to physically go to her bank to access her account, she has three options available: calling in, using her computer, or using her mobile device. In each of these scenarios, Alice accesses the bank from a remote location. Thus, the bank is the server and Alice is on a remote client from which she must verify.

The three remaining options available to Alice each present different problems. For Alice to call in and access her account, the bank must store information on Alice and that information must be accessible to any employee in the bank’s call center. Alice calling into the bank is considered centralized verification because even though she is remote, all of the actual verification occurs at the bank (in their call center). Centralized verification is depicted in the center of Figure 1.1.

In the centralized scenario, Alice is subject to an insider attack from a bank employee and a replay attack if an attacker eavesdrops on the call. This scenario exposes her data, allowing an attacker access to it. This scenario also leaves Alice open to a man-in-the-middle attack. The server is faced with the problem of not knowing if the transaction is actually Alice, or an attacker posing as Alice. In other words, in this verification scenario, the server can not trust Alice.

In the remote verification scenario, as illustrated in the bottom of Figure 1.1, Alice verifies herself on her remote device. Alice enters her username and password into her remote device and is then verified on her device. Her device then generates the appropriate key (token) as expected by the bank for access. This token assures the bank that Alice
properly verified herself with her device. That verification token is then sent to the bank so Alice can access her account. For remote verification, the bank expects a token from Alice, but what happens if someone else intercepts that key in transmission and later uses that key to impersonate Alice (replay attack)? What if an attacker steals Alice’s phone and makes it send the key no matter the password entered?

Considering the scenario of Alice accessing her bank account using voice as a biometric identifier highlights the issues associated with current biometric solutions. If Alice calls in to her bank, she would have to give voice samples to verify her identity. Her voice samples could be in the form of pass phrases or answers to questions. Someone eavesdropping on her call would then possess her raw biometric data needed to verify. Possessing the voice recordings gives an attacker all the necessary information to perform a replay attack. In these examples, Alice’s voice data is sent to the server at the bank so the bank may compare and verify it: centralized verification. For a comparison to take place, the bank must also maintain a copy of her verification data, in one form or another. The bank maintaining a copy of user’s biometric data raises privacy concerns which will be introduced later in Section 2.1.3. While the cost of storage is declining steadily, the cost of data security is not. There is a non-trivial cost for the bank to store and secure such biometric data. There is also a cost associated with transmission of voice data to the server at the bank.

With current voice biometric technologies, for Alice to use her computer or mobile
device to access her account, she needs to record her voice on her computer or mobile
device, and that recording is sent to a server for analysis and verification. This scenario
exposes her biometric data, allowing an attacker access to it, leaving Alice open to a man-
in-the-middle and replay attacks. We instead consider the scenario of Alice using her
voice for remote verification.

We use our scenarios to illustrate the pitfalls of current voice biometric techniques
when used for remote verification. In remote verification, the verification of her identity
would take place on Alice’s remote hardware (her mobile device or her computer). Using
current biometric verification techniques, Alice can use her voice to prove that she really
is Alice to her device. To verify with the bank, her device needs to inform the bank
of its assurance of her identity. As with most access protocols, current voice biometric
protocols result in a released key in one form or another. If the generated key needs to
be sent from the remote device to the server, it is exposed to the same replay and man-
in-the-middle attacks as centralized solutions. Would the bank be able to detect that it is
actually Alice and not an attacker who is on her device or performing an attack? What if
her device was made to always respond affirmatively?

Also, the process of verification becomes more scalable in a remote verification sce-
nario. In remote verification, the necessary storage and computing power requirements
are shifted from the Bank and offloaded to the remote user. Decreasing the load on the
bank increases the number of users the bank can processes without increasing the bank’s
resource requirements, maximizing scalability while minimizing cost.

We now turn our attention to another type of problem where Alice and Bob need to communicate securely. In the first scenario, Alice and Bob use asymmetric RSA keys for their communication [Jonsson and Kaliski, 2003]. For the purpose of this illustration, we use $RB_{priv}, RB_{pub}, RA_{priv}$ and $RA_{pub}$ to represent the private and public keys for Alice and Bob, respectively. Bob, using his mobile device, initiates a connection with Alice by sending a message encrypted with $RA_{pub}$ then $RB_{priv}$. Alice then receives the message on her mobile device and decrypts it using $RB_{pub}$ then $RA_{priv}$. After decryption, Alice responds with a message of her own, encrypted with $RB_{pub}$ then $RA_{priv}$. In this scenario, both Alice and Bob can be sure that the device on the opposite end of the communication possesses the appropriate private keys, but what about the identity of the user?

In the second scenario, Alice and Bob generate a shared symmetric key (person to person key exchange), using Diffie-Hellman [Diffie and Hellman, 1976] in this example. They start with some shared secret $S$ and each generate a private data $DA_{priv}$ and $DB_{priv}$. Each of them combine their private data with the shared secret to generate $SA_{pub}$ and $SB_{pub}$. Then they exchange $SA_{pub}$ and $SB_{pub}$. By combining $SA_{pub}$ with $DB_{priv}$ and $SB_{pub}$ with $DA_{priv}$, they each generate the same common private key $DAB_{priv}$. In this scenario, Alice and Bob have successfully exchanged a shared private key, but does it tell them anything about the identity of the individual they exchanged a key with? Currently
person-to-person key exchange makes no claim of identity of the individual, only proof of possession of the proper keys/devices.

In remote verification scenarios, problems exist that the current state of the art can not solve. Those problems include:

1. Insider attacks.

2. Replay attacks.


4. Liveness detection.

5. Privacy Concerns.


7. Verification of the individuals involved in key exchange.

1.2 Contributions of this Research

In the rest of this chapter, we look to understand the major problem being addressed in this work and what contributions this work makes to solve this problem. To address the questions posed in this section, we start by looking at the existing literature. We begin by breaking it into two parts, Chapters 2 and 3. In Chapter 2, we look at the use of
voice as a biometric identifier from its beginnings to its current state. We then turn our attention to template protection in Chapter 3. After getting familiar with the topics being discussed, we turn our attention to the beginnings of our solution: the idea of challenge-response in Chapter 4. Then, in Chapter 5, we explore the different facets of Vaulted Voice Verification and how it solves the problems defined in the introduction and subsequent chapters. We then examine datasets used and the experimental results in Chapter 6 before concluding in Chapter 7.

This work looks to solve the problems mentioned in Section 1.1. The main contributions of this research are:

1. Adaptation of Vaulted Verification to voice. [Johnson et al., 2013b]. Vaulted Verification is introduced in 4.3 and the adaptation is detailed in Section 5.1.

2. Extension of the challenge-response protocol to include different types of challenges. Introduced in Section 5.2. [Johnson et al., 2013a]

3. Combination of text-dependent and text-independent models into a protocol to enhance the overall robustness of speaker verification. This is also examined in 5.2. [Johnson et al., 2013a]

4. Creation of an index-based challenge-response protocol in which no biometric data ever leaves the remote device and in which communication is minimized is discussed in Section 5.3. [Johnson and Boult, 2013]
5. Creation of a privacy-enhanced biometrically authenticated remote key exchange protocol, which is discussed in Section 5.3. [Johnson and Boult, 2013]

6. Creation of a new and publicly available voice dataset made specifically for the mobile environment, as discussed in Section 6.

7. Extension of prior binomial modeling to use guessing entropy based probability estimations of both the biometrics and knowledge infused biometrics is discussed in Section 5.4 using data from Section 6.

8. Creation of a security and privacy evaluation of the impact of depth and width of Vaulted Voice Verification questions on overall bits of security is presented in 5.4 Section 5.4 using data from Section 6.
Chapter 2

Background on Voice as a Biometric

In this chapter we first examine current authentication methods. Then we focus on the progression of voice as an authentication method. Finally, we examine two of the main techniques used for modeling voice for use as an authentication method.

2.1 Authentication Technology

There are many different technologies for authentication. In practice, the authentication space is currently dominated by passwords. In this section, we look to understand some of the strengths and weaknesses of different authentication methods and protocols that are commonly in use. We examine the following authentication methods:

1. passwords
2. id/token cards

3. biometrics

We also explore some of the attacks to which the different authentication methods are susceptible. These attacks include:

1. stolen password

2. compromised server

3. insider attack

2.1.1 Issues of Security and Authentication

Many issues of note must be dealt with when working with authentication protocols. According to [Bellare and Rogaway, 1994], one uses the term “entity authentication” when authentication is defined as “the process by which an agent in a distributed system gains confidence in the identity of a communication partner.” Later on, [Lowe, 1997] goes on to say that, to authenticate, “one agent should become sure of the identity of the other.” The official definition, according to [ISO/IEC, 2009], states authentication is the “provision of assurance in the claimed identity of an entity.” The wording differs, but the intent is the same; authentication is a process intended to make someone else aware of your identity and have confidence in that knowledge. Thus, authentication systems and protocols prove that the users of the system are who they claim to be. It is important to
note that authentication differs from authorization, which concerns the level of access that a user might posses.

Security and authentication go hand in hand. For proper authentication, one must be certain that security is maintained before, during, and after the authentication takes place. There are many different authentication technologies that are commonly in use.

The most common method of authentication is passwords. Passwords dominate the authentication space because there are several advantages to using passwords. For example, passwords can be easily changed and a single user may have several different passwords. While text passwords are the dominating authentication method, there are different security issues that exist with passwords. People often forget their passwords. There is a very real IT cost associated with forgotten passwords, from infrastructure for changing passwords and sending reminders to person-hours for IT and tech support for user assistance. Also, most companies use email as a backup tool for resetting passwords. If an attacker gains access to the email account of a user, the attacker could take over accounts associated with that email account by requesting a password reset be sent [Jakobsson, 2005]. Because passwords only cover one of the three main factors of authentication, it is less effective at deterring attackers than other methods that incorporate more information. If, for a given system, a password is the sole method of authentication, and an attacker steals/cracks/hacks the password of an authentic user, that attacker would be able to fully
impersonate that user on the system. For example, in July 2013 someone hacked the Official Ubuntu Forums, walking away with 1.8 million passwords [Whittaker, 2013]. Also, a password-based authentication system is only as good as the passwords it requires. For example, if a system required an 8 character alpha-numeric password (the 8 characters must be taken from the standard 10 numbers and 26 letters), a brute force attack could be performed with at most \((36^8)\) attempts.

Another method of authentication is referred to as two-factor authentication. One of the ways this is implemented is for the user to have a secure token. The two factors of authentication secure token covers are, something you know and something you have. Multiple implementations of secure tokens exist; software-based “soft” tokens, “hard” tokens, usb tokens, smart cards, etc. While there are multiple implementations of secure tokens, they all effectively work in a similar fashion. Figure 2.2 shows an illustration of secure tokens. A user maintains a password, which they must remember, and the token. When it is time to authenticate, the user enters their password and the input from the
Figure 2.2: Authentication with a secure token. A user inputs their username/password information along with information from the secure token. When the bank receives the information, the token information is verified alongside the username/password information.

token. The input from the token changes at a predetermined time interval based on some predetermined algorithm. Figure 2.1 shows an example of a hard token. With a secure token, there is an increased level of difficulty for an attacker. Instead of needing to only obtain the password of the user, an attacker must also obtain either the token or enough of the information used to generate the output of the token.

Using secure tokens increases the level of security; however, there are multiple issues with the use of secure tokens: the cost of the token hardware/software replacement, forgotten tokens, and convincing users of the need to remember/carry the token. The cost of hard tokens is non-trivial. The hard token displayed in Figure 2.1 costs between $330 and $490 for a pack of 5 at the time of this writing. An advantage to using “soft” tokens, tokens based on software, over “hard” tokens, tokens tied to a dedicated hardware device,
is the cost of the device. As stated in [Bortolozzo et al., 2010], even with the increased level of security given by secure tokens, there are still security concerns with secure tokens. [Bortolozzo et al., 2010] details attacking security tokens based on PKCS#11 APIs.

Another way the two-factor authentication method is implemented is by using out-of-band information. In one scenario, the user previously entered their mobile number into the system. With this implementation, the user is sent a one-time use code or key to their mobile device. In order to authenticate, this code must be entered into the system, repeated to the operator, or otherwise incorporated into the authentication process. An example of this is detailed in [Aloul et al., 2009]. As with other secure token-based schemes, an obvious issue with this implementation is if the mobile device is stolen. Another possible issue is SMS man-in-the-middle attacks, as mentioned in [Schneier, 2005]. Simply, an attacker impersonates a bank and sends information to the authentic user such that the authentic user performs the SMS authentication and is unaware of the attacker. Such a scheme raises the question: How do you verify the receiver as well as the sender?

Biometric authentication is a way to enhance security as an implementation of a two-factor, or even three-factor, authentication and solve the issues presented with password and secure token-based authentication schemes. Biometrics give the ability to incorporate the third factor of authentication, something you are, into the process. With this, a two-factor scheme can now include something you are with something you know. This type of
two-factor scheme eliminates the necessity of tying the verification to a specific device. Also, with the addition of biometrics, a three-factor scheme can include something you are, something you have, and something you know. By incorporating information about the user into the authentication, biometric authentication schemes are able to authenticate users using all three of the main factors for authentication. With biometric authentication, if the device and password are stolen, the need for biometric input helps to maintain the security of the system.

Biometric authentication protocols do add in the third main factor of authentication, which enhances the overall security of the system; however, issues still remain. We consider the different issues from the security and privacy side of things. A major concern for biometric authentication systems is the somewhat public nature of some biometrics. For example, there are some systems that use facial features to prove the identity of the user. Aside from the common issues associated with pose and lighting, a big problem with the face as a biometric authentication method is how easy it is to replicate a face. For some systems, all that is required is a high quality video or photograph of the authentic user. There are international competitions and much research on both defeating and detecting such spoofs [Chakka et al., 2011] [Maatta et al., 2012] [Komulainen et al., 2013] [Erdogmus and Marcel, 2013].

Fingerprints are another common biometric used. While fingerprints are less public in nature than face, they are commonly left everywhere and are therefore easily reproduced.
Recently, a major mobile device maker embedded fingerprint biometric authentication hardware into their devices. While the protocol and implementation may prove secure, within a week of launch, users produced videos of how to bypass the authentication implementation by reproducing the fingerprint of the user using a printer [Bradley, 2013]. Thus, an attacker can bypass biometric security if the biometric is copied. Since biometrics are not revocable, security and privacy concerns would be raised if someone were to copy a biometric.

Storing biometrics on a server also raises concerns. If an attacker gained access to the server, the biometric would be compromised everywhere, not just on that server. This means that the attacker could take that biometric and access anything that the biometric is used to protect. For example, a user uses the same biometric to secure a locker at the gym that they use their bank account. If an attacker copied their biometric from the gym, they would not only gain access to the gym locker, but also to the bank account of the user; this poses a major problem.

2.1.2 Issues of Remote Authentication

With remote authentication, a user must authenticate from a remote location. That is, the user is in one location on their device and must authenticate with a server in a different location. Recall the verification scenario where Alice accesses her bank account using her mobile device. From the perspective of the bank, Alice is a remote user. As such, Alice
must remotely verify her identity to the bank.

There are two major classes of problems that exist with remote verification. One of the issues relates to a lack of mutual trust between the server and the client. The other issue revolves around the lack of knowledge between the parties. Most other issues that exist stem from or relate to these two.

In remote verification, trust is an issue for both the server and the client. When verification begins, the server can not trust that the client is who they claim to be. Because the server can not trust the user, the user must prove that they are who they claim to be. For example, when Alice wants to access her account, simply telling the bank that her name is Alice is not enough; she must prove she is Alice. In biometric verification, the server cannot simply send the client a biometric template to match against because the client can just send a positive response without a guarantee that biometrics were actually used. Therefore, the server can not trust answers from the remote client. The issue of trust for the server breaks down into two distinct questions. The first is whether or not the user possess the proper credentials (username, password, etc). The second is the proof of identity of the user (the actual person using the device).

There also exists a trust issue for the client. For the client to verify with the server, the client must send data to the server for the verification to take place. Sending the biometric data to the server raises privacy and trust issues for the client. How does the client ensure that their biometrics will not be misused or mishandled? What recourse would the client
have if the biometrics are misused or mishandled? For example, consider a scenario where a user, Bob, uses his biometric to secure a gym locker, access his bank account, access secure information (classified documents, chemical weapon launch codes, etc.) at work, and all three entities store his biometric data. All three entities have different security needs and will invest in security accordingly. If an attacker breaks into the gym’s data center and steals Bob’s biometric, it would be compromised across all three entities, not just the gym, leading to a much bigger problem. Even if Bob uses his biometric at the three places at different times, the gym as a teen, the bank as a young adult, and accessing secure information later on in their career, the problem remains. Thus, the client can not trust the security and privacy of their data given to the server.

The issue surrounding the lack of knowledge turns up during the initial verification and is similar to the enrollment dilemma. If the user needs to access the server, but has never done so before, how does the user verify? In traditional non-biometric verification, this issue is solved with trusted third parties (TTP). VeriSign is one such TTP. Public key cryptography is a way to establish trust between entities; between a server and a remote user or between remote users. With public key cryptography, two entities can exchange some information and generate a shared key for synchronous key encryption [Diffie and Hellman, 1976], or they can use public/private key pairs for asynchronous key encryption [Jonsson and Kaliski, 2003]. With synchronous key encryption, two entities exchange a key in a secure manner. This key exchange ensures that communication
can be trusted because together each user possesses the requisite key for encrypted communication. With asynchronous public key cryptography, if a user needs to verify with a server that they have never accessed, the server can go to the TTP and grab their verification information (their public key or certificate) and use it to verify the user. However, such verification does not take into account the identity of the user, only that the user possesses the correct credentials.

While not the focus of this thesis, biometric key infrastructure (BKI), detailed in [Al-bahdal et al., 2013], was created to solve the problem of identity incorporation into remote verification using TTPs. While incorporating biometrics into the process solves the issue of identifying the user, it also introduces new challenges. The biggest challenge for the user concerns how to protect the biometric. For the server, the biggest challenge is how to trust the remote user. Thus, the biggest challenge is finding something that sufficiently protects the user’s data while simultaneously convincing the server that a proper match took place and that the results can be trusted. Vaulted Voice Verification addresses this challenge.

Explored further in Chapter 3, current biometric solutions do not consider authentication that occurs remotely, and many of them have fundamental flaws that render them useless in any verification scenario. One big problem with current biometric verification systems is that they require the matching to take place on the hardware of the verifying
party (centralized verification). In other words, if Alice wants to remotely verify herself with her bank using current biometric protocols, she must send her biometric data to the bank for verification. Sending her biometrics to the bank for verification increases security for the bank, but raises privacy concerns for Alice.

2.1.3 Trade-offs: Privacy and Security

As mentioned in Section 2.1.1, security is an important part of biometric verification. Privacy is another aspect of biometric verification that needs to be addressed. The widespread use of remote biometric verification for security potentially opens Pandora’s Box for privacy issues. As defined in [Prabhakar et al., 2003], ”privacy is the ability to lead your life free of intrusions, to remain autonomous, and to control access to your personal information.” In light of this definition, let us show how such privacy concerns exist with the growing use of current biometric verification technologies.

One privacy concern, eluded to in the previous section, is referred to as the Biometric Dilemma [Scheirer et al., 2013] [Scheirer et al., 2010]. The biometric dilemma states that as the use of biometrics increases, so does the associated security risk. We see an example of this with Bob using his biometrics at three different locations: at the gym to access his locker, at the bank to access his accounts, and at work to access secure/sensitive information. While his work might go to great lengths to properly protect his data, what about his gym, where the added protection of a biometric lock is merely a convenient
feature? If an attacker wants to impersonate Bob to access secure information from his work, the attacker could obtain the necessary biometric data from the lower security area. Often times, a person’s biometric data can be obtained legitimately without the need to go after low hanging fruit for attacks. For example, [Scheirer et al., 2013] cites an article from 2001 where the state of Colorado tried to sell its DMV fingerprint and face databases to any willing buyer.

Another privacy concern is the lack of anonymity resulting from incorporating biometric information. For example, as Section 2.1.2 mentioned, remote mobile biometric verification involves the transmission of biometric data for proof of identity. This may be fine when a user wants to verify themselves, but what about when a user wants to be anonymous? Raw biometrics are easily obtained: pictures of faces in a crowd, lifting latent fingerprints, recording a voice. Consider Alice as she submits her biometric sample for remote verification using her mobile device. If her biometric sample is later used for covert verification, her privacy is compromised.

With current biometric verification protocols, the biometrics, or a representation thereof, are stored on a server for later verification. With that biometric information, the institution maintaining that server possesses the ability to verify any other biometric sample against the sample data stored. Storing the biometric data presents a potential privacy issue if the stored biometric data is accessed and used to verify an individual in a fashion that is outside the agreed-upon scope. Such data could be shared and used to gain
additional information about an individual.

Such privacy-related concerns relate to the transmission of biometrics data to the verifying party; the verifying party must ensure security. In the earlier example of Alice remotely verifying herself with her bank, the bank needs to maintain its security even though the verification is happening remotely. For this to happen, Alice must transmit her biometrics to the bank for verification. This begs the question: Why must Alice give up her privacy for the bank to maintain its security? Alice has a need to maintain her privacy at the same time the bank maintains its security.

2.2 Recognition

2.2.1 Speech

Speech recognition is the art of determining what is being said. Because machines can’t process like the human ear, efforts are made to try to translate speech into something computers can understand.

Speech recognition has been around a long time. In the 1930s, Bell Laboratories worked towards developing a way to build a model for speech [Juang and Rabiner, 2005]. Since then, there have been many groups that have worked on the problem of speech recognition. In the 1950s Davis, Biddulph, and Balashek of Bell Laboratories developed a mechanism for recognizing digits. [Davis et al., 1952] This machine was used for a
single speaker speaking at ”normal speech rates”. With their machine, they were able to achieve about a 97% recognition rate once the machine was adjusted for a particular user. The work from [Huges and Halle, 1961] in the 1960s represents one of the first real attempts at doing some sort of statistical analysis on recordings. They realize that there ”exist several acoustic features that are stable over a wide range of voice inputs and phonetic environments”. These stable acoustic features emerged as part of the foundation of modern speech recognition [Furui, 2005]. Up until the late 1960s, computer processing was still not very fast or efficient when it came to general purpose computing. Because of this, most of the work that was done through the 1960s utilized hardware solutions that were made specifically for their purpose. In the late 1960s, [Viterbi, 1967] was published. In it, Viterbi describes a decoding algorithm for convolutional codes over a noisy channel. The algorithm describes a tree-like structure in which the path taken from root to leaf should be maximized by the related probabilities between each node in the path. Using the described algorithm, which came to be known as the Viterbi algorithm, the speech recognition community was able to more efficiently compute the most likely string of text in a given signal.

Previous to the 1970s, most speech recognition systems worked on simple words and short sentences for a single speaker. However, in the 1970s, a group out of Carnegie Mellon University (CMU), [Newell, 1978], created a system called Harpy that was able to work on continuous speech using a vocabulary of around 1,000 words and multiple
speakers. According to [Furui, 2005], Harpy “was the first to take advantage of a finite state network (FSN) to reduce computation and efficiently determine the closest matching string.”

Many of the speech recognition techniques developed before the 1980s were based on templates and pattern recognition. In the 1980s, focus shifted to models based on statistical analysis. This came in the form of the hidden Markov model (HMM) [Bahl et al., 1983] [Rabiner, 1989]. HMMs were first introduced in the 196’s by Baum, et al., [Baum and Petrie, 1966] [Baum and Eagon, 1967], and later utilized in speech recognition in the late 1970s [Jelinek, 1976]; however, HMMs did not gain wide understanding and use in speech processing until the 1980s.

A HMM is a statistical modeling process in which the system modeled is a Markov process with hidden states. As shown in, [Rabiner, 1989], a Markov process can be considered a system of states in which the states each have observable transition probabilities. This means that there is some probability that a system in state A will move to state B instead of staying in state A. The probabilities for the transitions can be expressed in the form of a state transition matrix. The transition matrix is known and observable. With this, a model of the system can be created based on the transition matrix and observed states. In an HMM, the transitions are hidden and, therefore, non-observable, but the outcomes are observable. The author gives an urn and ball example to explain. In the example, several urns each have several colored balls. A ball is randomly chosen from
a random urn, its color recorded, and placed back in the urn. Then, based on a random process associated with the current urn, the process of selecting a ball is repeated. The simplest HMM is one where the states are represented by the urns, each having an associated color probability, and the choice of each urn is guided by the HMM state transition matrix. In an HMM, the current state and the transition matrix are used to predict the next state. With an HMM, speech is modeled as a set of transition probabilities. Overly simplified, given a word, an HMM determines the probability of the next word.

Using HMMs, other groups worked on enhancements to speech recognition using maximum likelihood techniques for model adaptation. [Bahl et al., 1983] applied maximum likelihood techniques to HMMs for continuous speech recognition. [Gales, 1998] used the techniques to enhance HMM based speech recognition.

In the late 1990s, much was done to advance the state of speech recognition. The work performed in that time period successfully increased the number of recognizable words (vocabulary size) to around 65,000 [Lippmann, 1997]. In [Lippmann, 1997], the author described the state of speech recognition at the time and how different systems performed on six different corpora. The work presented the results in terms of word errors in recognition. The rates differed from one corpora to another, but in general, human performance was an order of magnitude better than machine. For example, using a Texas Instruments dataset from [Leonard, 1984] containing 25,000 digits and letters, the author plotted the error rates of humans versus a specially designed HMM machine
from [Chou et al., 1994]. In their results, the human error rate is 0.009% and the machine error rate is 0.72%, almost two orders of magnitude different.

Since the 1990s, speech recognition has taken off, and much research has been done. Groups have explored numerous techniques to attack different problems within the speech recognition process. For example, work using neural networks has looked at issues related to large-vocabulary speech recognition [Dahl et al., 2012]. In their work, they utilize a “pre-trained deep neural network hidden Markov model (DNN-HMM) hybrid architecture” and compare it to a GMM-HMM baseline.

In its current state of the art form, speech recognition can be broken down into four main steps: pre-processing, training, modeling, and decision making. The pre-processing steps of speech recognition and speaker recognition are very similar, but in speech recognition, additional steps are taken so the system can identify what is being said.

In the pre-processing step for speech recognition, the audio file is first broken up into windows. The features are then extracted from those windows, as in speaker recognition [Charbuillet et al., 2007] [Murty and Yegnanarayana, 2006] [Zheng et al., 2001] [Han et al., 2006] [Kuldip and Bishnu, 2003]. Instead of attempting to combine all the features into a speaker’s voice model, speech recognitions systems will make models of the different parts of speech using various techniques [Povey et al., 2010] [Baker et al., 2009a] [Baker et al., 2009b] [He and Hung, 2009] [Lee et al., 1989].

From a speech recognition system’s point of view, speech consists of sentences that
need to be broken down into smaller subparts before recognition can occur [Lee et al., 1989]. Sentences break down into words. Words break down into syllables. Syllables break down into phones (the basic unit of sound that is combined to produce speech, not to be confused with phonemes or allophones). There are between 40 and 50 phones in a language [Yan and Barnard, 1995]. On average, a word will contain about 7 phones. Given this, there are close to $40^7$, or just over 168 billion, possible words in a language. The Oxford English dictionary contains over 200,000 words; however, even a very educated person will rarely use more than 20,000 words, which makes the problem of speech recognition much more feasible.

Because there are so many possible combinations of phones that can be used, most recognition systems will use some sort of statistical analysis to assist the process [Juang and Rabiner, 2005] [Baker et al., 2009a] [Furui, 2005] [Hirsimaki et al., 2009] [Lu et al., 2011]. This is done by examining large volumes of sentences (books, newspapers, etc..) and keeping track of successive phones. Two successive phones are called diphones, three are called triphones, and four in a row are called quinphones. Using the knowledge gained from this examination process, speech recognition systems are able to better chose which phone it thinks is correct.

The way this decision is made depends on the system. In general, systems will construct some sort of graph or tree [Nolden et al., 2010] [Nallasamy et al., 2012] [Lee et al.,
1989] [Lee and Kawahara, 2009]. These structures are sorted based on the probabilities obtained from the training process. Using such a structure allows systems to quickly navigate the possibilities of phone combinations to determine the best possible choice.

Within the field of speech recognition, there exists text-dependent and text-independent recognition. Text-independent speech recognition methods focus on extracting and utilizing features independent of the phonetic context. The different sounds and articulations presented during testing are typically different than those in training, so the text-independent methods must account for a wide variety of phrases [Campbell Jr, 1997]. Text-dependent speech recognition methods utilize time alignment to make reliable comparisons between two different utterances of the same text. [Furui, 2005] In text-dependent systems, the phrases are known to the system at recognition time, and the user is prompted to speak the exact phrase.

2.2.2 Speaker

Speaker recognition is a wide-ranging field that possesses many similarities to speech recognition in terms of tools and techniques used to solve problems. The main difference between speaker recognition and speech recognition is that speaker recognition is not as concerned with what is being said as with who is saying it.

Speaker recognition consists of two main tasks: identification and verification. With identification, speech input is matched against a gallery of known speakers. Speaker
identification is used in both closed and open set scenarios. In closed set, there exists a set of known speakers to which input speech will be classified. As such, the new input speech will be matched with one of the known speakers, whether or not it came from one of the speakers. In open set, the system can accept and classify the speech input or reject the new speech input as not belonging to one of the existing known speakers. With verification, speech input has a claimed identity and the system must verify the claim. As such, the challenge of verification is how to deal with impostors.

The field of speaker recognition can be seen as a two-sided coin with cooperative recognition on one side and uncooperative recognition on the other. As stated in a 2000 NIST speaker recognition evaluation [Doddington et al., 2000], “When the speaker is cooperative, the system may know what the speaker is supposed to say, and better performance may be achieved. When the speaker is uncooperative or unaware, then the system will be handicapped by lack of this knowledge.” While the two sides of speaker recognition are different, there exists some overlap in the tools used. This work focuses on cooperative speaker recognition, but it is important to briefly mention uncooperative speaker recognition for completeness.

**Uncooperative**

The largest space for uncooperative speaker recognition is forensic speaker recognition [Champod and Meuwly, 2000] [Künzel, 1994] [Alexander, 2007] [Gonzalez-Rodriguez
et al., 2003] [Campbell et al., 2009] [Hollien et al., 2013]. The cases for use of forensic speaker recognition are more focused on the needs of various intelligence agencies and the criminal justice system.

Some of the problems associated with forensic speaker recognition are induced by speakers, and others are related to the signal itself. When recordings are being made of speakers who are unaware that they are being recorded, often times what is recorded is conversational speech among multiple parties. In conversations, people overlap each other, which can make speaker recognition difficult. Background noises are also a bigger problem for forensic speaker recognition because the speakers are not speaking directly into a recording device. With forensic speaker recognition, often times the only available information comes from previously recorded audio samples obtained in less than ideal situations [Hollien et al., 2013]. An example of this can be seen in [Jin et al., 2007], where the authors attempt to identify speakers who are whispering to avoid recognition. Other work, such as [Campbell et al., 2009] and [Rose, 2006], looks at the difficulty of the problem of forensic speaker recognition. Unlike with cooperative recognition, in forensic recognition, because the signal can be less than desirable and the channel noise significant, the recognition rates are generally much lower.

Uncooperative speaker recognition uses some of the same techniques as cooperative speaker recognition. Gaussian Mixture Model and i-vector techniques, discussed in Section 2.3.1 and 2.3.2, respectively, are common in both. However, in uncooperative speaker
recognition, additional processing takes place to enhance the signal [Scheffer et al., 2013].

Generally speaking, uncooperative speaker recognition is a hard problem. As stated in [Bonastre et al., 2003], “at the present time, there is no scientific process that enables one to uniquely characterize a person’s voice or to identify with absolute certainty an individual from his or her voice.” While advancements have occurred since the time of [Bonastre et al., 2003], the problem remains difficult.

**Cooperative**

Speaker recognition systems use information in the audio track to determine the identity of the speaker. The different speaker recognition systems employ a number of different techniques to do their job. In most modern cooperative systems, the process of speaker recognition can be broken down into three steps; pre-processing, voice modeling, and decision making.

In the pre-processing step, the audio track is first broken up into segments (windows) of speech and non-speech. This is done both to speed up processing and to eliminate adding silence information into the model. The window sizes are of arbitrary length, but are typically around 20-40 ms and are sometimes overlapped up to 50%. From there, the windows are then transformed from the speech spectrum, using a fast Fourier transform (FFT) or a discrete Fourier transform (DFT) respectively, and features are then extracted [Charbuillet et al., 2007] [Murty and Yegnanarayana, 2006] [Bimbot et al.,
2004] [Shriberg and Stolcke, 2011] [EhKan et al., 2011]. These features are known as cepstrum features. The term cepstrum, introduced in [Noll, 1964] for work on vocal-pitch detection, comes from reversing the first four letters of the word spectrum.

Once the features are extracted, they can be used to build a model of the speaker’s voice. The type of model depends on the application. One of the most popular types of models is the Gaussian Mixture Models (GMMs) [Kumar et al., 2010] [David, 2002]. A GMM can be considered a single state HMM. GMMs are popular because they are very good at representing data of this type. Recently, another type of type of voice model gaining popularity is the identity vector, or i-vector for short. The i-vector approach, having its basis in joint-factor analysis (JFA), addresses some session and speaker variabilities found in GMMs [Dehak et al., 2009] [Senoussaoui et al., 2010]. In Section 2.3.1, we will examine GMMs. In Section 2.3.2, we will explore i-vectors.

The recordings of many users will be compiled, and a model will be made based on their recordings. This model is generally referred to as the world model. An individual user of the system will make recordings that will be used to adapt the world model, so it will be tuned specifically for their voice. This process of adaptation for an individual user is referred to as the training, and as a result, each user will have a separate model stored in the system [Bonastre et al., 2008] [Zilca, 2002] [Scheffer and Bonastre, 2006].

Once the models have been built, no matter the method of construction, the system is then ready for making a decision. An unknown user will make a recording and claim
to be a known user in the gallery. The system will make a model of the unknown user’s voice and compare it against the model that is stored in the gallery for the known user. There are many different types of comparison algorithms available for systems to use, but in general it comes down to checking if the model of the unknown user is within some threshold of the claimed identity’s model [Zilca, 2002] [Shah et al., 2009] [Campbell Jr, 1997] [Senoussaoui et al., 2010].

2.3 Voice Modeling

One of the hardest things about using voice as part of a security scheme is how much the voice can change from day to day. When you wake up, the voice can be scratchy from not talking for the past few hours while you slept. If you’re sick, your voice will change because your nose is clogged up or your vocal chords are swollen. When you drink something hot or cold, the pitch of your voice can be lowered or raised. With so many variables, it is hard to make a good model based on voice that will account for all the normal deviations in your voice while not allowing an impostor to pass as you.

Many groups have been working on voice models over the years. As a result of years of research, there are a number of voice modeling techniques. Gaussian Mixture Models and i-vectors are two of the most popular modeling techniques. Because the focus of this research is not how to model the voice directly, but how to protect the voice biometric while maintaining security, state of the art software packages were examined and utilized
in pre-processing the data for use in this research.

### 2.3.1 Gaussian Mixture Models

Model generation is the process of turning raw data into something that can accurately represent the data in an efficient manner. This process takes a number of steps. These steps are different depending on the data that is being modeled. For voice models, the general steps are 1) sample the audio data, 2) create a distinctive feature set, and 3) represent those features in a meaningful way. This section will briefly describe one implementation of this.

Sampling the audio data is done by breaking up an audio stream into small segments, or frames. The size of the frame is dependent on the implementation, but they generally range from 20-50 ms. Once the data is sampled, it must be transformed from the time domain to the frequency domain. [Zheng et al., 2001] and [Murty and Yegnanarayana, 2006] explain, in greater detail, how the audio spectrum is broken up and turned into feature vectors, but the general idea can be explained in a fairly simple manner. The audio spectrum is divided into slightly overlapping frequency blocks, or frames, and then a transform of some type, typically a DFT, is used to transform each of those blocks into cepstrum coefficients. The specific type of Cepstrum Coefficients used is referred to as Mel-Frequency Cepstrum Coefficients (MFCCs). The MFCCs are what make up the feature vectors. The feature vectors are used to describe a given audio sample in a
meaningful and quantifiable way.

Once the feature vectors are created, a model can be generated. The model that is created can model almost anything about the audio sample depending on how the samples are grouped and how the feature vectors are created. Gaussian distributions have been found to be extremely useful in modeling the created feature vectors. Refer to [R. Rajeswara Rao, 2010] for a deeper understanding of Gaussian distributions, also known as Gaussian Mixture Models (GMMs), and why they are useful for modeling based on feature vectors.

The number of distributions chosen is dependent on the amount of data that is being modeled. The larger the number of distributions used, the finer the granularity of the model. If only a small amount of data is being modeled, there will not be enough data to necessitate a larger number of distributions.

**Speech Modeling Software Packages**

Initially this research began by using the Alize platform as a front end. As detailed in [Bonastre et al., 2008], Alize is an ”open source software for speaker recognition.” Alize was initially the responsibility of the Laboratoire dInformatique dAvignon (LIA) and funded by the French Research Ministry Technolangue. It was subsequently funded by two French National Research Agency projects: BIOBIMBO and MISTRAL. For speaker detection, the main program within the Alize project is called LIA_SpkDet. Hence, it is
often referred to as Mistral, Alize, or LIA_SpkDet.

The Alize toolkit uses two main types of speech modeling, GMMs, as mentioned earlier, and i-vectors, which will be discussed in Section 2.3.2. For its GMMs, it utilizes MFCCs for feature vectors. Alize uses a universal background model approach (UBM) as its base speaker recognition engine. This is done by first creating a general, or world, model that contains a representative sampling of voices from a given population, then creating a user-specific model.

After working with Alize for some time, it was decided to try another software package as a base platform, and CMUSphinx was selected. CMUSphinx is an open-source toolkit used for speech recognition and is described in [Lee et al., 1989]. It is a speech technology toolkit developed by Carnegie Mellon University. Unlike Alize, which is for speaker recognition, CMUSphinx is built for speech recognition.

Even though CMUSphinx was built for a different task, it actually ended up being a more robust platform for which to conduct this research. CMUSphinx uses the same modeling as Alize to model the vocal tracts while also modeling the different parts of speech to recognize what is being said.

**Comparisons**

Independent of the tool used to create the Gaussian Mixture Models, once the models are created, they need to be compared somehow. When looking into how to compare the
different models, many options were found that could have been suitable. The option finally chosen was Z-scores. Z-scores, also known as standard scores, were chosen for a number of reasons, primarily the simplicity of computation and the relative proximity of the components of the formula to the model data.

Each GMM contains a number of distributions, $D$, and each of those distributions have a number of components, $C$, for a total number of components of $N = D \times C$. For each Gaussian mixture component, there exists a mean, $\mu$, and a variance, $\sigma$. When comparing the similarity of two models, probe $p$ and gallery $g$, you have to use the variance of the probe or the gallery depending on if the intention is to see how similar the probe is to the gallery or how similar the gallery is to the probe. In equation 5.10, we are seeing how similar the probe is to the gallery, i.e., utilizing the variance of the gallery. As shown in equation 5.11, a final score, $S$, is the summation of the z-scores over the total number of components, $N$.

### 2.3.2 Joint-Factor Analysis and I-vectors

A recent extension of Gaussian Mixture Models, known as Joint Factor Analysis (JFA), allows for jointly modeling the session component and the speaker component [Kenny, 2005] [Kenny et al., 2007]. The session component is also known as the channel component. JFA is designed to capture information on the intersession variability in speech that GMMs do not capture. That is, it is designed to model the variation in recordings from
session to session for a given speaker. Modeling the variation requires multiple inputs from the same speaker using different channels. For example, telephone and microphone utterances can be combined during training. Properly labeling the samples allows for determining the origins of the variability between sessions.

The idea behind JFA is that a model generated from an utterance breaks down into two independent eigenspaces: the speaker (eigenvoice) and the channel (eigenchannel). Based on the independence of the two pieces, for models of two utterances of the same individual, $M$ and $M'$, there is a channel super vector, $c$, that can be used to synthesize $M'$ from $M$. The ability to separate the speaker and channel information allows for the channel effects to be treated discretely and separated during pre-processing. In theory, this allows for specifically derived channel effects to be added to a test utterance if the channel is known.

JFA addresses the issues associated with speaker and channel variability in GMMs. It does so by defining two distinct eigenspaces: the speaker and the channel. In [Dehak et al., 2009], they combine the spaces to make what they call “total variability” space. This total variability space is generated in the same fashion as in JFA except for the fact that they make no assumption as to the speaker for each utterance. That is, to generate the total variability space, each training utterance is regarded as if produced by different speakers. From here, the frames of an utterance are projected into the total variability space. The resulting feature vectors are called i-vectors.
One drawback associated with i-vectors is the large amount of training data required to build a proper total variability space. In [Senoussaoui et al., 2010], the authors look at ways to create a total variability space by augmenting insufficient (sparse) channel data with data from other channels. Specifically, the authors “estimate microphone eigenchannels (sparse data) with telephone eigenchannels (sufficient data).”

Since their introduction, i-vectors have grown in popularity because of their ability to closely model and compensate for intersession variability [Li et al., 2011]. In [Bousquet et al., 2011], the authors further examine ways to compensate for intersession variabilities. The authors examine a set of linear and non-linear techniques to account for and remove the intersession variability. Though relatively new, i-vectors appear an extremely useful tool for speaker recognition [Kanagasundaram et al., 2011] [Cumani et al., 2011].

**Scoring i-vectors**

As with GMMs, there are multiple ways to compare i-vectors. i-vectors live in high-dimensional eigenspace, so comparisons must be done using techniques suited for eigenspaces instead of ones normally associated with GMMs. Cosine distance, as seen in equation 2.1, is one such measurement.

\[
k(w_1, w_2) = \frac{\langle w_1, w_2 \rangle}{\|w_1\| \|w_2\|}
\]  

(2.1)

The linear kernel, denoted as \( \langle w_1, w_2 \rangle \), is normalized by both total variability factor
vectors, $w_1$ and $w_2$. This means that the linear measurement is first calculated for the two vectors, then the resultant value is normalized by the two vectors.

Support vector machines (SVMs) are a class of tools that are used to score i-vectors and are also used in multiple areas of computer science. SVMs are also used within different voice communities [Enqing et al., 2002] [Wan and Campbell, 2000] [Kinnunen et al., 2007]. They are fairly straightforward and are effective at classifying datasets. At their most basic level, SVMs perform the simple task of splitting data. That is, given a set of data, a linear SVM classifier will attempt to find a separator (line) that will divide the data. When trained properly, SVMs can tell you the likelihood of a datapoint belonging to one class or another. The likelihood is the SVM margin distance to the plane.
Chapter 3

Background on Protection: The Eternal Quest for Security and Privacy

3.1 Biometric Templates

The biggest issue with biometric security/privacy is that everyone has a limited number of biometrics, so the model/template used needs to be protected. If a system stored someone’s raw biometric data or template, and that system were to be compromised, that data would then be immediately compromised. Because of this, some form of template protection is needed. An absolute minimal level of protection is encryption of the templates. However, since they must be decrypted to use, this adds only minimal protection because they will be continually decrypted by the system, meaning the system must have the keys
This section briefly describes some state of the art template protection schemes. The majority of the schemes that currently exist can be put into two different categories. The first category includes the schemes created specifically for a certain biometric: face, fingerprint, voice. The other category includes those that are created as a general means of protecting a template once it exists.

### 3.1.1 Encrypted Templates

Encryption is a standard way to protect data. For standard password/certificate/key based encryption, matches can be done in the encrypted space; therefore, there is no need to ever have the raw password/information exposed. However, for biometric models, this is not the case. To understand why, we briefly examine basic rules of encryption.

In his seminal work on the theory of crypto systems [Shannon, 1949], Shannon names two properties for the operation of a secure cipher: confusion and diffusion. In his paper he gives a concise definition of the two terms. In simple terms, confusion means there should be an extremely complex and difficult relationship between the ciphertext and the generated key, and diffusion means there should be a complex and difficult relationship between the plaintext and the ciphertext. What this means in practice is that, for every 1 bit of input data changed, the resulting ciphertext (hash, key, etc) should have a change in roughly 1/2 of its bits.
Because the underlying data used to generate biometric templates is not stable, like the text of a password, small changes occur in the resultant template. When these templates are encrypted, even a small change will make it so the templates no longer match in encrypted space. With this, it is not feasible to match biometric templates in encrypted space. Even if the templates are encrypted, they must be decrypted every time a match is performed. The decryption key must be given to anyone who performs a match. Therefore, encryption alone provides minimal protection for the template.

### 3.1.2 Fuzzy Commitment

The fuzzy commitment scheme, first detailed in [Juels and Wattenberg, 1999], attempts to secure biometric data by sealing it in a “safe” that can only be “opened” using the original biometric. The goal of the fuzzy commitment scheme is to make it infeasible for anyone who receives the safe to open it without the biometric.

The general idea is that a key can be protected inside the safe using a biometric (a fingerprint in their example, but applicable to other biometrics). When a biometric that is similar to the original one is presented, the safe is opened and the key is released. To allow for corrupted/degraded biometrics to unlock the save, error-correction is used. [Lee and Kim, 2010] describes the scheme as one that focuses on tolerating more intra-class variation.
It has long been known that matching biometrics is a difficult problem due to the difficulty of obtaining exact matching samples [Jain et al., 2004]. The fuzzy commitment scheme was an initial attempt to account for the variability with the use of error correcting codes. The scheme, however, suffers from some deficiencies [Scherier and Boult, 2007]. One of the problems associated with the scheme is its intolerance of reordering the symbols generated by the biometric. The scheme will not work if the symbols are presented out of the order. Another issue of the fuzzy commitment scheme is the security over a non-uniform distribution. The scheme assumes that bits are generated from a uniform distribution $D$ over $0, 1^N$. If this is not the case, the security of the secured information will be unpredictably lessened. Also, [Ignatenko and Willems, 2010] points out details about information that is leaked in both the bound keys and the biometric templates. Other vulnerabilities associated with fuzzy commitment schemes are also discussed in [Stoianov et al., 2009].

### 3.1.3 Fuzzy Vaults

The idea of a fuzzy vault was introduced in [Juels and Sudan, 2006] by Juels et al. This work was created to account for the deficiencies found in the fuzzy commitment scheme.
Biometric Fuzzy Vaults (BFVs) [Scherier and Boult, 2007] are a kind of template protection based on the fuzzy vault scheme. While BFVs are typically applied to face and fingerprint templates, the underlying fuzzy extractor and fuzzy commitment schemes were created to protect any polynomial, not only those generated by face and fingerprints. BFVs are a way of protecting a key by means of biometric identifiers. In this scheme, biometric data is submitted and transformed into a dataset. This dataset consists of a bunch of numbers that uniquely identify the biometric. A polynomial, some $f(x)$, is then constructed with the data as coefficients. Data points are then used as inputs to the function, producing number pairs. A number of randomly selected chaff numbers are then generated so that they do not coincide with the real data. Pairs are also created from this chaff. In some instances, such as is described in [Scherier and Boult, 2007], there are 18 real data points and 200 chaff points. Without knowing which points are correct, it is extremely difficult to reconstruct $f(x)$.

Reconstructing $f(x)$ would boil down to the polynomial reconstruction problem. The polynomial reconstruction problem is a special case of the Reed-Solomon decoding problem, which is detailed in [Guruswami and Sudan, 1998]. This problem, as it relates to fuzzy vaults, is also explained in [Juels and Sudan, 2006]. While this template protection scheme has many advantages, it is also susceptible to different attacks, as detailed in [Scherier and Boult, 2007], some of which we discuss in Section 3.2.
3.1.4 Fuzzy Extractors

In [Dodis et al., 2004], [Dodis et al., 2006] and [Dodis et al., 2008], the authors discuss fuzzy extractors (FE). FEs are used to generate a key $R$ from some biometric $w$, and some helper data $P$. If given $P$, the extractor can regenerate $R$ using some $w'$ if $w'$ is “close enough” to $w$. For measurements of closeness, the authors use three metrics: set, edit, and Hamming. The authors also introduce the Secure Sketch (SS). The SS is similar to the FE but differs in the desired output. The FE desires the release a key with an approximate input, whereas the SS releases the original input with the submission of an approximate input [Boyen, 2004].

According to [Arakala et al., 2007], the FE is a combination of the SS and the Randomness Extractor (RE) primitives. The RE is used to generate a uniformly distributed string from the non-uniform input. The work of [Arakala et al., 2007] applies the FE to a fingerprint using a PinSketch, introduced in [Dodis et al., 2008]. The PinSketch is a construct that decreases the size and computational requirements of the SS to make it more efficient. In [Chang and Roy, 2007], the authors use FE to extract consistent bits from a fingerprint in a noisy environment.

An issue related to FEs is reusability. [Arakala et al., 2007] describes how the protected secret can be completely exposed when the fuzzy sketches and fuzzy extractors are
used multiple times. The authors point to issues concerning algorithmic, coding, and permutation vulnerabilities along with issues associated with noisy input. [Chang and Roy, 2007] points out other issues related to FE in terms of some of the assumptions made, such as noisy data. According to the authors, it is unclear how chaff generation affects the process under different noisy conditions. The authors of [Chang and Roy, 2007] propose a dimensionality reduction based technique to attempt to overcome some of the issues of the original implementation.

### 3.1.5 Biohashing

Biohashing is another method of template protection. In Biohashing, as described in [Jina et al., 2004] and [Belguechi et al., 2012], the biometric data is reduced to a code, sometimes referred to as a biocode, via a hashing technique. The code is generated by taking the inner product a random number and the biometric data.

During verification, a code produced by the subsequent submission is compared to the original code. A Hamming distance between the two codes is computed, and a decision is made based on the result. When the codes are being produced, a random number is inserted. For verification, the same random number must be used when creating the subsequent code. For revocation, a different random number can be used.

An issue with this method is that it is based on the random number. As mentioned previously, the same number needs to be used during both enrollment and verification.
If the random number is compromised, the scheme will become less secure. Also, as detailed in [Belguechi et al., 2012], if an attacker is able to obtain the code for as little as three users, the security of the scheme would be compromised.

### 3.1.6 Non-Invertible Transforms

For a transform to be non-invertible, there must be a significant enough difficulty associated with recovering the original biometric given the secure template [Nagar and Jain, 2009]. Another way to look at non-invertible transforms is think of a one-way transform being applied to a template [Maiorana et al., 2008]. The security of non-invertible transforms rests in the difficulty of inverting the transform, even if the defining parameters are known.

According to [Prabhakar et al., 2003], one of the drawbacks associated with non-invertible transforms is that they can not be very “strong.” This is because strong non-invertible transforms “cannot effectively deal with all the biometric signal variations (of the same person) in the transformed space.” This is similar to the issues seen with encrypted biometric templates. If the templates can not be matched in the transformed space, the transform becomes less useful.
3.1.7 Cryptographic Key Generation

In 2001 Monrose, et al. introduced some of the pioneering work in key generation from voice [Monrose et al., 2001]. That work described taking an unstable biometric, voice, and reliably generating a cryptographic key from it. The authors based this work on their earlier work for generating hardened typed passwords [Monrose et al., 2002b] (initially published in 1999 before online publication in 2002).

This work described taking advantage of techniques used in speech and speaker recognition to generate cryptographic keys. The authors generated cepstral feature vectors from speech samples and segmented them using segment vector quantization. Those segmented samples were then mapped to either 0 or 1. The resulting mapping was then used to look up elements of the derived key via a lookup table. The authors used a value, $k$, as a sensitivity measurement for their algorithm. As the authors state: “an important parameter is the value of $k$ used to define $B_a$ for a user $a$.” Monrose, et al. also discussed the ability to generate keys with 46 bits of entropy. In some of their later work [Monrose et al., 2002a], the authors described how to get up to 60 bits of entropy, albeit not from all individuals.

More recently, [Carrara and Adams, 2010] looked at applying randomized biometric templates (RBTs) to voice. Similar to the work in [Monrose et al., 2002b] and [Monrose et al., 2002a], the RBT algorithm, originally described in [Ballard et al., 2006], uses
quantization methods to account for perturbations. Departing from the use of cepstral coefficients for feature representation, Carrara et al. used perceptual linear predictive (PLP) analysis and delta analysis on each frame. These techniques are described in [Hermansky, 1990] and [Proakis and Hansen, 1993], respectively. The authors derive vectors consisting of PLP and delta coefficients. Using RBT generated keys, Carrara et al. reliably generated keys with a maximum of 51 bits of entropy.

Both groups used voice biometrics to generate keys suitable for use in cryptographic schemes. However, the keys generated by each group suffer from similar deficiencies. Both schemes make assumptions in their attack models that allow them to avoid the issue of replay attacks. In either protocol, if an attacker possessed a recording of the authentic user speaking their password, the attacker could completely bypass the protocols. While the later work shows an increase in the number of generated bits of entropy, the amount still represents a limiting factor in the usefulness of such protocols in remote verification.

### 3.1.8 Secure Biometric Templates

Once a biometric is reduced to a template and stored somewhere, protecting the template becomes very important. To overcome the issues related to template security, different groups have worked on generating secure biometric templates [Nandakumar et al., 2007] [Inthavisas and Lopresti, 2011b].
The work in [Nandakumar et al., 2007] looks to secure fingerprint templates by hardening them. They do so by adding a password based randomization to the biometric template. The resultant data is then secured using a fuzzy vault. The fuzzy fault is then encrypted using the same password used to randomize the biometric template.

In [Inthavisas and Lopresti, 2011b] and extended in [Inthavisas and Lopresti, 2012], they also attempt to secure the templates by using passwords. Different from the work of Nandakumar, Inthavisas et al. uses the passwords to generate the templates themselves. The authors describe a process of extracting features, mapping to binary, and “hardening.” Their overall goal is to find features of a certain length, \( D \), that the user can reliably generate.

For their process, they initialize the system using one of the training utterances. Then their system breaks down the remaining utterances into feature vectors. Feature vectors, as they describe them, consist of \( m \) frames that will be mapped to a binary string of length \( m \), one bit per frame. The resulting binary strings are called feature descriptors. Finally they define a set of what they call “distinguishing features” from sets of \( l \) feature descriptors. In [Inthavisas and Lopresti, 2011a], the Inthavisas et al. describe the goals of the hardening process: “Specifically, let the total number of bit[s] derived from the hardened template that corresponds to the distinguishing descriptors be \( T \); the system should yield \( T \) as less than or equal to \( D/2 \).” This process is described in detail in [Inthavisas and Lopresti, 2011a].
Both groups focus on securing biometric templates. Their focus is ensuring that the templates generated from biometrics are secure against attackers. Their work, as with the work of others, focuses on solving problems associated with centralized verification. Issues analyzed in this sort of work looks at attackers accessing the templates and attacking them. In both of these works, the systems release the same data every time they match. With this, an attacker only needs to see a positive answer once in a remote verification scenario to compromise the system; therefore, these systems are not suitable for remote verification.

3.2 Template Protection Analysis

In Section 3.1 we looked at some of the current template protection techniques available for protecting biometrics. While the previously mentioned techniques work well enough, they are all susceptible to attacks. Because of the attacks to which they are susceptible, they are not well suited for use in any biometric verification scenario, especially remote. [Scherier and Boult, 2007] was one of the first efforts of formalizing attacks against biometric templates. It has been cited hundreds of times with dozens of other types of attacks being added every year. This section gives a brief overview of some of the associated attacks.
3.2.1 Replay Attacks

While many schemes offer protection using biometrics, the protection is greatly mini-
mized if a copy of the biometric can be obtained. As they relate to voice, replay attacks
involve an attacker recording the user as they speak their passphrase and later using that
recording to attempt to gain access.

As previously mentioned, encryption alone is not a suitable defense against an at-
tacker. However, let us consider what effect a replay attack would have if biometrics
were, in fact, stable. If biometrics were stable, and could be used in encryption, they
would be similar to a password. If someone were to copy the biometric, they would effec-
tively obtain the password and would therefore possess access to any system protected by
the biometric. Because the key never changes and is, generally speaking, easily obtained,
an attack becomes trivial.

The fact that biometrics are variable and, in some cases, easily reproducible (lift-
ing latent fingerprints, recording voice, taking high quality photos or videos) leads to
the creation of more complex protection schemes. Many schemes utilize some form of
two-factor authentication to combat the issues associated with replay attack [Ratha et al.,
2001]. Two-factor authentication, as mentioned earlier, is achieved by the addition of a
pin or password. As mentioned in [Tang et al., 2008], biometrics are generally consid-
ered to be public information and so additional credentials, such as passwords and tokens,
3.2.2 Attacks via Record Multiplicity

Attacks via record multiplicity (ARM), as described in [Scherier and Boult, 2007], are attacks achieved by combing multiple instances of a protocol to reveal the originating biometric or key. Such attacks are also known as correlation attacks. These attacks take advantage of protocols by correlating the data of different templates created from the same biometric, allowing for an approximate recovery of the originating biometric or key.

In their work, Scheirer and Boult describe how ARM based attacks easily defeat fuzzy vaults. In general, the attack states that fuzzy vaults can be broken by overlaying two fuzzy vaults, assuming the same database is used to bind two different keys. Because the vaults contain authentic data hidden by randomized chaff, once the data from two vaults are overlaid, authentic data should align and the chaff should not. While some chaff from two vaults may align, the built in error correction will account for this. Due to the correlation of data based on the same biometric, overlaying two fuzzy vaults reveals the real biometric data from the chaff. This type of attack on a fuzzy vault allows for “at a minimum link[ed] records, and in the most case can retrieve the template $X$ and the secret,” the authors state.

The biohashing scheme is also susceptible to ARM based attacks. According to [Ouda et al., 2011], ARM based attacks “not only let the attacker to reveal some information
about the original biometric data, but also may enable him to recover the original features entirely even with a limited number of compromised templates.” In their work, the authors demonstrate how to reduce the transformation of biohashing from $2^{m-1}$-to-1 to $2^{m-k}$-to-1 to 1-to-1, and therefore invertable, as the number of compromised BioCodes, $k$, reaches $m$, with $m$ representing the number of compromised databases.

### 3.2.3 Surreptitious Key-Inversion Attack

In the Surreptitious Key-Inversion (SKI) attack, an attacker is somehow able to obtain the key that is released by the biometric protection scheme. According to [Scherier and Boult, 2007], this attack allows an attacker to decode the biometric template data, by identifying the values the are related to the key. Simply put, this class of attacks leads to the ability to guess the biometric data based on stolen keys. This class of attacks also allows an attacker to access the data secured by the authentic user’s biometric data without needing the authentic user’s actual biometric data. To better understand this attack, we use a fuzzy vault example. We will examine two methods of SKI attack. The first involves an insidious attack and the second focuses on a stolen key.

The insidious SKI attack takes advantage of the enrollment process. A fuzzy vault, as explained in Section 3.1.3, generates a polynomial during the enrollment process. That polynomial is based on the biometric data points of the user. For an insidious SKI attack, an attacker simply needs to make sure that the resulting polynomial also crosses enough
of their data points as well. Such an attack would not interfere with the authentic user in any way because both the attacker’s and authentic user’s biometric data would generate the same polynomial required to unlock the vault.

For a stolen key-based SKI attack, an attacker would take advantage of the verification process. With traditional biometric template protection schemes, the result is a key release. In a fuzzy vault, the released key is essentially the polynomial that gets released from the vault. For someone to verify an individual, they only need to possess the vault and the key. If an attacker is somehow able to obtain the key, that attacker could simply project it into the vault and see which data points line up, revealing the authentic biometric data points.

The SKI attack shows a major vulnerability in the fuzzy vault template protection scheme. The reason for giving someone a fuzzy vault and key pair is the lack of trust in that person. However, using the SKI attack, giving someone a fuzzy vault and a key gives them the biometric. As such, the scheme is broken at a fundamental level. With this attack, the data secured by the biometric is obtained along with the biometric data itself.

3.2.4 Blendend Substitution

Blended substitution is another class of attack. First described in [Scherier and Boult, 2007], blended substitution attacks occur when attacker modifies existing templates instead of replacing them, as they do in standard substitution attacks. An example of this
The blended substitution attack as originally described in [Scherier and Boult, 2007].

![Blended Substitution Attack Diagram]

Figure 3.1: Visual example of the blended substitution attack as originally described in [Scherier and Boult, 2007].

In this example we have our authentic user and our attacker. The attacker wants access to the system, but is afraid of getting caught. If the attacker replaces an authentic user’s template with their own, the chances of detection increase. So, instead of replacing an existing template, which could alert the authentic user, the attacker modifies an existing template so both the attacker and the authentic user can access the system using the template.

This type of attack is especially interesting because, when such an attack occurs, the authentic user has no indication. With the attacker’s data blended into the authentic record, there is no way to detect the attacker as they attempt to access the system.
warning signs such as failure of the authentic user to gain access); the attacker will appear as if they are an authentic user. This presents a big problem for traditional biometric template protection protocols because they do not address this issue.

### 3.2.5 Decoding Attack

The decoding attack is a statistical attack against the fuzzy commitment scheme [Rathgeb and Uhl, 2011]. In their work, the authors use their statistical attack to attack the error correction used in the fuzzy commitment scheme, in particular, “soft decoding forms the basis of the proposed attack.” Soft decoding refers to a procedure in which the error correction decoding always returns the nearest codeword.

In [Rathgeb and Uhl, 2011], which takes its methodology from the histogram based attack originally proposed in [Stoianov et al., 2009], the authors use randomly extracted feature vector chunks from impostors to perform successive decommitments. Due to the use of soft decoding, a codeword will result from each decommitment. The codewords are counted and histograms are generated. Based on the resulting histograms, the most likely error correction codeword for each chunk is found. When speaking of the decoding attack against the fuzzy commitment scheme, the authors say that it “is simple to implement and very effective.”

In [Kelkboom et al., 2011], the authors look at different decoding attacks presented against the fuzzy commitment scheme. The authors apply a “random bit-permutation
process” to secure the scheme from attacks such as the decoding attack. With the addition of the randomization process, the authors show that they neutralize the decoding attack. However, the authors point out that, with their additions, a successful cross-match attack will reveal the enrolled binary vectors. A cross-match attack is similar to the ARM attack in that it uses multiple templates. However, the cross-match attack focuses on templates from the same user.

### 3.3 Privacy in Perspective

This section will explore the privacy concerns as they relate to a single individual system and as they relate to a community (loosely connected network of autonomous systems). We will look at how the privacy concerns differ for a single system and for a connected community. We define a single system as a system living inside a bubble with no concern for any other biometric system being in existence. A community of systems is the current interconnected world of today.

The security and privacy concerns of a single biometric verification system differ from those of a community of systems. An individual system would not need to worry about the same types of attacks as a community. For example, an individual system would not need to worry about ARM (correlation based) attacks. These types of attacks take advantage of the correlation between multiple templates generated by the same biometric data. If given multiple templates, ARM based attacks reveal information about the original biometric
used to generate the templates. In an individual system, there exists only one template per person, so this attack is not feasible. The same is true for cross-match attacks. In an individual system, a user would only need to enroll once and therefore would only have one template. With only one template, the cross-match attack would not be possible. In a community of systems, the same person can enroll in multiple places, allowing attackers to potentially obtain multiple templates generated by the same biometric, making this attack feasible.

Trust becomes a different issue when looking at an individual verification system. An individual biometric verification system would need to collect the biometric information from all of its users to use for verification. Any user of the system can trust that their biometric information will not be used in any other application because none exists. The system can trust that the data obtained from each of its users is authentic because it collected the biometric data from the users.

In a community of systems, the user cannot trust the system and the system can not trust the user. The user can not trust that the data is being maintained securely once it leaves their possession. With traditional biometric systems in a community, the user must give the key to the system for decryption. If an attacker accesses one of their templates, the attacker can use that template to try and recover their biometric data for use on another system. Also, often times in a larger community of systems, each individual system will not directly collect the biometric data from each user. The system needs to trust data
coming from an external source that is beyond their control; since the external source is beyond their control, however, the system can not explicitly trust it.

With an individual system, the detection of malicious access attempts is a smaller problem. If the total number of people having access to the system are the people enrolled in the system, the task of discovering malicious access attempts becomes a simpler task. As seen by the large and growing biometric security industry, detection of malicious access attempts is a large problem in biometric security systems in a community.

In an individual system there exists only one system for attackers to focus on. Being the sole focus of attack means that the system’s defenses must be strong enough to withstand such attacks. In a community of systems, not every system has the same security needs; for example, a gym access system has lower security needs than a missile defense system. Having multiple levels of security gives way to low hanging fruit, which makes an attackers job easier and security a more difficult problem.

Insider attacks are a problem for both individual systems and systems inside a large community of systems. If an insider breaks into an individual system and compromises it, the users will lose their ability to verify on that system. The problem is greater, however, in a community of systems because, if an insider breaks into one system in the community, they can potentially access other systems as well.

While an individual system where the users all must submit their biometrics directly into the system is more resilient to different attacks, it is not a realistic view. With the
examples given, its easier to see that the needs of the many outweigh the needs of the few [Bennett et al., 1982].
Chapter 4

Know Without Knowing

How can a person (or a server) know something without knowing it? To begin to answer this question, we must first answer two questions:

- What is it to have knowledge?

- What does it mean to know someone?

In this chapter, we explore what it is to know something or someone, to have assurance in that knowledge, and to not actually know them. This is the essence of remote biometric verification. Before we get into the chapter, we must answer the questions presented above.

Knowledge can be defined in many ways. Generally speaking, knowledge can be defined as awareness that is gained by experience; therefore, to have knowledge of something is to have experience with it. In terms of biometrics, knowledge of a biometric
comes from encountering it. Put simply, to have knowledge of someone’s voice, you must first hear it; to know someone’s face, you must first see it.

We define knowing someone as “the ability to verify information about the person.” For two people to know each other, they would need to verify their identities to each other. Recall the scenario where Alice is accessing her bank account. If Alice walks into her bank, the teller can physically see and interact with her. This type of in-person verification is the simplest case. A slightly more complicated scenario occurs when Alice calls the bank. In this scenario, a teller answers the phone, hears Alice’s voice, and recognizes her as she speaks. If the teller does not recognize Alice from her voice alone, she can ask Alice questions that only Alice should know the answer to. In this type of centralized verification, Alice gives the teller knowledge of herself so the teller can know it’s really Alice.

In remote biometric verification, Alice would use a remote device to verify with a server in such a way that she did not reveal anything about her biometric to the server. Simply put, the server must know you without having any knowledge about your biometric: to know without knowing.

### 4.1 Introduction to Remote Verification

Remote verification is one of two main verification paradigms, along with centralized verification. The two different verification paradigms have unique challenges associated
with them. In this section we look at what makes remote verification different by first looking at centralized verification for comparison.

Figure 4.1 gives a visual representation of centralized verification. In centralized verification, there is a single location that must maintain all necessary data for verification. For centralized verification, a user must send their verification information to the centralized location so it can be verified.

The easiest way to visualize centralized verification is to think of call centers. With call centers, a user verifies their identity by calling in and answering a series of questions. Users are verified based on their answers to the questions. In centralized biometric verification, the biometric data is sent to the server, exposing it.

Alternatively, in remote verification, the verification takes place on a remote device instead of at the centralized point of verification. This leads to questions of trust between the server and the remote device. How can the server trust the responses from the remote device? How does the server know if it is the correct person using the remote device and not an attacker? We examine a novel challenge-response protocol that answers these questions.
4.2 Challenge-Response

This section explores the concept of challenge-response and how it is used to solve the problems associated with remote verification as presented in this thesis. A problem that challenge-response solves is allowing the server to trust the responses from the user. Only the correct user should be able to correctly answer the questions presented in the challenge. Therefore, by challenging the user, the server gains confidence in the authenticity of the user.

Challenge-response is not only used in biometrics and security; it has broad uses because it allows one part of the system to challenge another part of the system to prove its authenticity. For example, [Liang and Wang, 2005] examined challenge-response in wireless networks in which a mobile user must authenticate using a challenge-response when on a public wireless network. Recently, [Rzasa et al., 2012] examined challenge-response authentication in control systems, where supervisory and plant control units authenticate with each other using challenge-response.

In a remote verification scenario, the server can not implicitly trust the client. Given that the server can not trust the client, there must exist a way for the client to build trust. Challenge-response is a simple way for the client to build the server’s trust. The basic challenge-response process involves an initial information exchange to later verify against, the enrollment stage, and a verification stage.
There are a number of issues that need be addressed during enrollment. The first, and possibly most important, is how the server knows that the identity of the individual enrolling matches reality. As mentioned in Section 2.1.2, Trusted third parties (TTP) were created to solve this problem for public key systems. The recent work done in biometric key infrastructure (BKI), as detailed in [Albahdal et al., 2013], looks to incorporate biometrics into public key cryptography. The BKI would allow biometric verification information to be stored in certificates held by TTPs, giving servers the ability to verify an individual without needing to enroll them directly (collect their biometrics directly). Short of the BKI, the server can not reliably verify the identity of the user without collecting their own samples. However, the server can verify that the claimed identity belongs to the same individual from enrollment with challenge-response. In its simplest form, challenge-response can be represented in terms of a dialog between two parties. For more information on the BKI, we direct the reader to [Scheirer et al., 2010] and [Albahdal et al., 2013].

### 4.2.1 Basic Challenge-Response

To properly explain the challenge-response process, we must first give some definitions. Recall the scenario in which Alice is accessing her bank account. When Alice signed up for her account, she first needed to give the bank her information (enrollment data) for later verification. Given Alice and her bank, let us define Alice’s enrollment data,
$D$, where $D$ consists of multiple pieces of individual enrollment data, $D_i$. Each piece of enrollment data is a pair consisting of a challenge, $C$, and a response, $R$. Let $E$ be the full enrollment template. Let $N$ be the total amount of enrollment information given to the bank such that the following holds:

$$E = D_1 \cup D_2 \cup \ldots \cup D_N \quad \text{where} \quad D_i = (C_i, R_i) \quad (4.1)$$

This says that for a full enrollment with the bank, a user must give $N$ pieces of enrollment data to the bank. As such, when Alice enrolled, she gave her bank all the necessary information needed for a full enrollment.

Verification is similar to enrollment, but differences exist. During verification, the bank sends Alice a series of challenges to which Alice responds, and the bank compares her responses against the responses given during enrollment. Realistically, users do not have to give every piece of information the bank has on file from enrollment in order to verify. Instead, the bank usually requires some subset of the verification data such that:

$$V \subset E \quad \text{where} \quad V = D_1 \cup \ldots \cup D_M \quad (4.2)$$

where $V$ represents verification, $M$ is the number of questions asked, and

$$M = m + \sum_{j=0}^{m} R_j \quad M \leq N \quad (4.3)$$
where $m$ is a lower bound on the number of verification questions asked. These equations mean that, for a user to verify with the bank, they must provide enough verification data; based on the number of correct responses given, the user might have to provide additional data.

The dialog occurs as data is sent back and forth between the bank and the user. A visual example of the dialog between Alice and the bank during enrollment and verification is given in Figure 4.2. During enrollment, the bank saves the responses given by Alice for later verification. During verification, the bank uses the data from enrollment to verify the new responses. By matching responses from challenges sent to the user claiming to be Alice, the bank builds confidence that the user is, in fact, Alice.

But what, if anything, does this dialog really confirm about the user? This type of challenge-response confirms nothing about the actual user, only that the user possesses the correct information to answer the questions.

Traditionally, challenge-response protocols are used to validate something the user knows. There are also variants of challenge-response where the data is tied to a device which allows the server to validate something the user has. The work of this thesis details a novel challenge-response protocol to also incorporate something the user is.
Figure 4.2: Challenge-Response question and answer dialog. In the enrollment dialog, 4.2a, the bank saves the info received from the client. In the verification dialog, 4.2b, the bank uses the responses from the client to check against the previously stored responses.
4.2.2 Biometric Challenge-Response

The basic challenge-response allows the server to trust the authenticity of the remote user. It does not, however, address the identity of the remote user. What is needed is a way to prove the identity of the individual. Including biometrics in this process allows for proof of the individual’s identity during the challenge-response process. While the inclusion of biometrics in the process allows the identity of the individual to be included, it also leads to more questions.

The first question revolves around how the biometric data will be incorporated into the process. The second focuses on how the biometric data will be obtained for verification. Another question involves the security the user must give up and the trust the user must posses in order to send their biometric data to a server for verification.

To include biometric data into the challenge-response process, we must first consider the medium. A mobile device, the focus of this thesis, has only so many means of collecting biometric data. To date, the most common ones are the camera, the microphone, and the recently introduced fingerprint scanner. For now, let us consider them equivalent in the sense that they all collect some biometric data. Once the biometric is collected, it must be included into the challenge-response process.

As mentioned previously, the challenge-response process consists of a dialog between two parties, Alice and her bank in the example. There are two different ways to include
Alice’s biometric data into that dialog: all at once, or in pieces. Regardless of how Alice’s biometrics were included, the server would need to store the biometric data sent during enrollment and verify against it during verification.

Current biometric verification protocols are designed to give a yes/no answer or release a key if a match occurs. If Alice incorporated her biometrics into the challenge-response process with current techniques being used, the result would be either single or multiple yes/no answers on her biometric data. The responses would be static or based on static information, and therefore would be subject to replay attacks. Also, since her data would be sent to the server, her data could be copied in transit and used to impersonate her at a later time or on another server.

The idea of matching on a remote device, allowing the server to trust the match, including biometrics in the matches, and not needing to send any biometric data is the motivation behind this thesis. However, many questions remain. How can this be accomplished without leaking information to the server? How will the data be hidden? Is it even possible to remotely verify using biometrics without sending any biometric data from the device? To answer these and other questions we first look to existing literature. Vaulted Verification, of which this work borrows its initial concepts, is a step toward answering some of these questions.
Figure 4.3: Visual example of Vaulted Verification enrollment [Wilber and Boult, 2012].

4.3 Vaulted Verification

As with any client-server verification process, the Vaulted Verification process utilizes two main steps: enrollment and verification. Vaulted Voice Verification shares its base concept with the Vaulted Verification protocol, so this section will provide a general overview of the Vaulted Verification process. Vaulted Verification is described in greater detail in [Wilber and Boult, 2012] and [Wilber et al., 2012].

We first define some basic terms to assist in understanding the Vaulted Verification ($V^2$) process. $C$ refers to the client software running on the mobile device. $U$ refers to a person using $C$. $S$ refers to the machine/device to which $U$ is trying to gain access. $K_S$ and $K_U$ refer to the encryption keys of $S$ and $U$, respectively.
During the enrollment processes of $V^2$, $U$ gives a biometric sample to the $C$. $C$ takes that sample and creates feature vectors from the sample. The feature vectors are then sectioned into blocks. $C$ then creates a chaff block for each of the feature vector blocks in such a way that it is indistinguishable from the real feature vector blocks to anyone but the user with the same original features. In [Wilber and Boult, 2012], Wilber et al describes the chaff blocks as being created by randomly taking a corresponding feature vector from a different person. As shown in Figure 4.3, each block is then encrypted, first using $K_U$ and again using $K_S$. The blocks are then sent to $S$ in such a way that $S$ knows which of the blocks are real and which blocks are chaff.

As can be seen in Figure 4.4, when $U$ logs in during the verification process, $S$ will send $U$ a locked challenge. For this, $S$ creates a random binary bitstring of $n$ bits such that roughly $n/2$ bits are 1’s. $S$ then sends $n$ block pairs to $C$ in an order that depends on
the value of the binary bitstring; if the bit is a 0, the real block is sent first, and if the bit is a 1, the chaff block is sent first. \( U \) then submits an image to \( C \). \( C \) will then decrypt and analyze each block pair and respond with a 0 or 1 for each pair depending on which \( C \) believes is real and which is chaff.

In this scenario, the blocks stored on \( S \) are secure even if \( S \) is compromised because \( S \) does not have the encryption keys and, therefore, does not know the contents of the blocks. The only information \( S \) has is the knowledge of real vs chaff. \( S \) does not even concern itself with how the blocks are authenticated, just whether or not the responses are correct.

Vaulted Verification will provide security with privacy, but it does have its limitations. One of the main limitations of Vaulted Verification is the limited amount of information available in face and iris data. Vaulted Verification is not able to vary the data in the challenge-response process. Also, because of the limited amount of data, there are only so many challenge-responses that can be generated. The data limitations of Vaulted Verification that are imposed by face and iris data are the same for both face and iris, so this explanation will refer only to face as shown in Figures 4.3 and 4.4.

As a part of the enrollment, the image is split into blocklets. These blocklets are what get mixed with chaff. Choosing between the real and the chaff blocklets is the basis for vaulted verification. When sectioning the image into blocklets, a large enough section must be chosen so differentiation can happen. If the sections are too small, the real and
the chaff will blend together, defeating the purpose Vaulted Verification serves. Inversely, the larger the blocklets, the smaller number of challenge-response pairs.

The limitation becomes notable when certain pairs are consistently guessed correctly by an attacker, which renders those challenges useless. The more pairs an attacker can consistently guess correctly, the weaker the security of the algorithm becomes. Because of the limited data, there is a limited number of pairs that can be generated, which in turn limits the effectiveness of the security system.

4.4 Combined Approach

Remote verification has the weakness of the server not being able to trust the matches that occur remotely. Basic biometric challenge-response has the weakness of needing to send biometric data off of the client’s device to be verified. Combining the two ideas gives a way to keep all of the strengths and none of the weaknesses of each. With their abilities combined, an interesting possibility emerges: the possibility of matching remotely without the need to send the biometric data to the server. In effect, this method ensures that the server knows that the user has biometrically verified themselves without knowing the biometric data of the user.

As shown, Vaulted Verification is a step towards solving the issues presented in this work. This work takes the concepts of biometric challenge-response with the concepts of remote verification as presented in the introduction of Vaulted Verification and builds
on them in several ways. Where the original work in Vaulted Verification concentrated on physical biometrics such as iris and fingerprint, this work focuses on a behavioral biometric, voice.

There needs to be a solution where the biometrics are incorporated into the verification process, the biometrics do not leave the device, the user’s privacy is preserved, and the protocol is efficient enough to run on a mobile device. The simple combination of challenge-response and remote verification can not solve these issues alone, but Vaulted Voice Verification is able to accomplish all these tasks.
Chapter 5

My Voice Is My Key

This section will describe the general procedure of how Vaulted Voice Verification will be utilized in the creation of this novel scheme. Vaulted Voice Verification improves upon Vaulted Verification in multiple ways. Vaulted Voice Verification can expand the available data using different words and phrases. If some models are compromised, they can be discarded, and other models can be generated using new words and phrases. An additional improvement is a further generalization of Vaulted Verification via its application using voice. Vaulted Voice Verification also has enhanced variability and security through the question selection process. With Vaulted Voice Verification the server, $S$, does not know what the questions or the answers are; only that for some encrypted question, there is a correct response among the encrypted choices. Finally, Vaulted Voice Verification will be easy to use and implement on a mobile platform because there is no need for specialized
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>The server</td>
</tr>
<tr>
<td>$U$</td>
<td>The User</td>
</tr>
<tr>
<td>$D$</td>
<td>The user’s mobile device</td>
</tr>
<tr>
<td>$K$</td>
<td>Keys generated and used for encryption and decryption.</td>
</tr>
<tr>
<td>$K_U$</td>
<td>The user’s key</td>
</tr>
<tr>
<td>$K_S$</td>
<td>The server’s key</td>
</tr>
<tr>
<td>$M$</td>
<td>Message being sent/received</td>
</tr>
<tr>
<td>$C$</td>
<td>The set of challenges</td>
</tr>
<tr>
<td>$c$</td>
<td>An individual challenge</td>
</tr>
<tr>
<td>$Q$</td>
<td>The set of questions</td>
</tr>
<tr>
<td>$q$</td>
<td>An Individual question</td>
</tr>
<tr>
<td>$R$</td>
<td>The set of responses</td>
</tr>
<tr>
<td>$r$</td>
<td>An individual response</td>
</tr>
<tr>
<td>$P$</td>
<td>The password</td>
</tr>
<tr>
<td>$=&gt;$</td>
<td>Transmission of some form (speaking to, typing in, sending over network)</td>
</tr>
<tr>
<td>$&lt;=$</td>
<td>Back and forth communication</td>
</tr>
</tbody>
</table>

Table 5.1: Symbol definitions
hardware beyond a microphone, which all mobile phones have.

Initially, Vaulted Voice Verification was an attempt to see if the general concepts and ideas from Vaulted Verification could be applied to the voice space. After the foundation of Vaulted Voice Verification was created, it was extended to incorporate a greater diversity of challenges as could only be done via voice. Finally, with the index-based version, we solve three of the major problems associated with the initial implementations: models leaving the control of the user, communication overhead, and key generation suitable for public keys.

This section includes a number of symbols. Table 5.1 lists all the symbols and their definitions for clarity.

### 5.1 Vaulted Voice Verification

One of the biggest issues in biometric systems is the security of enrollment and verification processes for biometric data. Once a biometric sample (scan, recording, picture, etc.) has been obtained by the system, the information has to be not only saved securely, but the system also has to be able to verify against it, or the resulting saved data, with a second sample in order to verify the validity of the second sample. To solve the security issues with the processes, there are a few requirements that must be met. [Scherier and Boult, 2007] These requirements are:
1. Ensure that the biometric can not be compromised if the server is compromised and the model is acquired by an attacker.

2. Prevent man-in-the-middle and replay attacks. A “man-in-the-middle” is an attack where someone intercepts the communication from the server to the client and replaces the information on both sides. This is similar to the “replay” attack where an attacker saves the captured information to be used again later.

3. Ensure that the system has revocability. If the user needs to invalidate their current model or something happens where the model is compromised, the user should be able to invalidate their current model and issue a new model.

4. Withstand blended substitution attacks. In the blended substitution attack, an attacker combines their template with the stored template so that both the user and the attacker can authenticate using the same template. The system should be able to detect and prevent someone from substituting a user’s model with their own.

The prior work on the Vaulted Verification technique addresses the previously mentioned requirements. In the current work, we are extending this to Vaulted Voice Verification, a novel challenge-response protocol appropriate for voice based verification. The Vaulted Voice Verification protocol is a new challenge-response approach to authentication which is well suited to mobile devices. The problem Vaulted Voice Verification truly solves is how can this be done in a secure way while also preserving the privacy
of the user. One of the advantages of Vaulted Voice Verification is that security can be increased by combining speech content with the challenges. As an example, Figure 5.1a shows multiple people trying to access a system while all claiming to be Bob. Using the novel Vaulted Voice Verification protocol, a simplified version of which is shown in Figure 5.1b, the server is able to use the challenge-response pairs to ensure the authenticity of the users. This example shows the server challenging the users by having them speak three words/phrases. The server sends pairs of models to the user, with one authentic and one chaff model for each phrase. The user must decide which of each pair is authentic based on their response, 0 if they think the real is first and a 1 otherwise. In Figure 5.1b the top user should respond 100, the middle user should respond 110, and the bottom user should respond 001. The authentic user should have no difficulty choosing the correct bits while impostors have to guess. There are other layers of security; the algorithm is described in greater detail in section 5.1.3.

5.1.1 Enrollment Process

The first thing that must happen is enrollment. A simplified version of the enrollment process is shown in Figure 5.2. The steps mentioned in this section refer to the numbered arrows in Figure 5.2. For layered security during transmission, \( M \) will first be encrypted with private \( K_U \) and again with public \( K_S \). \( K_S \) is obtained when initial contact for enrollment is made from \( D \) to \( U \). \( U \) needs to give enough information so they will be able
(a) The Problem: Multiple users trying to access server as Bob.

(b) The Answer: Ask them to say words/phrases and match real vs imposter.

Figure 5.1: Purpose of Vaulted Voice Verification

Figure 5.2: Initial Vaulted Voice Verification - Enrollment Process.
to verify their identity at a later time. The process involves $U$ supplying $S$ with some different information.

The first step, labeled step 1, is for $U$ to give $D$ identifying information, so $S$ can identify $U$ with their data. $U$ tells $D$ their name and possibly other pieces of identification that $S$ will be able to use during the verification process. $U$ also either inputs $P$ or a random $P$ is automatically generated and stored on $D$. Next $D$ gives $S$ the identifying enrollment information. This information is also encrypted with $K_U$, so the only thing $S$ knows is an id and a hash.

$$U(id,P) => D$$  \hspace{1cm} (5.1)

In step 2, $C$ asks $U$ to repeat a series of phrases/questions. These phrases are what $S$ will use to challenge $D$ during verification. The questions are designed to get the maximum amount of information out of the fewest number of phrases. The goal of the questions is to be able to generate a prescribed number of challenge-response pairs, thus having enough bits to give proper identity security.

$$S(q) => D(q) => U$$  \hspace{1cm} (5.2)

As step 3 illustrates, $D$ will create adapted models from these responses. The models are created as GMMs. Each GMM contains a number of distributions, $Dis$, and each of
those distributions has a number of components, $Com$, for a total number of components of $N = Dis * Com$. How these models are used and compared will be detailed in section 5.1.2.

The models and transcriptions of the responses are stored together. Each of the model/transcription pairs are encrypted as a block of data. Step 4 shows the models as they are first encrypted with $K_U$, and then again with $K_S$. This will ensure that $S$ has no knowledge of what the responses/questions are. The only thing $S$ is aware of is that, for some encrypted question, there exists a correct response among the encrypted responses provided.

$$U(r) \Rightarrow D$$  \hspace{1cm} (5.3)

$S$ then receives and stores the challenge-response pairs for future verification, illustrated in step 5.

### 5.1.2 Verification Process

The verification process is initiated when a user attempts to gain access to the system. A simplified version of the verification processes is shown in Figure 5.3. All steps mentioned in this section refer to Figure 5.3. In this scheme, a user responds to questions asked by $D$. This happens in the steps as follows.

First, $U$ tells $D$ that they would like to log in and access their data. At this point, $D$
does not know who $U$ is or even who $U$ is claiming to be. $U$ tells $D$ who they are by either entering their name or speaking their name/id. This is illustrated in step 1.

$$U(id) => D \quad (5.4)$$

Then $D$ asks for the initial password/passphrase. While this happens, $D$ also sends a request to $S$ for $U$’s information, as illustrated in step 2. $D$ uses $K_U$, derived from $P$, to decrypt the transmissions from $S$. If $D$ does not have the correct $K_U$, there will be no
way to get the rest of the information for the challenge responses, ending the process.

\[ U(P) \Rightarrow D, D(id) \Leftrightarrow S \] (5.5)

Once \( D \) is able to decrypt the information packet \( M \) from \( S \), \( D \) compares the response to the information received from \( U \).

\[ K_U(M) = id \text{ or } K_U(M) = 0 \] (5.6)

If \( D \) is able to decrypt \( M \) using \( K_U \), the challenge response process begins. In the challenge response process, \( S \) will send the \( Q \) to \( D \). \( Q \) is comprised of reordered pairs of \((q,r)\) and is appended with a nonce. The pairs, as illustrated in step 3, have been reordered based on a random binary string that \( S \) generates.

\[ S(Q) \Rightarrow D \] (5.7)

Step 4 illustrates \( S \) sending encrypted \( Q \) to \( D \). \( Q \) is block encrypted with public \( K_U \) before transmission. This ensures that the blocks are different with every transmission based on the random binary and the nonce. In step 5, \( D \) decrypts and displays/asks \( Q \) to \( U \), one at a time, \( q \).
\[ D(q) \rightarrow U \]  \hspace{1cm} (5.8)

\( U \) then responds to \( q \). The response depends on the type of question asked. If the question is multiple choice, \( U \) would speak the correct answer. If the question is a passage to read, then \( U \) simply reads what is asked.

\[ U(r) \rightarrow D \]  \hspace{1cm} (5.9)

\( D \) processes the response into a model. A decision is made by comparing the response from \( U \) to the options presented from \( S \). Step 6 illustrates an example of two choices that could result from comparing real and impostor/chaff models. When comparing the similarity of the two models, probe (\( pr \)) and gallery (\( g \)), either the variance of \( pr \) or \( g \) will be used depending on if the intention is 1) to see how similar \( pr \) is to \( g \) or 2) how similar \( g \) is to \( pr \). Observe that for each gaussian mixture component, there is a mean \( \mu \) and a variance \( \sigma \). In equation 5.10, we are seeing how similar \( pr \) is to \( g \), so we utilize the variance of \( g \). As shown in equation 5.11, a final score is the summation of the z-scores over the total number of components, \( N \).

\[ Z_i = \frac{x_{pr} - \mu_g}{\sigma_g} \]  \hspace{1cm} (5.10)
Once the models are compared, \( D \) will make a decision as to which of the models from \( S \) is closest to the model generated from \( U \).

Illustrated in step 7, \( D \) will send a binary encoded response back to the server. How this response will be encoded depends on the type of question. If the question has two options, then 0—1 are appropriate. For a multipart question, multiple bits may be required. The process will repeat until reaching a stopping condition. The stopping condition will generally be the required number of bits being reached.

After this exchange has finished, \( D \) will have generated a large bitstring to send the server. The bitstring will then be evaluated by the server. If the score is above the threshold, access will be granted, illustrated in step 8.

\[
Score = \sum_{i}^{N} (Z_i)
\]  
\hspace{1cm} (5.11)

\[
D(\text{bitstring}) \Rightarrow S
\]  
\hspace{1cm} (5.12)

\[
S(\text{yes or no}) \Rightarrow D
\]  
\hspace{1cm} (5.13)
5.1.3 Initial Analysis

How is Vaulted Voice Verificationable to solve the security issues mentioned in Section 5.1? As with most secure biometric systems, this protocol contains many layers of security.

After enrollment, the client never sends out non-encrypted data; the matching is done on the client after decryption with the user supplied password. From enrollment to verification, it is assumed that the communication is happening over a secure encryption protocol, SSL or TLS for example, and an attacker cannot eavesdrop on the communication. If the encryption is maintained, the system is secure, so we consider various levels of compromise.

If an attack is somehow able to break the encryption on the communication, and the attacker tries to impersonate $S$ and stage a man-in-the-middle attack, the attacker would not be able to gain any additional information about the biometric to give them access. This is because, even if the attacker is able to see the data from $D$ to $S$, the data is still encrypted with $K_U$, as illustrated in step 4 of Figure 5.2.

An attacker could also attempt a man-in-the-middle attack by recording the encrypted and randomly ordered data pairs and responses. This would prove pointless because $Q$ is randomly ordered, has a nonce added, and is block-encrypted with $K_S$ for every session, making the transmitted blocks different for every session. Thus, any reordering would fail
to decrypt and no information is gained by examining it since the recorded information would be different for the next session. With this, there is effectively no man-in-the-middle.

If the device was lost, it provides little information as it just has an encrypted block of data that the client cannot decode. If an attacker were to obtain both the server’s private key and the user’s device, the attacker might be able to authenticate on that particular server. This is because they would be able to decrypt the models and, through trial and error, return the correct response to the challenges by examining the differences when models are swapped. However, because this is a phrase-based scheme, this would not allow an attacker access the user’s data on a different server. The reasons are two fold. The first is because the models are encrypted using the user’s private key. Without being able to access the models, the attacker would not be able to discern any information about it. The second is that, without recordings of the correct phrases, the attacker would not have the necessary information to discern between real and chaff data. If the device is lost, the enrolled model can and should be revoked, changing the encryption keys and phrases.

The worse case scenario for this protocol would be for the attacker to have access to $K_S$ and $K_U$. The attacker would then be able to authenticate as $U$ on $D$, but would still not know which of the $N$ blocks are real and which are chaff. On average, an attacker with all the keys must still make $2^N$ guesses to correctly identify the biometrics. At 56
blocks, that is $2^{56}$ attempts, which is not too difficult for computers today. If the number of blocks were increased to 256, with current computational models, this is sufficiently secure. Also, the user can still just revoke and reissue with new phrases and keys.

Though there could exist statistical differences in the models generated from a given user versus the general models, if chaff is chosen correctly, this should not be an issue. With this, we are assuming independent probabilities for each bit. That gives a probability of between $2^{-8}$ and $2^{-12}$ for an attacker with no foreknowledge, using a random chance brute force attack, to gain access. This is on top of encryption, which means it is an improvement, but we want better security than this.

To increase the security, we would need to increase the number of bits. One of the ways to do this would be to use multiple impostors instead of the one dedicated impostor. This would increase the number of bits based on the number of options. For example, if there were 3 impostors per question instead of 1, there would be 4 options from which to chose. The probability for each question would be $2^{-2}$, effectively doubling the number of bits per question. For the same 12 phrases, there would then be a probability of $2^{-24}$ that an attacker could authenticate as $U$ and gain access.

Again, it is necessary to note that this security analysis of Vaulted Voice Verification is speaking in terms of how much security is added on top of encryption. Vaulted Voice Verification would be providing $K$-bits of “identity verification” security in addition to the $N$-bits of security offered by encryption. Since the equal error rate provides a lower
bound on identification accuracy because of the associated biometric dictionary attack, the identity security offered by Vaulted Voice Verification is already much better than the prior state of the art.

5.2 Mixing Text-dependent and Text-independent Models

An issue with voice-based biometric verification protocols, such as Vaulted Voice Verification, is that they need to get as much information as possible out of as little user interaction as possible. If it takes a user 15 minutes to authenticate every time they use the system, they will soon stop using it. We need to address this concern specifically by looking at the number of bits that can be generated by a single user interaction. These “bits” represent the amount of information needed for model selection as defined by the protocol.

Another issue is that, with text-dependent voice templates/models, a system is susceptible to different voice conversion-based attacks. Such attacks are described in [Kinnunen et al., 2012]. As shown in [Alegre et al., 2012], text-independent voice templates/models are also vulnerable. Our work to extend Vaulted Voice Verification to include both text-dependent and text-independent-based models seeks to eliminate such vulnerabilities.

The main contributions of this work are:
• The inclusion of text-independent speaker models into Vaulted Voice Verification to increase the security.

• An increase of the security gain given per question.

• A security analysis of the Vaulted Voice Verification protocol in terms of attack models.

• Analyzing the accuracy vs. security trade-off from text-independent models.

In more detail, Vaulted Voice Verification, as described in [Johnson et al., 2013b], is a challenge-response protocol that uses a mixture of GMMs created from a user’s audio recordings and chaff GMMs created from recordings of other users or by modifying existing GMMs. The system presents the user with a series of phrases to repeat and subsequently generates models from the responses. For each model generated, a chaff model is also generated. Multiple methods exist to generate chaff models and deciding how exactly to generate chaff is often more art than science. Examples of chaff generation include perturbing models generated from the response of the user and using real models from the same user but from a different response. The idea is that an attacker would not be able to distinguish between the real and chaff model, i.e. only the voice that created the real model could be used to distinguish the real model from the chaff.

As with other verification systems, Vaulted Voice Verification includes both an enrollment and a verification process. We will introduce elements of the original Vaulted Voice
Verification below for the purpose of providing the necessary background to our novel extension.

RSA encryption is used to generate the server and the user key pairs in both the original and our extended version of the protocol for both enrollment and verification. Except where noted, when the user transmits data to the server, that data is first encrypted with the public key of the server, so only the server can decrypt it using its matching private key. Likewise, when the server transmits data to the device of the user, that data is first encrypted with the public key of the user, so only the user can decrypt it using their matching private key.
5.2.1 Enrollment

The enrollment process for our extended version of Vaulted Voice Verification is illustrated in Fig. 5.4. The steps described here refer to the numbered arrows in the figure. Differences in the original versus our extended version will be pointed out for both the enrollment and verification process.

In step 1, the user enters their information into their device. Then, in step 2, the device generates keys for the user. These steps follow the steps of the original protocol.

For steps 3 and 4 in the original protocol, the device interacts with the user, asking the user to repeat series of phrases to which the user responds. In our extended protocol, we use both text-dependent and text-independent prompts for the user. For the text-dependent mode, the device prompts the user with short phrases or small passages to repeat. For the text-independent mode, the device shows images that need a short and non-scripted description. For each of these prompts, a real model and a chaff model are created so the chaff model is similar to the real model.

In step 5, the device encrypts the models using the public key of the user. The encryption of the models occurs in the same manner as in the original Vaulted Voice Verification protocol. The public key of the user is used here, so only the user can decrypt the models with their private key.

In steps 6 and 7 of the original protocol, the encrypted models are sent to the server.
for storage until verification and are subsequently removed from the client device. In our extended protocol, once the server receives the models, it creates hashes for later verification, and the models are then deleted from the server. As a result of this, the encrypted models are able to remain on the client device until verification.

Lastly, in steps 8 and 9 of both the original and the extended protocols, the user receives a notification of the success/failure of enrollment.

5.2.2 Verification

The verification process for our extended Vaulted Voice Verification is illustrated in Fig. 5.5. The steps described here refer to the numbered arrows in the figure. Again, differences between our extension and the original will be noted.

In steps 1 and 2, the user inputs their information into the device, which sends a verification request to the server. In the original protocol for step 2, the device sends the ID and a request for verification. In our extended protocol, the device also sends the encrypted template because it is no longer stored on the server.

In step 3 of the original protocol, the server retrieves the models associated with that user and scrambles them according to some i.i.d. binary challenge string. In our extended protocol, the server first verifies the template by hashing it and comparing the hash against the previously stored value from enrollment. This allows the server to be sure it is the same data it received during enrollment without the need to store the data itself.
Steps 4 and 5 show the server sending the shuffled models to the client, which decrypts them using the information provided by the user in step 1. In steps 6 and 7 of the original protocol, the user interacts with the device, repeating the phrases as prompted. Our extension expands this interaction to include passages of text and images that must be described by the user.

In step 8, the device generates a new model from each response for each phrase from the user. These new models are used to select between each real and chaff model for the prompted phrases. As the device selects between the two to unscramble the models, it builds a response string. In step 9, when the device finishes building the string, it sends the response string to the server for verification. The device then compares the response string to the original challenge string. The remaining steps illustrate the server responding with an accept or decline decision based on whether the response string matches the challenge string.

While Vaulted Voice Verification provides a novel combination of techniques from two communities, the use of only text-dependent models makes it susceptible to the attacks mentioned in [Kinnunen et al., 2012]. Also, the use of binary models leads to either too many questions for the user to answer in a practical interactive system, or a system that is easily compromised due to the small number of bits of security that is provided.
5.2.3 Improving Security and Usability

In this work, we extend Vaulted Voice Verification beyond text-dependent modeling and single bit questions and answers. We look at mixing text-dependent modeling, where the models are based on specific word snippets, with text-independent modeling, where models are generated using larger word groupings. Also, we enhance the amount of information gathered from each question in the challenge response by extending from binary to multiple choice.

5.2.4 Text-dependent and Text-independent Models

Our work of mixing text-dependent and text-independent modeling is similar to that of [Sturim et al., 2002] and [Boakye and Peskin, 2004], where they focused on small
Where were you born?

Figure 5.6: How would you answer this question? Would you say “C,” “France,” “A hospital,” “Colorado,” or something else? This section looks at the potential of questions such as this one to augment the security of voice biometrics. Specifically, we focus on expanding the recently introduced Vaulted Voice Verification protocol [Johnson et al., 2013b].

In the previous section, as well as in [Johnson et al., 2013b], we had the user respond by repeating certain phrases. With that system, the phrases are known both during the time
Figure 5.7: Vaulted Voice Verification is extended by using open-ended challenges in the form of images instead of phrases the user must repeat. For example, how would you describe this image?

of enrollment as well as during verification. With this, the system is limited to the number of predetermined phrases it starts with and the challenges are simple in their complexity, as well as in the security they provide. The increase in security resulting from the addition of the text-independent models as described in this section will be explored in Sec. 5.4.

In this work, we look to extend the protocol from presenting the user with phrases to speak to showing the user images and asking them for short descriptions of the images. Examples of this can be seen in Fig. 5.6 and Fig. 5.7. In Fig. 5.7, we have an example of an image the user could be presented with. The idea is that different people will use different words to describe the same image. When we are using text-dependent models, the model implicitly incorporates the word and the voice model in the answer, thus improving the overall security. By doing this, we are able to create models that have a degree of freedom.
Multiple advantages exist with the proposed extensions of the protocol. The possibilities for the response to each challenge are as vast as the lexicon allows. This implicitly combines what you know with who you are. Since the models exist only on your phone, i.e. something you have, it is now a full three-factor authentication.

The protocol incorporates text-independent models to mitigate the threat of replay-attacks. By replay-attack, we mean an attacker recording audio responses and replaying them at a later time during an attempted attack. To do this, the server first generates a phrase and the user reads it; then the server both generates a text-independent speech model while performing speech recognition to verify proper content. This way, the phrase is unknown to the user until the server challenges them. This makes a replay-attack, even using a pre-recorded voice of the subject, impractical. The security not only relies on the general text-independent model, but also on the spoken description. More information on the different types of attacks our research works to defend against will be detailed in Sec. 5.4.

5.2.5 Increased Information from Responses

A challenge that needs to be addressed with voice based biometrics is how to get the most information from users without it taking so long that they do not want to use the system. Initially, with the original Vaulted Voice Verification protocol, the users would answer a series of questions, and each question would only generate one bit of security. We have
now extended the protocol so that each question is able to generate multiple bits. To achieve this, we turned each binary choice question into a multiple choice question. Our experiments used four-choice questions. This improves security by adding more bits of information to each question and makes the system less cumbersome and time-consuming for the user.

With this protocol, a portion of the security comes from the server scrambling data to generate a challenge. Given this, each challenge must contain a finite number of possible choices for the server to scramble and send to the client. This differs from traditional biometrics in that this protocol does not extract data from the response and compare it to a stored value; rather, it uses the response to choose between multiple possibilities as presented by the server.

In our experiments, this extension to four possible answers has doubled the amount of bits that are produced per question. For our experiments, we assume that the questions are independent of each other. We further assume that an attacker is choosing at random from the possible answers. These assumptions are for a naive attacker model. Sec. 5.4 will address a more sophisticated attack analysis. If the user is asked a series of five questions, the probability of getting them all correct based on the original Vaulted Voice Verification work would be $2^{-5}$, with one bit per question. With the extension implemented in our experiments, two bits are generated per question. For the same given questions, there would now be 10 bits of security. With this, the probability of randomly accepting the
identity would decrease to $2^{-10}$.

The number of questions for the system can be chosen to balance the chance of randomly accepting an identity with the accuracy of the voice biometric. We can then balance the number of questions with the biometric error rate (e.g. FAR). It is important to understand that the biometric error rate provides a lower bound on identification accuracy because of the associated biometric dictionary attack [Scheirer et al., 2010]. Thus, the identity security offered by Vaulted Voice Verification with only five questions is already much better than the state-of-the-art biometric error rate for voice [Inthaviras and Lopresti, 2012].

### 5.3 Index-Based Vaulted Voice Verification

The originally proposed Vaulted Voice Verification protocol is able to securely and remotely verify an individual using voice as a biometric identifier. In this work, we look to use the basic ideas from Vaulted Voice Verification to create a protocol that is able to securely and remotely identify an individual while exchanging keys that are suitable for secure communication, all while radically reducing communication overhead.

As with Vaulted Voice Verification, our index-based protocol for secure key exchange has both an enrollment and a verification stage. What makes this protocol truly novel is its use of indexed tables instead of sending models between the server and the client. We will examine the construction of the five different types of tables that must be created before
we detail how the protocol is used for secure key exchange. The tables are as follows: model table, client table, server challenge table, matching table and user table. During matching, the biometrics are combined with these tables to generate a set of results for the matching challenge. The five tables with sample data are shown in Section 5.3.3.

### 5.3.1 Index-based Enrollment

The enrollment process for our new index-based version of Vaulted Voice Verification is similar to the enrollment of the originally proposed Vaulted Voice Verification. As shown in Fig. 5.8, the enrollment process occurs in steps. In step 1, the user enters their ID and password. One option is for the user to supply both an ID and a password; another...
option is to base their generation on the client device (i.e. phone number and hardware ID), making the enrollment device specific.

In step 2, the client device generates the RSA key pair $K_{\text{priv}}, K_{\text{pub}}$ for the user. In step 3, the client device selects questions from a larger list, asks the user the questions, and the user responds. In step 4 the client generates models based on the user’s responses, one model per response.

Once the models are generated, the process moves to step 5, in which the client device generates the model table shown in Fig. 5.11 and described in Section 5.3.3. It takes the private key $K_{\text{priv}}$ from step 2 and a set of generated random pad $R$ and generates a xored version of the key $\hat{K} = K_{\text{priv}} \oplus R$. It breaks $\hat{K}$ into smaller segments, e.g. 32 bit segments, and stores the components in the client table as $N_i$ associated with hash value $h_{2,i}$ associated with correct model $i$ and associating a random $N_i$ with chaff pairs. It encrypts the model and client table with the user-device password. It takes the associated segment of the pad $R_i$, and adds the pair $(h_{2,i}, R_i)$ into a temporary server table. It also stores the question string for each pair – note the client table does NOT contain the question strings. Then the temporary server table is encrypted using the server’s public key before sending it to the server. The server then receives the table, decrypts it, and generates a hash based on the content. In step 6, the server stores the generated hash as the data hash for the user in the system users table. Then the server adds an enrollment date/time/ID, encrypts the resultant “server table” with its public key, sends it to the user in step 7 and may then
Figure 5.9: Index-based Vaulted Voice Verification: Verification. 1. Input user ID and password. 2. Use password to decrypt Saved Tables and Model files. Send ID and encrypted Server Table to request verification. 3. Decrypt and hash Server Table and verify against stored hash. Choose questions from within Server Table. Generate random string S. Shift rows of Server Table according to S, creating Challenge Table. Generate Challenge nonce N and hash entries with nonce. 4. Send Challenge Table to Client. 5. Use Client Table to turn Challenge Table entries into (Model, Index, Nonce). Locate models associated with each question. 6. Ask user questions. 7. Respond to questions. 8. Identify correct models via Responses. Unscramble Challenge using correct models. Compute Responses. Send a message using the generated Response as the Key. 9. Verify response.

delete its local copy of the server table. The data is encrypted with the public key of the server so the client is not able to access the content, but gives a copy to the client so the server does not retain any information other than the hash and user ID, reducing any risk from insiders trying to decrypt the data. In steps 8 and 9, the client stores the server table, deletes the temporary server table and notifies the user that the enrollment is complete.

5.3.2 Index-based Verification

The verification process for our new index-based version of Vaulted Voice Verification is a variant of the one originally proposed. Unlike the original, however, this manages only
indices and extracts a private key.

In step 1, the ID and password are obtained: either the user enters them or the device generates them. Next, the client device uses the password to attempt to decrypt the tables and model files. If successful, the ID of the user and the encrypted server table are sent to the server to begin the verification process.

The server receives the request, and, in step 3, decrypts the table and hashes it to compare the request against the previously stored hash of the table that was sent to the given user. Then the server selects a subset and permutation of the questions from the server table. The server then generates a string of random numbers for which each number is less than the total number of available answers for a given question. The length of the string equals the number of selected questions. Then the server cyclically shifts the rows of the selected questions from the server table according to the bit string. It generates a challenge nonce and appends it to the first elements of row pairs, hashing the results. The resulting shifted/rehashed subset of entries plus the challenge nonce form the server challenge table. Once the challenge table is complete, the server sends the table to the client, as shown in step 4.

Using the client table, the first value from the double in the challenge table, and the nonce from the challenge table, the client searches the challenge table for matching entries, then produces a triple with the hash $H_m$, index $i$, and potential key fragment $(N_i \oplus R_i)$. Using the triples and the model table, the client device locates the associated
model files, step 5. In step 6, the client device uses the questions from the challenge table to prompt the user, to which the user responds. In step 7, the client device generates models for the questions from the answer given by the user. Similar to Vaulted Voice Verification as originally proposed, in step 8, the client device then compares the models against the possible models. This allows for the identification of the correct triples.

Obtaining the correct triples allows for the identification of the shift of the rows of the challenge. The client device accomplishes this by examining the index variable of each triple. If, for example, the triple contains an index of 3, yet it is in position 1 of the matching challenge table, then the server applied a shift of 2. The client device does this for each row, generating a string of presumed shifts. The client device can also take the potential partial key fragments and combine them to obtain the private key. This could be simple reordering and concatenation, xoring, or, ideally, it can be accomplished using an error correcting polynomial.

From here, the client device computes its response. The response from the client device to the server comprises information only available after generating the presumed shifts. The client device responds to each challenge from the server with a message encrypted with the recovered private key $K_{priv}$. The client device then encrypts a series of hashes ($h_3$) generated from each question. Each $h_3$ results from hashing the correctly ordered hashes from the Client Table, the applied shift for the question, and the challenge nonce. The client device then sends this encrypted message to the server for verification.
The server receives the response, as shown in step 9. The server then verifies that the responses are correct by decrypting the message using the associated public key $K_{pub}$ that was stored at enrollment. The server recreates the expected hashes and compares them with the decrypted response. If correct, the server sends a response message that is first encrypted with server’s private key, then the user’s public key $K_{pub}$. The response may include a session encryption key for secure streaming communication, or the two can use the now verified public/private key for exchanges. Thus, a key that depends on the user’s biometric and password is recreated and exchanged such that the biometric and password never leave the device, providing increased security and no loss in privacy.

### 5.3.3 Index-based Vaulted Voice Verification - A walkthrough

In this section we will walk through the index-based Vaulted Voice Verification using a toy example. We will look at how the index-based process is able to securely perform remote verification without biometric data leaving the possession of the user. Having previously detailed this process, we will focus mainly on how the tables interact to perform the enrollment and verification tasks.

**Walkthrough Enrollment**

During the enrollment process, the user is presented with a series of questions, $Q$. For each question, one matching model and multiple non-matching models are generated. In
Figure 5.10: Match / Non-match: This table is never saved. It is merely a construct used for visualization purposes to show that for a given question, there will be several non-matching models and a single matching model.

<table>
<thead>
<tr>
<th>Model ID</th>
<th>Match / Non-match</th>
</tr>
</thead>
<tbody>
<tr>
<td>modelA</td>
<td>Non-match</td>
</tr>
<tr>
<td>modelB</td>
<td>Non-match</td>
</tr>
<tr>
<td>modelC</td>
<td>Match</td>
</tr>
<tr>
<td>modelD</td>
<td>Non-match</td>
</tr>
</tbody>
</table>

the single question toy example given in this section, we use three non-matching models. In reality, however, the number of matching models is only limited by the discriminability of the underlying models. In Fig. 5.10, we have a match / non-match table, which is used only for visualization purposes to show matching and non-matching. In the left column of this table, we see the models themselves (location to the model file). In the right column, we see if they are the correct matching model for this question. In this example, we see the modelC is the correct matching model for the question.

Once the models for the question are generated and we are aware of which one is correct, the models are each hashed. Each of the model IDs and the hashes are then entered into the table of models, as seen in Fig. 5.11. The hashes of the models are entered in the left column and the model IDs are entered on the right. Note that in this table, there is no indication as to which one is the correct matching model for the question; the table
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Hm</td>
<td>Model ID</td>
</tr>
<tr>
<td>4r5t6y</td>
<td>modelA</td>
</tr>
<tr>
<td>6y7u8i</td>
<td>modelB</td>
</tr>
<tr>
<td>7u8i9o</td>
<td>modelC</td>
</tr>
<tr>
<td>3e4r5t</td>
<td>modelD</td>
</tr>
</tbody>
</table>

Figure 5.11: Model Table: In the right column is the model itself (location to the actual model), and in the left column is a hash of the model.

...only contains a list of the models and their associated hashes. The row corresponding to the matching model is highlighted for illustration purposes.

Different variations exist for the construction of client and server tables, but for this toy example, we focus on embedding the private key. At this stage of enrollment, the server table is considered temporary because it is sent to the server after creation and not saved in its current form. Before the next set of tables are generated, the user creates their public/private key pair, $KU_{pub}$ and $KU_{priv}$. The public key is sent to the server along with the server table. The private key is used in the creation of the client and server tables and is then deleted. The client and temporary server tables require nonce $N_1$ and $N_2$, respectively. For the matching model, these nonces are generated from $KU_{pri}$, but they are randomly generated for others. First, $N_1$ is randomly generated to match the length of $KU_{pri}$. Then $N_2$ is generated by taking the $\oplus$ of $N_1$ and $KU_{pri}$. 
Hash $h_2$ | $(Hm, i, N_1)$
---|---
1q2w | $(4r5t6y, 1, 9935)$
qaws | $(7u8i9o, 3, 2354)$
azsx | $(6y7u8i, 2, 5539)$
fvgb | $(3e4r5t, 4, 2283)$

<table>
<thead>
<tr>
<th>Q</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>$(h_2, N_2)$</td>
<td>$(h_2, N_2)$</td>
<td>$(h_2, N_2)$</td>
<td>$(h_2, N_2)$</td>
</tr>
<tr>
<td>1</td>
<td>(fvgb, 2446)</td>
<td>(1q2w, 8678)</td>
<td>(qaws, 2479)</td>
<td>(azsx, 3454)</td>
</tr>
</tbody>
</table>

User: Bob

Figure 5.12: Client and Server Tables. These tables are co-dependent and generated together. The client table (top) contains a triple consisting of a model hash, an index into the server table, and a nonce, $N_1$, which is used to generate the private key. The triple is hashed, $h_2$, and becomes part of the server table entries. The server table (bottom) contains a question index, the user’s ID, and four doubles. The doubles consist of $h_2$ from the client table and a nonce, $N_2$, which is used to generate the private key. The data based on the matching model is highlighted for illustration purposes.

$$N_2 = N_1 \oplus KU_{pri} \quad (5.14)$$

The client and temporary server tables are commingled and are therefore constructed simultaneously. The client table contains two columns, one consisting of a triple and the other the hash of the triple, $h_2$. The triple consists of a hash of a model, an index ($i$) into the temporary server table, and a nonce ($N_1$). The triple is constructed so that only the
$i$ of the correctly matching model will point to the correct index in the temporary server table; others will randomly point to other non-matching locations. The temporary server table consists of a question index, multiple doubles (four in this case), and the user ID. Each of the doubles consists of a hash, $h_2$, and a piece of information used to generate the private key. Each of the doubles resides in a slot that corresponds to the index, $i$, from the client table. As shown in Fig. 5.12, the correct matching model data has $i = 3$ and the corresponding hash, $h_2$, resides in the third index position of the server table. The non-matching model data each point to an incorrect position. From here the temporary server table is sent to the server along with $K_U^{pub}$. Both $K_U^{pub}$ and the temporary server table are encrypted with the server’s public key, $K_S^{pub}$, before transmission over a secure channel (SSL, TLS, etc). The user then deletes its copy of the temporary server table.

Upon receipt of the temporary server table and $K_U^{pub}$, the server decrypts and begins to process the data. The server creates a new entry into its table of users and adds the username and $K_U^{pub}$ to the table. The server then adds a random nonce to the table and generates a hash of the entire table, including the newly generated nonce. The newly added nonce is never stored or shared with the user, ensuring that the user is not able to later modify the contents of the server table without detection. The user table is shown in Fig. 5.13. The server then encrypts the table using $K_S^{pub}$, so only it can decrypt the table, sends the table to the user, and deletes its local copy. Once the user receives the encrypted data from the server, the enrollment process is complete.
Walkthrough Verification

When the user needs to verify with the server, the first thing the user must do is send the server table, previously encrypted with $K_S^{\text{pub}}$, to the server. The server decrypts it, generates a hash from it, and verifies that the hash matches the data hash saved for the user in question. Once verification is complete, the server can generate a challenge.

When generating a challenge, the first thing the server will do is select a number of questions in a random order from the server table. In this toy example, only one question exists, so it is selected. The server then generates a random number string, the challenge string; the string’s length corresponds to the number of questions selected. Based on the

<table>
<thead>
<tr>
<th>ID</th>
<th>Data Hash</th>
<th>Public Key</th>
<th>Challenge Nonce</th>
<th>Challenge String</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>u7yt5r4</td>
<td>3095</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.13: Table of Users. Each row of this table contains a user ID, a hash of the user’s server table, the user’s public key, a challenge nonce, $N_3$, and a challenge string. A new $N_3$ is generated for each challenge. The challenge string stores the shift data for each challenge.

At the end of enrollment, the user has a model table, a client table, an encrypted server table and a list of questions. Of the rows in the model table, nothing exists indicating which model matches for the question. Of the client table, nothing exists detailing which rows combine for a question, or which of the entries will properly generate the $K_{U_{\text{pri}}}$. 
values of the challenge string, the server applies a shift to the rows of the server table. The server then generates a challenge nonce, $N_3$, and hashes the first value in each of the doubles ($h_2$ from the user’s client table). The shift is applied and the hashed values can be seen in Fig. 5.14. The server then encrypts and sends the challenge back to the user.

The user receives the challenge table and uses the client and model tables to turn the information received into a question that must be answered. As shown in Fig. 5.15, using $N_3$ and the client tables, a lookup table is generated and compared against each of the double values in the challenge table. Using the lookup table, each corresponding client table entry is located. Once the client table entries are located, the model table is used to
Figure 5.15: User Unscrambling Challenge (described from top to bottom). Note: the correct answers are highlighted for illustration purposes. Hashing the challenge nonce, $N_3$, against the hashes, $h_2$, in the client table locates each corresponding client table entry. The client table entries, in turn, reveal the associated model hashes. Using the model table hashes, the models are located.

locate the individual models. With this information, along with the text of the question from the list of questions, the user can respond to the question.

A model is generated from the user’s response. This new model is then scored against the question models. The model receiving the top score is selected as the matching answer. The user selects the corresponding client table entry based on the information associated with the matched model. Using the nonce associated with the entry, $N_1$, and the nonce from the server challenge, $N_2$, the user attempts to generate the correct $KU_{pri}$
Using the corresponding entry in the client table, the user calculates the shift applied by the server. The shift applied is calculated by comparing the index value from the client table with the location of the entry in the challenge table. In Fig. 5.15, we see that the client table entry’s index says 3, and the corresponding value was found in slot 1, indicating that a shift of two was applied.

Once a key is generated, the user constructs a response to the server. There exist a number of different response possibilities. Here, we give a response similar to that of Section 5.3.2 and [Johnson and Boult, 2013]. The response is a message consisting of a hash, $h_3$, consisting of the correctly ordered entries, the shift applied, and the challenge nonce, encrypted by $KU_{pri}$.

$$h_3 = h(\text{fvgb, 1q2w, qaws, azsx, 2, zzz})$$

(5.16)

$$M = KU_{pri}(h_3)$$

(5.17)

### 5.3.4 Communication Overhead

The term communication overhead encompasses many different things. In this work, we define it as "the amount of data transmitted between the client device and the server during
an enrollment and a verification of a single challenge-response pair.” We examine the
difference in communication overhead between Vaulted Voice Verification as originally
proposed and our new index-based version of the protocol.

During enrollment in the original protocol, four models are generated and sent to the
server for a total of 820K of data transmission. During verification, the four models are
sent twice; once from the client device to the server, so the server can scramble their
order and back to the client device from the server in the form of a challenge. This
process creates another 1,640K of data transmission per question. So, for enrollment
and verification of one challenge-response pair in Vaulted Voice Verification as originally
proposed, it takes roughly 2,460K of data transmission.

During enrollment of our new index-based protocol, a temporary server table con-
taining eight hashes is sent to the server from the client device and an encrypted server
table containing eight hashes and a nonce is returned. That is 17 values of 64 bytes each,
totaling 1,088 bytes or roughly 1K. During verification, an encrypted server table is first
sent to the server from the client device (eight hashes and a nonce), then a challenge table
is sent back (eight hashes and a nonce), and an encrypted response comprising of four
hashes, a single number and a nonce is finally sent to the server. That is 20 hashes, 3
nonces and a single number, totaling 1,476 bytes or roughly 1.5K. So, for enrollment
and verification of one challenge-response pair for our new index-based protocol, it takes
roughly 2,564 bytes or 2.5K.
Our new index-based protocol transmits 0.1% of the data that the original Vaulted Voice Verification does, i.e. it reduces communication overhead by 1000x. From this, it can be seen that the communication overhead in Vaulted Voice Verification as originally proposed is orders of magnitude larger than that of our new index-based protocol.

5.3.5 Storage Requirements

Similar to communication overhead, the storage requirements of both protocols must be defined. In this work, we define the storage requirements by the size of the data being stored. We only pay attention to final storage and not temporary or intermediate storage requirements.

When enrollment is complete in the original protocol, the client device contains four models and the server contains a hash of the template. This generates 820K of required storage on the client device and 64 bytes on the server. During verification, no extra data is stored. The total rounded storage requirement for the originally proposed Vaulted Voice Verification protocol is 821K.

When enrollment is complete in our new index-based version of the protocol, there exists a model table, a client table and an encrypted server table on the client device and a user table on the server. The model table contains 4 models and 4 hashes, 820K + 256 bytes, for a rounded total of 821K. The client table contains 4 hashes and 4 triples, 256 bytes + 4*(64+4+64), for a rounded total of 784 bytes; 0.7K. The encrypted server table
contains 8 hashes and a nonce, 512 + 64, for a rounded total of 576 bytes; 0.5K This gives an enrollment storage for the client device with our index-based version of 822K. The server contains a table for the user. This user table contains a hash of the data, 64 bytes, a public key, 64 bytes, a challenge nonce, 64 bytes, and a challenge string, 1 number, for a rounded total of 196 bytes. This gives an overall enrollment storage for our index-based version of 823K. No extra data is generated for storage at the completion of verification for our index-based version of the protocol. Thus, the number remains 823K.

As shown here, the difference in necessary storage requirements is negligible between the two versions of the protocol. While our new index-based version does contain additional tables compared to the original version, the size of the tables is negligible.

### 5.3.6 Key Management and challenge strength

The last area of focus in our experiments is key management and challenge strength. For this, we look at the trade-off between system usability and the ability to generate/manage keys and challenges of a given size for both protocols. That is, we examine how many user interactions are required to generate a key, release a key, and how much security does that key or challenge actually provide. We define a user interaction as the user being asked and responding to a single question. We consider each question and its associated responses to be independent. We consider generating keys suitable for RSA. Because we use SHA-512 in combination with RSA to generate the keys [Barker et al., 2012],
512 is the minimum number of bits required and the minimum size RSA keys generated is 1,024. For this discussion, we define usability as 20 or fewer interactions during enrollment/verification.

With Vaulted Voice Verification as originally proposed, every user interaction produced two bits, considering four possible choices. With this, to generate 512 bits would require 256 user interactions. This is clearly not acceptable. Even if the number of choices per question increased to 8, the resulting 64 interactions required renders the system nearly useless. While increasing the amount of choices per question seems a plausible solution, it is bounded by the discriminability of the models and therefore cannot be relied upon as a valid solution.

With our new index-based Vaulted Voice Verification, each interaction generates far more bits than in the original Vaulted Voice Verification protocol, using the added fields in the index tables. This occurs because responses are created from hashes and indexes rather than binary (or 4-choice) decisions. Our index-based protocol generates an RSA private key for the final response with a wide range of usability/security choices. One could, in theory, have a 512 bit key even with just a single challenge question. Remember, the key is already protected by the users password; our challenge is about added security beyond the standard password protection of the private key. In addition, the challenge is about how difficult is it for someone with knowledge of the password and the private key to still impersonate the user or the owner to “share” the key. Having multiple questions
allows us to split the private key into sets of scrambled components $K_{\text{priv}}$, which must be properly ordered and combined with the appropriate data from the challenge $R_i$. $R_i$ is not stored locally, so, even when it comes in from the server, the association is unknown. Recovering the key and returning the correct response, given all passwords and the server’s encryption keys, is still $O(2^{(2q)})$, where $q$ is the number of challenge questions. If we assume there are also $k$ bits of user password security, and $s$ bit of server key security, then the new index system protection is $k + s + 2q$ overall, of which $2q$ bits of “biometric identity” security.

As shown, our new index-based version of the Vaulted Voice Verification protocol is able to manage keys suitable for biometrically authenticated secure communication and public-key operations. This results from decoupling the models from the stored question and replacing that link with a novel, table-based index scheme.

Our index-based version of the Vaulted Voice Verification protocol supports keys of sufficient size for use in RSA keys while allowing variable levels of biometric identity security in the challenge. This new form allows standard PKI/public-key operations to be tied to biometric verification. Importantly, the novel design also overcomes the problem in the original Vaulted Voice Verification protocol wherein the server challenge was the only “key;” hence someone with sufficient access to the server might impersonate the user. In our new design, the use of private-key signed return values means that, even with full access to the server and every communication, one cannot impersonate the user. Thus,
when a key is shared by this approach, both parties have strong assurance the correct user is in possession of the remote device and is not being impersonated.

## 5.4 Security Analysis

In this section we will analyze the security of both index-based and non-index-based Vaulted Voice Verification protocol. We continue the definitions as given in Table 5.1. As introduced in [Johnson et al., 2013a], the security analysis of the Vaulted Voice Verification protocol can only be properly given by examining the protocol in different scenarios. The reason for breaking the analysis down into different scenarios is because the security of the protocol is dependent on different factors at different times. Thus, the security discussion put forth in this section is in terms of different attack scenarios. In [Johnson et al., 2013a], the security analysis looked at the different bits of security generated by the protocol in terms of knowledge, biometrics, server encryption, and password-based encryption. Here, we formalize the analysis for both index and non-index Vaulted Voice Verification.

### 5.4.1 Security in “Bits”

The security analysis for our work on Vaulted Voice Verification is straightforward, simple and similar to that of the original protocol. The security analysis for the original protocol
is given in [Johnson et al., 2013b], but it lacks analysis based on specifically defined attack models. For our research, we looked at the security of our extended version of Vaulted Voice Verification in terms of six different attack models listed in order of likelihood of attack.

1. The knowledgable impersonator “borrows” the device: An attacker, most likely a “friend” in this case, grabs your device, knows your password and your answers.

2. Compromised transmissions: An attacker is able to capture and isolate all the data in transmission, but has not obtained any of the encryption keys.

3. Stolen device and password: An attacker is able to obtain the keys of the user and access the data, but does not have access to the server.

4. Compromised server: An attacker is able to obtain the keys for the server but cannot manipulate the server’s software/network.

5. “Insider attack”: An attacker is able to obtain the keys of the server and is able to access data stored there.

6. The “mission impossible” attacker: An attacker spends time to make recordings of your voice and steals your device.

It is necessary to note that this analysis of security in terms of bits is speaking in terms of how much security is added on top of encryption. Our implementation of Vaulted
Voice Verification provides $P$-bits of security for the encryption from the salted user password with the assumption that the device will lock up after a set number of attempts, $S$-bits of security from the server encryption of the template, $K$-bits of “knowledge-based” security, and $B$-bits of “biometric-identity” security. When we refer to $K$-bits of knowledge-based security, we mean to say security that is gained per challenge-response question due to something that the authentic user knows and an attacker does not. Thus, depending on the attack model, the odds for an attacker guessing correctly mirror that of random chance. When we speak of $B$-bits of biometric-identity security, we mean security that is gained through the use of voice-based models that take advantage of the difference in the voices and speech patterns of different speakers.

For attack model 1, because the attacker has access to the device and the password, they can bypass the encryption. For the multiple choice questions, the attacker knows the answers. Because the attacker does not have the correct voice, this reduces to the $B$-bits of biometric-identity security. For a set of $N$ questions, the remaining security would then rely on only the scrambling of the models, which can only be correctly ordered using the original biometric.

For attack model 2, the attacker has obtained the data. Another way to look at this attack model is to imagine that an attacker is able to watch all transmission of data between the server and the client, but is not able to decrypt said transmissions. During both enrollment and verification, only encrypted data is being transmitted back and forth
between the client and the server. Without either key, the attacker would not be able to access the data. Because the data blocks are scrambled before encryption, every time data is sent, the ciphertext is different. Thus an attacker is not able to gain anything from the information they are able to collect. With this attack model, the bits of security are a total of $P + S + K + B$.

For attack model 3, similar to attack model 1, an attacker is able to compromise the keys and the data of the user. The attacker would not know the answer to the multiple choice question, reducing their chances to random guessing (again, making the assumption that the question choices are independent). The attacker would also not have the correct voice pattern for the text-dependent or text-independent matching problem. The password and user keys are compromised, but the remaining bits of security are $B + K$.

For attack model 4, the attacker has somehow obtained the encryption keys for the server and can scan the server’s disks but cannot modify the operational software. Because the server has no stored data, there is not much they can do with the data found on the server. They could set up a phishing application and launch a man-in-the-middle attack, but with this they can only decrypt the template. In this attack model, there would still exist $P + K + B$ bits of security.

For the next attack model, number 5, the insider can ignore all protocols on the server, making verification inconsequential. However, to impersonate a user on any other server, there would still exist $P + K + B$ bits of security to overcome. Because the model files
are encrypted, the attacker gains no information about the raw biometric and therefore cannot identify or impersonate the user anywhere else.

Attack 6 is a classic movie plot. A dedicated attacker is able to obtain multiple voice samples of the user. An outside attacker, without any of the keys, will have gained nothing from doing this. The user would still be protected by $P + S$ bits of security. If one were to believe this movie plot threat has a higher probability of occurring for this individual, they could specify more text-independent questions so that no replay attack can be used. However, this attack seems to occur only in the movies. The most likely scenario for this attack is to have malicious code on the phone, which is why we include at least one text-independent question.

Reviewing these attacks, it becomes clear that the most likely and most invasive attack is a snooping “friend”. This “friend” would posses enough access and knowledge to bypass multiple layers of the security provided, but would still face the challenge of matching the correct biometric. Given the breakdown of the layers of security provided by Vaulted Voice Verification into “bits”, the remainder of our analysis will focus on analyzing the different bits of security.
5.4.2 Overall Security

Biometrics

As described in previous sections, the biometric matching that occurs in the Vaulted Voice Verification protocol is an integral part of the security that is provided by the system. Here we breakdown the security of the biometric matching in terms of the challenges.

The challenge, $C$, consists of a number of questions, $Q$, and each $Q$ contains a number of possible answers, $A$. Given a biometric question, the user’s biometric must match the correct model in order to successfully respond to the question. We assume a well formed question such that, for an attacker, there exists an equal probability of matching the correct model versus matching an incorrect model. In other words, given a single question in $Q$, if there exists $a$ possible answers and all possibilities are equally likely, an attacker has a $\frac{1}{a}$ chance of answering $Q$ based on their matching biometric. However, this assumes that the biometric data is perfectly distributed and there is no error tolerance (the equal error rate of the system is 0). Since we can not assume a perfect world, we assume that an attacker posses a slightly better than random chance at matching the correct model. Thus, we give the attacker an $\epsilon$ advantage. For the purposes of the formulas presented, we consider the random chance of an attacker guessing correctly as $R$. The value $R$ is bound on random probability. If we over estimate $R$, it becomes an upper bound. If we underestimate $R$, it becomes a lower bound.
Based on the equal error rate of the system, we can not assume that the authentic user will always match perfectly. What we can assume is that the authentic user will have a much greater chance of matching the correct model over an attacker. For this, we give the authentic user a $\beta$ advantage over random, or an $\alpha$ disadvantage from the ideal of 1. For simplicity sake, we also assume that all $\epsilon$s and $\alpha$s are independent and identical.

Thus, for a single question, the probability, $P$, of an attacker or an authentic user matching the question based on biometrics would be:

\[ P_{\text{attacker}} = R + \epsilon \]  \hspace{1cm} (5.18)

\[ P_{\text{authentic}} = 1 - \alpha \]  \hspace{1cm} (5.19)

where $\epsilon$ and $\alpha$ are small and

\[ \beta = (1 - \alpha) - R \]  \hspace{1cm} (5.20)

such that $\beta \geq \epsilon$.

We’ve calculated the probability of acceptance for an attacker in Equation 5.18. The probabilities for the rejection is the difference between 1 and the probabilities of acceptance $(1 - P)$. That is to say, the probabilities of acceptance and rejection combined are equal to 1. If we give the attacker a slight advantage over random chance for getting a
question correct, we must then give a slight disadvantage to the remaining choices so the equation balances. Thus, the probability for an attacker for a single question is expressed as:

\[
P_{\text{attacker accept}} = (R + \epsilon) \quad \text{(5.21)}
\]

\[
P_{\text{attacker reject}} = \left( R - \frac{\epsilon}{a - 1} \right) \quad \text{(5.22)}
\]

and the probability for an authentic user is expressed as:

\[
P_{\text{authentic accept}} = (1 - \alpha) \quad \text{(5.23)}
\]

\[
P_{\text{authentic reject}} = \left( R - \frac{\beta}{a - 1} \right) \quad \text{(5.24)}
\]

For an individual question, the probabilities are quite simple. For multiple questions, however, the probabilities for the Vaulted Voice Verification protocol is a bit more complex. In looking for the probabilities, we must first break down what it is we really seek. To do that we must look at the problem slightly differently. The goal of the protocol is to verify an identity. To do so, the protocol selects a subset of \( m \) questions from the total enrollment of \( n \) questions during verification (which very well could be the entire enrollment or one question). From that subset, the user attempting to verify must get an
unordered subset, $k$, correct in order to pass verification (have a successful result, generate the correct key, etc). Thus, for a series of questions, Vaulted Voice Verification follows the binomial distribution, as explained in [Rice, 2007]:

Suppose that $n$ independent experiments, or trials, are performed, where $n$ is a fixed number, and that each experiment results in a “success” with probability $p$ and a “failure” with probability $1 - p$. The total number of successes, $X$, is a binomial random variable with parameters $n$ and $p$. For example, a coin is tossed 10 times and the total number of heads is counted (“head” is identified with “success”).

The probability that $X = k$, or $p(k)$, can be found in the following way:

Any particular sequence of $k$ successes occurs with probability $p^k(1 - p)^{n-k}$, from the multiplication principle. The total number of such sequences is $\binom{n}{k}$, since there are $\binom{n}{k}$ ways to assign $k$ successes to $n$ trials. $P(X = k)$ is thus the probability of any particular sequence times the number of such sequences:

$$p(k) = \binom{n}{k} p^k (1 - p)^{n-k} \quad (5.25)$$

To put that in terms of the Vaulted Voice Verification protocol: Given $n$ total questions, each with a probability $p$ of being correctly answered or matched and a probability
of incorrectly matching or answered incorrectly, the probability of any sequence of \( k \) successes given the total number of ways to assign \( k \) successes to \( n \) questions is Equation 5.25 where \( k \) is the number of correctly answered questions (either biometrically matched or correctly chosen answers). Furthermore, in the Vaulted Voice Verification protocol, a subset of \( n \) questions is selected from the available questions. In the Vaulted Voice Verification protocol, the number of questions needed to be guessed correctly is a minimum number, \( k \), such that \( k \leq n \). With this, the probability must be expressed in terms of the probability, \( p \), of getting \( k \) to \( n \) questions correct. Thus, the equation becomes:

\[
P(k \leq X \leq n) = \sum_{k}^{n} \binom{n}{k} p^k (1 - p)^{n-k}
\]

Given \( P \) for the random case, we must then give an \( \epsilon \) advantage to an attacker and a \( \beta \) advantage to the authentic user. For an attacker, the probability of verifying, which is considered a false accept, is:

\[
P_{attacker}(k) = \binom{n}{k} (R + \epsilon)^k (1 - (R + \epsilon))^{n-k}
\]

For an authentic user, the probabilities for successfully verifying is:

\[
P_{authentic}(k) = \binom{n}{k} (1 - \alpha)^k (1 - (1 - \alpha))^{n-k}
\]

where \( X \) is the actual number of successes, \( k \) is the minimum number of successes to
verify, \( n \) is the total number of questions, \( m \) is the number of questions selected, \( a \) is the number of possible correct answers per question, \( \epsilon \) is the attacker’s advantage, and \( \alpha \) is the authentic user’s disadvantage (\( 1 - \alpha = \beta \)).

Now that we have the definition for \( P(X) \), we can use it to calculate the security of Vaulted Voice Verification for an attacker and for an authentic user. However, because \( P(X) \) also depends on \( \epsilon \) and \( \beta \) advantages, we must define these terms in a meaningful way before we can calculate the security.

With Vaulted Voice Verification we consider that the \( \epsilon \) advantage given to the attacker is equivalent to the increase above random chance that the attacker has at correctly matching the biometric or guessing the answer correctly. This directly corresponds to the False Match Rates (FMR) as given in Section 6.4.2. Based on the FMR of the system, as experimentally determined in Figure 6.4, we bound \( \epsilon \) to be between 0.00 and 0.05.

Conversely, the \( \beta \) advantage given to the authentic user represents the margin of error that exists for an authentic user to match with their own biometric model or get the answer correct. This directly corresponds to the False Non-Match Rates (FNR) as given in Section 6.4.2. Based on the FNR of the system, as experimentally determined in Figure 6.4 we bound \( \beta \) to be between 0.00 and 0.12.

With a bounded \( \epsilon \) and \( \beta \), we can now consider the security of the Vaulted Voice Verification protocol. To calculate the security for an attacker and an authentic user respectively, we must examine things from two separate perspectives. The first is the probability of
success and the related estimated number of bits of uncertainty. The second is the guessing entropy, which considers the number of guesses required to produce a correct answer based on the probabilities per possible answer. The reason for the two separate models is that, while looking the binomial examines the bits of security in terms of uncertainty, it does not account for the user’s ability to guess and how the possible need for subsequent guesses per question relates to the overall security of the Vaulted Voice Verification protocol.

The probability of success, as calculated above, follows the binomial distribution. Given this and the approximation for the $\epsilon$ and $\beta$ values, we can compute the bits of uncertainty for the distribution.

Because the probability is simply the likelihood of some outcome, or saying there exists roughly a 1 in $n$ chance of success, it directly relates to the number of guesses required to achieve a given outcome. If the response to those guesses are either correct or incorrect, the $n$ can then be represented as $2^n$, or how many guesses are necessary, where $n$ is now the number of bits required to represent the guesses. Thus, the relation of probability, $P(x)$, to bits is:
\[ P(x) = \frac{1}{2^n} \]
\[ = 2^{-n} \]
\[ \log_2(P(x)) = \log_2(2^{-n}) \]
\[ = -n \log_2(2) \]
\[ = -n \]
\[ - \log_2(P(x)) = n \]

where \( n \), in this case, is the number of bits needed to represent the probability.

So, with this information and a series of probabilities, we can begin to look at the range of bits of uncertainty. To make better sense of things, we consider an example. In this example we have a 10 question challenge in which the user must get at least 6 questions correct. We will use 0.025 as the \( \epsilon \) advantage (the mid-range of the FMR) and 0.06 for the \( \alpha \) disadvantage (the mid-range of the FNR). We consider 4 possible models to match per question.

\[
P_{\text{attack}}(x) = \sum_{i=6}^{10} \left( \binom{10}{i} \left( \frac{1}{4} + 0.025 \right)^i \left( 1 - \left( \frac{1}{4} + 0.025 \right) \right)^{10-i} \right) \]
\[
= \sum_{i=6}^{10} \left( \binom{10}{i} (0.275)^i (0.725)^{10-i} \right) \]
\[
= 0.031374 \]
\[ P_{\text{attacker}}(x) = 0.031374 = 2^{-n} \]
\[ -\log_2(0.031374) = n \quad (5.31) \]
\[ 4.9942 = n \]
\[ P_{\text{authentic}}(x) = \sum_{i=6}^{10} \left( \binom{10}{i} (1 - 0.06)^i (1 - (1 - 0.06))^{10-i} \right) \]
\[ = \sum_{i=6}^{10} \left( \binom{10}{i} (0.94)^i (0.06)^{10-i} \right) \quad (5.32) \]
\[ = 0.999848 \]
\[ P_{\text{authentic}}(x) = 0.999848 = 2^{-n} \]
\[ -\log_2(0.999848) = n \quad (5.33) \]
\[ 0.000219 = n \]

Thus, for this example, the number of bits (rounded up to the nearest whole bit) required to represent the uncertainty of the attacker would be about 5 bits, whereas the number of bits to represent the uncertainty of the authentic user would be about 1 bit. In other words, the smaller probability of the attacker leads to a much greater uncertainty. To better understand what these results mean, we refer to Table 5.2 in which we examine a similar scenario with 20 total questions instead of 10.

According to works such as [Davis et al., 2004], basing their derivations on [Massey, 1994] and [Bonneau et al., 2010], the per question guessing entropy, \( G(x) \), is calculated
<table>
<thead>
<tr>
<th># Correct</th>
<th>$P_{attacker}$</th>
<th>$n_{attacker}$</th>
<th>$P_{authentic}$</th>
<th>$n_{authentic}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/20</td>
<td>0.998391</td>
<td>0.002323</td>
<td>1.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>2/20</td>
<td>0.986179</td>
<td>0.020079</td>
<td>1.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>3/20</td>
<td>0.942174</td>
<td>0.085935</td>
<td>1.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>4/20</td>
<td>0.842024</td>
<td>0.248066</td>
<td>1.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>5/20</td>
<td>0.680576</td>
<td>0.555172</td>
<td>1.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>6/20</td>
<td>0.484611</td>
<td>1.045100</td>
<td>1.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>7/20</td>
<td>0.298783</td>
<td>1.742830</td>
<td>1.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>8/20</td>
<td>0.157810</td>
<td>2.663743</td>
<td>1.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>9/20</td>
<td>0.070917</td>
<td>3.817733</td>
<td>1.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>10/20</td>
<td>0.026971</td>
<td>5.212464</td>
<td>1.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>11/20</td>
<td>0.008635</td>
<td>6.855649</td>
<td>1.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>12/20</td>
<td>0.002312</td>
<td>8.756730</td>
<td>0.999999</td>
<td>0.000001</td>
</tr>
<tr>
<td>13/20</td>
<td>0.000513</td>
<td>10.928362</td>
<td>0.999989</td>
<td>0.000016</td>
</tr>
<tr>
<td>14/20</td>
<td>0.000093</td>
<td>13.388094</td>
<td>0.999892</td>
<td>0.000156</td>
</tr>
<tr>
<td>15/20</td>
<td>0.000014</td>
<td>16.160749</td>
<td>0.999132</td>
<td>0.001254</td>
</tr>
<tr>
<td>16/20</td>
<td>0.000002</td>
<td>19.282360</td>
<td>0.994366</td>
<td>0.008151</td>
</tr>
<tr>
<td>17/20</td>
<td>0.000000</td>
<td>22.807539</td>
<td>0.971034</td>
<td>0.042406</td>
</tr>
<tr>
<td>18/20</td>
<td>0.000000</td>
<td>26.825441</td>
<td>0.885028</td>
<td>0.176206</td>
</tr>
<tr>
<td>19/20</td>
<td>0.000000</td>
<td>31.502346</td>
<td>0.660455</td>
<td>0.598469</td>
</tr>
<tr>
<td>20/20</td>
<td>0.000000</td>
<td>37.249928</td>
<td>0.290106</td>
<td>1.785347</td>
</tr>
</tbody>
</table>

Table 5.2: This table shows how the probability and number of bits of security change as the minimum number of questions required for positive verification changes. The first column, # Correct shows the minimum number of questions required for positive verification. The second and forth columns, $P_{attacker}$ and $P_{authentic}$ respectively, show the changing probabilities. The third and fifth columns, $n_{attacker}$ and $n_{authentic}$ respectively, show the uncertainty in terms of number of bits. The values used for $\epsilon$ and $\beta$ are 0.025 and 0.06 respectively.
Figure 5.16: This figure shows the security in terms of the number of bits needed to represent the probability. The Y-axis shows the number of bits of security and the X-axis shows the number of questions that must be correct (out of 20). The calculations in this figure are based on multiple choice questions with 4 possible answers each (width = 4). The different curves for the Binomial and “Guessing Binomial” show how changing the $\epsilon$ advantage effects the results. The “Guessing Binomial” is calculated by using $G(x)$ instead of $P(x)$ in the binomial distribution calculation. It is described in Equation 5.44.
as

\[ G(x) = \sum_{i=0}^{p} (i \times P_{\text{answer}}) \]  

(5.34)

where \( p \) denotes the total possible number of answers, \( P_{\text{answer}} \) denotes the probability of a single given answer (each given in decreasing order of probability), and \( i \) represents the guess number.

On a per question basis, the \( \epsilon \) is only available on the first guess. After that, each guess has an equal, and random, probability of being the correctly matched answer. With each guess, the available number of guesses from which to choose decreases, and therefore, the probability of success, for the next guess, increases. For example, a question with 4 possible answers, with the attacker having an \( \epsilon \) advantage, would generate a guessing entropy of:

\[ 1 \times \left( \frac{1}{4} + \epsilon \right) + 2 \times \left( \frac{1}{3} \right) + 3 \times \left( \frac{1}{2} \right) + 4 \times (1) \]  

(5.35)

However, the overall probability for the question remains constant no matter the number of guesses made. Thus, the guessing entropy for said question would be calculated as:

\[ 1 \times \left( \frac{1}{4} + \epsilon \right) + 2 \times \left( \frac{1}{4} - \frac{\epsilon}{n-1} \right) + 3 \times \left( \frac{1}{4} - \frac{\epsilon}{n-1} \right) + 4 \times \left( \frac{1}{4} - \frac{\epsilon}{n-1} \right) \]  

(5.36)

To calculate the entropy of a number of questions, we must look at things in terms
of how many questions from a given set must be answered correctly. Given the method of question selection used in Vaulted Voice Verification, the entropy calculation is related to the number of questions selected as a subset that need be correct from a group of questions. So, the entropy for a set of questions, $G(x)$ is simply:

$$G(x) = \sum_{i=1}^{k} G(x_i)$$

(5.37)

where $G(x)$ is the per question entropy and $k$ is the number of questions the user must correctly match. We observe that, in some cases, the underlying questions will each possess the same guessing entropy, $G(x)$, in which case the value is then simply multiplied by the number of questions. While this might be a common occurrence, it's not the only possibility, and therefore the equation must be generalized.

The entropy of the subset of answers that generate a positive verification, $k \leq x \leq n$, is the sum of the entropies from 1 to the minimum required:

$$G(k \leq x \leq n) = G(x) = \sum_{i=1}^{k} (G(x_i))$$

(5.38)

because the user is not required to get any more questions correct than the minimum. i.e. any number of correctly matched answers above the minimum will generate a positive verification.

To make better sense of things, we consider the same example as our previous one.
We have a 10 question challenge of which the user must get at least 6 questions correct. We will use 0.025 as the $\epsilon$ advantage (the mid-range of the FMR) and 0.06 for the $\alpha$ disadvantage (the mid-range of the FNR). We consider 4 possible models to match per question.

\[
G_{\text{attacker}}(x) = \sum_{i=0}^{4} (i \cdot P_{\text{answer}})
\]

\[
= 1 \cdot (1/4 + 0.025) + 2 \cdot (1/4 - 0.025/3) + 3 \cdot (1/4 - 0.025/3) + 4 \cdot (1/4 - 0.025/3)
\]

\[
= 0.275 + 0.483 + 0.725 + 0.966
\]

\[
= 2.449
\]

\[
G_{\text{attacker}}(6 \leq x \leq 10) = \sum_{i=1}^{6} G(x_i)
\]

\[
= \sum_{i=1}^{6} (2.449)
\]

\[
= 14.694
\]
\[ G_{\text{authentic}}(x) = \sum_{i=0}^{4} (i \cdot P_{\text{answer}}) \]

\[ = 1 \cdot (1 - 0.06) + 2 \cdot (0 + 0.06/3) \]

\[ + 3 \cdot (0 + 0.06/3) + 4 \cdot (0 + 0.06/3) \]

\[ = 0.94 + 0.04 + 0.06 + 0.08 \]

\[ = 1.120 \] (5.41)

\[ G_{\text{authentic}}(k \leq x \leq n) = \sum_{i=1}^{k} G(x_i) \]

\[ = \sum_{i=1}^{6} (1.120) \] (5.42)

\[ = 6.720 \]

Equations 5.40 and 5.42 tell us that, for an attacker and an authentic user to answer at minimum 6 of 10 questions correctly, it would take them (rounded up to the nearest whole guess) 15 and 7 guesses, respectively. As a result, for the given scenario, the verification system could reject any user who attempted to verify and took more than 7 guesses. To see how the guessing entropy changes as the number of available options per question changes, we refer to table 5.3.

A relationship exists between the guessing entropy, \( G(x) \), and the Shannon entropy, \( H(x) \). As described in [McEliece and Yu, 1995], the relationship is:
Table 5.3: This table shows how the guessing entropy, $G(x)$, for a single question changes as the number of choices changes for an attacker and for an authentic user. The value for $\epsilon$ and $\beta$ in this table are 0.025 and 0.06 respectively.

<table>
<thead>
<tr>
<th># Choices</th>
<th>$G_{\text{attacker}}(x)$</th>
<th>$G_{\text{authentic}}(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.025000</td>
<td>0.940000</td>
</tr>
<tr>
<td>2</td>
<td>1.475000</td>
<td>1.060000</td>
</tr>
<tr>
<td>3</td>
<td>1.962500</td>
<td>1.090000</td>
</tr>
<tr>
<td>4</td>
<td>2.450000</td>
<td>1.120000</td>
</tr>
<tr>
<td>5</td>
<td>2.937500</td>
<td>1.150000</td>
</tr>
<tr>
<td>6</td>
<td>3.425000</td>
<td>1.180000</td>
</tr>
<tr>
<td>7</td>
<td>3.912500</td>
<td>1.210000</td>
</tr>
<tr>
<td>8</td>
<td>4.400000</td>
<td>1.240000</td>
</tr>
<tr>
<td>9</td>
<td>4.887500</td>
<td>1.270000</td>
</tr>
<tr>
<td>10</td>
<td>5.375000</td>
<td>1.300000</td>
</tr>
<tr>
<td>11</td>
<td>5.862500</td>
<td>1.330000</td>
</tr>
<tr>
<td>12</td>
<td>6.350000</td>
<td>1.360000</td>
</tr>
<tr>
<td>13</td>
<td>6.837501</td>
<td>1.390000</td>
</tr>
<tr>
<td>14</td>
<td>7.325000</td>
<td>1.420000</td>
</tr>
<tr>
<td>15</td>
<td>7.812500</td>
<td>1.450000</td>
</tr>
<tr>
<td>16</td>
<td>8.300001</td>
<td>1.480000</td>
</tr>
</tbody>
</table>
Figure 5.17: This figure shows the guessing entropy, $G(x)$ for a single question as the number of choices changes for an attacker. The different curves show the effects of the $\epsilon$ advantage.
where \(|\mathcal{X}|\) represents the cardinality of the distribution (for Vaulted Voice Verification this is the number of possible answer for the given question). So we use this conversion to make the comparison between the guessing entropy, \(G(x)\), and the Shannon entropy, \(H(x)\). By converting \(G(x)\) to \(H(x)\), we are able to make an apples-to-apples comparison between the two for a single question. Given the equation 5.43, we know that the guessing entropy should represent a lower bound of the Shannon entropy, and the resultant security in bits should also represent a lower bound. The simple calculation is to simply plug in \(G(x)\) into equation 5.43 to give the lower-bounded Shannon entropy, \(H(x)\), for a single question and then to scale the values for multiple questions. However, because the Vaulted Voice Verification protocol follows the binomial distribution, as previously described, we know that we can not simply scale the lower-bound of the entropy for a single question to compute the entropy of the minimum number of correct matching answers required for a successful verification.

To solve this problem, we examine the meaning of the guessing entropy, \(G(x)\), as compared to the single question probability, \(P(x)\). Because we focus on the security for this analysis, we explore the relationship in terms of security against an attacker only (omitting the simple calculations for an authentic user). In equation 5.18, we discussed that for a single question, the attacker is given some \(\epsilon\) advantage over the random chance,
of successfully answering the question. Another way to look at this is to say that the number of options that the attacker must pick from is slightly smaller than the total available number of options available. A simpler example would be to say that if there exist 10 choices, then the $\epsilon$ advantage says that the attacker knows that one of them is for sure not the correct answer, so their chances of random guessing improves from $\frac{1}{10}$ to $\frac{1}{9}$.

What the guessing entropy, $G(x)$, really says is the number of guesses the attacker must make on average in order to successfully verify. The calculations for the guessing entropy, shown in Equation 5.34, takes into account the probability of success of each guess, as influenced by the $\epsilon$ advantage. Therefore, what the guessing entropy gives us is a lower bound on the number of guesses required to successfully answer a question given the probabilities of each guess: in other words, an upper bound on the random chance of getting a question correct. So, to compute the number of bits of security given by the guessing entropy over the distribution (getting $K$ questions correct from a set of $N$), we simply plug in the guessing entropy into the binomial distribution formula, replacing $R$ and $\epsilon$ with $\frac{1}{G(x)}$.

$$PG_{attacker}(k) = \binom{n}{k} \left( \frac{1}{G(x)} \right)^k \left( 1 - \left( \frac{1}{G(x)} \right) \right)^{n-k}$$ (5.44)

Equation 5.29 can then be applied to generate the security, in terms of bits. Figure 5.16 shows that the guessing entropy does, in fact, represent a lower bound for the number of bits of security provided by the Vaulted Voice Verification protocol.
After working through this analysis, a few observations can be made. The binomial distribution used to analyze the the Vaulted Voice Verification protocol is only accurate given the assumption that each question contains the same probability. Removing that assumption means the protocol will follow a slightly different distribution, the Poisson binomial distribution [Le Cam et al., 1960]. The Poisson binomial distribution follows the same logic of the binomial distribution, but it makes no assumptions about the probabilities from question to question. Also, the same observation can be made when converting the guessing entropy into bits of uncertainty. If the underlying guessing entropy changes from question to question, then the overall lower-bounded entropy that results should be generated from the Poisson binomial distribution [Le Cam et al., 1960] [Daskalakis et al., 2012].

This analysis shows that the biometric portion of the overall security of the Vaulted Voice Verification protocol is secure. By separating and analyzing only the biometric matching, we show that the Vaulted Voice Verification protocol is able to generate over 35 bits, lower bounded around 30 bits (the lower bound being a better estimator) of security for 20 questions (4 choices wide each question). We also show that by increasing the number of choices per question, we can increase the security of the biometric matching. For example, by increasing the number of choices per question to 8, the lower bound of the security in bits for the Vaulted Voice Verification protocol increases to over 45 bits for the same number of questions. In later sections, we show how this can be accomplished
even without the need to increase the number of models required for matching. When
we compare this to other results for bits of security generated by other voice biometric
protocols, such as the 46 bits as generated by [Monrose et al., 2002b], the 60 bits as
generated by [Monrose et al., 2002a] (but shown to be an unstable maximum), and the
51 bits as generated by [Carrara and Adams, 2010], we see that the generated biometric
security alone is comparable to the current state of the art. As previously mentioned
with the Vaulted Voice Verification protocol, the security gains from the biometrics are
only one aspect of the overall security increase above standard encryption and passwords
alone.

The security demonstrated in this section strictly follows the security generated by the
biometric portion of the protocol. In the remaining parts of this section, we examine the
remaining “bits” that contribute to the overall security of the Vaulted Voice Verification
protocol.

**Knowledge**

The security provided by knowledge is similar to the security provided by the biometric
in that it is based on selection by the user, but it differs in that there are multiple ways
knowledge is utilized in the protocol based on the types of questions presented to the user.
If the user is presented with a multiple choice question containing \( n \) possible answers and
all possibilities are equally likely, then the possibility of an attacker guessing the answer
correctly is $\frac{1}{n}$, as with the biometric. If, however, the user is presented with an open-ended question, the attacker’s chances are reduced based on the possible vocabulary size. For such questions, the value of $R$, the random chance of a correct answer/response, differs based on the frequency of usage for non-equally likely outcomes. For example, the space for 8 character passwords is quite large, but the actually used space is considerably non-uniform. Therefore, the actual $R$ would be much larger than it would be for a truly randomly selected 8 character sequence.

Similar to the biometric case, for some number of questions, the probability of an attacker having knowledge enough to answer the questions correctly is:

$$P_{\text{attacker}} = R + \epsilon$$

(5.45)

where $a$ varies based on the number of possible answers (ex. the number of options in multiple choice or the total number of words in the dictionary), and the probability for an authentic user is:

$$P_{\text{authentic}} = 1 - \alpha$$

(5.46)

such that $(1 - \alpha = \beta)$.

Following the same logic and derivation as with the biometrics, the security in bits offered by the protocol for an attacker and an authentic user, respectively, is calculated by plugging in the $\epsilon$ and $\beta$ advantages into the probability formula as with Equations 5.27.
and 5.28:

\[ P_{\text{attacker}}(x) = \sum_{i=k}^{n} \binom{n}{i} (R + \epsilon)^i (1 - (R + \epsilon))^{n-i} \]  \hspace{1cm} (5.47) 

\[ P_{\text{authentic}}(x) = \sum_{i=k}^{n} \binom{n}{i} (1 - \alpha)^i (1 - (1 - \alpha))^{n-i} \]  \hspace{1cm} (5.48)

and then taking the values generated from these formulas and then applying Equation 5.29, where the \( \epsilon \) advantage now refers to an attacker’s ability to make a guess that is slightly better than random chance, and the \( \alpha \) disadvantage refers to an authentic user’s lack of perfect recall of the answers. For the purposes of this research, we assume that \( \epsilon = 0 \) and \( \alpha = 0 \). However, we include the values in this formula to show that it is possible to model the guesses of an attacker and the recall of the authentic user. As with the biometric security bits, the knowledge-based security is calculated based on getting some minimum or more number of questions correct based on the questions selected.

Within the set of knowledge questions used by the Vaulted Voice Verification protocol, there are also questions of the type that ask a user to describe an image. This is similar to the graphical passwords as discussed in [Davis et al., 2004]; however, in their research, the authors have users select from a set of images as opposed to describing the contents of an image, as is done for this research. Because the choice of words used to describe
Figure 5.18: This figure shows an example of the images used in the data collection efforts. A small random sampling of 30 users from the data collection revealed 18 unique descriptions for this image. In this sampling, only one description was repeated: “tree” (for this sampling, “a tree” is considered the same response as “tree”).

the image is left completely up to the user, the $R$ value for this type of questions could, at first, be considered to be a factor of the total available vocabulary. However, we know that, for a given image, the vocabulary used in its given descriptions are a highly reduced subset of the total vocabulary.

Another important aspect of the image description types of question is showing the diversity of vocabulary used to describe the same image. An example of this is seen in the ongoing data collection related to this research, described in Section 6.3.3. As a part of the data collection effort, the users are asked to describe different images. Figure 5.18, for example, shows one of the images a user is asked to describe. Even for an image as simple as this, a random sampling of 30 different users generated 18 unique descriptions. Of the
18, unique descriptions, only 1 was repeated. Based on the small random sampling, the image in Figure 5.18 generates a guessing entropy of 6.1337, as shown in Equation 5.49.

\[
G(x) = \sum_{i=0}^{18} (i \cdot P_{\text{answer}})
\]

\[
= \left( 1 \cdot \frac{14}{30} \right) + \left( 2 \cdot \frac{1}{30} \right) + \ldots + \left( 18 \cdot \frac{1}{30} \right)
\]

\[
= 0.4667 + 5.667
\]

\[
= 6.1337
\]

Comparing that to the guessing entropy of a standard multiple choice knowledge question (4 choices with \( \epsilon = 0 \)), which generates a guessing entropy of 2.5, we see that even with simple images, description-based knowledge questions are able to generate significant entropy.

**Knowledge Infused Biometric**

With Vaulted Voice Verification there also exists the notion of the hybrid knowledge-infused biometrics question. As its name implies, such questions require matching the correct biometric while answering the question correctly. The analysis for such questions remains the same as both the strict knowledge or biometric question, but the calculation of the random chance value, \( R \), varies.

The \( R \) value for these types of questions is calculated by combining the values for the
Table 5.4: This shows an example of what a knowledge infused biometric question looks like. The user must answer the question by selecting the appropriate answer to complete the sentence. The response is considered correct if the user is able to select the correct answer, and their biometric matches the appropriate model.

knowledge and the biometric questions. The random possibility of getting a knowledge-infused biometric question, $R_{kb}$, correct becomes:

$$R_{kb} = R_k \ast R_b$$  \hspace{1cm} (5.50)

where $R_k$ is the random chance of getting the knowledge portion correct, and $R_b$ is the random chance of matching the correct biometric.

The knowledge infused biometric question allows for the incorporation of both the biometric and the knowledge into a challenge question. Such a challenge question allows for multiple factors to be addressed for a single question. For example, to successfully authenticate using a series of knowledge-infused biometric questions, a user needs to both answer the questions correctly and match the correct biometric model for said answer.

At its most basic level, what the knowledge-infused biometric question does is increase the possible options on which to match. In terms of shuffling and matching, what
the knowledge-infused biometric question does is increase the possible answers from which an attacker must select in order to correctly respond to the question. For example, if the question presented to the user gave them 4 different colors to choose from and asked them to repeat the sentence, as shown in Table 5.4 where they needed to fill in the blank by speaking the correct color of the 4 presented, the first thing the user would need to do is select the correct color. If the user selects the correct color, the audio model generated would then need to match the correct model. Given 4 models from which to match per question, there would exists 16 possibilities that an attacker would need to select from. Each possible answer (of the 4 colors) could even contain the same 4 biometric models from which to choose, but without the correct answer or the correct biometric, the chances would not be improved.

To calculate the security generated by this type of question, the same logic would be applied as in the case of the biometric. The difference in this case is that number of options available from which to select to match the correct answer is increased. So, for a multiple choice question of 4 choices, each choice containing 4 possible models to match, the resulting guessing entropy would be similar to that of a 16 choice wide biometric question. However, it would not be the exact same unless the $\epsilon$ advantage were the same for the biometric and knowledge portion of the question.
Defense In Depth and Width

With the index-based approach of the Vaulted Voice Verification protocol, the security gained by a set of questions is limited only by how well the underlying matching protocol is able to discriminate between users. We examined how the number of models to match per question, the width, influences the security of the protocol. For this, we considered a scenario that more closely relates to a real world consumer test: security given considering that an authentic user can make at most 1 error. The security in this graph refers to the number bits needed to represent the number of attempts an attacker would need to make in order to gain access.

We designed the scenario, as seen in Figure 5.19, at four different question widths (width referring to the number of possible underlying matching models). For the number of questions asked, the depth, we looked at values from 5 to 20 and considered, based on the underlying distribution, how many questions the authentic user would need to get right before their error tolerance reached 1 bit (they would need to answer more than 1 question again).

As shown in Figure 5.19, the security gained from increasing the width of the questions is significant. At as few as 10 questions, the gain in security bits roughly doubles. Also, due to the fact that the error rates, in terms of model matching, are a factor in the underlying biometric comparison algorithm, increasing the width does does not have a
Figure 5.19: This plot shows how both depth, in terms of numbers of questions, and width, in terms of number of possible answers, affect the amount of security given by the Vaulted Voice Verification protocol. The plot shows the maximum amount of security when holding the error tolerances (number of incorrect responses allowed from an authentic user) to 1 or less. The value for $\epsilon$ and $\beta$ in this table are 0.025 and 0.06, respectively.

The security as listed in Figure 5.19 shows that with a width of 32, 10 questions gives just over 40 bits of security. When compared to a standard password, in which a 7 character case-sensitive password gives you 40 bits, we see that the Vaulted Voice Verification protocol offers comparable security.

Encryption

To discuss the security provided by the different encryption layers of the protocol, we must first constrain the discussion to the three states that the data is in while encrypted. While in use in the Vaulted Voice Verification protocol, the data is either on the user’s
device, in transmission, or on the server. In these three states, the data is protected by the user’s password based encryption key $K_p$, the server’s public/private key based encryption $K_s$, the user’s public/private key based encryption $K_u$, or some combination thereof. In this work, we assume that both RSA and SHA-256 in and of themselves are secure and possess sufficient entropy. For a deeper insight into RSA and SHA-256, we refer to [Jonsson and Kaliski, 2003] and [Gilbert and Handschuh, 2004].

In terms of entropy, the security added by the user’s encryption relies on the user generated password. Thus, the strength of the encryption, in terms of security provided, depends on the strength of the user’s password. Using our previous discussion as a guideline, the entropy of the password that a user selects is similar to the entropy of the rest of the Vaulted Voice Verification protocol; however, some differences exist.

According to the NIST publication [Burr et al., 2006], the calculation of password entropy is determined by two items: the length in characters and the size of the alphabet. In [Burr et al., 2006], the authors give the following formula to calculate entropy passwords:

$$H = \log_2(b^l)$$

(5.51)

where $b$ is the size of the vocabulary and $l$ is the length of the password. It is important to note that this formula is given for randomly selected passwords that have a uniform random distribution.
For user generated passwords, the authors refer to the “guessing entropy estimate“ instead of formula 5.51. In this context, the guessing entropy is equivalent to figuring out how hard it is to guess your password. The standard entropy, which is also known as the Shannon entropy, differs from the guessing entropy because the guessing entropy focuses more on the difficulty of guessing versus the uncertainty. The guessing entropy is related to the Shannon entropy, as introduced in [Shannon, 1944]. As stated in [Davis et al., 2004] and [McEliece and Yu, 1995], given the guessing entropy $H(X)$ and the Shannon entropy $G(X)$ for some random variable $X$ taking on values in $\mathcal{X}$, the relationship is as follows:

$$H(X) \geq \frac{2\log |\mathcal{X}|}{|\mathcal{X}| - 1} (G(X) - 1)$$

(5.52)

which is to say that the guessing entropy is a lower bound of the Shannon entropy.

In terms of how the data is encrypted, we refer to the three states in which the data resides: on the user’s device, in transit to/from the server, and on the server. For this, we first examine the data stored on the user’s device. The data in question includes the user’s table data (model and client tables), $D_t$, and the server table (after a complete enrollment), $D_s$. The user’s table data is protected by the user’s password based encryption. The server table is protected by the user’s password based encryption and the server’s key. Even the valid user can not access/view the server table while it is stored on their device. Thus, for
the stored state on the user’s device:

\[ D_t = K_p(data) \]  \hspace{1cm} (5.53)  

\[ D_s = K_p(K_s(data)) \]  \hspace{1cm} (5.54)  

During transmission, the data is protected by \( K_s \) and \( K_u \). When the user sends data to the server, it first encrypts it with its private key, then encrypts it with the server’s public key, letting the server know that the data came from someone with the correct certificate while ensuring that no one can see the data except someone with the server’s private key. In general, when the server sends data to the user, it is similarly encrypted; first with the server’s private key and then with the user’s public key. The exception to this occurs when the server sends the server table to the client for storage. During this transmission, the server first encrypts the table with its public key, ensuring that only the server can read the data.

\[ User: \ K_s(K_u(data)) \rightarrow Server \]  \hspace{1cm} (5.55)  

\[ Server: \ K_u(K_s(data)) \rightarrow User \]  \hspace{1cm} (5.56)  

Minimal data is stored on the server. The server stores only the user’s table, $D_u$. We assume that the server is generally secure, but not immune to attack. In general, this table need not be encrypted because it stores only publicly available information, until a verification takes place. While a verification takes place, the user’s table contains the information needed for a user to verify, and for this reason the table is also encrypted with the server’s key. Someone with authorized access to the server can access its data, but it is otherwise encrypted.

$$D_u = K_s(data)$$
Chapter 6

Can You Hear Me Now? Testing Testing

1-2-3

6.1 Expectations and Resources

With the newly developed Vaulted Voice Verification protocol, a user should be able to speak their answers, be recognized by them, and remotely verify their identity without their biometric data/templates leaving their device. Also, there should be reasonable confidence that an attacker can not impersonate an authentic user.

In order to fulfill expectations of this research, there are a number of resources required. First, we need an implementation of Vaulted Voice Verification upon which to
test the hypothesis. We need proper datasets containing voice samples of sufficient duration and quality. Lastly, we also need a tool or application that will allow us to collect data to generate our own dataset.

In the remainder of this chapter, we will look at different aspects of the research in terms of the three major needs: implementation, data, and evaluation.

6.2 Vaulted Voice Verification Implementation

The Vaulted Voice Verification protocol is currently implemented in multiple forms. The implementations utilize the latest revision of the Vaulted Voice Verification protocol, the index-based version. In general, the implementations take, or create, models from the audio source, generate the appropriate tables, and communicate with the server for verification. In this section we will look at the implementation and discuss their creation.

The initial implementation is in C++ using a client server model. The implementation consists of a client that simulates running on a mobile device, and a server that handles the verification requests. The implementation loads audio data from a dataset and processes it as if it were live. The purpose of this implementation is to ensure repeatability and consistency of experiments.

Another implementation of the Vaulted Voice Verification protocol, similar to the initial implementation, is developed in C++ and allows live testing. This implementation also simulates a mobile device that is communicating with a server, but it allows a user to
interact with it and give live responses.

A third implementation of the Vaulted Voice Verification protocol is developed as a combination of C++ and Java. This third implementation allows for live data collection and experimentation on a mobile device. For this implementation, a java application was created to run on Android mobile devices. This application interacts with the user, recording live audio responses, and initiates the Vaulted Voice Verification protocol in an attempt to verify with the remote server.

6.2.1 Server

The server side of the implementation contains everything necessary to perform the server end of the Vaulted Voice Verification protocol. It is a multithreaded C++ application. The design includes a few modules to accomplish the given tasks. The system accepts a new connection, determines if the new connection is an enrollment or a verification and acts accordingly.

If the new connection is an enrollment, the server will compute the necessary hashes, add the necessary nonces, and store the data in its table of users. If the new connection is a verification, the server will generate the appropriate challenge by selecting questions, shuffling the questions according to a generated random string, and sending the scrambled challenge back to the client.
6.2.2 Client

The client implementations differ depending on their purpose. One client is designed for reading datasets, one is designed for live interactions at a computer, and the last is designed for live interactions on a mobile device. In contrast with the server, the client performs the majority of the work required to enroll and verify. In general, the different client implementations will do some subset of the tasks described in this section.

Model Generation

There are two types of models used by the Vaulted Voice Verification protocol implementations: text-dependent and text-independent. Two different software packages are utilized to generate the two types of models: CMUSphinx [Lee, 1989] and Mistral/Alize [Bonastre et al., 2008]. The CMUSphinx package, generally referred to as Sphinx in this document, is primarily designed for speech recognition, so it utilizes text-independent models. The Mistral/Alize package, referred to as Alize from here, is primarily designed for speaker recognition and primarily utilizes text-dependent models. The pipelines are similar in that they turn audio files into models, but they do contain differences.

For the Sphinx model generation pipeline, the implementation follows Figure 6.1. The features are extracted from the audio sample, some statistical analysis is done on the features, and the resulting values are then used to adapt the current model before
Figure 6.1: This figure shows the basic pipeline for processing audio files using the CMUSphinx software. For this research, CMUSphinx is used to generate the (text-independent) GMM models.

printing out the final model. In the Sphinx pipeline, a model is adapted and updated using incoming audio files and the matching transcription.

For the purposes of this research, the Alize pipeline features two different types of models: GMMs and i-vectors. For comparison, both types of models are included in the implementation. The implementations of the Alize pipelines are shown in Figure 6.2. The features are first extracted from the audio files, then the energy normalization takes place. At this point the two implementations diverge. For the GMMs, a world model is created along with individual models for testing purposes. For the i-vectors, the total variability matrix is created along with the individual i-vectors for testing.
6.3 Datasets

There are a number of things to look for when locating/creating a dataset for testing the Vaulted Voice Verification protocol. The Vaulted Voice Verification protocol utilizes both
text-dependent and text-independent questions and models; therefore, any dataset used to test the protocol needs to contain data from multiple people saying a variety of things multiple times.

Some of the issues encountered with finding a proper dataset with which to test the Vaulted Voice Verification protocol are the lack of mobile specific data, the quality of the recordings, and the number of participants in the data collection effort. With such issues, the available datasets are extremely limited. In this research, we compare against some of the latest available datasets and also work to produce a new, crowd-sourced, mobile-specific dataset.

### 6.3.1 MIT Mobile Dataset

The MIT mobile device speaker verification corpus [Ram H. Woo and Hazen, 2006](MIT dataset) is the dataset selected for the initial tests. The reasons for choosing this dataset for the initial work are two fold: it appears in the most recent work in secure voice templates, and it closely represents the type of voice data desired to properly examine the Vaulted Voice Verification protocol. Because [Inthavisas and Lopresti, 2011b](Inthavisas and Lopresti, 2011b) is the most current work to utilize this dataset, it is used as the state of the art benchmark against which to measure the initial results.

The MIT dataset is comprised on 48 speakers: 22 female and 26 male. Short phrases, names and ice cream flavors, were recorded in 20 minute sessions. Enrollment data was
created in two separate sessions. The imposter data was recorded in a separate session. Each person has a dedicated imposter, meaning all imposter files for a user come from one person instead of each file being made from a different person’s voice.

As mentioned, this dataset is the closest matching publicly available dataset designed for the mobile environment. However, the dataset is too limited in size, both in the number of participants and the variety of words and phrases. The limitations of the MIT dataset inspired the exploration of other datasets that could be used to test the Vaulted Voice Verification protocol, as well as the creation of a new dataset for use with text-dependent and text-independent speaker verification systems in the mobile environment.

6.3.2 MOBIO

The MOBIO dataset [McCool et al., 2012] consists of both audio and video data recorded from 152 people. For the purposes of this research, only the audio portion is of use. The dataset contains 100 male and 52 female participants with data from participants collected at six different sites in five different countries. The dataset collection contains two phases and contains both native and non-native English speakers.

The data collection consists of 12 sessions per participants: the first 6 sessions for their phase 1 questions and the other 6 sessions for their phase 2. Phase 1 consists of 21 different questions from different question categories. The question categories are short response free speech, short response questions, set speech, and free speech. Phase
2 consists of 11 different questions from the categories of short response question, set speech, and free speech.

The data captured in this dataset was recorded using a mobile phone, a NOKIA N93i, and a laptop computer, a 2008 MacBook. The first session contains a mixture of data from the laptop and the mobile phone.

According to [Khoury et al., 2013], using current technologies, the state of the art equal error rates on the MOBIO dataset are between 10-20%. We show that, using the Vaulted Voice Verification protocol and the standard i-vector approach (default values without adjusting parameters) for model generation, we achieve a 15% error rate, within the error rates given in [Khoury et al., 2013], when matching users across different different sessions and locations (as opposed to separate evaluation for location or session).

6.3.3 UCCS Voices

As a part of this research, we are developing a new dataset to release to the general scientific community. The new dataset, UnConstrained Crowd Sourced Voices (UCCS Voices for short), represents a novel dataset that includes mobile data captured in the wild from a large sampling of devices. While the impact of this research stands by itself, there currently does not exist a dataset that can, on its own, fully test the different aspects of this research. The dataset is created specifically using mobile devices to capture data to be
used with text-dependent and text-independent voice research and to test the knowledge-based security added by description-based Vaulted Voice Verification questions.

The dataset consists of 160 unique prompts that users respond to. The prompts are recorded in sessions, 20 prompts at a time. The prompts are divided into one of three categories: a question to answer, a phrase to repeat, or an image to describe. For the questions to answer and the images to describe, the user must type their response after they’ve recorded it. For the phrase to repeat, the user simply needs to repeat the exact phrase.

For a complete collection from a user, the user must run through the list of 160 prompts 5 times, for a total of 40 individual sessions. The user is allowed to record the sessions in any audio environment they wish using their mobile device. After each session, the user must fill out information regarding the session. This information includes the location (indoors/outdoors, quite/loud) and microphone type (internal mic, headset, or bluetooth). Each user is also required to fill out a short profile that includes different data points so the resulting dataset will be useful to other members of the voice biometric community. An example of the captured data points is located in Table 6.1.

Each audio sample recording possesses a matching metadata file that contains all necessary information regarding the audio sample. The metadata is post-processed after submission and checked to ensure as much accuracy as possible.

The dataset is specifically designed to test the aspects of Vaulted Voice Verification
Table 6.1: The metadata collected by the VVV Data Collection Android application used to generate the UCCS Voices dataset. The entries on the left are the names of the data fields. The entries on the right are the corresponding values. In this example, the type of question being asked of the user is to repeat the words displayed on the screen. The task is to repeat the phrase “Small Red Bean.” For these types of tasks, the user’s answer should mirror the task, so a user provided answer is not applicable. The sample length is 2251 milliseconds. The remaining data is a combination of the user’s profile, the date the sample was made, the job number, and the device type.

<table>
<thead>
<tr>
<th>Question</th>
<th>Repeat the word(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>Small Red Bean</td>
</tr>
<tr>
<td>Answer</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Time:</td>
<td>2251</td>
</tr>
<tr>
<td>Username</td>
<td>User014</td>
</tr>
<tr>
<td>Age</td>
<td>22</td>
</tr>
<tr>
<td>Gender</td>
<td>Male</td>
</tr>
<tr>
<td>Height</td>
<td>5’6</td>
</tr>
<tr>
<td>Weight</td>
<td>120</td>
</tr>
<tr>
<td>Smoker</td>
<td>Non-Smoker</td>
</tr>
<tr>
<td>Dentures</td>
<td>None</td>
</tr>
<tr>
<td>Speaker</td>
<td>Non-Native</td>
</tr>
<tr>
<td>Dialect</td>
<td>French</td>
</tr>
<tr>
<td>Hearing</td>
<td>No</td>
</tr>
<tr>
<td>Location</td>
<td>Indoors, Quiet</td>
</tr>
<tr>
<td>Mic</td>
<td>Internal Mic</td>
</tr>
<tr>
<td>Date</td>
<td>Sat Feb 01 12:23:17 MST 2014</td>
</tr>
<tr>
<td>Job Number</td>
<td>1</td>
</tr>
<tr>
<td>Device Manufacturer</td>
<td>rockchip</td>
</tr>
<tr>
<td>Device Model</td>
<td>CT920</td>
</tr>
</tbody>
</table>
that no other dataset can, as well as to generate data for others to make use of. For example, one of the aspects of the security of Vaulted Voice Verification comes from user’s varying descriptions of an image. To our knowledge, no other datasets currently exist that contain audio captured on mobile devices in which users freely give descriptions of different images.

The data collection effort relies on users being freely able to submit data at their leisure. As a result, not all users give a full 40 sessions. While such user data is incomplete, the data from such users remains valuable for use as distractors and importers in the testing; using such data helps to enhance the realism of the testing.

Another interesting aspect of the dataset is that crowd-sourcing was used in its creation. An application to collect the data was put in the Google Play store and users were allowed to freely download and submit data. The services of Amazon Mechanical Turk (MTurk) were also utilized to incentivize various individuals to submit data to the dataset. With the design of the data collection, the resulting dataset is intended to grow continually.

Another interesting aspect of this new dataset is the potential to test knowledge-based questions. Other voice datasets have either questions that the user responds to or prompts that the user reads. To our knowledge, this is the first dataset that possesses image-based knowledge questions to which the user responds.
6.4 Analysis

6.4.1 Initial Text-dependent Results

Performance is measured in terms of the false accept rate (FAR) and the false reject rate (FRR). FAR measures the percentage of people that are allowed access to the system when they are posing as someone else. FRR measures the percentage of people who are falsely denied access to the system. Where these values meet when plotted against each other is called the equal error rate (EER). It is crucial to mention that the EER in Vaulted Voice Verification only applies to security after the attacker has cracked/compromised the keys ($K_U$ and $K_S$). Before these keys are compromised, an attacker would get nothing.

In these experiments, we are assuming all keys have been compromised and are examining the amount of security Vaulted Voice Verification adds on top of the encryption. This is what Lopresti [Inthavisas and Lopresti, 2011b] calls Scenario II, for which he achieved 11% EER. A test was run to make a direct comparison to the work done by Lopresti [Inthavisas and Lopresti, 2011b, Inthavisas and Lopresti, 2012]. This test also assumed that all keys had been compromised, as was done in Scenario II of Lopresti’s work. This test did not cover the scenario in which the passwords and keys were not compromised because, if the data is not decrypted, no valid answer can be supplied by the attacker. This test was implemented by comparing the proximity of $g$ to $pr$ and $pr_i$.
Figure 6.3: ROC plots from a baseline algorithms on Scenario-II as well as Vaulted Voice Verification on a larger “all-vs-all” scenario with a larger imposter set. Shown as points are Vaulted Voice Verification and Lopresti’s work on Scenario II, with an ERR of 0% and 11% [Inthavis and Lopresti, 2011b] respectively. This plot was created using the MIT dataset.

for each given phrase for an individual. The $pr$ and $pr_i$ pairs were presented in a random order and $g$ had to decide which it was closest to and respond accordingly. For apples-to-apples comparison, we also used the MIT data to generate our plot. In this test, there were only two choices per phrase, so each response was binary. In this test an EER of 0% was generated. Plotting a 0% EER does not illustrate anything compelling, so it was plotted as a single point on Figure 6.3. It is necessary to note, however, that this test directly compares to the test in which Lopresti reported 11%, but the ERR of 0% is unrealistic because of the small number of impostors.

To provide a stronger experimental validation, we implemented a new baseline algorithm, which is not privacy/security enhanced, and expanded the testing to a larger test
Figure 6.3 shows the ROC curves for these test results. There are results from the two baseline tests and results from the two Vaulted Voice Verification expanded tests. The baseline tests were created by comparing the scores between \( pr \) and \( g \) in the same manner as Vaulted Voice Verification. In the baseline tests, however, only raw scores were recorded, removing the inherent pairwise thresholding that is added by Vaulted Voice Verification.

To generate the data for the first baseline curve, each person in the gallery was tested against \( pr \) and their dedicated imposter, \( pr_i \). In these tests, the variance of \( g \) is used. This gives an approximate equal error rate of 8%. A second baseline was also generated (labeled Baseline 2). This baseline was created to show a similar separation of data exists when using the variance of \( pr \) as opposed to using the variance of \( g \). The EER of this baseline test is 6%. Both of these results show an improvement over the state of the art benchmark of 11%. We note, however, that the baseline has no privacy enhancements.

Vaulted Voice Verification performed significantly better than the baseline because Vaulted Voice Verification has an inherent pairwise thresholding step which allows it to make every decision on a case by case basis. For demonstration purposes, we give the following scenario of two phrases. In this scenario, let us assume that a score can range from 0 to 10. For the first phrase, \( ph_0 \), \( g \) generated a score of 5 for \( pr_0 \) and a score of 3 for \( pr_{0i} \). For the second phrase, \( ph_1 \), \( g \) generated a score of 9 for \( pr_1 \) and 6 for \( pr_{1i} \). According to the baseline scores, no clear threshold exists that would separate probes and
their impostors. However, because Vaulted Voice Verification thresholds on a case by case bases, it is able to identify and discern between the probes and impostors for both the first and second phrases.

To better estimate the EER of the Vaulted Voice Verification protocol, a second set of tests was performed. These tests are labeled in Figure 6.3 as all vs all. In the first all vs all test, labeled Voice Verification -all vs all, we tested all people in the gallery against all \( pr \) and \( pr_i \) pairs. By doing so, we have expanded the negative matches to more thoroughly examine the false match rate of the algorithm. In keeping with the tests performed to generate a baseline, a second all vs all test was performed. This test, labeled Voice Verification 2 - all vs all, was done using the variance of the probe and imposter against the gallery data. As Figure 6.3 shows, these two tests have an EER of approximately 6%.

### 6.4.2 Initial Mixed Text-independent and Text-dependent Results

For these experiments, the data was separated to yield gallery and probe sets that contain separate audio samples. At random, 60% of the enrollment data was designated as gallery and the other 40% as probe, ensuring an appropriate balance between sessions. The impostor data was used in its entirety. In the tests, the data are further separated on a per-phrase basis; this way, speech-dependent models could be created.

Our tests are conducted by testing every person against all other people in the gallery.
This method of expanding the dataset is referred to as “all vs. all” testing in the original work. In early work, we performed same sex and mixed set testing; same sex testing yielded higher accuracy. However, the individual same sex sets are small, and the GMMs easily discriminate the data within them. We felt that since mixed set testing increases the data available for testing, as well as the potential for collisions, it was a more realistic case to report. The comparison, or scoring, of the models is done using a z-score, as in the original work. For comparison, tests were first performed on the original text-dependent binary versions of the baseline and the Vaulted Voice Verification algorithms.

Fig. 6.4 shows four different Linear-Scale DET (LDET) curve plots generated with the MIT dataset. The figure contains a baseline and highlights three different variants of Vaulted Voice Verification with varying question types and text-dependent and text-independent challenges. An operational system would likely fuse multiple choice text-dependent and text-independent models with the number of each type, dependent on the desired security model. The three plots will be examined and discussed in terms of the knowledge-based security (K-bits) and biometric identity security (B-bits).

The plot labeled “Baseline GMM” is generated by scoring each binary challenge-response pair (binary meaning two possible answers to choose from) and applying a threshold across the results. The curve represents a baseline result for a single question model, i.e. a single general threshold is applied for each question. It has an approximate equal error rate of 8% and represents the error rate of the biometric matching (B-bits of
security). To see how this value translates into actual bits, we refer readers to 5.4.2 and 5.4.2.

The binary Vaulted Voice Verification plot is generated by applying the Vaulted Voice Verification protocol to binary challenge-response pairs. In Vaulted Voice Verification there is an inherent pairwise thresholding that takes place. This pairwise thresholding allows Vaulted Voice Verification to account for variation from phrase to phrase. For each phrase, a different threshold is applied based on which model is the closest. As shown in the figure, Vaulted Voice Verification outperforms the baseline for binary challenge-response pairs. This curve shows an approximate equal error rate of 6% and also represents $B$-bits of security.
The next curve we will discuss is the one generated from multiple choice questions. For this experiment, there are four possible answers for each question. Multiple choice questions expand the security of the overall system by adding $K$-bits of knowledge-based security. This Vaulted Voice Verification curve improved from the binary case, with an approximate equal error rate of 1%; with multiple choice questions, there are $B + K$ bits of security. This means that, on top of the biometric security (something you are), additional security is provided by knowledge (something that you know).

The final curve that we will discuss is the text-independent curve. This curve not only represents $B + K$ bits of security, but also includes security added by repeating a random passage verbatim. The curve is generated by comparing models that are created from all phrases in the gallery for each user against models that are created from all the probe phrases for each given user. As such, for each user there is one real model and one impostor model, as in the case of the binary text-dependent models. For Vaulted Voice Verification, there is a single bit for each challenge response pair, resulting in a 4% equal error rate. The overall system security is improved because the security for this comes from balancing the biometric bits of security with the ability to accurately speak the correct passage to generate the model. This greatly reduces the possibility of a movie-style replay attack.

Many LDET curves are presented as false accept versus true accept rate. We instead chose to present our results as match rate versus non-match rate because acceptance is
Figure 6.5: DET plots for the MOBIO data for Vaulted Voice Verification as well as two baselines. The males only trial, the female only trial, and the mixed trial with both males and females. The trials were 8 choices wide and consisted of 394 male and 199 female trials. The two baseline plots represent the male and female plots using standard thresholding. The separation of the data follows the protocol of [Khoury et al., 2013].

a function of the overall system – here we are analyzing only the biometric matching component. One can estimate the impact of the other layers by rescaling the false match rate by $2^{-N}$ where the un-compromised layers provide $N$-bits of security. For example, in attack model 4, the system would still have $P + K$ other bits of security. So, if $P=10$ and $K=10$, the system’s false accept rate scales the false match rate by a factor of $1/1,000,000$.

6.4.3 MOBIO Analysis

For the analysis of the MOBIO dataset, we follow the same protocol as used in [Khoury et al., 2013]. The data is separated by gender and then further refined by world, training, and testing data. The world model is constructed from 7,104 audio files for males and
2,496 files for females. 190 files were used to generate 38 unique training models for the males, while 100 files were used to generate 20 uniques training models for the females. The total number of tests for male participants was 151,620; for females participants, there were 42,000 total tests.

The Vaulted Voice Verification tests were conducted using the same split of data. To construct the gallery vaults, each authentic model was paired with 7 impostor/chaff model from others and were selected randomly. Text-independent models were generated and used with the index-based Vaulted Voice Verification protocol. We chose to use text-independent models in the same manner as others so to make a more direct comparison between our work and others. A toy example of the vault construction is shown in Figure ref:toy-vault.

As Figure 6.6 shows, using an off the shelf recognition system, Alize, with no optimizations, we achieved around 15% equal error rate using 8 choice wide trials on the MOBIO dataset. This compares to the 10-20% error rates as reported in [Khoury et al., 2013]. Our results were achieved using an off the shelf version of Mistral/Alize for i-vector model generation and applying the index-based Vaulted Voice Verification protocol to those models. The results given in [Khoury et al., 2013] were generated by optimizing the model generation and scoring for the best possible equal error rates. We again mention that these results only highlight the biometric error rates. Vaulted Voice Verification has multiple other layers of security on top of the biometric matching.
Figure 6.6: Example of how the VVV vaults are created using the VVV protocol. The top row shows the enrollment files for a series of users. The second row shows the vaults created for each of the users using their enrollment data. For each column, the correct match is in bold. At the bottom user1 is scored against the vaults for userA and userB. User1 gets 1 of 4 correct for vaultA and 4 of 4 correct for vaultB. With this, user1 would successfully verify as userB.
6.4.4 UCCS Voices Analysis

For the analysis on the UCCS Voices dataset, we have data from a total of 172 participants (104 male, 68 female). The age ranges of the participants are from 18 to 65. As of the writing, we’ve collected a total of 15,760 voice samples from various users.

The DET curve on the UCCS Voices data within the Vaulted Voice Verification protocol, using 4 wide models can be seen in Figure 6.7. For the tests, an off the shelf version of the i-vector implementation from the Mistral/Alize software package was used. The models used to generate the figure are text-independent models based on the different sessions for the different users.

For each of the image description type prompts, many different descriptions were
gathered. The diversity of the descriptions help to show how effective such prompts can be in different security applications. For example, the image of the tree (as shown previously) has, at the time of this writing, generated 76 unique descriptions. What makes the number of unique descriptions interesting is the shape of the distribution. It follows the well known Power Law distribution [Mitzenmacher, 2004]. The distribution of the different descriptions can be seen in Figure 6.8.

The results from the data collection have led to interesting observations about the construction of image-based security questions, among other things. For example, what would the results look like if the users were instructed not to use the words “green” or “tree” in their descriptions?

### 6.4.5 Effects of Repetitions

It is commonly known that an increased amount of data will lead to improved stability in the resulting models. When using simple phrases to generate models, there is the possibility of not having enough information to produce a stable model. To analyze the effect of length of audio data and model matching stability, we generate multiple models, each with a different length of audio data, and compare the results.

For these experiments, the data was separated to yield gallery and probe sets that contain varying lengths of separate audio samples. Because the data is generated using words and short phrases (along with the fact that the Vaulted Voice Verification protocol
Figure 6.8: This is an example of the results from the image description knowledge-infused biometric prompts in the UCCS Voices dataset. With data from 172 users, a simple image like this generated 76 unique descriptions using 120 unique words to describe this image. The distribution follows the power law. The most common description, “Tree” occurs 67 times, and is followed by “Green Tree” with 33 times.
uses word and phrase-based questions), we vary the length of audio samples by using
word repetition within from the given dataset. The models generated for these tests are
text-dependent models from the MIT dataset. The testing is done on a phrase by phrase
basis, i.e. the compared models are generated from different users speaking the same
phrase. For each test within the experiment the probes tested were generated from the
same number of utterances.

The audio samples were separated such that for each $N$ samples per phrase, there
exists a gallery model generated using from 1 to $N-1$ samples, and corresponding probe
models generated using $N-1$ to 1 samples. The separation was done in a manner that
ensured no file is simultaneously in the probe and gallery model files in a given set; the
reason Table 6.2 is only half full. For example, if a user repeated a phrase 10 times, then
one set would have 6 repetitions used to generate the gallery model and 4 to generate the
probe, while another set would have 1 repetition used to generate the gallery and 9 to
represent the probe.

The experiments done to test how various lengths of audio input affects the model
matching accuracy, as described in this section, are illustrated in Table 6.2, which results
from testing every person against all other people in the gallery. This method of expanding
the dataset beyond the dedicated impostor testing is referred to as “all vs. all” testing
in [Johnson et al., 2013b]. The scoring methods used in these tests are $z$-scores. As Table
6.2 shows, as the number of repetitions of the utterances used to generate the gallery and
Table 6.2: In this table we see the effects of the number of utterances used in generating models on the equal error rate of the protocol. The columns represent the number of utterances used to generate the probe models. The rows represent the number of utterances used in generating the gallery models. The maximum number of utterances within the MIT dataset for phrases is 12; therefore, the chart shows values from 1 to 11. This data was generated using the biometric matching portion of the Vaulted Voice Verification protocol with random 4-wide matching choices (1 matching and 3 randomly selected non-matching models).

The accuracy also increases.

Data and VVV

In this section we used different datasets to examine the different aspects of the Vaulted Voice Verification protocol. We used the MIT dataset to examine how Vaulted Voice Verification does on text-dependent and simple text-independent speech. We used the MOBIO dataset to examine how well Vaulted Voice Verification does on text-independent speech. We also introduced our ongoing data collection effort for the UnConstrained
Crowd Sourced Voices (UCCS Voices) that will help to further highlight the impact of the Vaulted Voice Verification protocol. We’ve seen that the Vaulted Voice Verification protocol is able to give security comparable to standard passwords while having error rates on standard datasets that are comparable to the state of the art.
Chapter 7

Conclusions

In this work, we addressed the problems as laid out in Section 1.1. We introduced a novel protocol, Vaulted Voice Verification, that is able to securely and remotely verify an individual using something they know, something they are, and something they have, without any of their biometric data leaving their possession. We achieved this with minimal communication overhead and a minimal footprint on the remote server.

The main contributions of this research are:

1. Adaptation of Vaulted Verification to voice to provide a privacy enhanced remote protocol using voice. [Johnson et al., 2013b]. Vaulted Verification was introduced in 4.3 and the adaptation was detailed in Section 5.1.

2. Extension of the types of match protocols used in Vaulted Verification to include different types of security knowledge, including knowledge infused biometrics.
3. Combination of text-dependent and text-independent models into a protocol to enhance the overall robustness of speaker verification. We examined this in 5.2. [Johnson et al., 2013a]

4. Creation of an index-based challenge-response protocol in which no biometric data ever leaves the remote device and communication is minimized, as discussed in Section 5.3. [Johnson and Boult, 2013]

5. Creation of a privacy-enhanced biometrically authenticated remote key exchange protocol was discussed in Section 5.3. [Johnson and Boult, 2013]

6. Creation of a publicly available voice dataset made specifically for the mobile environment, discussed in Section 6.

7. Extension of prior binomial modeling to use guessing entropy based probability estimations of both the biometrics and knowledge infused biometrics.

8. Creation of a security and privacy evaluation of the impact of depth and width of Vaulted Voice Verification questions on overall bits of security is presented in 5.4 Section 5.4 using data from Section 6.

As shown, the Vaulted Voice Verification protocol is resistant to replay attacks because of its scrambled challenge response. It is also resistant to attacks via record multiplicity
because every challenge is unique and includes nonces, so no one challenge will ever resemble another at the data level. Because the keys generated using the Vaulted Voice Verification are not created until a challenge is complete, the protocol is also resistant to SKI attacks.

The Vaulted Voice Verification protocol is well-suited for real world applications. Such applications include mobile access to secure information, such as calling in to check on banking or credit card information, or biometrically secured person-to-person communication that does not require sending biometric data while trust is established. Vaulted Voice Verification is able to ensure that the person who attempts to access the account by phone is the person they claim to be. The protocol allows for such communication to happen without the need to ask the user an excessive number of questions for the purpose of obtaining a reasonable level of security.

7.1 Suggestions For Future Work

The Vaulted Voice Verification protocol, as introduced in this thesis, is applicable to a wide variety of uses not necessarily discussed in this research. The next logical research step would be to look into taking the Vaulted Voice Verification protocol and adapting it into the Biometric Key Infrastructure (BKI). Another possible direction would be to use the Vaulted Voice Verification protocol to verify with a local device, then use that verification as a key generation/release tool in a FIDO compliant manner [Baghdasaryan
and Hill, 2014]. Another interesting area of exploration for this research is that of multilingual users. There are numerous possibilities of ways to integrate different languages into the verification process. Mixing multiple languages with the different prompts is an interesting possible direction for future research with Vaulted Voice Verification.

In terms of performance, there are a number of issues that are beyond the scope of this research but are worth pursuing. For example, the use of the protocol on mobile devices requires that models are generated in real time. Currently, available technologies are not intended for such model generation and, therefore, suffer from non-optimal performance. Future research should look into optimizing the tools used for generating voice-print models to be specially suited/optimized for mobile devices.

Datasets better designed for testing actual voice based biometrics applications need to be consistent with their protocol. As first steps towards a dataset that is sufficient for the type of protocols purposed, we believe we are the first group to use Amazon Mechanical Turk to collect a mobile voice dataset specifically suited for biometric verification. Some of our lessons learned include how to properly incentivize and solicit people for participation in multi-session collection. It is also important to consider how the collection will be viewed by Amazon Mechanical Turk workers. For example, as surprising as it is, more people were willing to complete 40 sessions for a single $10.00 payment than were willing to be paid $0.50 per session for 40 sessions ($20.00). Future work will explore this dataset in alternative ways beyond that discussed in this thesis. For example,
a full protocol analysis with text-independent and text-dependent questions in an actual application setting.

The Vaulted Voice Verification protocol is designed to be independent of the biometric measured. However, each biometric comes with a specific set of issues to be tackled. With future research, the Vaulted Voice Verification protocol can be extended to work with other modalities. The newly introduced index-based protocol is designed to be extended to modalities such as face, fingerprint, signature, etc. Future work should be conducted for the purpose of including additional biometric identifiers, along with other identifiers, within the same framework.

The current scoring mechanisms utilized in the Vaulted Voice Verification protocol have no explicit measure for fusing scores from different modalities into a single value. Once another modality is implemented, readers are suggested to investigate how to combine the scores from different modalities. For example, if a user spoke a phrase and scanned their fingerprint, how should the protocol assess the quality of each input and then compare them? Research into this topic, such as the work on Meta Recognition, has been gaining momentum in recent years in an attempt to try to solve this very problem [Scheirer et al., 2011]. This is an extremely interesting area of research that can be applied to the Vaulted Voice Verification protocol as well as numerous other types of applications.

Obviously, the field of speech and voice recognition still has a long way to go. Future
work should research improving the pattern recognition methods used for both speech and voice recognition. While the field has made some interesting advancements in recent years, from speech-to-text software for desktop machines to speech recognition engines on mobile phones, such as Apple Inc.’s Siri or Google Inc’s speech recognition engine, there is much room for improvement.

In ideal conditions, current technology does quite well at recognizing speakers and their speech. However, given real world scenarios, current technology is still a far cry from being perfect. Improving this piece of the puzzle is crucial to the viability of protocols like Vaulted Voice Verification.
Bibliography


IEEE.


Appendix A

UCCS Voices Data Collection

In order to perform the data collection, we first needed to get it approved by the Institutional Review Board. The research was approved as “Vaulted Voice Verification, IRB #13-216”. Once approved, an app needed to be developed to collect the necessary data. To incentivize users to submit data, jobs were posted to Amazon Mechanical Turk. Once we had enough data, it needed to be properly sorted and anonymized to protect the identities of all who chose to participate.

A.1 App

A mobile app was written to perform the required data collection for this research. The initial version of the app was written specifically for the Android operating system. At the time of this writing, another version of the app is currently in development for iOS
The application walks users through 160 different prompts, 20 at a time. There are three different types of prompts: repeat the word(s), describe the image, and answer the question. For any set, the user is given a mixture of the prompts. An example of a prompts, a phrase to repeat, is given in Figure A.1.

The audio samples are recorded as wav files. The wav files have a sample rate of 16000kHz, are single channel, 16 bits per sample, and 16 bit PCM.
Figure A.2: This figure shows different screenshots from the “VVV Data Collection” app created specifically to collect audio data for this research. The top screenshot is of the main screen of the app. The middle image shows the profile information. The bottom screen shows the submission screen.
A.2 Mechanical Turk

Amazon Mechanical Turk (MTurk) has proven itself to be an invaluable tool for researchers and companies alike in their quest to encourage the completion of simple tasks. So, to incentivize the submission of data, we posted jobs to MTurk. The primary difficulty of using Mechanical Turk was figuring out what amount to offer for each job. We settled on a 3-tier payment structure to encourage as many submissions as possible. With MTurk in place, we were able to offer monetary incentives to participants while still keeping submissions anonymous.

A.3 Crowdsourced Collection

Another novel aspect of the data collection is that it is produced using social media and crowdsourcing instead of traditional means (having participants come and sit in a location for sessions). The benefits of crowdsourcing the data collection are many. The data gathered contains users recording the same phrases in multiple non-uniform environments. With this, the dataset more closely mimics real world conditions in which people interact with their devices. There are no restrictions/instructions/guidelines on how the user’s should interact with their device. The users are merely given screens that walk them through the recording processes.
For the data collection efforts, messages and information about the data collection were disseminated using various social media outlets (Facebook, twitter, etc...) as well as mailing lists and newsgroups. This allowed for a diverse (age, race, gender) population within the dataset.

Some drawbacks come from using crowdsourcing as a means of collecting data. One of the main drawbacks is that there is no control over the number of submissions from a given user. Some of the users, as of the time of this writing, have produced less than 5 sessions of audio data. Since this does not allow us to generate full models for these users, we used their data to create imposter models as well as the world (for GMM) and TotalVariability (for i-vector) models.

### A.4 Data Structure

The layout of the data from the data collection is structured based on username and prompt iteration. Users can choose to select or receive a username when submitting data, but those names are re-mapped to UserXXX, where XXX is a user number assigned to that user. Also, each prompt from the data collection application is re-mapped to promptXXX, where XXX is the prompt number. For each prompt, there is a audio file and a text file. The audio file is a 1600kHz wav file. The text file contains the matching metadata, as is described in Table 6.1.

After submitting a full 40 sessions, a user will generate 5 responses from the same
prompt. The different responses are divided into folders 00X, where 00X is the response repetition number, within the user’s folder. An example of the directory structure for the data is as follows:

```
User001
  001
    _prompt001.wav
    _prompt001.txt
    _prompt002.wav
    _prompt002.txt
    _prompt003.wav
    _prompt003.txt
    ...
  002
    _prompt001.wav
    _prompt001.txt
    _prompt002.wav
    _prompt002.txt
    ...
  ...
User002
  ...
  003
    ...
      _prompt158.wav
      _prompt159.txt
      _prompt159.wav
      _prompt159.txt
  004
    ...
      _prompt012.wav
      _prompt012.txt
      ...
  ...
User...
```
Appendix B

Working Vaulted Voice Verification

Implementation

What good is a protocol without an implementation? The Vaulted Voice Verification protocol was implemented on a mobile platform. The device used for implementation was an Asus T100 tablet running Android. The backend server was a standalone server implemented in C++. This appendix contains small example snippets of code taken from the existing implementation.

B.1 Client/App

The core Vaulted Voice Verification functionality of the app is in the table creation. Below are snippets of code that were used in the generation of the different tables for index-based
version of the Vaulted Voice Verification protocol.

The first thing done is to load the list of challenges for enrollment. The loading of the challenges is done differently in the different implementations, being either loaded from a file or pulled from the app’s stored data. The following is a small example of that:

```c
char *first, *type, *num, *question;
Challenge ch[numChallenges];
char buffer[BUF];
for(int i=0;i<numChallenges && fgets(buffer,BUF,f) != NULL;) {
    first = buffer;
    while(first[0]== ' ') first++;
    if(first[0] == '#' || first[0] == '
') continue;
    type = strtok(first," 	");
    num = strtok(NULL," 	");
    question = strtok(NULL,"\t\n");
    ch[i++].set(type, atoi(num), U.getID(), question);
}
```

Depending on the implementation, once the challenges are loaded, either the user is prompted with the challenges and their responses processed, or they are read from a dataset. Once the responses are received and the models are generated, we generate the model table:

```c
for(int i=0;i<numChallenges;i++) {
    for(int j=0;j<ch[i].getNum();j++) {
        modelRow mr;
        ch[i].getModel(j)->computeHash("SHA512",mr.hash);
        mr.model = ch[i].getModelData(j);
        U.addToModelTable(mr);
    }
}
```

Using the model table and the responses from the user (or data from the dataset), a temporary answer table is generated to make note of the correct/matching response. Note
that the answer table is never actually saved; the information is instead used to generate the remaining tables. By not saving the answer table, a record of the matching/non-matching response does not exist.

```cpp
for(int i=0; i<numChallenges; i++) {
    for(int j=0; j< ch[i].getNum(); j++) {
        answerRow ar;
        // set answer row's hash to match the models
        ch[i].getModel(j)->computeHash("SHA512", ar.hash);
        if(correct[i] == j) ar.correct = true;
        else ar.correct = false;
        memset(ar.question,0,1024);
        sprintf(ar.question,"%s", ch[i].getQuestion());
        U.addToAnswerTable(ar);
    }
}
```

Once the model and answer tables are created, the client and temporary server tables are generated. The following code snippet accomplishes this task:

```cpp
bool User::buildTables() {
    if(modelTable.size() < 1 || answerTable.size() < 1)
        return false;
    char *q = answerTable[0].question;
    bool correct[MAX_CHOICES];
    clientRow cr[MAX_CHOICES];
    for(int i=0; i < MAX_CHOICES; i++) {
        if(i < (int)answerTable.size())
            correct[i] = answerTable[i].correct;
        else correct[i] = false;
    }
    for(int j=0, total=0; j <= (int)answerTable.size();
        j++, total++) {
        if(strcmp(q, answerTable[j].question) ||
            j == (int)answerTable.size()) {
            // shuffle the non-correct entries into table slots
            bool used[total];
            for(int i=0; i<total; i++) used[i] = false;
```
for(int i=0; i<total; i++) {
    if(!correct[i]) {
        while(!correct[i]) {
            int s = rand()%total;
            if(used[s]) continue;
            used[s] = true;
            cr[i].tuple.index = s;
            if(!correct[cr[i].tuple.index]) break;
        }
    } else cr[i].tuple.index = i;
generateHash(&cr[i]);
    clientTable.push_back(cr[i]);
}

serverRow sr;
memset(sr.question, 0, 1024);
sprintf(sr.question, "%s", q);
for(int i=0; i<MAX_CHOICES; i++)
    memset(sr.modelHash[i], 0, HASH_SIZE);
for(int i=0; i<total; i++) {
    sprintf(sr.modelHash[cr[i].tuple.index], "%s", cr[i].hash);
}
serverTable.push_back(sr);
q = answerTable[j].question;
total = 0;
}
correct[total] = answerTable[j].correct;
memset(cr[total].tuple.hash, 0, HASH_SIZE);
memset(cr[total].tuple.nonce, 0, HASH_SIZE);
sprintf(cr[total].tuple.hash, "%s", answerTable[j].hash);
sprintf(cr[total].tuple.nonce, "%d", rand()%10000);
    cr[total].tuple.index = -1;
}

The server table is then sent to the server, hashed, encrypted, and returned. The encrypted table is then stored with the client.
When the client receives a challenge from the server during verification, the client must respond. As a part of that response, the client must match against the models within the challenge. The user is first given the appropriate prompt; then, based on their response, a model is generated and compared against the challenge models (this snippet being based strictly on a biometric model comparison):

```java
float scores[challenge.getNum()];
float least = 100.00;
for(int i=0;i<challenge.getNum();i++) {
    scores[i] = challenge.getModel(i)->compare(m);
    if(fabs(scores[i]) < least) least = scores[i];
}
int correct = -1;
for(int i=0;i<challenge.getNum();i++) {
    if(scores[i]==least) correct=i;
}
```

Then client table is then used to generate a guess of the shift applied to the challenge:

```java
char hash[HASH_SIZE];
memset(hash,0,HASH_SIZE);
challenge.getModel(correct)->computeHash("SHA512",hash);
int should = -1;
for(int i=0;i<(int)clientTable.size();i++) {
    if(!strcmp(hash,clientTable[i].tuple.hash)) {
        should = clientTable[i].tuple.index;
    }
}
int shift = abs(should-correct);
```

From there, the information is used to generate the response to the server. The response can take on any number of forms. The following is a snippet that follows the methodology used in [Johnson and Boult, 2013] in which the response is comprised of the guessed correct order of the challenge hashes, the shift applied, and the challenge nonce:
char resp[2048];
for(int i=0;i<challenge.getNum();i++) {
    strcat(resp,guessedOrder[i]);
    strcat(resp,"	");
}
char t[2];
t[0] = ’0’+shift;
t[1] = ‘\0’;
strcat(resp,t);
strcat(resp,"	");
strcat(resp,challengeNonce);

B.2 Server

As mentioned, the server for the implementation is a simple standalone C++ program. It has all the basic features needed to simulate a real server for the Vaulted Voice Verification protocol. Of that code, the interesting code snippet deals with the generation of the challenges.

for(int i=0;i<ClassConfig.maxQuestions && i < totalChallenges; i++) {
    int n;
    for(n=0;n<MAX_CHOICES;n++)
        if(strlen(st.sr[i].modelHash[n]) < 1)
            break;
    challenges[i] = ’0’+ rand()%n;
}
//6) Apply the transform to the server table
for(int i=0;i<totalChallenges;i++) {
    int cs= 0, shifts = challenges[i]’0’;
    for(;strlen(st.sr[i].modelHash[cs]) > 0;cs++){
        while(shifts--) shiftRight(st.sr[i].modelHash,cs);
    }
}
//7) Hash values with challenge nonce
char myTable[BUF];
memset(myTable,0,BUF);
sprintf(myTable,"TempChallengeTable-%d",rand()%1000);
hashWithNonce(&st);
saveServerTable(myTable,st,totalChallenges);
hashTable(myTable,hTable);
//8) encrypt table
char encTable[BUF];
encryptTable(st,encTable);
//9) send challenge table to client
sendFile(sock,encTable);