A Sound-based Authentication Protocol

Shi Tai Liu
刘世泰

School of Computer Science
McGill University
Montreal, Canada

May 2013

A thesis submitted to McGill University in partial fulfilment of the requirements for the degree of Master of Science.

Dedication

À ma femme, Zhao
À mes parents

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Muthucumaru Maheswaran, for his indispensable guidance and invaluable advice throughout my graduate studies at McGill University. He is an intelligent and conscientious supervisor with long-term visions.

I am very grateful to Prof. Luc Devroye for his guidance, encouragements and advices during my graduate life. I feel honoured to have taken his two great courses.

I sincerely thank my fellow colleagues at Advanced Networking Research Lab for their support and friendship. Special thanks to Jonathan Tremblay for checking the French version abstract. I would also like to thank all my friends I met at McGill in these two years.

Finally, I am greatly grateful to my dear wife Zhao Dong, my parents and my parents-in-law for everything they did for me, especially to my wife, without whom this thesis could not have been completed.

Abstract

The enormous popularity of the Internet-based services is forcing users to create large number of online accounts to use variety of different services such as online social networking, online banking, online gaming, and blogging. The first step in securing an online service is the establishment of a secure channel between the server and the user. To achieve this, we need an authentication protocol that would verify some user supplied credential and validate the identity of the user. Traditionally, three different schemes "what you know," "what you have," and "what you are" are used individually or in combinations to authenticate a user. The most widely applied scheme among them is the "what you know" scheme.

In this thesis, I propose a strong authentication protocol by using the sound channel between a mobile phone and a desktop machine. In this protocol, the mobile phone is used as a hardware token that responds to a challenge issued by the desktop machine. The mobile phone needs to have the correct secret key to respond to the challenge in the expected way. We can use the sound-based authentication provided by the mobile phone as a standalone user authentication mechanism or as a second factor to increase the security of a password-based authentication scheme.

One of the features of our sound-based authentication scheme is that it does not require the mobile phone to have a data connection. The only requirement is the shared secret with the authenticating server that needs to be established prior to running the authentication protocol. Another feature is the one-time-password-like structure of the scheme, which makes it harder for an attacker to subvert the authentication scheme.

Résumé

La grande popularité des services basés sur Internet oblige les utilisateurs à créer de nombreux comptes en ligne pour profiter de différents services, tels que réseaux sociaux en ligne, banques en ligne, jeux de vidéo en ligne et blogs. La première étape a assurer pour un service en ligne est la mise en place d'un canal sécurisé entre le serveur et l'usager. Pour atteindre cet objectif, nous avons besoin d'un protocole d'authentification qui vérifie certaines informations d'identifications fournis par l'usager, en plus de valider son identité. Généralement, nous utilisons trois systèmes différents : « ce que vous savez », « ce que vous avez », ainsi que « ce que vous êtes », individuellement ou en combinaison afin d'authentifier un usager. Le système le plus largement répandu d'entre eux est celui du « ce que vous savez ».

Au sein de ce mémoire, je propose un robuste protocole d'authentification en employant le canal sonore entre un téléphone portable et un ordinateur. Dans ce protocole, le téléphone mobile est utilisé en tant que jeton matériel qui répond à un défi lancé par l'ordinateur. Le téléphone doit avoir la bonne clé secrète appropriée pour réussir le défi. Nous pouvons nous servir de ce type d'authentification comme mécanisme unique d'authentification d'usager ou encore en tant que mécanisme additionnel de sécurité au système d'authentification classique de mot de passe.

L'une des caractéristiques de notre système d'authentification basé sur le son réside sur le fait qu'il ne demande pas au téléphone mobile d'avoir une connexion à une base de données. La seule exigence est le secret partagé avec le serveur d'authentification qui doit être mis en place avant l'exécution du protocole d'authentification. Une autre caractéristique du système est sa structure ressemblant au mot de passe unique, ce qui rend plus difficile pour un pirate informatique de contourner le mécanisme d'authentification.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, Internet plays a significant role in our daily life. According to a most recent report on January 16th, 2013 conducted by Pingdom, there are 2.4 billion Internet users worldwide by 2012 which represents 34.4% of the world population and a 566.4% growth compared to 2000. [1] Along with the exploring of the Internet, an enormous number online accounts are created, such as online social networks, online shopping, online games, online banking, etc. Traditionally, to authenticate oneself, a user needs to have a user name and password pair, this framework is known as password-based authentication. For higher security requirement, an electronic token might be used in addition to the password, known as two-factor authentication(Google refer it as two-step authentication). Furthermore, there is a third factor called biometric authentication, which uses fingerprint scan, iris scan or voice recognition.

Among these three factors, password-based authentication is most widely used. To ensure the security, it is necessary to have different passwords complex enough for each account, but this ideal situation rarely happens in real life. A report [2] from a credit checking firm Experian said that in the United Kingdom, the average person had as

many as 26 online accounts, but only five passwords. Apparently, this phenomenon hardly satisfies our security requirement; however it is reasonable to explain since to memorize a good number of complicated passwords is a difficult task for the majority of the Internet users.

1.1 Motivations

Due to the fact that people reuse their passwords in different places, the exposure of one password might lead to potential danger to other accounts. Another vulnerability is that an attacker could manipulate on-line phishing to lure users to register a new account or login to a fake web site, so as to get the user's password and probably to impersonate this user in other online services.

An electronic token is a great complement to the traditional password-based authentication. In this framework, a user is not able to authenticate with the server even he/she enters the right password; he/she still need to successfully type in a one time verification key to finish to authentication. This one time verification key can be obtained by SMS text message, generated by a mobile application or a hard token generator. Moreover, a variation to the two-step authentication(we use two-step here because the challenge questions should be classified to "What you know") uses challenge questions rather than one time verification key, which is obvious less secure since the answer to the security questions might also be familiar to someone else who knows the user.

The biometric authentication is not widely used in general purpose due to the installation of complicated hardware and software. By contrary, it is extensively efficient in military and national defence related fields. Since the subject of this thesis

focuses on the online authentication applications in domestic domain, this specific method will not be discussed further in the following chapters.

Now that we have found that password-based authentication is less secure in many senses. The two-factor authentication is more secure, at the price of its process more complex, for the user has to type in more characters. In the meantime, since not all the web sites support two-factor authentication, if the user reuses his/her password in different web sites, similar problems risk to happen.

We would like to develop a new authentication protocol that has the advantages of both password-based authentication and two-factor authentication, but none disadvantages of the two. To be precise, we would like to establish an authentication system without memorizing different complex passwords nor typing in plenty of characters, which at the same time, the process becomes strong enough from common threats.

1.2 Thesis Contribution

In this thesis, we propose a new idea of using sound to authenticate. We assume that a user of online services should have a smartphone that could be used as a token. The smartphone has the function of recording a piece of sound signal and analyse the information in it, as well as performing a certain transform to obtain a binary number from the information carried by the sound. On the other hand, the smartphone should also have the ability to transform a binary number to a piece of sound and play it back. Thus it is possible for us to make use of the smartphone to communicate with the server and exchange necessary information, such as shared secret or hash value of it.

By carefully designing the protocol, we manage to achieve a situation where the user is not required to remember any password, as the result of the authentication process taken over by the smartphone, which in the meantime, guarantees the whole process secure enough from being attacked.

We can also use the sound-based authentication as a second factor to increase the security of a password-based authentication scheme, like the traditional token in two-factor authentication.

1.3 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 introduces the technical background of authentication application and development such as current authentication frameworks, problems in those frameworks, sound recognition technique and Fourier transforms. Chapter 3 discusses the evolution of the sound-based authentication protocol and Chapter 4 focuses on the concrete implementation of the protocol and the experimental results. Chapter 5 presents the related works to sound-based protocol. Finally, Chapter 6 concludes the main idea of the thesis and discusses the future work.

Chapter 2

Background Information

2.1 Current Authentication Framework

2.1.1 Password-based authentication

Password-based authentication is a currently well-accepted authentication framework, whose key concept is to authenticate users by what they know, which typically means the passwords: the shared secret between the user and the server. Figure 2.1 shows

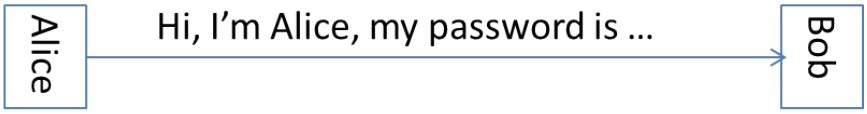


Figure 2.1 Simple Password-based Authentication

one of the simplest cases of password-based authentication protocol, where user Alice authenticates herself with server Bob by sending him the shared secret between the two. It is not difficult to discover that, the most serious problem with such an authentication protocol is eavesdropping, especially under present network circum-

stances. An intruder Eve can stay between Alice and Bob and capture the messages sent between them, since the messages are in plain text, Eve can extract Alice's password and use it to impersonate Alice. One supplement to this system is to construct a secure channel by using Transport Layer Security (TLS) or its predecessor, Secure Sockets Layer (SSL) on the Internet. When an SSL connection is established, a session key is used to encrypt the messages between client and server. Whenever Eve captures the encrypted password, she can never use it to impersonate Alice, since she does not have the session key. For web application, Hypertext Transfer Protocol Secure (HTTPS) is widely used to add the security capabilities of SSL/TLS to standard Hypertext Transfer Protocol (HTTP) communications.

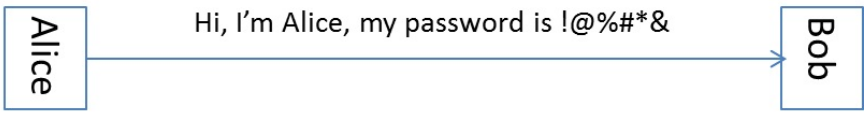


Figure 2.2 Password-based Authentication over secure channel

Now it seems that the scheme is pretty safe by double insurances, however, there still exists several vulnerabilities:

- **Malicious software:** Malware such as key logger is a common threat from unknown workstations. The key logger can be installed on workstations that the user never used before, when he/she logins using the compromised workstation, the key logger logs the user's keyboard strokes to steal password. It is also possible for the attacker to lure the user to install the malware by camouflage it in a certain way.

- **Online phishing** is another threat that widely exists on the Internet today.

The attacker uses some tactics to lure the users to login to a fake web page, which looks very similar to the original web page where the user usually logins. Although most modern browsers are able to prompt warning messages of untrusted CAs or potential threatens in the web sites, the content of such fake web page can be well chosen by the attackers that it attracts the non-professional user to click anyway,since a non-professional user lacks specific knowledge to recognize this kind of tricky attacks.

- **Reuse password and simple password:** Nowadays, people have numerous different accounts to facilitate on-line activities, and it becomes harder and harder for people to remember all of them along with their accumulated accounts. Although it is highly unrecommended, people tend to use the same password for different web sites. Moreover, in order to remember the password, people also tend to use simple passwords instead of complex ones in combination with special characters. As a result, if one of the user's passwords is exposed, all his/her other accounts for this user are in great danger; if the password is not complex enough or even can be found in a known password dictionary, it is possible for the attacker to do an off-line exhaustive key search. Even the server database is encrypted, it still possible to do it by transforming the password to encrypted key and comparing with the database.

To solve the problems mentioned above, different supplementaries are introduced.

- **Coarse IP check:** In this approach, nothing needs to be done on the user's side; on the other side, the server keeps tracking the IP addresses where the user logins. By tracking the user's IP addresses, the server can perform a transformation from IP addresses to coarse geometric location. When the user tries

to login from a new geographic location, which is physically far from the one he/she frequently logins to, the server can issue challenges based on certain informations, i.e., security questions, non-sensitive account related informations. For instance, Facebook uses this approach to protect user account when a user tries to login from a new country.

- **Two-factor authentication:** In this approach, when the user successfully enters his/her password, he/she is not immediately authenticated with the server, because the server waits an additional one time verification key to fulfil the whole authentication, this is known as the second factor, which is actually "what you have". The second verification key can be obtained either by a physical token or a mobile application, which generates one time keys periodically, or by an SMS message sent to the user's cell phone. Obtaining only one of the password and one time verification key is not sufficient to impersonate a user, the intruder has to acquire them both to perform an attack. This scheme enhances the security but makes the authentication process more complex which takes more time. A compromised approach of passing the second verification for a certain period of time after the user successfully logins to a server using a trusted workstation is usually applied in order to reduce repetitive authentication procedure.

- **Challenge question:** This approach is similar to the previous one, but the challenge question itself is rather "what you know", so it is not classified to two-factor authentication. As only the password does not suffice for the authentication, the user needs to provide answers to challenge questions as well as his/her password. The challenge question does not show up every time the user tries to login; instead, when the user logins from an unknown workstation

or browser, the challenge questions are issued. This approach allows the user to decide his/her trusted workstations/browsers, bank account login, for example, is a typical real-life scenario. The biggest problem for this approach is social engineering, because the challenge questions are based on the user's experience or background, and thus easy to be remembered by the user. Therefore, the answers could be guessed by a close friend, or even could be exposed by on-line phishing as well as phone phishing.

- **Positive defence:** Despite the efforts on the server side, the user can also protect his/her own security. In [3], Gross *et al.* classify user accounts to throw-away accounts, routine accounts, spokesperson accounts, sensitive accounts and very high-value transaction accounts. If the user can also roughly classify his/her accounts to those categorises, he/she can choose different user names or more complex passwords for sensitive accounts. Thus, even the accounts in lower sensitivities are stolen, it will not affect the security of sensitive accounts.

More or less, the approaches discussed above enhance the security for password-based authentication, but at the same time, the entire authentication process risks to be more complex, which can be annoying to users.

2.1.2 Address-based authentication

The basic idea of address-based authentication is "where you are", rather than "what you know". Unlike the password-based authentication, in address-based authentication, the identity of the user is inferred in the basis of the user's network address. This method was adopted by UNIX early during the revolution of computer network. It is clearly not a satisfactory candidate for authentication protocols, since most peo-

ple have multiple ends, including PCs, laptops, tablets, smart phones, etc., and thus many people have different IP addresses associated, because these devices are also not used in fixed locations. Although this framework is out of date today, its idea appears in the enhancement of password-based authentication, as discussed above.

2.1.3 Cryptographic authentication

The cryptographic authentication is more secure than password-based authentication and address-based authentication. In cryptographic authentication, Bob issues a random value r to challenge Alice, to prove her identity, Alice performs a cryptographic operation on the quantity Bob issued, as shown in Figure 2.3. The cryptographic operation is based on Alice's cryptographic key. The cryptographic key can be a shared secret known by both Alice and Bob, or a public/private key pairs, where Alice holds her private key, and Bob knows her public key.

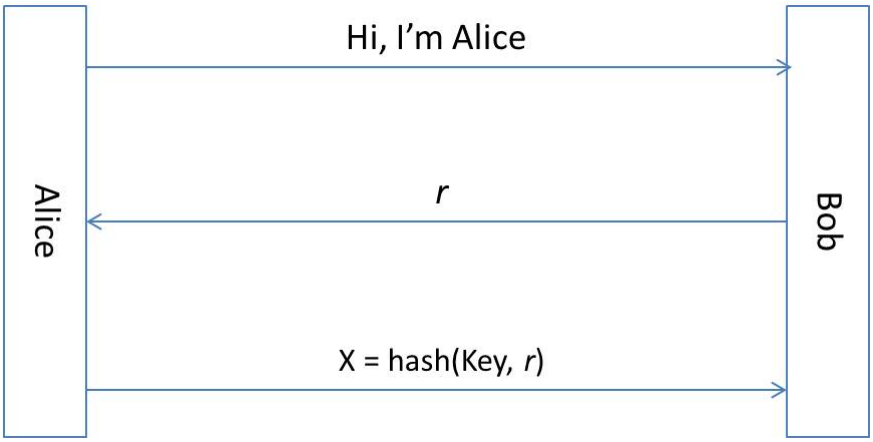


Figure 2.3 Cryptographic Authentication

This idea will be the essential part of the sound-based authentication. Combine with the sophisticate design, it can form a strong authentication scheme.

2.1.4 Biometric authentication

Biometric authentication relies on "what you are", rather than "what you know" or "where you are". The characteristics to describe and identify individuals are called biometric identifier, which should be measurable and distinctive. There are two types of biometric identifier: physiological and behavioural. A physiological biometric can be one's fingerprint, voice, iris, etc.; a behavioural biometric can be typing rhythm and speech. The biggest advantage of biometric authentication is that the user no longer has to memorize anything, because he/she is the identifier him/herself. Moreover, no matter where he/she is located, he/she always "carry" his/her identifiers. It seems that biometric authentication is a perfectly decent solution, but unfortunately it has its own problems:

- **Special hardware and software have to be installed:** To collect biometric information for authentication, it is vitally necessary to install special hardware as well as software to achieve the goal, such as fingerprint scanner or iris scanner. This restrains the login environment of users, who are certainly not able to login everywhere by using biometric authentication equipment.

- **False accept rate or false match rate (FAR or FMR):** The FAR or FMR is the measure of the probability that the biometric system will incorrectly accept an access attempt by an unauthorized user. The false accept rate is extremely severe, compare to false reject rate or false non-match rate (FRR or FNMR), and it is more harmful to have one chance to tolerate false accept.

- **Not cancelable:** Unlike password-based authentication, once a biometric character is issued, it can never be re-issued, which means that the user has to use it

forever. Some biometric characters are not safe enough from being stolen, like fingerprint, for it is possible to make a mould of the fingerprint of the biometric character owner, the person who has this mould can access any system where the owner logins with his/her fingerprint, but the owner can never change his own fingerprint.

- **Privacy issue:** Since the biometric information is very sensitive, the security requirement for the service providers who use the biometric information as authentication factor is very high. The user has to have a highly trusted relationship with the service provider, since the lost of one's biometric data will lead to identity fraud, unintended access or even personal functional information leak.

- **Danger to biometric info owner:** It is possible that the attacker physically attacks the biometric info owner to gain access if the attacker cannot get access to certain secured property. Under this circumstance, the physical damage to the user could be irreversible and could cost more than the secured property. For instance, it was reported that several Malaysian car thieves cut off the finger of a car owner when they trying to steal his car in 2005. [4]

2.1.5 Mobile device based authentication

In B. Soleymani's Master's thesis [5], he proposed a new social factor: "who you know" to be part of the authentication process, which is based on the process of vouching. In this approach the user asks a friend to vouch for him/her, who must recognize the user and then issues some proof of this recognition, which the user then uses to log in to the service. The thesis proposed a way of using cellphones to automate this whole

process.

There was also an project named MICA conducted by Advanced Networking Research Lab at McGill University, which used mobile devices to authenticate users in another way. A basic assumption for this approach is that people have multiple mobile devices nowadays, such as smartphones, tablets, most of which have blue tooth module and can be used to connect to servers. The users first registers his devices with the server, then set a threshold. When the number of devices found nearby are larger than the threshold, the user will be automatically authenticated with the server. The requirement of a set number of devices be present before a user can authenticate prevents an attacker from being able to authentication using only one stolen device.

Both of the mobile device based authentication frameworks discussed above have successfully reduced the information users have to remember, but they are nevertheless both vulnerable to social engineering. The intruder can fraud identity in the first protocol, he/she could steal the user's identity when he/she is physically near the user.

2.2 Problems In Current Authentication Protocols

There are always trade-off in authentication protocols: when the protocol is safer from being attacked, the user has to pay more. The cost can vary from memorizing more information to finishing more complex steps before logged in, and even putting him/herself into physical danger.

Our goal is to develop a new authentication protocol, through which less requires to be done by the user; at the same time the protocol itself is safe enough from attacks. The idea is to use sound to transfer information instead of using network.

In this new sound-based authentication protocol, users do not need to memorize any password; meanwhile, it has some interesting properties that can ensure the security of the authentication process. We will discuss this protocol in detail in Chapter 3, now let us review some technological issues in the sound processing.

2.3 Sound Recognition

According to Encyclopaedia Britannica, sound is "a mechanical disturbance from a state of equilibrium that propagates through an elastic material medium." [6] To be precise, sound is a mechanical wave or a sequence of waves. To describe sound waves, we often simplify them to sine waves, which are characterized by the following generic properties: [7]

- Frequency, or its inverse, the period
- Wavelength
- Wavenumber
- Amplitude
- Sound pressure
- Sound intensity
- Speed of sound
- Direction

The two key characters of sound recognition are **Frequency**: "number of waves that pass a fixed point in unit time; also the number of cycles or vibrations undergone

during one unit of time by a body in periodic motion. A body in periodic motion is said to have undergone one cycle or one vibration after passing through a series of events or positions and returning to its original state." [8] and **Amplitude**: "the maximum displacement or distance moved by a point on a vibrating body or wave measured from its equilibrium position. It is equal to one-half the length of the vibration path. The amplitude of a pendulum is thus one-half the distance that the bob traverses in moving from one side to the other. Waves are generated by vibrating sources, their amplitude being proportional to the amplitude of the source." [9]

In the real life, a sound consists of numerous different frequencies with different amplitudes. To analyse a sound and extract its feature or fingerprint, we need to capture a certain number of its frequencies and amplitudes.

2.4 Fourier Transform

The Fourier transform, deriving from the study of Fourier series, named after the famous French mathematician and physicist Jean Baptiste Joseph Fourier, is an applied mathematical transform in engineering and physics. In our case, it can transform sound from time domain to frequency domain, or vice versa. Time domain is the analysis of sound signals with respect to time in which, the signal's value are real numbers, that is to say, either continuous time or discrete time. On the other hand, the frequency domain is the domain for sound signal analysis with respect to frequency, not time. To be more precise, a time domain graph shows how sound signals change from time to time, whereas a frequency domain graph shows how much of the sound signals lies within every single frequency which is located within a certain range of

frequencies.

$$f(x) = \int_{-\infty}^{+\infty} g(y)\mathrm{e}^{2\pi ixy}\mathrm{d}y \quad \forall x \in R \tag{2.1}$$

$$g(y) = \int_{-\infty}^{+\infty} f(x)\mathrm{e}^{-2\pi ixy}\mathrm{d}x \quad \forall y \in R \tag{2.2}$$

The variable x represents time, where the variable y represents frequency.

The Equations 2.1 and 2.2 form a Fourier transform pair. [10] Here we are using a modern way instead of its original form introduced by Fourier. [11]

The Equations 2.1 and 2.2 cannot be used directly to the sound-based protocol, since the digital sound signals are not continuous. Actually computers can only work with information that is discrete and finite in length. The discrete Fourier transform is the way to solve this problem: it can convert a finite number of equally-spaced samples of a function from time domain to frequency domain.

An N-periodic sequence of complex numbers can be obtained from the N digital signal samples according to the DFT formula:

$$X_k = \sum_{n=0}^{N-1} x_n * \mathrm{e}^{-i2\pi kn/N} \tag{2.3}$$

The formula for the inverse DFT is:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k * \mathrm{e}^{i2\pi kn/N} \tag{2.4}$$

To compute the DFT of N sample points by using the definition will take $O(N^2)$ arithmetical operations, which is so ineffective for real-time applications such as the sound-based authentication protocol, that we need a more efficient way to calculate the Fourier transform.

A fast Fourier transform (FFT) is an algorithm to compute the DFT and its inverse in a faster way than computing DFT by definition, and it FFT can produce the exact same output as computing DFT by definition. All known FFT algorithms can have a complexity of $\Theta(N \log(N))$ instead of $O(N^2)$, which reduces the time of computation. The famous FFT algorithm includes: Cooley–Tukey algorithm, Prime-factor FFT algorithm, Bruun's FFT algorithm, Rader's FFT algorithm, and Bluestein's FFT algorithm. [12] We are not going to discuss the variety of FFT algorithms, but to emphasize that the FFT algorithm is much faster than DFT. In practise, to improve the performance of computing FFT, it is possible to compute FFT in a multi-thread fashion, which is a way can decrease the running time for FFT dramatically.

Chapter 3

The Sound-based Authentication Protocol

In this chapter, I discuss the evolution of the sound-based authentication protocol. I begin by illustrating a simple version of the protocol then discuss the issues arising from using such a protocol; after that, a newer version of the protocol is proposed.

Along with the evolution of the sound-based authentication protocol, we solve its typical threats, as well as some specific threats. We achieve that even the secret information are being exchanged through open channel, such as sound, the protocol maintains secure.

3.1 Sound Signal Selection

3.1.1 Use music as inter-medium

In order to have a sound-based authentication system, we need to have a way to represent a password (and user name) by a sound clip. One approach is to select

the sound clip from a huge music library, or portions of a song, considering the time constraints. To test this idea, we set up the following experiment:

(i) **A testing music library:** Several free wav format classic songs from the Internet.

(ii) **A sound playing program:** Java programs to play a certain song, or a particular part of a song in the library.

(iii) **A sound feature extraction program:** Java programs to extract a sound signature from listening to a certain piece of music.

To examine the validity of this system, we need to consider two restrictions:

(i) **Time to finish the protocol:** For authentication purposes, we suppose that users cannot accept time longer than a certain threshold, which we arbitrarily set as the time required to finish a traditional password-based authentication. We set this threshold as two to five seconds. In order to complete the protocol within this time constraint, we can only play a limited piece of a selected song, instead of playing the whole song, whose duration is always longer than our threshold.

(ii) **Error rate:** For user's convenience, a relatively high error rate is not acceptable. We assume that if the user is holding the correct shared secret(typically a password) with the server, the server should be able to authenticate the user without retrying the protocol too many times.

We illustrate the authentication process by example. Let us say user Alice wants to authenticate with server Bob, instead of typing the password, Alice plays a piece

of sound that lasts for t seconds. To authenticate Alice, Bob first records the sound Alice played, then he performs an FFT on the sound to transform this sound from time domain to frequency domain. Now Bob is able to calculate the magnitude of each frequency. All the frequencies and their magnitudes form the features of a particular piece of sound, but it is infeasible to use all these information to form the feature, since it is sensitive to background noise. Suppose a piece of sound contains a frequency with a very low magnitude, but this frequency is relatively high in the background noise when the sound is being played, when this sound is recorded, the background noise changes the magnitude of this very frequency, then the feature changes undesirably. To extract the feature from a piece of sound, we would like to use the frequencies with the highest magnitudes in the sound. Instead of using all the frequencies, we rather use several key frequencies. Our approach is to divide the frequency band into sub bands, in each sub frequency band, we find the frequency with the highest magnitude, and use this frequency as a key frequency of this sound, thus we have the number of sub frequency bands with many key frequencies, that form the feature of a piece of sound.

From Figure 3.1 we can see that the frequencies of music are almost in the interval of 20 Hz to 2000 Hz. Thus the signature of a sound piece is extracted from 20 Hz to 2000 Hz. Bob divides this interval into n sub frequency bands, and then finds the strongest frequency in each sub band - this set of frequencies forms the feature of the sound played by Alice.

Unfortunately, in a real environment, there exist some background noise frequencies, which are particularly in lower frequency band. This fact can be easily observed by using any frequency spectrum software. To fix this problem, Bob slices the sound

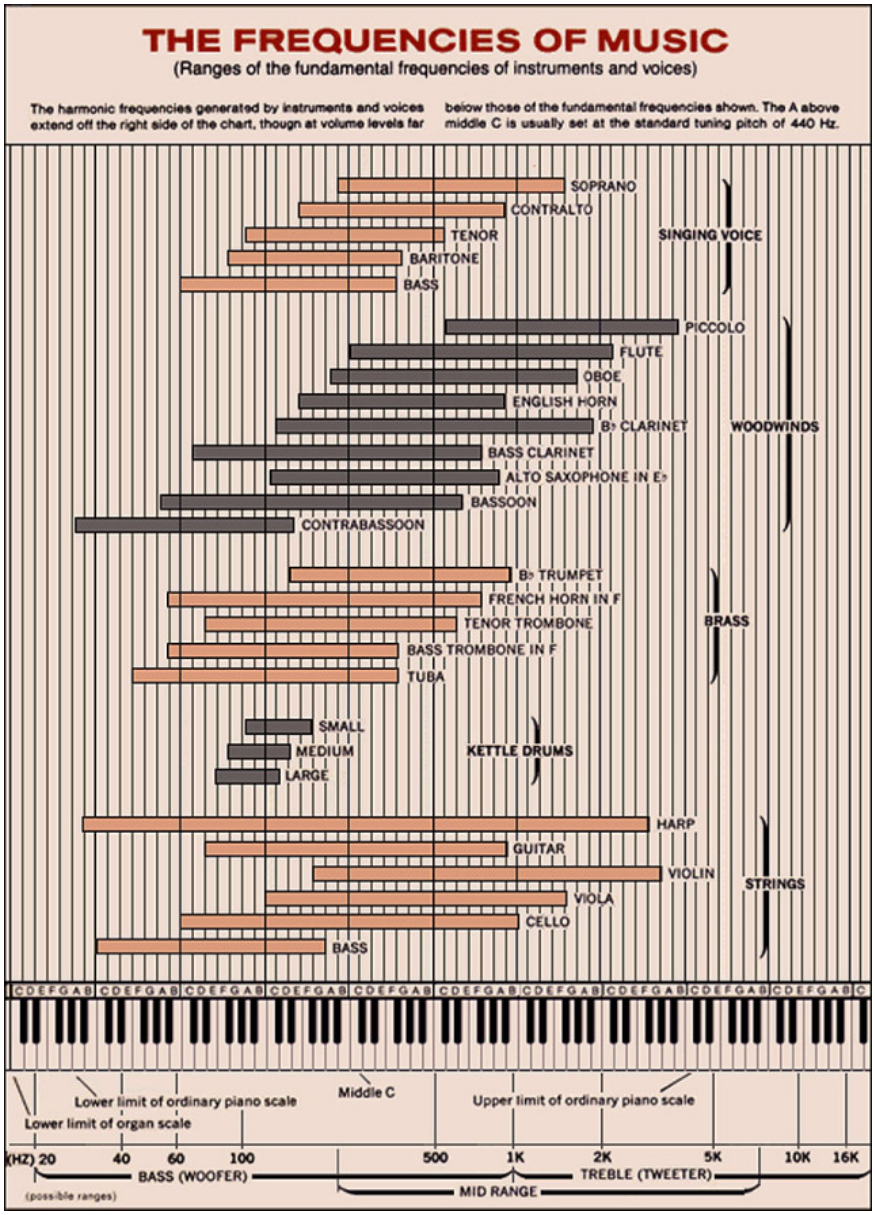


Figure 3.1 Copyright © 1980 by Hachette Filipacchi Magazines, Inc. Reprinted from Stereo Review, April 1980.

Alice played into small chunks of c seconds, and performs FFT on each of the chunks. By doing so, Bob has $\frac{t}{c}$ features rather than one feature. With this set up, it is possible for Bob to set a number of matches as a threshold, and when the matches are more than this threshold, Alice is authenticated, *i.e.*, Alice plays a piece of music for 3 seconds, Bob records it and chops it into 15 chunks of 0.2 second, thus he has 15 signatures, Bob tries to match all of them with his password database, he gets 12 matches, if the threshold is set to 10, then Bob authenticates Alice; if the threshold is set to 14, then the authentication for Alice is failed.

In practice, this variation still cannot fulfil our requirement, since the background noise of lower frequency band affects the original sound a lot. In order to match the sound played by Alice with Bob's password database, Alice has to play the sound at least for 30 seconds. This conclusion is also supported by many famous online music recognition services, such as EchoPrint(*http://echoprint.me/*). We believe that others are using some similar technique to achieve this task.

3.1.2 Use a set of frequencies as inter-medium

To avoid the interference of the background noise, we take two ways into consideration:

- Enhance the characteristic frequency
- Avoid lower frequency band

To enhance the characteristic frequency, it is better to play the sound of some particular frequencies, rather than using the music files. Fortunately, this idea is easy to achieve. In Java, the package *javax.sound.sampled* specifies interfaces for capture, mix, and playback of digital (sampled) audio. On Android, AudioTrack class manages

and plays a single audio resource for Java applications. They allow streaming PCM audio buffer to the audio hardware for playback.

Thus, it is possible to generate any particular frequency we want. To push the audio buffer to the SourceDataLine(Java) or AudioTrack(Android) object, a write method is needed. The audio buffer is represented by an array of sampled points of sound waves. As we know, the sound wave is a sine wave $y = A\sin(\omega t + \varphi), \varphi = 0$ *here*. Thus the audio buffer can be obtained by Equation 3.1.

$$Buffer_i = A\sin(2\pi\frac{i * frequency}{SAMPLERATE}) \tag{3.1}$$

To achieve generating multiple sound waves at one time, Equation 3.2 is needed, where $|F|$ is the number of frequencies to be generated.

$$Buffer_i = \frac{A}{|F|}\sum_{f\in F} A\sin(2\pi\frac{i * f}{SAMPLERATE}) \tag{3.2}$$

Now we can generate any combination of frequencies and play it, thus it is possible to encode the information we want to exchange between Alice and Bob.

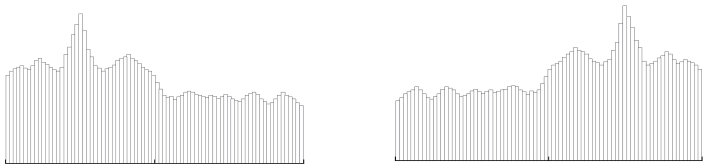
3.2 Encoding Method

In this section, we discuss how to use frequency tones to represent information to be exchanged between client and server.

It is not difficult to come out with the idea of using a single frequency to represent a binary bit, *i.e.*, when a certain frequency is stronger than some threshold, then the bit represented by this frequency is set to 1, otherwise it is set to 0.

In practice, we found that this approach has a problem: the magnitudes of the

sound vary from lower frequencies to higher frequencies, which means it is impossible to set one threshold for all frequencies. Therefore we developed another approach to solve this problem, we divide the frequency band into n sub frequency bands, where n is the number of frequencies to be used. In each sub frequency band, we divide it into two parts, the first part representing 0, and the second part representing 1. In this approach, when Alice wants to generate a zero, she can generate a frequency in the first half of the sub band(she should always choose the middle point of it). Similarly, when she wants to generate a one, she can generate a frequency in the second half of the sub band. When Bob receives the sound played by Alice, he transfers the sound from time domain to frequency domain using FFT, then finds the strongest frequency in each sub band. If the strongest frequency falls into the first half of the sub band, Bob knows that this bit is set to zero. He can easily obtain a one similarly. This approach also tolerates the false caused by the background noise and FFT transformation. Figure 3.2a shows how zero looks like, and Figure 3.2b shows how one looks like.



(a) 0 (b) 1

Figure 3.2 Sample waves

Now the information can be carried by sound. To transfer an n bits number, we only need to choose a frequency band from 0 Hz to 44100 Hz, and chop it into n chunks(sub bands), then assign each chunk to a binary bit. This approach is used in

the sound based authentication protocol.

3.3 Assumptions and Threat Models

In this section, I make the following assumptions and threat models for the sound-based authentication protocol:

3.3.1 Assumptions

1. The user Alice and the server Bob share a secret before authenticating. This secret can be either the password of Alice, or the shared secret key between Alice and Bob, which is stored in both Alice's smartphone and Bob's database. This step is done before the first authentication attempt. The public key and digital signature technologies ensure the security of this key exchange.

2. The smartphone of the user does not need network connection during the authentication process.

3. The secret between Alice and Bob is established or modified through a secure network channel. This can be done automatically when the smartphone is connected to the Internet.

4. The workstation that the user Alice uses to login is trusted. That is, it is not controlled by an adversary.

5. The user Alice initiates the authentication by typing in her user name, the following process is done automatically between Alice's smartphone and the server Bob.

6. The password or secret key is secure stored in smartphone and server.

7. The smartphone is possessed by Alice securely.

3.3.2 Threat Models

1. Eavesdropping on the sound channel.

2. Eavesdropping on the network channel.

3. Attack without proper smartphone.

4. Fake server. Typically this attack can be prevented by certification authorities, but the non-professional users might still be lured.

In the following sections, we discuss the different versions of protocols. We assume the information exchanged directly by Alice and Bob are carried by sound by default, unless explicitly described.

3.4 Protocol Version 0

In traditional password-based authentication, the critical fact is not who you are, but what you know, which is acquired by the password. In the sound-based authentication protocol, this password is transferred by sound. So the first version of this protocol can be done as shown in Figure 3.3. When Alice wants to authenticate with Bob, she tells Bob her user name, as well as her password by playing a piece of sound.

Like the password-based authentication, this protocol cannot prevent eavesdropping especially while we are using sound channel. In network channel, when an intruder Eve wants to eavesdrop to the communication between Alice and Bob, she

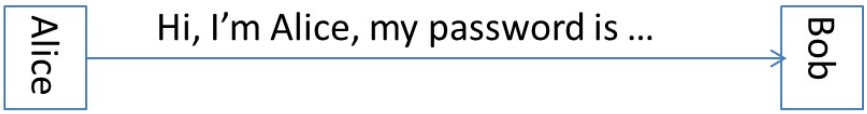


Figure 3.3 Protocol version 0

need to be in the right place in the network. But for sound channel, if she is physically presenting in the place Alice is authenticating, she can easily listen to this channel, then record it and save it to impersonate Alice in the future. On the other hand, it is more difficult for Eve to eavesdrop from a remote place, unless she can control a microphone remotely. This security issue directly leads to the protocol version 1.

3.5 Protocol Version 1

Intuitively we would like to send Alice's password with a secret key instead of sending the password directly. Alice calculates a hash value of her password based on the secret key shared between Alice and Bob, and sends it to Bob. This is shown in Figure 3.4. Depending on how Bob saves Alice's password, he can either compare Alice's hash value with the database or calculate the hash value of Alice's password in the database and compare it with the value sent by Alice. Since the secret key does not change from time to time, it is not difficult to see that the encrypted password is always the same one. Although the password is encrypted, as long as it is carried by sound channel, the vulnerability remains. The eavesdropper does not care whether the password is encrypted or not, she can simply record it and replay it to Bob at any time. It is possible to establish a secure channel before Alice sends her password to Bob, but it is better to use cryptographic authentication itself to fulfil the authentication

requirement.

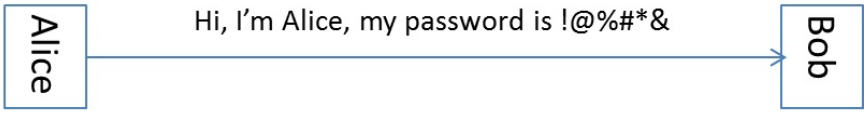


Figure 3.4 Protocol version 0 with secret key

Cryptographic authentication protocols are more secure than only password only protocols. To achieve this goal, Alice does not provide her password directly to Bob, but performs a cryptographic operation based on a quantity Bob supplies. In this scheme, Alice and Bob share a secret key instead of having a password for Alice. When Alice initializes the authentication, Bob sends a random value r to Alice, then Alice uses the secret key to calculate a hash value on r and sends back it to Bob. Since Bob knows this secret key, he can verify the correctness of Alice's message. This process is shown in Figure 3.5.

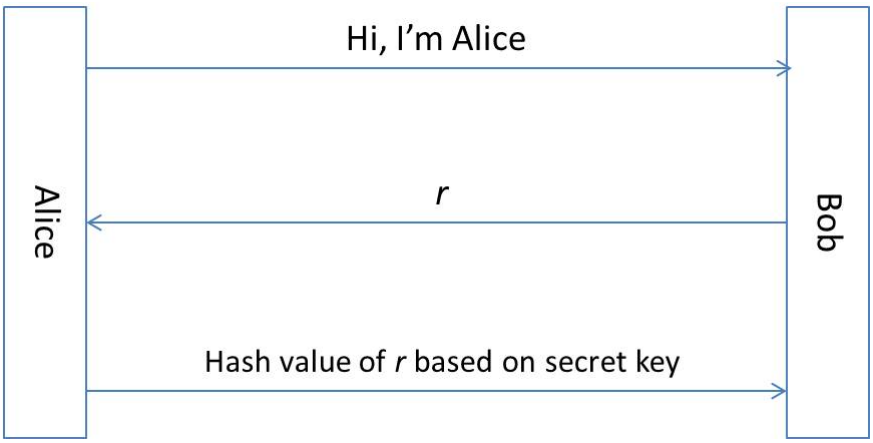


Figure 3.5 Protocol version 1

Version 1 seems to solve the problem of the protocol in Figure 3.4, but it still has

its vulnerability, because it is not mutual authentication. Let us see the scenario in Figure 3.6, suppose the intruder Eve can make Alice believe her address is Bob's

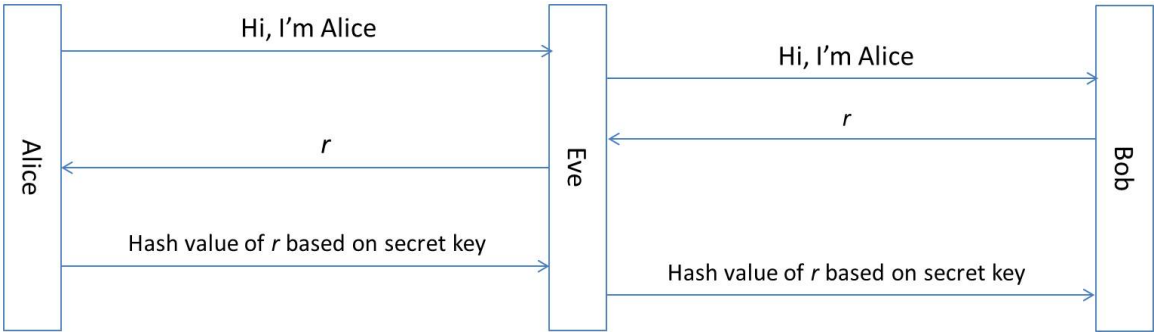


Figure 3.6 Attack against version 1

address, which can be done by presenting a fake web site to Alice, then the initial authentication request from Alice is sent to Eve. After Eve receives this request, she initiates an authentication request with the real Bob. When Bob receives Eve's request, he thinks it comes from Alice and then sends a random number r to Eve. Eve then sends this number to Alice, Alice calculates the hash value based on her secret key and r, then sends to Eve. Now Eve has enough information to establish a session with Bob, so she sends this hash value to Bob, and finally be authenticated as Alice with Bob.

Another significant problem is dictionary attack. We know that Bob sends a random number r whenever Alice initiates the protocol, so there are $|r|$ (denote the number of possible r's) hash values that Alice can send back to Bob. This $|r|$ depends on the number of bits of r. Ideally in a traditional cryptography scenario, we can choose a very long number to be r. But in the sound-based authentication protocol, this r has to be sent by using sound, which limits the length of r. Thus, it is possible for an eavesdropper to record all the possibilities of the r and hash value pairs after

a long time. When this eavesdropper gathers enough information, she is able to play as Alice. This is also known as off-line attack.

In contrast, in an online attack, it is not hard to imagine that the intruder Eve tries every possible response at server Bob to verify the real response. It is not difficult, however, to prevent this from happening since Bob can easily interdict any user from guessing too many times. There is a better way to achieve this goal which will be discussed in the later section.

3.6 Protocol Version 2

Let us fix the attack in Figure 3.6 first, since the dictionary attack problem is a little bit more sophisticated. In the Protocol version 1, suppose the time for Bob to play sound r to Alice is t', and the time for Alice to play the hash value back to Bob is also t'. The whole protocol takes $2t$'. When the intruder Eve wants to perform the attack, she needs to intercept r from Bob as well as the hash value from Alice, which means she needs twice the time of the protocol, which is $4t$'. This observation gives us an important hint: if a time out mechanism is added to the protocol, *i.e.*, when Bob issues the random number r, he starts a timer t equalling to $3t$', which is long enough for Alice to finish the hash value calculation, but not sufficient to Eve to perform the man in the middle attack. If the timer goes to zero and Bob has not receive the hash value, the protocol terminated, thus the attack is eliminated.

Now it is time to consider a trickier problem: dictionary attack. The secret key is shared by only Alice and Bob, as well as the hash function used to calculate the hash value. These two factors are never known by the intruder Eve. On the other hand, if Eve listens to every single authentication of Alice, she can obtain a set S of correct

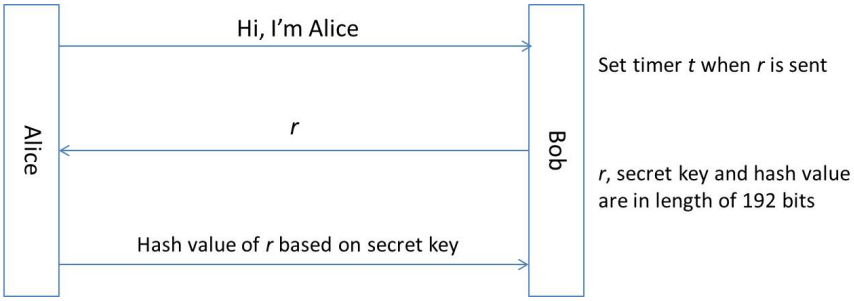


Figure 3.7 Protocol version 2

response along with the random number r. As we discussed in the previous section, Eve can wait for this set to grow to a size large enough, that she has $\frac{|S|}{|r|}$ possibility to have to right answer to the random r. What else can Eve do? She can use the famous hash functions to hash all the possible keys and compare with the set S. If there is a match, she can test the hash function and response she found with the other values in S. Considering the computation power today, especially with GPU computation, the hash function needs to be carefully chosen, so that Eve will not be able to guess it, since it is not very hard to exhaust the key space. Let us quantify it by taking GPU exhaustive key search. Let k denote the bit length of the shared key, as well as the bit length of the hash value, and let t denote the time unit in the GPU, c denote the cores in the GPU, thus the time for exhaustive key search can be calculate as shown in Equation 3.3.

$$\frac{2^k * t}{2 * c} \tag{3.3}$$

In the GPU today, t is in the level of 0.1 second, c is in the level of hundreds. If we want the average time to be 10000 days, we roughly need 64 bits key. If we want the key to be strong enough for life time, a roughly 192 bits key might be needed.

3.7 Protocol Version 3

In Protocol version 2, we need to encode a large number of bit information into the sound to ensure the security. As we know, along with the increase of bits encoded, the interference appears. Also, when the computation power increases, we need to increase the bit length again to maintain the security level. At the same time, the whole frequency band remains unchanged. We need to have a better scheme to reduce the importance of the bit length. To solve the problems mentioned above, a one time password scheme is raised. One-time means that the password can be used only once, afterwards it changes. One of the most famous one-time password schemes is Lamport's hash [13]. In Lamport's hash, the server saves for each user the following information:

- user name
- An integer n, decreased by one when the server successfully authenticate the user each time
- $hash^n(password)$, *i.e.*, hash function applied on password n times

When the user Alice wishes to login to the server Bob, Alice enters her user name and password on her workstation, the workstation sends her user name to Bob. Then Bob sends n back to the workstation, the workstation applies the hash function on Alice's password *n-1* times and sends this quantity to Bob. Bob hashes this received quantity once and compares it with his database, if it matches, the session establishes and Bob replaces the quantity he stored with the newly received quantity and decreases n to *n-1*.

Let us combine the protocol discussed above with Lamport's Hash to form protocol version 3.

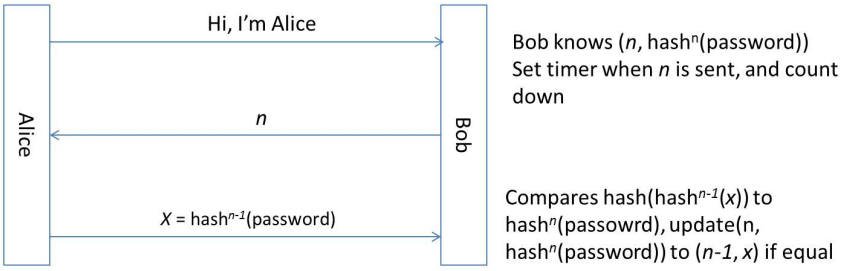


Figure 3.8 Protocol version 3

In version 3, Alice holds her password, Bob holds the following information for each user:

- user name
- An integer n known by Bob, decreased by one when Bob successfully authenticate the user each time
- $hash^n(password)$, *i.e.*, hash function applied on password n times

It is clear that the intruder Eve does not have enough time to perform an attack in Figure 3.6, since Bob sets a timer that shorter than the time of the attack needed. Meanwhile, since the hash value sent by Alice changes every time, Eve is never able to create the dictionary, either. It seems that protocol version 3 has solved all the problems, except online password guessing(Notice by adding a maximum failing counter, online password guessing can be prevented, but since n is fixed whenever the authentication does not succeed, this attack is not naturally immune), however, it is still

vulnerable. One minor problem is that Alice can only log in n times to server Bob, after which they have to reset another integer n.

A more severe problem is small n attack. Suppose an intruder Eve impersonating Bob and waiting for Alice to authenticate. When Alice initiates the protocol, Eve sends back a small n to Alice, then Alice sends the hash value with this small n to Eve. Now Eve can save this n, x pair for further use. Notice that n is always sent by sound channel, and the current n can be easily obtained by Eve, which can help her to choose the small n and impersonate Alice in the near future. Although this can be prevent by saving copies of n's for each server that Alice might authenticate with, we would like to propose a more sophisticated way to solve this problem in version 4.

3.8 Protocol Version 4

We borrow the idea of Lamport's hash and combine it with the cryptographic authentication scheme to form our protocol version 4. The key change of protocol version 4 is, instead of using an integer n, Bob still chooses a random number r, while a specific hash function is carefully selected. Let us add another variable, the current UNIX time t(in minute) in the hash function. When Bob receives Alice's authentication request, he selects a random number r and sends it to Alice, doing three more things at this moment as follows:

- Set a timer and count down, where timer is 1.5 times of standard protocol time.
- Save current time, accurate to minute
- calculate the hash value $X' = \text{hash}(key, r, t)$ locally, save it to compare with the value sent by Alice.

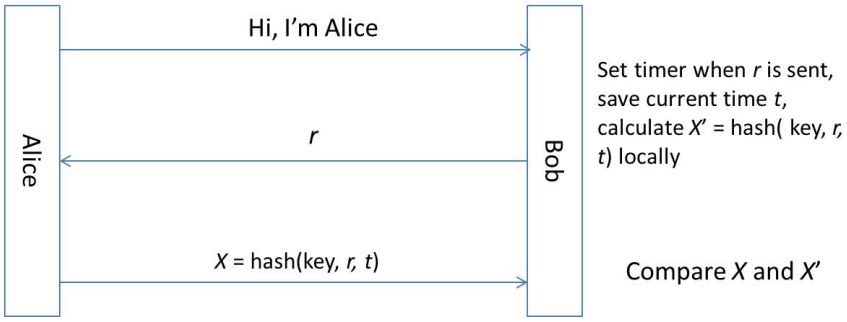


Figure 3.9 Protocol version 4

When Alice receives the random number r, she calculates a hash value X based on the secret key and r, as well as the current time t, then sends X back to Bob.

In this protocol, we assume that the local time of Alice and Bob are synchronized, which is a reasonable assumption. The server Bob is connected to Internet, so his time is the standard time. The user Alice has her cell phone as her token, which has time synchronized with the carrier's server. These two indicators of times should be the same.

The new added time variable t is the essence of protocol version 4. We use a variation of UNIX time to represent t, *i.e.*, only the part until minute is used. This makes the valid hash value of each random number r change every minute, which means the exhaustive key search is impossible in this protocol. Suppose the whole protocol takes 1 second to finish, if the intruder Eve wants to operate an exhaustive key search, the maximum number of times she can try is 60 for a particular random number n, even if the n does not change from time to time. This result is very important to the sound-based authentication protocol, in the sense of the length of key is not as important as in traditional cryptographic authentication. Even if the bit length of key is set to 16 bits in this protocol, any intruder can only have at most $\frac{60}{2^{16}} \approx 0.0009$

chance to guess the correct response, after this very minute, any information from the previous tests will be useless to the intruder. In our experiments, we finally push up the bit length of one piece of sound to 32 bits, and obtain 64 bits length by combining two pieces of 32-bit sound. Thus, the chance to hit the right response by applying brute-force attack is 0.000000014 for 32 bits encoding and 3.25×10^{-18} for 64 bits, which is secure enough.

Protocol version 4 perfectly solves all the issues we discussed above, but there still might have a pitfall regrading the retry mechanism. As we discussed before, the messages between Alice and Bob are carried by sound channel, where the error rate is not guaranteed to be zero, *i.e.*, it is possible that the authentication fails because of information lost in transmission, so a retry mechanism is needed in the protocol. The intruder Eve must be very happy about this mechanism, since it is possible for her to send a random response to Bob to trigger the retry mechanism, which keeps the session between Eve and Bob alive. To be more precise, Eve impersonates Bob to Alice when she initiates the authentication protocol, then Eve sends Alice's initial message to Bob to get the random value r he sends. Then Eve sends a fake hash value to Bob to keep the authentication process between Eve and Bob alive, at the same time, she sends this random r to Alice to get her correct hash value to this random r. Once Eve successfully gets this hash value, she sends it back to Bob to impersonate Alice. This can be illustrated in Figure 3.10.

The attack like in the Figure 3.6 appears again, but we have two solutions to it. The easiest way is to modify the retry mechanism slightly. We would rather make the authentication protocol fail, instead of waiting for the retry. In this mechanism, when Eve returns a fake hash value to Bob, Bob terminates the protocol and and considers

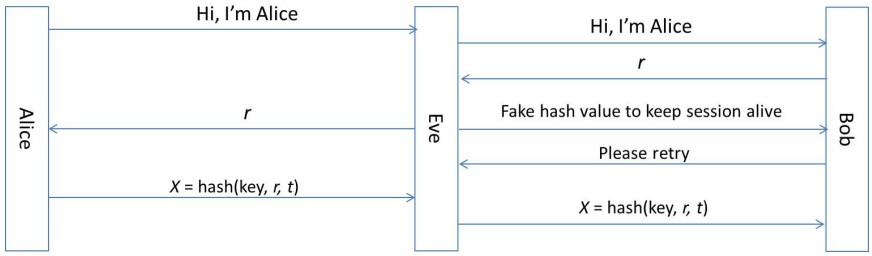


Figure 3.10 Attack against version 4

the response invalid. In this new mechanism, if the sound-based authentication fails accidently due to abrupt noise, the user has to restart the protocol again, which make it inconvenient. We can make the server automatically re-issue a new challenge after the first failure. Also, the client application will listen to it and perform the whole authentication process again.

The other way is to let Alice authenticate Bob, too. The mutual authentication protocol is discussed in Section 3.9.

3.9 Protocol Version 5

With a mutual authentication, Alice can make sure that she is talking with Bob. The easiest way to achieve mutual authentication is to exchange authentication informations in both direction shown in Figure 3.11. The reason why Alice identifies herself first instead of Bob doing so is to prevent reflection attack from happening. This attack is shown in Figure 3.12.

Let us say the intruder Eve wants to impersonate Alice to Bob, and she can initiate the protocol as shown in Figure 3.13. Of course when she receives the random number r_{Bob}, she is not able to calculate the hash value based on it. However, as long as server

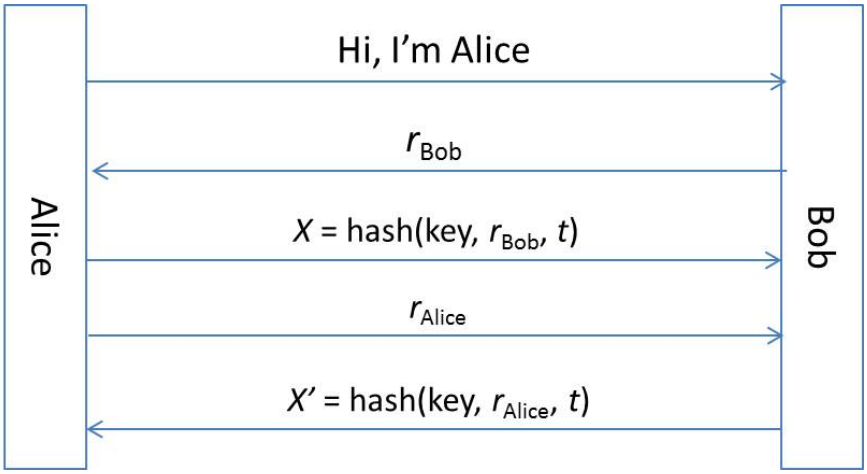


Figure 3.11 Mutual authentication

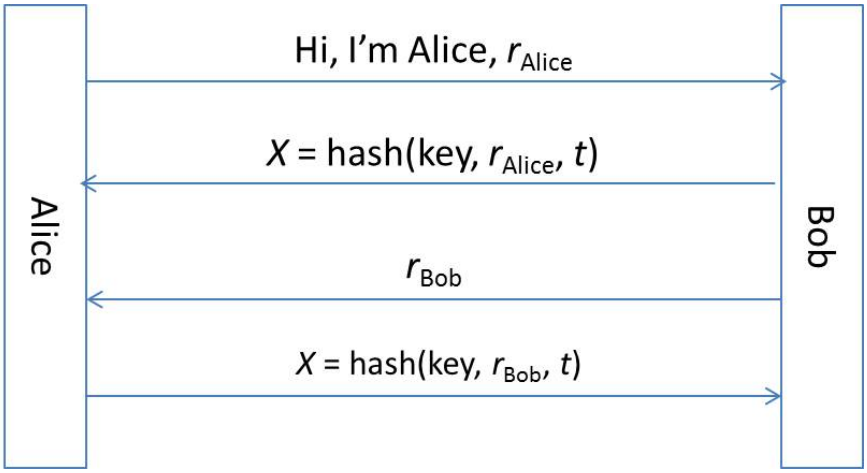


Figure 3.12 Mutual authentication with reflection attack threat

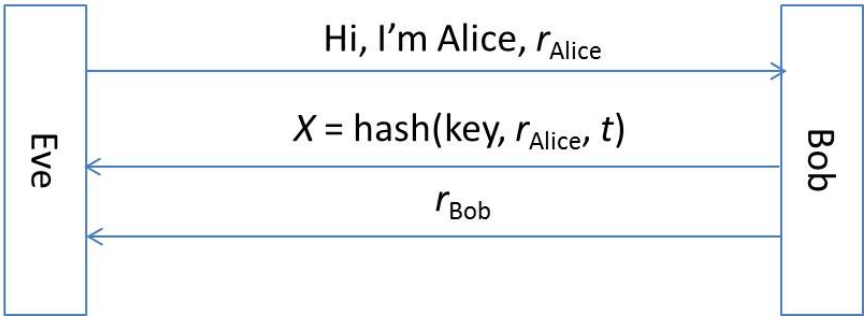


Figure 3.13 Phase one of reflection attack

Bob supports multiple session, Eve can start another session with Bob by using r_{Bob}, then Bob will calculate the hash value for her. When she finally get this correct hash value, Eve is able to continue the first session and impersonate Alice.

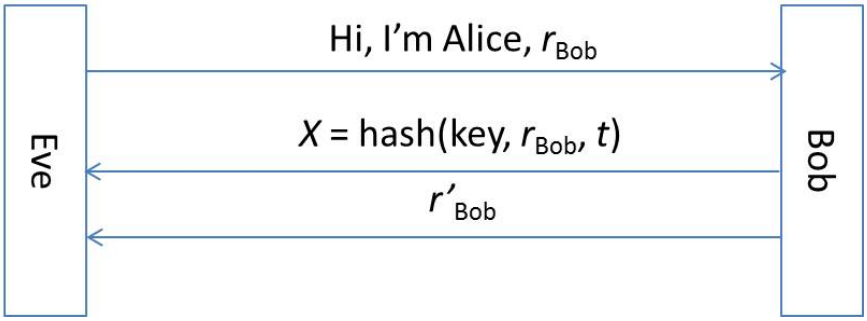


Figure 3.14 Phase two of reflection attack

As we discussed, if the user Alice needs to identify herself first, the intruder Eve can not be able to perform this reflection attack.

Another way to prevent reflection attack is, instead of having only one shared secret - the secret key between Alice and Bob, to add a second shared secret which identifies the server Bob. For each user/server pair, two totally different secret keys are saved at both ends, denoted by $K_{Alice-Bob}$ and $K_{Bob-Alice}$, respectively, as shown

in Figure 3.15.

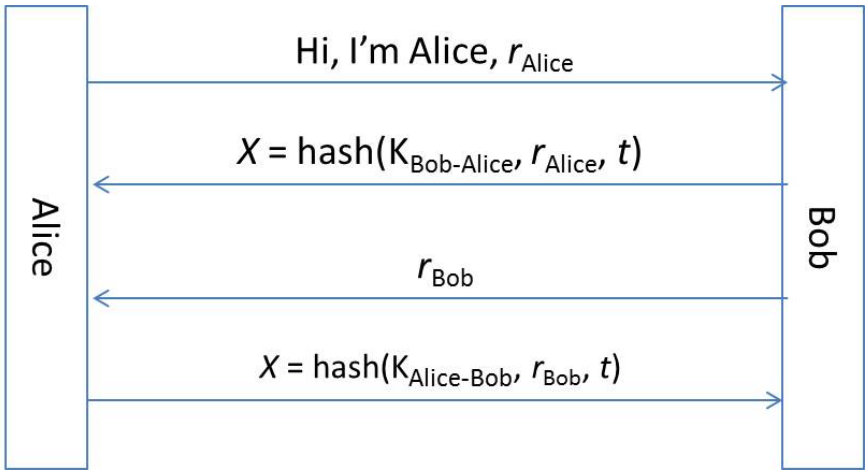


Figure 3.15 Protocol version 5

Now let us compare protocol version 4 with protocol version 5. The advantage of version 4 is, only two pieces of sound are played during the authentication, where version 5 needs two extra pieces of sound. The advantage of version 5 is, the user knows he/she is definitely communicating with the right server. Here we claim that this benefit is actually useless. Let us consider the following scenarios:

- **The intruder impersonate the server:** When Eve impersonate Bob to Alice, and Alice provides her correct hash value to Eve, Alice actually loses nothing. Once Alice logins to Eve, there is no information of Alice in Eve's database, as long as Alice does not provide further sensitive information, Eve gains nothing. The only information Eve can get in this step is a one time hash value to a particular challenge r.

- **The intruder impersonate the user:** Since Eve can only have a hash value to one specific challenge r, it is almost impossible for her to get the same challenge

when she tries to impersonate Alice to Bob, moreover, this correct hash value expires after 1 minute.

We recommend to use protocol version 4 in practise, but if theoretically higher level of security is needed, protocol version 5 is a good candidate.

So far, a strong protocol is developed, but the discussion is limited in a theoretical way. In next section, we will discuss an applied version of the sound-based protocol, which illustrates how to use this protocol in practice.

3.10 Applied Protocol

In this section, we will discuss how to apply this sound-based protocol to a real-life scenario. The first observation is, Alice and Bob can never talk directly during a sound-based authentication process. It is not difficult to understand this issue, since the server Bob sits at another end of Internet when the sound channel is being used during the authentication, there is no way for them to talk via sound. Thus, there must be an agent sitting between Alice and Bob, which could be a workstation or a browser extension for example. Let us first apply protocol version 5. The whole protocol consists of the following steps:

1. Alice tells the workstation that she wants to start the protocol, and sends out a challenge r_{Alice} through her smartphone.

2. The workstation tells Bob that Alice wants to start the authentication protocol along with her challenge r_{Alice}. This step is done through network channel.

3. Bob calculates a hash value based on his secret key, Alice's challenge r_{Alice}, and

the current time t. He also issues his challenge r_{Bob}. These two information are sent to the workstation through network channel.

4. The workstation sends the hash value and Bob's challenge r_{Bob} to Alice through sound channel.

5. Alice calculates a hash value based on her secret key, Bob's challenge r_{Bob}, and the current time t. This hash value is sent to the workstation through sound channel.

6. The workstation sends Alice's hash value to Bob through network channel.

This whole protocol can be shown in Figure 3.16, where the left-hand rounds are

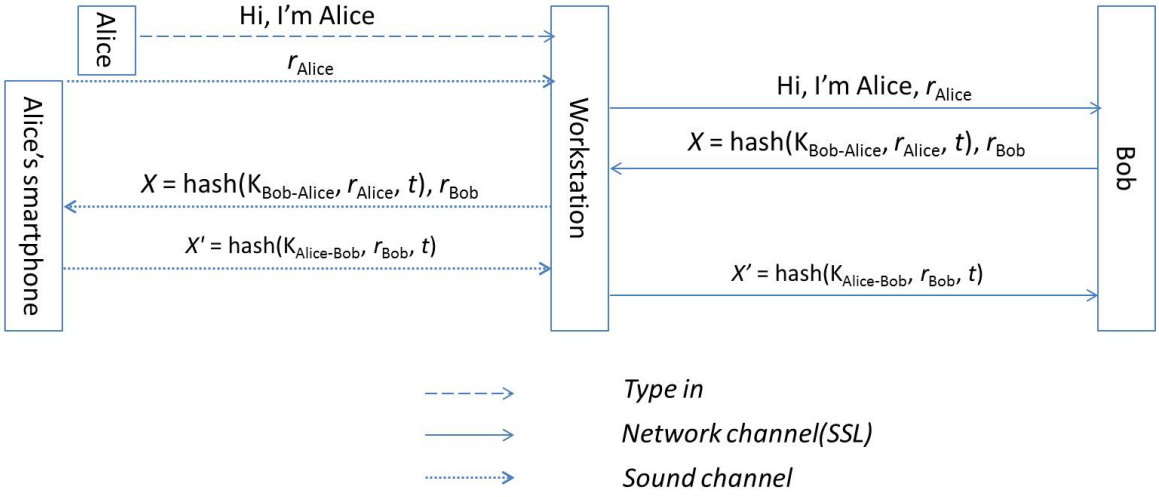


Figure 3.16 Applied protocol with mutual authentication

completed through sound channel, whereas the right-hand rounds are done through network channel. We assume that the right-hand side takes no time to finish.

Since the right-hand side is done on network channel, a potential serious threat is man in the middle attack. If the intruder Eve can intercept the network connection between Workstation and Bob, she is able to establish two separate connections with them and make them believe they are talking to each other. This threat can be prevented by Public Key Infrastructure(PKI), although the modern browsers can verify the CAs, the user can still choose to trust the untrusted certificates. In this case, the mutual authentication does not work as desired.

Let us apply protocol version 4 and solve the man in the middle attack. Similarly, the protocol consists of the following steps, as shown in Figure 3.17:

1. Alice tells the workstation that she wants to start the protocol, along with her user name.

2. The workstation tells Bob that Alice wants to start the authentication protocol along with her user name. This step is done through network channel.

3. Bob issues his challenge r to the workstation through network channel.

4. The workstation sends Bob's challenge r to Alice through sound channel.

5. Alice calculates a hash value based on her password, Bob's challenge r and the current time t. This hash value is sent to the workstation through sound channel.

6. The workstation sends Alice's hash value to Bob through network channel.

Since we assume we cannot rely on the correctness of server's identity, we propose another solution to the man in the middle attack. If we can pre-share a secret between the Workstation and the server Bob, the man in the middle attack is eliminated, but

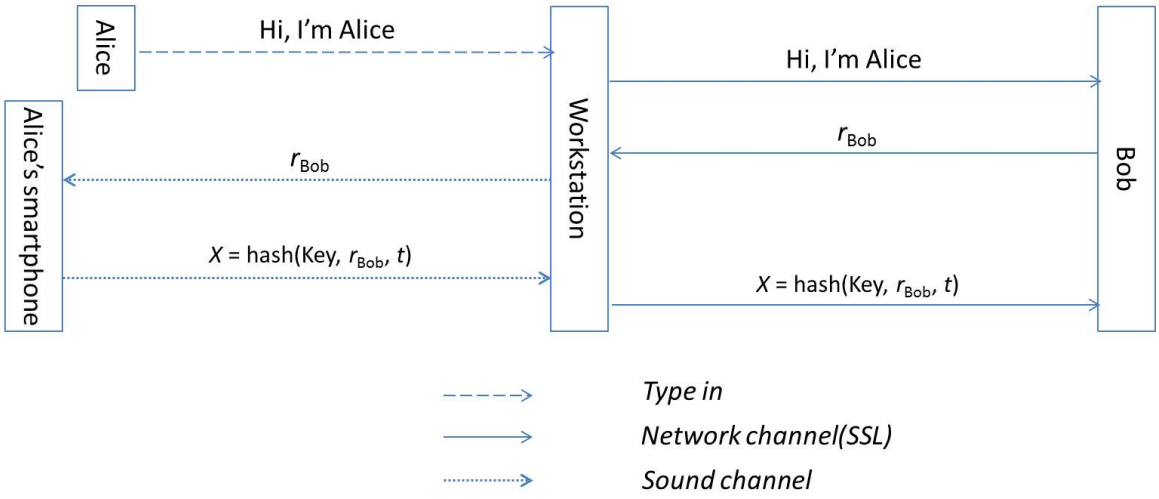


Figure 3.17 Applied protocol without mutual authentication

we cannot guarantee that any workstation has the pre-shared knowledge with the server Alice would like to authenticate. Our choice is to use a second channel to prevent man in the middle attack. The basic idea of this approach is to use Alice's smartphone to calculate a checksum based on her hash value that sent to Bob, and send this checksum to Bob. The authentication can be prove only if Bob receives this checksum from the correct source. Since the intruder Eve cannot hold Alice's smartphone at this time, the protocol is secure enough.

Chapter 4

Implementation

In this chapter we discuss the design of the working prototype of the sound-based authentication protocol that has been implemented using the Java programming language, as well as the experimental results.

4.1 System Architecture

The sound-based authentication protocol is implemented using an Android phone and a laptop. The phone model is Sony Ericsson MK16a with Android version 4.0.4 Ice Cream Sandwich. The laptop is Lenovo ThinkPad T430s. In this set up, the Android phone works as the hardware token and the laptop as the authentication server. The sound piece that contains information to be sent has a length of 0.25 second, during which it ensures the feature in the sound piece can be extracted using FFT correctly; also, the whole sound-based protocol can be finished within one second. To encode N-bit information into a sound piece, a frequency band of length L Hz is needed. Each bit is represented by a sub frequency band of length l Hz, where $l = L/N$. According

to my test, this 0.25-second piece does not sound unpleasant if 16-bit PCM is used, despite the choosing of the frequency band. When the frequency band is near 20 kHz, it is almost impossible to hear the encoded sound. So depending on whether the user wants to be aware of the the authentication during the process, the frequency band can be chosen to near 20 kHz band or not.

To fulfil the sound-based authentication protocol, both token and server should have the ability to record sound, extract sound feature, encode features into sound and play sound. Thus the programs in both end need to have these three modules. In the next two sections, we discuss the programs based on functionality, in regardless of the ends.

4.2 Recording and Extracting Feature

To distinguish the sound sent by the token or server from the environment sound, a flag frequency is encoded to any sound piece sent by either token or server. The flag frequency is one frequency that has a certain distance to the encoding frequency band, and strengthened by the sender. Whenever this flag frequency is captured, it means the sound that is being played is sent either by token or by server. The disappearance of flag frequency indicates the end of an encoded sound.

An independent thread is used to record the sound continuously with a sample rate of 44100 Hz, which means there are 44100 sample points for a one-second sound. These one-second sounds are chopped into 0.25-second chunks in time domain, thus each 0.25-second chunk has 11025 sample points. After applying FFT, the frequency domain is precise to 4 Hz. This operation decreases the duration of a sound unit to 0.25 second instead of 1-second, without losing accuracy. Thanks to the computation

power of smart phones and laptops, with a multi-threaded FFT algorithm, the real-time sound processing is possible.

For each frequency, we define the magnitude of the frequency in equation 4.1 to scale down the absolute value. In the equation, r is the real part of the frequency in frequency domain, where i is the imaginary part. When the magnitude of flag frequency is larger than a certain threshold during the first phase, it indicates that the sound being sent is the encoded sound sent by either the user or the server.

$$Mag_{frequency} = \log(\sqrt{(r^2 + i^2)}) \tag{4.1}$$

After the encoded sound is found, the program performs operations on the array of these sample points. It first locates the frequency band that used for encoding the bit information, after that, it scans from the left border to the right border. For every single frequency in this frequency band, the program first checks the look-up table to find which sub frequency band it falls in, *i.e.*, which bit this frequency represents. The program keeps tracking and saving the frequency with the largest magnitude for each sub frequency band, thus an array of these record points is obtained.

The program takes this array and transfers it to bit information, the lower frequencies are in lower bits. As shown in equation 4.2 (L is the length of the frequency band, N is the bit length), it checks whether the largest magnitude falls in the first half of the sub frequency band or the last half to determine whether it is a 1 or a 0 for this sub frequency band. The program repeats this operation on all the sub frequency band to get the bit information for every single bit and then adds them up

to an integer for convenience.

$$Int = \sum_{i=1}^{L/N} \lfloor \frac{|recordPoint[i] - (Lower\_freq\_boarder + L/4N + i * L/N)|}{L/4N} \rfloor * 2^i \quad (4.2)$$

After decoding the code for the encoded sound, the real information is obtained. Depending on which step it is in the sound-based protocol, the program either compares this integer number with the number it calculated, or performs a certain calculation on it to prepare to send it out after encoding into a piece of sound.

4.3 Encoding Feature and Playing

When the user needs to encode bit information into a piece of sound, the program calculates how to convert the integer number to frequencies by using equation 4.3.

$$recordPoint[i] = (Lower\_freq\_boarder + L/4N + i * L/N + int\_digit[i] * L/2N) \quad (4.3)$$

This equation gives every record point in each sub frequency band. We combine all the waves of these frequencies together by using equation 4.4. An important thing is to scale the result down to byte in order to fulfil the requirement of the programming API.

$$Buffer_i = \frac{A}{|F|} \sum_{f \in F} A \sin(2\pi \frac{i * f}{SAMPLERATE}) \quad (4.4)$$

By pushing the buffer array obtained by equation 4.4, the hardware will know how to play the sound generated. Finally, the flag frequency has also to be added into equation 4.4 to ensure the receiver can pick it up.

4.4 Android Application

The Android application is the token for user to apply the sound-based authentication protocol. When user Alice wants to authenticate with server Bob using our protocol, she types in her user name, then launches the application and hits start. The reason for Alice to manually launch and start the application is power management. If the application runs in background, the continuous recording and FFT transformation will drain the battery power quickly. Figure 4.1 shows the interface of the Android

Figure 4.1 Android application

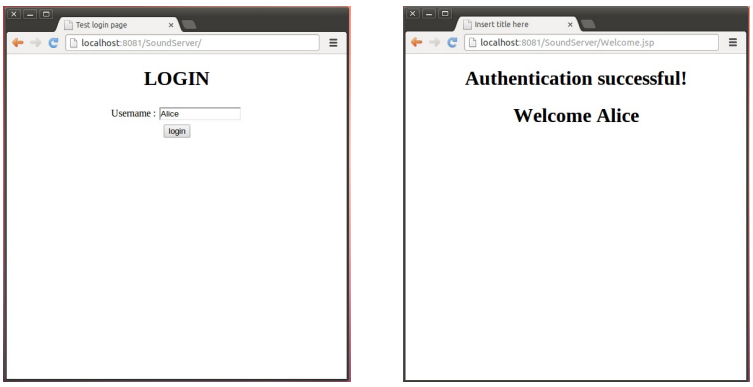
application. When Alice hits the start button, the application starts recording and applying the FFT algorithm to find the flag frequency. After finding the flag frequency, the application extracts the feature in the encoded sound and transforms the feature into an integer number, which is the challenge R issued by Bob. The application calculates a hash value based on this R, as well as Alice's password and current UNIX time. Then this hash value is transformed to an array of sample points, which forms

the encoded sampled sound. The application plays this calculated sound to Bob, and then Alice is authenticated automatically.

The text area with "Idle..." indicates the status of the application from idle, recording, calculating to sent. The area below is used for batch test. We would like to evaluate the accuracy of the sound-based authentication by executing the protocol multiple times automatically. To achieve this, we create a socket connection between the application and the server to control the execution of the protocol. After entering the server's IP address and hitting connect button, the application connects to the server with default port. Then the socket connection is used to control the application and the server to perform n times of the authentication process.

4.5 Sample Web Site Using Sound-based Authentication Protocol

We create a sample web site that supports the sound-based authentication by JavaServer Pages (JSP) and Java Servlet.



(a) login front page (b) welcome page

Figure 4.2 Sample login page

Figure 4.2a shows a login front page, where user Alice enters her user name and clicks login button. Then the server generates a random value R, which is encoded to a piece of sound and sent through sound channel. After that the server waits for the reply from the token. Once the correct response is recorded by the server, the authentication finishes successfully.

4.6 Experimental Result

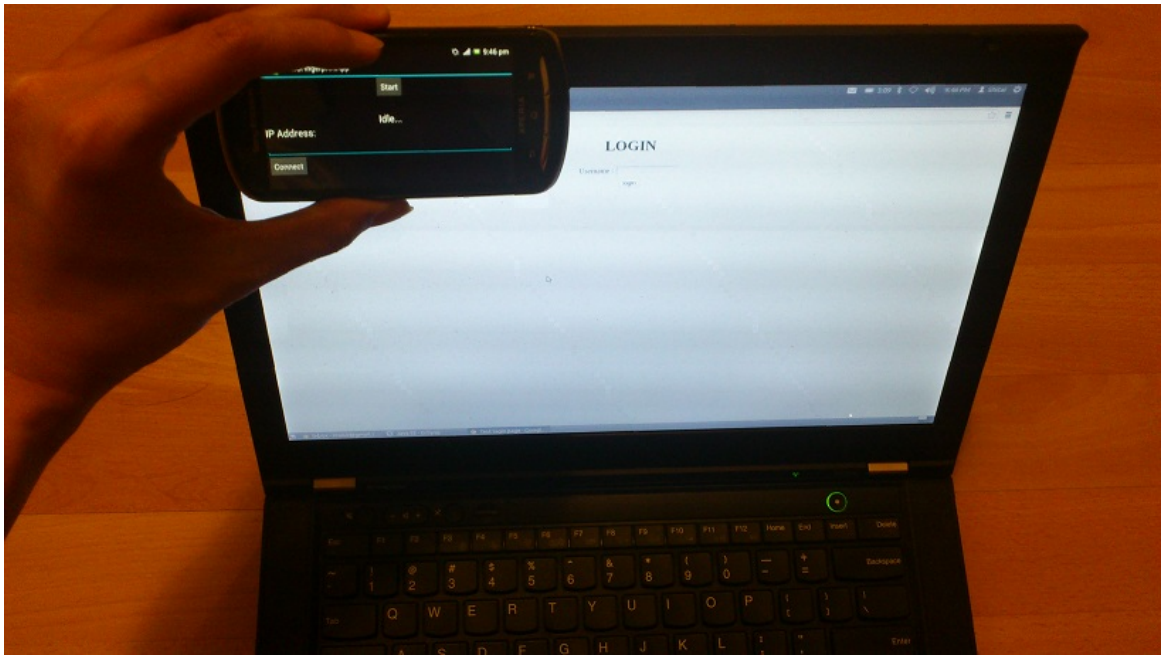


Figure 4.3 Sound-based Authentication

We implement 16 bits, 32 bits and 64 bits length for the sound-based authentication protocol. To our best knowledge, we can encode 32 bits into a piece of 0.25-second sound, which can be recognized by both smart phone and laptop. When we encode 64 bits into a piece of 0.25-second sound, the smart phone can recognize the sound

played by the laptop; but the sound played by the smart phone cannot be recognized by the laptop. This is probably because of the hardware constraint on the smart phone end, because we can successfully simulate the 64 bits experiments from laptop to laptop, but not from smart phone to laptop. In the implementation of 64 bits, we send one piece of 64 bits sound from laptop to smart phone, and send two pieces of 32 bits sound(with 1000 milliseconds silence between the two pieces of sound) to form a 64 bits number from smart phone to laptop.

The experiment results show that if the encoding band falls in frequency lower than 1000 Hz, the error rate is high due to the background noises. For 16 bits we choose the frequency band from 7920 Hz to 9200 Hz with a sub band size of 80 Hz to encode the 16 bits into it, as shown in Figure 4.4. Similarly, we choose the frequency band from 7920 Hz to 10480 Hz to encode the 32 bits into it and the frequency band from 5360 Hz to 10480 Hz to encode the 64 bits. Then we use this configuration to

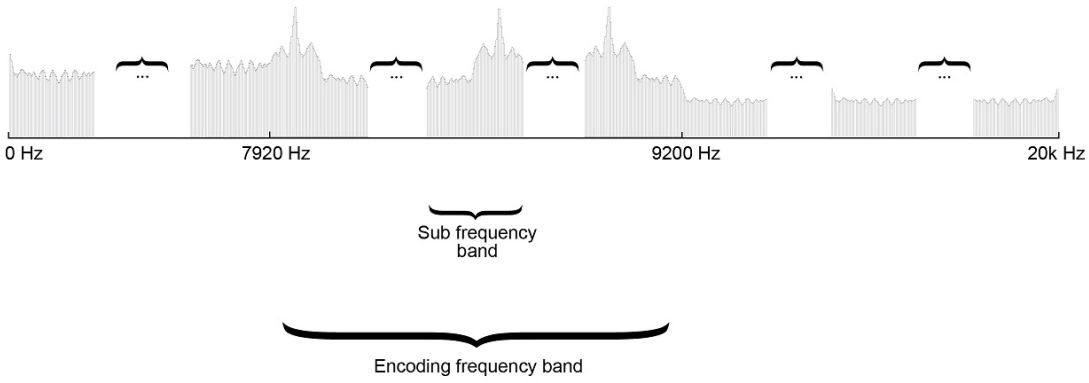


Figure 4.4 Encoding band

perform the sound-based authentication in different environments. The result is in

the following table. The ANRL lab is a spacy lab with people discussing all the time during the experiments. The Trottier Building refers to an open area in the edifice, where more people talk and walk around. The author's apartment is the smallest test environment, where construction works make noises during day time outside the window.

Table 4.1 Error rate

Bits \ Places	ANRL lab	Trottier Building	Author's apt	Time
16 Bits	0%	0%	0%	2366 ms
32 Bits	5.3%	3.7%	6.3%	2368 ms
64 Bits	10.5%	12.1%	15.6%	3401 ms

4.7 Application Prospect

The sound-based authentication scheme is not only a good candidate for stand alone for authentication purpose, but also a good one for two-factor authentication. The smart phone can support multiple user names and secrets for different accounts. These different user names, secrets and accounts are independent from each other, this also makes the scheme resist against phishing attacks.

A new alliance named FIDO (Fast IDentity Online) Alliance was formed in July 2012 and is emerging recently. It addresses the lack of interoperability among strong authentication devices as well as the problems which users face with creating and remembering multiple user names and passwords. We find the approach that FIDO uses to solve this kind of problems has the same main idea as ours.

One advantage that both FIDO and sound-based authentication have is single

token multiple accounts. With one device, the users can have multiple user names and secrets with different accounts. This approach ensures the security and eliminates the passwords in the same time.

Another advantage is that both approaches are resistant against typical on-line attacks, such as phishing attacks, replay attacks and man in the middle attacks.

To the best of our knowledge, we think the sound-based authentication protocol can work as a supplement to FIDO. FIDO achieves two-factor authentication by combining two factors among "something you have", "something you know" and "something you are". As we discussed, there are several disadvantages when using "something you are", so it is better to use the other two factors. The sound-based authentication scheme can work as "something you have" with zero cost, whereas the FIDO hardware device costs more. Also, the sound-based authentication scheme can work with FIDO hardware device as two different things you know, thus the user can totally eliminate password, and the biometric information is not involved.

We have implemented the sound-based authentication protocol for web sites, a possible future for it is to integrate with FIDO solution, since it needs the server side support as well as browser extensions, which is exactly what FIDO needs to work.

Chapter 5

Related Work

In this chapter, we put forward other strong authentication frameworks. As we discussed before, typically there are three factors that can be used in authentication purpose, "what you know", "what you have" and "what you are". In an authentication framework, they can be respectively presented by shared secret(also known as password), authentication token, and biometric information.

Generally, in the most widely applied strong authentication frameworks, a combination of two factors are used to enhance the security, in which, the factor password and authentication token are most commonly used. In the following sections, we talk about several famous and widely applied solutions with two-factor authentication,as well as several existing sound related authentication solutions.

5.1 RSA SecurID Solution

The RSA SecurID solution is a widely emplyed two-factor user authentication framework. According to [14], RSA Security had 72 percent market share in the hardware

token based authentication market for 2003.

The RSA mechanism is developed by RSA(also known as Security Dynamics before) for performing two-factor authentication, whose basic idea is to deal with two factors in the authentication protocol - what you know and what you have. The factor "what you know" here is the password of the user, and the factor "what you have" is an RSA SecurID token.

An RSA SecurID token - either a hardware token or a software token - is assigned to the user who wants to use this two-factor authentication to communicate with his/her service. The token can generate an authentication code at every fixed time interval(often 60 seconds) based on its built-in seed and time. The seed is unique for each RSA SecurID token, and this seed(secret) is shared between the token and the server.

When a user authenticates with a server, he/she needs both the password and the generated key by the RSA SecurID token at the same time; the server also has to keep tracking the the valid keys based on the user's seed. When a user enters his/her RSA SecurID key, the server compares this key with the key it has calculated, and then decides the authentication result. An important thing is the clock at the RSA SecurID token has to be synchronized with the clock at the server end.

It is obvious that the RSA SecurID solution enhances the traditional password-based authentication; meanwhile, it brings some problems as follows.

- **Time synchronization:** In order to authenticate with the server successfully each time, it is necessary for the clock in the RSA SecurID token to be synchronized with the clock at the server. Typically the synchronization is not handled by the user, but rather by the hardware itself. The problem appears when the

hardware token runs out of battery or expires, which means a new token has to be re-issued. It will affect the user during the token being replaced, it is also expensive to replace the token periodically.

- **Time consuming:** In every two-step authentication, it takes long to fulfil the whole process. The user has to enter his/her password, then check the RSA SecurID key on his/her token and enter it. A compromised way to solve this problem is to concatenate password and RSA SecurID key as one password, which reduces one step, but the amount of characters the user has to enter remains the same.

- **vulnerability to attacks:** One possible attack is to obtain the RSA SecurID key and use it within the valid time. This can be done by reading the key when a user is using it or using a key logger. The RSA SecurID authentication server has a mechanism to prevent this attack from happening by declining the authentications happened in a time interval, i.e., if there are more than one authentication requests happening within a certain time, the server assumes that there must be at least an attacker presents, so it rejects all the authentication requests.

  The most severe problem with RSA SecurID solution is the vulnerability to man in the middle attack. Suppose the attacker can impersonate the server, then he/she is able to get the password and the RSA SecurID key and use them to authenticate with the server within the valid time for the key.

- **Multiple devices:** For different online services that a user may exploit, multiple RSA SecurID tokens have to be issued. Consider that only very important

accounts need higher security, for which a user still might have more than one physical tokens or mobile applications, the complexity and mutation among different devices will affect the usability a lot.

5.2 Google Two-step Verification

A prevalent variation for two-factor authentication today is to use cellphone as the second factor. It is unnecessary to be a smartphone with an application to do the hardware token's job. Unlike the RSA SecurID solution, the cellphone only needs to have the function to receive SMS message, and use this SMS message as the key in the second step of authentication. Let us take Google's two-step verification as example here.

Google launched its two-step verification for Google accounts like Gmail and other Google services in 2011. [15] The first step is the traditional user name and password authentication, during which a user goes to the front page of Gmail and enters his/her Google user name and passowrd, as shown in figure 5.1. The second step comes when the user passes the first step verification. In the second step, a six-digit one time key is sent to the user's cellphone via SMS message, who then has to enter this six-digit code to finish the authentication process to use his/her Google services. as shown in figure 5.2. The user can choose to trust this computer at the time he/she is entering this six-digit code, so that, he/she won't be asked for the second step verification when he/she uses the same computer. However, when the user tries to login with another computer, the scheme is triggered again. It is up to the user to choose whether to use SMS message or an application for smartphone to receive or generate the second step code. In general, this two-step verification with mobile application is

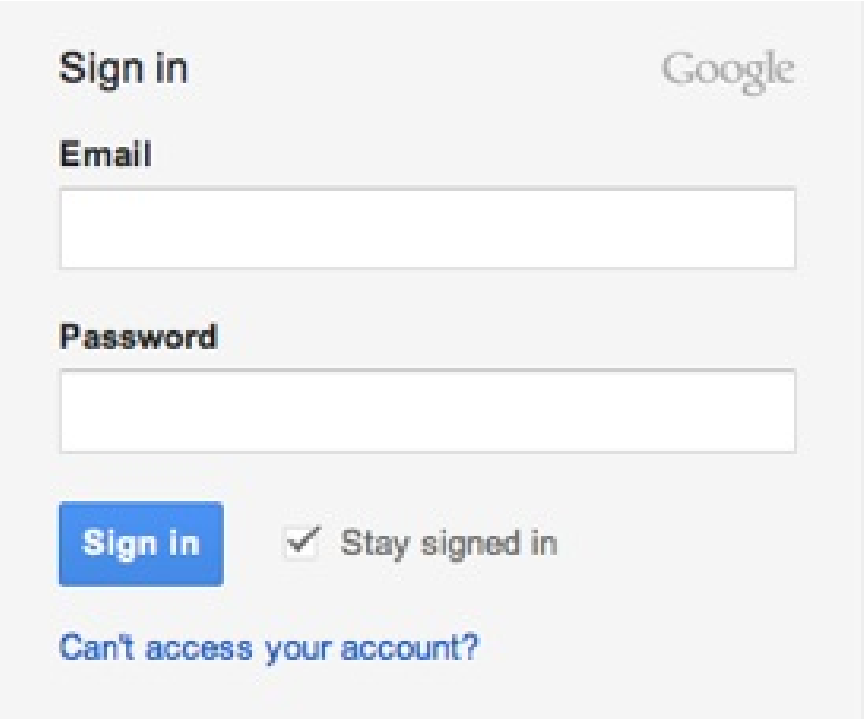


Figure 5.1 Two-Step Verification: Step One

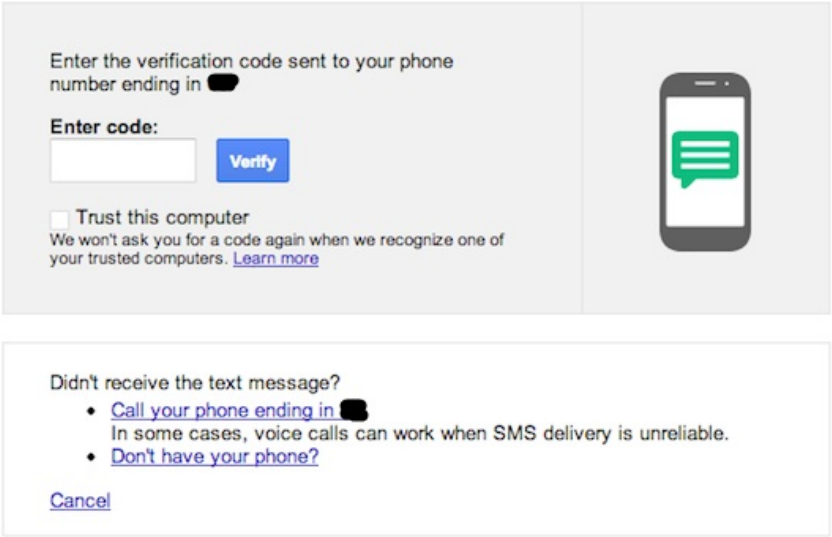


Figure 5.2 Two-Step Verification: Step Two

almost the same as the RSA securID solution, but without time synchronization issue. The smartphone can always synchronize its time with online time server or carrier's time server. The two-step virification with SMS message does not either have time synchronization problem compared with RSA SecurID solution.

5.3 YubiKey

Yubico, founded in 2007, provides a strong two-factor authentication by taking advantage of their product, the YubiKey. [16] They claim millions of users over 100 countries using YubiKey as two-factor authentication to secure access to computers, mobile devices, and online services. The YubiKey is a is a hardware two-factor authentication token, which provides strong two-factor authentication, combining with user's password. Unlike RSA SecurID and Google two-step verification, YubiKey produces 128 bits strong key with AES encryption. This feature not only enhances the security of the second step of the authentication, but also liberates the user from entering a second step key code thanks to the YubiKey which enters the second step authentication code automatically through USB port. Another advantage of Yubikey is, it has no battery, the built in clock uses the power from the USB port.

In a most recent paper from Google [3], Eric Grosse *et al.* mentioned a Smartcard-like USB Token solution to sufficiently keep user safe. Interestingly an article [17] pointed out that Google is working with Yubico with intentions of creating a sort of ring-like USB key which contains "Password Key".

5.4 VASCO DIGIPASS

VASCO focuses on strong authentication and e-signature solutions, specializing in financial, enterprise security, e-commerce and e-government industries. [18] DIGIPASS is the client side product of VASCO for strong authentication. The DIGIPASS product line includes: Single Button DIGIPASS, E-signature DIGIPASS, Software DIGIPASS, Embedded DIGIPASS, Card Reader DIGIPASS, PKI DIGIPASS. The DIGIPASS, as in other two-factor authentications, is a token to generate one time password for user to login in strong authentications. A good feature for DIGIPASS is its ability to generate 3 different one time passwords at the same time. Also, the three different passwords can be different in length, as well as a customized message if desired. As a multiple OTP device, the DIGIPASS can be used in different applications for one service provider. For instance, a bank customer can use a DIGIPASS for login, payment validation and money withdrawals with different passwords.

5.5 VeriSign VIP

VeriSign Validation & ID Protection (VIP) is another two-factor authentication product. Each time a user logins to a VIP member Web site, in addition to entering user name and password, a code generated by VIP is needed to prove the identity of this user. The VeriSign VIP product line is consisted of three categories: VIP Access for Mobile, VIP Access Desktop, VIP Security Token or Card. The VeriSign VIP generates a 6 digits one-time password as the second factor in the authentication process, which is applied in various VIP member Web site, such as Paypal, T-Mobile, etc.

5.6 Audio-based Self-organizing Authentication

In 2009, Kim *et al.* [19] proposed an audio-based self-organizing authentication scheme which allows devices to authenticate each other whenever they are in the same physical audio region without pre-shared secret or human involved.

This approach focuses on the authentication problem that is the process to prove a device's identity by detecting its current region. Figure 5.3 is an example scenario

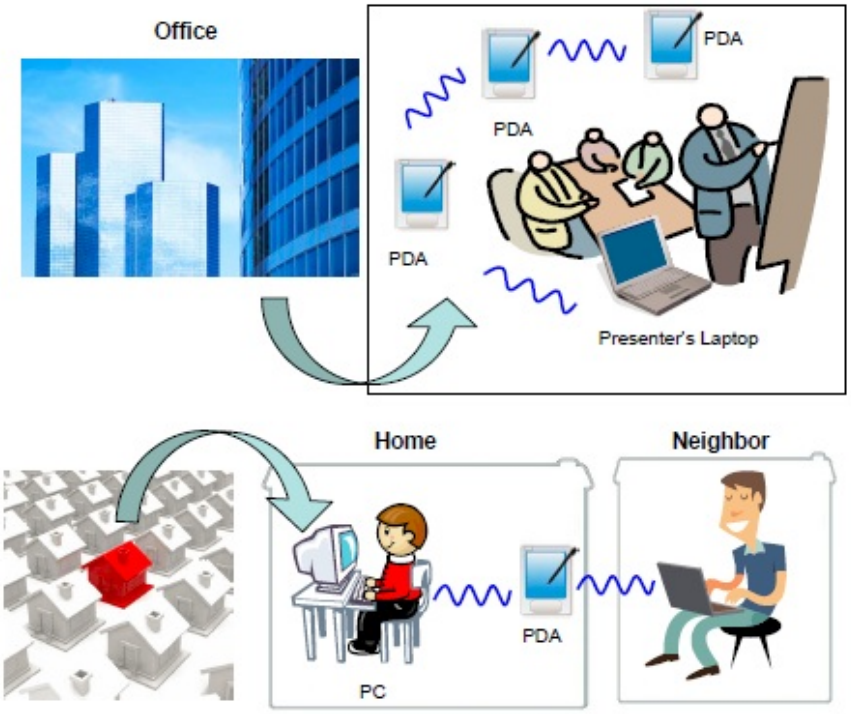


Figure 5.3 Smart Space, taken from [19]

for audio-based self-organizing authentication. A user Alice works during the day, when she has a meeting with her colleagues, their smart devices such as smart phones and laptops form a group, which allows the exchange of file and contact information

within this group. Clearly the information shared within this group is sensitive, and they hope that nobody within the communication range can remotely take advantage of a powerful antenna to eavesdrop and learn their secret data. The way to achieve

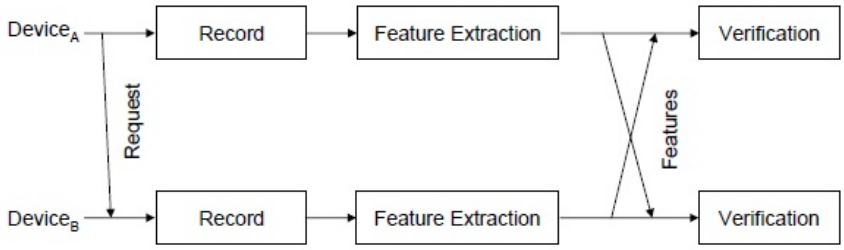


Figure 5.4 Authentication protocol, taken from [19]

this is to record the environment sound at each device independently, after that, the devices verify the sound recorded with each other. If the sound recorded are the same, it implies they must be physically in the same region. The proposed authentication protocol contains three steps, as show in figure 5.4.

- **Recording** In this step, each device records the environment sound for a certain time. There are two important factors in this step, which are synchronization and record duration. To verify with each other, the devices must be synchronized well to make sure the sounds recorded are the same. The record duration is another crucial factor in this approach, the longer it records, the better performance it should have; but when the record time is too long, the usability is affected.

- **Feature extraction** In this step, a feature for a particular piece of sound is extracted. The way to extract the feature is like the approach in this thesis: by

applying FFT, an N-point FFT (length N) is obtained. The final feature set is just this original set.

- **Verification** After exchanging the extracted feature, the devices are able to compare them and verify whether they are in the similar region. If so, the authentication is successful.

5.7 Clear and Loud

In 2006, Goodrich *et al.* [20] proposed a human-assisted authentication for secure device paring by using sound channel as well as wireless channel. In C&L, two devices exchange their public keys through the wireless channel first. The hash of the target device's key has to be verified by humans holding the devices. Since the hash value is relatively lengthy(more than 100 bits), the string of bits are translated to syntactically correct (but usually nonsensical) an English-like sentence. One device plays it by using text-to-speech technology and the other displays it. The human users compare them and verify their keys.

5.8 HAPADEP

In 2008, Soriente *et al.* [21] proposed Human-Assisted Pure Audio Device Pairing(HAPADEP), which is similar to C&L but without a display. In HAPADEP, the public keys are exchanged through sound channel. The HAPADEP protocol consists of two phases, as shown in Figure 5.5. During phase one, the personal device uses a so called fast codec to encode the public key into a 3.4-second unpleasant sound and plays it to the target device. When the target device receives this sound, it decodes it and takes

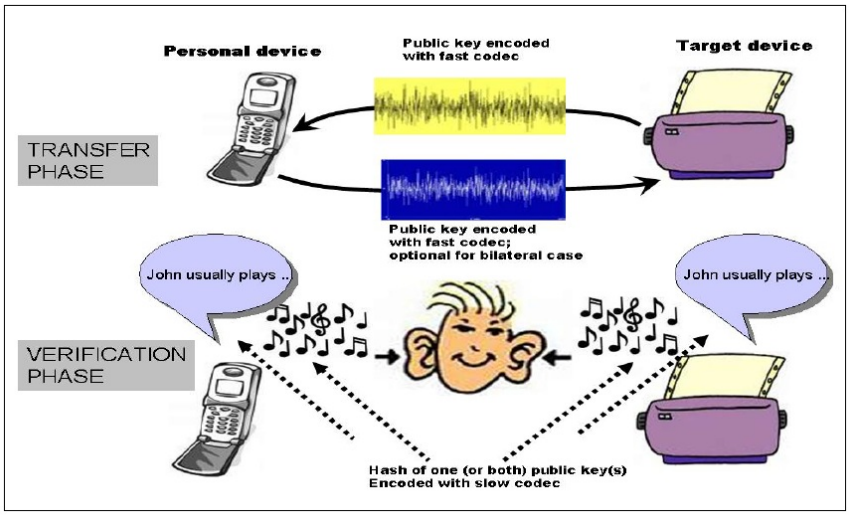


Figure 5.5 HAPADEP protocol, taken from [21]

the hash of the exchanged cryptographic protocol data to produce a 5.6-second piece of MIDI music. After this, both the personal device and the target device play the same MIDI music, the user has to listen to these two pieces of MIDI music and decide whether they are the same.

5.9 Smart Phone User Authentication Using Audio Channels

After finishing this thesis, I found that Lee *et al.* [22] proposed a similar idea to my protocol in 2012. In [22], The smart phone works as the hardware token to prove its owner's identity by receiving audio challenges from a target system. Figure 5.6 shows the authentication protocol as well as the performance sequence. The solid, dotted, and dashed lines represent http, audio, and socket communications, respectively. This system is only a prototype-like system and very vulnerable compared to the work done by this thesis, the system in [22] cannot prevent attacks such as man in the middle

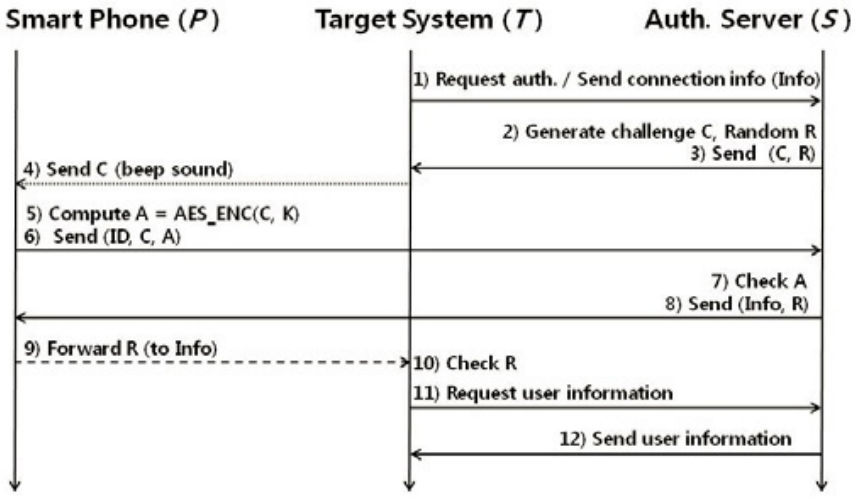


Figure 5.6 User Authentication Using Audio Channels, taken from [22]

attack and replay attack, reflection attack. The reason is that the nature of the sound channel is open to everybody. The audio challenge in [22] is transmitted using beep sounds with a simple encoding method in which each digit is assigned a specific frequency. The audio channel used is from 15 kHz to 20 kHz with a distance of at least 600Hz from each other. In the protocol, 1-second beep can contain 4 bits and a challenge is composed of 16 bits, so 4 beeps are needed to transmit a challenge. Because one beep takes for 1 second and there is also a 1-second gap between beeps to prevent interference, challenge transmission requires 7 seconds. Meanwhile, the response from the mobile token to the server is carried by network channel, thus the approach relies on both sound channel and network channel. It means the mobile token does not take all the advantages of sound channel. Without a one time password scheme, at least an 128 bits key is needed to ensure the security, which cannot be achieved by this approach. In sum, this work only provides a similar idea to the thesis, but far away from an applied protocol.

5.10 A Novel User Authentication Scheme Based on QR-Code

Liao *et al.* [23] proposed an interesting authentication scheme based on QR-code in 2010. The idea in this paper is to use QR-code as carrier of one time password, just like the idea in this thesis. A Japanese company Denso-Wave introduced a two-dimensional barcode - QR-code [24] - in 1994. A QR-code can contain a great volume of information: 7,089 numeric only, 4,296 alphanumeric characters, 2,953 bytes of binary (8 bits). This capability makes QR-code a great candidate for authentication purpose, since it can represent a complex enough one time password. Considering the web camera is so common on every laptop or mobile phone, as well as the QR-code encoding/decoding software is easy to obtain or embedded in the mobile phone, thus it is possible to develop an authentication scheme based on QR-code.

The scheme in [23] consists of two phase: Registration and Verification phases. During the registration phase, the mobile phone and the service provider share a secret key x_A, the authentication phase is shown in figure 5.8, where the notation can be found in figure 5.7. We can see that the QR-code is only used in step two, which is from service provider to user. This means the service provider need to display this QR-code to the user in some way, which might affect the visual presentation of the service.

Google is also interested in this QR-code based authentication scheme [25], which is a variation of Google's two-step authentication. Like the scheme in the paper, a QR-code is displayed on a special Google Web page. Once the QR-code is scanned, it will return a URL on the user's mobile phone and once this URL is tapped, the desktop browser will automatically redirect to the user's loginned Google Account without

Notation	Description
$h(\cdot)$	An one-way hash function
$E_{QR}(\cdot)$	A function that encodes data into QR-code image
$D_{QR}(\cdot)$	A function that decodes the QR-code image captured in an embedded camera device
s	*SP*'s long-term secret key
T_1, T_2	Time stamps

Figure 5.7 QR-code authentication: notation, taken from [23]

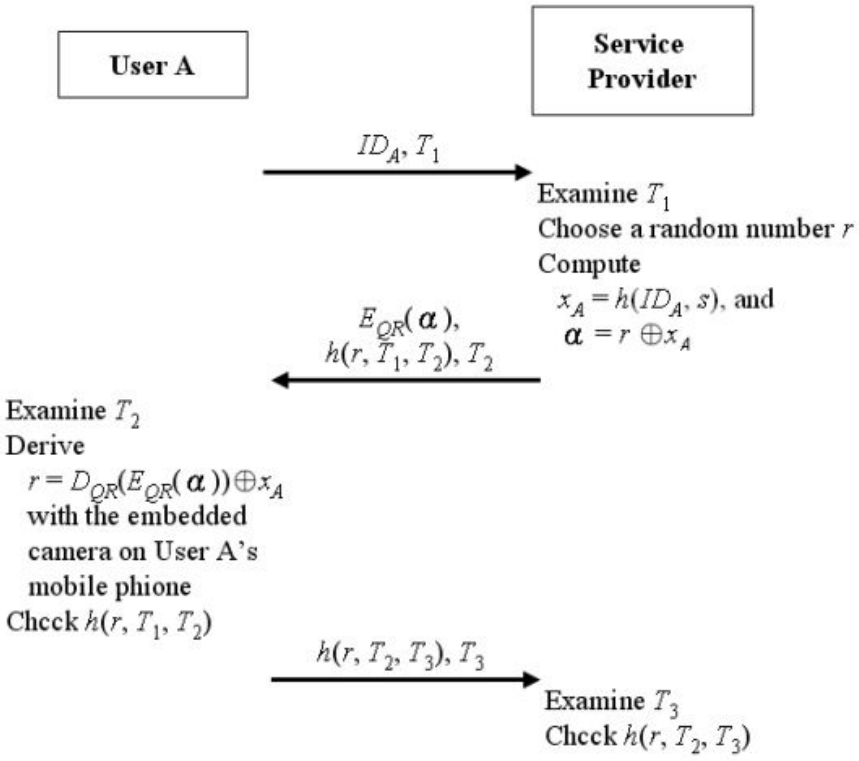


Figure 5.8 QR-code authentication: verification phase, taken from [23]

requiring a password. As we can imagine, this scheme needs a high integration of the mobile phone and desktop system.

Chapter 6

Conclusions and Future Works

6.1 Conclusions

This thesis seeks to develop an efficient and safe authentication scheme for a general authentication purpose. Three key factors are used in authentication frameworks: "What you know", "What you have", "What you are". The most widely applied authentication scheme is "What you know", *i.e.*, the shared secret between user and server, also know as password. To achieve strong authentication, two factors of the three are often combined together. The choice is always "What you know" and "What you have", *i.e.*, the password and the token which generates one time password.

In this thesis, we propose a sound-based authentication framework, which only needs one factor "What you have" to achieve strong authentication. In this framework, the user does not need to memorize or even enter password by him/herself. The whole protocol is finished between the server and the user's token automatically. Thanks to the sophisticated design of the protocol, for only one factor authentication is strong enough from attacks. Since the protocol make use of sound channel, whenever a token

functions for authentication, the user can be aware of if he/she is physically present near the token. In the sound-based authentication framework, the token is always a smartphone or other smart devices, we assume that the token is always carried by its owner, so it is impossible to hijack the token and launch the application without letting the owner know.

Testing results show that the error rate for the sound-based authentication protocol is relevantly low when the speaker is near enough to the microphone. In most of the tests conducted by the author, the sound recognition program can pick up the sound signal correctly and get the right sound signature. The whole protocol costs roughly one second, which is sufficient enough for authentication purpose.

The sound-based authentication framework can be applied to securing access to personal computers, servers, and online services. Especially for a user who often has countless online services accounts, it whill keep him/her from memorizing and entering different passwords for each online service, while enhancing the security at the same time by using the sound-based authentication framework.

6.2 Future Works

The implemented prototype for sound-based authentication shows this new idea works fine. In this chapter we discuss several potential future works to extend this protocol to daily use.

6.2.1 Browser extensions

In Chapter 4 we have seen a sample web site specially designed for sound-based protocol. To encourage the online service providers to use sound-based protocol and

reduce their costs, it is necessary to develop extensions for main browsers that convert the message sent between server and user to sound signal, and vice versa. With such extensions installed to browsers, the sound-based authentication protocol can be applied on any web site, without to much work for the service providers. Also, it is the user and the service provider who decide whether or when to use sound-based protocol: it is flexible.

6.2.2 Sound library

In the current design and implementation, a sound piece with different frequencies is generated and exchanged for authentication. This sound is meaningless and not amusing to users currently. It can be more attractive in the form of a piece of pleasant music when being played for authentication, instead of a sound with random frequencies. The trade-off is a pre-defined music library which has to be installed along with the application, and will cost extra storage both on the smartphone and the server.

There are two challenges in this idea. First, as we have seen in figure 3.1, the frequencies of music mostly fall into the frequency band under 5K Hz, and most of them are under 1K Hz. It should also be bared in mind that the lower frequency band contains more information, which means more accuracy is required in lower frequency band. Since most of the background noise and music notes are located in a narrow band from 20 Hz to 1K Hz, it is very challenging to extract the key frequencies from a music piece. The second challenge is the one way hash function. When a music library is selected, it is necessary to have the hash value for any music piece is also the feature of a music piece in the same library, better without collisions. This requirement demands a very carefully chosen music library and carefully designed hash function

or map.

6.2.3 Enhanced encoding

We have proved that a 32 bits sound-based authentication is more secure than the PIN for bank account, but for security purpose, the more secure the better. Also, we would like to achieve 64 bits or even longer bit length by one piece of sound, instead of our current approach. Based on our experimental results, when more information is encoded in one sound piece, more interference shows up, and more it becomes difficult to extract the right sound signature along with the information encoded in the sound increases. I tested a noise cancellation algorithm to increase the accuracy, in which the program keeps recording and analysing the background sound, when a sound piece needs to be sent for authentication, it also add a wave with the same amplitude but with inverted phase to the original sound. This approach does not increase the accuracy significantly, because of the noise sound sources are everywhere, to cancel all of them, multiple speakers are needed. Thus, to encode more information in one piece of sound, more sound based analyse must be involved.

Bibliography

[1] Pingdom, "Internet 2012 in numbers." http://royal.pingdom.com/2013/01/16/internet-2012-in-numbers/, 2013.

[2] R. Waugh, "No wonder hackers have it easy: Most of us now have 26 different online accounts - but only five passwords." http://www.dailymail.co.uk/sciencetech/article-2174274/No-wonder-hackers-easy-Most-26-different-online-accounts--passwords.html, 2012.

[3] E. Grosse and M. Upadhyay, "Authentication at scale," *IEEE Security & Privacy*, vol. 11, no. 1, pp. 15–22, 2013.

[4] J. Kent, "Malaysia car thieves steal finger." http://news.bbc.co.uk/2/hi/asia-pacific/4396831.stm, 2005.

[5] B. Soleymani, "Social authentication for mobile phones," Master's thesis, Electrical and Computer Engineering McGill University, 2009.

[6] E. Britannica, "Sound — encyclopaedia Britannica." http://www.britannica.com/EBchecked/topic/555255/sound. [Online; accessed 03-March-2013].

[7] Wikipedia, "Sound — Wikipedia, the free encyclopedia," 2013. [Online; accessed 03-March-2013].

[8] E. Britannica, "Frequency — encyclopaedia Britannica." [Online; accessed 03-March-2013].

[9] E. Britannica, "Amplitude — encyclopaedia Britannica." [Online; accessed 03-March-2013].

[10] M. Rahman, *Applications of Fourier Transforms to Generalized Functions*, ch. 1. WIT Press, 2011.

[11] J. B. J. Fourier, *The analytical theory of heat*. The University Press, 1878.

[12] Wikipedia, "Fast fourier transform — Wikipedia, the free encyclopedia," 2013. [Online; accessed 03-March-2013].

[13] L. Lamport, "Password authentication with insecure communication," *Commun. ACM*, vol. 24, pp. 770–772, Nov. 1981.

[14] RSA.com, "Rsa securid solution named best third-party authentication device by windows it pro magazine readers' choice 2004," 2004.

[15] A. PASH, "Set up google's two-step verification now for seriously enhanced security for your google account," 2011.

[16] Yubico. `http://www.yubico.com/about/`.

[17] C. Burns, "Google taps yubico for password usb rings of the future." `http://www.slashgear.com/google-taps-yubico-for-password-usb-rings-of-the-future-18265897/`, 2013.

[18] VASCO. `http://www.vasco.com/company/about_vasco/short_overview/short_overview.aspx`.

[19] S. J. Kim and S. K. S. Gupta, "Audio-based self-organizing authentication for pervasive computing: A cyber-physical approach," in *Proceedings of the 2009 International Conference on Parallel Processing Workshops*, ICPPW '09, (Washington, DC, USA), pp. 362–369, IEEE Computer Society, 2009.

[20] M. Goodrich, M. Sirivianos, J. Solis, G. Tsudik, and E. Uzun, "Loud and clear: Human-verifiable authentication based on audio," in *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*, pp. 10–10.

[21] C. Soriente, G. Tsudik, and E. Uzun, "Hapadep: Human-assisted pure audio device pairing," in *Proceedings of the 11th international conference on Information Security*, ISC '08, (Berlin, Heidelberg), pp. 385–400, Springer-Verlag, 2008.

[22] M.-K. Lee, J. B. Kim, and J. E. Song, "Smart phone user authentication using audio channels," in *Consumer Electronics (ICCE), 2012 IEEE International Conference on*, pp. 735–736, Jan.

[23] K.-C. Liao and W.-H. Lee, "A novel user authentication scheme based on QR-code," *Journal of Networks*, 2010.

[24] I. 18004:2000., "Iso/iec 18004:2000. information technology-automatic identification and data capture techniques-bar code symbology-qr code," 2000.

[25] R. Naraine, "Google testing login authentication via qr codes." `http://www.zdnet.com/blog/security/google-testing-login-authentication-via-qr-codes/10105`. [Online; accessed 03-March-2013].

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd ed., 2009.

[27] P. Duhamel and M. Vetterli, "Fast fourier transforms: A tutorial review and a state of the art," *Signal Processing*, vol. 19, no. 4, pp. 259 – 299, 1990.

[28] H. Bojinov and D. Boneh, "Mobile token-based authentication on a budget," in *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*, HotMobile '11, (New York, NY, USA), pp. 14–19, ACM, 2011.

[29] "Fips 46-3: The official document describing the des standard." `http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf`. [Online; accessed 03-March-2013].

[30] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, pp. 120–126, 1978.

[31] C. Kaufman, R. Perlman, and M. Speciner, *Network security: private communication in a public world, second edition*. Upper Saddle River, NJ, USA: Prentice Hall Press, second ed., 2002.

[32] B. Soleymani and M. Maheswaran, "Social authentication protocol for mobile phones," in *Computational Science and Engineering, 2009. CSE '09. International Conference on*, vol. 4, pp. 436–441, 2009.

[33] J. Clark and J. Jacob, "A survey of authentication protocol literature: Version 1.0," 1997.