Offline Two Factor Authentication using Encrypted Audio

Shrey Gupta
University of Illinois Urbana Champaign
Champaign, Illinois, United States
shreysg2@illinois.edu

Namrata Vajrala
University of Illinois Urbana Champaign
Champaign, Illinois, United States
vajrala3@illinois.edu

Adarsh Suresh
University of Illinois Urbana Champaign
Champaign, Illinois, United States
adarshs3@illinois.edu

ABSTRACT

Two-Factor Authentication (2FA) provides a very important extra layer of security as compared to just passwords. A common 2FA method is for users to confirm their password-based login on a smartphone app. However, lots of users find it annoying to pull out their smartphone and confirm their login on an app [1]. This paper presents a novel solution to this problem by using encrypted audio to provide a hands-off method to perform 2FA using a user's smartphone.

CCS CONCEPTS

• **Security and privacy → Multi-factor authentication**.

KEYWORDS

two-factor authentication, encryption

ACM Reference Format:
Shrey Gupta, Namrata Vajrala, and Adarsh Suresh. 2023. Offline Two Factor Authentication using Encrypted Audio. In *Proceedings of Smart-X '22: Fake ACM Symposium on Smart cities, homes, phones, and beyond (Smart-X '22).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Currently, many services use 2-Factor authentication for account login which involves the use of two unrelated authentication methods to secure an account. It is nearly impossible to secure an account with just one password so services use a second form of verification such as a one-time password, verification code, QR codes, or a hardware token. While these methods may vary, the most common method involves sending a message with a verification code to the user's phone or email which the user has to then enter on their device. This establishes two layers of verification.

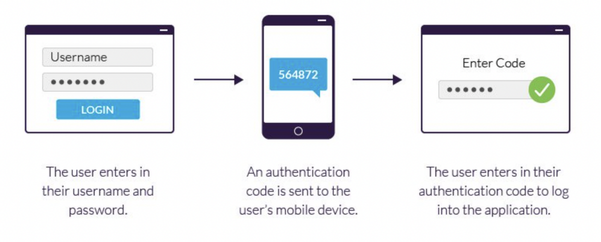
Username
•••••••
LOGIN
564872
Enter Code
••••••
The user enters in their username and password.
An authentication code is sent to the user's mobile device.
The user enters in their authentication code to log into the application.

Figure 1: Original/Baseline Method for 2FA

Smart-X '22, Jan 18–May 05, 2022, Urbana, IL
2023. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

While this process may work well for verification, users have to follow this process every single time they login to their account. This could get very tedious and take up an unnecessarily long time. This also requires a user to have a stable internet connection so that they can receive their text messages or emails. However, this may not always be possible and it could result to them being locked out of their account. This issue has been tackled in projects called

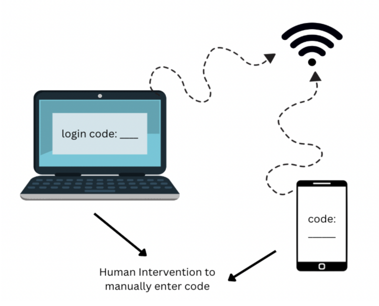

Figure 2: Baseline Method Requirements

ChirpMeIn and Sound-Proof [3] that enable automatic 2FA with the use of audio. When a login code is requested, an audio clip is played to the other device that would contain the code. However, this method is not secure. Another device can easily hack into it and play the same audio to impersonate the user. In addition, the process they used involved playing a random audio from their playlist and they cross-verified this audio from their PC. This method is also not secure and it heavily relies on the internet for uploading and downloading audio files. There are also numerous other solutions such as the Duo Mobile application, hardware tokens, camera based 2FA such as Pixie, etc. However, each of these solutions either require an internet connection or tedious human interaction. The Duo Mobile application, for example, requires a network connection to provide verification and it involves the user to actively access their phone and click a button. This product aims to provide a solution which reliably, securely, and automatically verifies a login code without using a network connection. SoundID [4] uses a mechanism that has the 2FA device (phone) communicate with the login device (desktop) and vice-vers in order to mitigate man-in-the-middle attacks with 2FA. However, this mechanism relies on the produced and received audio passing above a certain similarity score. This reduces the accuracy and effectiveness of the system.

We propose a 2-Factor Authentication System that will allow users to authenticate by playing audio in the ultra band frequency encrypted using cryptographic algorithms not relying on an internet connection for encryption/decryption keys and having the

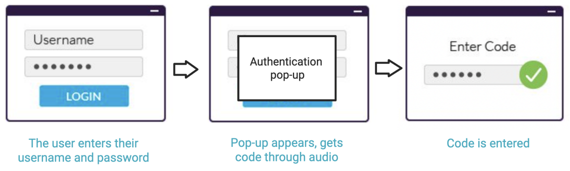
Username
•••••••
LOGIN
Authentication pop-up
Enter Code
••••••
The user enters their username and password
Pop-up appears, gets code through audio
Code is entered

Figure 3: Proposed Method for 2FA

least user intervention, allowing users to authenticate easily and hassle-free. The audio will use data-over-sound technology so that an encrypted text is effectively exchanged using the sound medium. Thus, we provide a more exact means of 2FA since we rely on the exact expected match of the decrypted text extracted from the audio.

Realizing this type of 2FA is, however, non-trivial. There are two main challenges which need to be resolved:

1) Encrypted Audio:
Current 2FA systems use text to authenticate, but it can easily be exploited and can be a hassle to remember and then type into the system. To overcome this scenario we are using audio to authenticate the devices. We need to embed our secret text inside an inaudible audio file.

2) Offline Cryptography:
To make this system work offline, we need to make sure the encryption key used to encrypt the secret text inside the audio is always synchronized with the decryption key which will be used to decrypt the same audio, without any internet connection by using time-based encryption. Using a timestamp on the order of seconds, any inconsistencies with time synchronization on the mobile device and desktop should be negligible.

Our proposed approach consists of our cloud-based web app requiring 2FA for login, a smartphone that has our audio-based 2FA method installed on it, and a user attempting to login to the website and confirming their login using our audio-based 2FA method. We will have three types of tests. One test will be when the smartphone actually emits audio to perform 2FA when the website requests it, another one will be the smartphone emitting incorrect audio, and the third one will be the smartphone emitting no audio at all. In terms of steps, the first step will be the user logging into the website using their password. Then the website will expect audio from the phone. Based upon the test being run, the phone either emits the correct audio, incorrect audio, or no audio at all. Then whether or not the website accepts the 2FA is observed. Results will be evaluated based on the false negative rate of the first test and the false positive rate of the last two tests. A false negative is when the website doesn't accept 2FA when the intention was for 2FA to work as the phone emitted the correct audio. A false positive is when the website accepts 2FA when the intention was for 2FA to not work as the phone emitted an incorrect audio or no audio at all. The efficacy of this paper's proposed 2FA system is correlated with how low the false negative and false positive rates are.

2 RELATED WORK

2.1 Ambient Background Noise Similarity

2.1.1 Sound-Proof [3].

Sound-Proof is an effective way of using audio to facilitate 2FA and reduce user intervention with their phones. They address the issue of using Bluetooth to facilitate 2FA as it requires registration between the phone and the laptop which defeats the purpose of decreasing user intervention. Additionally, JavaScript does not have native Bluetooth functionality, so this method isn't feasible in the first place irrespective of intervention expectations. They also address the issue of proximity present with most modern 2FA methods. Most modern 2FA methods don't require the user to be physically near the device trying to login. They just require the user to press a button on their smartphone. With Sound-Proof, the need for sound recordings from the laptop and phone to be similar requires the user's phone to be within reasonable proximity to the laptop. With all these features, the Sound-Proof paper reported results in which a majority of surveyed users used less time with Sound-Proof than Google 2FA and preferred Sound-Proof.

Though Sound-Proof reported great results, there are several ways to improve it. Firstly, during Sound-Proof's presentation at USENIX, Bill Cheswick of the University of Pennsylvania pointed out that a presentation at a previous USENIX conference displayed that most computers can produce and record sound in the ultrasound range. Thus, Sound-Proof can be improved so that one device produces audio in the ultrasound range while another devices records that audio while most other humans aren't aware that this is happening. Additionally, Sound-Proof relies on a similarity score between the audio recorded by the laptop and the audio recorded by the phone. This creates a degree of uncertainty in the process because a similarity score is not an exact means of detecting equality. The system proposed in this paper improves upon this by requiring exact equality between the text generated and the text received in the laptop that is extracted from the audio recorded from the laptop. Another inconvenience of Sound-Proof is that it requires the user's phone to be connected to the internet to receive the recording from the server. Our paper proposes an offline approach to 2FA such that the user's phone doesn't need an internet connection, reducing the overall network overhead of the 2FA process.

2.1.2 SoundID [4].

SoundID uses mechanisms similar to Sound-Proof to facilitate Two-Factor Authentication via audio. It focuses on mitigating Man-in-the-Middle (MITM) attacks that can be exploited by systems such as Sound-Proof. An example of a MITM attack is when a bad actor knows the user's username and password along with their location. The bad actor can place a smartphone near the user, and then login to the user's account from a distant location. The user's registered phone will then play some sound which is picked up by the nearby smartphone of the bad actor which then relays this audio to the bad actor's login device. Thus, the bad actor can mimic being near the user's registered phone and pass 2FA. Another attack is when the bad actor knows the user's username and password and logs in to the user's account within close proximity of the user so the login device can pick up the audio from the user's phone.

SoundID mitigates such an attack by using two stages: 'Contextual Co-Presence Detection' and 'Dynamic Fingerprints'. The first stage uses a similarity score between the audio that the phone generates and the audio that the phone's microphone and laptop's microphone picks up. The second stage has the server and the phone generate a dynamic fingerprint that depends on the generated and recorded audio in each authentication session. This fingerprint cannot be forged so it provides a strong sense of confidence that the user logging in has kept the registered phone that generated the audio.

Limitations of SoundID include that using a similarity score threshold is a non-exact way of detecting proximity and is more prone to errors, similar to Sound-Proof. Additionally, generating and comparing dynamic fingerprints is a complex implementation. Our paper proposes a simpler method that uses ultrasound to transmit raw data using data-over-sound technologies. This requires an exact equality between the data transmitted and the data expected which provides a stronger sense of confidence. Additionally, this isn't prone to MITM attacks because we use SSH-like encryption schemes to encrypt the data that is transmitted.

3 SYSTEM OVERVIEW

A 2-Factor Authentication System has 3 main components:

(1) An Authenticating Party (some websites like login portals) of interest

(2) an Authentication Provider (such as Duo) that provides a platform for the users to securely verify themselves

(3) Authenticator Key is used by Authentication Provider to verify the user, these are generally short one-time passwords sent via SMS or email but can also be some kind of unique link or some secret text combination that only the authenticating user may know.

These systems are already integrated into login systems of verified Authenticating Parties. Once a user enters their credentials for logging in, the authenticating party sends a request along with the identifier (usually the user's email) to Authentication Provider and asks them to verify. Now it's up to the authentication provider how to authenticate the asked user. Usually, this authentication is done by sending the user an Email or Text-SMS with a unique generated code on their personal device. They have to enter this code on their platform in order to verify themselves.

While this system works well, there is a limitation that arises when there is no internet since this process only works with a stable internet connection. In addition, there is tedious, extra human effort that is required to manually remember and enter the secret text over different devices when trying to authenticate. In our proposed system, we are implementing an Authentication Provider which continuously listens for encrypted ultra-frequency audio. This audio can only be produced from a pre-authenticated device which does not need to rely on an active internet connection. This way, users can verify themselves seamlessly with the least human intervention needed. Effectively, our system has the same system overview as that of Duo or another well-known 2FA provider with the difference being that the Authenticator Key is transmitted via audio from the user's device to the Authenticating Party instead of from the user's device to a backend server to the Authenticating Party.

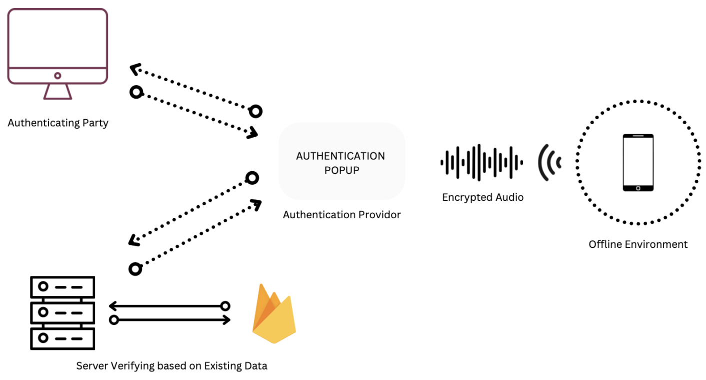

Figure 4: System Overview

We propose a system using audio that follows the following steps:

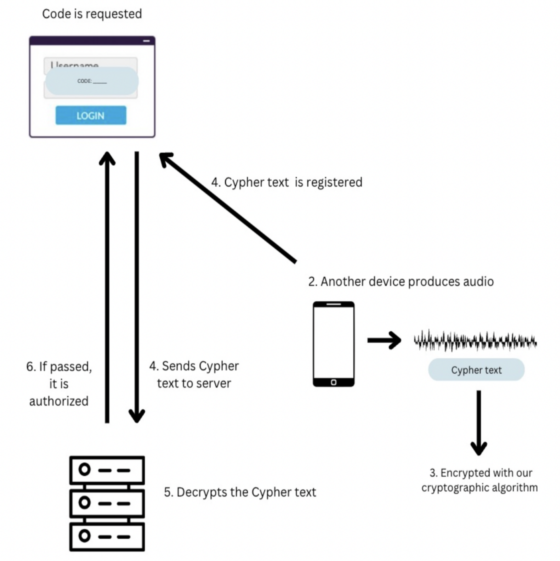
Code is requested
Username
CODE: ____
LOGIN
4. Cypher text is registered
2. Another device produces audio
Cypher text
6. If passed, it is authorized
4. Sends Cypher text to server
3. Encrypted with our cryptographic algorithm
5. Decrypts the Cypher text

Figure 5: Process steps

(1) When a user tries to verify themselves on the platform, the system will start capturing audio using a microphone

(2) Simultaneously, their pre-authenticated device (usually their mobile phone) will start producing an ultra frequency sound containing a cipher text.

(3) This cipher-text is a user-identifier (assigned when the user first logins into their device) encrypted using a particular

key pair from a set of key pairs, which is selected based on the current timestamp using our cryptographic algorithm.

(4) Once the system has completed capturing the encrypted audio, it will extract the cypher text from the received audio and send it to the authenticator server for verification.

(5) The server will then decrypt the received cipher text using the key pair which was supposed to be used at that timestamp and cross-check the decrypted user identifier with the assigned value at the time of login.

(6) If it passes the previous check, then the user is authorized and the authenticating party is notified. However, if it fails, the system repeats the process. If it fails again, the user is assigned unauthorized and again the party is notified.

With this process, we can ensure that the 2FA code gets properly transmitted from one device to another while ensuring safety and by not using any internet connection.

4 PROPOSED APPROACH

Traditional Two-Factor Authentication systems can be inconvenient for users, as they may require additional steps or may be time-consuming to use. This can lead to user frustration and may discourage users from using the system. Moreover, many current two-factor authentication systems rely on a network via mediums, such as WiFi or Carrier Data, as the internet provider. This can be inconvenient for users who do not have an internet connection at all times. To add on, some authentication systems even require users to perform additional steps or enter additional information to complete the authentication process. This can be inconvenient and disrupt the user's workflow.

To overcome all these mentioned issues, we have developed Audio based Offline Two-Factor Authentication System which resolves some major flaws of current authentication systems. This approach doesn't rely on the internet connection, so even in times of random wireless connectivity issues, authentication between both devices would be seamless. Additionally, as this authentication system uses audio as its medium of authentication, this wouldn't hassle users with remembering their on-time passwords instead will have the least user intervention with their mobile devices. As the current two-factor authentication systems lack these capabilities, this system would make the better choice.

To make two-factor authentication work offline and be independent of network connection, we have come up with a time-based cryptographic system which once synced then it won't need to communicate with the server for passing authentication token. So the user will require an active internet connection for the first time when they are registering themselves but won't need it ever after. As we want to communicate with our authenticating device seamlessly, we are using audio as the medium. In current systems, users are asked to remember their one-time passwords to authenticate but here, using audio, the system will authenticate itself given that both are in the audible range. This would avoid users from remembering and entering these random passwords and share all this information discretely using audio between all the devices. The sensitive information from the authenticator device would be encrypted using advanced cryptographic algorithms and embedded into playable audio using state-of-the-art steganography techniques. This would make it very difficult for any intruder to gain unauthorized access using our system.

This approach being offline, secure and requiring the least human intervention would make it one of the best two-authentication systems available in the industry. This system can be advantageous in scenarios where we have multiple devices to authenticate at a time because a single authentication audio would allow all devices to authenticate all together whereas, current systems will we need a separate one-time password for each authenticating device which sometimes can be difficult to find the correct password for the system. As the sensitive information is encrypted using AES 256-Bit with random IV, the authentication is secure more than ever.

We have seen how we are going address our problems with current two-factor authentication systems, such as having the least steps for authentication or making it less reliant on a network connection or even making it more secure. In the upcoming sections, we will be discussing how this system will be working in real-time.

5 IMPLEMENTATION

To achieve this type of two-factor authentication system, we need to develop our web interface for acting as authenticating party and mobile app acting as an authenticated party to authorize any transaction.

5.1 Mobile Application

Our mobile end application needs to send information via audio and maintain synchronization for our time based cryptographic encryption to work.

5.1.1 Time-based Cryptographic Encryption.

To gain complete offline synchronization to work, we decided to use a timestamp. This timestamp is an integer that indicates the number of seconds from 1 January 1970 to the current exact time. The timestamp is supposed to be synced for every device given that the correct time is set on their device. If we can select our encryption key based on the current timestamp, we will always end up selecting the same key on all the devices at that given time frame.

There can be several ways to select an encryption key from a set of keys based on timestamp but we plan to select our key based on division on our timestamp. We select a prime number and have a set of encryption keys of the size same as the prime number. If we perform a modulus operation on the current timestamp using our selected prime number, we will get the result an integer which is less than the selected prime number. We can use the result as an index to select the key from the set of encryption keys. This way we will always have the same key selected on all our devices given that the same prime number is selected.

For example, suppose the selected prime number is 7, so we will generate seven random keys when the user registers and store them as a set of keys in our database and local storage. When the user tries to authenticate offline, the current timestamp is recorded. Suppose the current timestamp value is 392042034, if we use modulus operation on 392042031 using our selected prime number seven, we get three as our remainder. We can fetch the key at the third index from the set of keys saved in our local storage during registration and use that key for our AES Algorithm. This would make our system more secure as our key will be changing every minute and won't be easy to decode. Here selected key was a low prime number, seven, if we select a bigger prime number such as 2539, it would create a lot of randomness in our key selection and become more difficult to crack.

Once we have selected our key and used it to encrypt our user information we need to send it to the authenticating party so that it can complete the verification process. To communicate between these platforms seamlessly we need to send cipher information using sound.

5.1.2 Data-over-Sound.

Data-over-sound refers to sending data in a format such as text through the physical medium of sound. With respect to our proposed system, we needed to transmit encrypted text through audio, so we leveraged current data-over-sound technologies.

ggwave [2]

In particular, we used the *ggwave* library in JavaScript (web client and server) and Swift (iOS app) to achieve this. The library uses *m*-ary Frequency-Shift Keying modulation to transmit the data. The steps are as follows:

(1) Data to be transmitted is split into 4-bit chunks.
(2) 6 chunks (3 bytes) are transmitted at a time.
(3) The frequency corresponding to each chunk is outlined by the following table

Freq (Hz)	Value	...	Freq (Hz)	Value
F0 + 0 * δf	Chunk 0: 0000	...	F0 + 80 * δf	Chunk 5: 0000
F0 + 1 * δf	Chunk 0: 0001	...	F0 + 81 * δf	Chunk 5: 0001
F0 + 2 * δf	Chunk 0: 0010	...	F0 + 82 * δf	Chunk 5: 0010
...	...	...	...	...
F0 + 15 * δf	Chunk 0: 1111	...	F0 + 95 * δf	Chunk 5: 1111

ggwave has different protocols such as non-ultrasonic and ultrasonic which have to do with the type of audio that's intended to be transmitted. δf is 36.875 Hz for all protocols. F0 is 1875 Hz for the non-ultrasonic protocol and is 15000 Hz for the ultrasonic protocol as that is a rough lower bound for frequencies that humans have trouble hearing.
(4) This is then encoded with Reed-Solomon error correcting codes [5] to add a layer of redundancy and robustness when transmitting audio as it can be affected by loud background noises. The number of error-correcting code bytes is directly proportional to the length of the data being transmitted.
(5) On the sender's side, a BEGIN sound marker is sent, followed by the audio encoding the data, and finally the END sound marker.
(6) On the receiver's side, once the BEGIN marker is recorded, the receiver records the audio until the END marker appears. The audio in between the markers is Fourier transformed to obtain the frequency distribution over that span of time. The frequencies are reverse-mapped to their 4-bit chunks using the aforementioned table and protocol setting. Using Reed-Solomon error correcting codes, the data is finally decoded to produce the original data.

Using this approach we can embed our cipher information inside audio and we need to play this using the mobile speaker until the device in range trying to authenticate can complete their verification.

5.2 Web Interface

Our web interface will always be connected to the internet as it is the one trying to authenticate, therefore, it can always easily access all users and their keys anytime. Whenever a user tries to authenticate, the web app starts listening for audio in the background using a system microphone. Once the mobile app finishes playing audio it plays the audio finished marker as mentioned above triggering the web app to start processing the audio. Once the encrypted cipher text is extracted from the audio, it is decrypted using AES 256-Bit Algorithm and the key which was supposed to be selected at that specific timestamp. Once decryption succeeds, the extracted information is cross-verified from the authenticating user. If verification succeeds, the user is prompted to the next page else they are asked to repeat the process until they succeed.

6 EVALUATION METRICS

In order to evaluate the accuracy of this product, we evaluated it against three different metrics. First, it needs to successfully transfer the code through sound in a quiet environment, in an environment with moderate/normal noise, and in a noisy environment. In addition to the level of noise, the distance of the second device(phone) from the first device(laptop) needs to be evaluated. Lastly, a check needs to be made to see if it works with no audio, the correct audio, and the incorrect audio.

To begin our evaluation, the product was tested in 6 different environments to test the quiet/moderate noise level metric. These environments include:

(1) Library (3 trials) -> quiet level
(2) Apartment (3 trials) -> moderately quiet level, few people in it
(3) Classroom (3 trials) -> moderately noisy level, students talked in regular voices
(4) Computer Science building at UIUC (3 trials) -> moderately noisy level, many students talked in regular voices
(5) Restaurant (3 trials) -> moderately noisy level, not overly busy, but some people talked in regular voices
(6) Room with music (3 trials) -> moderately noisy level, music was not set to be too loud

After running 3 test trials in 6 different environments, the 36 trials resulted in a 95% accuracy of sending the code in.

The next set of tests was run in 6 different noisy environments to test the noisy metric. These include the following:

(1) In a park (3 trials) -> noisy level, with many children in loud voices
(2) Apartment (3 trials) -> noisy level, many people(20) inside
(3) Classroom (3 trials) -> noisy level, many students talked in regular voices
(4) Computer Science building at UIUC (3 trials) -> noisy level, many students talked in regular voices during poster presentation

(5) Restaurant (3 trials) -> very noisy level, very busy and crowded restaurant
(6) Room with loud music (3 trials) -> very noisy level, music was set to be really loud

After running 3 test trials in 6 different environments, the 36 trials resulted in a 85% accuracy of sending the code in. This is likely caused due to the different audio sounds getting mixed up. When this happens, the algorithm gets confused on which sound it's hearing.

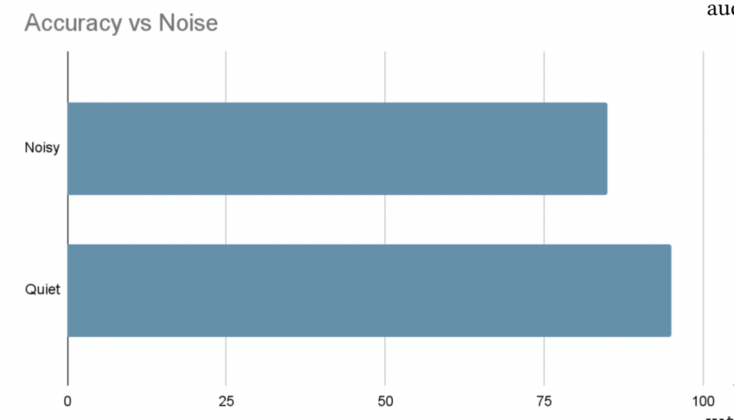

Figure 6: Accuracy vs Noise Evaluation

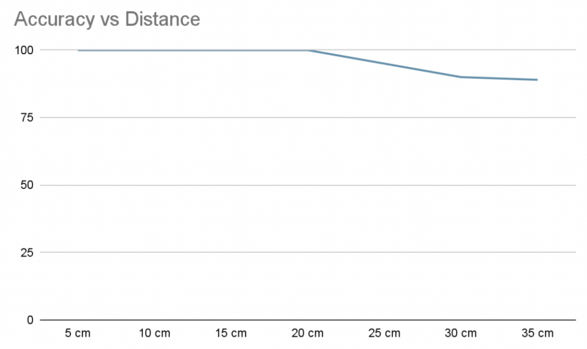

Figure 7: Accuracy vs Distance Evaluation

The next set of tests that this project was evaluated on is the distance. The goal of this project is to ensure the safety of the 2FA authentication. A second device that's doing authenticating should always be authenticating the first device from a distance that is close to the first device. This was one of the drawbacks of Duo Mobile that this project improves upon. Duo allows you to approve an authentication request from large distances. However, since this reduces the safety in guaranteeing that the second device really is supposed to approve the message, this 2FA method only works from a short distance. So if audio is played at a large distance, it should be less likely that the first device hears it and that the code is transferred.

In order to evaluate this, we measured the performance of the project in 5 cm increments from 5 cm all the way up till 35 cm(5 cm, 10 cm, 15 cm, 20 cm, 25 cm, 30 cm, 35 cm) and found that there was a slow decrease of accuracy which is exactly what was to be expected. The following figure presents the accuracy results. At each distance, we performed 5 trials to find the accuracy at that distance.

Our last set of tests was to check if this project worked produced the expected result with no audio, the correct audio, and an incorrect audio.

(1) When no audio is played to the first device, the web app does nothing and it proceeds to listen. Which is exactly as expected.
(2) When the correct audio is played to the device, the web app correctly identifies the audio with the accuracy's described above.
(3) When an incorrect audio is played to the first device, the web app never identifies it as a valid audio 100% of the time. This was tested on 10 different trials and found to be performing as expected each time.

With these tests, this project worked fairly well. The solution worked with a very high accuracy in quiet and moderately noisy environments. However, this product could still need improvements to make it work better in noisy environments. One solution for the future could be to apply methods to filter out the background noise. When evaluated with distance, it correctly followed the downward trend needed to ensure safety for 2FA.

7 FUTURE WORK

There are a few different categories of this project that could be improved or added to. These include improvements on the noise accuracy, using a method of ultrasound instead of audible audio, and making the device contactless.

7.1 Noise detection Accuracy

One limitation of this project is its performance in a very noisy environment. The results from the evaluation resulted in an 85% accuracy. Future work could focus on creating a filter that separates the background noise from the specific type of noise used for this method. Another solution could be to use ultrasound where this issue could be eliminated.

7.2 Use of ultrasound

While the use of audible sound allowed this method in our use cases to work fairly, the use of ultrasound could eliminate a lot of the limitations of this project. Future work could focus on replacing the audible audio with ultrasound instead. This could eliminate the issue mentioned above so that the background noise won't effect the sound that is played. It would also be more practical to use in day-to-day situations because the sound that's played will not be heard by other people. Ultrasound reduces the drawbacks while maintaining the advantages of this method.

7.3 Creating a contactless system

Currently, 2FA systems require some level of contact with the second device when providing authentication. While this method works on creating as little of a tedious process as possible, the user still needs to click a button on the mobile app in order for the audio to be played. This can be tedious, especially when done several times a day. One solution that followed is adding a Bluetooth connection so that the sound can be automatically produced by the second device. Once a Bluetooth connection is set up, the first device could send a push notification to the second device which would cause the second device to play an audio. In order for the Bluetooth connection to work, the mobile app would need additional permissions set up.

REFERENCES

[1] Jessica Colnago, Summer Devlin, Maggie Oates, Chelse Swoopes, Lujo Bauer, Lorrie Cranor, and Nicolas Christin. 2018. "It's Not Actually That Horrible": Exploring Adoption of Two-Factor Authentication at a University. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3173574.3174030

[2] Georgi Gerganov. 2020. ggwave. https://github.com/ggerganov/ggwave.

[3] Nikolaos Karapanos, Claudio Marforio, Claudio Soriente, and Srdjan Capkun. 2015. Sound-Proof: Usable Two-Factor Authentication Based on Ambient Sound. *Proceedings of the 24th USENIX Security Symposium (USENIX Security '15)* (03 2015).

[4] Dan Liu, Qian Wang, Man Zhou, Peipei Jiang, Qi Li, Chao Shen, and Cong Wang. 2022. SoundID: Securing Mobile Two-Factor Authentication via Acoustic Signals. *IEEE Transactions on Dependable and Secure Computing* (2022), 1–1. https://doi.org/10.1109/TDSC.2022.3162718

[5] Martyn Riley and Iain Richardson. 1998. An introduction to Reed-Solomon codes: principles, architecture and implementation. https://www.cs.cmu.edu/~guyb/realworld/reedsolomon/reed_solomon_codes.html. (1998).