Exploiting Sound Masking for Audio Privacy in Smartphones

Yu-Chih Tung
The University of Michigan
yctung@umich.edu

Kang G. Shin
The University of Michigan
kgshin@umich.edu

ABSTRACT

Privacy leakage via malware's unauthorized audio recording has become an emerging threat to mobile users. To counter this threat, we propose SafeChat which prevents unauthorized recording without requiring new audio privacy settings in operating systems. SafeChat utilizes sound masking to differentiate audio information between authorized and unauthorized recording apps. Specifically, SafeChat enables authorized recording apps to recover more private/secret information than the unauthorized apps even though both of them record identical audio signals from the same microphone. We have implemented SafeChat as an Android chat app. Our experiments on several commodity phones have shown that SafeChat can make an up-to-26dB difference in signal strength between authorized and unauthorized recording apps. This difference reduces the accuracy of state-of-the-art speech recognition engines, like Google Speech API, to less than 0.1% in understanding the unauthorized recording while comprehending the authorized recording with high accuracy. Moreover, none of the 317 testing participants we recruited online could comprehend the masked speech. Our usability study shows that only 35% of the participants were aware of this threat of privacy leakage and 60% of them wanted to use SafeChat to protect their private/secret information from unauthorized recording.

CCS CONCEPTS

• **Security and privacy** → **Mobile platform security**; *Privacy-preserving protocols*; • **Networks** → **Mobile and wireless security**.

KEYWORDS

Audio privacy; sound masking; mobile systems

ACM Reference Format:
Yu-Chih Tung and Kang G. Shin. 2019. Exploiting Sound Masking for Audio Privacy in Smartphones. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS '19), July 9–12, 2019, Auckland, New Zealand.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3321705.3329799

AsiaCCS '19, July 9–12, 2019, Auckland, New Zealand
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6752-3/19/07…$15.00
https://doi.org/10.1145/3321705.3329799

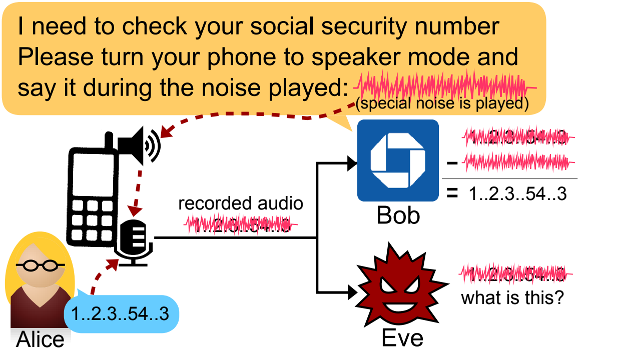

I need to check your social security number
Please turn your phone to speaker mode and
say it during the noise played:
(special noise is played)
recorded audio
1..2.3..54..3
–
= 1..2.3..54..3
Bob
1..2.3..54..3
Alice
what is this?
Eve

Figure 1: Private/secret information is protected by sound masking. Audio malware installed in a user's phone is unable to comprehend the sniffed secret because that message is obfuscated by the noise generated and added by the intended receiver.

1 INTRODUCTION

Privacy information leakage through phone microphones has become a critical concern to mobile users. Spyware that records conversations stealthily in the background is gaining popularity in app markets [4, 6, 8]. It has also been shown that a malware can easily recognize the users' credit card numbers by using state-of-the-art speech recognition engines [24]. This threat becomes even more alarming when we consider the possibility of the same exploitation embedded in numerous apps which have already acquired the audio access permission from users. To counter this threat, we propose SafeChat, a novel way of preventing privacy information leakage via *unauthorized audio recording* with phone microphones. Specifically, SafeChat focuses on a common threat scenario shown in Fig. 1, where a user (Alice) is telling her secret information to a service provider (Bob), over the phone, while the malware (Eve) is recording and then leaking the recorded information stealthily.

Conventional solutions to this problem feed fake audio data to apps or block audio access under a fine-grained sensor privacy policy [13, 16, 17, 22, 29]. These solutions, however, require a major modification of the phone permission control system and also a non-trivial effort for users to select the *correct* permission/privacy policy. Moreover, the service provider (usually the party requesting secret information) still has no way to tell whether the user has such a security function enabled with the expected privacy policy setting. For example, the service provider cannot determine which apps on the user's phone have access to the device's microphone, and whether all those apps are trustworthy.

Our solution, SafeChat, prevents the privacy leakage via unauthorized recording without requiring any additional audio privacy/permission control, and lets both the user and the service provider invoke this security feature, whenever necessary. SafeChat utilizes *sound masking* to achieve the above-mentioned protection. Fig. 1 shows an example of a bank representative, Bob, requesting private information from a customer, Alice, over the phone. In this case, Bob first asks Alice to change the phone to the speaker mode and say the requested information while playing the masking sound/noise (this operation can be handled automatically by SafeChat as described later). The phone's microphone will record the combination of the noise played by Bob and the private information spoken by Alice. The private information is thus obfuscated by this superpositioned noise and can only be recovered by Bob. The masking sound cannot be removed by unauthorized apps because only Bob knows how this noise is generated and added.

SafeChat is inspired by (i) the existing solutions' need for an additional user-defined policy and (ii) the fact that both authorized and unauthorized recording apps receive the same copy of recorded audio and the distinction between secret and normal conversations is not well-defined in apps. SafeChat solves this problem by moving privacy protection from a mobile device to the service provider (henceforth called the *intended receiver*). This design offers two distinct benefits. First, the privacy protection can be computed remotely, thus making SafeChat backward-compatible (see Sec. 5 & 7 for the supported devices and system requirements of SafeChat). Second, the intended receiver can initiate a secure chatting channel,[1] whenever necessary. For example, users can still use call recording apps to save most of their conversations. But the intended receiver can enable privacy protection whenever he asks for private/secret information from users, and only this part of conversation is obfuscated by SafeChat from the call recording apps.

There are several challenges to overcome for the realization of SafeChat. One of them is that the masking sound recorded by the microphone is not exactly identical to the originally-generated audio signal due to the multipath and resonant properties of audio transmissions as well as the device speaker/microphone's distortion. Thus, it is necessary to design a proper sound masking (noise) signal and its removal algorithm that can effectively obfuscate the private information while enabling the intended receiver to remove the masking sound. We have implemented SafeChat as an Android chat app to validate its functionality. We assume the communication is established via our app, and the secure masking sound can be played/removed when a specific (soft) button on the app is pressed. Implementing SafeChat in other forms, such as a third-party library for existing chat apps or as a system service to support common telecommunications, is part of our future work.

Our experimental results on several Android devices have shown that the added masking noise in unauthorized recording can be up to 26dB higher than the recovered recording at the intended receiver. Assuming malware can easily identify the recorded secrets with speech recognition engines or crowdsourcing [24], SafeChat can lower the word accuracy of an 8-digit sniffed secret from 99% to 0.1% when the state-of-the-art Google Speech API [5] is used for its recognition. Our signal removal algorithm can restore this recognition accuracy back to 95%, thus creating a significant difference of information comprehended by the intended receivers and the unauthorized recording apps. Moreover, when 317 participants recruited online were asked "to identify numbers hidden in the noise," none of them could fully recognize the masked secrets/numbers while the intended receivers could recover them correctly. The effect of possible preprocessing/filtering for attackers to remove this masking sound is also discussed in Sec. 4 and 7.

This paper makes the following 4 main contributions:

- Design of app-level protection on Android devices to provide audio privacy based on sound masking [Sec. 3];
- Security analysis of sound masking on mobile devices for preventing unauthorized recording [Sec. 4];
- Extensive evaluation showing that Google Speech API and 317 testing participants comprehend authorized recordings with higher than 95% accuracy while the unauthorized recordings were incomprehensible [Sec. 5]; and
- Measuring the usability and demonstrating the real-world benefits of SafeChat [Sec. 6].

2 RELATED WORK

Privacy leakage via a phone's microphone, known as *unauthorized recording*, has become an emerging threat to smartphone users because any installed app with audio access permission can stealthily record any information at any time. Malware exploiting this leakage can easily collect users' credit card information and social security numbers [24]. Existing defenses against this unauthorized recording are usually designed by feeding fake audio data to apps or by disabling the recording function according to a pre-defined sensor privacy policy. For example, some systems [17, 29] suggest always feeding fabricated audio data to apps when telecommunication is on, while other systems [22, 31] create a lattice-like privacy table to represent conflicts between apps and disabling of the audio recording function at a proper time. A programmable sensor privacy policy was proposed in [16] so that a trusted third party may design a proper policy for mobile users to control the audio recording. In general, however, it is difficult to define a "proper" privacy policy. For example, how can a policy enable the phone call recording app while preventing private information leakage to the same app? Moreover, all these solutions require major modifications to the existing privacy control system, thus making them less likely deployed in commodity mobile devices.

SafeChat is an app-level solution that preserves audio privacy with sound masking without requiring any modification to existing systems. *Sound masking* is an audio obfuscation to prevent privacy leakage via wiretapping or nearby devices' eavesdropping [28]. Traditional sound masking is designed under the assumption that the audio recorded in the sniffing devices has lower quality or signal strength (due to farther distances to, or the distortion of wiretapping equipment), hence generating enough noise in the background to prevent eavesdroppers from comprehending the spoken information while preserving its comprehension by the intended receiver. Sound masking has recently been used to build a wall-less but privacy-reserved office by beamforming noise from a speaker array installed on the ceiling [3, 7]. However, this use-case is not applicable to mobile scenarios because both the installed malware and the

[1] Dual-tone multi-frequency (DTMF) signaling is another way to send private information via dial tones, but it is also vulnerable to unauthorized recording [26].

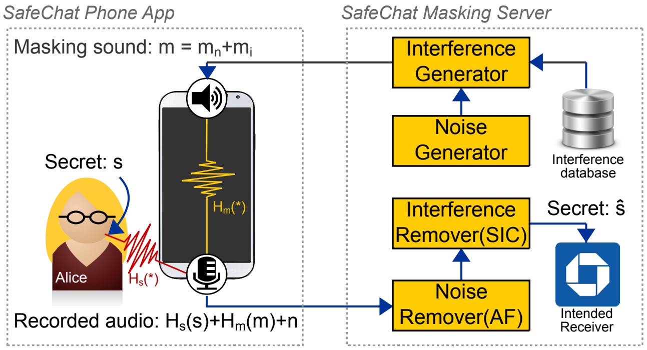


Figure 2: System overview. Masking sound, m, and secret information, s, are sent via different paths, $H_s(*)$ and $H_m(*)$, to microphones, so it requires a special signal processing to remove the masking sound from the recorded audio at the intended receiver.

intended receiver record an identical signal from the same phone microphone, and hence adding noise will obfuscate both. SafeChat solves this problem by generating a special masking sound and then removing it later at the intended receiver, thus achieving app-level protection, which is akin to providing differential audio privacy between unauthorized and authorized apps.

The closest to SafeChat is mSieve [23], which relies on differential privacy providing more audio information to certain apps than to others. However, It did not address how to process the signal to derive the mathematical model, and it also requires system modifications for pre-processing audio signals before the app accesses them. Non-cryptographic security solutions in wireless networks are also related to SafeChat. For example, transmitters can generate an artificial noise at the null space of channel state information (CSI) to the intended receiver [11, 14], so that eavesdroppers located at different positions from the intended receiver will be unable to remove the added noise and decode the sniffed packets correctly. The intended receiver can also broadcast an artificial noise during the receipt of packets and recover the secret by a CSI-based signal removal process [27]. Even though the concepts of these wireless security solutions are similar to SafeChat, they can't solve the unauthorized recording problem directly. For example, human voice is not modulated/precoded as wireless signals, and hence there is no OFDM preamble that can be used to estimate CSI. In contrast, SafeChat uses a novel way of estimating the channel response from the speaker to the microphone by adaptive filtering and then removing the residual added noise by successive interference cancellation (SIC) [25]. Other researchers also explored ways of utilizing acoustic signals to build secure digital channels, e.g., Dhwani [20] and PriWhisper [30]. SafeChat shared similarities to these systems but aims at a different problem: project sensitive conversions rather than encoding secret information in the sound.

3 SYSTEM DESIGN

Fig. 2 presents an overview of SafeChat. When it is enabled to transmit the secret information, s, a masking sound, m, is played by the device's speaker. Since the sound is broadcast over the air and travels through multiple paths, the recorded sound is actually a combination of multiple delayed and attenuated copies of the originally-played sound. Suppose this combination is linear, then a channel response, $H(*)$, can be used to represent how the sound is combined at the microphone for recording, i.e., $recorded(sound) = H(played(sound))$. We will use $H_s(*)$ to denote the channel response of the secret information spoken by Alice, while the channel response of the played masking sound is denoted by $H_m(*)$. Thus, the audio recorded, r, becomes

$$recorded(sound) = r = H_s(s) + H_m(m) + n, \qquad (1)$$

where n is a Gaussian environmental noise. SafeChat's effectiveness in neutralizing unauthorized recording depends on the selection of masking sounds. For example, sending the masking sound as an audio including millions of spoken sentences from different people should be able to hide the secret effectively because the listener won't be able to tell which sentence is the secret. However, this design will also be problematic to the intended receiver since the intended receiver doesn't know how "distorted" the masking sound is in the recorded audio and how to remove the distortion source(s). Moreover, if the attacker knows the proper context of the user's voice (like pitch), the secret in the chat might still be extracted by a specially-designed filter. Thus, designing a proper masking sound and ensuring its removal at the intended receiver are critical for SafeChat to function effectively.

3.1 Selection of Masking Sound

SafeChat chooses the masking sound as a combination of two signal components, i.e., $m = m_i + m_n$. The first component is the masking interference, m_i, which includes several pre-recorded human-spoken sentences to confuse and prevent the malware from extracting the secret information. Since our main goal is to protect the secret information like credit card or social security numbers, we choose this interference from a sound database including audio clips of digits, called TIDIGITS [18]. Note that this masking interference can be chosen from other sources (can be recorded by Alice as well) to improve system performance further, which is part of our future work.

The second component is the masking noise, m_n, which is generated as a Gaussian noise and filtered out by a 16kHz low-pass filter. SafeChat only keeps the frequencies of noise below 16kHz because that frequency range covers most human speech spectrums. Adding this Gaussian noise helps reduce the SNR of secret information, which corresponds to the speech intelligence of audio recorded by malware [10]. Moreover, it also helps the intended receiver recover the secret and avoid the filter-based separation of masking sounds, e.g., keeping only the sounds in the user's pitch range.

Note that in our implementation, a pilot signal, composed of several 10k–24kHz chirps, is played before the masking sound. This pilot is designed to help synchronize the time offset between the device speaker and microphones because there is a delay of a few hundred milliseconds between a program asking to play a sound and actually playing that sound (owing to the non-real-time OS of commodity phones). Without this synchronization, a larger depth needs to be set in the adaptive filter for characterizing the channel response, which incurs a 100x more computation time.

3.2 Removal of Masking Noise and Interference

The effectiveness of removing the noise and the interference at the intended receivers is vital to the obfuscation of secret information. Unlike the CSI-based techniques widely used in wireless systems to remove the added noise [14, 27], it is non-trivial to remove the masking sounds in smartphones due to the ignorance of channel responses, $H_s(*)$ and $H_m(*)$.

To solve this problem, SafeChat uses the masking noise component to estimate the channel response $H_m(*)$. Specifically, we treat the masking interference as part of secret information and use an existing adaptive filter [15] to separate the masking noise from the other signals. In our current setting, the depth of this adaptive filter is set to 500 samples to handle the sound delay spreads on different devices. This process can determine the best estimation of channel response, $\hat{H_m}(*)$, by minimizing:

$$e(\hat{H_m}(*)) = r - \hat{H_m}(m_n), \quad (2)$$

where $e(\hat{H_m}(*))$ represents the residual error of adaptive filtering under the assumption that the added noise has no correlation with the human-spoken sounds. Note that it is impossible to isolate the masking interference component with the same adaptive filter because the added human-spoken sounds have non-zero correlation with the secret (and also human-spoken) information.

According to the theory of adaptive filtering, the residual error, $e(\hat{H_m}(*))$, represents a combination of secret information, s, and the added masking interference, m_i. SafeChat then applies the successive interference cancellation (SIC) to recover the secret information by removing $\hat{H_m}(m_i)$ from the residual noise:

$$\hat{s} = e(\hat{H_m}(*)) - \hat{H_m}(m_i), \quad (3)$$

where $\hat{s}$ is the recovered secret at the intended receivers. Note that the eavesdropping apps cannot apply the same processing to extract the secret information because the masking sounds are generated by, and only known to, the intended receiver. Fig. 3 shows an example of this noise and interference removal process for the masked audio recorded by Nexus 6P. In this example, more than 15dB of the masking noise and interference was removed from the recorded audio, thus generating an enough gap of recorded information between authorized and unauthorized recordings.

3.3 Sound Masking Metrics

Fig. 4 shows an example of energy envelopes where a 3-digit secret is masked/recovered by SafeChat, as well as the definition of 4 important sound masking metrics. We will henceforth use the Masking sound to Noise Ratio (MNR) and Masking sound to Residual noise Ratio (MRR) to represent the intensity of added masking sound and the effectiveness of removing this masking sound, respectively. The former (latter) is defined as $||r||/||n||$ ($||r||/||\hat{s}_{nonspeech}||$), where the noise term, n, represents the background (not the masking) noise and $\hat{s}_{nonspeech}$ denotes the residual noise in the recovered signals without the user's speech.

On the other hand, the Masking sound to Speech Ratio (MSR) and Speech to Recovered noise Ratio (SRR) represent the amount of secret information hidden from the malware and recovered at the intended receiver, respectively. The former is defined as $||m||/||\hat{s}_{speech}||$, while the latter is defined as $||\hat{s}_{speech}||/||\hat{s}_{nonspeech}||$. The sections of speech/non-speech signals can be identified by standard Voice Activity Detection (VAD) algorithms, like G.729 [12]. To ensure a low processing delay of SafeChat, we only implemented a simple VAD based on energy thresholding. Specifically, we used the 80-th and 20-th percentiles of energy envelop of $\hat{s}$ to represent $||\hat{s}_{speech}||$ and $||\hat{s}_{nonspeech}||$, respectively. As shown in Fig. 4, this simplification makes $\hat{s}_{speech}$ dominated by the loudest part of the spoken secret, reflecting the purpose of SafeChat to prevent privacy leakage of the entire spoken secret. This simplification might slightly over-estimate MRR in a few cases when there is no speech information recorded. This issue will not harm our evaluation much because most measurements include the spoken secrets and the final performance of SafeChat is characterized by the difference of recognition accuracy between masked and recovered secrets. Thus, without loss of generality and to make the threshold settings consistent, we choose to report MRR based on this setting.

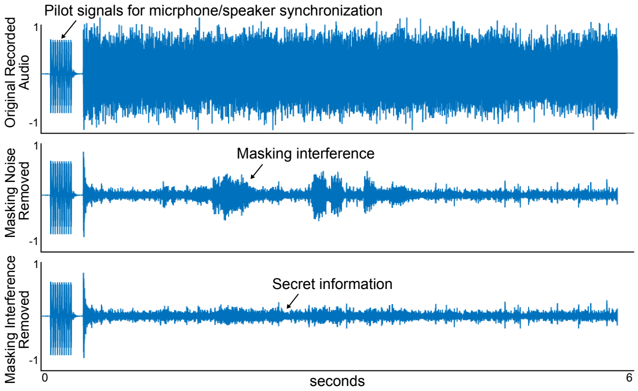
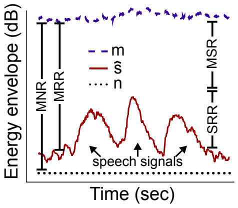


Figure 3: Example of masking sound removal. The masking noise is first removed by an adaptive filter and the masking interference is then removed via successive interference cancellation.

In summary, MNR and MRR describe the device's hardware capability of sound masking, i.e., not relevant to $\hat{s}_{speech}$, while MSR and SRR capture the SafeChat's performance in defending against unauthorized recording. Ideally, SafeChat achieves the best performance when the value of these four metrics are as high as possible. However, these metrics implicate each other. For example, a high SRR implies a low MSR because the amount of removed masking sound is limited. The details of our design choices to find a balance between these metrics will be presented next.

3.4 Device Volume Control

Speech intelligence is shown to correspond to the Signal-to-Noise Ratio (SNR) of the spoken voice [10], which is negatively related to the volume of played masking sounds. Thus, the masking sound should be played loud enough (i.e., high MNR) to lower the speech intelligence of secret information. However, the masking sound cannot be played with an infinitely large volume due to hardware limitation. It is also unwise to play the masking sound too loud because the assumption of linear channel response becomes invalid



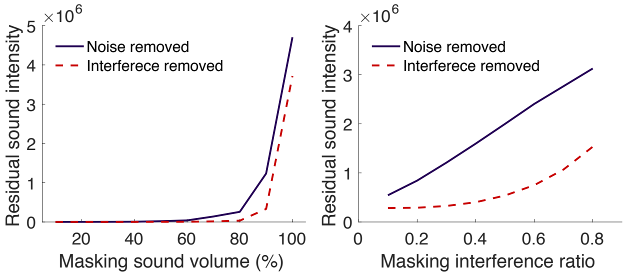
Metric	Description
MNR	Masking sound to noise ratio
MRR	Masking sound to residual noise ratio
MSR	Masking sound to speech signal ratio
SRR	Speech signal to residual noise ratio

Figure 4: Explanation of sound masking metrics. MNR and MRR represent the device capability to play/remove the masking sound while MSR and SRR indicate how much of secret information is leaked/received to malware and the intended receiver.

when the played/recorded sound swings into the non-linear region of phone speaker/microphone. On the other hand, playing the masking sound with a low volume not only fails to hide the secret information, but also leaves more residual noise in our masking sound removal process due to the inaccurate estimation of channel response.

Fig. 5(a) shows the average sound intensity when the masking sound is played with different speaker volumes and then removed on Nexus 6P. In this example, audio is recorded in a quiet environment without including any spoken speech, and thus the residual sound intensity shown in Fig. 5(a) represents the residual errors of our noise and interference removal algorithm. As shown in this figure, when the sound is played with larger than 90% of the maximum volume, the residual error of the adaptive filter surges due to the non-linear distortion, thus hindering the recovery process in the presence of such a high-volume masking sound.

Besides the selection of an optimal speaker volume to play the entire masking sound, we also need to find the balance of sharing this finite volume budget to mask noise and interference. It is meaningless to play a masking interference with a lower volume than that of recorded secret information. However, using too high a volume to play the masking interference increases the residual error of its removal with SIC, thus making it difficult for the intended receiver to comprehend the recovered audio. This is a common issue of removing signals with SIC, where the system achieves the optimal performance when the signals removed first (i.e., masking noise) have a greater signal strength than those to be removed later (i.e., masking interference) [25]. Fig. 5(b) shows an example of fixing the device speaker volume at 90% of the maximum volume while varying the ratio of volume to play the masking interference. As shown in this figure, a high ratio of volume to play the masking interference not only leaves more residual energy of adaptive filtering but also increases the residual error of SIC due to the inaccurate estimation of channel responses. In our current design, the energy ratio of masking interference is always fixed at 10dB lower than the volume of masking noise no matter how large the speaker volume is. This selection is based on our experimental findings and the assumption that the energy of secret speech is 13dB lower than the played masking sound. The performance of this masking interference setting will be detailed in Section 5.



(a) Noise Volume (b) Interference Volume

Figure 5: Effectiveness of sound masking. Larger the volume, more accurate the estimation of channel response. However, the channel response becomes non-linear when the masking sound is played with maximum volume, thus leaving large residual errors.

Algorithm 1 Device Calibration

```
Input: recorded audio: $r(vol, ch)$ and a $MRR$ threshold: $thr_{MRR}$
Output: microphone channel and speaker volume: $ch_{calib}$, $vol_{calib}$
$n = estimate\_background\_noise$(r)
$\hat{s}(vol, ch) = remove\_masking\_sound(r(vol, ch))$
$MNR(vol, ch) = 20log10(\|r(vol, ch)\|/\|n\|)$
$MRR(vol, ch) = 20log10(\|r(vol, ch)\|/\|\hat{s}_{nonspeech}(vol, ch)\|)$
$minMNR = inf$ and $maxMRR = -inf$
for $ch$ for all microphone channels do
    $meanMNR = mean(MNR(:, ch))$
    if $minMNR > meanMNR$ then
        $ch_{calib} = ch$
        $minMNR = meanMNR$
for $vol$ for all volumes do
    if $maxMRR < MRR(vol, ch_{calib})$ then
        $vol_{calib} = vol$
        $maxMRR = MRR(vol, ch_{calib})$
if $maxMRR < thr_{MRR}$ then
    return fail
for $vol$ from the largest volume to the smallest volume do
    if $|maxMRR - MRR(vol, ch_{calib})| < 1dB$ then
        $vol_{calib} = vol$
        $break$
return $ch_{calib}$ & $vol_{calib}$
```

3.5 Device Calibration

As mentioned earlier, SafeChat needs to find a reasonable volume to play masking sound and a proper ratio of volume to play the masking interference. It is also necessary to find a proper microphone as a reference to recover the secret information. In our experiments, due to different microphones' positions, the recorded signal strength of masking sound in one microphone could be 20dB higher than the other microphone. Thus, SafeChat makes one-time calibration to search for the best microphone as a reference and also the best speaker volume to play masking sound as shown in Algorithm 1.

In this one-time calibration, SafeChat automatically plays the masking sound with different speaker volumes, *vol*, and records it by different device microphones, *ch*. The recorded signals with different volume settings, $r(vol, ch)$, are first processed by our masking sound removal algorithm to recover the secret information, $\hat{s}(vol, ch)$. We then estimate both MNR and MRR for all volume and channel settings. In the calibration process, the reference microphone used to recover the secret is selected based on MNR. Specifically, SafeChat always chooses the microphone with the lowest MNR as the reference, because the low MNR implies that the microphone receives less masking sounds (due to the position farther away from the speaker or different microphone gain settings). It is necessary for SafeChat to ensure the protection of this "weakest" microphone by receiving strong enough masking sounds. In our experiments, when the masking sound volume is tuned properly to protect this reference microphone, it naturally saturates the other microphone on the mobile device.

Once the microphone channel, ch_{clib}, to the reference is determined, SafeChat will look for a volume with the highest MRR, which represents SafeChat's capability of removing the masking sound effectively. SafeChat sets a threshold for this maximum MRR to pass. This threshold is set to 18dB based on our current testing of devices. The calibration fails if it is unable to find any masking sound volume that meets this condition. Note that the volume selection based on the highest MRR is to ensure the recovery of secret. However, the main goal of SafeChat is still to prevent unauthorized recording, so our calibration process aggressively searches for a higher volume, if any, with a similar MRR (i.e., 1dB less than the maximum MRR), and sets that volume as the calibrated volume vol_{calib}. The results of this calibration process for different devices will be presented in Section 5.

3.6 Speech Volume Check

Besides the device's capability of playing and removing masking sounds, the user's speaking volume is also critical to the system performance. If the secret is spoken too loudly, the selected masking sound might not be able to obfuscate the secret effectively. On the other hand, too low a speaking volume might make the intended receiver unable to comprehend the recovered secret.

One possible way to solve this problem is to dynamically adapt the masking sound volume to the user's speaking volume. However, as explained above, the masking sound should not be too loud in order not to sway into the non-linear range of microphone/speaker hardware. Moreover, based on our preliminary experimental results, users tend to increase their sound volume when the volume of masking sound is raised, thus canceling the effect of adapting the masking sound volume dynamically. Currently, SafeChat fixes the masking sound volume via one-time device calibration and adds a user training phase before the secret is spoken. The purpose of this training phase is to ensure a proper speaking volume that splits the limited MRR budget to guarantee a high masking sound to speech ratio (MSR) and a high speech to recovered noise ratio (SRR).

In the training phase, users are asked to speak a 3-digit number for checking their speaking volume. Users are considered to have succeeded in this training when the MSR of this training recording is higher than 13dB and SRR is higher than 3dB. This setting ensures the energy of human's speech is 13dB lower than the masking sound. In our usability study, users needed only an average of 1.6 rounds to pass the training criteria. This number decreases to 1.3 when users are asked to try the same training task again 5 minutes later. Considering each training takes about 5–6 seconds, the whole training time would be less than 10 seconds. Most testing participants said that it is easy to keep their voice compatible with our current setting. The details of this usability study, like how users can keep their voice in the safe range after the training, will be presented in Section 6.

4 SECURITY ANALYSIS

While SafeChat is designed as an app-level prevention of audio privacy leakage, there still exist several requirements to make SafeChat work in a realistic setting. This section will introduce those requirements, the targeting threat model, and also a potential attack on SafeChat.

4.1 Threat Model

The purpose of SafeChat is to prevent audio privacy leakage through malware's unauthorized recording. Even though the sound masking may prevent the privacy leakage through other channels (e.g., wiretapping), they are beyond the scope of this paper, and hence not the focus of our current SafeChat design.

SafeChat assumes the malware and our installed app have the same capability and permissions to access device microphones and speakers. While SafeChat need not modify the device OS, it requires the user's device not to be rooted nor compromised. This requirement is necessary because a compromised OS could help the malware extract the masking sound through the speaker's audio chain and then remove it easily by the removal process introduced earlier. This is a reasonable assumption because other existing audio privacy systems [13, 16, 17, 22, 29] will also be breached easily when the device OS is compromised.

After recording the audio stealthily, we assume malware can easily recognize secret by the speech recognition engines or crowdsourcing like the malware *Soundcomber* [24]. We also assume that the speech recognition engines used by malware are equipped with noise-handling mechanisms and malware has the knowledge of common audio pre-processing, such as source separation, to facilitate their attack on the masked secret. The security provided by SafeChat is characterized as the probability of the masked secret being recovered by malware under the above assumptions.

4.2 Security Guarantee

Unlike other cryptography-based security systems which can provide an exact security guarantee like the estimated time to break the system, it is less likely for SafeChat to provide such a guarantee. This is nonetheless a common issue for non-cryptographic systems, such as Wyner's wiretap channel [28] or other physical security wireless systems. For example, most wireless security systems that "mask" packets by sending artificial noise [11, 14] model their security guarantee by ensuring the SNR of packets received at eavesdroppers is less than the code/modulation capacity of packets, and hence it is information-theoretically impossible for eavesdroppers to recover the packets received with masking noise.

We follow the same line of analysis to estimate the security guarantee SafeChat can provide. Specifically, we evaluate the SNR difference of private/secret information between the authorized and unauthorized recordings, i.e., MSR in our measurements. However, no "code/modulation capacity" guarantee exists for SafeChat because human speech is not modulated in the same way as wireless packets. For example, there usually exists redundant information in the speech signals so that machines can comprehend human's speech by just analyzing statistical features like MFCC or polynomial residual error. Thus, we still don't know how large the MSR should be.

In this paper, we evaluate SafeChat's privacy/secret protection based on the fact that the probability of guessing a secret x is not different from knowing its encrypted secret y, i.e., $P(x|y) = P(x)$. Assuming SafeChat knows the most powerful speech engines or crowdsourcing resources that malware can use to recover secrets, we can define the security of SafeChat by answering the following question: "What is the difference between the probability of malware recognizing the masked secret from that of a pure random guess?" SafeChat achieves the secrecy if the malware is shown unable to use the assumed methodology to make a better guess of masked secrets than a random guess. Note that SafeChat delegates the noise handling to the speech recognition engines and crowdsourcing. We will also discuss a few other potential pre-processing methods that might help attackers uncover secrets, but SafeChat is resilient to them as well, as discussed below and in Section 7.

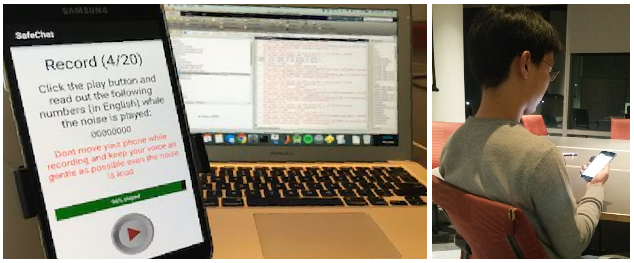
5 EVALUATION

We have implemented SafeChat as a chat app in Android and a remote sound masking server in Matlab. Implementing SafeChat as an app helps automatically turn the device to speaker mode and calibrate the device and speech volume, if needed. Note that this design doesn't sacrifice SafeChat's benefits much. A bank representative (Bob) can still talk securely to their customers (Alice) via the installed chat app, not via the normal telecommunication. Asking customers to install a chat app is also much easier than asking the phone manufacturer to update the latest audio privacy security feature, if any, with a new system patch.

We have conducted experiments to assess the effectiveness of SafeChat. Our evaluation is designed to determine if SafeChat can reduce the speech intelligence of unauthorized recording to the extent that the malware has no better way to recover the masked secret than making a random guess. As the threat model discussed in Section 4, we focus on the scenario in which the malware has the knowledge/resource of utilizing the human's mind or machine learning to recover the secret. Without loss of generality, we chose the masked secret in the following evaluation to be an 8-digit number, and the speech intelligence is measured by the accuracy in identifying this 8-digit secret. This 8-digit number is chosen mainly because it is the longest corpus in TIDIGITS dataset. Our evaluation result should be applicable to other lengths of information, such as 6-digit iPhone passcodes or 9-digit security numbers.

5.1 Experimental Settings

Two recording datasets are collected in our experiments, where the recorded speech (secret information) was either played by a laptop speaker near the testing device or was spoken by a participant at a similar location as shown in Fig. 6. The laptop-played speech is chosen randomly from any 8-digit recording in the TIDIGITS sound dataset [18] while a random 8-digit number was read by testing participants.


SAMSUNG
SafeChat
Record (4/20)
Click the play button and read out the following numbers (in English) while the noise is played:
00000000
Dont move your phone while recording and keep your voice as gentle as possible even the noise is loud
96% played

(a) Speech by laptop speakers (b) Human speech

Figure 6: Experimental setting. Spoken audios from TIDIGITS were played by a laptop next to the phone running SafeChat. Testing participants are asked to read a randomly-generated 8-digit number.

We recruited 27 participants (17 males and 10 females) to build the human voice dataset. Six participants helped record multiple times/rounds while the others only recorded 6 speeches with the final setting of SafeChat. For example, we asked these 6 participants to intentionally vary their speaking volume even when it is outside of SafeChat's operation range. The laptop dataset eases the automation of testing SafeChat under different scenarios, such as varying masking volume, interference gain, or microphone channels, while the human dataset helps us understand the real-world performance of SafeChat. By the end of our experiment and user study, we had collected more than 1600 recordings spanning more than 3.5 hours. These recordings were later recognized by state-of-the-art Google Speech API [5] or by 317 real users recruited via Amazon Mechanical Turk [1].

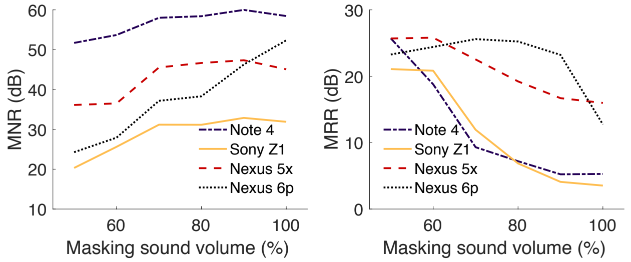
To avoid any recognition accuracy drops due to the speech artifacts of our removal algorithm rather than the masking sound, Google Speech API is only used to identify recordings of adding the masking sound recorded in the laptop dataset to a random speech from the TIDIGIT dataset. This setting is beneficial to the attack performance by assuming that the attacker has knowledge of signal pre-processing to completely remove speech artifacts. We have also tried to retrain an open-source speech recognition engine with our dataset (i.e., the masked speech), but omitted the result since the attack performance is significantly worse than using the state-of-the-art Google Speech API. Characterizing the attack performance of utilizing other speech recognition engines is part of our future work.

5.2 Effectiveness of Masking Sound and its Removal

We first evaluate the effectiveness in playing and removing the masking sound in terms of MNR and MRR. To show the different devices' capabilities, we apply our calibration process on 6 different devices, as shown in Table 1, while changing the device speaker volume from 50% to 100%. The results of adding/removing masking sound on 4 of 6 devices are shown in Fig. 7.

Device	ch_{calib}	$vol_{calib}(\%)$	$MNR(dB)$	$MRR(dB)$
Galaxy S4	2	70	35	18
Galaxy S5	1	70	32	20
Galaxy Note4	1	50	51	26
Sony Z1	1	60	24	21
Nexus 5X	1	60	38	26
Nexus 6P	1	80	39	26

Table 1: Calibrated settings of SafeChat on different devices



(a) MNR

(b) MRR

Figure 7: Effectiveness of masking sound on different devices. Some devices have better masking performance when the masking sound is played with a high speaker volume while others are unable to remove the masking sound effectively if it is played too loud. (Results of Galaxy S4/5 are omitted for readability.)

During the calibration, we found all tested devices can generate higher than 30dB MNR with their maximum speaker volumes, indicating that the speaker is loud enough to hide speech under proper settings. Regarding the effectiveness of masking sound removal, most devices showed better removal performance in the 50–80% speaker range, where the 50% speaker volume can usually achieve more than 20dB MRR among the devices we tested. When the volume is set based on our calibration algorithm, SafeChat was able to remove up to 26dB masking sound from the recorded audios.

Note that the calibration behavior varies with device. For example, as shown in Fig. 7(a), when the MNR of Note 4, Nexus 5X, and Sony Z1 stop increasing when the speaker volume increased over 80%, Nexus 6P can still have 10dB higher MNR when the masking sound is played with a full volume. This phenomenon might have been caused by different dynamic ranges of microphones and speakers equipped in the device. However, even though Nexus 6P seems to have a high enough dynamic range to play masking sound with a full volume, as mentioned before, playing sound with a full volume causes non-linear distortion and microphone saturations which prevents SafeChat from removing the masking sound effectively. For example, as shown in Fig. 7(b), the MRR of Nexus 6P drops from 26dB to 13dB when the masking sound volume increases from 80% to 100%. The other devices follow the same behavior, so our calibration algorithm sets their speaker volumes between 50% to 80% for making the best balance between MNR and MSR. In our experiment with the speech played by a nearby laptop, Nexus 6P was capable of hiding the highest laptop speaker volume while Galaxy S4 was the one only capable of hiding speech when it was played with 70% of the laptop speaker's volume. The details of our calibration setting are summarized in Table 1.

5.3 Robustness of Masking Sound Removal

User movements and ambient noise might affect our masking sound removal process. For example, the MRR of Nexus 6P was found to drop from 26dB to 23dB when users changed the position from sitting to walking, but the MRR remained high enough to support SafeChat. The reason for this is that the estimated sound response, $H_s(*)$, is determined by how the played masking sounds are combined/received at the microphone and a device's movement changes the behavior of sound transmissions. For example, shaking the phone greatly incurs a 10-15dB performance drop, but it is not a typical phone user's behavior.

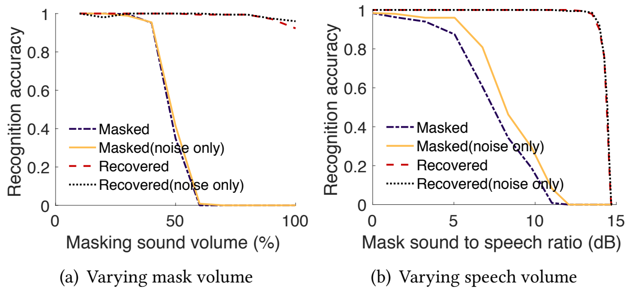
SafeChat is resilient to common ambient noise. For example, only a 3dB MRR drop was observed when a loud rock music was played by a laptop in a setting similar to Fig. 6(a). The data from 21 participants were also collected near a cafe of a crowded student activity center, but no significant performance drop was observed. A common ambient noise does not affect the performance of masking sound removal much because the ambient noise is not correlated to the masking sound. A loud background noise can actually be considered helpful to hide spoken secrets.

A specific issue caused by the ambient noise in our experiments is the under-estimation of MSR. For example, speech energy envelopes might surge due to loud laughs of other people near the testing site, thus decreasing the estimated MSR by using this wrong speech signal as a reference. In such a case, users fail to pass the training even when the masking sound is loud enough to hide the secret. One way to solve this problem is to ask the intended receiver to check the recovered signal of failed training and determine if it is due to the spoken secret or the ambient noise.

Even though SafeChat might suffer from some other extreme cases, one should note that SafeChat is triggered only during a secret conversation. Users will unlikely have a private/secret conversation while moving the phone abruptly, or in a very noisy environment. The intended receiver is also likely to ask users to find a quiet environment to say the secret, if need. Our experiments have shown that SafeChat is resilient to typical real-life environments, like walking and speaking in a typical public area.

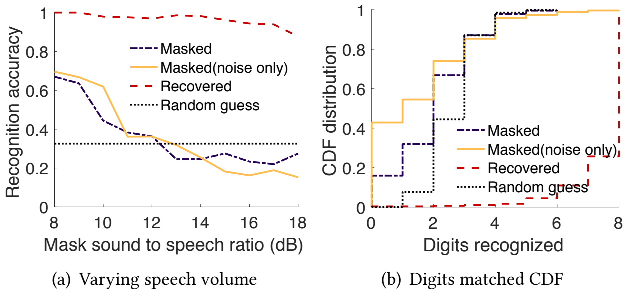
5.4 Attack Performance Against Google Speech API

As mentioned earlier, a threat model we considered is that the malware can try to identify the masked secret from the masked recording by using state-of-the-art speech recognition engines. Specifically, we used the Google Speech API to recognize the laptop traces under different masking sound settings. Note that the recordings fed to Google Speech API are not pre-processed because the API has already been designed to handle noisy audio and it is suggested that noise reduction pre-processing will degrade the recognition accuracy. This built-in noise handling helps the malware uncover the masked secret.



(a) Varying mask volume (b) Varying speech volume

Figure 8: Recognition accuracy via Google Speech API. The large accuracy gap between the masked and the recovered audios represents the effectiveness of achieved audio privacy.



(a) Varying speech volume (b) Digits matched CDF

Figure 9: Recognition accuracy against humans. 75% of recovered secrets can be recognized correctly while participants (humans) can't fully identify any of the masked recordings.

We use the standard word recognition accuracy to evaluate the performance of SafeChat. The word recognition accuracy is defined as $(N-D-S-I)/N$ where N is digits of numbers in the masked speech and I and D are the erroneous digits added or missed in the recognition text. S represents how many digits are substituted by wrong digits. The results of recognition accuracy for Galaxy S5 under different experimental settings are plotted in Fig. 8.

Fig. 8(a) shows the result when a 13dB lower speech is added to the masking sound, and then recorded on Galaxy S5, i.e., MSR is fixed at 13dB. The recognition accuracy is shown to decrease as the volume of played masking sound increases. When the speaker volume is set higher than 50%, the accuracy of recognizing the masked speech decreases sharply while the recovered speech still is recognized by Google Speech API with a high probability. This large performance gap represents SafeChat's effectiveness in providing differential information between authorized and unauthorized recordings. The difference between the masking sound with only masking noise and that with both masking interference and noise is also shown in this figure, but not very clear because the machine-learning algorithm usually identifies speech by statistical properties, such as MFCC or polynomial residual errors, where adding a structured interference like another human's voice with a low volume does not change those features much. The effect of adding an interference is more pronounced when the sound is identified by humans.

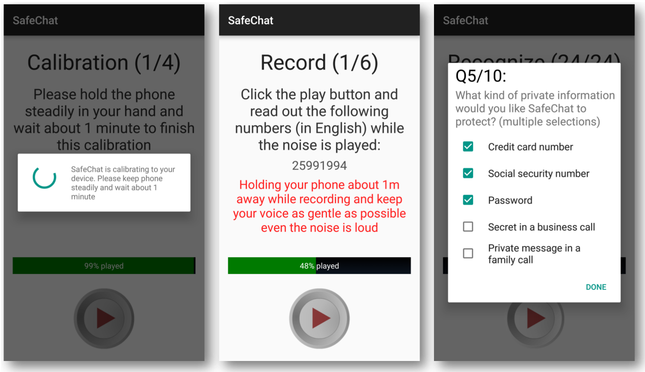
After learning the performance change with different masking sound volumes, Fig. 8(b) shows the recognition accuracy when the masking sound is played with 70% of the speaker's volume (i.e., the calibrated setting in Galaxy S5) and a different volume of speech is added. The purpose of this experiment is to understand the level of speech volume that can be protected with SafeChat. As shown in this figure, the best operation range for Galaxy S5 to prevent secret from being recovered by Google Speech API is when the MSR is higher than 11dB and lower than 15dB. Under this setting, Google Speech API can achieve 98% accuracy in recognizing the uncovered speeches while having less than 0.1% accuracy to understand the masked speech. Note that this operation range varies with device but SafeChat generally sets the lower bound of MSR to 12–15dB to prevent the leakage of secret information. The usability of this setting, like how easy people can maintain their voice with the proper volume, will be discussed further in Section 6.

5.5 Attack Performance Against Humans' Recognition

To evaluate the performance of SafeChat in hiding secrets from humans' recognition, we recruited more than 317 users via Amazon Mechanical Turk to recognize the audios recorded by 6 testing participants. This user study was ruled by our university to be IRB nonregulated since it does not interference human behaviors nor intrude user privacy. We intentionally told the users that (i) the content to recognize is an 8-digit number and (ii) the number to identify can be hidden in the noise. The users are allowed to repeat the recognition of each recorded audio at will and edit their answers as many times as they want. This setting is designed to mimic the scenario when the malware tries to recover the secret with crowdsourcing.

Fig. 9(a) shows the recognition accuracy for the audios recorded on Nexus 6P while playing the masking sound under its calibrated setting. As shown in this figure, once MSR is higher than 13dB, the speech intelligence perceived by the users is found to be no more than the result of randomly guessing an 8-digit number. The accuracy of recognizing the masked audios by the participants is generally higher than that by Google Speech API, because the latter tends to return a null string in case of recognizing the masked audios which are not recognizable, while the participants already know there is a hidden number, and usually return some (even incorrect) answers.

On the other hand, the secret in the recovered recordings can be recognized with 95% accuracy. The performance starts to drop when MSR>17dB, because the residual mask noise may hinder the users from recognizing some of them with low-volume speech signals (e.g., SRR<3dB). This large accuracy gap between recognizing masked and recovered secrets is consistent with the previous results with Google Speech API, thus demonstrating the effectiveness of SafeChat in hiding secrets from being recovered by speech recognition engines and humans. According to these results, we set the threshold of MSR and SRR for users' speech to 13dB and 4dB, respectively.



SafeChat

Calibration (1/4)

Please hold the phone steadily in your hand and wait about 1 minute to finish this calibration

SafeChat is calibrating to your device. Please keep phone steadily and wait about 1 minute

99% played

SafeChat

Record (1/6)

Click the play button and read out the following numbers (in English) while the noise is played:

25991994

Holding your phone about 1m away while recording and keep your voice as gentle as possible even the noise is loud

48% played

SafeChat

Recognize (24/24)

Q5/10:

What kind of private information would you like SafeChat to protect? (multiple selections)

☑ Credit card number

☑ Social security number

☑ Password

☐ Secret in a business call

☐ Private message in a family call

DONE

Figure 10: User study app interface. After users finished the self-calibration stage, they were asked to pass the training phase, record multiple audio clips, and then filled out survey questions.

Fig. 9(b) plots the CDF of digits being recognized with our final setting. More than 75% of recovered recordings are found to be fully recognized by the intended receiver while none of masked audios can be recognized by malware. Moreover, more than 95% of recovered recordings have less than 2 digit errors. Since users usually have 5% or more word error rate in recognizing telephone speeches [21], talking via SafeChat shouldn't impose more burden on the intended receivers than a normal speech. Most recovered secrets recognized with 1–2 digit errors are found to have also been recognized correctly by other users. A majority-vote decision [19] may further avoid these user errors in evaluating the performance of SafeChat. This accuracy is adequate to support most use-cases, considering the fact that the intended receiver usually asks users to repeat the secret when it is not recognizable (as is the case even without SafeChat). Assuming that each recording is independent of others, our result can be interpreted as more than 98% of the 8-digit secrets can be fully recovered within 2 repetitions.

While comparing the effectiveness of masking interference (i.e., added spoken sentences from existing databases), users are found to have a better chance of correctly identifying 7 (of 8) digits on the recordings that include only masking noise, i.e., m_n. These usually happen in the corner cases where the user's speech volume is not fully masked by a masking noise only. In such a case, the added interference, i.e., m_i, helps SafeChat prevent this leakage by making the malware confused with the co-existence of multiple speech sources. By adding both masking interference and noise, SafeChat can ensure the probability of malware guessing the masked secret not better than just a random guess as shown in Fig. 9(b). Note that users have a higher probability of correctly recognizing 2 (of 8) digits in the masked recordings with interference because humans mis-identify the digit of added interference, and there is no such interference to identify in the recordings with only masking noise.

6 USABILITY STUDY

The purpose of this usability study is to assess (1) if users can easily control their speech volume within SafeChat's operation range, (2) if users can tolerate the emitted noise while reading private/secret information, and (3) if users would like to use SafeChat even when it imposes some usage constraints, considering its purpose of protecting audio privacy.

To answer these questions, we asked 27 participants to try the demo app as shown in Fig. 10. Note that this usability study is done during the first time of using SafeChat (a few participants are asked later to join the other testing with varying recording volumes). We first introduced the idea of using noise to protect their private/secret information on smartphones and let them try the demo app of SafeChat. The demo app is configured to first ask users to pass the training phase as described in Section 3 and then ask users to finish 3 tasks of reading a random 8-digit number with SafeChat enabled. Whenever the recording goes out of the operation range of SafeChat, i.e., (MSR < 13dB or SSR < 4dB), an alert message will pop up, asking the user to repeat the same task with a louder or quieter voice. After users finish the testing, i.e., 1 training and 3 recordings, they are asked to fill out a survey questionnaire. The participants are asked to try the same demo app again after they fill out the survey. This second time of test is used to determine if they can meet the requirement of SafeChat more easily since they now know how to use it.

Instructing participants to record in a certain position is found to have a higher chance to record a speech compatible to SafeChat than just asking them to be quiet. Based on our preliminary experiments, people usually speak louder than SafeChat expected, and they tend to put the microphone close to their mouth when they are told to speak quietly. This human instinct turns out to make the recorded secret even louder (due to speaking close to the microphone) when only the "speak quietly" instruction is given. In contrast, asking users to hold the phone in a certain posture is easier to follow, especially when a picture of the desired posture is shown to them. Specifically, we asked all participants in this usability study to record the voice in a similar posture as shown in Fig. 6(b). Keeping such a posture reduces not only the recorded secret volume but also the interference in hearing the masking sound.

According to the above collection of results, users need 1.6 rounds to pass the initial training phase during their first time of using SafeChat. Considering the fact that each training takes about 6 seconds (including the processing time), the overall time overhead of the training is less than 10 seconds on average. After passing the training phase, users need only 1.3 rounds to read an 8-digit secret with the volume compatible with SafeChat's setting. This reduction of repetitions indicates how effectively the training can guide users to learn the correct speech volume. Note that we currently validate the speech criteria only after recording the entire speech. Considering our planned improvement to monitor the speech volume in real time and disable the conversation once the digits are spoken too loudly, the overhead of meeting the requirement can be reduced further. 70% of testing participants would be able to complete the speech in 1 round and 30% of them need one more round to complete it. Moreover, the number of rounds necessary to speak correctly is reduced further as users get familiar with SafeChat. For example, when users perform the second 8-digit recordings, 85% of them can finish it with their first attempt and users need only 1.3 rounds, on average, to pass the training again after filling out the survey.

Recordings in our usability study achieve a similar performance as shown in the previous section. Accuracy in recognizing the masked and recovered recordings is 22% and 93%, respectively. According to the responses to our survey, 18 users think the speech volume requirement is easy to meet and 24 users can tolerate the emitted noise (80% volume calibrated for this study). For the purpose of preventing privacy leakage via unauthorized recording, 22 of the users are willing to use SafeChat, especially when they are telling credit card information or business secrets over the phone. Most participants felt surprised when they learned apps can record audio in the background. Few participants hope the speech volume threshold to be higher while most of them can adjust their volume to meet the current setting. Only 2 participants complained about the masking sounds. One participant left the comment that this technology is useful and can actually be applied to a voice message recording app as well. Considering our current design which needs users to call via SafeChat, it seems a stronger use-case of SafeChat to thwart audio privacy leakage when users are recording secret voice messages (because they need an app to do so anyway).

7 DISCUSSION

We have proposed SafeChat to mitigate the emerging audio privacy leakage via malware's unauthorized recording. SafeChat protects the audio privacy in smartphones without requiring any modification to phone OSes. Our evaluation has demonstrated the effectiveness of SafeChat based on the assumed threat models. Specifically, SafeChat has been shown to be able to effectively hide and recover secret conversations against the state-of-the-art Google Speech API and 371 online-recruited users. One frequently asked question regarding SafeChat is "how about applying a low-pass filter before recognizing the masked secret?" Our testing results show SafeChat to be resilient to such a low-pass attack because the added masking noise covers the entire human speech spectrum. A similar potential attack based on "blind" source separation, such as the independent component analysis (ICA), might be theoretically capable of removing the masking sounds if the number of recording microphones is larger than that of playing speakers. However, we often notice that, in reality, SafeChat saturates all microphones except the one calibrated to recover the secure message, thus failing ICA to separate the "independent" sources. Analyzing and evaluating the security guarantee over other potential attacks, if any, is part of our future work.

Another potential issue of SafeChat is that the phone's echo cancellation might treat the masking signals as the echos and then cancel them automatically. Note that we have already turned on the Android's NoiseSuppressor and AcousticEchoCanceler in our experiments but have not noticed the occurrence of such a cancellation. However, after we online-recruited another 78 testing participants to install SafeChat on their devices, we found a special case where the masking sound was removed by the device OS or hardware before apps receive it in LYF Flame 1. Since this case can be easily identified with our calibration algorithm, SafeChat can inform/warn the user that it cannot support the user's device. After examining the source code of Nexus 6P further, we found a similar phenomenon might also occur when users enable SafeChat first and then make/receive a phone call (via telecommunication). In this case, the system service, i.e., CallManager, has a higher privilege than SafeChat to alter the audio chain and enable the echo cancellation. Note that this echo cancellation is implemented in the chip level, i.e., Qualcomm's proprietary *Fluence* voice enhancement [9]. In our experiment, a normal app API, like the AcousticEchoCanceler, is unable to trigger this function successfully. Instead, this function can be controlled by modifying the persist.audio.fluence.voicecall. Unfortunately, SafeChat is now unable to deactivate this function automatically because it requires the system permission. One possible future direction is to ask phone manufacturers to provide a proper API to control the low-level echo cancellation function. This mechanism might be necessary for other purposes than SafeChat since echo cancellation can be triggered erroneously in many use-cases. See the forum discussions to disable the echo cancellation in Android [2]. Another short-term solution we are working on is to block telecommunication while using SafeChat or implement SafeChat as a system service.

8 CONCLUSION

In this paper, we have shown the plausibility of using sound masking to thwart privacy leakage through malware's unauthorized recording. We have designed, implemented and evaluated SafeChat, a novel combination of Android chat app and a remote sound masking server, which can provide audio privacy on mobile devices without modifying audio privacy/permission scheme in existing OSes. Our extensive experimental evaluation has shown SafeChat to be able to make an up-to-26dB signal strength difference between authorized and unauthorized recordings. With a proper setting, this signal difference could prevent people or state-of-the-art speech recognition algorithms from comprehending the masked sound while the authorized app can understand most of the hidden speeches by our masking sound removing algorithm. Our usability study participants supported the above findings and most of them wanted to use SafeChat to protect their private information such as credit card numbers or passwords.

REFERENCES

[1] [n.d.]. Amazon Mechanical Turk. https://www.mturk.com/mturk/welcome.

[2] [n.d.]. Android Mic (Noise Cancellation) Issue. https://forums.oneplus.net/threads/fixed-for-good-microphone-issue.222059/.

[3] [n.d.]. Cambridge Sound Masking System. http://www.soundmasking.com/selectmasking.html.

[4] [n.d.]. FlexiSpy app. https://www.flexispy.com/en/mobile-and-cell-phone-spy-features.htm.

[5] [n.d.]. Google Speech API. https://cloud.google.com/speech.

[6] [n.d.]. MobiStealth app. http://www.mobistealth.com/features.php.

[7] [n.d.]. Soft dB Sound Masking System. http://www.softdb.com/sound-masking/.

[8] [n.d.]. TheTruthSpy app. http://thetruthspy.com/features/.

[9] [n.d.]. Wideband vocoder and Fluence TM noise cancellation - Qualcomm. https://www.qualcomm.com/media/documents/files/hd-voice.ppt.

[10] 2008. Standard Test Method for Objective Measurement of Speech Privacy in Open Offices Using Articulation Index. *ASTM International* (2008).

[11] N. Anand, Sung-Ju Lee, and E. W. Knightly. [n.d.]. STROBE: Actively securing wireless communications using Zero-Forcing Beamforming. In *Proceedings of IEEE INFOCOM '12*. 720–728.

[12] A. Benyassine, E. Shlomot, H. Y. Su, D. Massaloux, C. Lamblin, and J. P. Petit. 1997. ITU-T Recommendation G.729 Annex B: a silence compression scheme for use with G.729 optimized for V.70 digital simultaneous voice and data applications. *IEEE Communications Magazine* 35, 9 (1997), 64–73.

[13] Soteris Demetriou, Xiao-yong Zhou, Muhammad Naveed, Yeonjoon Lee, Kan Yuan, XiaoFeng Wang, and Carl A Gunter. [n.d.]. What's in Your Dongle and

Bank Account? Mandatory and Discretionary Protection of Android External Resources.. In *NDSS '15*.

[14] S. Goel and R. Negi. [n.d.]. Guaranteeing Secrecy Using Artificial Noise. *IEEE Trans. Wireless. Comm.* ([n. d.]), 2180–2189.

[15] Monson H Hayes. 2009. *Statistical digital signal processing and modeling*. John Wiley & Sons.

[16] Stephan Heuser, Adwait Nadkarni, William Enck, and Ahmad-Reza Sadeghi. [n.d.]. ASM: A Programmable Interface for Extending Android Security. In *Proceedings of USENIX SEC'14*. 1005–1019.

[17] Peter Hornyack, Seungyeop Han, Jaeyeon Jung, Stuart Schechter, and David Wetherall. [n.d.]. These Aren'T the Droids You'Re Looking for: Retrofitting Android to Protect Data from Imperious Applications. In *Proceedings of ACM CCS '11*. 639–652.

[18] R. Gary Leonard and George Doddington. [n.d.]. TIDIGITS Dataset. https://catalog.ldc.upenn.edu/ldc93s10.

[19] Matthew Marge, Satanjeev Banerjee, and Alexander I. Rudnicky. [n.d.]. Using the Amazon Mechanical Turk to Transcribe and Annotate Meeting Speech for Extractive Summarization. In *Proceedings of CSLDAMT '10*. 99–107.

[20] Rajalakshmi Nandakumar, Krishna Kant Chintalapudi, Venkat Padmanabhan, and Ramarathnam Venkatesan. [n.d.]. Dhwani: Secure Peer-to-peer Acoustic NFC. In *Proceedings of the ACM SIGCOMM '13*. 63–74.

[21] Scott Novotney and Chris Callison-Burch. [n.d.]. Cheap, Fast and Good Enough: Automatic Speech Recognition with Non-expert Transcription. In *Proceedings of HLT '10*. 207–215.

[22] Giuseppe Petracca, Yuqiong Sun, Trent Jaeger, and Ahmad Atamli. [n.d.]. AuDroid: Preventing Attacks on Audio Channels in Mobile Devices. In *Proceedings of ACM ACSAC '15*. 181–190.

[23] Nazir Saleheen, Supriyo Chakraborty, Nasir Ali, Md Mahbubur Rahman, Syed Monowar Hossain, Rummana Bari, Eugene Buder, Mani Srivastava, and Santosh Kumar. [n.d.]. mSieve: Differential Behavioral Privacy in Time Series of Mobile Sensor Data. In *Proceedings of ACM UbiComp '16*. 706–717.

[24] Roman Schlegel, Kehuan Zhang, Xiao-yong Zhou, Mehool Intwala, Apu Kapadia, and XiaoFeng Wang. [n.d.]. Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones.. In *NDSS '11*.

[25] Souvik Sen, Naveen Santhapuri, Romit Roy Choudhury, and Srihari Nelakuditi. [n.d.]. Successive Interference Cancellation: A Back-of-the-envelope Perspective. In *Proceedings of the 9th ACM Hotnets '10*. 17:1–17:6.

[26] Manish Shukla, Purushotam Radadia, Shirish Subhash Karande, and Sachin Lodha. [n.d.]. DEMO: On the real-time masking of the sound of credit cards using hot patching. In *Proceedings of the ACM CCS '13*. 1351–1354.

[27] Yu-Chih Tung, Kang G. Shin, and Kyu-Han Kim. [n.d.]. Analog Man-in-the-middle Attack Against Link-based Packet Source Identification. In *Proceedings of ACM MobiHoc '16*. 331–340.

[28] A. D. Wyner. 1975. The wire-tap channel. *The Bell System Technical Journal* 54, 8 (1975), 1355–1387.

[29] Zhi Xu and Sencun Zhu. [n.d.]. SemaDroid: A Privacy-Aware Sensor Management Framework for Smartphones. In *Proceedings of ACM CODASPY '15*. 61–72.

[30] B. Zhang, Q. Zhan, S. Chen, M. Li, K. Ren, C. Wang, and D. Ma. 2014. PriWhisper: Enabling Keyless Secure Acoustic Communication for Smartphones. *IEEE Internet of Things Journal* 1, 1 (2014), 33–45. https://doi.org/10.1109/JIOT.2014.2297998

[31] N. Zhang, K. Yuan, M. Naveed, X. Zhou, and X. Wang. [n.d.]. Leave Me Alone: App-Level Protection against Runtime Information Gathering on Android. In *2015 IEEE Symposium on Security and Privacy*. 915–930.