Acoustic-channel Attack and Defence Methods for Personal Voice Assistants

Peng Cheng

Supervisor: Prof. Utz Roedig

School of Computing and Communications
Lancaster University

This dissertation is submitted for the degree of

Doctor of Philosophy
November 2020

I would like to dedicate this thesis to my loving parents . . .

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation was carried out under the guidance of Prof. Utz Roedig.

Peng Cheng
November 2020

Acknowledgements

Many people have offered support and suggestions to me over my last four-year doctoral study journey, and it is my privilege to recognise them here. I am deeply grateful to my supervisor Utz Roedig. I cannot thank him enough for all of the guidance, advice, and support he has offered me through out these years. He is not only a supervisor to me who taught me how to do research, but also more of a good friend in my life who trusts me and always encourages me to pursue my goals.

I would like to thank my viva voce examiners, Antonios Gouglidis and Matthias Hollick, for their comments and suggestions. I thank my viva voce chair, Paul Rayson, for him hosting the examination. And I owe much to my previous supervisor, Jeff Yan, for offering me the chance to study in Lancaster University and for all of the precious guidance on research from him.

I feel grateful to have many lovely colleagues and friends in my department who not only helped me with my work but also made my days in Lancaster more delightful. I would like to thank Ibrahim Ethem Bagci, Ben Green, James Boorman, Ric Derbyshire, Edward Dearden, Marven Ramokapane, Pierre Ciholas, Hamed Balogun, Charles Weir, Joe Gardiner, John Vidler, Gengshen Wu, Shuo Zhang, Gingfung Yeung, Wenjuan Yu, Helena Tendedez, Kelly Widdicks, James Devine, Peter Garraghan, Alistair Baron, Joe Finney, and Bingsheng Zhang. I am also grateful to current and former SCC staff Karen Coupe, Claire Anne Oulton, Debbie Stubbs, and Odette Allonby for the help they offered to me.

I have had many great friends in Lancaster and Manchester. I feel grateful to Yujia Huang, Yuxin Bai, Shuangfa Huang, Yueqi Wu, Cong Sun and Xiaoyun Chen for the time we spent together over the last few years.

I thank my good friends who contribute to my life more than I let them know: Zesu Wei, Ti Jiang, and Qing Yang.

Finally, I would like to thank my parents for their love and support. They are always my biggest supporters and I am extremely grateful that I could be their son.

Publications

Some of the contents in this thesis are published in various journals and workshops. These publications and my contributions to them are listed down below. The work of one chapter of this thesis which is under review and another one which is to be submitted are also listed.

1. P. Cheng, I. E. Bagci, U. Roedig, and J. Yan, "Sonarsnoop: Active acoustic side-channel attacks," CoRR, vol. abs/1808.10250, 2018, later accepted by International Journal of Information Security, Jul 2019. Available: https://doi.org/10.1007/s10207-019-00449-8. [Online]. Available: http://arxiv.org/abs/1808.10250

   Contributions: I contributed to the idea of the paper. I implemented the Android app code using Android Studio and the Digital Signal Processing (DSP) code on Matlab. I designed and organised the user study. I carried out the most of the evaluation of the proposed mechanism. I am the main author of the paper.

2. P. Cheng, I. E. Bagci, J. Yan, and U. Roedig, "Towards reactive acoustic jamming for personal voice assistants," in Proceedings of the 2nd ACM International Workshop on Multimedia Privacy and Security (MPS '18). New York, NY, USA: ACM, 2018, pp. 12–17. [Online]. Available: http://doi.acm.org/10.1145/3267357.3267359

   Contributions: I contributed to the idea of the paper. I crafted the jamming signal on Matlab. I designed the jamming evaluation platform using AlexaPi and Raspberry Pi. I conducted all of the evaluation of the proposed mechanism. I am the main author of the paper.

3. P. Cheng, I. Bagci, J. Yan, and U. Roedig, "Smart speaker privacy control - acoustic tagging for personal voice assistants," in Proceedings of the IEEE Workshop on the Internet of Safe Things (SafeThings '19). IEEE, May 2019.

   Contributions: I contributed to the idea of the paper. I crafted the test tag signal on Matlab. I designed the tagging device prototype. I carried out all of the evaluation of the proposed mechanism. I am the main author of the paper.

4. P. Cheng and U. Roedig, "Personal Voice AssistantSecurity and Privacy - A Survey", under review, submitted to Proceedings of the IEEE

   Contributions: I contributed to the structure of the paper. I summarised all of the reviewed paper. I am the main author of the paper.

5. P. Cheng, I. E. Bagci, and U. Roedig, "Adversarial Command Detection Using Parallel Speech Recognition Systems", to be submitted

   Contributions: I contributed to the idea of the paper. I contributed to the adversarial command detection framework design. I designed the experiment. I carried out most of the evaluation of the proposed mechanism. I am the main author of the paper.

Abstract

Personal Voice Assistants (PVAs) are increasingly used as interface to digital environments. Voice commands are used to interact with phones, smart homes or cars. In the US alone the number of smart speakers such as Amazon's Echo and Google Home has grown by 78% to 118.5 million and 21% of the US population own at least one device. Given the increasing dependency of society on PVAs, security and privacy of these has become a major concern of users, manufacturers and policy makers. Consequently, a steep increase in research efforts addressing security and privacy of PVAs can be observed in recent years. While some security and privacy research applicable to the PVA domain predates their recent increase in popularity and many new research strands have emerged, there lacks research dedicated to PVA security and privacy. The most important interaction interface between users and a PVA is the acoustic channel and acoustic channel related security and privacy studies are desirable and required.

The aim of the work presented in this thesis is to enhance the cognition of security and privacy issues of PVA usage related to the acoustic channel, to propose principles and solutions to key usage scenarios to mitigate potential security threats, and to present a novel type of dangerous attack which can be launched only by using a PVA alone.

The five core contributions of this thesis are: (i) a taxonomy is built for the research domain of PVA security and privacy issues related to acoustic channel. An extensive research overview on the state of the art is provided, describing a comprehensive research map for PVA security and privacy. It is also shown in this taxonomy where the contributions of this thesis lie; (ii) Work has emerged aiming to generate adversarial audio inputs which sound harmless to humans but can trick a PVA to recognise harmful commands. The majority of work has been focused on the attack side, but there rarely exists work on how to defend against this type of attack. A defence method against white-box adversarial commands is proposed and implemented as a prototype. It is shown that a defence Automatic Speech Recognition (ASR) can work in parallel with the PVA's main one, and adversarial audio input is detected if the difference in the speech decoding results between both ASR surpasses a threshold. It is demonstrated that an ASR that differs in architecture and/or training data from the the PVA's main ASR is usable as protection ASR; (iii) PVAs continuously monitor conversations which may be transported to a cloud back end where they are stored, processed and maybe even passed on to other service providers. A user has limited control over this process when a PVA is triggered without user's intent or a PVA belongs to others. A user is unable to control the recording behaviour of surrounding PVAs, unable to signal privacy requirements and unable to track conversation recordings. An acoustic tagging solution is proposed aiming to embed additional information into acoustic signals processed by PVAs. A user employs a tagging device which emits an acoustic signal when PVA activity is assumed. Any active PVA will embed this tag into their recorded audio stream. The tag may signal a cooperating PVA or back-end system that a user has not given a recording consent. The tag may also be used to trace when and where a recording was taken if necessary. A prototype tagging device based on PocketSphinx is implemented. Using Google Home Mini as the PVA, it is demonstrated that the device can tag conversations and the tagging signal can be retrieved from conversations stored in the Google back-end system; (iv) Acoustic tagging provides users the capability to signal their permission to the back-end PVA service, and another solution inspired by Denial of Service (DoS) is proposed as well for protecting user privacy. Although PVAs are very helpful, they are also continuously monitoring conversations. When a PVA detects a wake word, the immediately following conversation is recorded and transported to a cloud system for further analysis. An active protection mechanism is proposed: reactive jamming. A Protection Jamming Device (PJD) is employed to observe conversations. Upon detection of a PVA wake word the PJD emits an acoustic jamming signal. The PJD must detect the wake word faster than the PVA such that the jamming signal still prevents wake word detection by the PVA. An evaluation of the effectiveness of different jamming signals and overlap between wake words and the jamming signals is carried out. 100% jamming success can be achieved with an overlap of at least 60% with a negligible false positive rate; (v) Acoustic components (speakers and microphones) on a PVA can potentially be re-purposed to achieve acoustic sensing. This has great security and privacy implication due to the key role of PVAs in digital environments. The first active acoustic side-channel attack is proposed. Speakers are used to emit human inaudible acoustic signals and the echo is recorded via microphones, turning the acoustic system of a smartphone into a sonar system. The echo signal can be used to profile user interaction with the device. For example, a victim's finger movement can be monitored to steal Android unlock patterns. The number of candidate unlock patterns that an attacker must try to authenticate herself to a Samsung S4 phone can be reduced by up to 70% using this novel unnoticeable acoustic side-channel.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ACD	Adversarial Command Detection
ADC	Analog-to-digital converter
AGC	Automatic Gain Control
AIC	Active Inaudible-voice-command Cancellation
AM	amplitude modulation
ANC	Active Noise Control
AoA	Angle of Arrival
API	Application Programming Interface
ASR	Automatic Speech Recognition
ASV	Automatic Speaker Verification
AUC	Area Under the Curve
AVI	Automatic Voice Identification
AWGN	Additive White Gaussian Noise
BB	Bounding Box
CC	Connected Component
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
CER	Character Error Rate
CM	Spoofing Countermeasure
CNN	Convolutional Neural Network
COTS	commercial off-the-shelf
CSR	Character Successful Rate
CTC	Connectionist Temporal Classication
dBA	A-weighted decibels
DAC	Digital-to-analog converter
DNN	Deep Neural Network
DNN-HMM	Deep Neural Network - Hidden Markov Model
DPA	Data Protection Act
DFT	Discrete Fourier Transform
DCT	Discrete Cosine Transform
DoS	Denial of Service

DSP	Digital Signal Processing
ECMs	Electret Condenser Microphones
EEMD	Ensemble Empirical Mode Decomposition
EER	Equal Error Rate
EOT	Expectation Over Transformation
ESD	Energy Spectrum Density
EU	European Union
FBANK	Mel-filter bank
FFT	Fast Fourier Transform
FGM	Fooling Gradient Method
FGSM	Fast Gradient Sign Method
FM	Frequency Modulation
FPR	False Positive Rate
GAN	Generative Adversarial Network
GD	Gradient Descent
GDPR	General Data Protection Regulation
GMM	Gaussian Mixture Model
GMM-HMM	Gaussian Mixture Model - Hidden Markov Model
HCI	Human Computer Interaction
HMM	Hidden Markov Models
HVCI	Hidden Voice Command Injection
IFFT	Inverse Fast Fourier transform
IoT	Internet of Things
KWS	Keyword Spotting
LA	logical access
LDA	Linear Discriminant Analysis
LOS	line-of-sight
LSTM	Long Short-Term Memory
LVCSR	Large Vocabulary Continuous Speech Recognition
MCD	Mel Cepstral Distortion
MEMS	Micro Electro Mechanical Systems
MFC	Mel-Frequency Cepstrum
MFCC	Mel-Frequency Cepstrum Coefficient
MOGA	Multi-Objective Genetic Algorithm
NHS	National Health Service
NLOS	none-line-of-sight
NLP	Natural Language Processing
NN	Neural Network
OFDM	Orthogonal Frequency Division Multiplexing
OOB	out-of-band
OS	Operating System

PA	physical access
PCA	principal component analysis
PESQ	Perceptual Evaluation of Speech Quality
PGD	Projected Gradient Descent
PIN	Personal Identification Number
PJD	Protection Jamming Device
PVA	Personal Voice Assistant
PN	pseudonoise
RIR	Room Impulse Response
RNN	Recurrent Neural Network
RNN-CTC	Recurrent Neural Network - Connectionist Temporal Classication
ROC	Receiver Operating Characteristic
SNR	Signal-to-noise Ratio
SNRseg	Segmental Signal-to-Noise Ratio
SSA	Singular Spectrum Analysis
SSW	Spectrum Audio Watermarking
STFT	Short-Time Fourier Transform
SVM	Supported Vector Machine
t-DCF	tandem decision cost function
TDoA	Time Difference Of Arrival
TPR	True Positive Rate
TSR	Transcript Successful Rate
TTS	Text-to-Speech
VA	Voice Authentication
VC	voice conversion
VCS	Voice Controllable System
VoIP	Voice over Internet Protocol
WER	Word Error Rates
WSJ	Wall Street Journal

Chapter 1

Introduction

A Personal Voice Assistant (PVA) is a device which is able to understand spoken commands and to carry out actions accordingly. A PVA contains hardware and software to record, process and analyse sound and in most cases contains also speakers to provide users with acoustic feedback. Thus, PVAs are often referred to as *Smart Speakers*. The term *Smart Speakers* is fitting for purpose built PVA such as the Amazon Echo or Google Home. However, it is also possible to create a PVA by incorporating specialised software in existing computing platforms with sound processing capabilities such as mobile phones, game consoles or car navigation systems. In this case, solutions such as Siri or Cortana are referred to as *Intelligent Assistants*. Both *Smart Speakers* and *Intelligent Assistants* need the back-end cloud support for more complicated analysis (e.g., Large Vocabulary Continuous Speech Recognition (LVCSR)). I use in this thesis the more general term *Personal Voice Assistant (PVA)* which encompasses *Smart Speakers* and *Intelligent Assistants*.

PVAs such as mentioned Amazon Echo, Google Home and Siri are pervasive in people's daily life and they are changing the way people interact with computer systems. People are becoming used to interacting with smart wearable devices, smart appliances in their smart home, and entertainment systems in cars via speech commands. PVAs act as the central hub for receiving, analysing, sending speech commands, and providing users corresponding feedback. Therefore, they are becoming the major interface for Human Computer Interaction (HCI). Various forms of PVAs include standalone devices like smart speakers (Amazon Echo and Google Home), appliances like TVs and set-top boxes (LG and SKYQ) and part of automobile electronic system (Mercedes and Jaguar). Some appliances have PVA capability without users being aware of it and many others feature microphones that are dormant but can be activated by software updates [139].

1.1 Problem Statement and Thesis Goals

PVA deployment density is surging due to its usefulness and the convenience they bring to users. It is safe to say that we are always in range of at least one PVA which exist in one of the possible forms (e.g., smart speakers, intelligent assistants on smartphones or smart watches). With this level of popularity, PVAs' natural intimacy with users, and their critical roles in Internet of Things (IoT) eco-system, security and privacy issues accompanying PVA usage arise.

People probably are not fully aware of the existence of PVAs in their surrounding. Even though they realise the PVA presence, they may not be able to influence their behaviours, let alone deactivate them (i.e., PVAs that belong to others may be activated and start recording). Privacy violation related to PVA usage has become the most prominent concern for users as they are capable of monitoring and understanding speech [78]. I believe that *in a world observed by numerous PVAs it is important to understand how people could express privacy requirements about how their conversations will be handled if these conversations are recorded accidentally by surrounding PVAs.* Privacy requirements could include not giving recording consent, recording locations and recording time stamps, etc.

Besides the obvious capability of listening to conversations, PVAs systems can also identify user activities such as laughter, crying and eating [21, 110] due to rich information embedded in speech signals. Another example is that it might also be possible to identify room characteristics such as room size or shape by just analysing recordings. These information can be revealed by processing the audio samples recorded by microphones on a PVA, which is categorised as PVA passive sensing in this thesis. In addition, PVA speakers can be used to emit (inaudible) sound and reflections can be received by microphones [30], which is categorised as PVA active sensing in this thesis. I believe *due to these PVA sensing capabilities it is important to understand how much information about user daily routine and their living environment are exposed as these information are highly related to user information security and privacy.*

As PVAs are an interface to computer systems and smart environments, it is necessary to police access. Malicious commands can be injected to PVAs without a user's awareness in various ways: using a malware to monitor the environment to seek a proper attack moment [42], transmitting inaudible commands [158] or exploiting psychoacoustic characteristics to hide abnormality in a benign audio command [118]. Speaker identification can be employed to authenticate user interactions, so a PVA can be trained to recognise an individual registered beforehand [14]. However, such identification mechanism can be circumvented by replaying recordings [163] or using speech synthesised [38]. I believe *it is important to understand how to practically circumvent PVA access control with methods for breaking authentication* *mechanisms or with novel methods by exploiting specific PVA software or hardware. On the other hand, it is also important to design robust user authentication methods immune to these bypassing attacks and defence methods against malicious speech commands on PVAs.*

Resilient PVA operations are necessary in many scenarios. However, preventing a PVA from operating normally or stop it from operating at all is possible by using Denial of Service (DoS) methods and interfering with audio processing [31] (or speech recognition [76]). I believe *it is important to understand what kind of DoS attack methods are possible against PVAs and what security and privacy implication of these methods may be for PVAs.*

Apart from these security and privacy concerns from security research perspective, some PVA market statistics are shown below. Industry and public views of security and privacy in the PVA domain are also summarised to enhance the understanding more comprehensively.

1.1.1 PVA Markets and Cybersecurity

PVAs are becoming a main interface for digital environments. Reports show that 21% of the US population have at least one smart speaker, and the the total number of smart speakers has reached 118.5 million [97]. The 2019 Australia Smart Speaker Consumer Adoption Report [138] shows 29.3% of the Australia adult population (i.e., 5.7 million Australians) owned smart speakers. Strategy Analystics [144] shows that the UK, Ireland, Canada, South Korea, Australia, Germany and France will reach the 50% adoption threshold within the next four years. However, concerns about PVA security and privacy arise as the PVA market grows [78]. 2019 Voice Report [98] conducted by Microsoft shows that 41% of PVA users are concerned about trust, privacy and passive listening. Obviously it is necessary to advance our understanding of PVA cybersecurity.

PVAs applies voice control as a natural and convenient interface. Although PVA is only one amongst many possible interaction solutions, there is a trend that it is becoming the dominant interface for future systems , even for safety critical systems. In this thesis, I use the term safety critical systems to refer to systems which on failure may cause serious injury or even lead to loss of life. PVAs have the potential to become a major interface for safety critical systems. For instance, Amazon and the UK National Health Service (NHS) have announced a partnership to enable users to obtain NHS advice via PVAs [44].

When using PVAs on such a large scale as anticipated and applying them for safety critical applications, it is absolutely essential to improve our understanding of existing and potential risks.

1.1.2 Public PVA Security and Privacy Perception

Recently there are a good many highly visible news and articles about PVA security and privacy. These news have drawn great attention from the public. In April 2019 Amazon admitted their workers listen to user recordings collected from PVAs regularly for improving services [18]. In July 2019 Google admitted their contractors listen to voice recordings obtained by their PVAs [135] regularly [135]. There have also been frequent reports on incidents where PVAs record or trigger actions without user intent. A prominent case on the headline was the UK MP Gavin Williamson got interrupted by Apple's Siri when he was addressing the House of Commons [96]. Industry has started to tackle this growing public concern. Amazon Echo, Google Home and Siri have integrated speaker authentication/identification function. However, their main focus is to distinguish multiple speakers (except Siri) in a household sharing a PVA. In 2019 Amazon introduced the command "Delete everything I say today" to provide users with more privacy control. Other than the mainline PVA manufacturers, third-party companies are proposing solutions as well. For instance, Project Alias [68] is a device that feeds a smart speaker constant white noise to disable it, providing users with an extra "safe belt" to control when to activate the PVA. Mycroft [119] is a privacy-prioritised PVA focusing on local speech command processing to avoid cloud analysis of recordings.

Legislators in some countries are investigating the legal context of PVA systems. Currently there are debates about the necessity of new laws and what their forms should be. In Germany the Parliament investigated legality of PVA data collection, and concluded that there remain questions on how third parties and minors can be excluded from data collection to comply with laws [145]. Furthermore, it was also unclear how third parties may use the collected data in the future. In California Assembly Bill 1395 is proposed which would prohibit smart speaker operators from retaining or distributing voice recordings or transcriptions without the user's consent [22].

The general public is concerned about PVA security and privacy, and industry and legislators are starting to react. However, as this thesis later shows in Chapter 3, research has already identified much more sophisticated and serious security challenges than the ones currently triggering public debate.

The goal of this thesis is to improve the understanding of acoustic-channel related security and privacy challenges of PVA usage from the four security research perspectives mentioned above. I first present my work which builds a taxonomy and related state-of-the-art studies are surveyed. Each of the main parts of this taxonomy is about one of these perspectives. Then I present my empirical work which contribute to each part.

1.2 Motivation

A PVA is a networked computer system and as such it is subject to general cybersecurity threats. A PVA can be hacked and, for example, be used as a node in a botnet. The PVA cloud infrastructure can be breached and user specific data can be stolen. Such classical security challenges must obviously also be addressed within PVAs and associated service infrastructures. However, these are out of scope for this PhD thesis. The major and the most important interaction way between users and PVAs is the acoustic channel. The above mentioned research problems all emerge from the acoustic channel interaction, so this thesis focus on PVA security and privacy challenges arising from the use of the acoustic channel. All PVA security and privacy topics mentioned in this thesis are confined within this context if without special mention.

To understand and to answer these challenges, it is helpful to survey the state-of-the-art research and summarise a systematic overview. This overview shows clear categorisation of research domain directly reflecting the questions mentioned in Section 1.1 and includes existing studies in the category where they fit. The overview is a roadmap answering what existing research could be applied to enhance current PVA security and privacy status. In addition, it also points out which areas are critical but lack attention in the research community. A *Taxonomy* for the research domain of PVA security and privacy related to acoustic channel usage is presented in this thesis.

ASR system is a critical part of PVA eco-system. Its simpler form Keyword Spotting (KWS) is usually deployed locally on the front-end PVA device to recognise wake word, and a high-performance ASR is necessary at the back-end cloud for more complicated speech recognition. In recent years, due to great research interest in deep learning security and PVAs applying deep learning as the core technique for ASR, most related studies are about generating adversarial audio input which sounds no harm to human but can trick an ASR to transcribe the audio as harmful commands. These work focus on increasing the audio quality to hide the abnormality added to the original audio commands, increasing the attack robustness to survive over-the-air transmission, or trying to reducing the prior knowledge of the attack knowledge. However, there are barely studies discussing how to defend against these attacks. It is important to study effective defence method against these attacks. A defence method against adversarial attack targeting ASR system is presented in this thesis.

Most studies in PVA security and privacy focus on adversarial commands, but most PVA users are not aware of this threat and their major concern is speech content privacy. PVAs nowadays apply wake word mechanism which helps in preserving privacy. PVAs stay inactive until certain pre-defined wake words are spoken (e.g., "Hey Google" for Google Home). However, activation by accident often happens which is not uncommon in news, so wake word mechanism is not a strong guarantee. There are few work studying better privacy protection mechanism. A unique speech content management method working in the PVA environment is proposed in this thesis.

Due to the increasing deployment of PVAs and its key role as one of the major interfaces between users and digital environment, PVAs are important sensing and computing nodes in the IoT network system. Classic cybersecurity DoS technique in the context of PVA nodes has a great study potential. DoS can be launched by adversaries to disturb the normal operations of PVA, but it can also be utilised creatively to protect users and to prevent the PVA being triggered by accident. An active protection mechanism making use of DoS technique is presented in this thesis.

The major usage of acoustic components (speakers and microphones) on a PVA is voice command interaction, but they could be repurposed to achieve acoustic sensing. There is a large amount of work in the acoustic sensing area, and most of these work could fit in PVA domain. The popularity of PVA also means that it is not uncommon that a PVA is in close proximity to the user daily life, so acoustic sensing studies focusing on security and privacy implication are desirable. Most related sensing security studies are about virtual/actual keyboard keystroke prediction and only microphones are used to receive acoustic signals (passive sensing). However, if speakers of the PVA are used to emit pre-designed signal and echos reflected by nearby objects are captured by the PVA, more types of information can be sensed. This is called active sensing, and an active acoustic sensing attack achieved by turning the acoustic system on a smartphone into a sonar system is presented in this thesis.

1.3 Contributions

Five core contributions of this thesis are:

- A *Taxonomy* for the research domain of PVA security and privacy related to acoustic channel. This taxonomy illustrates an extensive research overview on the state of the art and detailed introduction to work related to research topics in this thesis is also presented.

- A defence method against white-box adversarial attack targeting ASR system. The method can detect if an audio input is suspect to be an adversarial attack.

- A speech content management method working in the PVA environment. This method enables users to signal their consent information which is tagged with their speech content.

- An active protection method against PVAs making use of DoS technique. This method utilises a portable device to listen to the wake word for a PVA, and to emit jamming signal to break the keyword recognition process on a PVA once the device recognises part of the wake word.
- A novel active acoustic sensing attack achieved by turning the acoustic system on a smartphone into a sonar system. This attack uses speakers on a smartphone to emit inaudible pre-defined sound wave and uses microphones to record echoes reflected by nearby objects. Information about movement can be revealed by analysing the echo signal.

1.3.1 A *Taxonomy* for PVA Security and Privacy Regarding Challenges from the Acoustic Channel

Given the increasing dependency of society on PVAs, security and privacy of these has become a major concern of users, manufacturers and policy makers. Consequently, a steep increase in research efforts addressing security and privacy of PVAs can be observed in recent years. While some security and privacy research applicable to the PVA domain predates their recent increase in popularity, many new research strands have emerged. In this thesis a survey of the state of the art in PVA security and privacy is presented. The focus is on security and privacy challenges arising from the use of the acoustic channel. The work that describes attacks and also countermeasures are discussed. Established areas such as Voice Authentication and new areas such as Acoustic DoS that deserve more attention are highlighted. Research areas are described where the threat is relatively well understood but for which countermeasures are lacked as, for example, in the area of hidden voice commands. The work that looks at privacy implications are discussed; for example, work on management of recording consent. This part of the thesis is intended to provide a comprehensive research map for PVA security and privacy.

This work is summarised as a survey paper which is under review for Proceedings of the IEEE. I am the first author and main contributor of this work.

1.3.2 Adversarial Command Detection Using Parallel Speech Recognition Systems

PVAs are used to interact with digital infrastructures and services and security of this interface has become a concern. PVAs are susceptible to adversarial commands; an attacker is able to modify an audio signal such that humans do not notice this modification but the ASR will recognise a command of the attacker's choice. A novel defence method against such adversarial commands is presented. By using a second ASR in parallel to the main ASR of the PVA it is possible to detect adversarial commands. It is infeasible for an attacker to craft an adversarial command that is able to force two different ASR into recognising the adversarial command while ensuring inaudibility. The feasibility of this defence mechanism for practical setups is demonstrated. Our evaluation shows that ASR that differs in architecture and/or training data is usable as protection ASR. In our experimentation setup, the ASR PocketSphinx turned out to be most effective but the other candidates are usable too. Another option of the set up might be to avoid false positive while still obtaining certain adversarial detection capability. For instance, 35% of the time an attack will be detected while not preventing normal use of the system if PocketSphinx is used in our evaluation.

1.3.3 Smart Speaker Privacy Control - Acoustic Tagging for Personal Voice Assistants

PVAs continuously monitor conversations which may be transported to a cloud back end where they are stored, processed and maybe even passed on to other service providers. A user has little control over this process. She is unable to control the recording behaviour of surrounding PVAs, unable to signal her privacy requirements to back-end systems and unable to track conversation recordings. In this thesis techniques for embedding additional information into acoustic signals processed by PVAs are explored. A user employs a tagging device which emits an acoustic signal when PVA activity is assumed. Any active PVA will embed this tag into their recorded audio stream. The tag may signal a cooperating PVA or back-end system that a user has not given a recording consent. The tag may also be used to trace when and where a recording was taken. Different tagging techniques and application scenarios are discussed, and the implementation of a prototype tagging device based on PocketSphinx is described. Using the popular PVA Google Home Mini a demonstrate is presented that the device can tag conversations and that the tagging signal can be retrieved from conversations stored in the Google back-end system.

This work has been published in Proceedings of the IEEE Workshop on the Internet of Safe Things (SafeThings'19). I am the first author and main contributor of the paper.

1.3.4 Towards Reactive Acoustic Jamming for Personal Voice Assistants

PVAs such as the Amazon Echo are commonplace and it is now likely to always be in range of at least one PVA. Although the devices are very helpful they are also continuously monitoring conversations. When a PVA detects a wake word, the immediately following conversation is recorded and transported to a cloud system for further analysis. In this thesis an active protection mechanism called reactive jamming against PVAs is investigated. A Protection Jamming Device (PJD) is employed to observe conversations. Upon detection of a PVA wake word the PJD emits an acoustic jamming signal. The PJD must detect the wake word faster than the PVA such that the jamming signal still prevents wake word detection by the PVA. The thesis presents an evaluation of the effectiveness of different jamming signals. The impact of jamming signal and wake word overlap on jamming success is quantified. Furthermore, the jamming false positive rate in dependence of the overlap is quantified. The evaluation shows that a 100% jamming success can be achieved with an overlap of at least 60% with a negligible false positive rate. Thus, reactive jamming of PVAs is feasible without creating a system perceived as a noise nuisance.

This work has been published in Proceedings of the 2nd ACM International Workshop on Multimedia Privacy and Security (MPS '18). I am the first author and main contributor of this work.

1.3.5 SonarSnoop: Active Acoustic Side-channel Attacks

The first active acoustic side-channel attack is reported in this thesis. Speakers are used to emit human inaudible acoustic signals and the echo is recorded via microphones, turning the acoustic system of a smartphone (i.e., acoustic system of a PVA) into a sonar system. The echo signal can be used to profile user interaction with the device. For example, a victim's finger movements can be inferred to steal Android unlock patterns. In our empirical study, the number of candidate unlock patterns that an attacker must try to authenticate herself to a Samsung S4 phone can be reduced by up to 70% using this novel acoustic side-channel. The attack is entirely unnoticeable to victims. Our approach can be easily applied to other application scenarios and device types. Overall, our work highlights a new family of security threats.

This work has been published in International Journal of Information Security, Jul 2019. I am the first author and main contributor of this work. This work was a finalist for the Pwnie Awards [17] for the most innovative research in 2019.

1.4 Thesis Outline

Chapter 2 introduces the background of topics related to the work in this thesis. Specifically, a PVA technology background describing PVA service environments and underpinning technologies such as signal processing and ASR are introduced.

Chapter 3 presents a taxonomy for PVA security and privacy. Related work are discussed in the categories of this taxonomy respectively. Especially, work of this thesis are also highlighted in the taxonomy and brief summaries of these work are made in each category.

Chapter 4 presents a defence method against white-box adversarial attack on ASR system. This method utilise a defensive ASR working in parallel to the main ASR.

Chapter 5 explores embedding acoustic tags to normal conversation which may be recorded by nearby PVAs to grant users more capability of privacy control.

Chapter 6 proposes a reactive DoS jamming method to prevent people's voice contents being recorded and uploaded to the back end server by PVAs.

Chapter 7 introduces an active acoustic attack aiming at revealing user interaction information by compromising a smartphone used by users.

Chapter 8 summarises each chapter, concludes the thesis, and discusses future work.

Chapter 2

PVA Primer

This section provides fundamental knowledge of PVA ecosystem and the main PVA components. Attacks described in the related work in Chapter 3 usually target one specific point in the PVA processing chain that we describe here in detail. Working mechanism for Hidden voice commands is introduced as well due to its close relationship with ASR introduced in Section 2.2 and audio processing introduced in Section 2.3.

2.1 PVA Ecosystem

In most scenarios the PVA is a distributed system and part of its functionality is located away from the device. For example, speech command processing or command interpretation is usually carried out in a back end infrastructure. Command actuation is usually carried out by other distributed IoT components.

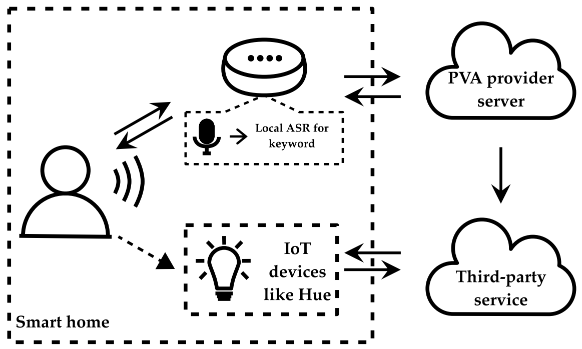


Fig. 2.1 An example of how user interacts with the PVA Ecosystem to turn on a smart bulb (inspired by [4])

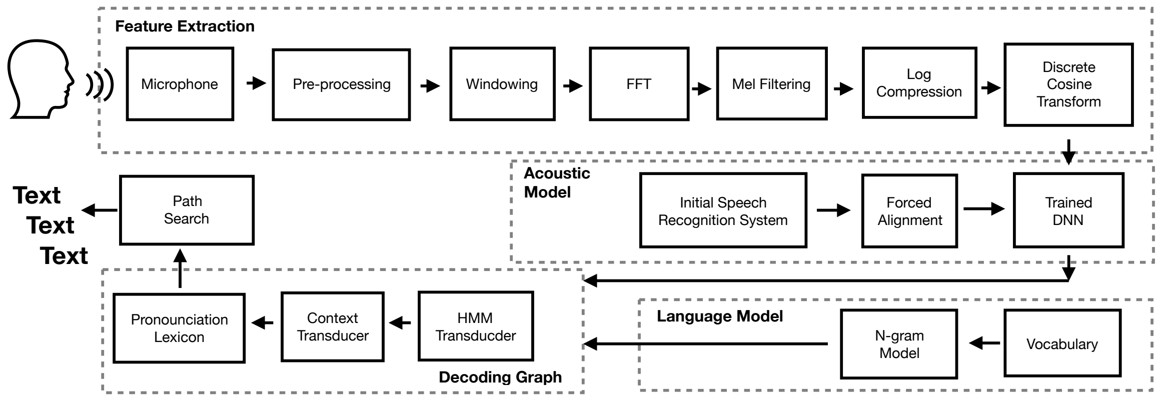


Fig. 2.2 Components and steps of a typical Automatic Speech Recognition (ASR) system

A PVA records sound continuously to perform ASR for wake word detection (*"Alexa"* in case of Amazon's Echo). Once a wake word is detected the PVA submits the recent audio recording to a cloud based back end system where sophisticated ASR is carried out. The speech is analysed, any commands requested are executed and a response might be formed and sent to the PVA to be played out via device speakers. Recordings are stored in the back end and can be used for continuous ASR algorithm improvements and other services.

A typical PVA smart home use case is depicted in Figure 2.1. The user speaks a command such as *Alexa, turn the light on.* The PVA recognises the keyword *Alexa* and the following audio *Turn the light on* is transported to a back end. There, ASR is used to transcribe the audio signal into text. Thereafter, the back end translates the textual command into an API call to the lighting system installed in the users home. The known location of the PVA can be used to determine the correct light to be switched on. The back end may generate audio feedback (e.g. *Light has been switched on)* and send this back to the PVA to play via speaker; however, in this case visual feedback from the light may not require additional voice feedback.

2.2 Automatic Speech Recognition (ASR)

ASR is an inter-disciplinary field of research, incorporating linguistics, computer science and electrical engineering. The goal of speech recognition is to transcribe speech to text automatically, and then to analyse the intent of the speech from transcribed texts. A classic ASR system mimics how humans process speech by transforming the analog acoustic signal to digital representations, from which features are extracted to which machine learning and statistics analysis are applied to extract phonemes (a speech sound) and to finally compose text [155].

ASR is a hot topic in machine learning area and it has gone through four development stages. Gaussian Mixture Model - Hidden Markov Model (GMM-HMM) is the traditional ASR [109]; Deep Neural Network - Hidden Markov Model (DNN-HMM) ASR, replacing the Gaussian Mixture Model (GMM) element with a Deep Neural Network (DNN) appeared after 2012 [36]; Recurrent Neural Network - Connectionist Temporal Classication (RNN-CTC) ASR replaced the Hidden Markov Models (HMM) with a a Connectionist Temporal Classication (CTC) [115]. In recent years, end-to-end ASR techniques using a single Neural Network (NN) to directly map audio input to text has drawn much attention [7]. Researchers have been studying attention/transformer-based end-to-end ASR [115]. A detailed description of these recent developments in ASR design is out of scope of this thesis. In the next paragraphs we introduce the classic ASR steps to illustrate fundamental principles of ASR. The individual ASR steps based on a typical DNN-HMM structure are shown in Figure 2.2.

It must be noted that in the context of this thesis we target English speech recognition; for other languages some ASR steps are constructed differently to deal better with language specifics (for example, to deal with tonal languages such as Mandarin). Each ASR step can be targeted by an attacker to achieve their goal. Thus, detailed understanding of the individual processing steps is essential for an attacker to craft a successful attack and for the defence to implement appropriate countermeasures. All steps in state-of-the-art ASR systems are normally designed to provide robust speech recognition, to deal with noisy environments or accents. However, they are generally not designed to handle audio sources with malicious intent that aim specifically to manipulate their processing outcomes.

Feature Extraction The classical ASR processing chain starts with *signal pre-processing*. Besides steps such as dithering and DC-removal, this step includes pre-emphasis to amplify higher frequencies. The lower frequencies of the speech signal have the greatest energy while higher frequencies have the greatest intelligibility [89]. Next, *windowing* is performed to divide the speech signal (a time-variant signal) into overlapping frames (each small enough to be stationary for analysis, typically $20ms$ long). *Short-Time Fourier Transform (STFT)* is applied to each frame to transfer the signal from the time domain into the frequency domain. The human ear can not discern the difference between two closely spaced frequencies and this effect becomes more pronounced as the frequencies increase [64, 89]. *Mel filtering* is applied to determine how much energy exists in various frequency regions; this information is sufficient for the following ASR steps. Mel filtering is followed by *log compression* to mimic the non-linear perception of loudness of the human auditory system (we don't hear loudness on a linear scale). The output of this operation is called the filterbank coefficients. Next, the *Discrete Cosine Transform (DCT)* of the log filterbank coefficients is calculated to decorrelate energies and to drop higher DCT coefficients representing fast changes which degrade ASR performance [90]. The outputs are Mel-Frequency Cepstrum Coefficient (MFCC) features representing the observed speech.

It has to be noted that the feature extraction steps described here fit best in a classical GMM-HMM ASR system; the recent use of a DNN in the following processing steps usually uses different features such as Mel-filter bank (FBANK) as input (MFCC can be simply treated as FBANK + DCT). The DCT ensures the independence among input features which makes it easier to build a GMM model, however, a DNN has a better learning capability and it is not necessary to remove dependence among features. Moreover, FBANK contains more original information which can benefit the speech decoding process. Other forms of input features such as waveform are being studied but are at a very early research stage.

The next processing steps aim to find the best word sequence given the extracted (e.g. MFCCs) features. For this purpose, machine learning techniques are used.

Speech Decoding In the next steps it is the aim to find the most likely word sequence $W = \{W_1, ..., W_N\}$ for a given sequence of observations (features) $O = \{O_1, ..., O_N\}$. The word sequence maximising the posterior probability $P(W|O)$ has to be found (called posterior, as we can only know its value after observing speech). Applying Bayes' rule the posterior probability can be described as:

$$P(W|O) = \frac{P(O|W)P(W)}{P(O)} \tag{2.1}$$

The probability maximisation does not depend on $P(O)$ and the problem can be specified as:

$$W = argmax_W P(O|W)P(W) \tag{2.2}$$

Equation 2.2 is the fundamental equation of ASR [56]. The component $P(O|W)$ of Equation 2.2 is known as the *acoustic model*, and the component $P(W)$ is known as the *language model*. The acoustic model computes the likelihood of the digital representation of the observed speech signal for a given possible word sequence. The language model describes the likelihood of a sequence of words in a given context.

In most PVAs used today, acoustic modelling uses *Deep Neural Networks (DNNs)* for frame-level predictions followed by the application of *Hidden Markov Models (HMM)* to provide sequential prediction based on discrete frame-level ones [89]. The discrete events are the states of context-dependent phonemes. Multiple states sharing similar properties are tied together as so called senone which is used to compute the acoustic model scores. A DNN labelling all the HMM states of all senones is used for this purpose. To *train* the DNN an GMM system or a previously trained DNN with the same senone configuration is used, and *forced alignment* is applied to label all frames of the training data with the corresponding senone labels. With these preparations, a DNN acoustic model is trained. Various deep learning neural network structures such as feedforward, convolution and recurrent have been studied and applied as the acoustic model [37, 63, 55, 3].

The language model represents the knowledge about language in a certain application context. The basic concept of language modelling is to assign high probabilities to likely utterances (the smallest unit of speech; a continuous piece of speech beginning and ending with a clear pause) and low probabilities to the unlikely ones. A finite *vocabulary* is built from training data and every combination of words from the vocabulary is possible, although the probability of each combination will vary. An *N-gram model* is generally used to encode the probabilities for a vocabulary.

A *decoding graph* is applied to find the word sequences which jointly maximise the acoustic model score and word sequence score obtained from the language model (an implementation of Equation 2.2). This decoding graph is comprised of an *HMM transducer* which maps sequences of *HMM states* (senone labels) to sequences of HMM model names (triphones which is a model which takes into consideration left and right neighbouring phonemes), an optional but widely used *context transducer* which maps sequences of triphones to sequences of phonemes, a *pronunciation lexicon* which maps sequences of phonemes to sequences of words, and finally the language model assigns weights to the phrases. Thus the final scores of the word sequences are obtained. The *path search algorithm* will apply the above procedures and find the most likely word sequence which is the goal of ASR.

2.3 Audio Processing

A PVA includes at least one microphone and associated digitisation processing chain as shown in Figure 2.3. Figure 2.3 is a detailed view of the Microphone block shown in Figure 2.2. It is important to also consider this analogue part of the processing chain as some attacks focus on this particular element of a PVA.

The system usually consists of components including a microphone, a pre-amplifier, a low-pass filter, and an ADC. There are two types of microphones: Electret Condenser Microphones (ECMs) and Micro Electro Mechanical Systems (MEMS). However, MEMS

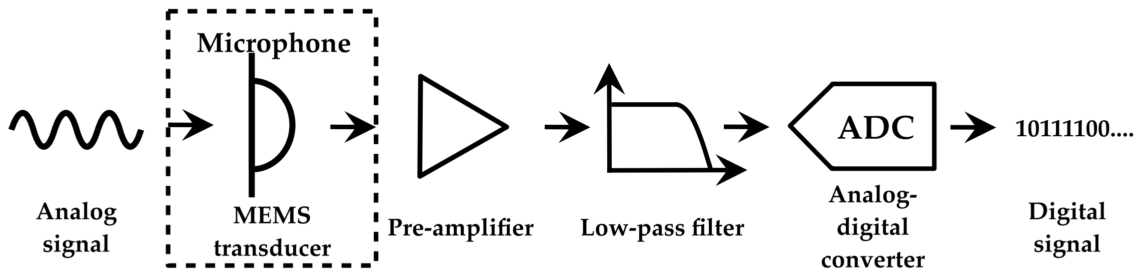


Fig. 2.3 Microphone components and the sound signal flow through them [111]. Note that we do not show the sampling and hold and quantisation details of the Analog-to-digital converter (ADC), but show the final digital signal instead.

dominates the smart device market due to advantages in packaging and power consumption [158]. The microphone is a transducer converting the airborne acoustic signal to an electric signal which only reacts to sound within the spectrum from 20 Hz to 20 kHz. The pre-amplifier amplifies the signal for processing in later stages. The low-pass filter removes noise above 20 kHz which is outside the audible sound range. The ADC converts the analog signal to digital form. The sampling frequency of the ADC is normally 44.1 kHz which restricts the maximum frequency of the analog signal to 22 kHz as described by the Nyquist theorem [158].

The speaker system used in a PVA is in principle the exact reverse of the microphone. The digital signal is converted to analog form via an Digital-to-analog converter (DAC). Then the analog electrical signal is transferred to an air pressure signal via the vibration of the diaphragm of the speaker [111].

2.4 Hidden Voice Commands

Hidden Voice Command Injection (HVCI) or Hidden Command Injection for short aims to inject voice commands into a PVA without users noticing this injection. The injected command is *hidden* from users in the vicinity of the PVA. The attacker aims to trigger actions without legitimate users noticing the interaction of the attacker with the PVA. In order to conceal this interaction existing work has looked at various techniques ensuring that a person is unable to hear the submitted command while the PVA's ASR is able to understand it. While these techniques are the essential component to enable hidden voice commands it is also often necessary for an attacker to modify other elements of PVA interaction. After submitting a command, the PVA usually responds with a confirmation via its speakers. For example, the voice command for a home automation system "Alexa, open the front door" would result in a response “Front door opened” which an attacker would need to suppress too in order to achieve a fully hidden interaction. Research has focused on achieving hidden command injection which we discuss here, less work has considered how to conceal all PVA interaction.

It has to be noted that there is also work (such as by Diao et al. [42]) which aims to hide interaction with a PVA by carefully choosing times of command injection and volume level. Although such work can conceal interaction these techniques do not aim at modification of speech signals and are therefore not outlined in detail in the later related work section.

We distinguish in this thesis three categories of HVCI: *C1.2.1 - Hardware Non-Linearity*, *C1.2.2 - Obfuscated Commands* and *C1.2.3 - Adversarial Commands*. The indexes introduced here are to be consistent with the index of these three categories presented in the next Chapter.

Work in the first category targets the analogue signal processing path of a PVA and makes use of the fact that humans are unable to hear in the high frequency range (typically above 18kHz). The voice command is submitted in the frequency space unnoticeable to users while the non-linear behaviour of the analogue signal processing path ensures that the signal is processed by the ASR.

The second class of work aims at submission of an audio signal which humans perceive as noise, the command is understood by PVAs but not by humans. For this purpose, the attacker starts with the target command and this audio signal is gradually changed until it becomes unintelligible for a human but the PVA still decodes the command. The resulting audio signal is called the obfuscated command.

The third class is similar to the second. However, the attacker uses two audio signals, the perceived command and the desired target command. The audio signal of the perceived command is gradually modified until the PVA recognises the target command while a human still hears the perceived command. The resulting audio signal is called the adversarial command.

Note that obfuscated commands and adversarial commands are also called obfuscated/ adversarial examples in this thesis.

2.4.1 Hardware Non-linearity

The non-linearity of the PVA’s analogue front-end enables signal processing of voice commands that are transmitted in an inaudible frequency range. This non-linearity exists in the pre-amplifier and the microphone itself as shown in Figure 2.3 [158]. The effect can be modeled similarly for both components and we use the pre-amplifier as an example to illustrate the effect. The ideal function of an amplifier can be described as: $S_{out} = A_1 S_{in}$. The input signal S_{in} produces the linear amplified signal S_{out}. However, for frequencies typically

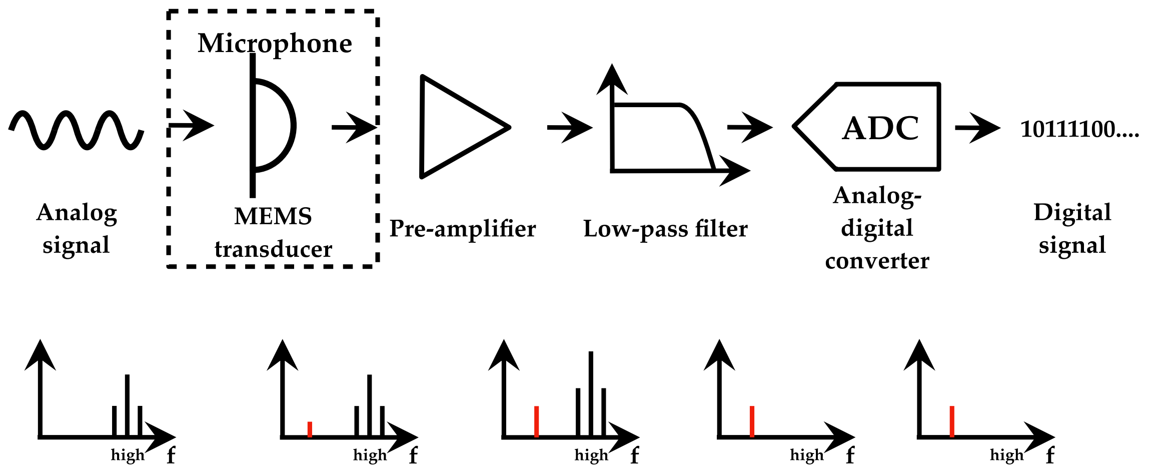


Fig. 2.4 Illustration of the demodulation achieved by non-linearity effects

above 25 kHz [111], the non-linearity of the amplifier will also introduce higher-order signal components:

$$S_{out} = A_1 S_{in} + A_2 {S_{in}}^2 + A_3 {S_{in}}^3 + \dots \tag{2.3}$$

Signal components above second order can usually be neglected as they are too weak. However, the second order component must be considered. Adversaries exploit this feature to demodulate a high frequency attack signal to the baseband from the inaudible frequency range. In the next paragraphs this mechanism using a simple signal example is described.

Assume the attack input signal is S_{in}:

$$S_{in} = Cos(\omega_1 t) + Cos(\omega_2 t) \tag{2.4}$$

Where ω_1 and ω_2 are above 20kHz, such that S_{in} is inaudible. The amplifier function as Eq. 2.3 with input signal Eq. 2.4 now produces the following output:

$$S_{out} = A_1(Cos\omega_1 t + Cos\omega_2 t) + A_2(Cos\omega_1 t + Cos\omega_2 t)^2 \tag{2.5}$$

For simplicity, we assume A_1 and A_2 to be 1. The first term will be filtered out as it only contains ω_1 and ω_2 frequency components and both are above 20 kHz. The second order term generates various frequency components:

$$(Cos\omega_1 t + Cos\omega_2 t)^2 = 1 + \frac{1}{2}Cos2\omega_1 t + \frac{1}{2}Cos2\omega_2 t + Cos(\omega_1 - \omega_2)t + Cos(\omega_1 + \omega_2)t \tag{2.6}$$

All these frequency components, except $cos(\omega_1 - \omega_2)t$ will be filtered as they are above 20kHz. The adversary only needs to ensure that $\omega_1 - \omega_2$ is below 20 kHz. In this case this frequency component is now processed by the ASR system. Figure 2.4 illustrates this outlined mechanism.

2.4.2 Obfuscated Commands

The aim of obfuscated commands is the creation of an audio signal which humans perceive as noise while the PVA interprets a command. The attacker starts with a target command that is gradually changed until it is unintelligible for a human while the PVA is still able to decode the command.

The purpose of ASR is to transcribe speech to corresponding text. This process can be defined as:

$$y = \underset{\tilde{y}}{\operatorname{argmax}}\, p(\tilde{y}|x) \tag{2.7}$$

x here is the audio input, and $\tilde{y}$ are all possible transcription candidates. The ASR aims to find the most likely transcription y given the audio input x. Once the ASR has been trained it's function is $y = f(x)$.

A human listening to the audio signal x also interprets the signal and normally would conclude that the same transcription y recognised by the ASR is the meaning of the command. This process can be described as $y = f_H(x)$ with f_H describing the human's processing capability.

An adversary can modify an input signal x by adding perturbation δ, resulting in $x' = x + \delta$. The following situation arises when an ASR decodes x':

$$y = f(x') \tag{2.8}$$

y here is the obfuscated command transcription which remains the same as the one decoded from unperturbed input x. However, a human cannot perceive the same transcription y this time from the audio signal x' (it is perceived as noise; $f_H(x') = \emptyset$ which means the human transcription is empty). The audio input x' is called the obfuscated command.

There is as well the other situation where $y = f_H(x')$ and $\emptyset = f(x')$ which means the ASR is unable to transcribe the input while a human is understanding the command well. There is work in this direction (such as work by Abdullah et al. [6]) which aims to prevent machines listening into conversations.

There is as well the third situation where $y = f_H(x')$ and $y' = f(x')$ which means the ASR transcription and human transcription are different. In this case x' is called an adversarial command and we discuss this situation in the next section.

To create an either obfuscated or adversarial command it is helpful for the attacker to have access to the internal workings of the ASR. An attack relying on internal knowledge of the ASR (e.g. such as the trained DNN model) is referred to as a *whitebox* attack. If the attacker is not able to access the internal ASR workings the attack is classified as *blackbox* attack. Generally, attacks assuming the ASR as a blackbox are more difficult to execute.

2.4.3 Adversarial Commands

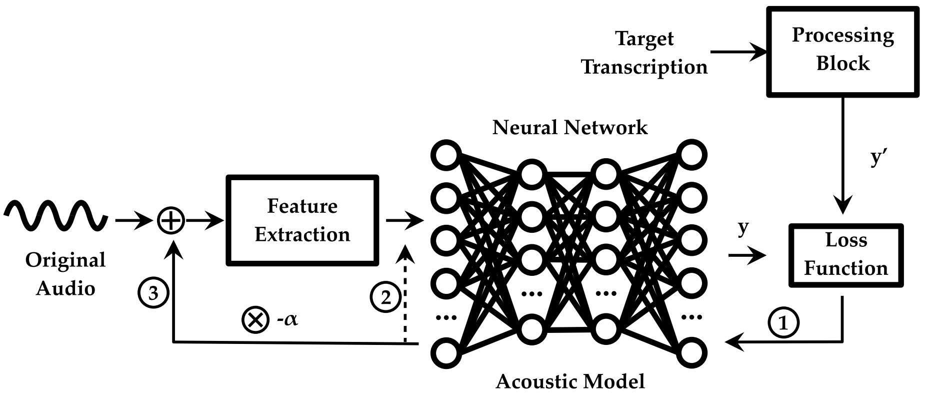


Fig. 2.5 Illustration of the adversarial examples generation

Adversarial commands build on the previously described obfuscated commands. Obfuscated and adversarial commands both achieve a perception difference between ASR and human. The difference is that for an obfuscated command the added perturbation results in a signal perceived by a human as noise while the ASR still successfully decodes the original command as the transcription y. In case of the adversarial command, even with the added perturbation, a human still perceives the adversarial audio input x' as original benign command transcription y, while an ASR recognises the audio input x' as the adversarial command transcription y'.

We distinguish so called *targeted* and *non-targeted* adversarial commands. In case of a *targeted* adversarial command the attacker is interested in one specific command transcription T which is carefully selected ($y' = T$). In case of a *non-targeted* adversarial command the attacker does not care about what specific command would be decoded by the ASR; the attacker only wants to ensure that human and machine transcription are not the same. This may affect the performance of some paralinguistic systems which analyse how people speak.

The analysis results can usually be used in many applications (e.g., Automatic Speaker Verification (ASV), health status detection, or mood detection) [53].

Figure 2.5 is an illustration of the general adversarial command signal generation process. The exact process may vary depending on the ASR model, black-box/white-box assumption and perturbation target such as feature vectors or raw audio input. We use the generation of adversarial commands for a DNN-HMM ASR as example. Adversarial commands are generated through an iterative process. In each iteration, the output y of the DNN (the acoustic model) is compared with the target y' using a loss function. Then the gradient of the loss function with respect to the corresponding input is calculated through back-propagation (shown in Figure 2.5 as edge 1 and edge 3). By finding the perturbed input resulting in the local/global minimum it is ensured that the input is transcribed as the target command. In addition, the perturbation value is constrained by a threshold, ensuring that people cannot perceive the difference between the new signal and the original audio input. There are variations in different studies in regard to techniques on where to add the perturbations. Edge 2, shown as dashed line in Figure 2.5 refers to the perturbation for creation of adversarial examples is added to the feature vectors such as MFCC; Edge 3 refers to perturbation added directly to the raw audio input.

Chapter 3

Taxonomy for PVA Security and Privacy Related to Acoustic Channel

3.1 Taxonomy

In this chapter, we provide an extensive research overview on the state of the art. We summarise significant work carried out to date and show where in our given taxonomy the work belongs. Our classification shows areas that are well researched and others that require more attention. We also show areas where security and privacy problems are quite well understood but where solutions are still lacking. Our work shows the relationship between different existing fields of work. Researchers have looked at security and privacy issues of speech recognition systems, voice control systems, general audio systems and recently PVAs and smart speakers. We summarise the state of the art of these research communities that complement each other and contextualise these for the PVA domain.

Chapter 2 gives a PVA technology background describing PVA service environments and underpinning technologies like signal processing and ASR. This chapter describes security and privacy PVA research taxonomy that is used to categorise the research contributions of this thesis, and to report on existing cybersecurity work in the PVA context with specific focus on work targeting the acoustic channel. We consider work that reports on attacks but also consider reported defence mechanisms.

The four sections included in this chapter describe work in the four main branches of our taxonomy *C1 - Access Control*, *C2 - Privacy*, *C3 - DoS* and *C4 - Sensing*. Section 3.3

summarises and outlines areas in which we feel research advances are necessary to satisfy industry and public demand for secure operations of PVAs.

As described in Chapter 1, the focus of this thesis is on acoustic channel security and privacy. The CIA triad of confidentiality (CIA-C), integrity (CIA-I) and availability (CIA-A) underpins computer security [127]. Security controls and vulnerabilities can be associated to these areas. Confidentiality means that data and resources are protected from unauthorised access. Integrity specifies that data and services are protected from unauthorised changes and are reliable. Availability means that that data and services are available when required. Next I describe more specifically the PVA security and privacy context of the work discussed and presented in this thesis and how they relate to the classical CIA view of computer security. In the PVA context, the biggest security threat is compromise of authentication which allows an adversary to access data and interact with services controlled by the device. Work in the category C1 - Access Control is related to confidentiality in the CIA Triad (CIA - C). The work described in C1 considers compromise of authentication via the acoustic channel only and classical attacks (e.g. attacks on the OS or communication network of the PVA) is out of scope of this work. Specific focus is also on compromise of authentication without awareness of legitimate users which is not a specific concern in the general CIA triad. As the compromise of authentication allows an adversary to interact with services controlled by the PVA and access to data the category C1 of the taxonomy used also relates to CIA - I as integrity is compromised when authentication fails. In the PVA context users have strong privacy concerns. Users would like to keep conversations overheard by a PVA private. A PVA can be activated by accident and a conversation may be recorded. Also, speech recordings can reveal additional information about a user such as speaker identity or speaker emotions. Privacy is the right of individuals of protecting their personal information while confidentiality deals more broadly with unwanted disclosure of information. In the context of this work I therefore used category C2- Privacy as a more appropriate category which is related to CIA-C. In the PVA context access to data is access to a speech signal, potentially revealing very private information. Category C3 - DoS is well aligned with CIA - A of the Triad model. In the PVA context availability means specifically the availability of services provided via the PVA. Hence, availability in the context of this work can be described more specifically as DoS which is the method an attacker can apply via the acoustic channel to influence availability. C2 - Privacy is about information embedded within a user's speech signal. However, PVAs placed in users' private spaces can obtain additional information using acoustic sensing. The acoustic components on a modern PVA today are adequate to sense a users environment and interaction with it. This is a large area of work and distinct from the privacy of the speech signal itself. For this reason a unique category is used - C4 -

Sensing which relates to the CIA Triad element of CIA-C. As described, each category in the taxonomy relates to one or more elements in the CIA model. C1 - Access Control relates to both CIA - C and CIA - I. C2 - Privacy relates to CIA - C. C3 - DoS relates to CIA - A. C4 - Sensing relates to CIA - C. I structured the taxonomy using C1, C2, C3 and C4 rather than the classical CIA elements as it is more fine-grained and better fits the nature of security and privacy work in the PVA domain.

We believe that this is a useful categorisation as it aligns with public perception of cybersecurity in the PVA domain and existing work can be placed in these four categories. In some cases existing work falls into several categories but in most cases work can be attributed clearly to one of the categories. Research work is placed in the category to which the work contributes most. In most cases it is clearly possible to identify this area. For instance, attacks described in DolphinAttack'17 enable an attacker to inject an inaudible command into a PVA. Even though this attack scenario has also privacy implications (an attacker may access user data by these means), the main purpose of the work is to bypass access control. C1 - Access Control is therefore used to categorise this work. Cheng'19 proposes a method for embedding additional information in voice signals recorded by PVAs. The main purpose of this work is to provide users with more control over voice recordings. The work is therefore classified under C2 - Privacy (specifically C2.2 - Consent Management). The method could also be used to implement access control but it is not the main contribution of this work. Cheng'18 proposes a jamming method to prevent a PVA from being activated. This method can be utilised to protect user privacy. However, the main contribution of the work is a detailed study of DoS methods and it is therefore listed in C3 - DoS. SonarSnoop'18 proposes a framework repurposing a PVA to be a sonar system which can sense finger movements thus breaking Android unlock pattern. The application scenario is related to user privacy and data confidentiality. However, the main contribution of the work is a sensing framework and therefore it is categorised in C4 - Sensing.

Figure 3.1 shows the categorisation of the surveyed work according to this taxonomy. In the following paragraphs we describe each category in more detail and explain further division in subcategories. The work in this thesis are highlighted in the figure. Three of them have been published (Cheng'19, Cheng'18 and SonarSnoop'18), and they are shown as green nodes. Cheng'20 is to be submitted and it is shown as a yellow node. These work are presented in detail in later chapters of this thesis.

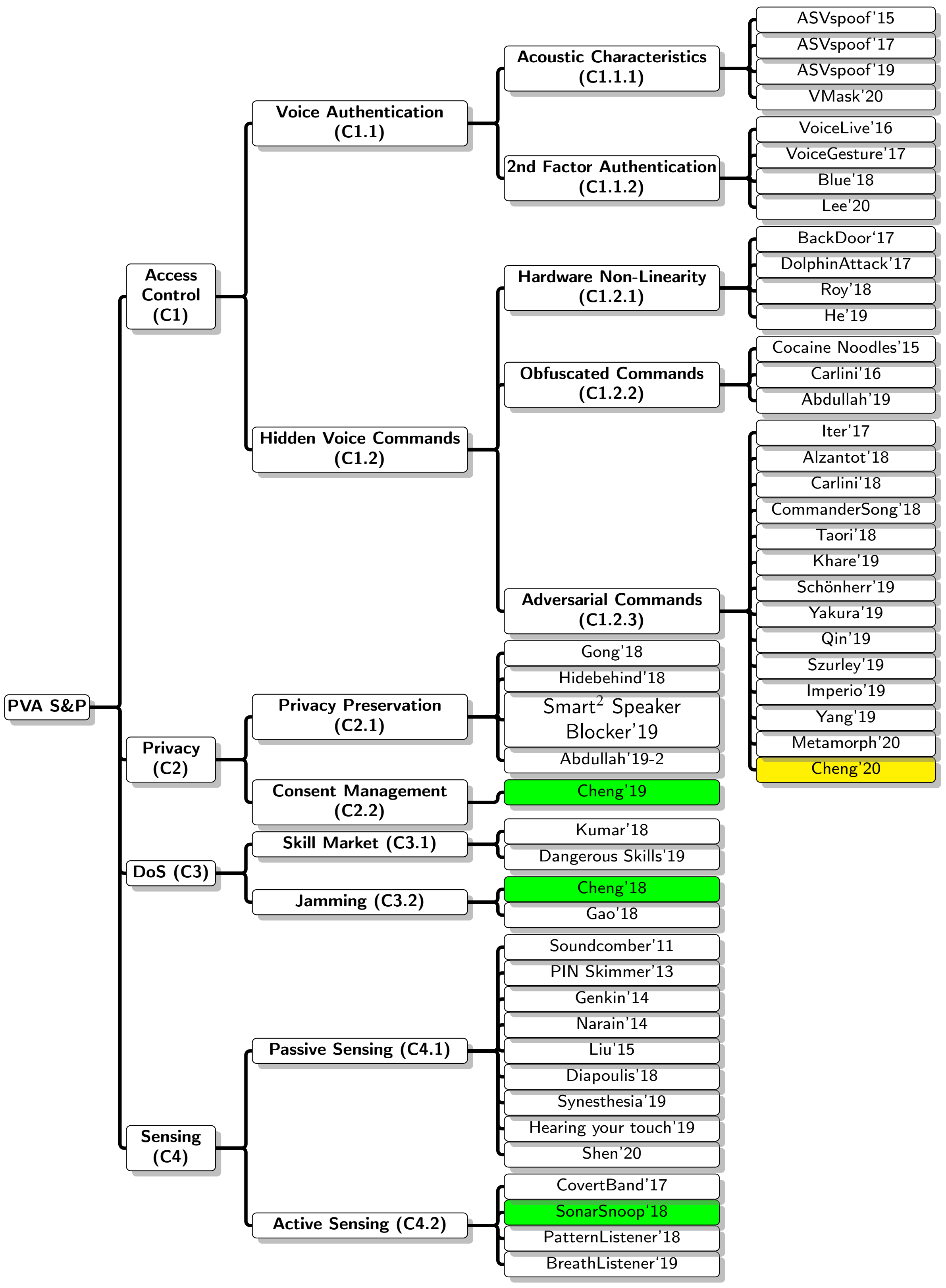


Fig. 3.1 Taxonomy of PVA security and privacy challenges

3.1.1 C1 - Access Control

In this category falls all work that aims at circumventing access control to services a user can access via the acoustic channel of the PVA. Furthermore, work on the defence side are included which aim to prevent bypassing the access control.

We use two subcategories here labeled *Voice Authentication (C1.1)* and *Hidden Voice Commands (C1.2)*.

C1.1 groups work that aims at authenticating a speaker and two significant lines of work are distinguished here: *Acoustic Characteristics (C1.1.1)* and *2nd Factor Authentication (C1.1.2)*. C1.1.1 describes work which exploits acoustic characteristics of the speech signal for spoofing detection. C1.1.2 lists work that correlates a second source of information with the speech signal. The acoustic channel itself may provide this secondary source of information. For example, speakers and microphones can be used as sonar system to read facial expressions of a human speaker. Classical two-factor authentication uses a combination of two different authentication factors: something you know, something you have, something you are. In this work, 2nd Factor authentication is a different concept and refers to the fact that a secondary data source (secondary as it is not the speech signal itself) is used to detect a spoofed speech signal.

C1.2 contains work where the PVA's speech recognition chain is manipulated to trigger unauthorised action. Generally, these forms of PVA attacks are referred to as *command injection* and the attacker aims to carry out the attack without being noticed. Three general types of Hidden Voice Commands are distinguished: *Hardware Non-Linearity (C1.2.1)*, *Obfuscated Commands (C1.2.2)* and *Adversarial Commands (C1.2.3)*.

Work in category C1.2.1 targets the analogue signal processing path of a PVA. The second class of work (C1.2.2) aims at submission of an audio signal which humans perceive as noise while the command is understood by the PVA. Work in category C1.2.3 aims at generating an audio signal which is interpreted differently by humans and the ASR.

Most related work covered in this chapter falls into category C1; it is the most active research active area. However, it is not the area most users are concerned with.

3.1.2 C2 - Privacy

The use of the PVA's acoustic channel can lead to a loss of privacy as voice recordings may reveal sensitive user information.

Methods have been proposed to ensure voice recordings do not reveal such user information. In a simple case, the PVA is prevented from recording speech. More sophisticated methods aim at transforming the audio signal before it reaches the PVA, removing cues that may enable speaker identification or detection of speaker characteristics such as their mood. These works are summarised under the category *Privacy Preservation (C2.1)*.

The category *Consent Management (C2.2)* details work that aims at giving users some degree of control in regards to PVA recordings. Users would like to be able to control which PVA in their vicinity records their voice.

Although there is not much research work in this privacy category it is the most concerning area for the general public. Many recent media reports point out privacy concerns of users. This imbalance between category C1 and C2 may indicate that research should pay more attention to PVA privacy research.

3.1.3 C3 - DoS

The acoustic channel can be subject to a DoS attack. The attacker aims at blocking services provided by a PVA using the acoustic channel.

Current work in this area falls in two categories *Skill Market (C3.1)* and *Jamming (C3.2)*. Attacks in C3.1 aim to manipulate the back-end processing chain of the service invocation after speech recognition. Jamming attacks (3.2) usually target wake word recognition of the PVA.

Not much research work has investigated this area given that PVA will be used as interface for safety critical systems where continuous operation should be ensured. For example, PVAs are used so support surgeons and it is vital that voice commands for operating equipment are not blocked.

3.1.4 C4 - Sensing

This category summarises work in which the acoustic channel is used for an attack but the focus is not on the speech processing chain. The PVA's acoustic system is used to extract security relevant information by analysing sound.

In category C4.1 we outline existing work using *Passive Sensing*. The acoustic channel is observed and information is extracted. The attacker may use audio data recorded by the PVA itself or use an additional device.

Category C4.2 describes *Active Sensing*. The PVA might also be used actively and an acoustic signal is emitted and the response is evaluated.

Quite a lot of work has investigated active and passive sensing in the acoustic domain which are directly applicable to the PVA context. It has to be noted that sensing also leads to a loss of privacy (C2). However, the distinction between category C2 and C4 is that while C2 purely assumes extraction of information from a voice signal, C4 is broader and considers sensing in general.

3.1.5 Paper Inclusion

In this chapter we include existing work that describes attacks and/or countermeasures in the outlined four areas *C1 - Access Control*, *C2 - Privacy*, *C3 - DoS* and *C4 - Sensing*. Sometimes, existing work in the acoustic domain does not use a PVA as example system when describing or evaluating work. However, when it is clear that the findings can be directly applied to the PVA domain we also include such work.

For example, work on speaker identification is generally not described in the context of PVA but is clearly a technology that could be included directly in a PVA's processing chain to facilitate authentication.

3.1.6 Naming Convention

When we describe existing work we try to use an easy recognisable term. Many authors have given their system, attack or countermeasure a distinct name and in these cases we use this established term together with the year they were published to refer to the work. Examples are DolphinAttack'17 [158], SonarSnoop'18 [30] or CommanderSong'18 [156]. If the authors have not introduced a specific name in their work we use the combination of the lead author's surname and the year of publication as reference (e.g. Carlini'16 [23] or Cheng'19 [29]).

3.2 Related Work

3.2.1 C1 - Access Control

In category *C1 - Access Control* we describe work that broadly relates to the large research areas of voice authentication and command injection. Voice authentication (Category C1.1 in this chapter) is a mature research field and its results are now slowly applied to the PVA domain. Injection of (hidden) voice commands (Category C1.2 in this chapter) is a relatively recent research field, triggered by the development of modern ASR systems and the appearance of PVAs. It has to be noted that methods of voice authentication are not designed as a direct countermeasure to voice injection, however, it might be useful in this role.

3.2.1-A Voice Authentication

Voice Authentication (VA) (also known as speaker verification, speaker authentication and ASV [92]) is increasingly used on smart devices. Voice, among other biometric modalities such as fingerprint, facial and iris, has been widely adopted due to two reasons. Firstly, it can be carried out remotely over communication channels [160], and, secondly, it represents a very natural way for users to interact with machines [92]. The goal of VA is using voice features to identify (or verify) the identity of a speaker. Analogous to a fingerprint the voice profile can be referred to as a voiceprint. It is solving the problem of "who is speaking", in contrast to ASR which tries to solve the issue of "what was spoken".

Some PVAs such as Siri already apply VA and the device is not triggered if the voiceprint does not match a legitimate profile. However, VA is not used by all PVAs. For example, the Google Assistant can still be triggered by a stranger. Smart speakers usually only check if the keyword semantic content is correct (is "Alexa" or "Hey Google" being said?) but do not verify the voiceprint. Even though both, Amazon and Google support voice recognition as reported [136, 61], these features are not used for access control but only to provide a more personalised service by linking commands to user profiles. For instance, any person or even Text-to-Speech (TTS) speech can trigger Amazon Echo and obtain a response [80].

VA in itself is subject to attacks called spoofing attack where via replay or generated sound an attacker aims to circumvent authentication. Recent work on security of VA is structured around defence mechanisms against spoofing attacks. We distinguish four categories of work: *Acoustic Characteristics*, *2nd Factor Authentication*, *Copy Detection* and *Challenge Response*.

The first group of studies exploit acoustic characteristics of the speech signal for spoofing detection. The acoustic signal is not only analysed for the purpose of ASR and VA. In addition, signal features are used that enable identification of a spoofed signal.

The second group of work uses a second source of information that is then correlated with the speech signal. For example, a camera can be used for lip reading and the result can be compared to the ASR result. To avoid an additional sensor, the acoustic channel may also be used to provide this secondary source of information. For example, speakers and microphones can be used as sonar system to read articulatory gestures (VoiceGesture‘17 [160]).

Copy detection aims at comparing previous voice commands to the currently issued command to ensure each voice command is unique [99, 86].

There is also work that employs protocols between speaker and PVA to ensure that each command is genuine. Such prompted-phrase ASV [126, 106, 73] has attracted much work in the past.

In the next sections we focus on work in the first two categories as these attracted most of the recent work. Simple copy detection is not seen anymore as feasible countermeasure as it is resource intensive to keep copies of previous commands and it is too easy to circumvent this mechanism with now available voice generation techniques. Challenge response protocols can prevent attacks but are seen impractical in a PVA context as they disturb natural speech based user interaction.

3.2.1-A1 Voice Authentication (VA) Mechanism VA uses two steps: enrollment and authentication. During enrollment, the user is asked to say a specific phrase several times and the unique voice features are extracted from these speech samples to form a user-specific model. Then, during the authentication phase, this model is used for comparison with the voice utterance to verify if the current speaker matches the model. There exists two variations in the enrollment and authentication process: text-dependent and text-independent VA. For the former type, both enrollment and authentication phase use the same pre-defined utterance. However, for the latter, user cooperation is not required as any spoken words can be used for enrollment. Text-independent VA is more user friendly as no restriction on the speech content is imposed but it is more difficult to achieve the same good performance as with text-dependent VA.

Note that VA can be used in different ways. Speaker identification aims to identify to which speaker a voice belongs while considering a set of candidate voices. Speaker verification only aims to verify if the voice belongs to the target speaker. Both variants can be implemented using similar techniques.

3.2.1-A2 Spoofing Attacks Four different forms of spoofing attacks [58, 39, 142, 91] on VA are distinguished: replay attacks, impersonation attacks, speech synthesis attacks, and voice conversion attacks [161].

A *replay attack* refers to a playback of a legitimate user's voice sample, pre-recorded by an attacker. This attack type is powerful [137, 82, 40, 58, 72] and easy to execute which is why it has gained much attention in the research community. Numerous countermeasures have been developed to prevent this type of attack as we discuss in the next sections.

If the attack requires a spoken sentence that could not be pre-recorded a simple replay is not possible anymore. The audio signal has to be created, matching the target speaker profile. Impersonation attacks, speech synthesis attacks, and voice conversion attacks can be used for this purpose.

An *impersonation attack* refers to an attacker imitating the target's voice with the aim of fooling the VA without the help of any computing devices. However, this attack type can be defended well with the help of an ASV system using advanced models [13, 58, 12, 161, 59, 101]. Thus, impersonation attacks can be considered not to be a threat on modern VA systems.

In a *voice synthesis attack* the victim's voice is generated from scratch [82, 8]. A recording of the victim's voice is required but the exact phrase used for the attack is not needed.

A *voice conversion attack* relies on the manipulation of a given voice sample to match the target voice [74, 128, 150]. Once the conversion function is learned, the source voice can be modified to be sound like the target [150]. A recording of the victim's voice speaking the exact target phrase is again not required.

3.2.1-A3 Spoofing Detection via Acoustic Characteristics Voice spoofing attacks and countermeasures have attracted a lot of interest in both, research and industry communities. There is a large amount of literature, including surveys, covering this specific area [148, 58, 39, 142, 91, 137, 82, 40, 58, 13, 12, 59, 101, 82, 8, 74, 128, 150, 75]. Thus, we do not aim to discuss exhaustively existing work in this specific area. However, we present the research progress and state-of-the-art in this domain. ASV systems have been proved to be vulnerable to spoofing attacks. To mitigate the spoofing threat countermeasures can be developed that can be integrated in existing ASV.

A common problem is that it is difficult to compare proposed spoofing countermeasures as the different works use different datasets, experiment configurations and evaluation metrics. To address these limitation, ASVspoof [149] has been founded. ASVspoof collects and distributes standard datasets, evaluation protocols and metrics and facilitates competitions in which participants test their proposed algorithms. ASVspoof aims to develop generalised countermeasures effective for detecting various known and unknown attacks by controlling the prior knowledge provided to the competitors [149]. A known attack here means that the competitors are aware of the algorithm used to generate attack samples. Unknown attacks are the cases where the algorithm used to generate the attack signal is not known beforehand and it is therefore impossible to use this knowledge in constructing the countermeasure.

The first ASVspoof challenge was held in 2015 (referred to as **ASVspoof'15** in our taxonomy), and it only focused on voice synthesis and voice conversion attacks. Organisers provided freely accessible datasets generated by 10 different synthesis and voice conversion attack algorithms. Participants submitted their spoofing detection algorithms which were assessed using a provided evaluation metric. The spoofing algorithms used were not accessible to the participants.

The second ASVspoof challenge was held in 2017 (referred to as **ASVspoof'17** in our taxonomy), and unlike the first version of the challenge, ASVspoof'17 focused on replay attack detection rather than synthesis and voice conversion attack. The biggest challenge in detecting replay attacks from acoustic characteristics is the variations in the recorded audio samples. ASVspoof'17 tried to explore the practical limits of replay attack detection and facilitated development of countermeasures robust enough to detect a replay attack in varying acoustic environments. ASVspoof'17 provided a baseline spoofing classifier. Subsequently, ASVspoof organisers published the second version dataset of the 2017 challenge for patching a number of data anomalies in the first version [41], together with new meta-data, updated replay detection performance, enhancements to the original baseline system, and corresponding comparison results.

The third challenge in 2019 (referred to as **ASVspoof'19** in our taxonomy) introduces changes from the previous two events. Firstly, ASVspoof'19 considers both logical access (LA) and in addition physical access (PA) scenarios. In an LA scenario the attack signals are generated by TTS synthesis and voice conversion (VC) techniques and are directly fed to the ASV. In a PA scenario speech is captured by a microphone in a physical and reverberant environment (a real-world setup). This scenario is used to study replay attacks; to distinguish a voice signal from a human speaker and a replay from an loud speaker. Secondly, ASVspoof'19 considers all three types of spoofing attacks (voice conversion, synthesis, and replay attacks). Thirdly, attack samples belonging to these types are generated using state-of-the-art neural network or waveform models, and replay attack samples result from a better controlled process. Lastly, ASVspoof'19 takes tandem decision cost function (t-DCF) as the primary new evaluation metric and uses Equal Error Rate (EER) as the secondary one. EER refers to the Spoofing Countermeasure (CM) operating point where the CM miss and false acceptance rates are equivalent. t-DCF aims to assess the pooled performance of the tandem system consisting of CM and ASV [35].

The new metric mechanism ensures that the evaluation scores do not only appraise how well the spoofing attack detection is, but also the impact of spoofing and countermeasure on the ASV system. There were 63 teams in ASVspoof'19, and more than half of them report a spoofing detection system better than two provided baseline systems. In the LA scenario, the best result of t-DCF of 0.0069 and EER of 0.22% are achieved by one of the teams. In the PA scenario, the best system achieves a t-DCF of 0.0096 and EER of 0.39%. For example, an EER of 0.39% means that in 0.39% of cases the system rejects a genuine command as it falsely assumes it is spoofed. At the same time 0.39% of spoofed commands are accepted as they are not recognised as spoofed.

The ASVspoof challenge is focused on evaluating the performance of state-of-the-art countermeasures against spoofing attacks proposed by the contestants. As mentioned, ASVspoof does not directly consider the PVA application context. Next we describe the specific work VMask'20 [159] as it represents the first practical adversarial example attack targeting the ASV system in a PVA environment.

VMask'20 [159]: This work proposes VMask which generates speech samples by adding perturbations that are verified as belonging to a target speaker (the victim). The altered speech samples still sound as if they are spoken by the original speaker. VMask restricts the magnitude of perturbations and applies psychoacoustic masking (see also Schönherr et al. [118]). Note that the main difference between this paper and papers introduced later in Section 3.2.1-B3 is that the adversarial examples here aim to mislead the speaker verification, while adversarial examples in the latter sections aim to mislead ASR.

This work first achieves a grey-box attack. Grey-box means that the attackers do not have knowledge of the target ASV structure and parameters but they are able to obtain verification results and prediction confidence scores. Based on the difference between multiple confidence scores from multiple enquiries to the ASV, a gradient is calculated and used to update the adversarial perturbations. Psychoacoustic masking is applied to restrict the perturbations that are applied.

The grey-box attack proves the efficacy of the approach of adding perturbations. The work then presents a black-box attack attempt where the internal ASV structure is unknown and confidence scores cannot be obtained. It is assumed that despite different ASV model details, they all share a common property which is to project the high dimensional speech space to a low dimensional speaker space (A similar limited set of features is used for authentication). Thus, adversarial examples created using a white-box ASV system may be able to work as well on another unknown system. VMask trains a specific model (in this case a Convolutional Neural Network (CNN) model) which extracts a voiceprint from audio samples. The process is usually called speaker embedding as embedding vectors are extracted from the speaker's utterances by the ASV model. The speaker profile of the victim (the victim speaker embedding) is extracted using this model. Finally, starting from an audio containing the authentication keyword (e.g., Alexa) perturbations are added with the guidance of the target victim's embedding and a psychoacoustic model.

The approach is evaluated using the black-box attack on the Microsoft Azure Speaker Verification API. Four participants speak 10 verification phrases. For each phrase, one speaker is chosen and adversarial examples are crafted to imitate three other speaker. Success is counted if they can pass once for each phrase. A 70% success rate is achieved. To further testify the practicability, the attack is evaluated using five Apple devices with the aim to fool Siri to switch on an Aqara smart LED bulb. An average success rate of 67.5% is achieved.

This work demonstrates that ASV as method of access control for a PVA can be circumvented.

3.2.1-A4 Spoofing Detection via 2nd Factor A second data source can be used to detect spoofing. In many cases the extra information is not obtained from the acoustic channel (i.e. a camera or RF transceiver might be used). However, to avoid an additional sensor, the acoustic channel may also be used to provide this secondary source of information. We first summarise briefly work that relies on additional data obtained from outside of the acoustic space. Thereafter we give details on work that are entirely based on data from the acoustic domain.

Chen'17 [27] observes that all machine-based voice impersonation attacks require a loudspeaker to play voice samples for spoofing attacks. As most loudspeakers use magnets to generate sound a magnetic field can be observed. It is assumed that the presence of a varying magnetic field indicates a machine-based voice instead of a human speaker. As the magnetic field declines quadratic with distance the approach is distance limited.

Feng'17 [46] executes continuous authentication by matching body vibration against the voice signal received by microphones of a PVA. The researchers use an accelerometer to collect body vibration data. The work has limitations as a speaker must wear an accelerometer.

Wang'19 [140] uses low-cost motion sensors to capture the audio-induced vibration. The signature is extracted from the sensor data and machine learning methods are used to detect illegitimite voice commands. The approach requires deployment of motion sensors.

Pradhan et al. [103] determine a speaker's breathing rate using WiFi channel characteristics for liveness detection. Their replay attack detection utilize information from both the acoustic and the WiFi channel in tandem.

The startup company Liopa developed a optical lip reading technology which can be used to correlate the output of an ASR with the output of the lip reading system for authentication purpose [83].

VoiceLive'16 [161]: This work introduces a liveness detection method called VoiceLive to defend against replay attacks aiming at voice authentication on smartphones. VoiceLive works based on a user's unique vocal system and stereo recording on a smartphone for capturing Time Difference Of Arrival (TDoA) of phonemes. It captures TDoA changes of a sequence of phoneme sounds to the two microphones of a smartphone, and calculates the TDoA similarity between the captured samples and that of the stored ones to detect if a replay attack is in progress. This method is supposed to work because the TDoA difference phenomenon between phonemes does not exist for replay attacks. However, one limitation of this paper is that it requires a user to hold the phone close enough to the mouth, otherwise it cannot work reliably. TDoA values are different for various phonemes because of the voice generation principles. A phoneme is the smallest distinct unit sound. Different vowels are generated mainly due to different tongue positions, and different consonants can be produced by combinations of articulation place and manner. In a nutshell, these factors result in unique sound origins for each phoneme spoken by a person. These discernible sound source locations lead to unique TDoA for a phoneme. VoiceLive uses the stereo recordings to obtain a sequence of TDoA values (acquired through word recognition, phoneme and segmentation), then compares these values with the corresponding ones stored at the training stage to flag a replay attack. The comprehensive experimental results show the viability and generality of the method by considering different phone models, phone placements and sampling rates. VoiceLive is used to detect two types of replay attacks: playback attacks and replace attacks. Attackers launch a playback attack by playing pre-recorded samples using a speaker in front of the microphone (Identical to a replay attack as defined in Section 3.2.1-A2). A replace attack means the adversary replaces the malicious samples by the original ones before or during transmission by leveraging a virtual recorder to bypass the real microphone or by intercepting and replacing the original speech signal during transmission. The evaluation with 12 participants reveals that overall VoiceLive achieves over 99% detection accuracy with 1% EER.

The VoiceLive approach is limited to close range and a speakers mouth must be in front of the microphone. However, it might be possible in some situations to use this technology in a more generic PVA environment.

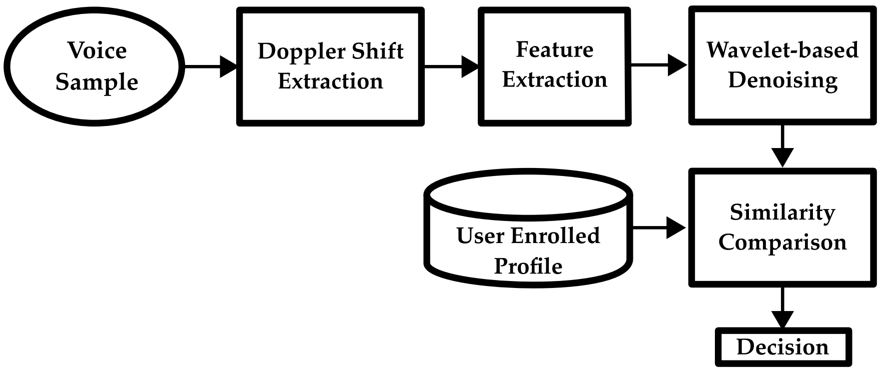


Fig. 3.2 Working flow of VoiceGesture’17: steps to extract features reflecting articulatory movements and to compare them against ones stored to decide which phonemes are being spoken

VoiceGesture'17 [160]: This work introduces another liveliness detection method called VoiceGesture to defend against replay attacks aiming at voice authentication system on smartphones. VoiceGesture uses the smartphone audio system (speaker and microphone) as a sonar to detect the unique articulatory gestures of a user when they speak a passphrase. VoiceGesture is a better liveliness detection system than VoiceLive'16 in the regards of usability for users because VoiceGesture not only supports holding the phone in front of the mouth, but also works when users hold the phone at their ears. VoiceGesture is also less susceptible to environmental noises.

When a person speaks a phoneme, multidimensional articulatory movements called articulatory gesture (lip protrusion, lip closure, tongue tip and tongue body constriction, and jaw angle) are involved during the process. Even if same type of articulator gesture is used, the movement speed and intensity may be different for different phonemes. VoiceGesture emits a 20 kHz sound wave and records all of the reflections from articulators using the built-in speaker and microphone of the phone. Analysing the spectrogram of the recording, the movement of articulators can be revealed. Articulator movement velocity is related to the frequency shifts. The energy level shown in the spectrogram relates to the distance between the articulators and the microphone. Because mainstream loudspeakers rely on a diaphragm which only moves in one dimension and the slight differences in articulatory gesture among people when saying the same phoneme, VoiceGesture can be used as a voice authentication system which is robust against replay attacks. The procedure (shown in Figure 3.2) for VoiceGesture is: the Doppler shifts for each phoneme in the spoken utterance are extracted in Doppler Shift Extraction stage; then frequency and energy distribution features are extracted from these Doppler shifts; wavelet-based denoising technique is applied to remove noise from these features and reconstruct them; finally features are compared with the one stored during enrolment for liveliness detection.

The comprehensive experimental evaluation shows the effectiveness of the system against playback attacks and mimicry attacks. A playback attack here is what we defined as replay attack in Section 3.2.1-A2, a speaker is used to play the passphrase. A mimicry attack is launched by playing a pre-recorded passphrase using a speaker while the attacker mimics the articulatory gestures of the victim. Overall, an over 99% accuracy and EER of 1% are achieved.

The VoiceGesture approach is designed for the mobile phone context. However, it should be possible to use this technology in a generic PVA environment. The evaluation is limited, no test with piezoelectric speakers is carried out and a more diverse set of test subjects is necessary (limited to young and educated people).

We list this work here in category *C1.1.2 - 2nd Factor Authentication* as this is the main objective of this work. However, the work is also related to *C4.2 - Active Sensing* as an active acoustic sensing signal is used.

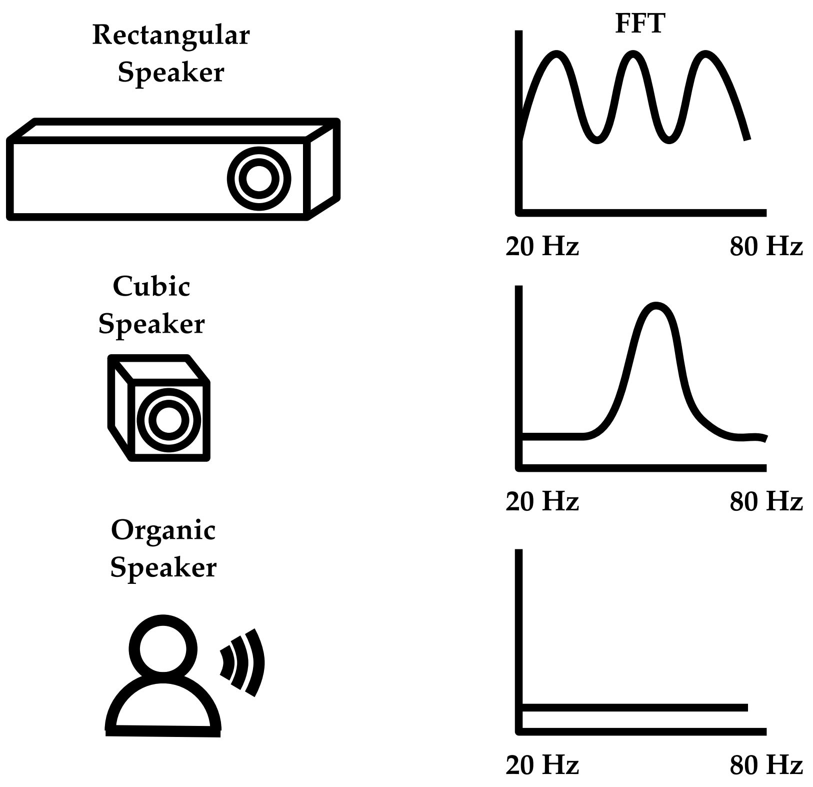


Fig. 3.3 The key idea in Blue'18: audio played by commercial off-the-shelf (COTS) speakers show distinctive energy within the sub-base frequency range while there is no such phenomenon for voice of human speakers

Blue'18 [19]: This work introduces a detection method to differentiate commands spoken by a human from the ones played by an electronic speaker.

This method explores the fact that over-excitation exists in the sub-base frequency range (20 - 80 Hz) of audio signals generated by off-the-shelf speakers, which does not apply to a human voice (shown in Figure 3.3). By identifying the rather significant energy in the sub-base area, commands injected via electronic speakers can be detected. The sub-base over-excitation is caused by the resonance of the enclosure material or case of the speaker. Although resonance of material is an unavoidable fact, the resonance frequency could be affected during design process. These enclosures resonate in the sub-base (20 - 80 Hz) region which is designed by engineers on purpose as it is such a low frequency area that the sound would not be heard by humans. Whenever the speaker produces sound, the enclosure resonates. Detection of this phenomenon composes of two steps: (i) audio preprocessing; (ii) metric construction.

During the first step, recorded audio is processed with equalising, amplitude normalisation and noise filtering. For the second step, the pre-processed audio will be subject to: a) windowing which makes the detection be robust against the sub-base variation with respect to time when different phoneme is spoken; b) 4096 FFT computation followed by averaging and cropping, limiting to frequencies between 20 and 250 Hz; c) generating an energy value by integrating the FFT curve; d) defining the cutoff frequency point to identify the sub-base area, which generates the energy balance metric representing the percentage of energy in the sub-base; e) Removing outliers based on the skewness of the data.

Using these steps to calculate the sub-base energy level (the energy balance metric) of samples from electric speakers and human speakers, the authors then set the threshold for the metric by analysing the ROC curve and the trade-off between the True Positive Rate (TPR) and False Positive Rate (FPR). If the metric value for an audio is higher than the threshold, this audio will be labeled as being from an electronic speaker. During the evaluation, the threshold is chosen to ensure 95% TPR while minimising the FPR to be not greater than 5% even in high noise environments, additionally, with a 100%/1.72% TPR/FPR in quiet environments. It is also shown that the detection is effective in defending against obfuscated hidden commands injections as in Carlini [23] (see Section 3.2.1-B2). All commands (spoken by human or played by speakers) are recorded by a microphone array instead of a smartphone or a smart speaker. However, the authors claim that the array is similar to the one on a smart speaker such as the Amazon Echo.

This method of spoofing detection is a useful approach for the PVA context and has less limitations regarding usability compared to VoiceLive'16 and VoiceGesture'17. However, the defence cannot work if the adversary alters the physical design of the speaker used for attack rather than just using a COTS.

Lee'20 [79]: This work proposes a sonar-based framework to defend against remote attacks targeting PVAs. Remote attacks here means adversaries hack a network-connected smart devices (like TV or radio) equipped with speakers to play malicious voice commands (audible or hidden commands as introduced in Section 2.4).

The core idea is utilising liveness detection to check if the direction of voice commands is consistent with the human speaker position. The work assumes that speakers are willing to compromise usability for security as voice commands need to be spoken within a 2-meter radius from a PVA.

The framework consists of four parts: the Spectrum Preparation module creates the crafted high-frequency signal bins and performs Fast Fourier Transform (FFT) and windowing on the received echos; the User Direction Analyser searches Doppler shift to detect the occurrences of movement. The direction of such movements is calculated via TDoA. Command Direction Analyser uses TDoA to obtain the voice command direction. Finally, Consistency Checker module compares two directions and decides whether to accept or reject the commands. A prototype Speaker-Sonar is built using a raspberry pi 3b, a commercial microphone array and a speaker to mimic an Amazon Echo. The experiment results show that the prototype can reject remote attacks at 95.5% success rate.

This work shows a general defense mechanism for most attacks described in Section 3.2.1. The limitations are that a hardware modification is needed and that the user needs to stay close to the PVA. Again, the work is also related to *C4.2 - Active Sensing* as an active acoustic sensing signal is used.

3.2.1-B Hidden Voice Commands

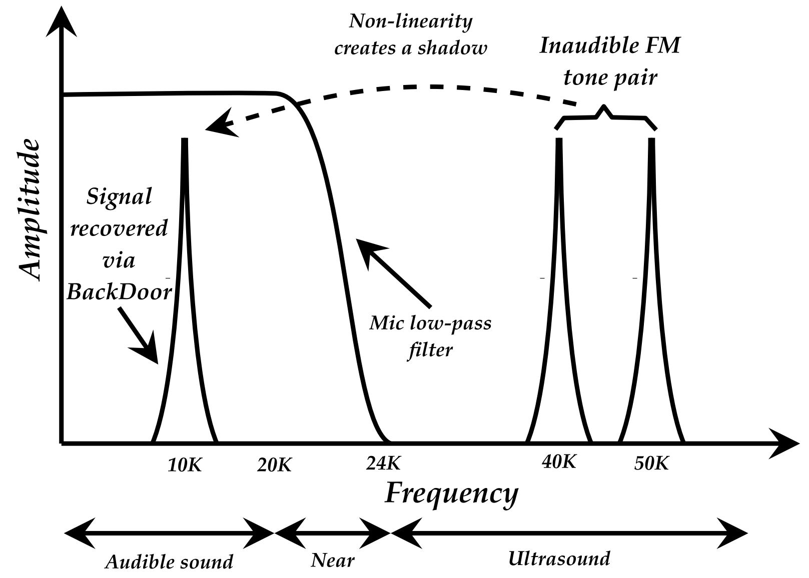


Fig. 3.4 How non-linearity recovers the signal carrying attack messages in BackDoor‘17.

3.2.1-B1 Hardware Non-linearity BackDoor’17 [111]: This work proposes to exploit the non-linearity of microphones to transfer human inaudible but microphone recordable sounds. This technique is harnessed for three purposes: (i) to provide an acoustic but inaudible channel to a microphone; (ii) to achieve true high data rate (4kbps at 1 metre and

2kbps at 1.5 metre) inaudible acoustic communication; (iii) to provide room-level privacy protection via jamming. Although the work has a strong focus on jamming we report the work here and not in category *C3.2- Jamming* as it also is the first work describing the use of the audio non-linearity property.

This work describes inaudible command injection by exploiting the technique described in Section 2.4.1 of Chapter 2. A "shadow" signal in the audible frequency range is recorded. At first, this work considers amplitude modulation to modulate attack signals onto a high frequency carrier signal. By doing so the attack signal can be demodulated without additional software and is recovered at the microphone due to the non-linearity properties. However, the attack signal will also be audible when the modulated signal is played by the speaker as the non-linearity feature also applies to the speaker. This work applies therefore Frequency Modulation (FM) to address this problem. The signal is modulated with the attack signal using FM and in addition another high frequency signal is also needed as shown in Figure 3.4. The work further discusses how to reduce the ringing effect; the FM signal is slightly audible due to this effect. To decode the attack signal, additional software is necessary at the receiver to demodulate the FM. It has to be pointed out that the speaker non-linearity is not only central to this attack, it also makes it difficult to construct a fully inaudible signal.

Jamming described in this work uses two methods. It is shown that by adding an inaudible high frequency signal component with a high volume the Automatic Gain Control (AGC) of the microphone can be exploited. The AGC regulates the volume of the entire signal based on the strongest signal component. Thus, the volume is reduced based on the high frequency signal component which leaves the low frequency components containing voice too weak to be processed. In a second jamming approach, noise is added to the inaudible high frequency range which is then demodulated by the non-linearity behaviour of the microphone and distorts the audible sound range. Thus, the ASR is unable to decode speech.

In this work, the researchers consider jamming to be used as a DoS attack in an untrusted environment. The work also describes how non-linearity features can be exploited for inaudible communication. This work contributes significantly to two areas (C1.2.1 and also C3.2), however, it is shown in the taxonomy (see Figure 3.1) under category C1.2.1. It has to be noted that this work requires additional software to decode the attack signal and cannot be directly applied to a PVAs. It requires specific transmitters as well.

DolphinAttack'17 [158]: This work investigates injection of inaudible commands into PVAs by using the same idea of exploiting non-linearity features of microphones as BackDoor'17 [111]. The work differs from BackDoor'17 in two main aspects. Firstly, the messages modulated to the carrier are audio commands (not just simple signalling tones). Secondly, DolphinAttack applies only amplitude modulation (AM) to modulate the base-band attack commands on the ultrasound carrier signal. The attack message can be demodulate to the base band and recovered just by the non-linearity feature of microphones. Because no additional demodulating software is needed, the attacker is capable of achieving a hidden attack towards an off-the-shelf PVA.

To make sure the recovered commands are to be interpreted by PVAs, this work generates inaudible attack samples for both the wake word commands for PVAs and general audio commands with the technique outlined in Section 3.2.1-A2. As some PVAs such as Siri apply VA, the activation commands can only be effective if it contains the wake word and if it contains similar features to the voice of the legitimate user. This work tries to compose the wake word using TTS-based brute force and concatenative synthesis to satisfy this requirement.

For experimental evaluation a powerful transmitter is used (a smartphone as signal source, a high-end signal generator for modulation and a high-end ultrasonic speaker). An empirical study is carried out to test the attack on all major PVAs publicly available using the same experimental setup, including Siri on 10 various Apple devices, Google Now on two Nexus models, S Voice on a Galaxy smartphone, HiVoice on a Huawei smartphone, Cortana on a ThinkPad laptop, Alexa on Amazon Echo and an Audi Q3 automobile PVA. For PVAs supporting wake word detection, inaudible activation commands are used for the attack, furthermore, a set of commands such as starting a FaceTime call on an iPhone, turning on the airplane mode on a Galaxy S6 Edge, and manipulating the navigation system of an Audi car are chosen for the attack to test if the ASR systems can correctly recognise the commands. Results show that wake word detection and command recognition attacks are all successful on all of the PVA systems. However, different PVA systems do require various parameters to ensure the success of these attacks.

The work also investigates both, hardware and software defence solutions. The former includes suppressing the ultrasound frequency sensitivity of COTS microphones, and an extra module to detect the modulated attack signal for canceling the attack messages. The latter defence utilises Supported Vector Machine (SVM) to classify DolphinAttack and benign audio samples based on the fact that they show different frequency domain characteristics from 500 to 1000 Hz.

The main limit of this work is the requirement for very high-end equipment. Also, the attack distance is not very large (about 175cm).

Roy'18 [112]: This work builds on DolphinAttack'17 [158] and aims at injecting commands into PVAs. DolphinAttack'17 has a limited attack range of 5 ft (175 cm, roughly 5 ft). This work introduces methods to increase the attack range to 25 ft while maintaining inaudibility of commands. Also, this work proposes a first step towards defence by proposing methods to detect non-linearity traces. To achieve the increased attack range, more power is required at the speaker. Due to the existence of non-linearity in speakers, attack messages $m(t)$ using AM can be heard when they are played. This is more obvious when more power is used for increasing the attack distance. This work addresses this by separating parts of the AM attack signal to multiple speakers each of which only plays a certain frequency segment of the original attack signal. By doing so the attack message $m(t)$ does not exist and the audio leakage $m(t)^2$ per microphone is energy and bandwidth limited. To ensure the signal addition is still inaudible, this work applies a psychoacoustic model to ensure that sound intensity is below the hearing threshold depending on frequency to maintain inaudibility. An audio segment partition algorithm is used to find the best way of segmenting the attack signal.

This work further discusses possible defenses; spectrum correlation between the sub-50 Hz band and the spectrum above 50 Hz is used. This correlation exists because self-convolution (caused by non-linearity) of signals from fundamental frequency band and harmonics would generate signal copies.

With this approach it is feasible to inject inaudible commands into a PVA. However, specific transmitter hardware and software is still required.

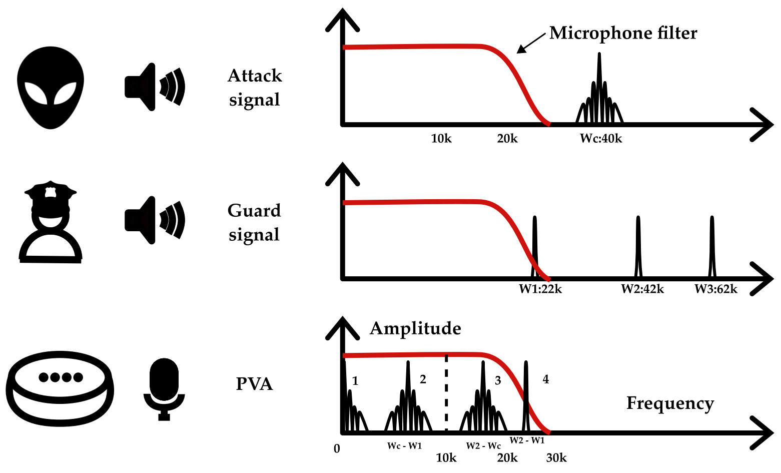


Fig. 3.5 Mix-frequent signals generated due to non-linearity effect on the attack and the guard signal which are utilised for later attack signal cancellation in He'19

He'19 [60]: This research contribution provides a novel defence against attacks exploiting the microphone non-linearity for hidden voice command injection. Limitations in the defence mechanisms proposed by DolphinAttack'17 [158] and Roy'18 [112] are discussed. A method called Active Inaudible-voice-command Cancellation (AIC) which detects and cancels out attack signals while maintaining the legitimate command is presented.

The work outlines the software defence proposed by DolphinAttack'17 which utilises SVM to classify malicious and benign audio samples based on their difference in their frequency range. Then it shows how to bypass the defence using a filter with inverse frequency response of the filter in ultrasonic speakers used for the DolphinAttack'17 to hide the features. The researchers then point out the principles of defence in Roy'18 and then shape the attack signal by adding custom signal components and noise to successfully mitigate amplitude skew, power concentration in the sub-50 Hz area and signal correlation, thus bypassing the defence described in Roy'18.

Figure 3.5 illustrates their defense mechanism. An attack signal is modulted to 40kHz to attack a PVA making use of non-linearity of the microphone. The authors introduce a guard signal which is designed as a multi-tone signal with a 20kHz interval, placing bins at W1:22kHz, W2:42kHz and W3:62kHz to frame the attack signal copy to be lying within 10kHz to 20kHz (referred to as Signal 3 in the Figure), which will be used as a reference signal to cancel the recovered attack command (Signal 1) and Signal 2 which is an accompanying result due to the design of the guard signal. Going through the inherent non-linearity feature, these guard signals together with the inaudible attack signal generate mixing frequency components (which is modeled by the second order component in Equation 2.3) including recovered attack message (Signal 1) and other components (Signal 2 centering at Wc-W1 and Signal 3 centering at W2-Wc). These two signals contain the attack message as well, which are further utilised to cancel out the attack signal. The cancellation following the detection is modelled and inspired by Active Noise Control (ANC) techniques. In general, the defence depends on using two parts (part A on the left side of 10kHz and part B on the right side) of the newly generated mixing frequency signals (all components are shown in the last Amplitude-Frequency coorinate) to cancel each other. The core of signal cancellation is to create an exact copy (Signal 3) of the target signals (Signal 1 and Signal 2) and to use the copy to subtract from the target. Signal processing techniques like filtering and normalisation are applied to extract Signal 3 from all of the components and execute the cancellation of Signal 2 and Signal 1. The attack signal and the side-effect signal from the guard signal are all neutralised after the these AIC procedures.

Because the attacker may be aware of the proposed defence and apply more sophisticated attacks, the authors discuss possible attacks: (i) Frequency hopping attacks aim to compromise the convergence of the adaptive filter of the AIC as hopping between frequencies results in not having enough time for the filter to adapt to the optimal parameters; however, this can be detected and cancelled using pre-calculated parameters for the adaptive filter. (ii)

Table 3.1 Comparison of hardware non-linearity work (Category C1.2.1). *Modulation*: modulation techniques used to add attack commands to the high-frequency carrier signal. *Demodulation*: additional software needed at the receiver to recover the attack commands. *Distance*: the longest attack distance achieved. *Defense*: defenses against the proposed attack (Hardware and/or software defense methods).

Papers	**Modulation**	**Demodulation**	**Distance**	**Defence**
BackDoor'17	frequency modulation	software needed	150 cm	N/A
Dolphin-Attack'17	amplitude modulation	autonomous	175 cm	HW/SW
Roy'18	amplitude modulation	autonomous	762 cm	SW

Selected frequency attacks. This attack manipulates the attack signal frequency to make B appear at a certain frequency which deteriorates the cancellation efficiency. (iii) Guard signal cancellation attack aims to cancel the guard signal, however it is impractical as it is extremely difficult to measure characteristics of the guard signal due to unpredictable factors such as noise and fading; (iv) Noise injection attacks aim to pollute C to prevent AIC to extract B for further cancellation. However, noise injection can make the attack audible or can affect the correct transcribing of attack messages.

The prototype implemented is tested with comprehensive experiments to show that the design is valid. The main disadvantage of this approach is that this defence needs a specific speaker array for emitting the guard signal.

Comparison: A comparison between these studies on hardware non-linearity is shown in Table 3.1. *Modulation* indicates the modulation technique used to modulate the attack commands to the high-frequency carrier signal. *Demodulation* describes if additional software is needed at the receiver end to recover the attack commands. *Distance* shows the furthest attack distance achieved. *Defense* shows if defenses against the proposed attack have been discussed. If so, are these potential defenses implemented on software or hardware level? Note that He'19 is a work fully focusing on defense and it is therefore not included in the table; this work requires an additional transmitter and software update on the PVA to achieve protection.

3.2.1-B2 Obfuscated Commands **Cocaine Noodles'15** [134]: This work examines the gap between the human auditory system and modern ASR in terms of decoding speech. The work received its name from an online post on Reddit, where it is pointed out that the phrase 'Cocaine Noodles' is often misinterpreted by Google Now as its wake word 'OK Google'.

The researchers successfully craft speech signals which are recognised by the ASR but are unintelligible for humans. The acoustic features are extracted via MFCC from the raw audio input and are then fed into the following acoustic model. The MFCC can be tuned using a variety of parameters. It is possible to change these parameters such that the ASR still recognises the correct command (four parameters are used). By defining a reverse MFCC function the extracted features are translated back into an audio signal. This audio signal is unintelligible for humans but the MFCC produces a feature set that results in the desired command being recognised.

The work uses Google's Speech Recognition engine through an Application Programming Interface (API) to generate the mangled audio examples. Then these samples are tested on Google Now on a Samsung Galaxy S4 smartphone by being played over the air in a large room with 50 dB background noise. The work also describes a user study on the Amazon Mechanical Turk platform to test how humans perceive the crafted signals. The results show that these crafted signals can activate commands while they are difficult to understand.

Carlini'16 [23]: This work continues the work of Cocaine Noodles'15, exploring how ASR can be attacked by hidden voice commands which are able to be interpreted by PVAs but unintelligible to human beings.

The work reproduces the black box attack as describe in Cocaine Noodles'15. Furthermore, the work describes a white box attack showing that by knowing the parameters of the system, commands that are better hidden from human ears can be crafted. Finally, defence mechanisms are evaluated.

The black box attack uses similar mechanism as the work in Cocaine Noodles'15 but considers more practical settings and targets a newer Google ASR. The white-box attack targets the CMU Sphinx which uses GMM as the model to assign probabilities given a piece of audio frame corresponding to a given phoneme. With the knowledge of the target ASR, first an attack similar to the black-box attack is launched by targeting specific MFCC vectors. This attack in refined in two aspects. First, instead of MFCC vectors, the attack aims at attack phrases and thus target phonemes and target HMM states, which enables the flexibility of different MFCC. Second, as few frames per phoneme as possible are used for crafting the attack input signal which prevents human understanding of the mangled speech but causing not much distortion for the ASR. Starting from a target phrase and its corresponding HMM states, the target MFCC output vectors are found, and then gradient descent is used to generate the audio input which is the desired obfuscated command. The evaluation shows the new obfuscated commands can be recognised by a machine at a rate of 82% but with a rate of nearly 0% by humans.

The work examines defence mechanisms against this attack method. Firstly, notifying the user with sound, vibration or flash whenever the device receives a voice command. This can be easily implemented but users may become accustomed to the alerts and ignore them. Secondly, defence with challenges such as audio Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) which should be understood and answered by human easily but would be difficult for an ASR. Unfortunately, the authors examined two audio CAPTCHA choices and found them unsuitable as they can be solved using ASR. More efforts are required in this line of work. Thirdly, detection techniques including speaker verification, filter and human/machine voice classifier are proposed. The filter method leads to a drop in recognition rate of obfuscated commands from 41.74% to 26.6% while the recognition rate of legitimate commands only drops by 1.06%. A machine learning model trained with normal commands and obfuscated commands from the black-box attack is used to classify human voice and machine-generated obfuscated voice. This model has a 0.8% false positive rate when tested with all legitimate commands. When tested with all white-box generated obfuscated commands, it flags 69.3% of them as malicious ones.

Abdullah'19 [5]: This work proposes a methodology for generating hidden voice commands attacking multiple state-of-the-art ASR and speaker recognition systems combined with multiple acoustic hardware configurations without knowledge of the underlying systems.

To make attack samples applicable for different ASRs, the work focuses on the signal processing phase almost every ASR needs. Signal features are perturbed that are important for the human auditory system but not critical for ASR recognition. This results in obfuscated commands that can still be transcribed correctly but are not understandable to humans. Four aspects of signal features are targeted by the proposed algorithm: Time Domain Inversion, Random Phase Generation, High Frequency Addition, and Time Scaling.

In the evaluation 5 cloud based ASR APIs, 2 cloud based speaker verification APIs, and 5 local ASR are tested. Over-the-line attacks (i.e., directly feeding the attack example to the ASR modules) and over-the-air attacks (i.e., playing the audio through loudspeakers) are tried. In both scenarios, almost all phrases were successfully transcribed and speaker identification models made the correct classification. However, the evaluation misses to quantify how unintelligible the perturbed examples are for humans.

Comparison: A comparison of these studies on obfuscated commands is shown as Table 3.2. Note there are some common features of these works, so we do not include them in the table. These features are: the obfuscated commands are all full sentences, their attacks have all been tested over the air in a room with echos.

Table 3.2 Comparison of ASR obfuscated commands work (Category 1.2.2). The *ASR* column includes *Type* (the technique used by the targeted ASR) and *Model* (the targeted ASR name). *Context*: whether black-box or white-box ASR is assumed. *Generation Tech*: specific techniques used. Gradient Descent (GD) refers to Gradient Descent. *Perception Measurement*: how the obfuscation distortion is measured.

Papers	**ASR**		**Context**	**Generation Tech**	**Perception Measurement**
	Type	Model			
Cocaine Noodles'15	unknown	Google Now	black-box	MFCC tuning	user study
Carlini'16	unknown & GMM-HMM	Google Now & CMU Sphinx	black-box white-box	MFCC tuning & GD	user study
Abdullah'19	multiple	multiple	black-box	acoustic features tuning	psychoacoustic knowledge

The first part of the table is the *ASR* column including sub-columns *Type* and *Model*. *Type* indicates the essential technique the ASR utilises. *Model* shows the name of the targeted ASR. Abdullah'19 test their work with various proprietary ASRs online through API and locally. *Context* points out whether black-box or white-box attacks are used in the study. *Generation Tech* indicates what core techniques are used to generate the obfuscated commands and what the objective is during generation. If only experimentally adjustment is used, information is not displayed in the table. This is the case for Cocaine Noodles'15 and Abdullah'19. *Perception Measurement* describes how the obfuscation distortion is measured.

3.2.1-B3 Adversarial Commands **Iter'17** [67]: This work shows that it is possible to generate adversarial commands which fool ASR systems by using methods usually applied to image recognition. The attack assumes the ASR to be a white box and targeting and non-targeting attacks are considered. Fast Gradient Sign Method (FGSM) and Fooling Gradient Method (FGM) algorithms for producing adversarial commands are proposed.

FGSM is a linear perturbation algorithm which adds imperceptible small vectors whose elements are equal to the sign of the gradients of the cost function with respect to the input [54]. It is used to achieve an aimless adversarial attack which results in misspellings to entire different transcriptions y' compared to the original transcription result y. The introduced FGM corresponds to the method introduced at the beginning of Section 2.4.3.

Targeted adversarial commands T are generated by adding perturbations to the original input x following the guidance of the gradients of the loss function comparing target T and the temporary prediction results which is the original prediction y gradually moving towards the target T due to the added perturbations. The perturbations are small to ensure they are imperceptible. Gradient Method is usually used to train a neural network but in this case the updated objective are no longer the parameters of the network but the inputs of the network. Hence this method is named as FGM.

FGSM is evaluated using one-word examples and full sentences on the ASR model called WaveNet [70], pre-trained with the VCTK Corpus. In the evaluation with one-word examples, 93% of the tested examples have been changed (the transcription differs) after 10 iterations of applying FGSM. When considering sentences, 64.06% of the examples are changed after 10 interactions. FGM is tested with one-word and three-word short sentences extracted from the VCTK corpus on a WaveNet pre-trained model. Only three adversarial commands are generated. Specifically, 3000 iterations are required to fool the ASR to recognise "rainbow" to be "thunder", while only 800 iterations needed to fool a three-word sentence. This shows that generating longer adversarial commands does not suffer from scaling issues. It is worth noting the WaveNet model accepts MFCC as input and experiments all use the MFCC of the original audio input. After a new MFCCs is generated, it needs to be converted to an audio signal, the adversarial command. As this is a lossy transformation the converted audio input is different from the original input (conversion loss + perturbation). To ensure similarity between the reconstructed audio and the orginal, an approximate inverse transformation function is used. A user study or any metric to measure the perception distortion is missing from this study. Thus, it is not entirely clear how 'hidden' the produced adversarial commands are.

Alzantot'18 [11]: This work introduces a method for generating one-word targeted adversarial examples assuming a black-box ASR. The proposed algorithm is a gradient-free genetic algorithm without requirement of the target ASR structure and parameters.

The algorithm proposed in this work takes an original audio sample x and the target command y' as inputs, then adds perturbations to x to generation a population of intermediate adversarial commands. Based on the prediction scores of the ASR for these intermediate commands, the algorithm picks candidates that fit best the target command as the base of the next generation round. The selected intermediate adversarial commands are mixed to generate a new *child*. The process then starts again using this child as new input. This process repeats for a predefined number of rounds or until the attack is successful.

This work uses the Speech Commands classification model [114] used for keyword spotting, written using Tensorflow [2] based on convolutional neural networks. The speech commands dataset [143] is used for evaluation. Choosing 500 audio clips from this data set with 50 clips for each keyword label, their attack is successful with an average rate of 87%. 89% of the participants in the user study cannot differentiate the adversarial command from the original command.

Carlini'18 [24]: This work successfully constructs white-box targeted adversarial commands for the DeepSpeech ASR [57], a state-of-the-art transcription neural network implemented by Mozilla. A small perturbation is added to the audio input, resulting in an audio signal which is over 99.9% similar to the input. This is the first robust targeted adversarial attack study, resulting in audio output can be influenced such that theoretically any chosen phrase can be transcribed (with limitations such as length difference between original and adversarial commands).

This work formulates the generation of the adversarial commands as an optimisation problem in order to find the minimum necessary perturbations. Connectionist Temporal Classication (CTC) loss is used as the loss function; CTC is a method which maps input sequences to output sequences according to probability, and alignment between two sequences beforehand is unnecessary as it is embedded in the algorithm. Initially, the optimisation process may not be converging and the perturbation value at first is not minimised but instead the ceiling value of the allowed perturbation is gradually reduced in each iteration. The gradient of the loss function with respect to the initial audio sample but not the Mel-Frequency Cepstrum (MFC) result is calculated. The specific gradient algorithm applied is called Adam [71]. Furthermore, the optimisation covers the complete audio sample simultaneously which means perturbation to all frames of the sample are added, in contrast to only calculating a perturbation vector for a single frame at a time.

The first 100 instances of the Mozilla Common Voice data set are used to construct adversarial commands to attack DeepSpeech. For each instance, 10 different incorrect random transcriptions are chosen as the target. This initial method can generate targeted adversarial examples with 100% success and an average -31dB perturbation. It is found that the longer the target transcription is, the harder it is to create the adversarial command as each extra character requires another 0.1dB in perturbation. Also, the longer the original command, the easier it is to target a given sentence. Some special application cases are also described in the work. For example, it is possible to modify a non-speech signal to be recognised as a target phrase. It is also possible to prevent a command being recognised by adding noise.

CommanderSong'18 [156]: CommanderSong'18 achieves an adversarial white-box attack against the Kaldi ASR by creating modified songs which are perceived by listeners as songs but recognised by the ASR as commands. The robust adversarial commands are played over-the-air and are effective on the ASR. Thus, real-world settings are taken into account in this work.

Kaldi [1, 102] is a representative DNN-HMM ASR as introduced in Section 2.2. It consists of feature extraction, acoustic model and the decoding graph. During the adversarial example generation, the original song and the target audio commands are both fed into Kaldi in parallel. After going through feature extraction and the DNN acoustic model, the acoustic model result of the song is obtained which is in the form of a vector (denoted as y) based on DNN predictions; additionally, the decoding results of the target command in the form of phoneme identifiers are mapped into their corresponding acoustic model result (denoted as the target y'). Then a loss function is constructed as the distance between y and y', and the gradient of the loss function with respect to the input song x is calculated. Perturbations to the song x are calculated iteratively to minimise the loss function while ensuring a perturbation limit. The duration of the added perturbation is also limited to further maintain the fidelity of the original song to human perception. Robust adversarial examples are created that can be played over-the-air following the same gradient method but with an additional noise factor added to the song together with the perturbation. In this way the loss optimisation process can ensure that commander song can still be recognised by the ASR as the noise in the environment has been considered during the generation.

For the evaluation 26 randomly pick songs with 12 common commands are directly fed to the ASR. Over 200 songs are generated and all of them succeed in being recognised by the ASR. Signal-to-noise Ratio (SNR) is used for measuring the distortion and it ranges from 14 to 18.6dB, which is about less than 4% of the orginal song, and it is argued that the distortion is unlikely to be noticed. For the over-the-air experiment, within a meeting room (16x8x4 m^3), 3 speakers play the adversarial commands to a PVA located at 1.5 m away respectively. The PVA is an iPhone 6S for recording, and the audio is sent to the Kaldi API for decoding. With a JBL speaker a success rate of 90% and 96% for two different commands is achieved. SNRs of all the commands are below 2dB. A user study is executed to evaluate the distortion level. For the directly fed adversarial examples, none of the commands embedded in the song can be recognised. However, 28% of the participants realise an abnormality in the song (for the genre Rock Music, for other genres the rate is lower). The rate of detection is higher when evaluating examples crafted to be played over the air. However, embedded command itself is not recognised. Transferrability of the adversarial samples to a black-box ASR iFLYTEK is also evaluated. Directly feeding the sample results in 100% success rate for 2 commands and 66% for the third chosen command "Airplane mode on". Over-the-air the first commands are also recognised but not the "Airplane mode on" command. Their examples cannot be transferred to the ASR DeepSpeech. However, after using the algorithm of Carlini'18 to further modify their adversarial examples they are successfully decoded by DeepSpeech and these samples then also work on Kaldi. Spreading adversarial commands through online platforms and raido are also tested to prove the efficacy. Two defense mechanisms are proposed. First, adding noise before ASR decoding is successful for the direct fed examples when the SNR is lower than 15%. It is not effective for robust over-the-air examples. Second, downsampling is proposed. With a 70% downsampling rate, the two types of commands used can only achieve 0% and 8% success rate while benign commands can still survive with a 91% rate.

Taori'18 [130]: This work introduces a new method for generating targeted adversarial commands assuming a blackbox ASR system. The work improves on Alzantot'18 [11] as a more complex ASR system (i.e., DeepSpeech) is targeted.

This work generates adversarial commands from normal samples of the Common Voice data set such that these are interpreted as two words from the 1000 most common English words. Their algorithm combines the genetic methodology of Alzantot'18 and gradient estimation techniques. The former method is good at selecting candidates from a large mutation space. Once the intermediate examples are close to the target, gradient estimation is used to select more targeted perturbations.

With 100 adversarial command samples generated from the Common Voice data set a 89.25% similarity between decoded text and target text in terms of character distance can be achieved. In 35% of cases the adversarial command can be decoded to be the exact target phrase. An average of 94.6% similarity between the corresponding two audio samples is achieved. This audio similarity is calculated using cross correlation coefficient between the generated adversarial audio examples and the original audio samples.

Khare'19 [69]: This work proposes a new framework using multi-objective evolutionary optimisation to generate adversarial commands for both non-targeted and targeted attacks on a blackbox ASRs.

The adversarial command generation usually needs to achieve conflicting objectives: 1. High similarity between original and adversarial audio samples; 2. Significant dissimilarity between transcription texts between the original and adversarial samples. Two fitness evaluation criteria, namely acoustic similarity and text dissimilarity are used as the guideline to adjust the perturbation added to the original input for generating both un-targeted and targeted adversarial examples. For un-targeted examples, high acoustic similarity and high transcription text dissimilarity between the original and generated examples are pursued; for targeted examples, high acoustic similarity but low text dissimilarity between the transcription of the target and that of the generated examples are to be achieved.

For the generation process, a set of original audio inputs is selected and random uniform noise is added to them as the initialisation. Fitness scores based on these two criteria are evaluated for these initialisation examples. Due to the black-box attack nature, gradients cannot be used and a genetic algorithm is applied to pick good genes from candidate parents examples to generate child examples in each iteration. Multi-Objective Genetic Algorithm (MOGA) is used for the selection schemes to select the mating pool (parents) based on the fitness score. After mating pairs are selected, a crossover operation is performed to generate three children where one child has the characteristics of parents in equal proportion and either of the other two is dominated by that of one parent. Mutation by adding noise is applied to every gene (each parameter) of every individual (including all parents and children) to generate a new population. Fitness scores of these new offspring are then calculated and they are ranked based on the dominance score in MOGA and dominance score and crowding distance in Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II). The top N candidates are selected for the next iteration. This process repeats until the fitness goal is achieved or the maximum of iteration is reached.

The standard Mozilla Common Voice dataset (CVS) is used and Deepspeech and Kaldi-ASR are targeted. Word Error Rates (WER) and Connected Component (CC) are used as the metrics of the effectiveness of the samples, and a survey is conducted to determine how generated examples are perceived. The framework increases the WER by 980% for the non-targeted attacks and achieves an average of 1.0 (1.83) word difference compared to the target phrase. For both attack types (targeted and non-targeted) 0.98 and 0.97 similarity with the original audio calculated using CC is achieved. Compared with Taori'18 [130], the resulting command is closer to the target phrase while maintaining comparable acoustic similarity with the original sample.

Schönherr'19 [118]: This work proposes a targeted adversarial attack tricking human perception based on the psychoacoustic model of the human auditory system. The resulting modified audio is being transcribed as a target phrase by the state-of-the-art speech recognition system (Kaldi) while a human can only hear the original audio. The input is manipulated using perturbations in places and at margins that the human auditory system cannot notice these.

Targeting a trained white-box ASR Kaldi model, the adversarial command generation method first applies forced alignment between the original input and the target transcription to calculate the best possible temporal alignment between the original audio and the target transcription. Then, back-propagation is used to attractively calculate the perturbations required to force the ASR to transcribe the target output. Two critical points of this work rely on this back-propagation process: 1. Hearing threshold based on the original audio is calculated and applied in the back-propagation to limit the modifications, which results in changes hardly perceptible by the human auditory system; 2. The pre-processing step is integrated with the DNN into a joint network for the back-propagation. The authors deduct the derivative for each step inside the pre-processing such that the back-propagation calculation can pass through not only the DNN but all the way back to the raw audio input, directly resulting in the optimal adversarial audio signal. This was considered as a difficult problem by Alzantot'18 [11].

In the evaluation, WER is used to measure how well the target transcription is recognised; perceptibility of noise instead of SNR is used to measure the level of the percepitible perturbation. This metric only measures the portion of the noise which is in excess of the hearing threshold, so it is assumed to be more accurate than SNR as only the part beyond the hearing threshold is considered. For the evaluation process, a baseline simple attack without applying hearing threshold, an attack including hearing threshold, and an attack with both hearing threshold and forced alignment as the final version are gradually evaluated and comparisons are made to inspect how the WER and the perceptible noise are affected by these techniques. Results show that although adding a hearing threshold will increase WER compared to without threshold (resulting in the lowest WER, making the adversarial commands closest to the target), it can still generate reliable adversarial examples while minimising human perceptibility. Also, the result shows that it is easier to embed adversarial examples in music. Another finding is the effect of the length of the original audio sample and the length of the target transcription. These two factors boil down to the phone rate required to modify the original sample to be recognised as the target. The result shows 4 phones per second is a good choice. The relation between the number of iterations required to make the attack successful and the allowed amount of perturbation is discussed as well.

The evaluation results show that it is is possible to hide any target transcription in any audio file in up to 98% of cases. The modification can be performed in less than two minutes for a ten-second audio file on an Intel Core i7 processor. Compared to previous targeted adversarial commands, the perturbation noise of their work is significantly reduced as it considers the human auditory system properly. This is confirmed using a two-part audibility study consisting of a user study and a MUSHRA [116] test.

Yakura'19 [152]: This work proposes a method for generation of targeted adversarial commands against the state-of-the-art RNN-CTC ASR DeepSpeech. The described attack is the first over-the-air work against this type of ASR; previously described work aiming at this kind of ASR usually feeds adversarial commands directly to the ASR algorithms without taking speaker, room and microphone characteristics into account.

The method is based on a white-box assumption. The approach is built upon the conventional adversarial examples generation method as described in Section 2.4.3 plus incorporating transforms introduced due to playback, reverberation and recording distortions acting on the audio signals when the attack is launched over the air. The transforms include a bandpass filter (accounting for the audio frequency range possible on real audio hardware), convolution with impulse responses (accounting for the reverberation effect in various room conditions), and adding white noise (simulating the noise added to the audio samples during transmission). When the adversarial command is generated by considering these effects in a practical attack scenario during the process, they are robust even when played over the air.

The evaluation uses a four-second sample from Cello Suite No. 1 by Bach (as used in Carlini'18) and a cut from To The Sky by Owl City (the same sample used in CommanderSong'18). It is attempted to generate three target phrases; this limited sample size is used as combining the original sample and the target phrase takes 18 hours of processing. SNR, attack success rate and edit distance (the minimum number of steps required to transform one string to another by inserting, deleting and replacing characters) are used as evaluation metrics.

The results show that both two inputs can be used to generate adversarial commands which are transcribed as exactly the three input phrases, thus reaching a 100% success rate. For Bach, 9.3dB, 5.3dB and 0.2dB as the SNR between the power of the input sample and that of the perturbation are required to let the perturbed samples be transcribed as the three target phrases; while for Owl City input, 11.8dB, 13.4dB and 2.6dB are required. The results are compared with over-the-air results of CommanderSong'18, and the proposed method can generate samples with less perturbation while targeting a more complex ASR model. It is also confirmed that generating adversarial commands using a song is easier as it requires less perturbation, which supports CommanderSong'18's assumption that some phonemes from a singing voice assist in forming the target phrase. Their user study on the Amazon Turk also proves their attack can hardly be recognised. The robust adversarial commands are generated with the help of incorporating all three transformation techniques into the optimisation process. Different combinations of these transforms are also evaluated and it is found that all trails fail except when applying both, band pass filter and adding white noise. However, when using only two transforms compared to using three, a higher perturbation level is required.

Qin'19 [108]: This work improves on Carlini'18 [24] by using psychoacoustic principles to reduce imperceptibility of adversarial commands. A white-box adversarial example attack is considered. Furthermore, the work also considers the situation of playing the adversarial examples over the air by simulating the reverberation factor during generation of the adversarial commands.

The aim is to generate an adversarial command from an arbitrary input audio example (the two should have similar length), and to reduce the level of distortion noticeable by humans by making use of the masking threshold theory. The generation process is split in two steps. First, gradient descent is used to find a relatively small perturbation which causes the perturbed result being transcribed as the target phrase. This step is similar to Carlini'18. Then, the second step is applied to make the command imperceptible. The final perturb value in the first step is used as starting point. Then gradient descent is used again to calculate the new perturbation with a new loss function combining the network loss (cross entropy loss function) and imperceptibility loss (using the masking threshold knowledge). Furthermore, room simulators are used to process the perturbed input example and to calculate the not only imperceptible but also robust adversarial commands.

A user study is used to evaluate the imperceptibility of the adversarial commands. The test ASR is the Lingvo [121] classifier which is a state-of-the-art sequence-to-sequence model, and samples from the LibriSpeech dataset are used as test data. Imperceptible adversarial examples are developed which maintain 100% correctly transcribed output. The work is compared with Carlini'18 and it is shown that the resulting commands are better hidden. The attack is also somewhat more practical as realistic reverberation distortion is simulated. However, the attack is not trialled by playing the adversarial commands through a loudspeaker towards an ASR (an over-the-air attack).

Szurley'19 [129]: This paper proposes a method to generate white-box over-the-air targeted adversarial commands based on psychoacoustic properties.

To generate commands the global masking threshold per frame using a psychoacoustic model based on the MPEG-ISO standard is calculated. This reflects the relationship between the original audio signal and the human auditory system in the frequency domain. This work reformulates frequency-based perceptual loss, the distortion level of the perturbation added following the guidance of psychoacoustic model in the time domain, which results in less computation in each iteration of generating adversarial examples (calculation can be done solely in time domain) and less memory is required (There is no need to store real and complex components separately as in Schönherr'19 [118]). This improvement solves the instability issue during back-propagation encountered by Schönherr'19 and Qin'19 [118, 108]. The perceptual loss and adversarial loss are combined to calculate the Projected Gradient Descent (PGD). The attack signal is also improved for over the air attacks by incorporating various room impulse responses generated by a room simulator.

DeepSpeech is used as the ASR target, and WER, Character Error Rate (CER) and Perceptual Evaluation of Speech Quality (PESQ) are used as evaluation metrics. In comparison to WER, CER measures the distance between a generated transcription and the target on a character level instead of a word level. PESQ is an objective metric to automatically measure the speech quality of a system inducing distortion to a speech signal, ranging from 0.5 (poor) to 4.5 (excellent). The phrase "open the door" is used as target phrase for 100 randomly sampled audio files. 100% success rate is achieved for both including perceptual loss (PESQ = 4.0) and not including it (PESQ = 3.3). An interesting observation is that SNR decreases together with a PESQ increase when perceptual loss is included compared to the situation when only including the basic adversarial loss. When no perceptual loss (no psychoacoustic masking threshold) is induced, the attack can add perturbation equally spread over the whole frequency spectrum which reduces the overall amplitude for any one frequency component, resulting in lower distortion level thus higher SNR. However, when such technique is included, the perturbation power is shifted towards certain frequency components guided by the global masking threshold, causing a lowering of the SNR but an increase in PESQ as these distortions are within the masking threshold and are inaudible. The over-the-air evaluation is performed in an anechoic chamber. In the over-the-air experiment, the distance between speaker and microphone is increased gradually to observe the effect of multi-path and reduced signal power on the attack performance. A sound level meter to measure signal loudness in A-weighted decibels (dBA) is used; two decoders (a language model decoder and a greedy decoder) after the DeepSpeech ASR is used to measure WER and CER. Between 60-70 dBA, WER of 4.0 and CER of 2.4 are obtained for the language model, while 3.0 and 2.25 are achieved for the greedy decoder.

Imperio'19 [120]: This work proposes the first general algorithm to produce robust targeted adversarial commands against the Kaldi ASR assuming a white-box scenario. The adversarial Commands generated can be played over the air, and it is reasonably robust even when played in a case where the room characteristics are different than the one used in the generating process of the examples.

This work is built upon the previous implementation of Schönherr'19 [118]. It extends the implementation by adding another layer simulating the effect of sound being transmitted over the air. This layer consists of the convolution of the input audio signal and a random Room Impulse Response (RIR) from an RIR distribution. For the RIR implementation, AudioLabs [10] is used which calculates the RIR given room dimensions, audio decay time and positions of the source and receiver as the input. This maximises the probability of adversarial examples being transcribed as the target phrase when varying RIR representing different rooms and recording conditions are encountered. As in Schönherr'19 [118], gradient descent is used to update the audio signal based on the new implementation to generate the adversarial examples. The algorithm terminates either after a maximum number of iterations or when a sufficiently robust adversarial command is created.

A simulated over-the-air experiment is run in which the generated examples with added RIR of a real lab are fed to the ASR. Two RIR distributions are used to generate the commands in this experiment. This experiment shows that more noise has to be added to make the example robust when played over-the-air resulting in low Segmental Signal-to-Noise Ratio (SNRseg), and the examples generated with psychoacoustic hearing thresold have better SNRseg which means less distortion is added to the original when compared to not using a hearing threshold. The overall WER of these examples are low, but the authors claim that for the attacker only one example with 0% WER is sufficient. In a real over-the-air experiment in the lab a success full attack can still be launched (achieving 0% WER transcription) with 3.7% of the examples generated using a hearing threshold and 1.8% of examples without using it. The results also shows that the simulated attack gives a good enough prediction for outcome of a real attack.

Yang'19 [153]: This work proposes a defence method against adversarial commands. The countermeasures developed for the image adversarial example domain are evaluated to see if they are also effective in the audio domain. It is found that input transformation methods provide limited improvement against adaptive adversarial attacks in the audio domain. In contrast, inspired by spatial correlation in image processing, temporal dependency methods provide robust defence against audio adversarial examples and this holds true for even advanced adaptive attacks. This work first evaluates the robustness of input transformation and subsequently temporal dependency based methods against adversarial commands.

Considered state-of-the-art attacks are based on Alzontot'18 [11], CommanderSong'18 [156] and Carlini'18 [24]. The work classifies these attacks into two categories: audio classification attack (Alzontot'18) and speech-to-text targeted attacks (CommanderSong'18 and Carlini'18). The former aims to attack the neural network to mis-classify classes among "yes, no, up, down, etc.", and the latter aims to attack the model for transcription of the input to a targeted text. Four primitive but effective input transformation methods are applied: quantisation, local smoothing, down sampling and autoencoder. In theory, these input transformation methods act as a filter removing perturbations added to the original audio by the adversary. Thus, these defence methods still help to recover the original audio content given an adversarial input. The experiments show that in general, input transformation methods except autoencoder are effective in defending Alzontot'18, CommanderSong'18 and Carlini'18 attacks.

The work also explores how well temporal dependency based method behave in detecting audio adversarial attacks. The principle of the temporal dependency method is: given an audio sequence, the decoding result of the k-th portion of it as the input should be similar to the results of the prefix of length k of the transcribed result of the whole sequence as an input due to the temporal dependency of the audio sequence (i.e., correlations in consecutive waveform segments). However, perturbation is added to the original audio to transform the ASR output towards the target content, and so the temporal information is likely lost, resulting in missing similarity of consecutive segments. The evaluation shows that the temporal dependency method is able to discriminating attacks and benign inpup generated with methods described in Alzontot'18 [11], CommanderSong'18 [156] and Carlini'18 [24].

In an additional step, the work evaluates these two defence methods against adaptive attacks which are tailored by attackers who are aware of the defence mechanisms. Carlini'18 attacks are used as it is the strongest one among the three considered. Results show that input transformation methods fail in this scenario, however, these attacks still cannot overcome temporal dependency defence methods.

Metamorph'20 [28]: This work presents a system called Metamorph which generates robust over-the-air adversarial commands. Like most of the previous over-the-air attack studies such as Yakura'19 [152], Qin'19 [108], Szurley'19 [129] and Imperio'19 [120] this work also incorporates RIR convolution during the adversarial input generation process to let commands survive the realistic acoustic channel. However, this work uses empirical experiments to analyse how frequency selectivity, which is caused by device distortion, channel effects and background noise, impact on the attack success rate. The key finding are: background noise influence is minor, receiving device distortion exists but can be handled together with channel effects (if the speaker is controlled by the attackers and it is a high device with flat frequency response), channel effects are the most significant obstacle. When the distance between the speaker and the receiver is long (e.g., more than 8 m), the channel frequency selectivity effect is dominating and unpredictable. However, within a relatively short distance, multi-path effects are not very strong and it can be handled together with the microphone distortion to generate a robust over-the-air adversarial command.

First, a similar Expectation Over Transformation (EOT) approach to Imperio'19 [120] is applied to generate initial audio adversarial examples by incorporating multiple RIR measurements from an existing public RIR dataset. This work does not state explicitly which method (e.g., gradient descent) is used during the back propagation to obtain perturbations, but we assume it is done similarly to previous adversarial example studies. The transcript success rate of these generated examples dramatically drops when distance increases from 1m to 2m. Even though channel and device effects have been considered by considering

RIR, the adversarial examples are not generic enough and cannot adapt to a new over-the-air environments different to the ones (RIR picked from the dataset) used to generate them. To solve this issue, this work introduces a domain discriminator to extract the channel and device specific features. The adversarial examples generation process is thus optimised to also consider these features in a next step. After another robustness optimisation, the perturbation perception by humans is reduced by: 1. Shaping the perturbation to be more like real-world sound (e.g. bird chirp); 2. Reducing the perturbation coverage (if the perturbation magnitude on certain sampling point within a frame is small enough, the perturbation is ignored).

Evaluation is considering line-of-sight (LOS) and none-line-of-sight (NLOS) scenarios, and Character Successful Rate (CSR) and Transcript Successful Rate (TSR) are metrics used in various speaker and microphone distance situations. Mel Cepstral Distortion (MCD) and a user study are both used for measuring perturbation perception.

A high success rate of over 90% can be achieved in attack distance of up to 6m. While the distance is within 3m, perturbations are hidden really well, maintaining the high success rate.

Cheng'20: This is my work which is presented in Chapter 4 of this thesis. It proposes a practical defence method detecting adversarial command attack targeting ASRs. The assumption of this defence is that there is not enough transferrability for attack examples generated based on one ASR to be equally effective on another different ASR. Empirical experiments are carried out, showing the feasibility of this defence mechanism. Details are explained later in Chapter 4.

Comparison: A comparison among these studies on adversarial commands (or adversarial examples) is shown as Table 3.3. The first field is related to the threat and describes the length of the adversarial command being considered in the study. *Length* describes if the command considered contains only a few words or if it is a long sentence. *Practicality* captures relevant aspects to be considered when attempting to use the described method in a practical setting. *Test Method* describes how the adversarial command generation method was tested; i.e. was the command submitted over-the-air using a speaker and microphone or was a generated audio file directly fed into an ASR. *Room* gives a description of the environment in which such over-the-air evaluation was carried out. *Distance* shows the longest distance between the speaker and the microphone that was considered in case of an over-the-air evaluation scenario. *ASR* gives details on the used ASR, specifically *Type* and *Model*. *Type* shows the key components of the ASR framework used in the study, and *Model* shows the name of the specific ASR used. *Context* tells if the ASR framework and parameters are known when generating the adversarial commands (referred to as white- or black-box. *Technique* shows the key algorithms used in the study to generate the adversarial

Table 3.3 Comparison of ASR adversarial examples work (Category 1.2.3). *Length*: if the commands considered are only few words or they are sentences. *Practicality* consists of Test Methods, Room and Distance. *Test Method*: how the adversarial command generation method was tested; i.e. are the commands played over-the-air using a speaker and microphone or are the audio files directly fed into an ASR; *Room*: a description of the environment in which such over-the-air evaluation was carried out; *Distance*: the longest distance between the speaker and the microphone in case of an over-the-air evaluation scenario. *ASR* consists of Types and Models. *Type*: the key components of the ASR framework used in the study; *Model*: the name of the specific ASR used. *Context*: if the ASR framework and parameters are known when generating the adversarial commands (referred to as white- or black-box. *Generation Tech*: the key algorithms used in the study to generate the adversarial commands. *Metrics*: metrics used to measure the noise introduced to the original sound sample by the added perturbation. Studies tend to use objective metrics such as SNR as well as launching a user study to test human participants perception of these adversarial examples.

Papers	**Length**	**Practicality**			**ASR**		**Context**	**Generation Technology**	**Perception Metrics**
		Test Methods	Room	Distance	Types	Models			
Iter'17	word& three-word phrases	simulation	N/A	N/A	DNN	WaveNet ASR	white-box	FGSM FGM	observation
Alzantot'18	one word	simulation	N/A	N/A	CNN	Google CNN	black-box	genetic algorithm	user study
Carlini'18	sentences	simulation	N/A	N/A	RNN -CTC	DeepSpeech	white-box	Adam optimisation	decibel& observation
Commander -Song	sentences	over the air	N/A	1.5m	DNN -HMM	Kaldi & iFLYTEK	white-box	Gradient Descent (GD)	SNR & user study
Taori'18	two words	simulation	N/A	N/A	RNN -CTC	DeepSpeech	black-box	genetic& gradient	correlation coefficient
Khare'19	sentences	simulation	N/A	N/A	DNN -HMM& RNN -CTC	Kaldi& DeepSpeech	black-box	MOGA NSGA-II	Correlation Coefficient & user study
Schönherr'19	sentences	simulation	N/A	N/A	DNN-HMM	Kaldi	white-box	GD& psycho-acoustics	user study& MUSHRA
Yakura'19	two or three words	over the air	one room	0.5m	RNN -CTC	DeepSpeech	white-box	Adam optimisation	SNR & user study
Qin'19	sentences	room simulation	N/A	N/A	RNN & attention	Lingvo	white-box	GD& psycho-acoustics	user study
Szurley'19	one sentence	over the air	one anechoic room	0.16m no echo	RNN -CTC	DeepSpeech	white-box	PGD& psycho-acoustics	PESQ
Imperio'19	sentences	over the air	various rooms	4.3m	DNN-HMM	Kaldi	white-box	GD& psycho-acoustics	SNRseg
Metamorph'20	sentences	over the air	one room various locations	6m	RNN -CTC	DeepSpeech	white-box	Adam optimisation	MCD user study

commands. *Metrics* details the metric used to measure the noise introduced to the original sound sample by the added perturbation. Studies tend to use objective metrics such as SNR as well as launching a user study to test human participants perception of these adversarial examples.

Note that we could not include the success rates and user study results of evaluations as these works are evaluated using different datasets, different target commands and even different evaluation metrics of interest. Given the amount of work targeting this narrow field of PVA security and privacy it might be time to introduce a common evaluation framework, similar to the ASVspoof challenges used to profile work in the VA domain.

3.2.1-C Summary

Users access a PVA via the acoustic channel and access control is essential. Access control can achieved by using VA methods and to some degree by checking the semantics of a voice command. Most PVAs only check simple semantics (e.g., does the command contain 'Alexa' in case of the Amazon Echo) and some use few PVA use VA (e.g. Siri). However, even in the few cases where VA is used, potential attacks such as replay or spoofing on this mechanism are usually ignored. It is also assumed that users would realise if someone is interacting unauthorised with their PVA. In normal circumstances a user can hear voice commands spoken by an adversary or played by a speaker used by the adversary.

There is a large body of existing work looking at VA, how to attack VA and methods to detect and prevent such attacks. We have outlined these works in this chapter as we believe it is necessary to consider these relevant works now in the context of PVA.

The main threat to VA is spoofing. Work on spoofing attacks and spoofing countermeasures are categorised in this chapter under *C1.1.1 Acoustic Characteristics* and *C1.1.2 2nd Factor Authentication*. In the first group, we discuss findings of the ASVspoof challenge which already summarises the state-of-the-art progress in this field. We also include VMask'20 [159] as it is the first practical black-box attack targeting an ASV system and not a general ASR environment. In the second category work is described which uses a second data source to detect spoofing. We briefly mention work using sensors other than acoustic ones, and we introduce liveliness detection works based on human vocal system features to differentiate signals played by a speaker and voiced by a human. These studies focus on finding traces unique to human speakers in the acoustic signal. Other work in this category (Blue'18) aims to locate traces unique to COTS speakers to infer if the voice command source is a speaker system or a human.

It is clear that more studies are required in terms of VA security in the context of PVAs. How to apply spoofing attacks and defense techniques in a physical scenario should be explored further. The unique capabilities (in terms of available hardware and also speech processing software) of a PVA may enable novel attack methods and also new defence strategies. We believe that work using a 2nd Factor for authentication should be explored further, in particular when this information can also be derived from the acoustic channel as this makes such solution very practical. Finally, research also should look into methods for bypassing such defence methods; work in this space is missing.

The area of *C1.2 Hidden Voice Commands* is a specific domain attracting a lot of work in the research community. We described the 3 sub areas *C1.2.1 - Hardware Non-Linearity*, *C.1.2.2 - Obfuscated Commands* and *C.1.2.3 - Adversarial Commands*.

There are four works described in the *C1.2.1 - Hardware Non-Linearity* category. Three of these are describe attacks, and only one has a focus on defense. Relatively long-distance attacks have been achieved targeting PVAs with about 25ft (7.62m) distance. However, all attacks require high-end or highly-customised acoustic signal emitters. The defense work also requires specifically designed speaker arrays. Overall, work considering non-linear properties is an emerging area, more work for both attack and defense is necessary. We would like to see if the requirement for additional equipment can be overcome. PVA vendors should take note of this work and design hardware with the aim to exclude such attacks.

We described three studies on *C.1.2.2 - Obfuscated Commands* in detail. Two of these were relatively early work (before 2017), and the latest one was published in 2019. The key feature of obfuscated commands is generation of noise-like audio samples to attack an ASR. There is not much work in this area, the reason might be that hiding the malicious voice commands within noise is not 'so covert' and still can raise user's suspicions. These works lack convincing evaluation to show that an attack would not raise suspicion in a real-life attack scenario. The user studies on perception of the audio samples can only show that it is hard to understand the commands but it is not analysed in depth if this would be perceived as being suspicious. A more detailed investigation of feasibility of these attack types would be necessary.

C.1.2.3 - Adversarial Commands has attracted the largest interest from the research community. State-of-the-art ASR systems heavily use deep learning techniques and this is a popular research topic at the moment. Adversarial examples for ASR systems is a relatively new topic compared to similar work targeting computer vision systems. Considering the rapidly increasing population of PVAs more and more security and deep learning research groups focus on this promising area. From Table 3.3, it is clear that the trend is towards generation of adversarial complex sentences, departing from early work focusing only on words. Also, work is moving on from simple simulation to evaluation of full systems (over-the-air attack on of the shelf PVA). The subject ASRs used are mainly classic DNN-HMM (Kaldi) and end-to-end solutions using RNN-CTC (DeepSpeech). Most of the work is still considering white-box settings where the ASR internals are known to the attacker. Perturbations are generated mainly based on GD optimisation. Psychoacoustic masking is increasingly used to optimise addition of perturbations. However, the details on how perturbations are added best is subject of current work. Most works make use of user studies to evaluate how well adversarial examples are constructed.

More practical and robust adversarial command studies are required, considering evaluation in different reverberation (various room or one room but various settings) scenarios. So far, only Imperio'19 and Metamorph'20 are two practical works considering this. Current work lacks practical black-box assumptions; it is always assumed that the internals of an ASR are fully known. Adversarial commands are not evaluated against a range of COTS PVAs. It is highly possible, and [159] is an inspiration as it attacks devices assuming that different ASV systems share the same core concept, that a variety of systems can be attacked using a common adversarial command. CommanderSong'18 investigated this transferability of their attack using the iFLYTEK ASR considering a black-box attack. However, more work in this direction is required. There is also a lack of adversarial command research targeting the latest attention/transformer-based end-to-end ASR introduced in Section 2.2 with Qin'19 [108] being a notable exception. Considering new ASR systems may enable new attack and defence methods. Currently is a lack of metrics for measuring the perceptual distortion of adversarial commands. Research in this area could help building more effective adversarial example studies. We notice there is a recent work [133] developing this area. It also has to be noted that defence mechanisms preventing spoofing of VA and defence mechanisms preventing command injection may share algorithms and mechanisms. It would therefore be useful to consider both lines of work together when designing countermeasures.

3.2.2 C2 - Privacy

In category *C2 - Privacy* we summarise existing research work concerned with privacy. The use of the PVA's acoustic channel can lead to a loss of privacy as voice recordings may reveal sensitive user information. Work includes novel methods for privacy preservation and new ways of consent management.

3.2.2-A Privacy Preservation

To improve control over the information exposed to the PVA different approaches have been proposed. The commonality of these approaches is that the speech signal is subjected to some transformations before it reaches the PVA. Using these transformations the elements of the audio signal are removed which are considered to be sensitive.

Smart^2 Speaker Blocker'19 [25]: In this work physical infrastructure is proposed to improve control over the signals reaching the PVA. The PVA is placed in a soundproof

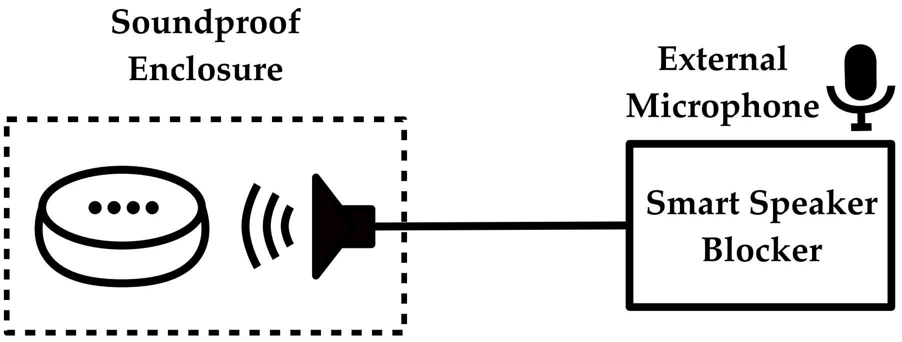


Fig. 3.6 Smart[2] Speaker Blocker is used as a filter to capture and pre-process the sound. Privacy can be preserved as only sanitised results can be played towards the genuine PVA

enclosure together with a speaker. The speaker is connected to a protection device called the Smart Speaker Blocker. The Smart Speaker Blocker is essentially a second PVA which is controlled by the user and is used to filter voice commands before they reach the main PVA (See Figure 3.6).

The Smart Speaker Blocker uses it’s own ASR system based on Sphinx4. Depending on filter rules set on the Smart Speaker Blocker actions are taken. If a voice command is accepted to be passed to the main PVA a TTS module is used to transfer the command to the PVA in the enclosure.

As the original voice is not passed to the PVA in the enclosure it is ensured that speaker identification cannot be carried out. Also, other cues such as emotions or health information cannot be extracted from the speech signal. Furthermore, it can be tightly controlled which speech signals reach the PVA; accidental overhearing of conversations can be prevented.

A shortcoming of this work is the practicality of the deployment as physical modifications are necessary. Another issue is that now a second PVA is introduced which may raise privacy concerns as well. Finally, additional latency is introduced as additional signal processing stages must be traversed. However, the provided user study indicates that the additional latency is acceptable.

Hidebehind’18 [107]: PVAs use ASR in the cloud. The cloud implementation of ASR brings privacy risks. Individual users can be identified (see Section 3.2.1-A) and voiceprints can be extracted from voice data of users to launch spoofing attacks.

This work introduces VoiceMask (the work is titled Hidebehind while the solution is termed VoiceMask) as an intermediary between the voice recording PVAs and the cloud or an untrusted app used for ASR to anonymise the voice before it reaches the untrusted system as shown in Figure 3.7. In this work two warping functions together with a differential privacy algorithm are used to construct a robust conversion algorithm which reduces the success

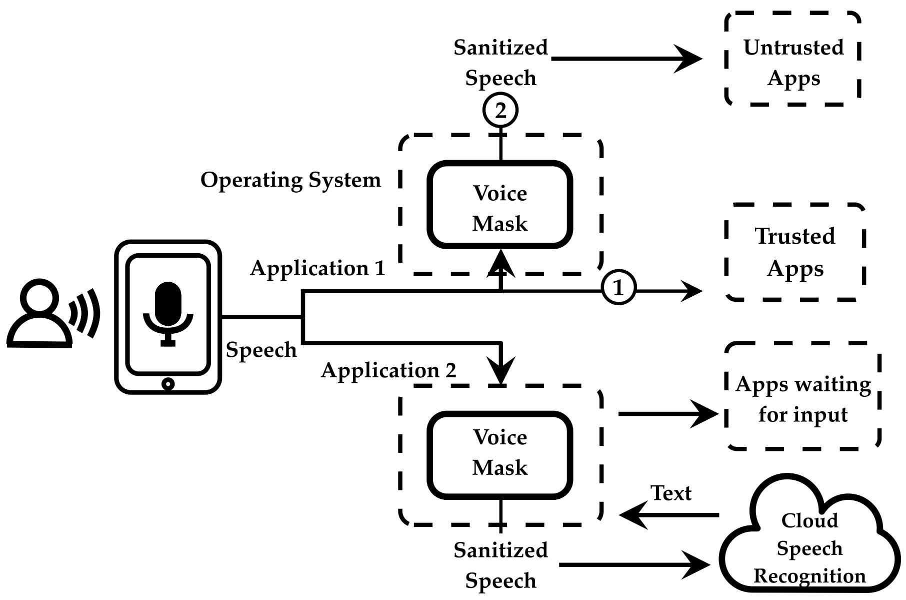


Fig. 3.7 Illustration of two application scenarios of VoiceMask in the Hidebehind'18 paper. First scenario is shown as the upper part. VoiceMask can be embedded into the Operating System (OS) and only sanitises the speech sent to untrusted apps which do not need to access to the orginal voice as shown in Route 2, and trusted ones which must obtain the original voice are granted the Route 1; second scenario is in the lower part. VoiceMask is the gateway between user speech and apps and cloud. VoiceMask masks original speech and send them to online ASR interface. After obtaining the feedback transcript, the speech masks are revoked. Transcripts are relayed to apps

rate of speaker recognition. A random number pair for the warping function is selected. A privacy protection algorithm is used to prevent reversing this process. The method provides resilience against de-anonymisation attacks while ensuring that the resulting signal is still recognisable by ASR.

The provided evaluation shows that the likelihood of identifying a person's voice from 50 candidates is reduced by a mean of 84%. At the same time the speech recognition accuracy is reduced by not more than 14.2%.

Gong'18 [53]: Paralinguistics studies vocal (and sometimes non-vocal) signals beyond the basic verbal message or speech. Paralinguistic information contained in human speech can be extracted for applications such as ASV, speech emotion detection and medical diagnostics. The work of Gong et al. introduces an attack on a speech paralinguistics detection algorithm. The aim is to distort a speech signal such that paralinguistic feature detection fails (produces classification errors) while the signal distortion is difficult to recognise by a human.

The work proposes an end-to-end scheme to craft adversarial audio signals starting with the original audio signal rather than with already extracted acoustic features as shown

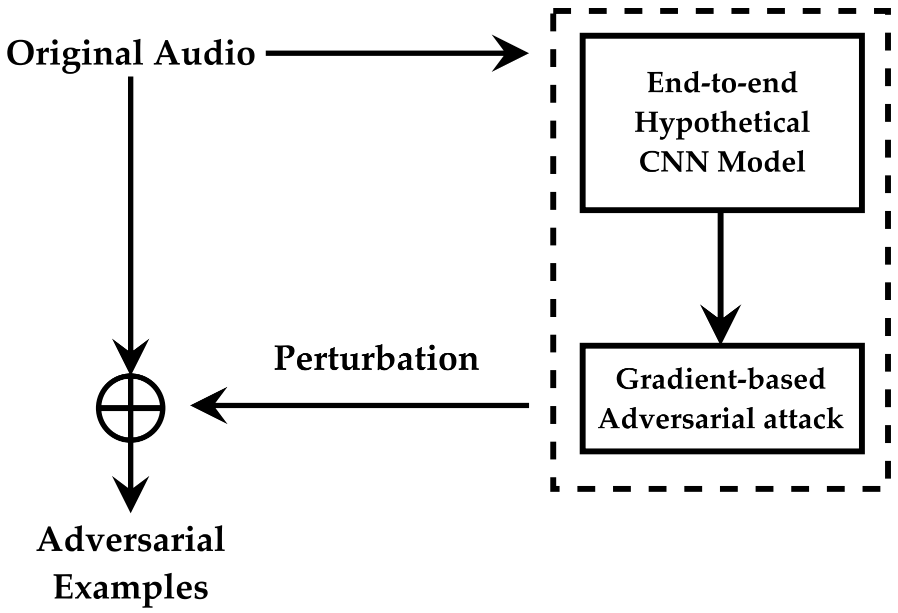

Original Audio
End-to-end Hypothetical CNN Model
Perturbation
Gradient-based Adversarial attack
Adversarial Examples

Fig. 3.8 Adversarial examples of speech paralinguistics applications are generated directly from the orginal audio via applying FGSM on a hypothetical end-to-end neural network

in Figure 3.8 (see also Section 2.4.3 on creation of adversarial commands where starting point are acoustic features). These adversarial signals result in performance drop of state-of-the-art NNs (specifically, Recurrent Neural Network (RNN) and CNN) used for speech paralinguistics analysis. By adding perturbations directly to the original audio signal instead of to acoustic features such as MFCC parameters, human noticeable signal distortion is reduced.

Initially, the state-of-the-art speech paralinguisitic model WaveRNN and gradient descent is used to construct adversarial signals. A gradient-based attack is applied to calculate the perturbations on the waveform input. However, it is shown that this approach suffers from the vanishing gradient problem. The gradient is the derivative of the loss function (distance between desired attack output and the original output) with respect to the input waveform. The gradient is required to calculate the perturbation in order to minimise the loss function. With RNN, the gradient for early time steps of a long input sequence becomes zero (vanishing). Thus, the gradient is only valid for the last few steps and perturbation is added correctly only to the last parts of the input sequence. Therefore, WaveRNN cannot be used as the model to construct the adversarial input. WaveCNN is used in this work as the vanishing gradient problem does not exist in a CNN structure. However, the attack signal generated by CNN is generalised enough to also affect RNN-based speech paralinguistic analysis.

The evaluation results show that with a perturbation factor of 0.02, the proposed method can already efficiently increase the classification error rate for WaveRNN and WaveCNN. A user study shows that the proposed method generates "natural" adversarial samples that can prevent paralinguistic analysis.

Although the authors did not describe the method as a privacy preserving mechanism we decided to place it in this category within this chapter. Similar to the previously described work Hidebehind'18 this method could be used to hide voice features from a PVA.

Abdullah'19-2 [6]: This work differs from previous outlined work in Section 3.2.1-B2. Instead of modifying an audio signal such that a machine is able to understand the command but a human is not, the signal is modified such that an ASR is unable to understand the command but a human is. Such methods can be used to preserve user privacy.

The aim of this work is an attack aiming at downgrading the performance of ASR and Automatic Voice Identification (AVI) in telephony networks. The audio signal is decomposed into components and components for which the signal strength is below human perception thresholds are discarded. The reason behind this approach is that low intensity components should not affect the audio quality perception of humans. However, the performance of the ASR and AVI is affected given the assumption that ASR and AVI depend on components of speech which are non-critical for human perception. An algorithm is developed to obtain the optimal threshold in a way that only a minimum of components are discarded while the audio is still misinterpreted by ASR and AVI.

This is a black-box attack only focusing on using signal processing to transform the representation of the original audio samples. Discrete Fourier Transform (DFT) and Singular Spectrum Analysis (SSA) are the two transformation techniques used. SSA decomposes an arbitrary time series into components called eigenvectors. These eigenvectors represent structural information of the signal.

The work uses TIMIT [50] and their own word-level dataset, four attack scenarios, two test environments, seven target models, and a user study to evaluate the efficacy of the attack. Specifically, for a word-level attack, SSA can ensures that 50% of the words are mis-transcribed. A DFT-based attack on phoneme perturbations shows that vowel perturbation leads to the best ASR mis-transcription rate. Attacks on AVI show a higher success rate than an attack on ASR when phonemes are targeted. For the transferability attack, adversarial examples generated based on Google are effective on other 6 other models at least 42% of the time. To test the robustness of the attack against the defence proposed by [153], the Area Under the Curve (AUC) score of the attack measured by this defence shows 0.527 which is far lower than the value of 0.936 for the attacks tested in the original work.

Table 3.4 Comparison among privacy preservation work. Target: the target point in the PVA ecosystem. Practicality: how practical the implementation of the privacy preserving methods are. Purpose: the purpose of the protection method. Technology: the main techniques used for the approach

Work	Target	Practicality	Purposes	Technology
Smart^2 Speaker Blocker'19	smart speakers	extra intermediary device	filtering speech content & paralinguistic information	speech recognition & TTS
Hidebehind'18	smart devices	no extra device	concealing speaker voiceprint	voice conversion & differential privacy
Gong'18	smart devices & cloud	no extra device	downgrading paralinguistic analysis performance	generating adversarial examples with FGSM

3.2.2-B Consent Management

People usually have little or no control over PVAs in their vicinity. A user can prevent his own device from recording but would not be able to stop other devices from recording conversations.

A user can prevent a PVA from recording voice completely by simply disabling the device. Such methods have been proposed and we discuss these in Section 3.2.3-B as they are effectively DoS attacks; DoS is employed as method of recording control.

For more fine-grained control other methods than DoS are required. Generally, methods to signal recording consent of individuals requires cooperation of the back end infrastructure. Speaker identification for access control purposes might be usable for implementation of a recording consent management system but such options have not yet been explored in literature.

Cheng'19 [29]: This is my work which is presented in Chapter 5 of this thesis. This work explores acoustic tags as means of recording consent management. Acoustic tags can be embedded with audio signals to indicate a cooperating back-end infrastructure that a user does not give the consent of recording her voice. An experiment with an off-the-shelf Google Home Mini and a Raspberry Pi as the reactive tagging device is carried out, showing the feasibility of applying acoustic tagging in modern PVA eco-system. Details are explained later in Chapter 5.

3.2.2-C Summary

Table 3.4 provides a comparison of the three closely related works in category C2.1. Aspects listed are: the target point in the PVA ecosystem (Target), how practical the implementation of the privacy preserving methods are (Practicality), the purpose of the protection method (Purpose) and the main techniques used for the approach (Technology).

Overall, there are few studies related to privacy in the PVA context. In the privacy preservation part, Smart^2 Speaker Blocker'19 intrusively introduce an additional PVA to sanitise the original voice commands, which protect privacy in regards of conversation content and paralinguistic information (user identity, emotion, etc.). The practicality is limited due to the hardware customisation. Hidebehind'18 introduces a software to conceal the voiceprints, which helps to protect privacy in terms of the user identity. Gong'18 proposes an adversarial examples targeting downgrading the performance of paralinguistic analysis, which can be used to protect user privacy such as user identity, emotion, and medical diagnostics. Abdullah'19-2 shows how to modify a speech signal such that an ASR is unable to understand the command but a human is.

There is only one work in the consent management category C2.2. Cheng'19 is a unique study investigating technical means of implementing consent management. Further exploration of this work area is necessary.

Both privacy preservation and consent management in the context of PVA are new and important topics arising with the fast development of PVA market. Although there is not much research work on privacy, it is the most concerning area for the general public. Many recent media reports point out privacy concerns of users. More research in this category is critical and necessary.

3.2.3 C3 - PVA Denial of Service (DoS)

The acoustic channel can be subject to a DoS attack. This form of attack on a PVA has attracted little research so far. However, given that we are increasingly depending on PVAs in our daily interaction with (also safety critical) computer systems, this needs to be considered.

Two categories of DoS attacks have been considered: *C3.1 - Skill Market* and *C3.2 - Jamming*. Attacks in C3.1 aim to manipulate the back-end processing (often referred to as the *Skill Market*). Jamming attacks (C3.2) target the audio channel directly, interrupting wake word recognition or ASR in general.

3.2.3-A Skill Market

PVA service and product providers like Amazon and Google provide skills (or actions resulting from the naming convention from Google) which are similar as apps to smartphones to enrich the capability of PVAs. Users (and service providers) can deploy code to the processing back end to enrich the PVA functionality. This is a potential entry point for an attacker to deploy malicious code into the PVA processing chain and possible vulnerabilities have been studied by the research community.

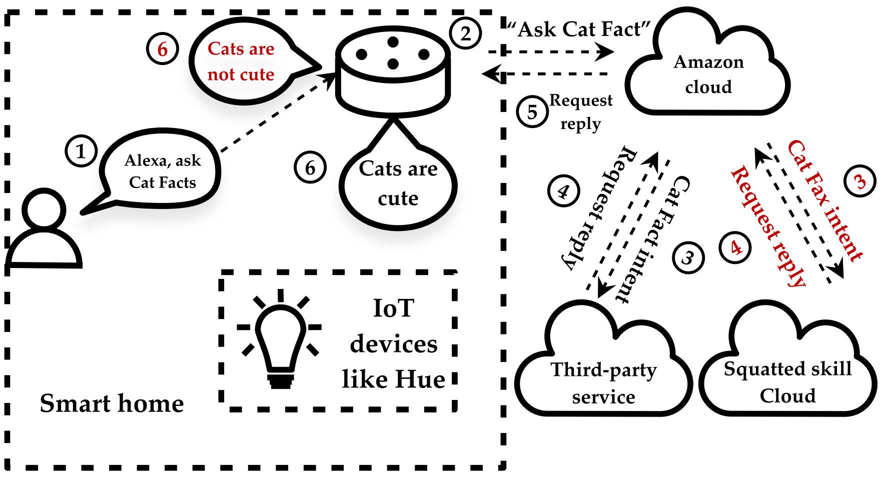


Fig. 3.9 Illustration of how a malicious skill created by adversary launches skill squatting attack. Steps labeled with black numbers are the normal procedures, while steps labeled with red especially Step 3, Step 4 and Step 6 would replace the original ones when the attack happens

Kumar'18 [76]: This work presents an empirical study of the misinterpretation error of the Amazon speech recognition service Alexa. Skills are invoked by the back end infrastructure depending on the transcribed text. Every ASR is subject to interpretation errors and these errors can be exploited to design a skill which is activated by accident when spoken user commands are interpreted wrongly. Some phrases are likely to be misinterpreted consistently in the same manor which can then be exploited to craft a skill which is activated on misinterpretation. This form of attack is called *skill squatting*. Figure 3.9 shows an example of a malicious skill taking advantage of the misinterpretation to squat attack the legitimate "Cat Fact" skill.

The work analyses the systematic interpretation error of Alexa. An evaluation skill is used to simply return the transcribed texts by Alexa back to the end user. 11460 spoken utterances are used as audio input to this evaluation skill to discover the potential consistent and systematic interpretation errors by Alexa. After creation of an interpretation error library a further analysis is carried out to see which phonemes within the words are the exact cause of misinterpretation. The result of the misinterpretation analysis is then used to identify already existing skills that are subject to confusion. The study finds 381 unique instances in which skills invoked by a user might accidentally trigger an already existing different skill. Finally, the work explores the feasibility of squatting attacks towards certain groups of people and label this technique *spear skill squatting*. It is shown that this attack could be tailored towards groups using demographic information such as region or gender.

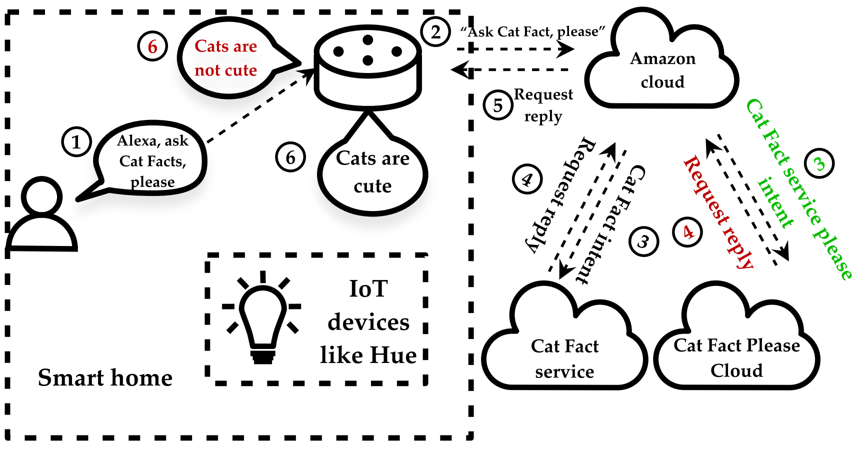


Fig. 3.10 Squatting attack in Dangerous Skills'19 paper: how the longest string matching can hijack the legitimate skill request

Dangerous Skills'19 [162]: This work looks at skill squatting attacks similar to the work of Kumar'18. However, in addition to these attacks based on misinterpretation errors, referred to as voice squatting in this work, additional word squatting and voice masquerading attacks are investigated.

Systems today apply a longest string matching strategy to identify which skills are being called by the users. Thus, word squatting is possible by registering an attack skill with the name as the one of a legitimate skill together with an additional utterance such as "please". For example, it is possible to register an attack skill invoked by the sentence "Cat Fact Service Please" in parallel to a legitimate service invoked by the phrase "Cat Fact Service" (See Figure 3.10). Due to the longest match, a user may accidentally invoke the attack skill instead of the legitimate skill simply by adding the word "please" to the command which is quite natural.

Voice masquerading attacks are exemplified in two scenarios. Both, Google Home and Amazon Echo allow only one active skill and it needs to be terminated before another one can execute. However, users may naively believe that a PVA supports skill switch. Users would ask to activate another skill while interacting with the current one. This opens a gate for a malicious skills to impersonate the desired one and to obtain sensitive information supposed to only be shared with the target skill. Users also depend on responses from a skill to tell them when it has terminated. A malicious skill could fake the termination by playing the audio response but keeps running. Even if the user uses commands like “stop”, or “cancel” to terminate a skill, a malicious skill can ignore these. If a user doesn’t interact with the PVA for a period of time after one round of enquiry and response, a PVA would re-prompt the user with an audio signal before terminating it. However, a malicious skill could create an inaudible audio re-prompt and try to stay active with the aim of stealing information from the user.

To evaluate the feasibility of voice squatting attacks. The author collects legitimate skill names mostly misinterpreted by a PVA. TTS and humans speaking these skill names are collected. Their corresponding misinterpreted names transcribed by the PVA are used to create shadow skills in so called test mode (These skills are not published). The authors also created skills whose name is the target skill name with an additional word such as “please” as a suffix. They then play TTS generated voice commands to test if these fake longer-name skills would be triggered. Experiment results show that these two attacks are valid. To evaluate both squatting attack and masquerading attack in real-world scenarios, the author registered several skills with similar name as a well known skill “Sleep and Relaxation Sounds”. The skill usage data revealed the efficacy of squatting attacks. Furthermore, out of the 9582 requests they collected, 52 were for skill switching and 485 were for terminating the current skill, which can both be exploited to achieve a voice masquerading attack.

As defence against the two types of attacks, first, they develop a skill scanner using utterance paraphrasing and pronunciation comparison to discover competitive invoke names in the current skill market, finding that 3655 out of 19670 invoke names of skills are highly similar, which again shows voice squatting is realistic. This scanner can be used for skill inspecting before new skills are to be published. The authors also develop a context-sensitive detector to detect the intent of switching skills and faking skill termination. The detector achieves 95.6% precision of detecting the intent in the real-world scenario test.

3.2.3-B Jamming

The acoustic channel can be subjected to noise which prevents ASR to function correctly. Jamming signals can either be applied continuously or be targeted more selectively to specific parts of an audio signal. For example, it is possible to target jamming towards wake word recognition to prevent a PVA from processing speech. Jamming is often applied for the purpose of privacy management and the work reviewed here could also be classed in Category

C2. However, as these works main focus is on the jamming component and not on privacy management we have decided to outline this work here.

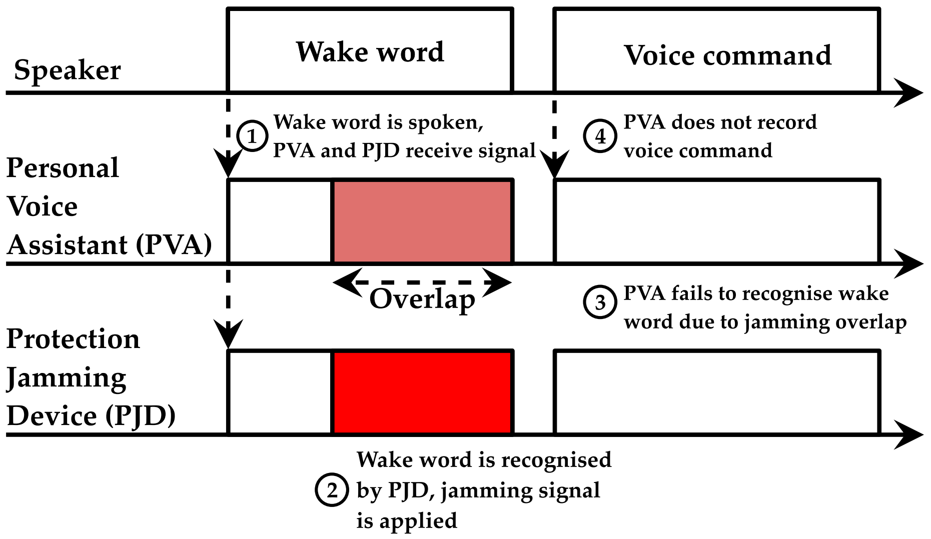

Speaker
Wake word
Voice command
1 Wake word is spoken, PVA and PJD receive signal
4 PVA does not record voice command
Personal Voice Assistant (PVA)
Overlap
3 PVA fails to recognise wake word due to jamming overlap
Protection Jamming Device (PJD)
2 Wake word is recognised by PJD, jamming signal is applied

Fig. 3.11 PJD recognises the keyword faster and emits jamming signal (Step 2) to overlap the rest part of the spoken keyword, thus the PVA would not be activated (Step 3) and the following voice commands would not be recorded and processed (Step 4)

Cheng'18 [31]: This is my work which is presented in Chapter 6 of this thesis. This work proposes a reactive denial-of-service jamming method to prevent people's voice contents being recorded and uploaded to the back end server by PVAs. This solution gives people the capability to stop nearby PVAs owned by others being activated and recording their conversation. A PJD device is proposed which listens to the same wake word as the PVA (e.g. "Hey Google" for Google Home, "Alexa" for Amazon Echo), recognises it faster than the PVA and emits a jamming signal to interfere with the wake word acquisition process on the PVA (See Figure 3.11). The method is intended to be used as a privacy management but it is possible to simply use it for DoS purposes. Details are explained later in Chapter 6.

Gao'18 [49] This work proposes a framework as shown in Fig 3.12 to address the potential privacy violation due to conversations recorded by PVAs and it also prevents unauthorised voice commands. Although there is no actual implementation and evaluation in this work, it proposes a well rounded solution considering practicality. The work suggests using an external obfuscator which continuously carries out jamming on the acoustic channel of a PVA to prevent it recording normal conversations. To make this constant jamming user friendly, inaudible jamming exploiting the non-linear property of microphones is used (See also Section 2.4.1 and Section 3.2.1-B1 where this technique is used to construct inaudible

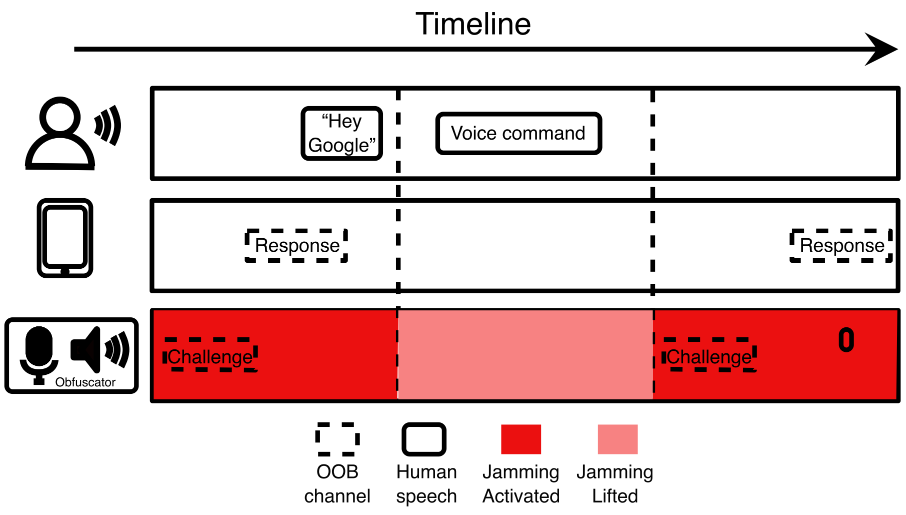


Fig. 3.12 The obfuscator continuously jams the acoustic channel until the keyword is heard by the special microphone of the obfuscator and authentication is fulfilled. Then the normal voice command is relayed to the a PVA

commands for command injection). The constant jamming should be lifted selectively to enable legitimate usage of a PVA. For this purpose an ultrasound microphone installed on the obfuscator is used to listen to wake words (e.g., "Alexa"). As this microphone still shows a linear property within very high frequency range, the ultrasound jamming signal sent by the speaker of the obfuscator won't create a shadow in the normal frequency range of the voice range of the microphone. In this case, the microphone can capture the wake word spoken by a user. This obfuscator then performs wake word recognition. If the wake word is detected and it passes the authentication which we explain in the next paragraph, it will stop jamming and communicates this keyword to the PVA via ultrasound injection. Subsequent voice commands will be relayed to the PVA.

To prevent a spoofing attack where an adversary may launch a replay attack towards the obfuscator, the obfuscator executes a challenge-response authentication with the legitimate user's smartphone using a secure out-of-band (OOB) channel such as Bluetooth. Once the keyword together with the authentication response is received by the obfuscator, it will lift the jamming and relay the keyword to the PVA for use in the normal case. This paper also discusses some pending challenges including the working efficiency of their framework facing beamforming technology, how to do fine-grained jamming in a multiple PVA environment and an unlikely attack scenario when the adversary is in the vicinity of the legitimate user, the obfuscator and the PVA.

3.2.3-C Summary

Kumar'18 and Dangerous Skills'19 focus on security vulnerabilities in the PVA back-end skill working mechanism. Cheng'18 and Gao'18 focus on front-end PVA devices preventing user privacy violation and unauthorised commands via DoS. Kumar'18 studies the interpretation error of the back-end ASR system systematically and discovers the error pattern. Dangerous Skills'19 discovers the same interpretation error, also a skill search issue, and malicious skill impersonation attack. Kumar'18 studies interpretation attacks more deeply which could be used as an instruction to improve the back-end system, while Dangerous skill'19 covers wider potential skill-related problems revealing back-end skill management needs more attention and study. Cheng'18 proposes protecting user privacy using DoS attack targeting wake word of a PVA. It carried out some early trial to test the feasibility. More comprehensive and systematic implementation of this idea in COTS devices scenario would be of interest. Gao'18 proposes a framework achieving a similar jamming idea as Cheng'18, and they also propose using a user's smartphone together with their obfuscator to achieve legitimate user authentication. This work is only a theoretical framework, it would be interesting to see research covering an implementation. Also, this framework is technically complex and it is not sure how practically feasible it is.

Overall, the potential of applying DoS techniques on the back-end PVA system and the front-end PVA devices has not been fully explored. Especially DoS attacks on the front-end have been used to implement some form of privacy control; a proper analysis of DoS capabilities and potential defence methods is missing. More studies of utilising DoS for both attack or defense purposes would be valuable.

3.2.4 C4 - Acoustic Sensing

In this category we describe work that uses an acoustic channel to perform sensing. We distinguish *C4.1 - Passive Sensing* and *C4.2 - Active Sensing*. For active sensing the PVA can be used to emit a sound signal and reflections are analysed for sensing purposes. Passive sensing relies on analysis of sound not specifically emitted for the sensing task itself.

Work in this category uses sound for sensing but in most cases not in the context of a PVA. In many cases a mobile phone which may include PVA components is considered, however, dedicated smart speaker devices have not yet been investigated. A number of works here aim to infer user interaction with a phone/tablet using sound (e.g. revealing user input). Nevertheless, the approaches discussed here can be applied to the PVA domain and they provide valuable insight into security and privacy of PVAs.

3.2.4-A Passive Sensing

Speech analysis in general could also be classified as a form of acoustic sensing. However, in this category here we include work that looks at information included in a sound signal other than voice.

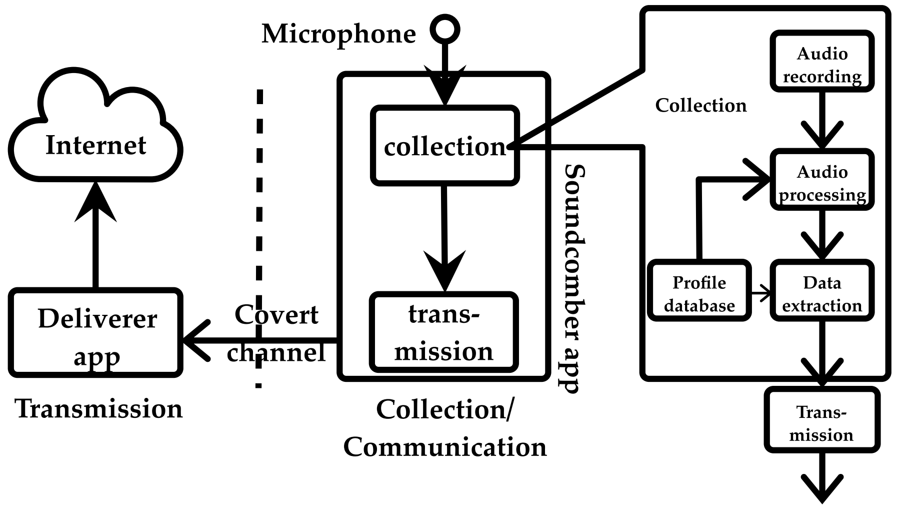


Fig. 3.13 Illustration of the working flow of Soundcomber'11: Main stages of data collection step and the process of delivering the collected sensitive data to the attacker

Soundcomber'11 [117]: This work proposes a covert smartphone attack which steals high-valued information such as credit card numbers or PIN numbers by analysis of the acoustic channel. The work assumes that malware components are installed on the phone, consisting of the Soundcomber and a separate element to deliver extracted information to the attacker (see Figure 3.13).

Soundcomber has access to the microphone during calls and has tone and speech recognition components. A built-in profile database is used to determine which segments of a call are likely to contain useful information for an attacker. Soundcomber records phone calls and analyses the beginning segment of the recording using speech recognition to decide if it is an interesting call based on the profile database (e.g. a call to a credit card customer service point). Each database entry contains a model (a finite-state machine) matching the service menu choices. When keywords are recognised at the beginning of the call, keyboard tone inputs and call segments using ASR are analysed to identify high-value elements such as provided credit card numbers. Then the Soundcomber either transmits this information through an existing application with Internet access or through a covert channels to another malware which then has Internet access. Covert channels explored in the paper include vibration settings, volume settings, screen and file locks. The work also proposes a defence against this attack, proposing a reference monitor which is implemented between the OS and the hardware. The monitor uses call numbers to decide if invocation of a second defensive stage, the controller is necessary. Once invoked, the controller activates exclusive mode which prevents all audio data from being delivered to applications during a call.

The experiments show that Soundcomber can detect more than half of the calls to critical hotlines, and it can recover 85% of all credit card numbers entered by users via tone recognition (one-digit error for the remaining 15%), while recover 55% of credit card numbers spoken by participants via speech recognition. The performance of the proposed defence mechanism is analysed and it is found to be a feasible solution for smartphones.

Additional acoustic information, the sound of PIN number inputs, is sensed to reveal critical information and to launch an attack.

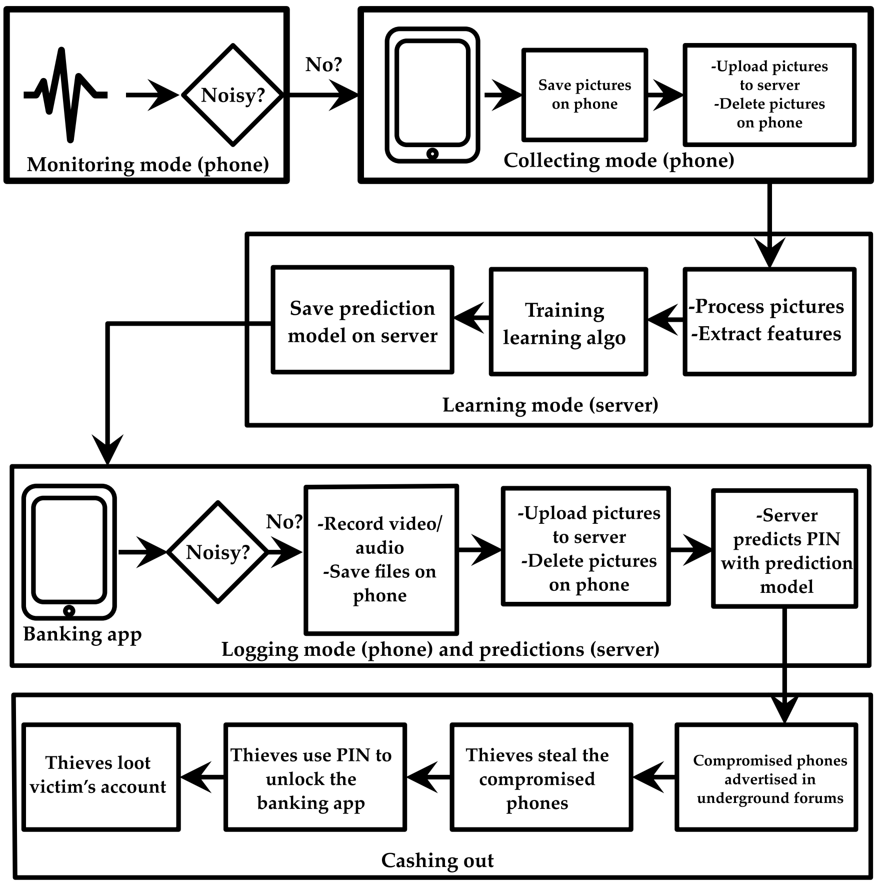

Noisy?
No?
Monitoring mode (phone)
Save pictures on phone
-Upload pictures to server
-Delete pictures on phone
Collecting mode (phone)
Save prediction model on server
Training learning algo
-Process pictures
-Extract features
Learning mode (server)
Noisy?
No?
-Record video/ audio
-Save files on phone
-Upload pictures to server
-Delete pictures on phone
-Server predicts PIN with prediction model
Banking app
Logging mode (phone) and predictions (server)
Thieves loot victim's account
Thieves use PIN to unlock the banking app
Thieves steal the compromised phones
Compromised phones advertised in underground forums
Cashing out

Fig. 3.14 Illustration of the working flow of PIN Skimmer'13. It shows the full attack cycle. It records a video to film the Personal Identification Number (PIN) input. Collecting mode obtains data and deliver it to a server to train the PIN prediction model. Secret PIN associated with Banking app can be predicted using covertly recorded video and the trained model.

PIN Skimmer'13 [124]: This work proposes a phone PIN attack which uses the picture from the front phone camera combined with the sound from the microphone to infer the PINs entered (see Figure 3.14). It is assumed that users will press OK after inputting a PIN. The malicious app used to extract the PIN disguises as a game to ask the user to input PIN numbers. It takes pictures when the user presses a key, and sends these pictures to server. The server will calculate the rotation relation between the images of each number pressed and the image when OK is pressed. These are the features for training a model for PIN prediction.

The malicious app cannot directly get access to the screen to reveal a PIN, but it can obtain access to device information such as camera and microphone. The malware will record a video using front camera and also record sound during the process of inputting the PINs. The server uses the sound emitted when a key is pressed to identify when to take an image from the video to predict the entered key. The evaluation shows that 50% of 50 4-digit PINs after 5 attempts can be predicted. On a 200 8-digit test set, 45% and 60% after 5 attempts and 10 attempts is achieved.

The work uses the acoustic channel as an additional crucial input to determine when the user interacts with the phone. The acoustic channel is used as support in an attack.

Narain'14 [95]: This work studies the feasibility of inferring keystrokes on virtual keyboards on an Android smartphone. Information extracted from microphones and gyroscope of the phone is analysed by tailored machine learning and statistics methods. The work flow is illustrated in Figure 3.15. Different input combinations such as only microphones, only gyroscope and microphone plus gyroscope are evaluated in terms of keystroke detection performance. As expected, attack performance is improved when combining data from both sensor sources.

Several models for the different sensor inputs are used to predict the specific character typed. If more than 75% of the models predict the same key, then no further step is needed. If needed, it is determined in which area of the keyboard a keystroke occurred (the keyboard is divided into areas). Thereafter, thew keystroke is predicted using a voting model comprising the two best-performing algorithms in the area indicated.

The method is evaluated on two smartphones (Samsung S2 and HTC One) and one tablet (Samsung Tab 8), showing more than 90% successful prediction with only one input attempt on the standard Android QWERTY and number keyboards.

This work has similarities with Soundcomber'11 where information from acoustic sensing is combined with visual information. Here acoustic sensing is combined with information obtained from gyroscopes.

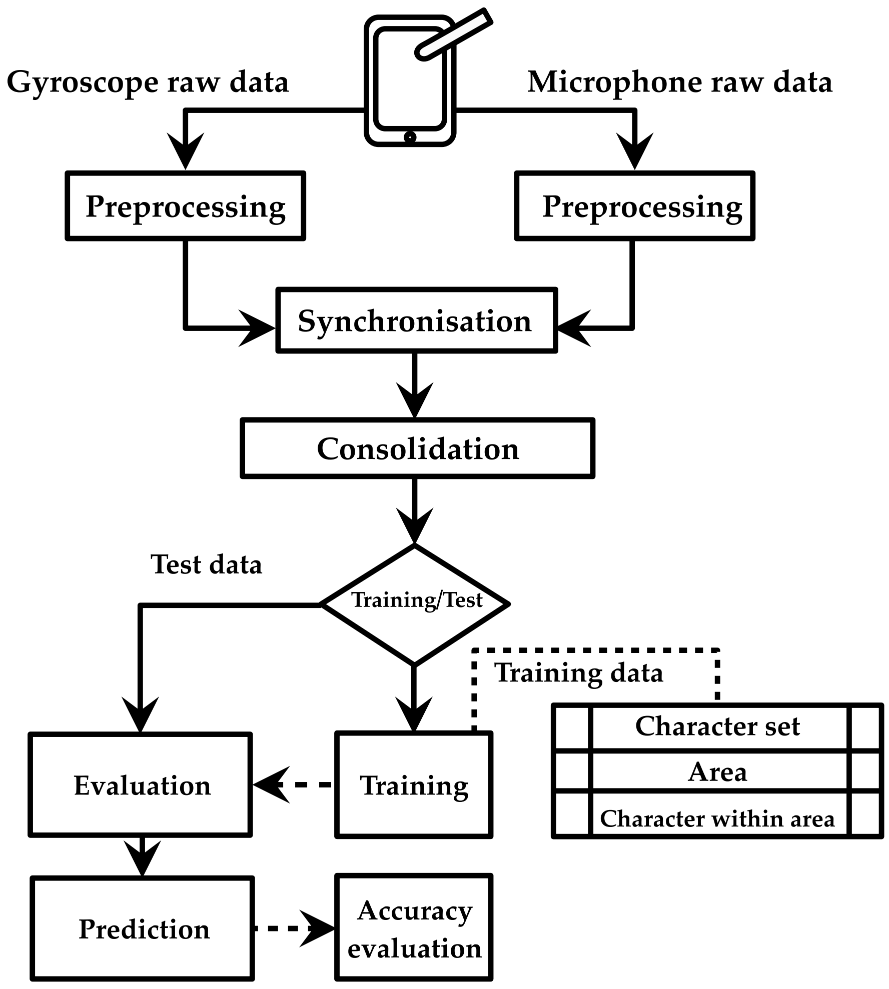


Fig. 3.15 Illustration of the working flow of Narain'14. Gyroscope and microphone raw data are first pre-processed (filtering, extracting and fitting). Then synchronisation is applied for later combining these two for predicton. Consolidation converts data format. Then these data are used for training the model or used as input for keystroke prediction.

Liu'15 [84]: This work introduces a keyboard keystroke snooping attack using the stereo recording from two microphones of a single smartphone. This attack does not require training efforts to label the keystrokes and it exploits TDoA and the unique acoustic MFCC features of a keystroke sound to achieve mm-level accuracy in key locating.

The framework first constructs theoretical TDoA knowledge for each key in a keyboard assuming a known keyboard layout and phone placement. Then it collects the keystroke sound from the keystroke. The sound is analysed to obtain the TDoA which will be matched against each theoretical TDoA value to decide which key was pressed. This method described is called single-keystroke based processing in this work and the efficiency is affected by limitations like the distance between microphones, relative position of the phone, sampling

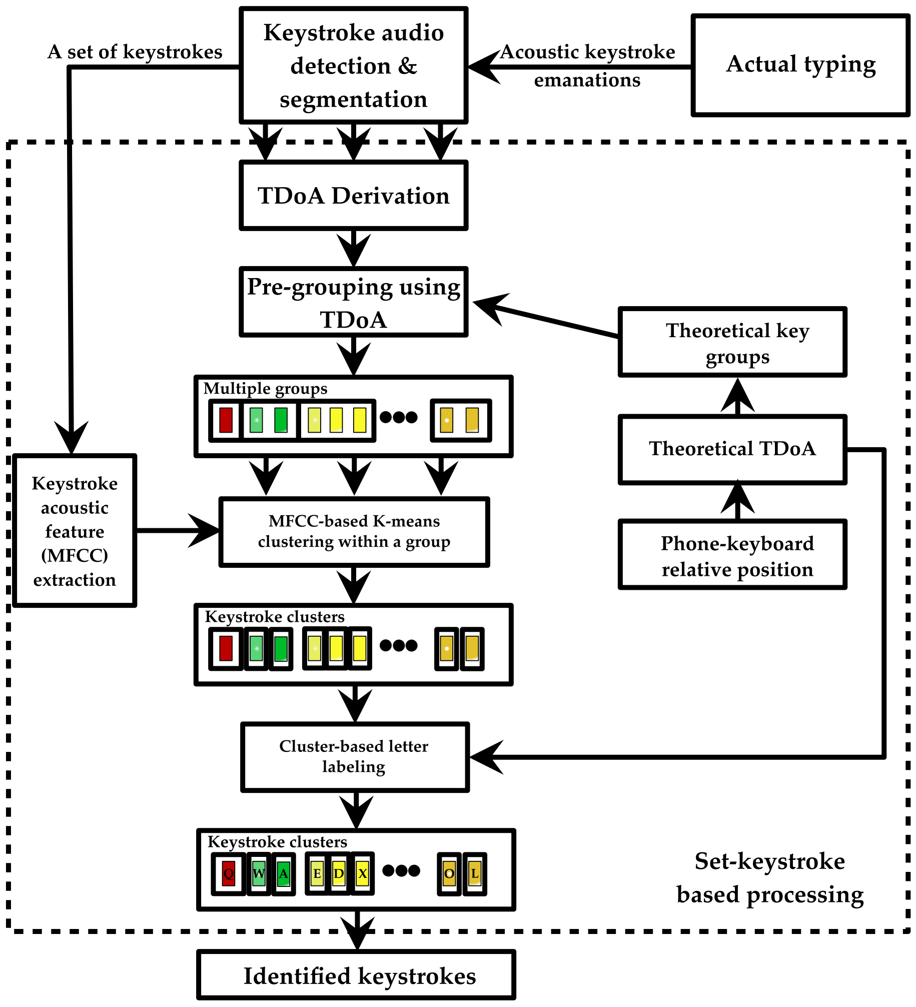

A set of keystrokes
Keystroke audio detection & segmentation
Acoustic keystroke emanations
Actual typing
TDoA Derivation
Pre-grouping using TDoA
Theoretical key groups
Multiple groups
Theoretical TDoA
Keystroke acoustic feature (MFCC) extraction
MFCC-based K-means clustering within a group
Phone-keyboard relative position
Keystroke clusters
Cluster-based letter labeling
Keystroke clusters
Q W A E D X O L
Set-keystroke based processing
Identified keystrokes

Fig. 3.16 Working flow of the set-keystroke based processing in Liu'15. Key observation is first categorising keystrokes into indistinguishable groups with the help of theoritical threshold prepared offline. Then MFCC is used to further predict keystrokes from the same key within the each group. Mean TDoA of this new group is used to match the theoretical one for final prediction

frequency and environment factors like multipath effects and noises. A better solution is set-keystroke based processing. It pre-groups the measured sound based on primitive TDoA measurement, then calculates the MFCC of each keystroke and further makes use of this parameter to cluster keystrokes within one group. Finally it compares the mean TDoA value of these clusters against the theoretical value to cancel the influence of noise and to label the key for each keystroke. Figure 3.16 illustrates the process.

This work considers and tests the impact of different keyboards, sampling rate, placement, multipath, and the presence of line of sight on the attack success rate. It also discusses practical issues of how to get the relative position of phone and keyboard for establishing the theoretical TDoA. Overall, the attack can achieve over 85% accuracy with 48kHz (94% with 192 kHz) identifying a set of keystrokes. If loosing the requirement to top-3 candidate, 97% accuracy can be achieved with 48kHz sampling frequency.

This work differs from PIN Skimmer'13 and Narain'14 regarding the sensing input and the attack target. This work only relies on acoustic information and does not use additional sensors to improve prediction of keystrokes. Also, the monitored keyboard is a real keyboard (not a virtual one displayed on a screen) external to the phone used to record sound. It shows that acoustic sensing with a PVA can provide detailed insight into user behaviour which is security and privacy relevant.

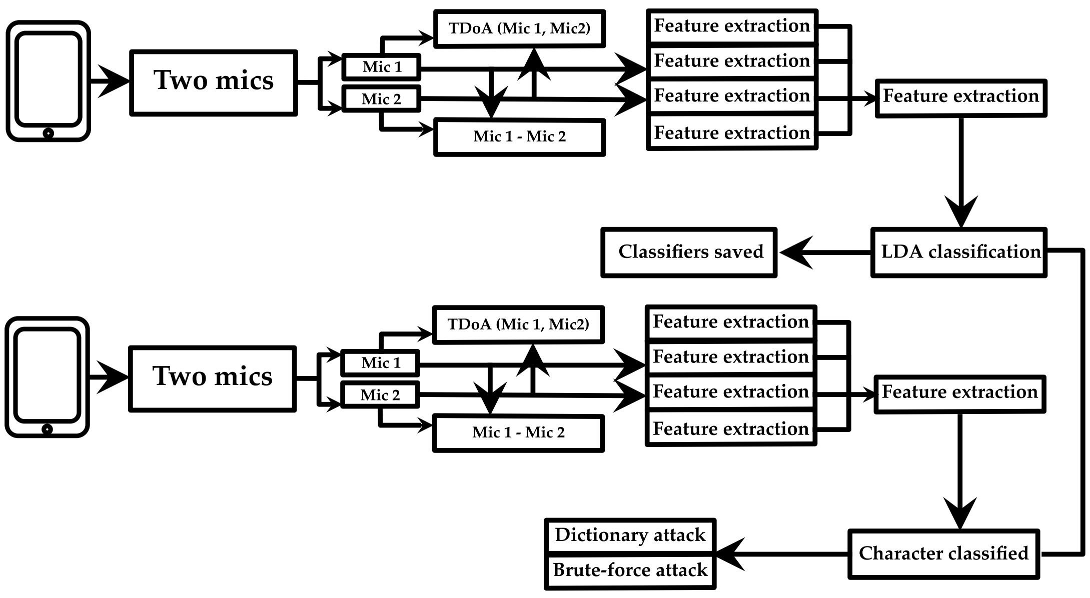


Fig. 3.17 The framework of Hearing Your Touch'19: Stereo recording using two mics on the phone to record sound when tapping on the virtual keyboard happens. TDoAs for each tapping is calculated. Also the cepstrum of the signal recorded by either mic is calculated. These two features are used to train an Linear Discriminant Analysis (LDA) classifier. This classifier is later used for prediction when features of the test data are fed into it.

Hearing your touch'19 [123]: This work proposes the first acoustic side-channel attack revealing the contents typed on the virtual keyboard of a touch screen on a smartphone or a tablet. The sound waves from the finger touching the virtual keyboard travels through the screen surface and the air. This acoustic signal can be captured by built-in microphones on devices. The distortions of the sound wave are related to the tap's location on the screen, so analysing the recorded acoustic signal can unveil the contents of the text users type on the screen.

The work uses pre-processed TDoA and the cepstrum of the first 128 samples of the audio data acquired by microphones after a tap on the virtual keyboard as main features. These features are input to an LDA classifier for keystroke prediction. The described workflow is shown in Figure 3.17.

Experiment results from studies with 45 participants are: 1. it is possible to recover 146 out of 200 PINs (4-digit) with 10 attempts on a Nexus 5; 2. it can predict 8 out of 27 random words with 20 attempts from the benchmark dataset corn-cob consisting of words with 7-13 letters. These attacks achieve better performance on a tablet than smartphone.

This work is different from Liu'15 as the target is the virtual keyboard on the phone, not an external keyboard located in the vicinity. Narain'14 is the most relevant study to this work as they both reveal keystrokes on a virtual keyboard. However, the earlier work Narain'14 combines microphones and motion sensors, and it is assumed that tap events would be reported. This work uses only microphones and also achieves automatic acoustic tap detection. Again, this work demonstrates that PVA can deduce detailed user behaviour from analysis of the acoustic channel alone.

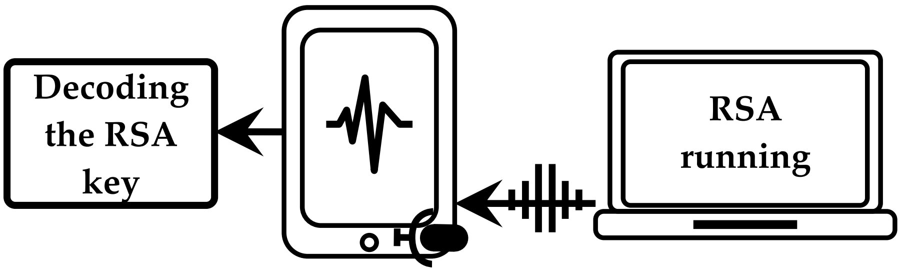


Fig. 3.18 Illustration of the working mechanism of Genkin'14

Genkin'14 [52]: This work introduces a novel cryptanalysis side-channel attack via acoustic emanations. In this work a full 4096-bit RSA decryption key (GnuPG's implementation of RSA) is extracted from laptops, using the noise generated by the electronic components of the computer executing decryption of chosen ciphertexts (Figure 3.18 is a simple illustration of this attack). The noise (acoustic emanations), emitted from the voltage regulation circuits is related to computing activities as the power draw of the CPU varies dramatically depending on the execution patterns of the running algorithms.

Experiments on various laptops are carried out using both a smartphone with the built-in microphone and a distant sensitive microphone. The work describes as well several hypothetical attack scenarios including eavesdropping via compromised mobile devices and dedicated listening devices.

Clearly such acoustic sensing task can also be carried out by a PVA. This highlights the sensing opportunities a PVA listening continuously with high quality microphones to the acoustic channel has.

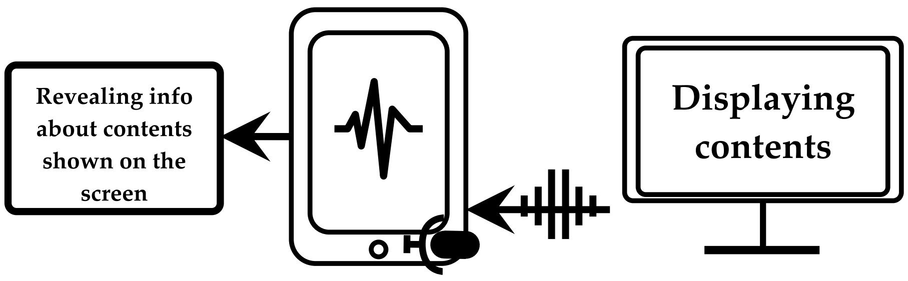


Fig. 3.19 Illustration of the working mechanism of Synesthesia

Synesthesia'19 [51]: This work introduces a novel side-channel attack method to reveal monitor display contents based on the acoustic emanation from the electronic components of the screen. Based on the mechanism of rendering images on a monitor screen, acoustic features are analysed based on repeating strip patterns (Figure 3.19 is a simple illustration). It is found that the spectrogram of the acoustic emanation is related to the pixel period, and the brightness of pixel lines is inversely related to the amplitude of the filtered acoustic signal. Signal processing algorithms are based on these observations. The work showcases different attack scenarios with the help of applying machine learning classifiers. It is demonstrated that detecting user input on the virtual keyboard on the screen, detecting on-screen texts, website fingerprinting attacks (detecting which website is being shown) and Voice over Internet Protocol (VoIP) attacks inferring if the user is watching the video call window or browsing the web are possible. In the later case, the attack can further reveal which website a user is watching (if it is one of the popular sites which has been used for training predicting models.)

This work shows that a PVA can potentially use sound cues to sense how users interact with electronic devices.

Diapoulis'18 [43]: This work describes how individuals can be identified using sound recordings of people walking on a wooden floor.

The acoustic event (walking sound) is detected by estimating the beginning of transient sound. For each event (onset) 11 features are extracted. The feature space is large considering the amount of events multiplied by 11 features. Two synthetic feature subspaces are created by applying principal component analysis (PCA). The first one is a 6-dimension feature subspace created using all 11 features, and the second is a 3-dimension one using only 5 out of these 11 to reduce skew. Last, LDA is performed to classify which individual the event belongs to.

The classifier is built with 10-fold cross validation with 85% of the data and tested using the remaining 15%. The evaluation results show that the 3-dimension subspace works better. Previous work aimed at walking gait recognition using computer vision, but this work shows the possibility of doing so by sound.

The work shows that a PVA can use sound cues in general to infer user behaviour in the vicinity of a device.

Shen'20 [122]: Based on the fact that a PVA is usually placed near a wall with power outlet, this work develops a PVA-tailored Angle of Arrival (AoA) algorithm. They also estimate the geometry of a nearby wall reflection. Both the AoAs and the parameters of the wall are used to calculate the sound source location.

A novel iterative align-and-cancel algorithm to first calculate the LOS direct path AoA of the acoustic signal and then the AoA of the next signal reaching the PVA which is the fastest multi-path signal reflected by the wall closest to the PVA is used. The location and orientation of the wall are deducted during the system initialisation with the help of the first 15 voice commands. With the two AoA, wall location and orientation and human height and combining practical observations, user location can be obtained. The specific algorithm is developed to achieve localisation with arbitrary sound signals and also to deal with the limited acoustic capabilities of the PVA; the PVA used in this work does not provide a sophisticated microphone array normally necessary for accurate acoustic localisation.

The system achieves 0.44m accuracy in different settings including rooms, clutters and user/microphone positions.

This work demonstrates that a PVA is generally able to locate sound sources within a room.

3.2.4-B Active Sensing

PatternListener'18 [164]: This work focuses on observing unlock patterns used on Android phones. Imperceptible sound is sent and record it with a microphone. In addition, motion sensors on the smartphone are used to detect click actions on the screen. The work is similar to PIN Skimmer'13 and Narain'14 as sensing information from the acoustic channel is combined with additional sensor data. However, active sensing is used here in the acoustic channel.

One microphone is used to record the acoustic signal and in addition, motion sensor data is collected. The work uses a finger movement detection technique introduced by Wang et. al [141] which detects phase changes in the emitted sound signal. Coherent demodulation

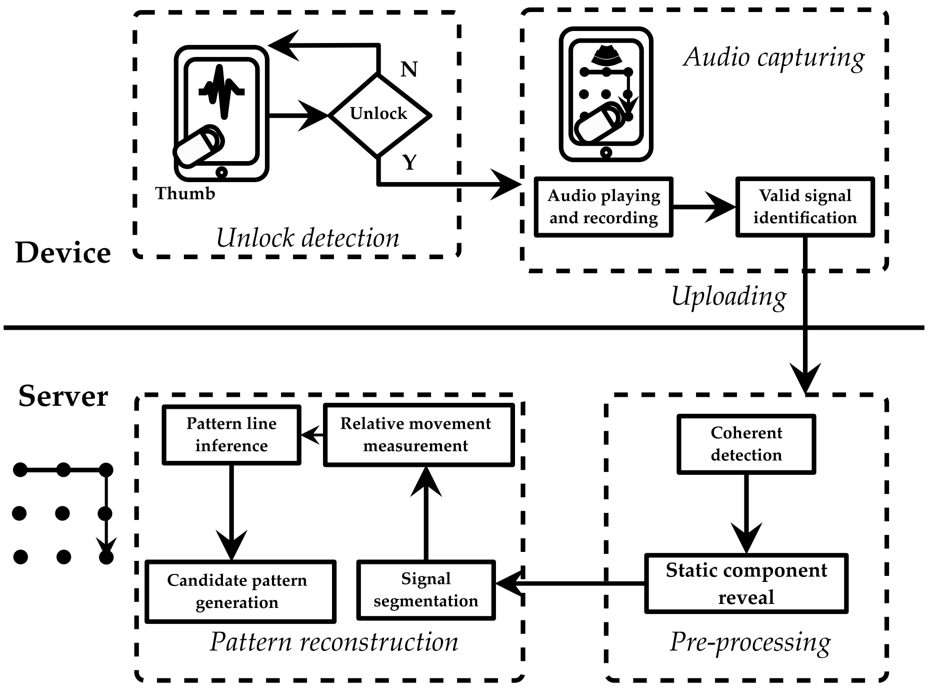


Fig. 3.20 Workflow of PatternListener'18: Unlock detection module detects screen unlock or app unlock activities; Audio capturing immediately emits inaudible acoustic signal and records echoes, and uploads signals related to unlock process; Coherent detection process the data and static component reveal removes static part to leave signal genuinely related to finger movement; Pattern reconstruction finally predicts movement from the signal, recovers lines and infers the candidate patterns

is used to observe phase changes to infer movement distance. The phase change profiles of all possible strokes are used as ground truth. Then user interaction is compared with this ground truth data to infer strokes. A pattern tree algorithm is used to infer complete unlock pattern. This process is illustrated in Figure 3.20.

130 unique patterns categorized as pattern with 2 lines, 3 lines and 5 lines are evaluated. 5 volunteers are asked to draw these patterns which are then predicted. The system can successfully reveal over 90% patterns within five attempts with only one sample drawn by the user.

This work shows that active sensing using PVA speakers is possible to extract information about user behaviour; in this case interaction patterns with the device are revealed.

SonarSnoop'18 [30]: This is my work which is presented in Chapter 7 of this thesis. This work introduces an active acoustic attack aiming at revealing user interaction information with a smartphone. The work differs from PatternListener'18 as only active acoustic sensing is used. In addition, used sensing signal is analysed differently, instead of phase changes several features of the received echo profile are used. The example scenario used is to

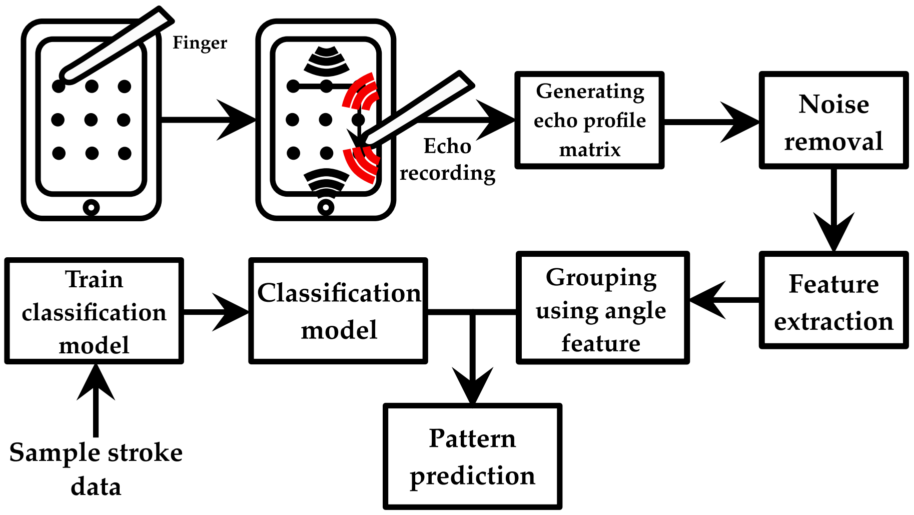


Fig. 3.21 Work flow in SonarSnoop'18: inaudible signals are emitted and echos are recorded while the unlock pattern is drawn; Echo profile matrix is calculated from the recording using DSP techniques; Features including angles and range are extracted from the signal after noise removing; then grouping strokes using the angle features and strokes are further predicted using both angle and range features, thus patterns are identified

discover a user's unlock pattern used on Android devices. Figure 3.21 shows the attack steps. An inaudible acoustic signal (in a frequency range beyond the perception boundary of an adult) consisting of repeated pre-designed signal patterns is used. It is assumed that the malware has access to the microphones and speakers on the phone. The malware transmits the inaudible sound wave through the built-in speakers once the victim draws the unlock pattern to activate the smartphone, meanwhile the malware records all the echoes via the built-in microphones. These recorded echos can be used to profile the finger movement composing the unlock pattern with the help of signal processing techniques like correlation and Gabor filter and machine learning algorithms. Details are explained later in Chapter 7.

CovertBand'17 [94]: Nandakuma et al. examines a covert acoustic sensing method with the help of music to track human presence and movements in both through-barrier and remote scenarios. Also, their method can differentiate different types of motion such as pumping arms, jumping and supine pelvic tilts based on the various spectrograms shown in the echos.

Correlation is used to construct the echo profiles and differentiate moving objects from static objects and by doing so it is possible to locate moving people. Based on this, further analysis of location, movement tracking and motion classification from multiple targets are feasible in most indoor environments. Figure 3.22 is a simple illustration of the application scenario.

Experiments were conducted in five homes, showing that localisation of multiple individuals, even through certain barriers is possible. Considering walking people a mean error

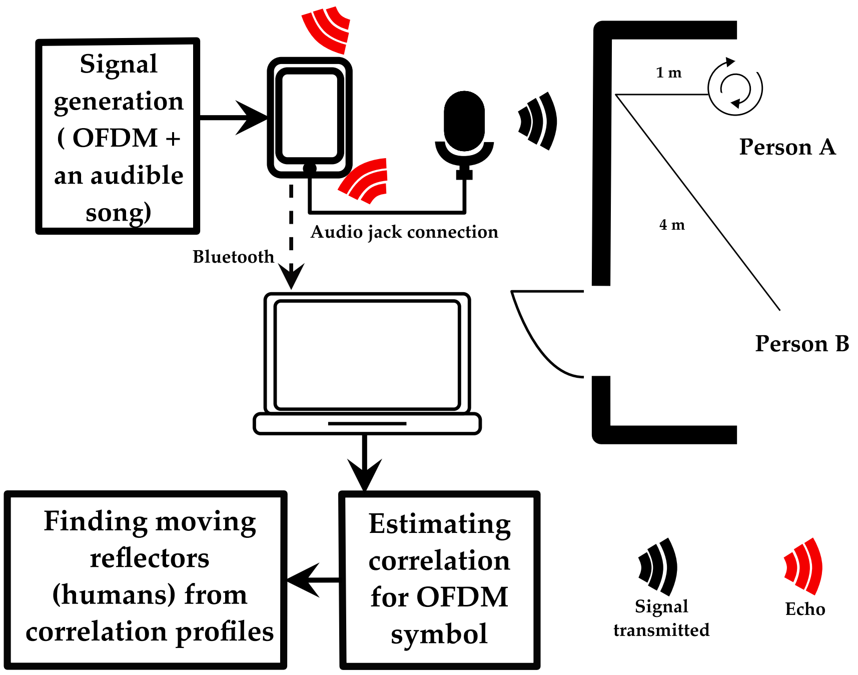

Signal generation (OFDM + an audible song)
Audio jack connection
Bluetooth
1 m
Person A
4 m
Person B
Finding moving reflectors (humans) from correlation profiles
Estimating correlation for OFDM symbol
Signal transmitted
Echo

Fig. 3.22 How CovertBand'17 works: smartphone emits prepared inaudible Orthogonal Frequency Division Multiplexing (OFDM) signal which is further masked by a song. The signal can pass through the walls by tuning volumn and echoes will be recorded by microphones on the phone. Recordings are delivered to a laptop for processing. Then Correlation profiling for OFDM is obtained, then static correlation profile when no person is moving shows all the reflector info in the environment. New correlation profile constructed as a human moves subtracting the static one reveals the movement, so moving reflectors can be found.

of 18 cm is achieved and when considering fixed positions an 8 cm error at up to 6m in line-of-sight and at 3m with barriers is achieved. It is also shown that listeners could only detect in 58% of the cases that the CovertBand'17 sensing signal is played together with songs. As this is close to a random guess it is argued that the sensing signal is not noticeable.

This work shows that normal PVA operations, as for example playing music, can be used to disguise an active sensing signal that collects fine grained environmental information.

BreathListener'19 [151]: This work applies active acoustic sensing to detect driver breath pattern in a noisy car environment and is capable of generating fine-grained breathing waveform in driving environment. Energy Spectrum Density (ESD) is utilised to capture all movements in the environment. Further interference (movements not related to breath) cancellation is performed by background subtraction and Ensemble Empirical Mode Decomposition (EEMD), resulting in Energy Spectrum Density (ESD) signal mainly containing breathing information. It is then transferred to the Hilbert spectrum format. A Generative

Adversarial Network (GAN) based deep learning architecture is used to generate a breathing waveform.

This active sensing work showcase the richness of information extraction active acoustic sensing can achieve. Active acoustic sensing focus on capturing movement information. Since movement is highly related to many types of activity, such as walking, exercising, unlocking phones, and even breathing, active acoustic sensing can indirectly reveal a great amount of information using a PVA.

3.2.4-C Summary

Acoustic sensing is an area of active research. However, few works are directly dedicated to security and privacy within the PVA context. Also, there is less work on active sensing than passive sensing.

Inferring user interaction with a keyboard is one main line of study in the security context as it relates to PIN or password entry. For passive sensing, we include PIN Skimmer'13, Narain'14, Liu'15 and Hearing your touch'19 as representatives as they infer keystrokes on a real/virtual keyboard utilising microphone recordings on a smartphone which fits our PVA definition. For active sensing, PatternListener'18 and SonarSnoop'18 are included. They both implement virtual keyboard active sensing using smartphones and showcase stealing unlock patterns.

Apart from keystroke prediction, acoustic sensing using PVA can reveal other information too as the acoustic signal can not only carry verbal messages but also other environmental information. In passive sensing, Soundcomber'11 attacks credit card secrets using call recordings, Genkin'14 and Synesthesia'19 use sound recordings on a smartphones to reveal cryptographic keys and even the monitor's display content. Diapoulis'18 identifies individuals using walking sounds, and Shen locates sound sources using a PVA. In active sensing, CovertBand'17 shows that physical activities can be differentiated using acoustic signals, and health data such as the breath pattern is extracted in BreathListener'19 using a smartphone.

Work in this section showcases what information can be extracted from the acoustic channel and to what extent it can be used. However, only a few works are directly related to security and privacy in the PVA context. We would like to see more studies making use of PVA acoustic sensing. Acoustic sensing using a PVA is of interest as modern PVA devices provide sophisticated microphone and speaker arrays and additional hardware does not have to be deployed for an attack. Furthermore, a PVAs provide sophisticated processing capabilities (on the device and back-end) an attacker could harness. Also, devices are already deployed at a large scale and do not raise suspicion. Finally, users expect sound to be emitted from these devices enabling an attacker to conceal an active sensing more easily. Existing work on acoustic sensing and security has not exploited these unique conditions to the full. A wide range of interesting security and privacy issues may not have been explored yet.

3.3 Chapter Conclusion and Discussion

PVAs are now commonplace and are significantly changing the way we interact with computer systems. We increasingly depend on PVAs as main or even single interface to computer systems and smart environments. Consequently, security of these devices has become focus of public attention and research efforts. Likewise, privacy is a concern for most users as PVAs record and observe speech, the most fundamental form of human interaction.

In this chapter we have collected the state of the art on cybersecurity research in the PVA domain with focus on the acoustic channel. Some of the work we included is directly applicable to the relatively new PVA domain (e.g., VA) but is not research work conducted specifically with PVA applications in mind. Other work however, such as hidden command injection, emerged alongside the new application area of PVAs. Work of this thesis are also summarised together with related work in this chapter, which illustrates more clearly where they contribute to the acoustic-channel PVA security and privacy research area. They are highlighted in Figure 3.1.

3.3.1 Public Perception and Industry

We observe that there is some misalignment between research efforts and public perception of security and privacy concerns.

The top concern for everyday users is privacy (judging by the number of articles on this topic appearing in news). However, the category *C2 - Privacy* does not list an exhaustive body of work. Even if we include work in category *C3 - DoS*, which in some cases is used as privacy management method, the volume of work is not comparable to work listed in category *C1 - Access Control*. We therefore conclude that significant research efforts should focus on privacy to supply users with tools that help them to address their main concerns. Industry has not yet addressed this issue sufficiently in existing PVA systems and it seems that this is due to a lack of appropriate methods and not due to a lack of user demand.

Issues on access control (Category C1) have also received some media attention. Users are aware of the fact that potentially any user may interact with their PVA and that even commands embedded in songs or adverts played over the radio may be used to interact with their system. However, it seems that pressure from the public is not yet forceful enough to encourage PVA manufacturers to adopt many elements of the large body of research work. However, we believe it is just a matter of time before existing VA techniques and command injection protection methods will see integration with existing PVAs. However, we assume that this will trigger subsequently more research work as adaptation of existing VA methods to PVA will reveal additional challenges that need consideration.

PVA are increasingly used as interface for safety critical systems. For example, PVAs are considered to control equipment in an operating theatre or to be used to control features in a car while driving. If used in safety critical settings availability must be ensured and it is vital to consider the threat of DoS attacks. However, so far this area of research is under-explored. The work we have listed in Category C3 mainly considers DoS as a mechanism for privacy control. To the best of our knowledge, PVA manufacturers have not yet integrated specific measures in products to deal with this situation. Clearly, more research work in this direction should be expanded.

Users are currently very conscious about cameras being used in private spaces. It is common for users to employ a mechanical cover on a laptop camera when not in use, in order to prevent a hacker taking control of the camera. However, this public perception seems not in general to apply to sound systems. Most PVAs do not provide a mechanical switch to disconnect speaker or microphone and where present (as in the Google Home Mini) users rarely make use of it. Users are aware that the device may listen into a conversation (covered in Section Privacy C2) but they are unaware of highly sophisticated acoustic sensing that is possible. For example, acoustic sensing can be used for very detailed imaging. The work in this area is limited but we expect that users will soon demand protection in regard to this PVA aspect. Again, more research work looking at the adaptation of existing acoustic sensing work to the PVA domain is necessary.

3.3.2 Research Directions

C1.1.1 - Acoustic Characteristics - Voice Authentication (VA) is a well researched area and it is possible to authenticate based on a voice profile. However, as this chapter has shown it is necessary to consider VA more specifically in the context of PVA instead of general ASR systems. Research work needs to consider how to apply VA in current PVA and how to protect specifically against spoofing attacks in this context.

C1.1.2 - 2nd Factor Authentication - We believe that work using a 2nd factor for authentication is promising, in particular when this information can also be derived from the acoustic channel. This approach makes such solution very practical. Although some work exists describing 2nd factor authentication, research has not looked into bypassing such methods. This is a crucial step to ensure such methods are robust against attacks.

C1.2.1 - Hardware Non-Linearity - Research has shown that such attacks are very feasible. However, there has not been much work on defence mechanisms against this type of attack. Furthermore, attacks (and the few reported defence mechanisms) rely on sophisticated hardware. We would like to see if the requirement for additional equipment can be overcome.

C.1.2.2 - Obfuscated Commands - Feasible attacks are described. However, this form of attack is not very convincing as noise will still be noted. For this line of work more detailed studies on perception of audio samples are required to fully determine feasibility of these attack types.

C.1.2.3 - Adversarial Commands - This line of work has produced very sophisticated attacks. Complex sentences can be embedded in audio samples, hidden from users. Psychoacoustic masking is increasingly used and attacks over the air considering room characteristics are feasible. However, most attacks still consider white-box scenarios, the internal structure of the ASR is known. Work should consider black-box scenarios and investigate how to craft hidden commands effective on different ASR. This field of work would benefit from a standardised evaluation environment to make attacks comparable. Research here should target the latest attention/transformer-based end-to-end ASR. Finally, more work to defend against such powerful attacks should attract more research work.

C.2.1 - Privacy Preservation - There is little work on privacy preservation in general and more work in this area is required. Work has looked at transforming speech signals such that user specific features (voice profile or paralinguistic information) cannot be extracted. Although such methods are effective, it is not clear how they can be integrated with existing systems and how a user would exercise control.

C.2.2 - Consent Management - Only one work has so far investigated how users can provide consent. Some work on DoS has been carried out as mechanism of revoking consent. We believe that this would be an important area for users and that more research in this domain is required.

C.3.1 Skill Market - The chapter has shown that the operation of the skill markets represent an attack surface. The mapping between voice commands and actions can be exploited by an attacker. Transcriptions of speech are subject to errors which can be exploited. However, a full scale systematic misinterpretation analysis is yet to be completed followed by work proposing suitable defense mechanisms.

Specifically, for the misinterpretation error, only more studies into the misinterpretation error of current COTS skill market are carried out, more advanced skill name cencership algorithm can be designed. For attacks exploiting implicit capability of skills such as skill switch, defence like static algorithm trying to detect malicious skill at the back end may fail as the implementation details of the registered skill is unknown and only the interface is registered at the market, which means malicious skill can evolve all the time. Therefore, as mentioned in Dangerous Skills'19, dynamic way of analysis is promising which can try to invoke malicious action from potential adversarial skills. Such skills can be discovered once suspicious actions are spotted.

C.3.2 Jamming - Jamming of PVAs via the acoustic channel is feasible. Noise can be added to prevent a PVA from functioning. Existing work does not use sophisticated jamming methods (i.e. inaudible jamming, jamming preventing detection and localisation). Also, jamming so far had the aim to block a signal entirely; however, it might also be possible to add interference very targeted to introduce ASR more subtle transcription errors. Defence methods to detect jamming or to design PVA resilient to jamming are missing.

C.4.1 Passive Sensing - The chapter has shown that the acoustic channel can provide a rich set of information in addition to speech. The acoustic channel has been extensively used to infer user interaction patterns with devices (mainly interaction with phones). It has also been shown that a wide variety of other user behaviour (walking, eating, sitting) can be inferred. However, an detailed analysis of what information can be extracted via a PVA is missing. Also, no defence mechanisms against the use of a PVA as acoustic sensor has been reported.

C.4.2 Active Sensing - The work in this category is similar to the line of work on passive sensing. However, as active signal generation is used, more detailed information can be obtained. It has not yet been investigated in detail how active sensing can be carried out on smart speaker type PVA, work so far has focused on phone based PVAs. Specifically how an active sensing signal can be hidden or embedded in expected audio signals (hidden sensing) has not attracted work. For example, sound (voice, music, ...) emitted from a smart speaker could be designed such that it functions well as active sensing signal too. Work on how to detect or defend against an active acoustic sensing signal has not been explored yet.

Chapter 4

Adversarial Command Detection Using Parallel Speech Recognition Systems

4.1 A Defence Method against Adversarial Commands Targeting ASR

A PVA can be integrated as functionality in other devices such as smartphones or TVs or may be implemented as dedicated device referred to as smart speaker. We use PVAs to interact with infrastructures such as our smart home and services such as e-mails and news.

There are a number of PVA security and privacy concerns and research has investigated a large variety of attacks on these systems. One prominent attack example is the so called *adversarial attack* as introduced in Section 2.4.3. Related work about it is presented in Section 3.2.1-B3. The aim of such attack is to supply a specially crafted voice signal, referred to as adversarial command, to the PVA which is interpreted differently by the PVA than it is by humans. For example, the supplied adversarial command may be interpreted by humans as *"Alexa, tell me what the weather is like"* while the ASR of the PVA interprets this signal as *"Alexa, open the front door"*. An adversarial command is created by adding small perturbations to an audio recording until the PVA's ASR recognises the intended command of the attacker instead of the command contained in the original audio recording. If the perturbations are small and added carefully, a human will not notice the modification of the audio signal while the ASR algorithms recognise different words.

How to create adversarial commands has been studied in detail 2.4.3, different methods exist to generate these [11, 130, 69] and studies have been carried out demonstrating their effectiveness [67, 11, 24, 156, 130, 69, 118, 152, 108, 129, 120, 28]. Less effort has been put into devising defence methods against this serious attack form [153].

In this chapter we describe a novel defence method against adversarial commands. Our method makes use of a second ASR - we call it the *protection ASR* - component within a PVA which analyses the supplied voice sample in parallel to the *main ASR*. The speech transcription output of the protection ASR is compared with the transcription output of the main ASR and only if both outputs are a close enough match the transcription output is accepted and the command is executed. The protection ASR uses different training data or even an entirely different ASR architecture compared to the main ASR. Thus, it is infeasible for an attacker to add unnoticeable perturbations to the original audio such that two entirely different ASRs are tricked into producing the same transcriptions.

In this chapter we demonstrate the feasibility of using a protection ASR to prevent an PVA from processing adversarial commands. We use the Kaldi ASR as main PVA ASR and demonstrate that an effective protection ASR based on either PocketSphinx [65] or Kaldi can be constructed.

The protection ASR does not have to produce the same transcription quality as the main ASR. Speech recognition of this component must only be sufficiently accurate to provide protection, transcription accuracy is delivered by the main ASR. Thus, the protection ASR can be simpler and can also be based on much smaller training data. Thus, it is possible to implement the protection ASR without much resource requirements and it is possible to use frequent re-training. Frequent re-training adds additional complexity for a potential attacker that may try to craft an adversarial command targeting main and protection ASR jointly. The structure of the protection ASR in this case is a moving target.

The main contributions of this chapter are:

- *Adversarial Command Detection (ACD):* We describe a novel protection mechanism against adversarial commands using parallel ASR systems.

- *Demonstration of ACD:* We demonstrate the effectiveness of ACD using 20 adversarial commands and show that our ACD using PocketSphinx and Kaldi can detect all adversarial commands. We also show that the ACD can be set to not prevent normal PVA operations due to false positives while still maintain certain adversarial command detection sensitivity.

- *ACD Complexity:* We show that the protection ASR can be significantly less complex than the main ASR in terms of architecture and training data. Thus, frequent retraining of the protection ASR is feasible, providing a ACD as moving target defence.

The remaining chapter is structured as follows. Section 4.2 provides a very brief refresh to ASR and describes adversarial command generation (more details are in Section 2.2 and Section 2.4.3). Section 4.3 discusses related work. Section 4.4 introduces our novel Adversarial Command Detection (ACD) method. In Section 4.5 and Section 4.6 we describe our evaluation setup and experimental results. Section 4.7 and Section 4.8 discusses and concludes the chapter.

4.2 Preliminaries

PVA working principles, adversarial commands and their generation are described in Section 2.1, Section 2.2 and Section 2.4.

In this work we assume one local ASR component is used to implement a PVA and we do not distinguish wake word recognition ASR and back-end ASR. However, our work can be applied to systems that distribute ASR.

In this chapter we focus on methods for hidden command detection of the third type: adversarial commands. We focus on this specific type as it is considered the most effective attack and consequently attracts currently most research effort.

4.3 Related Work

Overview of existing studies on adversarial commands targeting PVA are presented in Section 3.2.1-B3. In this section we stress the relation between existing adversarial example studies and our work.

The hypothesis about the defence proposed in this chapter is state-of-the-art targeted adversarial attack has limited transferability, which means adversarial examples generated based on one specific ASR cannot be equally effective on other types of ASR in terms of attack performance. To our knowledge, few existing studies have tested the efficacy of their adversarial examples on other ASR. Even some trials have been tested [156], the tested ASR is a proprietary one (iFLYTEK [66]) which might use the similar architecture as Kaldi which their attack samples are generated based on and the test samples are simple and limited. Also, their samples can not succeed on DeepSpeech which has a different architecture than Kaldi unless further tuning their examples with the algorithm [24] which aims at DeepSpeech for adversarial attack. To our knowledge, existing white-box adversarial attacks targeting ASRs barely show or even test the attack transferability. In addition, existing black-box adversarial attacks barely test transferability as well. In Abdullah'19-2 [6], although their attack shows reasonably good transferability, it is a *evasion* attack which is considered successful if the

ASR mistranscribe the audio or the AVI misidentifies the speaker. This type of attack is not as strong as targeted adversarial attack which hasn't shown great transferability in any existing studies. VMask'20 [159] tests black-box attacks on unknown system but it aims for ASV rather than ASR system. Naturally we assume to design a defence mechanism based on the weak transferability of adversarial attack in this chapter. Currently there is a serious lack in defence research in this area. Although the intuition of our defence seems very straightforward, there does not exists any work proving the efficacy of it. We cover this gap and propose insight about this defence framework.

4.4 Adversarial Command Detection (ACD)

In this section we describe our ACD method. We first describe the threat model that is considered and then outline our countermeasures in form of deploying parallel ASR.

4.4.1 Threat Model

The attacker may have access to a PVA's ASR when crafting adversarial commands. In addition, an attacker may also have access to the protection ASR we propose as defence method. We detail next the level of access to the PVA we assume an attacker will have.

A1: General PVA Access: We assume the attacker is only able to inject commands via the audio channel. There is no way for the attacker to bypass audio processing entirely (i.e. by a conventional hack of the PVA which may allow the attacker to bypass ASR entirely).

A2: Hidden Command Access via Adversarial Command: We assume that the attacker is only able to supply a rogue command as a hidden command constructed as adversarial command. We assume that the attacker must submit a command within an audio sample such that a present user is not aware of this embedded threat. We do not consider direct non-authorised interaction with the PVA or submission of hidden commands using different techniques (see Section 2.4.1).

A3: Main ASR Access: We assume that the attacker has access to the main ASR; i.e. we consider a white-box attack. The attacker has full access to the main ASR when crafting the attack signal. This is a reasonable assumption as it is not feasible to keep an ASR used for millions of devices a secret (The PVA industry does not make their ASR systems public available but we cannot assume that an attacker may not gain access).

A4: Protection ASR Access: We assume that the attacker does not have access to the protection ASR; i.e. we assume a black-box attack. As we will show, it is possible to frequently retrain the protection ASR which ensures that an attacker is not able to obtain a copy of the used protection ASR. It has to be noted that there is currently no study showing that it is feasible to construct an adversarial command targeting multiple ASR in parallel under either white or black-box assumption, which is described in Section 4.3. However, making the assumption that the protection ASR is black-box will make this problem significantly harder as the attacker would need to craft an adversarial command working with all possible ASR configurations at the same time.

4.4.2 Assumptions

Other assumptions are we don't consider adversarial attacks launched over the air but directly feeding the adversarial commands to ASR systems. It is a more practical situation where an attacker plays the adversarial examples near an PVA and these audio signals travel through the air towards an PVA, and considering direct feeding commands to the ASR excludes the inferences the physical environment has on these signal. This assumption gives the attacker an advantage to maintain the fidelity of their attack commands and require more robust defence method to protect the ASR from being tricked by them.

4.4.3 ACD Approach

The ACD approach is shown in Figure 4.1. The *main ASR* of the PVA is accompanied by a *protection ASR* and both process the incoming audio signal. Both ASR may share the same front-end (Microphone, filters, gain control, ...) and both ASR produce a transcription of voice. The main ASR transcription is only passed on to the PVA command execution if a comparison of both transcriptions determines that no adversarial command is present. Next we describe this process in more detail; in the next subsection we then discuss the internal structures and relationships of the used ASRs.

1. An input signal x is provided. Either the audio input signal is provided by a human speaker or it may be supplied by an attacker via a loudspeaker. In case of an attacker the supplied signal is the adversarial command.
2. The voice audio signal x is fed to the main ASR used to transcribe the voice signal into text y.
3. The main ASR is a sophisticated ASR, using a complex structure and trained with a large corpus to provide an accurate transcription y for a diverse set of speakers. The transcription process is described as: $y = f_M(x)$.

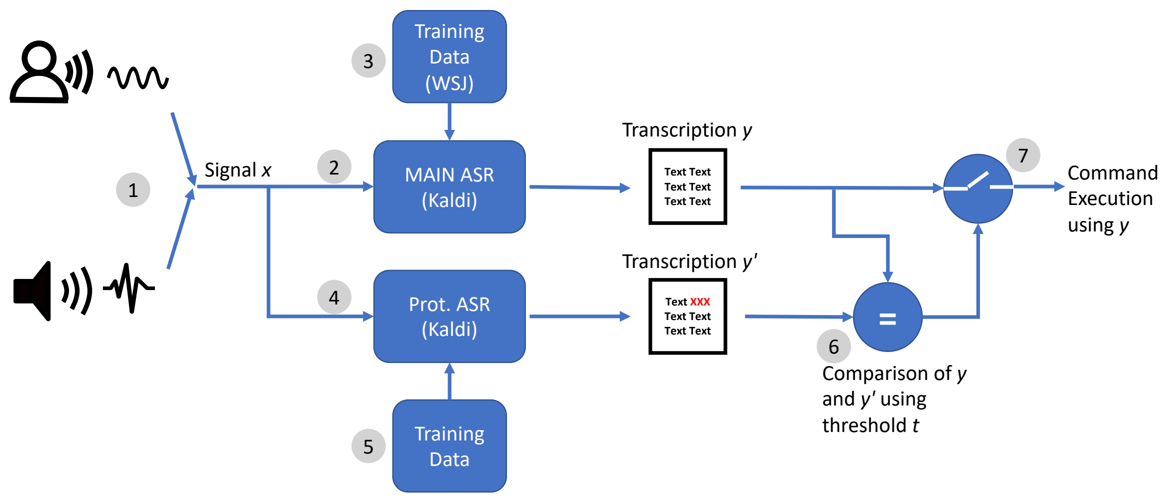

Training Data (WSJ)
3
Signal x
1
2
MAIN ASR (Kaldi)
Transcription y
Text Text Text Text Text Text
7
Command Execution using y
4
Prot. ASR (Kaldi)
Transcription y'
Text XXX Text Text Text Text
=
6
Comparison of y and y' using threshold t
5
Training Data

Fig. 4.1 Adversarial Command Detection (ACD) - The audio signal is analysed by the main ASR and the protection ASR in parallel. If both transcriptions differ significantly, defined by a threshold, the command is rejected.

4. A copy of the audio signal is also fed to a protection ASR which creates it's own text transcription y'.

5. The protection ASR is far less sophisticated than the main ASR and is also trained with a much smaller data set. The transcription is less accurate than that of the main ASR. The protection ASR produces the following transcription output: $y' = f_P(x)$.

6. The output of both ASRs y' and y is compared using the metric WER (details of WER is introduced in later Section 4.5.4) against an error threshold t. The WER difference between y' and y should not be greater than t. If the input is a legitimate voice command from a user, the difference between the two transcription is assumed to not be greater than the threshold. The threshold can be selected according to goals of the overall system. For example, it could be set such that the less accurate transcription of the protection ASR does not hamper normal PVA operations while adversarial commands are reliably detected.

7. Only if the recognition difference is below threshold t the transcribed command is considered valid and transcription y is passed to the PVA command execution.

4.4.4 ACD and Protection ASR Properties

To create an adversarial command (see Section 2.4.3) the attacker executes an iterative process in which the signal is modified by adding inaudible perturbations such that the ASR recognizes the desired command. The signal is adapted taking into account the ASR's model which is defined by (i) ASR architecture and (ii) ASR training data. We assume that if the protection ASR model differs from the main ASR it is infeasible for an attacker to modify iteratively the input signal such that two entirely different ASR are forced to produce the same transcription while ensuring that signal modifications remain inaudible.

4.4.4-1 Different Architectures Different ASR architectures may use different types of input features and may also extract different internal features. The adversarial example generation is an iterative optimisation process which aims to find the optima for the loss function with respect to the input by adding perturbation. It will be difficult for an attacker to add perturbations with the aim of changing features important for one architecture without changing features important to the other. It is likely that combined optimisation for two distinctively different ASR architectures will require much higher degrees of freedom in adding perturbations. Thus, to achieve such optimisation it may well be that perturbations above the human perception level are required which then would make the adversarial commands audible.

Therefore, we argue that the protection ASR should use a different architecture compared to the main ASR as this increases difficulties for an attacker in circumventing the proposed ACD.

4.4.4-2 Different Training Data The details of the model parameters used by the ASRs depend on the used training data. The parameter values keep being modified to improve prediction results during the training process. Thus, when generating an adversarial command the attacker needs to take into account not only the ASR architecture but also the specific trained model. An adversarial command generated for an ASR may only work when the same trained model is used.

Again, an attacker may aim to construct an adversarial command by taking into account two different models resulting from different training data. This might be an easier task than taking two entirely different ASR architectures into account as both ASR make use the same features. However, the protection ASR can be frequently retrained which makes the model of the protection ASR a moving target. The attacker will have to consider all possible trained models.

Therefore, we argue that the protection ASR should use a frequently changing trained model as this increases difficulties for an attacker in circumventing the proposed ACD.

4.5 Evaluation Setup

The ACD is evaluated using a number of different ASRs in the role of a protection ASR. A number of adversarial commands are used to evaluate detection ACD capabilities. Benign commands are also used to evaluate how ACD behaves in a normal user interaction scenario.

4.5.1 ASR Selection

Our selection of ASRs used for evaluation is summarised in Table 4.1. Each ASR model (comprising architecture and model) is given a label which we use in the remaining document for reference.

MASR1 As main ASR (Label *MASR1*) we use an nnet2 Kaldi model which is a DNN-HMM structure using the Wall Street Journal (WSJ) corpus as training data. Note that the latest Kaldi is an nnet3 chain model. However, we use a nnet2 Kaldi as the main ASR as we use adversarial command generation based on work by Schönherr et al. [118] which relies on this ASR variant.

The nnet2 Kaldi used by Schönherr et al. makes use of some modifications. The feature extraction and the DNN acoustic model are combined. This integration is for the convenience of adding perturbation directly to the input rather than the intermediate features when generating adversarial commands. According to Schönherr et al., this design modification does not affect the accuracy of the ASR system. We treat this modified nnet2 Kaldi model and the standard one as equivalent in this work. When evaluating adversarial commands we use this modified nnet2 Kaldi ASR; when evaluating benign commands we use the standard nnet2 Kaldi ASR.

PASR1 The first candidate of a protection ASR (Label *PASR1*) is the PocketSphinx running on a standard Raspberry Pi 3 Model B+. PocketSphinx is using a GMM-HMM model, while the the main ASR MASR1 is using DNN-HMM model. These two models have completely different acoustic model architectures. Although it is not clear what corpus is used to train PocketSphinx, it is safe to assume PocketSphinx is trained with different training corpus than the main ASR (PocketShinx is provided with the already trained model and a clear description of the used training data is not provided). GMM-HMM training requires less

Table 4.1 Kaldi using an nnet2 model is used as main ASR (MASR1). Three different ASR are used as protection ASRs, labeled PASR1, PASR2, PASR3 and PASR4. The protection ASRs are PocketSphinx and different Kaldi variations.

ASR Label	**ASR Variant**	**ASR Architecture**	**Training Data**
MASR1	Kaldi nnet2	DNN-HMM	WSJ
PASR1	PocketSphinx	GMM-HMM	Unknown1
PASR2	Kaldi nnet3	DNN-HMM	Unknown2
PASR3	Kaldi GMM	GMM-HMM	WSJ
PASR4	Kaldi GMM	GMM-HMM	50% of WSJ

resources and is considered to be an older fashion of ASR compared to DNN-HMM model. However, as the protection component can handle a lower transcription accuracy for the benefit of less complexity PocketSphinx is a suitable choice.

PASR2 The second candidate (Label *PASR2*) is from an open-source project called Zamia Speech [104] which provides pre-built Kaldi ASR packages for Raspbian (A commonly used OS for Raspberry Pi) complete with pre-trained models for English. It uses Kaldi nnet3 chain audio models. nnet3 and nnet2 are both DNN, but nnet3 supports more general networks. Therefore, we treat nnet3 as a variation of the Kaldi DNN. Specifically, we use $kaldi-generic-en-tdnn\_f$ which is a pre-trained nnet3 chain model trained on 1200 hours of audio. We treat it as an ASR with different architecture and trained with different training dataset compared to the main ASR.

PASR3 The third candidate (Label *PASR3*) is the standard GMM-HMM model from Kaldi trained using the the Wall Street Journal (WSJ) Corpus. Note that although the principal architecture of this ASR is the same as for PocketSphinx, but the specific parameters are different. Note this candidate is trained using the same dataset as the main one, but it has a completely different architecture.

PASR4 The fourth candidate (Label *PASR4*) is identical to PASR3 except the used training data. Only 50% of the Wall Street Journal (WSJ) Corpus are used for training the ASR.

In summary, MASR1 uses a DNN-HMM architecture using the WSJ corpus as training data. PASR1 uses a different architecture and different training data compared to MASR1. PASR2 shares the architecture with MASR1 but uses different training data. PASR3 shares the training data with MASR1 but uses a different architecture. PASR4 uses a less complex training dataset compared to PASR3.

4.5.2 Adversarial and Benign Command Generation

Adversarial Commands We generate the adversarial commands based on work by Schönherr'19 et al. [118][1]. 20 adversarial commands are generated; the commands are hidden in 20 music segments as provided on the GitHub repository [113].

Benign Commands To show how the main ASR and the defence ASRs perform in a normal setting we also generate benign commands. We generate 20 benign commands based on the transcriptions of the 20 adversarial commands. The 20 commands are spoken by a female native speaker with a neutral accent. Note that the recording environment is a normal room using COTS iMac microphone and the open source software Audacity rather than using professional recording equipment in an anechoic room.

4.5.3 Experiment Setup

We conduct four sets of experiments. For each set we used the main ASR MASR1 and one of the four protection ASRs PASR1 to PASR4. In each experimental set we evaluate how the 20 adversarial commands and the 20 benign commands are classified by the ACD. The audio commands are directly fed into the main ASR and the protection ASR candidates respectively. We do not use a speaker and microphone in the processing chain as described in Section 4.4.2.

For each adversarial and benign command, we compare the decoding results between the main ASR and the protection ASR using the WER metric (see next paragraph for details). The ACD decision (detection of an adversarial or benign command) in dependency of WER threshold t is recorded. Based on the ground truth we record if this was a true positive (TP), false positive (FP), true negative (TN) or false negative (FN) decision.

True positive means that we decide the command is an adversarial one and the decision is correct; false positive means we decide the command is an adversarial one and it turns out the command is benign; true negative means we decide the command is a benign one and this decision is correct; false negative means we decide the command is a benign one but actually it turns out to be adversarial.

Each experimental set produces therefore the following result set:

$$R = \{TP(t), FP(t), TN(t), FN(t)\} \tag{4.1}$$

[1]We are grateful that the authors provided us with their trained model used to generate adversarial commands.

The result set describes the performance of the ACD in normal situations and under attack.

4.5.4 Evaluation Metrics

Word Error Rates (WER) We use WER as the metric to evaluate ASRs transcription accuracy as it is the most common metric used in this research context. WER is defined as

$$WER = \frac{N_{sub} + N_{ins} + N_{del}}{N_{ref}} \tag{4.2}$$

where N_{sub} is the number of words which are incorrectly transcribed, N_{ins} is the number of words which appear in the current transcription but are not present in the reference, and N_{del} is the number of words in the reference that do not appear in the transcription. WER is computed using a string edit distance on the word level (derived from Levenshtein [81]). Note that WER can be greater than 100% as the transcription can be longer than the reference.

In our experimentation we use WER to evaluate ASR transcription performance using the obtained transcription of adversarial commands, obtained transcription of benign (human spoken) commands and the reference texts of these commands (the texts for both adversarial commands and benign commands are the same) as reference. In the context of ACD we evaluate the transcription of the protection ASR (i.e., PASR1, PASR2, PASR3 and PASR4) using the transcription of the main ASR (i.e., MASR1) as reference. The obtained WER is compared with threshold t to decide if the command is adversarial or not.

Receiver Operating Characteristic (ROC) We draw ROC curves for the ACD with the false positive rate as the x-axis and the true positive rate as the y-axis. Each of these ROC curves shows the false positive rate versus the true positive rate for all possible ACD decision thresholds. Using ROC curves the performance of different ACD configurations can be directly compared in one figure. The true positive rate TPR is defined as:

$$TPR(t) = \frac{TP(t)}{TP(t) + FN(t)} \tag{4.3}$$

TPR is also called *Sensitivity* or *Recall* introduced later. The FPR is defined as:

$$FPR(t) = \frac{FP(t)}{FP(t) + TN(t)} \tag{4.4}$$

For each ROC curve, the AUC is calculated and the ACD has a better prediction skill the greater the AUC value is. An ACD with no skill has an AUC of 0.5 and a useful ACD mut provide an AUC value above 0.5.

Note that ROC is useful in situations where there are roughly equal numbers of observations in the positive and negative classes. In our evaluation, the number of hidden commands and benign commands are both 20 which fits this assumption.

F Score We use the *F score* as one method to decide on the threshold of the ACD mechanism. F score is a very common metric for choosing threshold of a classifier. F score is defined as

$$FScore(t) = \frac{Precision \cdot Recall}{Precision + Recall} \tag{4.5}$$

where $Precision = TP(t)/(TP(t) + FP(t))$ and $Recall = TPR(t)$. Precision reflects of all commands we predict to be adversarial, what fraction actually are adversarial commands. Recall reflects of all commands that actually are adversarial, what fraction have we correctly detect as being adversarial. If the discrimination threshold is higher it means that we would like to only decide a command is adversarial if we are more confident, the *Precision* value will be higher and *Recall* will be lower. Oppositely, if the discrimination threshold is lower it means we would like to reduce the chance that we miss any adversarial commands, the *Precision* value will be lower and *Recall* will be higher. We could decide to find a optimal threshold which results in a high *Precision* and a high *Recall*. This can be achieved by selecting the ACD threshold t such that the $FScore$ is maximised.

4.6 Evaluation Results

We first present a performance evaluation of the five different ASR used (MASR1, PASR1, PASR2, PASR3, PASR4 as shown in Table 4.2 and Table 4.3). Each of the ASR are used to decode 20 benign and 20 adversarial commands. Then we evaluate the ACD performance where different combinations of main ASR and protection ASR are used.

4.6.1 Decoding Results of Normal Speech

The 20 benign commands are fed to the different ASR; the results are shown in Table 4.2. It has to be noted that there is always some level of WER; a low WER is corrected by the PVA during the command execution when the transcription is analysed regarding syntax. It

Table 4.2 The effect of the 20 benign human spoken commands on variations of ASR systems. The commands are best recognised by PASR2 which is the most recent Kaldi implementation. WER is high for other ASRs.

ASR Label	ASR Variants	ASR Architecture	WER
MASR1	Kaldi nnet2	DNN-HMM	58.9%
PASR1	PocketSphinx	GMM-HMM	75.34%
PASR2	Kaldi nnet3	DNN-HMM	15.75%
PASR3	Kaldi GMM	GMM-HMM	69.86%
PASR4	Kaldi GMM	GMM-HMM	64.38%

is notable that WERs in Table 4.2 show decoding results are obviously different from the truth content except PASR2 (WER of 15.75%). Note that normally there will be following Natural Language Processing (NLP) component to analyse intents and semantics of the ASR transcription then corresponding actions will be taken. This step can mitigate some errors generated in the ASR decoding step. Also the reasons why the performances for four out of five ASR models are not ideal are analysed in Section 4.7. Since our goal is not designing a high-performance ASR but verifying proposed defence method, the decoding performance is not the main concern in this work. The newest ASR system PASR2 provides the best transcription results. This is to be expected as it has the most advanced ASR architecture and a large corpus is used for training. The transcription result of PASR3 and PASR4 is similar, even though PASR4 only uses half the training data. The detailed results are as follows:

MASR1 First we use the nnet2 Kaldi (main ASR). Using this ASR results in WER of 58.9% with 5 insertions, 15 deletions and 66 substitutions.

PASR1 The PocketSphinx decoding results of these 20 human spoken commands result in a WER of 75.34% with 19 insertions, 8 deletions and 83 substitutions.

PASR2 The decoding results from Kaldi nnet3 compared to the ground truth transcription result in a WER of 15.75% with 4 insertions, 3 deletions and 16 substitutions.

PASR3 Feeding these commands to the Kaldi GMM-HMM model results in WER of 69.86% with 2 insertions, 66 deletions and 34 substitutions.

PASR4 The decoding results for the 20 human spoken commands are 64.38% with 2 insertions, 31 deletions and 61 substitutions.

Table 4.3 The effect of the 20 adversarial commands on variations of ASR systems. The adversarial commands are only effective on MASR1, the ASR for which the commands were generated. For all other ASR the WER is high, indicating an unsuccessful transcription.

ASR Label	ASR Variant	ASR Architecture	WER
MASR1	Kaldi nnet2	DNN-HMM	12.33%
PASR1	PocketSphinx	GMM-HMM	139.73%
PASR2	Kaldi nnet3	DNN-HMM	98.63%
PASR3	Kaldi GMM	GMM-HMM	97.26%
PASR4	Kaldi GMM	GMM-HMM	100%

4.6.2 Decoding Results of Adversarial Commands

The 20 adversarial commands are fed to the different ASRs. As shown in Table 4.3, it is clear that the crafted adversarial commands are only effective against the ASR used in the adversarial command generation. Any other ASR, differing in architecture, training data or both does not transcribe the commands correctly. It has to be noted that there is always some level of WER; a low WER is corrected by the PVA during the command execution when the transcription is analysed regarding syntax. The detailed results are as follows:

MASR1 The adversarial commands are successful resulting in the best overall WER of 12.33%. Specifically, 2 word insertions, 9 word deletions and 7 word substitutions are recorded when comparing the transcription with the reference. This proves the white-box attack is successful as expected. The adversarial commands are crafted specifically for MASR1. To provide a clearer impression how the adversarial examples are transcribed, the details of the MASR1 decoding results for 20 adversarial commands are presented in the Appendix A[2].

PASR1 The decoding results from PocketSphinx is far from the target, resulting in WER of 139.73% when compared with the reference text. Specifically, 64 word insertions, 11 word deletions and 129 word substitutions.

[2]Each result is compared with the target reference. Index number indicates the sequence of the command; the reference text is labeled as *ref* at the beginning; the decoding result is labeled with *hyp* at the beginning; if each word is correct, substituted with another word, deleted from the reference, or inserted compared to the reference, it is labeled with *C, S, D, I* respectively with *op* labeled at the beginning of the line.

PASR2 When feeding the adversarial commands to the Kaldi nnet3 chain model runnig on a raspberry Pi, none of the target sentences are correctly transcribed. Specifically, WER is 98.63% with 0 insertion, 129 deletions and 15 substitutions.

PASR3 None of the target sentences are correctly transcribed. Specifically, WER is 97.26% with 3 insertions, 101 deletions and 38 substitutions.

PASR4 The results are similar to PASR3 with a reduced training data. Specifically, WER is 100% with 1 insertion, 118 deletions and 27 substitutions.

4.6.3 Adversarial Command Detection

We evaluate the following ACD ASR combinations: MASR1/PASR1, MASR1/PASR2, MASR1/PASR3 and MASR1/PASR4. The 20 benign commands and 20 adversarial commands are fed to the system. For each combination, we evaluate the WER threshold, draw the ROC curve and calculate the F-score.

The ROC curves are shown in Figure 4.2. A good ACD should produce a ROC curve passing close to the top left corner (i.e. from (0,0) via point (0,1) to (1,1)). This would mean that a low FPR would produce a high TPR for certain WER threshold t. A curve following a diagonal (i.e from (0,0) to (1,1) would represent a bad ACD that cannot discriminate and represents a random guess. A good ACD would have an AUC value close to 1 while a bad ACD would have an AUC value close to 0.5.

As shown in Figure 4.2, the four pairs of ASR do not differ significantly in terms of detection performance. However, the ACD using PASR1 (PocketSphinx) provides the best ROC curve. The AUC value for PASR1 is $AUC_{PASR1} = 0.888$ which is the highest among the four ($AUC_{PASR3} = 0.844$, $AUC_{PASR2} = 0.762$, and $AUC_{PASR4} = 0.875$).

The Figure 4.2 and the corresponding AUC value indicates that PocketSphinx is the best protection ASR in our evaluated scenario.

PASR1 We see that PocketSphinx (PASR1) tends to transcribe our 20 music based adversarial commands to longer sentences than the original command transcription, which results in WER values above 100%. This makes it easier to select a threshold with higher true positive rate (TPR) and lower false positive rate (FPR).

PASR2 The Kaldi nnet3 (PASR2) model (which is shown as the orange line in Fig 4.2) often only transcribes a few words or even nothing when fed with our 20 adversarial commands.

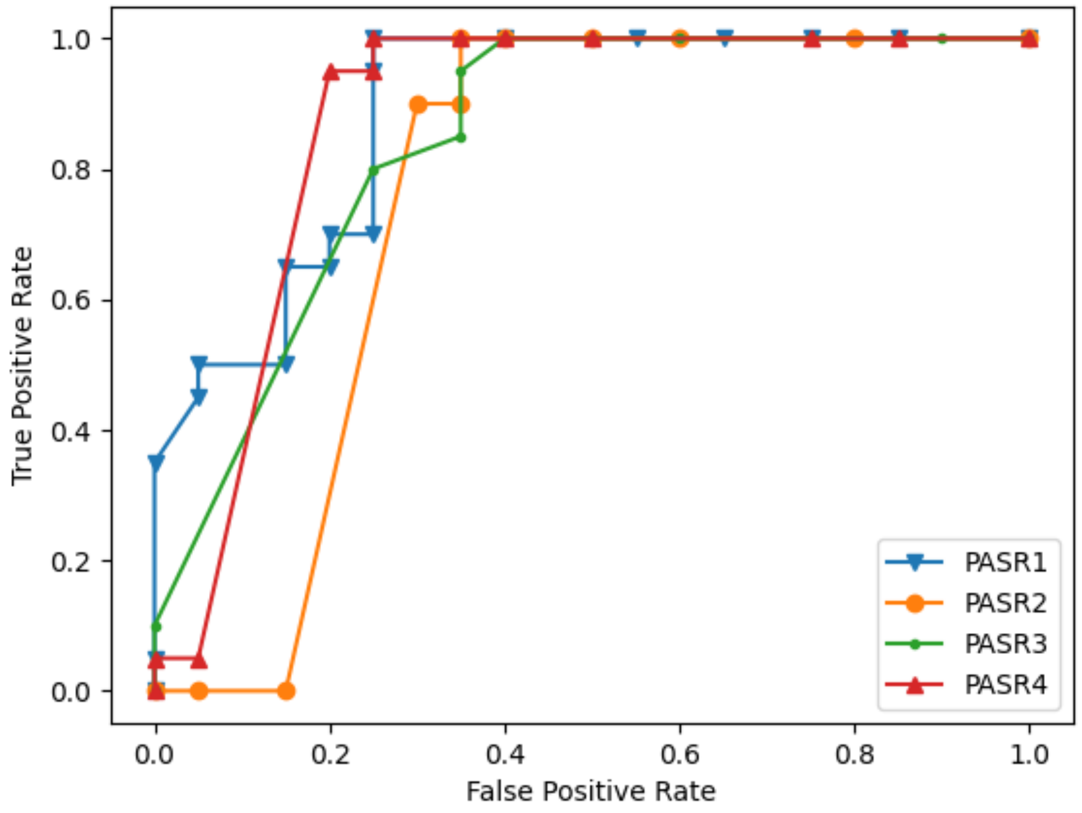


Fig. 4.2 ROC curves for all protection ASR options. PocketSphinx (PASR1) provides the best option for the ACD with an AUC value of $AUC_{PASR1} = 0.888$.

This results in a maximum WER of 100%. When decoding normal human spoken commands, as shown in Table 4.2, Kaldi nnet3 has a significantly better decoding performance in recognising the command content, which also results in a bigger WER difference when it is coupled with the main ASR. Our hypothesis is that Kaldi nnet3 is a better and more sophisticated ASR. when it is used as the defence ASR coupled with the main ASR, due to the capped maximum WER difference of adversarial command decoding result and the greater WER difference of normal human spoken commands, the space for a appropriate threshold is tight. Therefore, it is not the optimal choice here in our setup.

PASR3 and PASR4 The Kaldi GMM-HMM (PASR3) and Kaldi GMM-HMM (PASR4) trained with half of the training data of PASR3 are green line and red line in Fig 4.2 respectively. These two models have similar performances in decoding the 20 adversarial commands. The decoding results for most samples are either shorter than the target transcriptions or nothing can be decoded. There are only two WER results beyond 100% for both PASR3 and PASR4, Overall, the WERs for both PASR3 and PASR4 decoding 20 adversarial commands are high and around 100%, so the true positive value for both these two lines reach 100% easily. When decoding normal human spoken ones, performance of PASR4 is slightly better than PASR3 as shown in Table 4.2, thus PASR4 result is closer to MASR1. There are more high WER values for PASR3 than PASR4, so as the threshold varies the false positive rate of PASR3 grows faster than that of PASR4, which is presented in Figure 4.2.

F-score We calculate the F-score for each experiment set. For the PASR1 curve (AUC of 0.888), the F-scores at each point shown on the curve are [null, 0.095, 0.519, 0.6, 0.645, 0.606, 0.722, 0.703, 0.737, 0.718, 0.864, 0.889, 0.833, 0.784, 0.755, 0.727, 0.702, 0.667] (at point (0,0) the F-score cannot be calculated). A value of 0.889 is the highest score when the threshold is set as $t = 92.31\%$ as the WER. The AUC value of PASR1 is best across all the main/protection ASR combinations.

For PASR2 curve AUC of 0.762, the F-scores for each node are [nan, nan, nan, 0.818, 0.8, 0.851, 0.833, 0.8, 0.769, 0.714, 0.667]. The best F-score is 0.851 for $t = 75\%$.

For PASR3 curve AUC of 0.844, the F-scores for each node are [nan, 0.095, 0.182, 0.78, 0.773, 0.826, 0.833, 0.8, 0.769, 0.727, 0.702, 0.69, 0.667]. The best F-score is 0.833 for $t = 85.71\%$.

For PASR4 curve AUC of 0.875, the F-scores for each node are [nan, 0.095, 0.091, 0.884, 0.864, 0.889, 0.851, 0.833, 0.8, 0.727, 0.702, 0.667]. The best F-score is 0.889 for $t = 88.89\%$.

4.7 Discussion

Our experiments demonstrate that it is in principle feasible to construct an ACD as outlined in this chapter.

4.7.1 Findings

The experiments show that any ASR that differs in architecture and/or training data is usable as protection ASR. In our experimentation setup, PocketSphinx (PASR1) turned out to be most effective but the other candidates are usable too. Depending on the application scenario, different thresholds might be used. For example, a threshold could be set using the F-score to balance TPR and FPR. Another option might be to set $FPR = 0$ while still obtaining an $TPR = 0.35$ in case of PASR1. For such setting 35% of the time an attack will be detected while not preventing normal use of the system.

4.7.1-1 Different Architectures As the ACD mechanism should be integrated in a PVA ecosystem it is important to consider resource requirements of the protection mechanism. Using this protection, voice is analysed by two ASR components instead of one. The ACD implementation may run on a dedicated device (e.g. a smart speaker or phone) or within a cloud-based back-end infrastructure. In either case, additional resource use of the ACD should be limited. Hence, it would be desirable to use a much less resource intensive ASR as protection ASR. The experiments show that this approach is feasible. For example, PocketSphinx is an ASR designed for systems with limited resources and is less complex than Kaldi which we used as main ASR.

4.7.1-2 Different Training Data In order to increase difficulty for an attacker to bypass the ACD it should not be possible for the attacker to obtain knowledge on the internal workings of the protection ASR. This can be achieved by frequently changing the configuration of the ASR by using a different training data set. Thus, the protection ASR becomes a moving target.

However, frequent re-training requires resources and such effort should be limited. The effort can be limited by reducing the training effort by reduction of the used training data. Our experiments have shown that this is a feasible approach. PASR4 uses half the training data compared to PASR3 producing comparable ACD protection results.

We used a Dell XPS 13 laptop with specifications of i7-7560U CPU, 16GB RAM and integrated Graphics 640 for training PASR3 and PASR4. The training takes 4.7 hours for PASR3 and 3.7 hours for PASR4, which means training a standard Kaldi GMM-HMM using only half of the full dataset costs 78% of the time required for training the same architecture using the full dataset.

4.7.2 Limitations and Future Work

While our work shows the principle feasibility of ACD there are limitations which we would like to address in future work.

4.7.2-1 Evaluation Commands There are two limitations regarding the adversarial and benign commands that we used.

The adversarial commands are generated based on adding perturbations to music rather than normal speech. Commands can be hidden in any audio, music or speech [24, 156], and the principle mechanism is the same. However, using music may have a different effect on the decoding performance of various ASR when compared to speech. Thus, experiments using adversarial commands built on normal speech commands should also be investigated.

The sample size of our test adversarial examples and benign commands are both 20. A larger amount of test commands would be useful for the evaluations.

4.7.2-2 Speech Generation We also used the Google online TTS service to generate the 20 benign commands instead of using a person recording these. We found that the decoding results of generated voice is better than that of a human speaker. For instance, the TTS decoding of PASR3 produces a WER of 44.52% compared to decoding a human speaker which results in 69.86%. The TTS decoding performance for MASR1 is 39.73% compared to a human speaker which is 58.9%. We used human spoken commands as the test input in our study as it is closer to the real scenario. However, it might be of interest to investigate the reason behind this discrepancy.

4.7.2-3 ASR Performance We would like to mention that the normal speech command decoding performances of our ASR are not very ideal as shown in Table 4.2, and PASR2 shows the best performance. The MASR1 and PASR3 are trained using WSJ which is a corpus of read sentences and it contains about 80 hours of data. These two models are not state-of-the-art architecture and the training dataset is not large and not comprehensive enough to include training data from various recording environment. However, PASR2 is trained using the latest architecture of Kaldi and trained with 1200 hours of audio, so it is a more advanced and more robust model. These reasons explain why the performances are so when they decode 20 normal human spoken commands recorded in a non-professional recording environment. Compared to these three, PocketSphinx (PASR1) is the earliest open-sourced GMM-HMM model of the four and it is a lightweight version of the more complex CMUsphinx ASR. It might suit better for KWS but not strong at long sentence recognition, so it is reasonable that it has the highest WER error (Latest PocketSphinx has included DNN [34], but it is not used in the work of this chapter).

4.7.2-4 Systematic Evaluation of Architecture and Training Data Our hypothesis is that the ACD provides better protection if protection and main ASR use different architectures as then both ASR rely on different features and processing mechanisms. The experiments indicate that this might be the case (as PASR1 is a GMM-HMM while MASR1 is a DNN-HMM). However, a more detailed study is required to support this hypothesis.

We have shown that the reduction of training data for the protection ASR is feasible. PASR4 uses half the training data than PASR3 while providing similar ACD performance. A more detailed study is required to investigate in detail the dependency of training data reduction and ACD performance.

4.7.2-5 ACD Attacks There is currently no study showing that it is feasible to construct an adversarial command targeting multiple ASR in parallel under either white or black-box assumption. To defeat the proposed ACD mechanism, an adversarial command must be crafted that is recognised by two ASR at the same time while still being inaudible. We would like to study this aspect in more detail to see (i) if there are any MASR/PASR combinations (in architecture and training data) that could enable an ACD attack; (ii) if such MASR/PASR combination exists if the adversarial command can be inaudible; (iii) to construct a formal proof for MASR/PASR combinations that shows an inaudible adversarial command is impossible.

4.8 Chapter Conclusion

PVAs are susceptible to adversarial attacks. ASR is a major component of PVA and neural network based ASRs are vulnerable against adversarial audio examples, which has been verified by extensive studies in recent years as shown in Section 3.2.1-B3. However, there is barely research about how to defend against these threats. In this chapter, we propose a practical defence method against one of the state-of-the-art adversarial attacks using psychoacoustic hiding for adversarial example generation. By using a different ASR working in parallel with the main one which needs to be protected, comparison between decoding results of them helps in predicting if the audio input is an adversarial example or a benign normal command. In our evaluation, we have shown our defence method is feasible and promising. An ASR that differs in architecture and/or training data is usable as a protection ASR. In next steps we would like to evaluate the impact of training data and architecture (more variety on ASR types and thus their combinations) on detection performance more comprehensively with bigger test data pool. With this larger scale evaluation, a formal proof for this defence method could be constructed.

Chapter 5

Acoustic Tagging

5.1 Acoustic Tagging for PVA Privacy Control Method

As mentioned in Chapter 1, Siri, Amazon Echo, Google Home and the like are now commonplace PVAs. They are integrated into consumer electronics or are also used as stand-alone devices. PVAs are sometimes also referred to as Smart Speakers or Voice Controllable System (VCS). PVAs continuously monitor conversations and may transport conversation elements to a cloud back end where speech is stored, processed and maybe even passed on to other services.

A user has currently little control over how her conversations are treated. Not all PVAs are owned or managed by the user, and she is normally not in control of back-end systems and has no influence over how the services exchange conversation recordings. For example, when meeting people the user can switch off her own phone-based PVA but cannot control PVAs of others.

We argue that users desire more control on how their conversations are processed by PVAs. We propose to embed additional information (referred to as *tag*) into acoustic signals which can then be interpreted by the systems to implement security and privacy requirements of involved parties.

Many methods to generate acoustic tags exist, ranging from a simple signal overlay (e.g. addition of a single tone) to a hidden acoustic watermark, which in turn are suitable for different application scenarios. For example, a simple acoustic tag can be employed by users to signal that they have given no consent to recording, processing and distribution of conversations recorded in their presence. A cooperating PVA back end looking out for such tags may then not process the recorded audio to honor the wishes of individuals. An acoustic watermark hidden within a recorded audio sample may be used by individuals to identify the origin of recorded speech at a later stage; it might give individuals an opportunity to keep track of recordings they have never agreed to. In such a scenario, cooperation of the PVA back end is not necessary.

Besides the design of a tag and its usage, there is also the question of how the acoustic tag is generated. A device is needed to generate the tagging signal; a likely candidate is a mobile phone with a suitable app. As it is not efficient to continuously transmit tag information (and the tag signal may also be perceived as noise nuisance if audible) it must be determined when to emit a tag signal. This can be solved by having a tagging device listening for the same wake words as the PVA. Finally, as multiple users may want to tag, collisions must be avoided and a tagging protocol must be established.

This chapter explores the aforementioned design space of acoustic tagging for PVAs. We consider options for tagging devices, tagging signals and application scenarios. The specific contributions of the work in this chapter are:

- *Tagging Applications:* We give a description of application scenarios in which acoustic tagging can address user privacy and security concerns.
- *Tagging Signals an Protocols:* We provide a classification of tagging options and describe protocols for embedding tags of multiple users.
- *Tagging Evaluation:* We provide an evaluation of the signal path for simple overlay tagging using Google Home Mini. We show that tagging signals in the range between 4kHz and around 7.2kHz are usable.
- *Tagging Prototype:* We describe our prototype tagging device based on Pocket-Sphinx [65] and an evaluation of the system. The prototype shows that tagging can be used to signal non-consent in public spaces.

Section 5.2 describes PVA functionality. Section 5.3 discusses tagging application scenarios. In Section 5.4 we describe different tagging options followed by a description of tagging protocols in Section 5.5. Section 5.6 provides a tagging evaluation with Google Home Mini. Section 5.7 describes our prototype device and its evaluation. Section 5.8 concludes the chapter.

5.2 Personal Voice Assistant (PVA) Operation Cycle

The operation cycle of a PVA, shown in Figure 5.1, consists of two phases: *activation phase* and *recognition phase*.

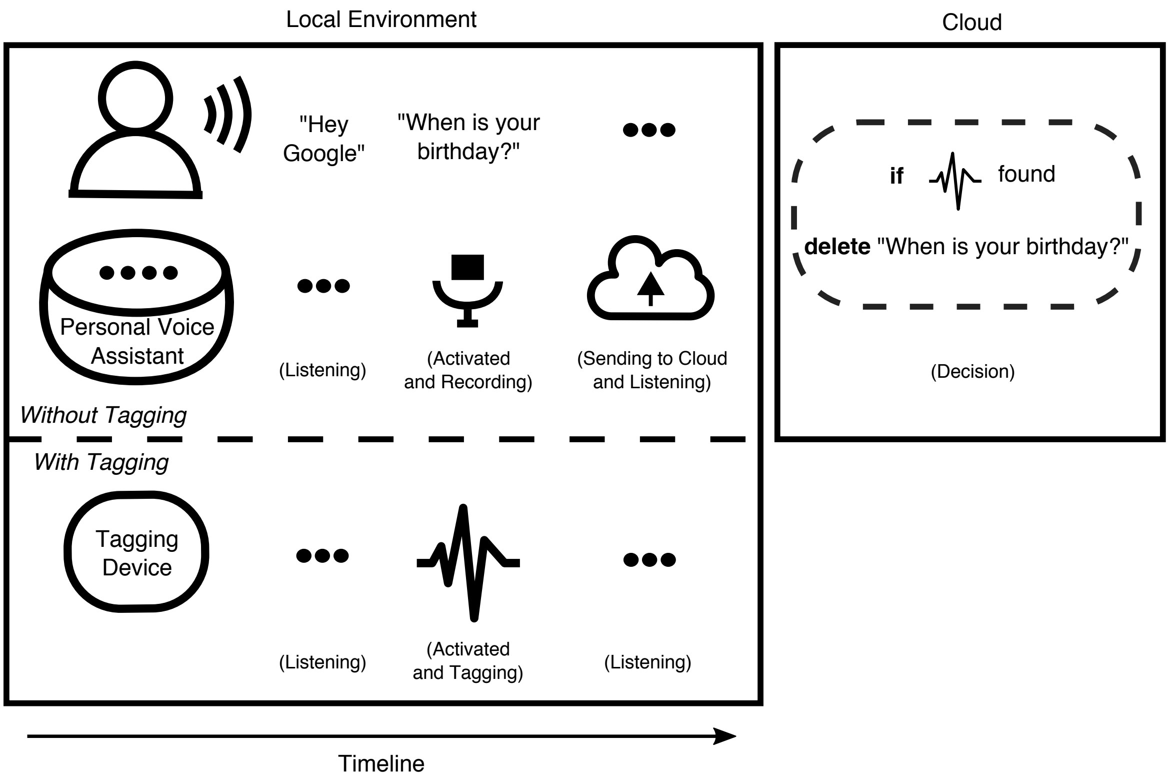


Fig. 5.1 The workflow of a personal voice assistant, without or with a tagging device.

In the activation phase the PVA waits for a user to activate speech recognition. A user may activate speech recognition by specific actions such as a button press (e.g. as used on a Sky Q remote) or by stating a specific wake word (e.g. *Alexa* in case of Amazon's Echo). In light of practicality, most systems utilize a wake word mechanism. The wake words may be speaker-dependent (trained to recognize a speaker) or speaker-independent (any user can state the wake word) [158].

On activation the PVA enters the recognition phase. In most scenarios, the PVAs streams the audio signals following the wake word to a back end for analysis. Speech recognition is carried out in the back end for several reasons: to keep computation-intensive tasks away from the device; to enable flexibility in updating speech recognition algorithms; to enable flexibility in PVA services. The back end may take actions in response to the processing result. A response might be sent to the local device or another action may be triggered.

The captured audio streams are stored by PVA providers, and the storage duration and the specific usage of the data is not clearly articulated [107, 157, 62].

5.3 Application Scenarios

Acoustic tagging in the context of PVAs can be used for a number of security and privacy related application scenarios. The set of scenarios provided here covers a broad range of possible scenarios but is not exhaustive. Acoustic tags may carry rich semantic information.

5.3.1 Signalling Recording Consent

People generally object to conversations being recorded without their given consent. Hence, laws exist in most countries defining (very differently) how recording consent has to be given. For example, Germany is a two-party consent state, which means that (phone call) recording without the consent of participants is a criminal offense. In the U.K., the Data Protection Act (DPA) of 1998 assumes tacit consent and individuals must be only given the option to opt out from recordings. Recent European Union (EU) General Data Protection Regulation (GDPR) legislation, superseding the aforementioned situation in U.K. and Germany, requires consent of all parties for a specific purpose. In the context of PVAs it is a question of how participants can signal consent or lack thereof. It is not always evident to people that there are PVAs nearby that record conversations. In addition, PVAs do not have an interface to provide consent information.

Acoustic signal tagging as we propose provides a technical solution to implement PVA compliance with legislation. An acoustic tag will be emitted by users who give no recording consent. Any PVA system detecting a tag could then refrain from processing or even recording a conversation. PVAs need not introduce an additional interface to interact with users, and all existing systems can use this mechanism by simply augmenting their audio processing capabilities. Tag signals can be emitted by simple user devices such as a smart phone. Tags do not have to be transmitted such that they interfere with users and their conversations. Tag transmission can be timed such that they are only transmitted when required; a signal strength and frequency will be chosen to minimize impact. This solution obviously requires cooperating PVAs that react to detected tag signals.

We describe and analyze an implementation of a tagging system for consent signalling in Sections 5.6 and 5.7.

5.3.2 Recording Identification

PVAs record conversations which are stored on back-end systems. Recorded conversations are potentially stored for long periods of time (years). Stored conversations can be accessed by anyone that has access to the back end. Usually, access to recordings is limited to PVA owners. However, it has to be noted that PVA owners and conversation participants may be different groups.

It is reasonable to assume that conversations are recorded (by accident or on purpose) without consent by nearby PVAs. Such recordings may later be used and it might be desirable to identify the context (e.g. location, time, participants) of the conversation.

An acoustic tag can be used to add the required meta information to conversations. The tag might be added in a way that it is hidden within the recorded audio signal in order to prevent detection and/or removal of the signal. We discuss tagging options and details in Section 5.4.

5.3.3 Data Trading

PVAs store conversation recordings on back-end systems. This data is an asset and the service providers employ it to improve their offerings. For example, stored conversation recordings are used to improve speech recognition algorithms. Significant improvements can be made by training speech recognition algorithms using samples from a large number of individuals.

PVA service providers may decide to trade conversation samples, for example for algorithm training purposes. We are not aware that any service providers currently engage in such data exchange; however, common PVA license agreements would allow the providers to engage in such activities [25].

A provider may tag samples in order to control further distribution or to simply mark the sample source.

5.4 Tagging Options

There are a number of options for embedding additional information in audio signals processed by PVAs. Generally, the additional information must be embedded within the frequency spectrum that is supported by the PVA microphone hardware, the PVA processing software and the PVA back end. An investigation of the usable spectrum for a typical device is provided in Section 5.6. Within the usable frequency spectrum, additional information can be embedded in different ways enabling a variety of application scenarios.

The amount of information that can be included using a tag depends on how obvious (audible) the tag can be (frequency range, power, encoding mechanism) and how much noise the PVA processing environment will add.

We identified four classes of tags, differing in their suitability for scenarios and implementation complexity:

Audible Tag A tag is embedded and its presence is clearly audible, e.g. in the form of audible noise. People will notice that noise during their conversation, making it obvious to everyone that something has happened. The information might be placed in a frequency space that is normally not occupied by voice. This simplifies the separation of voice and tag. It will be clear to anyone listening to the recorded sound that a tag is embedded; a spectrum analyzer will also clearly reveal the tag. As the tag can be clearly identified it can also later be removed.

An audible tag can be generated easily. A speaker can be used to generate the tag, which will be overlaid on a monitored conversation.

Unnoticeable Tag The tag is added to the audio signal such that its presence is not noticeable to a human. For example, the tag signal power might be small compared to the present voice signal power or the combination of power/frequency of the tag signal in relation to the power/frequency of the voice signal is such that users do not notice the tag. Embedding of the tag information will not disrupt users. Listening to the recorded audio will not reveal a tag. However, investigation of the signal using a spectrum analyzer may still reveal the tag. In addition, it would also be possible to remove later a tag from a recording.

An unnoticeable tag may be included using spread spectrum techniques, where narrowband tag information is transmitted over a large bandwidth, such that the signal energy added at each frequency leads to a non-audible change.

More challenging techniques may analyze the audio stream on the fly and then add information selectively which do not lead to noticeable audio changes. For example, properties of the human hearing can be exploited to place unnoticeable information. Audio compression algorithms such as MP3 [118] use similar mechanisms to decide which data to remove from a signal. In the same way such insight can be used to add information to a signal.

Inaudible Tag This approach is similar to the unnoticeable tag. The tag is added to the audio signal such that it cannot be perceived by a human. For example, the tag might be placed in a frequency range above 22kHz. However, when analyzing the signal in a spectrum analyzer, the additional tag signal will clearly be visible and could therefore also be removed later.

An inaudible tag is useful for similar applications as the unnoticeable tag. However, as the information does not have to be woven into the conversation, implementation is relatively straight forward. In particular, recovery of the signal is simplified, a simple filter can be used to extract the tag signal. Whilst this approach is preferable to the aforementioned unnoticeable tag, limitations on the usable frequency space may prevent this method in the context of specific PVAs.

Hidden Tag The tag is added to the audio signal such that it cannot be perceived by the user. In addition, it cannot be determined by other tools (e.g. spectrum analyzer, frequency analysis) that a tag is embedded in the signal. The only way to identify a present tag is to compare the original tag-free signal with the tagged one. In this case the tag could be considered an acoustic watermark.

A hidden tag has similar processing requirements as the unnoticeable tag. However, in addition the data has to be placed in such a way that it cannot be recovered by analyzing the recording. Using a cryptographic key, data has to be integrated with the conversation. In this case the tag should also be robust against transformations (e.g. downsampling, transcoding).

For example, a Spectrum Audio Watermarking (SSW) can be used where the tag information is distributed over a large frequency spectrum. A pseudonoise (PN) sequence is used to spread the tag information over the frequency space; to recover the tag from the signal the PN sequence must be known. SSW is difficult to remove; a wideband noise signal of high amplitude is required which is very noticeable.

5.5 Tagging Protocols

In many situations more than one tagging device might be present and a protocol is necessary to ensure that tag signals are added orderly such that tag recovery is possible.

The tagging of an acoustic signal must be timed as it is not feasible to emit a continuous tagging signal. A continuous signal might be perceived as noise nuisance and is resource inefficient; e.g. it will drain the battery of the tagging device. A tagging device will become active when needed, for example, when detecting a wake word. Thus, all tagging devices are likely to emit the tagging signal at the same time leading to collisions. Collisions may prevent recovery of the tag signals.

Devices may separate their tag signals in frequency, time or code domain to prevent collision. Alternatively some devices may refrain from tagging to ensure that only one device embeds a clear tag. As each device must determine frequency/time/code, coordination among tagging devices is necessary. This can be achieved by using an out-of-band control channel among devices (e.g. a local wireless link) or by using in-band methods as used in Medium Access Control (MAC) protocols for wireless communications. For example, tagging devices

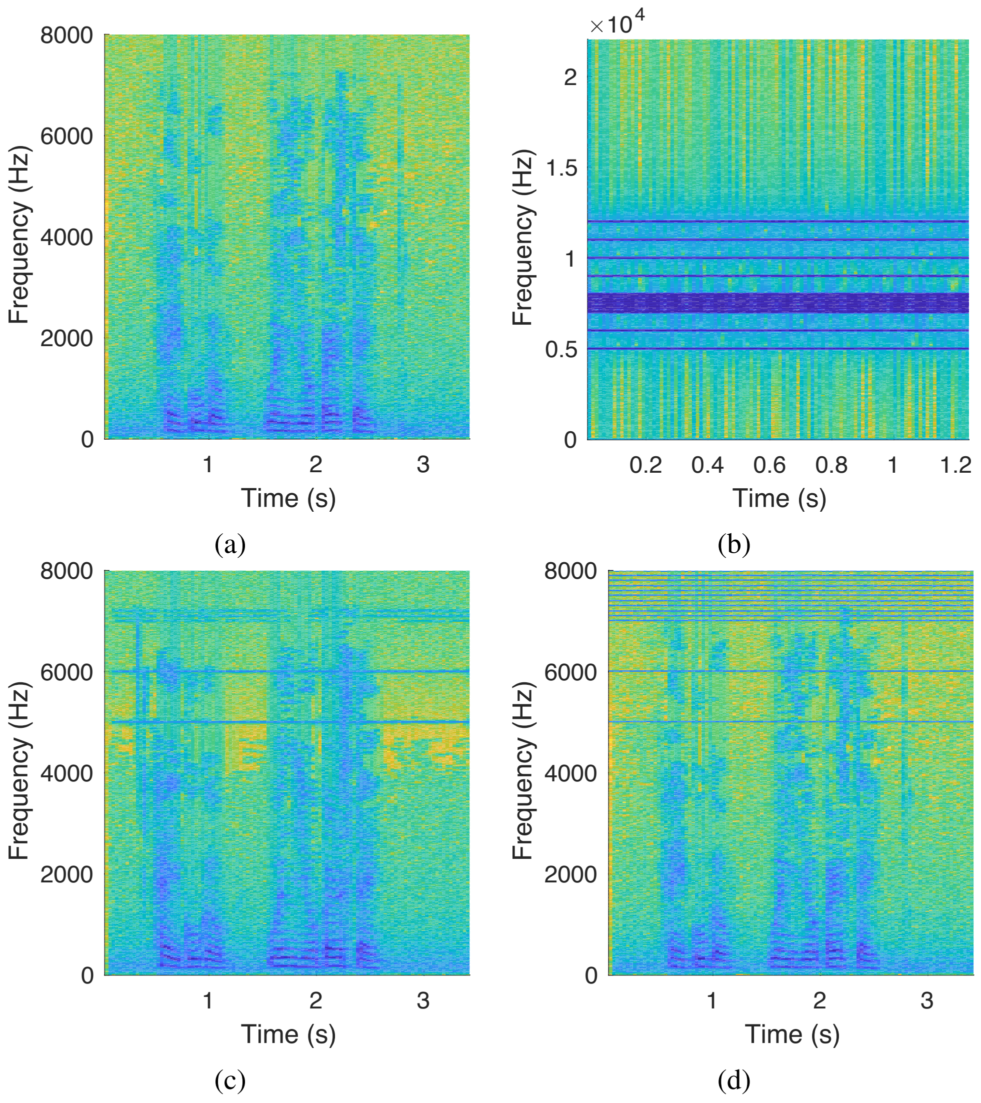


Fig. 5.2 The spectrograms of the audio signals, including (a) the original audio signal of "*Hey Google, when is your birthday*", (b) the man-made multi-tone tag signal, (c) the recording of the original audio signal together with the tag signal downloaded from the Google server, and (d) the same signal considering no loss and distortion during the whole process of propagation, recording, uploading and compression

might listen first if a tagging signal is already present, if so, a free frequency/code is chosen or the device delays tagging.

In-band coordination requires that tagging devices are aware of other tagging signals. Obviously, the tagging option used must allow devices to observe this process. When using hidden tags, in-band coordination might therefore not be feasible unless all devices are able to recover the hidden information.

The required coordination might also depend on the application scenario. If it is only necessary to determine a tag presence but decoding of information is not necessary, collisions are not an issue. For example, multiple devices could express that they do not consent to a recording with colliding signals; the PVA back end only needs to determine signal presence but does not necessarily have to decode information carried in the tag.

5.6 Tagging Analysis

We use a common PVA, the Google Home Mini, to evaluate tagging performance. The aim is to determine the usable tagging frequency range and to evaluate tag signal distortion. PVA microphone hardware, audio processing and compression on the PVA and the back end will limit the usable frequency range and will distort a tagging signal.

5.6.1 Recording Constraints

Before we investigate tagging performance we evaluate the general audio recording capabilities of the Google Home Mini. We speak the phrase "*Hey Google, when is your birthday?*". Then we use the developer mode of the Google Chrome browser to download the audio recording from Google's Myactivity website. All voice commands are recorded by the back end and can be accessed using the aforementioned method.

The recording obtained from the back end is an MP3 encoded file. However, it is not visible to us at which point this MP3 compression is carried out. It is also not clear if the audio recording is transcoded along its processing path. The Google Home Mini may transmit to the back end using another audio encoding. Also, the back end may internally use a different format. The conversion into MP3 may happen only on the download path to the user. However, it is reasonable to assume that the back end also uses MP3 as the internal storage format of recordings.

We use the software Audacity to evaluate the MP3 recording and find it to be a stereo, 16kHz MP3 format. The audio signal passes through a low-pass filter which attenuates frequency elements higher than 8 kHz. Due to practical non-ideal low-pass filters, the attenuation will also affect frequencies just below 8 kHz.

5.6.2 Audible Tag Constraints

We consider an audible tag as described in Section 5.4. Such a tag is audible when it is added, and it should not occupy the frequency range that the spectra of voice signals mainly reside in (up to 3.4 kHz). Thus, tag extraction can be performed simply by a band pass.

Figure 5.2a shows the spectrogram of the spoken command. The wake word "*Hey Google*" is clearly visible from 0.5s to 1.2s, and the command "*When is your birthday?*" is visible from 1.5s to 2.6s. The spectrogram indicates that most of the speech energy resides, as expected, below 3kHz. As a voice signal has its main frequency components below 4kHz, the tag signal should reside above this frequency (see [88]).

5.6.3 Test Tag

We create a simple audible test tag to evaluate tagging performance. A tag should reside between 4kHz and 8kHz to fit with both recording and tag constraints. It is our aim to see how a tag in this frequency range is affected by the recording process.

We create a frequency vector with the value of the elements set to zero except bins representing 5kHz to 7kHz (with a 1kHz interval), 7.1kHz to 8kHz (with 100Hz interval), and 9kHz to 12kHz (with 1kHz interval). Then we use Inverse Fast Fourier transform (IFFT) to generate the time domain tag signal. Figure 5.2b shows the spectrogram of this tag signal. Note that the upper limit of the frequency axis is around 22kHz as the sampling frequency of the signal is set to 44.1kHz. We use this signal shape to clearly see how the tag signal is attenuated close to the 8kHz boundary defined by the recording constraints.

5.6.4 Tagging Performance

We use a long tag signal (about 5 seconds) for testing. We start emitting the test signal from a speaker and then activate the Google Home Mini with the wake word "*Hey Google*" followed by the question "*When is your birthday?*".

Figure 5.2a shows the spectogram of the spoken command. Figure 5.2b shows the tag signal. Figure 5.2c shows the spectrogram of the recording retrieved from the back-end system. Figure 5.2d shows the audio signal with the overlaid tag signal below 8kHz for comparison.

It can be seen that the tag information above 7.2kHz is lost. The sampling frequency of the audio encoding is 16kHz, which means ideally all of the audio contents below 8kHz should be retained. However, only the audio contents below 7.2kHz remains, and we assume this may result from the unavoidable imperfection of the filter design.

Comparing the result in Figure 5.2c and the ideal condition in Figure 5.2d, it can also be seen that tag lines have widened due to signal recording and processing steps. These distortions would have to be considered within the tag design to ensure correct information retrieval.

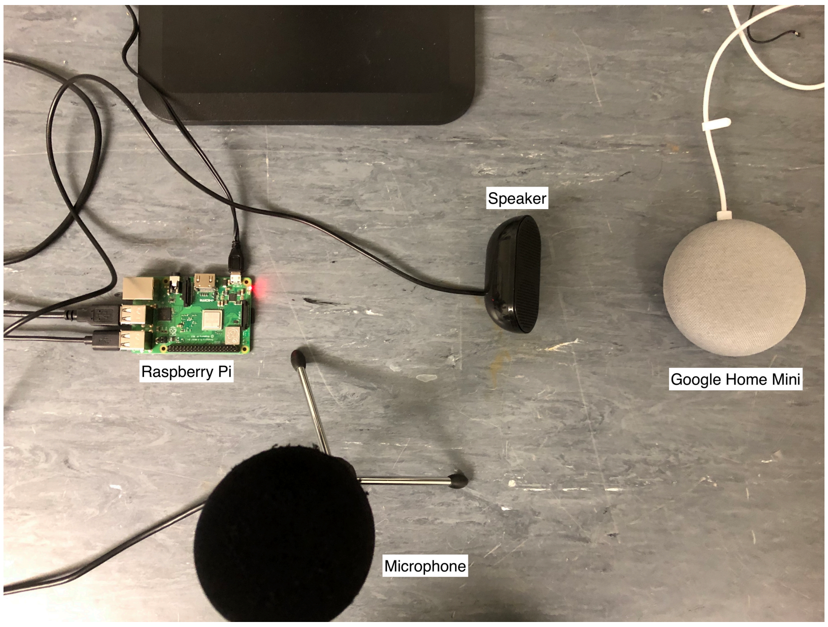


Fig. 5.3 Experiment Setup

5.7 A Prototype Tagging System

An audible tag as evaluated earlier should not be present continuously, since otherwise it would be perceived as noise nuisance. It is necessary to emit the tag signal only briefly and when required. In this section we describe and evaluate a tagging device which we design to perform this task. This is a proof-of-concept prototype, demonstrating the feasibility of building a practical tag device.

5.7.1 Tagging Device

As the hardware platform we select a Raspberry Pi 3 Model B+ with a simple USB microphone and a commodity speaker. We chose this platform as it provides prototyping flexibility while it is comparable in functionality to other platforms such as mobile phones that might be chosen to implement a tagging device.

The tagging device is required to emit the tag signal only when required. We use the same wake word that a potentially present PVA uses to tag transmission. We implement the wake word detection using PocketSphinx [65]. PocketSphinx is an optimization of CMU's SPHINX (an open source LVCSR system) for resource-limited embedded systems [31, 33]. PocketSphinx uses the more traditional GMM-HMM approach for wake word detection while current commercial PVAs such as Amazon Echo or Google Home use proprietary algorithms (For example, DNN-HMM in case of Amazon). As the tagging device uses a

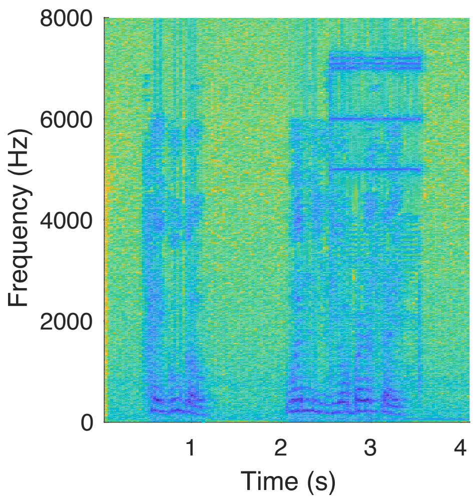


Fig. 5.4 The spectrogram of the downloaded signal resulting from the prototype tagging system

different algorithm than the PVA, it is possible that one device recognizes a key word while the other does not. However, in our experiments we did not observe this case of differing wake word detection results.

A simple Python script based on [146] (Appendix B) was used to detect the wake word and transmit a predefined audible tag. The Matlab script for generating the audible tag is included in Appendix C.

5.7.2 Evaluation

To evaluate the tagging device we use the experiment setup shown in Fig 5.3. A Google Home Mini is used as the PVA and the tagging device with speaker and microphone are placed next to it.

We use the tag signal as described in Section 5.6. The tag signal duration is set to one second. We then speak the sentence "*Hey Google, when is your birthday?*" to test the system. The wake word is recognized by the PVA and as well as the tagging device which emits the tag signal. Thereafter we use Google's Myactivity website to download the recording.

Figure 5.4 is the spectrogram of the audio file representing the whole experiment. "*Hey Google*" ends at around 1.2s after the start of the recording. A series of horizontal lines representing the tag signal which starts at 2.4s and lasts for 1s. Figure 5.4 suggests that it takes around 1.4s for the reactive tagging device to successfully recognize the wake word "*Hey Google*" and to start transmitting the tag signal.

Figure 5.4 also reveals how Google Home Mini handles its wake word recognition. The recording stored in the back end begins *before* the wake word is spoken. We can assume that the Google Home Mini continuously records sound, regardless of the presence of the wake word. Conversation fragments spoken before a keyword may be recorded by the back end.

5.7.3 Discussions

Our prototype demonstrates the feasibility of the tagging approach. For example, this approach can now be used to signal recording dissent (see application example in Section 5.3). A user who does not wish to be recorded can activate the tagging device. The tag will be embedded when the PVA is triggered and the back end may discard the recording on tag detection.

The tagging prototype is relatively slow and the tag signal is emitted after 1.4s. Software optimization would significantly reduce this time and allow us to place the tag signal between the wake word "*Hey Google*" and the command "*When is your birthday?*". This would provide a better user experience as the audible tag sound would fall in the quiet gap instead of overlapping with the command.

We did not plan for multiple tagging devices in this scenario; in this case a tagging protocol as sketched in Section 5.5 would be required. We also did not evaluate more complex tagging options as outlined in Section 5.4.

5.8 Chapter Conclusion

People generally object to conversations being recorded without consent and given the widespread use of PVAs it is necessary to provide better recording control than currently available. In this chapter we have shown that acoustic tagging is a viable option to signal to PVAs and their back-end systems how recordings should be handled. We have explored the design space of acoustic tagging in the PVA context and described the implementation and evaluation of an initial prototype. In next steps we aim to develop a full system, and to explore its performance and usability in a realistic setting.

Chapter 6

PVA Jamming

6.1 Acoustic Reactive Jamming to Disable PVA Activation

As mentioned in Chapter 1, PVAs are deployed pervasively in our life. They appear in various forms including smart speakers, voice assistants in smartphones, part of smart TVs and set-top boxes and even the interaction interface in modern cars. Thus, it is very likely that we are always at least in range of one PVA.

Using microphones, PVAs are monitoring the acoustic channel for a specific spoken word. Once this *wake word* has been detected the PVA records the immediately following audio signal which is then transported to a back-end system for further analysis. The back-end system then analyses the audio signal and extracts spoken user commands.

Out of security and privacy concerns, people would like to be in control of surrounding PVAs, for example, to disable the ability to issue commands or to prevent recording of conversations. When a person is in control of the PVA she can simply turn off the device. However, in environments such as public spaces this is impossible.

In this chapter we investigate *acoustic reactive jamming* as a method of disabling PVAs. A Protection Jamming Device (PJD) is used which emits an acoustic jamming signal to prevent a PVA from analyzing audio signals. However, instead of continuously jamming the channel, which would be perceived as continuous noise nuisance, jamming is applied reactively at specific moments. Specifically, the PJD recognizes the spoken wake word and applies a jamming signal to it. Thus, the PVA fails to recognize the wake word and it does not record the following audio signal.

The PJD will also be useful for other protection scenarios which are beyond the scope of this chapter. For example, recent work has shown that PVA can be triggered by attackers using inaudible voice commands (see Zhang et al. [158]) which could be recognized and jammed. Our focus is on designing a protection device, empowering people to control PVAs tapping into their conversations. It also has to be noted that the described mechanism can be exploited by a nefarious actor to disable a PVA.

Signal jamming has been previously employed as effective protection mechanism. For example, reactive jamming has been used to protect wireless communication networks [87, 20]. The packet header containing source and destination addresses is evaluated and, if required, a jamming signal is applied to the remainder of the packet, preventing packet reception. Our work transfers this reactive jamming method to the acoustic domain. A recent work by Roy et al. [111] demonstrates inaudible jamming in the acoustic domain. Non-linearity of the microphone shifts the white noise jamming signal in the ultrasound frequency range to the audible range. This work shows how to jam an acoustic system without people noticing the signal directly. However, this existing work is not tailored to the PVA context and uses continuous jamming signals which are inefficient and also might constitute a potential health risk. To the best of our knowledge, the work presented in this chapter is the first investigating reactive jamming targeting PVAs.

Reactive jamming requires two elements: *wake word detection* and *jamming*. The protection device must be able to detect the wake word faster than the PVA. The protection device can then apply the jamming signal to the later section of the wake word. If the overlap is sufficient, wake word detection by the PVA is prevented. The jamming signal must be effective and people should not consider it as noise nuisance. Thus, the signal should only be applied when needed (low false positive rate), be not too loud and somewhat pleasant to listen to.

This chapter provides two specific contributions. First, we quantify the impact of jamming signal and wake word overlap on the jamming success. We use four different wake words used by the popular Amazon Echo PVA and four different jamming signals for our study. Our evaluation shows that PVA wake words can be jammed with a 100% success rate in most cases when jamming signal (Additive White Gaussian Noise (AWGN)) and wake word overlap more than 60%. Second, We quantify the jamming false positive rate in dependence of the required overlap. The protection device has to recognize the keyword faster than the PVA leading to false positives; jamming is applied when not required. Our evaluation here shows that false positive rates are negligible.

6.2 PVA Activation Mechanism

Many companies nowadays have their own signature Personal Voice Assistant (PVA) software and hardware. For instance, Apple appliances integrate Siri, Amazon provides Alexa, Google integrates the Google voice assistant, and Samsung gadgets work with Bixby.

6.2.1 System Overview

As described previously in Section 5.2 of last chapter, there are two operation phases of a PVA: *activation phase* and *recognition phase*. In the activation phase, a user needs to activate the PVA to initiate speech recognition. Typically, a user presses a specific button (e.g. as used on a Sky Q remote) or simply says a wake word (e.g. *Alexa* in case of Amazon's Echo). Most systems implement a wake word as this improves usability. The wake words may be speaker-dependent (e.g. *Hey Siri*) or speaker-independent (e.g. *Alexa*) [158]. Wake word recognition is continuously active on PVA devices. Once the key word is detected, the PVA enters the recognition phase. For most systems the audio signal following the wake word is streamed to a back-end cloud service for analysis. The cloud service is used to analyze the captured audio signal to extract user commands. It also may store captured audio signals.

Central part of a PVA device is the wake word recognition implementation. Corporations such as Apple or Microsoft do not provide details of their implementations; however, open source toolkits such as AlexaPi based on PocketSphinx developed by CMU [65, 33] are available. All major wake word recognition implementations have similar performance from users' perception and are based on only several major approach options.

ASR is employed for wake word detection in the activation phase on the device and as well for command recognition on the back-end during the recognition phase. ASR for wake word recognition can be less complex compared to the one used in the back-end as only one or a few words must be recognized. Wake word recognition is a specific application of ASR referred to as KWS in the literature. Often, wake word recognition is implemented in dedicated hardware to improve energy consumption of battery powered devices. Wake words can be speaker-dependent and in this case the legitimate user has to train the PVA.

6.2.2 Automatic Speech Recognition and Keyword Spotting

We briefly provide a refresh about ASR and some necessary insights on KWS in this section. The process of ASR begins with separation of the audio input based on pauses. Each identified speech block is called an utterance which may be a word or a non-linguistic sound (e.g. cough, um, breath) [33]. Each utterance is then split into segments. A feature vector is extracted to represent each unit.

Models need to be constructed to predict what language elements these units represent. GMMs are the commonly used acoustic models. However, recently these have been replaced by models trained by DNN as they are more robust. These models can tolerate better environmental and hardware specific variations [147, 100, 63].

Further processing is necessary to deal with temporal variability [63]. HMM are normally applied to ASR and KWS. Some recent KWS solutions can also work without HMM [147, 26].

Other techniques using CNNs or RNNs/Long Short-Term Memorys (LSTMs) instead of the combination of DNN and HMM for KWS also exist [147].

Apart from these techniques, some KWS systems use LVCSR to decode audio and generate lattices [26, 147], then they can be used for indexing and keyword searching. These systems focus on large audio database applications rather than audio streaming applications, which is outside the scope of the work in this chapter.

6.2.3 PocketSphinx

AlexaPi based on PocketSphinx performs wake word recognition using the GMM-HMM approach for KWS described earlier [65]. PocketSphinx [65] is the optimized version of CMU's SPHINX (an open source LVCSR system) for resource limited embedded systems. PocketSphinx provides wake word selection and detection threshold tuning functions.

Products such as Amazon Echo/Echo Dot or Google Home use proprietary algorithms (often in combination with specialist hardware) based on more recent techniques discussed in the previous section to perform KWS. For example, Amazon products mainly use DNN-HMM solutions, Google and other vendors use solutions such as a single DNN followed by a posterior handling method.

Although PocketSphinx doesn't apply state-of-the-art modeling technique, it is still a reliable KWS solution [48, 77]. Because it is an open source speech recognition system it is often used instead of proprietary speech recognition toolkits. In particular, small companies or independent developers may make use of PocketSphinx.

In our evaluation we use AlexaPi based on PocketSphinx, which means at this stage we only focus on jamming wake word recognition based on GMM-HMM. We treat the wake word recognition as a black box, meaning that our jamming signals are not designed to the specifics of the wake word recognition algorithm.

6.2.4 Security Considerations

It is probable that the voice data sent to the back-end and also extracted commands are stored. Companies such as Amazon and Google can use the data in many ways. The command history extracted from audio streams can be used for marketing purposes (learning user behavior). The audio streams can be used as training data for the ASR system. Companies are not very clear in regards to how captured data is processed and used. A user must therefore assume that recorded audio streams or results from processing these remain in the back-end system.

Users might trigger the PVA unintentionally by using the wake word (e.g. by talking about your friend Alexa). In this case a conversation that is not a command to the PVA is recorded. Users might want to be in control, and ensure that wake word recognition is disabled. If users are not device owners this is difficult to achieve (e.g. in public spaces with PVA systems). Wake word jamming as described in this chapter can help in such situation.

Another threat is from adversaries who are familiar with the working mechanism of PVA and are skilled at producing human unnoticeable or human inaudible sound attacks. Some research [158, 112, 156] has shown how to send human inaudible or unnoticeable but machine understandable voice commands to attack PVAs or ASR software. Our work in this chapter sheds light on how to protect against such covert attacks from another angle.

An attacker might also use the jamming technique we explore in this chapter to prevent the use of PVA systems. A jamming device using a human inaudible reactive jamming signal will be very hard to detect and will prevent the use of any PVA in the vicinity. An attacker might further use jamming to hijack the system. The attacker can prevent the PVA to execute user commands and instead perform an action. The user has the impression of interacting with the system while another device is acting on its behalf.

6.3 PVA Jamming

It is our aim to design a reactive jamming framework which can be used to intentionally disable PVAs. We aim to interrupt the wake word detection in the activation phase so that the PVA never reaches the recognition phase. Jamming is applied selectively, preventing continuous jamming signals that can be perceived as noise nuisance or even have health implications. The jamming framework can be instantiated to become a PJD.

6.3.1 Jamming Approach

The overall design of the jamming approach is shown in Figure 6.1:

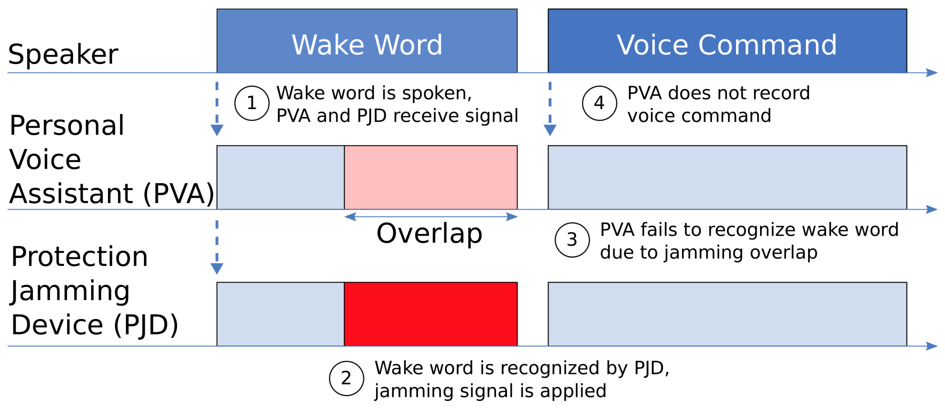

Speaker
Wake Word
Voice Command
Personal Voice Assistant (PVA)
1 Wake word is spoken, PVA and PJD receive signal
4 PVA does not record voice command
Overlap
3 PVA fails to recognize wake word due to jamming overlap
Protection Jamming Device (PJD)
2 Wake word is recognized by PJD, jamming signal is applied

Fig. 6.1 Wake word jamming framework. (1) The wake word is received by PVA and PJD. (2) The PJD detects the wake word faster than the PVA and applies jamming. (3) The PVA fails to recognise the wake word. (4) The PVA does not record the voice command.

1. PVA and PJD are continuously observing the acoustic channel. Both devices listen to the channel and try to determine if the wake word is spoken.

2. The PJD is set to detect the wake word earlier than the PVA by only looking for the first part of it. Upon detection, the PJD applies the jamming signal.

3. The jamming signal now overlaps with the remainder of the wake word currently still analyzed by the PVA. The jamming signal prevents recognition of the wake word by the PVA.

4. The following voice command is not recorded by the PVA as it does not transit from activation to recognition phase.

The outlined approach is possible as the PJD is set to detect the wake word much faster than the PVA. Thus, enough time is available for the PJD to apply an effective jamming signal. The time duration in which jamming signal coincides with the later part of the wake word is referred to as *overlap*.

6.3.2 Design Challenges

A PVA's wake word recognition is designed to be very accurate to prevent accidental triggering. We exploit this fact for the design of the PJD and deliberately sacrifice accuracy for detection speed. The PJD is set to react to the start of the wake word and triggers jamming when there is a match. Obviously, the PJD will be less accurate in wake word recognition than the PVA and false jamming will be the result. It is the aim to jam as effectively as possible which means that the overlap should be maximized. However, the overlap should be minimized to prevent unnecessary jamming. An overlap should be found that balances these optimization goals.

Reliable jamming requires a jamming signal that interferes with recognition of the target signal as much as possible. Intuitively, the jamming signal should be as strong as possible and it should also overlap with the target signal as much as possible. However, the design of the interfering signal is also important. The signal should not be perceived as noise nuisance and, thus, the most effective jamming signals might not be usable. Furthermore, the signal shape could be tailored to the ASR algorithms employed. This approach might result in a very effective jamming signal but will only be effective for the specific ASR system. Finally, the jamming signal could take the signal shape of the keyword into account to effectively jam its specific properties.

6.3.3 Jamming Evaluation Framework

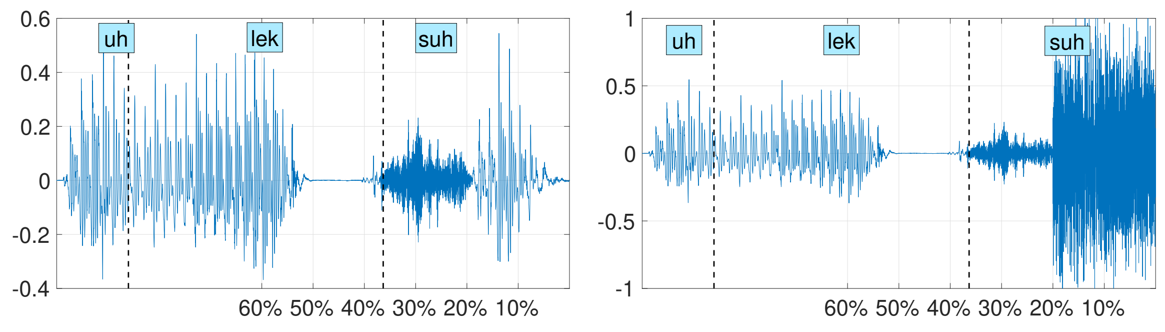
In our evaluation, described in the next sections, we explore and evaluate the aforementioned design space. We use the following evaluation framework.

We used an open source software called AlexaPi [9] running on a Raspberry Pi 3 Model B. The AlexaPi uses an open source ASR system called PocketSphinx developed by CMU [33] to detect wake words. The system uses the Amazon cloud service as back-end system for following speech analysis. The system is similar to commercial Amazon products such as the Amazon Echo; the difference is the implementation of the wake word recognition. The Raspberry Pi together with the AlexaPi and Amazon cloud service is the PVA system used in our experiment.

As we aim to evaluate systematically jamming performance we create the audio input signal for the PVA system artificially. This input signal contains the wake word and, with a set overlap, the jamming signal. The generated combination of wake word and jamming signal is directly fed into the AlexaPi via a virtual microphone input. By bypassing speakers, microphones and other system elements we can study accurately the impact of jamming signal, overlap and Signal-to-noise Ratio (SNR) without system related influences.

The wake word voice for the audio input signal is generated using the open source TTS software Festival [47]. This method provides us with a standardized signal that can be reproduced[1]. The wake word is then mixed with the chosen jamming signal using Matlab.

[1]The exception was the word *Computer* for which we used a human voice recording; the software was unable to synthesize the correct pronunciation of the *C*.



(a) Alexa (b) Alexa with AWGN

Fig. 6.2 The audio signal of the wake word *Alexa* without and with 20% overlap of the AWGN jamming signal.

6.4 Jamming Evaluation

We aim to quantify the impact of jamming signal and wake word overlap on the jamming success. We evaluate the effectiveness of audible jamming signals first; thereafter we investigate the practicality of inaudible jamming. Wake word variants, noise signal types, SNR and overlap are the four parameters varied in the experiments.

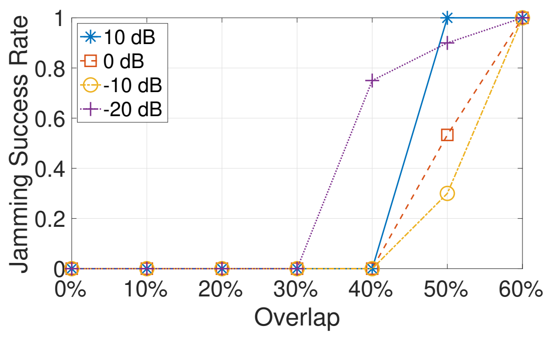
We use the wake words *Alexa*, *Amazon*, *Echo* and *Computer* because these are wake word options for Amazon Echo products.

In the first set of experiments, we chose AWGN, a *Ding*, and a short audio recording of ambient noise in a *Cafe* as noise signals (Audible Jamming). AWGN is known to have good interference properties while it is not a pleasant noise for users; the *Ding* and *Cafe* noise is less intrusive but has potentially less jamming capabilities. SNR levels of 10dB, 0dB, -10dB and -20dB are used. An overlap from 0% to 60% is selected.

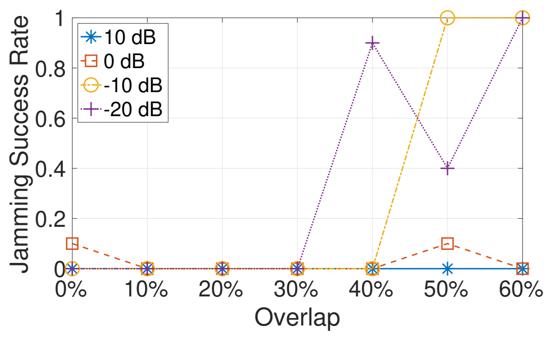
In the second set of experiments, we use inaudible jamming signals. An AWGN signal is created and then a high pass filter is used to filter out components below 20kHz. In practice, the signal is still audible as perfect filtering is not achievable (The signal is limited in the time domain). When the SNR is 10dB, 0dB and -10dB, the signal is barely noticeable. However, when the SNR reaches -20dB, the noise is obvious. Thus, a second noise signal with a band pass between 22kHz and 24kHz is used which is less audible as it has less frequency leakage in the lower frequency range. Again, an overlap from 0% to 60% is selected.

6.4.1 Audible Jamming

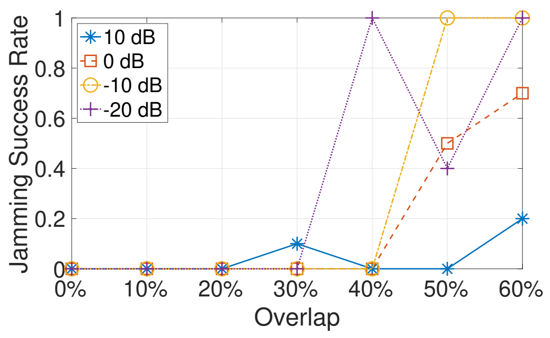
Alexa is used as wake word. The audio signal in time domain is shown in Figure 6.2a which also depicts the range for each syllable in the wake word. Figure 6.2b shows the same audio signal with added AWGN noise overlapping 20%.



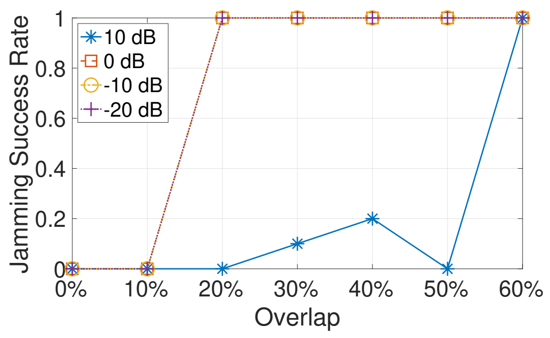
(a) Alexa - White Noise



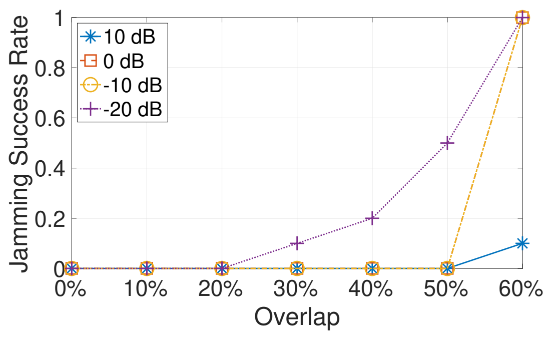
(b) Alexa - Cafe Noise



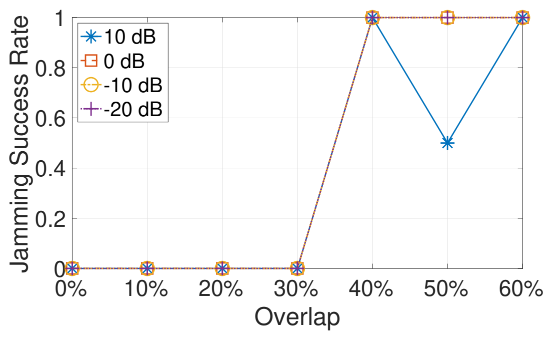
(c) Alexa - Ding Noise



(d) Amazon - White Noise



(e) Computer - White Noise

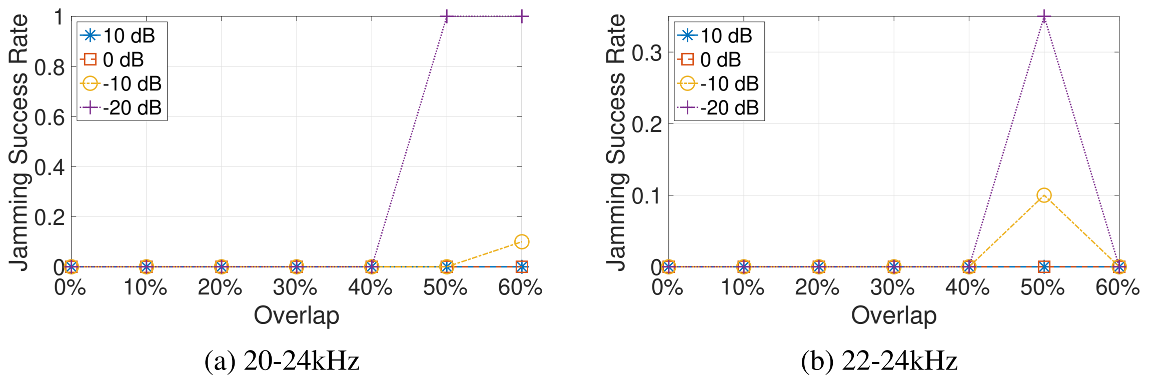


(f) Echo - White Noise

Fig. 6.3 Average jamming success rates when using audible jamming signals. The AWGN (White Noise) is the most successful jamming signal. The required jamming overlap is wake word dependent. An overlap of more than 60% achieves a 100% jamming success for all wake words.

Figure 6.3a shows the result of jamming *Alexa* with AWGN. The x-axis represents the overlap; how much of the signal was jammed, counting from the end of the wake word. The y-axis indicates the jamming success rate. For each SNR a different curve is included. Each data point is created by executing the experiment 10 times.

Figure 6.3a shows, as one would intuitively expect, that a larger overlap results in a better jamming success. Also, as expected, the strongest jamming signal with an SNR of -20dB is the most effective (with 40% overlap this signal can jam with a success rate of 75%).



(a) 20-24kHz (b) 22-24kHz

Fig. 6.4 Average jamming success rates when using inaudible jamming signals. Inaudible jamming of wake words is feasible.

However, when reaching a 50% overlap there is no consistent correlation between SNR and jamming success. For example, the 10dB signal can jam successfully with an overlap of 50% while signals with an SNR of 0dB and -10dB are less successful (53% and 30%). We suspect that this behavior is due to the specific structure of the wake word; the *k* sound in *Alexa* is similar to the interference signal used for jamming which sits at 50%. When the jamming length reaches 60%, jamming with all four SNR levels is 100% successful.

Results of jamming Alexa with a Cafe noise are shown in Figure 6.3b. As the jamming length increases, jamming success rate is negligible for SNR of 10dB and 0dB. For SNR of -10dB and -20dB, the jamming rate is higher. Jamming *Alexa* with a Ding noise (see Figure 6.3c) has similar results except higher jamming success rate for 0dB SNR.

From these experiments it is clear the AWGN is the best jamming signal. It is also shown that with an overlap of more than 60%, jamming is 100% successful, even when using a very quiet jamming signal of only 10dB.

AWGN is used to jam the different wake words *Amazon*, *Computer* and *Echo*. The results are depicted in Figure 6.3d, Figure 6.3e and Figure 6.3f. A noise with an SNR of 10dB can jam reliably the *Amazon* wake word with an overlap of more than 60%. However, once the SNR is less than 10dB, jamming is effective with an overlap of 20% or more. The wake word *Computer* requires an overlap of more than 60% for reliable jamming. *Echo* can be jammed with an overlap of 40%, with the exception of an SNR of 10dB where a jamming success of only 50% is achieved.

This evaluation shows that the required overlap is dependent on the wake word. However, it is also shown that an overlap of 60% is sufficient in most cases which gives a PJD enough time to apply the jamming signal.

Table 6.1 Frequency of the words that start with pronunciation similar to the keywords. The spoken word data is taken from British National Corpus 2014 [85]

Wake Word	Searched Letters	Frequency (%)	Overlap
Alexa	ale*	0.0014	52%
Amazon	am*	0.0896	56%
	em*	0.0625	
	Total	0.1585	
Computer	com*	0.2832	74%
Echo	ech*	0.00042	47%
	ek*	0.000061	
	ak*	0.00025	
	ach*	0.00292	
	Total	0.00798	

6.4.2 Inaudible Jamming

Using the wake word *Alexa* the effectiveness of the two inaudible jamming signals is investigated. The results of this experiment are shown in Figure 6.4a and Figure 6.4b.

With the 20kHz-24kHz jamming signal and an SNR of -20dB and an overlap above 50% reliable jamming is achieved. With the 22kHz-24kHz signal reliable jamming is achieved with a 50% overlap. The result here is interesting as it is possible to jam the system with a noise signal that exists outside the spectrum that voice occupies.

To this end we can only speculate why this approach is possible; we offer three explanations: (i) The wake word recognition algorithm extracts features from the spectrum; the inaudible frequency range below 24kHz may be included. (ii) The frequency leakage in the audible frequency range is sufficient to affect the recognition process. (iii) The PocketSphinx integrates a software AGC and the low frequency voice signal is reduced as the gain is adjusted to the high frequency noise.

The experiments show that there is potential to develop a jamming approach that makes use of jamming signals that are inaudible to humans and are therefore non intrusive.

6.5 False Positive Jamming Evaluation

The PJD may trigger unnecessary jamming as it must recognize the keyword before the PVA. With the time window available to the PJD, words with similar beginning to the wake word may trigger jamming. These *false positive* jamming events are not desirable as they introduce unnecessary noise nuisance.

We investigate the false positive jamming attempts by looking at the frequencies of words in a spoken word corpus that start with a pronunciation similar to the wake words. We use the British National Corpus 2014 [85] consisting of 11,422,617 words. We search for the words that start with similar pronunciation to the wake words. For example, we consider the words starting with *ale* for the keyword *Alexa*. It should be noted that some words starting with *ale* may not be pronounced similar to Alexa (for example, the word *ale* itself). Thus, the results represent a worst-case analysis.

Table 6.1 shows the results. If we consider *ale**, 0.0014% of commonly spoken words are similar to *Alexa*. The overlap for jamming in this case is 52%. If the PJD uses *ale** as trigger, the last 52% of the wake word can be jammed and we would expect a false positive jamming for 0.0014% of spoken words. The overlap here is according to syllables boundaries and an analysis of overlap values of exactly 50% or 60% is not sensible.

We believe that false positive rates are acceptable for a practical jamming device, especially if the jamming signal is in the inaudible frequency space.

6.6 Chapter Conclusion

Our work shows that reactive jamming of PVAs wake words is a feasible approach. The approach can be used for protection, to control when PVAs can function. However, the mechanism could also be exploited by an attacker to block PVA services.

We have demonstrated that modestly strong audio signals with 10dB SNR and an overlap of 60% (with AWGN) can block wake word detection with a 100% success rate in most cases. This means that the PJD has at least 40% of the wake word duration to make a jamming decision. We have shown that this may lead to a false jamming; however, the false jamming rate is very small and should be acceptable for most practical scenarios. We have also shown that it is feasible to move the jamming signal into the inaudible frequency range, making it more applicable.

Our next steps are to carry out an evaluation with off-the-shelf PVAs and to supply audio and jamming signals via speakers instead of directly supplying generated audio signals to the PVA. We also plan to improve the design of inaudible jamming signals and to construct a practical PJD.

Chapter 7

SonarSnoop

7.1 An Active Acoustic Side-channel Attack via Tracking Human Movements Using Smart Devices

As mentioned in the PVA definition in Chapter 1, PVAs have hardware and software for recording, processing and analysing sound. Speakers are used in PVAs for providing audio feedback and providing necessary functions (playing music or reading news). These acoustic components and the sufficient computing performance of PVAs have the full potential to be repurposed to be an advanced sensing system. For instance, Google Nest Guard has dormant microphones which can be activated by software updates to become a Google Assistant enabled device [139]. Similarly, acoustic sensing can be activated by mere software updates. Because PVAs usage scenarios are close to user daily life, PVA sensing studies focusing on user information security and privacy are critical. Both knowledge from acoustic sensing area and unique features of PVAs should be combined in these PVA sensing studies.

Radar and sonar systems use radio and sound waves to track objects, including humans. In recent years this technology has been developed extensively to support human computer interactions by tracking the movement of human bodies, arms, hands or even fingers [105, 93]. However, existing work has rarely investigated the security implications of those technologies.

In this chapter, we report some alarming security implications of tracking human movement via sound waves using acoustic components of a PVA (i.e., a smartphone). Specifically, we present the first active acoustic side-channel attack that can be used to steal sensitive information such as Android unlock patterns. In our attack, human inaudible acoustic signals are emitted via speakers and the echo is recorded via microphones, turning the acoustic system of a smartphone into a sonar system. The echo stream not only gives us information about a victim's finger movement but leaks her secrets too.

All known acoustic side-channels attacks are passive, meaning that acoustic signals in the side-channel are generated by the victim but are eavesdropped by the attacker. In contrast, our approach is an active side-channel, meaning that acoustic signals in the side-channel are induced by the attacker.

In our experiment we use an off-the-shelf Android phone as an example of a computer system with a high quality acoustic system. We re-purpose the acoustic system for our side-channel attack. An inaudible signal is emitted via speakers and the echo is recorded via microphones turning the acoustic system of the phone into a sonar system. Using this approach, an attacker that obtains control over a phone's speaker and microphone is able to observe user interaction, such as the movement of the user's fingers on the touch screen. As the emitted sound is inaudible for the user, it is hard to detect that the sound system is being used to gather information.

To illustrate the capability and potential of this novel active acoustic side-channel attack, we use the task of stealing Android unlock patterns as a case study. We choose this example, since Android is a popular phone OS, and since its unlock patterns are one of the most widely used authentication mechanisms. Our aim is to demonstrate the general viability of our new acoustic side channel, not to improve the specific task of stealing unlock patterns. It would be interesting future research to compare the effectiveness of our approach with older methods, and to identify the best method for stealing unlock patterns, but these are beyond the scope of the work in this chapter.

It might appear that our contribution is merely another phone-based side channel, among many of those that have been investigated for smartphones [125]. However, this is a false impression. Although our experiments are carried out with a phone, the method we show is applicable to many other kinds of computing devices and physical environments where microphones and speakers are available. Perhaps more importantly, when examined in the context of acoustic attacks, our work is particularly significant in that it is the first *active* acoustic side-channel to be reported.

Specific contributions of the work in this chapter include:

1. *SonarSnoop Framework*: We establish the first active acoustic side-channel attack and present *SonarSnoop*, the framework of generic value to construct and implement such attacks.

2. *Unlock Pattern Stealing*: We evaluate the performance of SonarSnoop in stealing unlock patterns, and show that the number of unlock patterns an attacker must try

until a successful authentication can be reduced by up to 70% using the acoustic side-channel. This attack is entirely unnoticeable to a victim; no noise and no vibration are induced.

3. *A family of security threats*: We discuss a number of new attack scenarios that extend our experiment setting. We show that SonarSnoop represents a family of new threats.

The next section describes relevant background on phone unlock patterns, the acoustic side-channel and how to exploit it for an effective attack. Section 7.3 describes SonarSnoop, the system used to spy on user interactions with a phone, discussing in detail the challenging aspects of signal generation and signal processing necessary to reveal user interaction. Section 7.4 describes our experimental evaluation using a user study. Specifically we evaluate the effectiveness of different decision making strategies. Section 7.5 discusses findings and limitations. Section 7.6 generalises the SonarSnoop attack, discusses further attack scenarios, potential countermeasures and broader implications of acoustic side-channel attacks. Section 7.7 concludes this chapter.

7.2 Stealing Phone Unlock Patterns via Acoustic Side-channel Attacks

Unlock patterns are often used to secure access to Android phones. We investigate a novel active acoustic side-channel attack to steal these patterns. Previous research investigated different methods for stealing unlock patterns, e.g. using smudges [15], accelerometer readings [16] or video footage [154]. These work did not mainly use acoustic channel, so details are not included in this thesis. Other related work making use of acoustic channel are described in Section 3.2.4-B.

7.2.1 Phone Unlock Patterns

We consider the unlock pattern mechanism available on Android phones. The user is presented with a 3×3 grid of positions. Using the touch screen the user has to draw a pattern, connecting positions in the correct sequence to authenticate.

Figure 7.1 shows some examples of such an *unlock pattern*. For the first pattern on the figure, the user has to connect 5 *positions* on the screen in the correct order starting from the top left position. The phone is blocked if the user fails to draw the correct pattern five times in a short period.

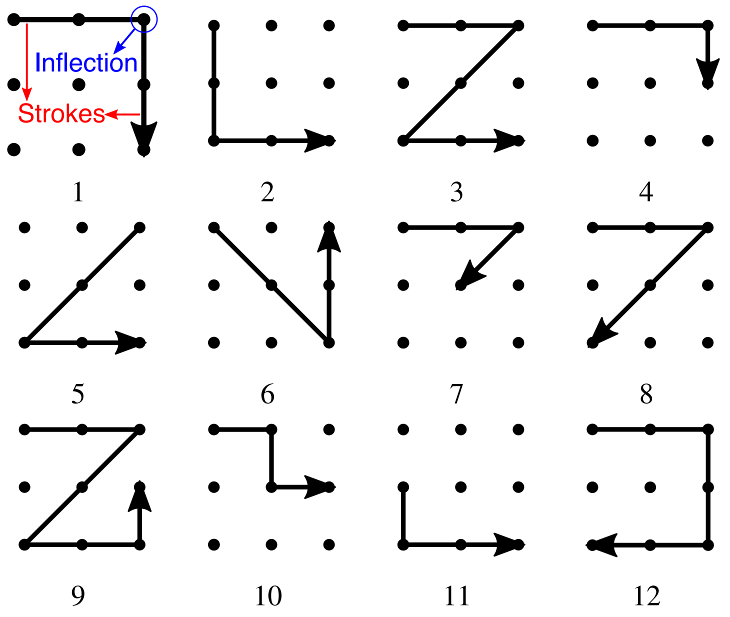


Fig. 7.1 The twelve most popular unlock patterns according to [32]. An unlock pattern connects a number of dots, and it can be decomposed into several strokes separated by inflections.

The unlock pattern can be decomposed in multiple *strokes* which are separated by *inflections*. Positions that can be reached without changing drawing direction make up one stroke. We use this approach of pattern decomposition later in this chapter.

In theory there are $389,112 \approx 2^{19}$ possible unlock patterns [132]. However, not every pattern is chosen by users with the same probability. Users have bias in choosing unlock patterns and models have been created to estimate the likelihood of unlock patterns [132]. This bias can be used by adversaries to improve their guess on unlock patterns.

Figure 7.1 gives the 12 most common unlock patterns in the real world, according to a recent empirical study [32]. In our user study, we focus on stealing these most likely patterns only, for the following reasons. First, unlock patterns have a highly non-uniform distribution, and those 12 common patterns account for more than 20% of user choices [32]. We aim for these high priority targets only, just like rational adversaries often choose to do. Second, we aim to make our user study reasonable to participants so that it will not take too much of their time to complete the study. An overly lengthy user study will be tedious and boring, and it will scare away potential participants. In the worst scenario, a bored participant can circumvent the study by producing useless data or otherwise jeopardising our experiment's validity. Third, as mentioned earlier, our purpose is not to steal the most unlock patterns or propose the best experiment of that kind. Instead, our modest aim is to use an experiment to testify the feasibility of our acoustic side-channel attack. We believe our design choice is sufficient for the purpose. Overall, our design choice is not a random decision, but one based on careful deliberation, with multiple factors and their trade-off taken into considerations.

7.2.2 An Acoustic Side-channel

The acoustic channel can be used to infer user behaviour using either a passive or active approach. A passive system assumes that the observation target itself emits sound that can be recorded for analysis. An active system uses speakers to emit a sound wave and microphones to collect echo data.

In this work we use the active approach. The speakers of the system emit an Orthogonal Frequency Division Multiplexing (OFDM) sound signal. We use OFDM because it has a good correlation property [93] and it is easy to confine the signal to the higher inaudible frequency range (see Section 7.3.3). The sound signal sits in a frequency range that is inaudible to most people (18 kHz to 20 kHz). The microphones are used to record the echo. By analysing the recorded echo, it is possible to deduce user interaction patterns with the touch screen.

When using this technique during a user's interaction with the unlock mechanism, information regarding the unlock pattern is leaked which constitutes the acoustic side-channel.

7.2.3 Threat Model

We consider an adversary's goal is to steal a user's unlock pattern. We assume that the adversary uses software deployed on the user's phone to achieve this goal. We further make the assumption that the adversary uses the acoustic system (speakers and microphones) on the phone to achieve this goal. We assume the adversary is able to deploy code on the user's phone which carries out the acoustic side-channel attack.

Typically such code might be installed in form of an App. The adversary may develop an App that incorporates code to execute the acoustic side-channel attack and presents itself to the user as a benign App such as a Weather App or a Game. The existence of Apps with such hidden malicious functionality in the Android Marketplace is well documented [165, 166].

The App will require access to microphones. The user will be asked to grant access to this when the application is first launched. Users often grant such access as they rarely question the necessity of these requests [45]. In addition, the App might be designed such that this permission seems reasonable to the user. For example, the App might have sound effects and offer voice control.

To be effective, the App will have to be active when the user enters the unlock pattern. Thus, the App has to be running in the background and become active when the phone starts.

The App may also make use of available communication channels to transport observed data to a back-end system. The back-end system can be used to analyse the acoustic traces, avoiding suspicious heavy computation on the user's phone. Again, such communication falls within normal operational behaviour of Apps and would not raise suspicion.

The acoustic system is specific to the phone model. Different phones provide a different number of speakers and microphones and they are placed differently. Thus, an active acoustic side-channel attack must be tuned to the model of the phone. We assume that the adversary is able to obtain the same phone model as used by the target in order to adjust signal processing parameters to the target environment.

7.2.4 Attack Success

An adversary is successful if he or she has: (i) deployed malicious code on the target's phone; (ii) collected sufficient data from the acoustic side-channel during the users' interaction with the unlock mechanism; (iii) analysed the data and extracted unlock pattern candidates; and (iv) the number of extracted pattern candidates is smaller than the number of trials the OS allows.

The challenging parts of this attack sequence include collecting useful data from the acoustic side-channel and designing a data analysis framework for inferring unlock patterns. The next sections will focus on these elements. We consider the deployment of malicious code on a target's phone a solved problem, as is the common practice in the literature [165, 166].

7.3 SonarSnoop

This section describes *SonarSnoop*, our framework to execute an acoustic side-channel attack on the Android phone unlock pattern. We call the framework SonarSnoop as we use the acoustic detection to snoop on user interaction, bearing similarities with sonar systems. The system is geared towards unlock patterns; however, by exchanging elements of the signal processing and decision making components the system could be re-purposed for other side-channel attacks such as observing user interaction with a banking App.

Our work was inspired by FingerIO [93], which was a system for user interaction based on active acoustic sonar. However, FingerIO was used to track movement of gestures in the vicinity of a phone while our system requires to track finger movements on the screen. Thus in SonarSnoop, the close proximity and the fact that the user holds the phone during interactions create additional complexity. We also have to modify both signal generation and processing to tackle our attack scenarios.

The speakers of the phone send an inaudible OFDM sound signal which all objects around the phone reflect. The microphones receive the signal and also the reflections (delayed copies of the signal). The time of arrival of all echoes does not change when objects are static. However, when an object (a finger) is moving a shift in arrival times is observed. The received signals are represented by a so called *echo profile matrix* which visualises this shift and allows us to observe movement. Combining observed movement from multiple microphones allows us to estimate strokes and inflections (see Figure 7.1). By combining the estimated sequence of observed strokes, we can then estimate the unlock pattern they represent.

The four main components of SonarSnoop are:

- *Signal Generation*: Using the speakers of the phone an OFDM signal is produced. The signal is inaudible and suitable for close-range tracking of fingers.
- *Data Collection*: Data is collected via the device's microphones.
- *Signal Processing*: Echo profiles are created followed by removal of noise and artefacts. Then features (finger movement direction and distance) are extracted.
- *Decision Making*: Using the extracted features the unlock patterns (represented by their decomposition in strokes and inflections) are discovered. We provide alternative methods to do this.

7.3.1 Signal Generation

Signal generation is based on FingerIO [93] with modifications tailored to our device and application scenario. We introduce some additional processing and filtering steps.

Identical to FingerIO, 48 kHz is used as the sampling frequency. According to Nyquist Theorem, this supports a sound wave of up to 24 kHz. This importantly supports frequencies above 18 kHz, which is the highest frequency most adults can hear. A vector comprising 64 subcarriers, each covering 375 Hz, is composed. All subcarriers outside of the intended band (18 kHz – 20 kHz) are set to 0, and all others are set to 1.

The next signal generation steps are in addition to the mechanism used by FingerIO and are used to adjust to our phone and application scenario. A copy of the vector is reverse ordered, and the two vectors concatenated, resulting in the vector shown in Figure 7.2a. The 128-sample time domain signal is generated using the IFFT. The real part is divided in half, and the first half used as the signal. As this introduces spectral leakage into the audible hearing range we remove unwanted low-frequencies using a Hanning window. The final signal in the time domain is shown in Figure 7.2b. The signal is padded with silence to introduce a 264-sample interval, and a duration of 5.5 ms. This ensures that all echoes are

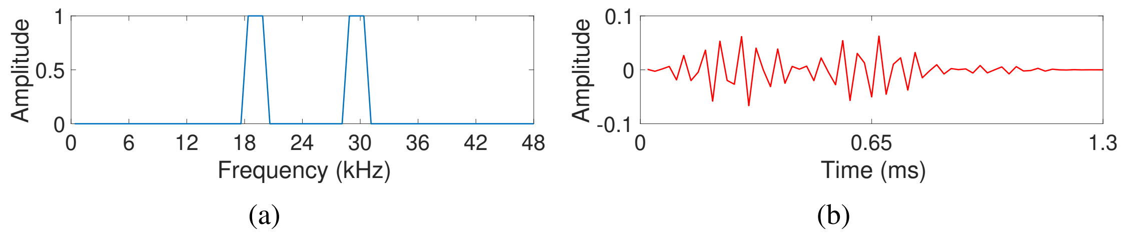


Fig. 7.2 Sonar signal generation. (a) 128-point vector in frequency domain and (b) 64-point signal in time domain.

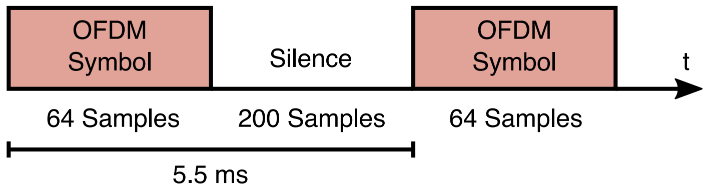


Fig. 7.3 The sound frame as played continuously over the speakers.

captured before the next pulse is emitted. The frame is repeated continuously, producing a signal as shown in Figure 7.3.

We expect that further optimisation is possible. However, we found the performance to be sufficient for our work.

7.3.2 Data Collection

The phone's microphones are used to record sound data using a sampling frequency of 48 kHz. Noise is introduced by the environment and is recorded together with the received OFDM signal. However, ambient noise does not interfere excessively with the signal processing stage.

7.3.3 Signal Processing

Signal processing comprises (i) echo profile generation, (ii) noise removal, and (iii) feature extraction.

Echo Profile Creation

Echo data is recorded and transformed into an echo profile for each present microphone. The processing for each microphone is the same, except for parameter settings taking into account the positioning of microphones in relation to movement locations. Most modern phones provide at least two microphones, one on top of the phone and one on the bottom; we tailor the following process to two microphones but the described methods can be extended to more microphone inputs.

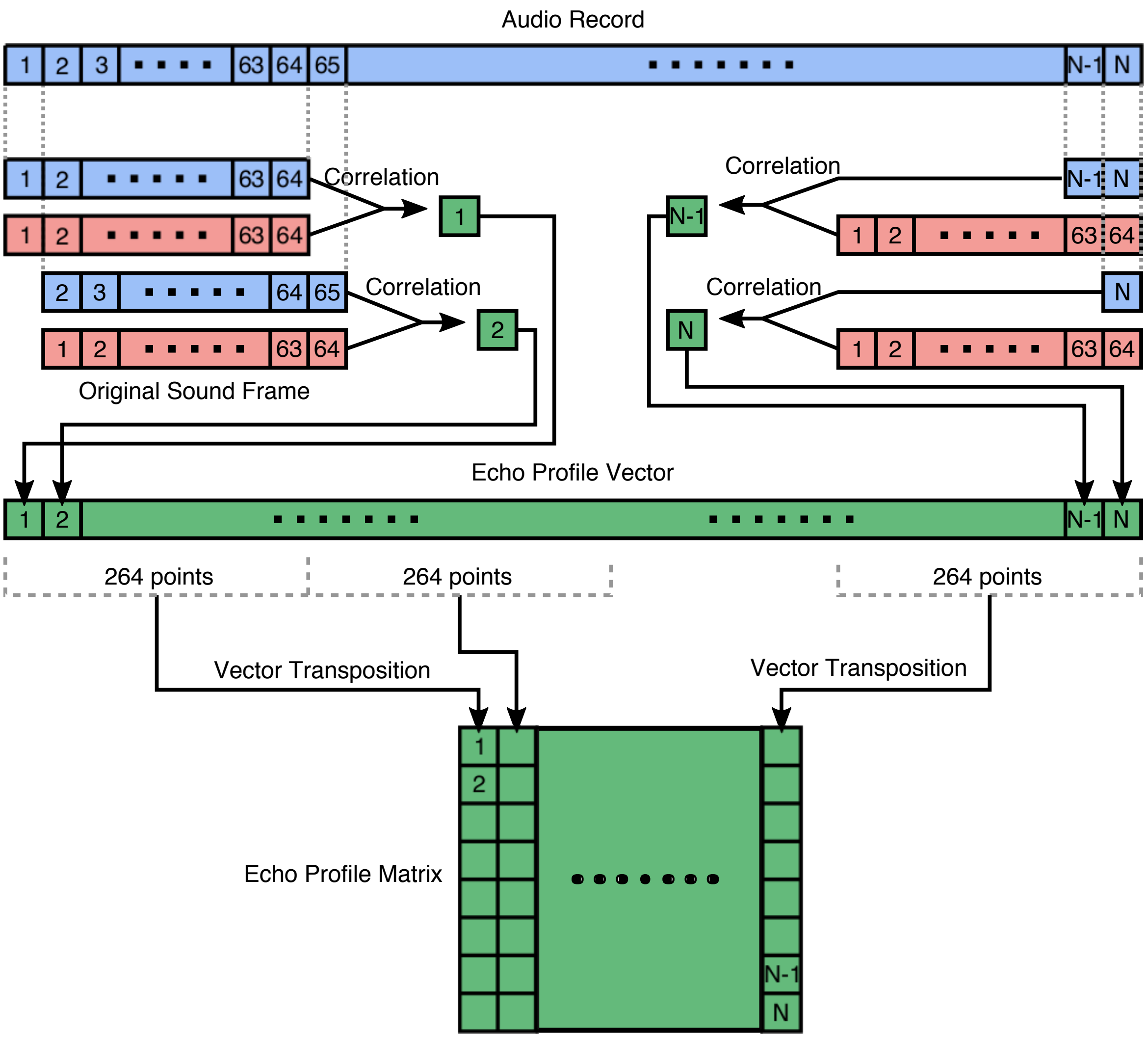


Fig. 7.4 The process describing the transformation of the audio signal into the echo profile vector and finally the echo profile matrix.

We create the echo profile by calculating the correlation between the original sound frame and the echo data (see Figure 7.4). The original sound frame is a 64-point signal (64 sample points in the time domain over a duration of 1.3ms). Therefore, we take 64-point sized chunks from the recorded echo data and apply a sliding window, shifting 1-point at a time, and calculating the correlation with the original sound frame. Each correlation result is then concatenated to create the *echo profile vector*.

The data emitted on the speakers consist of periodical 264-point long sound frames, and echoes are observed within this period. When there is no object movement, echoes will be observed at the same time within each 264-point frame. When objects move, echo positions will change within each following 264-point frame. This can be visualised by transforming the echo profile vector into an *echo profile matrix*. We take 264-point sized chunks from the echo profile vector and transpose them to create the echo profile matrix. Figure 7.5a shows an example echo profile matrix. The x-axis and y-axis of the matrix correspond to time and distance, respectively.

When an object moves, slight variations comparing one column of the echo profile matrix to the next can be observed. Depending on the microphone location in relation to the movement and the speed of moving objects, it is necessary to compare column i with column $i+\delta$ to see clear changes. For the phone used in our experiments we set $\delta = 8$ (44 ms separation) for the bottom microphone and $\delta = 16$ (88 ms separation) for the top microphone. We chose these values as they provided the best performance for our application case. Figure 7.5b shows an example of the echo profile matrix after subtraction of values in column i and $i+\delta$. The finger movements are now clearly visible and can be analysed by suitable algorithms.

Noise Removal

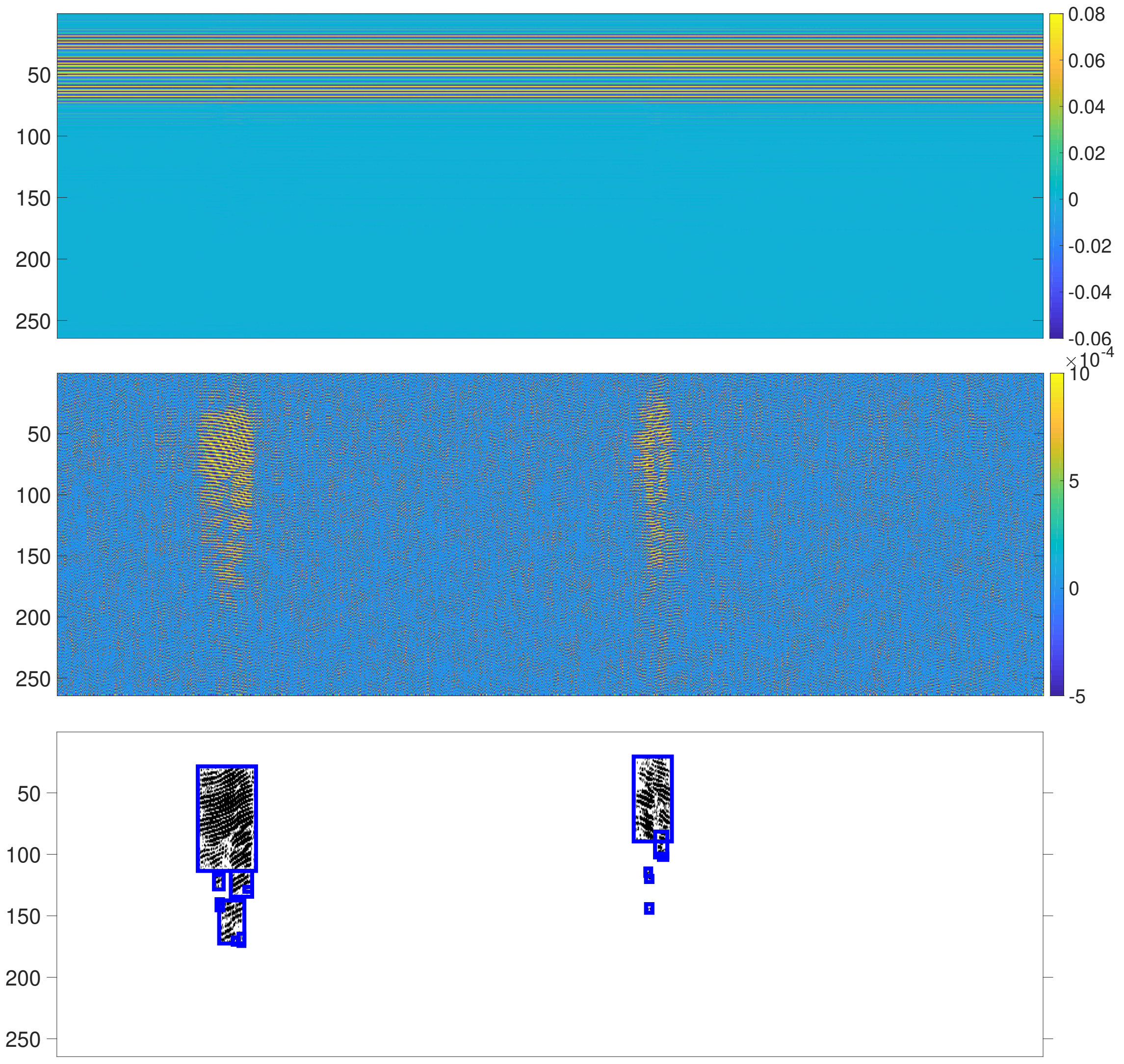
Before analysing data captured in the echo profile matrix, noise is removed. We consider here noise from the sonar system and not ambient sound, because such ambient noise does not correlate with our original signal and therefore does not interfere with the sonar signal analysis. An example result of this clean up procedure is shown in Figure 7.5c which corresponds to the data shown in Figure 7.5b.

First, we transform the echo profile matrix into a binary matrix by setting values above a threshold to 1 and below to 0. Thus, only correlation above the threshold is taken into account as this corresponds to significant movements. The threshold is chosen as the 94^{th} percentile of all the values in the matrix. We found that this threshold setting performs well in the context of our work.

We use image processing techniques to extract features from the echo profile matrix. Thus, our next step of noise removal is tailored to this method of feature extraction. We use the concept of Connected Components (CCs) to detect areas of activity (corresponding to strokes) in the binary echo profile matrix. Each CC is defined as area containing more than 20 connected 1s. We remove all 1s from our echo profile matrix that are not included in such CCs. Again, a threshold of 20 was found to be suitable for our application context.

Feature Extraction

We use the CCs to locate areas of movement in the binary echo profile matrix. Each CC is described by a Bounding Box (BB) which is the smallest rectangle surrounding the CC as shown in Figure 7.5c. CCs that identify one stroke are grouped together. For each group of



(c)

Fig. 7.5 (a) Raw echo profile matrix. (b) Echo profile matrix after column-wise subtraction. (c) Echo profile matrix after binarisation and segmentation; blue-coloured bounding boxes indicate detected strokes.

CCs we extract two features: (i) movement direction and (ii) movement distance. Movement direction relates to the angle of lines visible in the CCs of a group. Movement distance relates to the height of BBs in each group.

Before extracting features we exclude some CCs. We remove CCs that are only visible on one microphone input; movement should be detected clearly by both microphones at the same time. We also remove overlapping CCs when more than half of the smaller CC overlaps in x and y direction. Thus, the number of CC within a group is reduced, simplifying analysis without loosing accuracy.

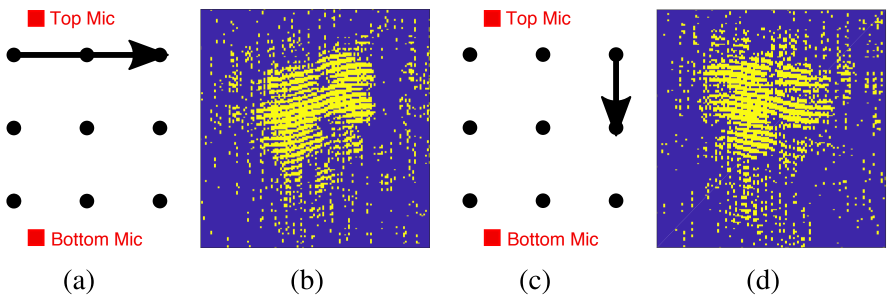


Fig. 7.6 Two strokes with different direction information. (a) A stroke that is moving away from the bottom microphone. (b) Connected Components (CC) of the stroke shown in *a* extracted from the bottom microphone. It shows an ascending trend. (c) A stroke that is moving towards the bottom microphone. (d) CC of the stroke in *c* extracted from the bottom microphone. It shows a descending trend.

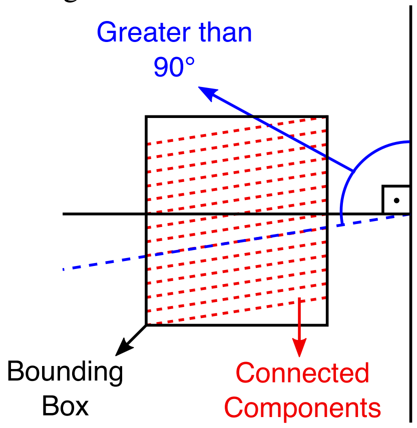


Fig. 7.7 The relation between the ascending trend of a Connected Component and the angle (orientation) result of Gabor filter.

CCs are assigned to groups by using a separation of 80 columns (i.e., by 440 ms) on the x axis. A user pauses between strokes and we use this separation to group CCs belonging to the same stroke. This method relies on a visible pause at inflections. Other data analysis methods need to be used to determine CC groups when this cue is not present. We did experiments during which a user does not need to pause at each inflection. The results remain similar only except CCs of different strokes connect together. This fact does not invalidate our approach but additional image processing techniques are in need to separate these CCs.

Objects moving away from a microphone produce an ascending trend in the CC, while objects moving towards produce a descending trend. Figure 7.6 gives an example of this behaviour pattern; two strokes captured by the bottom microphone are shown. The CC in Figure 7.6b has an ascending trend, and the CC in Figure 7.6d has a descending trend. To identify these trends automatically we use a Gabor filter [131], which is a well-known linear filter and often used for extracting orientation information. We quantify the orientation (i.e.

the angle) of lines within the CC in each BB using Gabor filter. If the angle is greater than 90° as shown in Figure 7.7, it means that an object is moving away from a microphone; while an angle smaller than 90° means that an object is moving towards the microphone. After obtaining the angle information of each BB belonging to a stroke we combine these into a single value. We weigh the angle information of each CC by the size of the BB. In the remainder of the chapter, we call this feature representing a stroke's direction the *angle*.

Movement distance can be inferred from the heights of the BBs within a group. As a group corresponds to a stroke in our case, the height of each BB contributes to the movement distance of a stroke. If the stroke is long, the BBs covers more space vertically. In the remainder of the chapter we refer to this feature as the *range*.

7.3.4 Decision Making

SonarSnoop gathers stroke information via the features angle and range. This information has to be translated into a meaningful user interaction pattern. Depending on the application, very different decision making processes can be appropriate.

We consider in this chapter only the task of stealing phone unlock patterns as described in Section 7.2.3. For this purpose we define 3 different decision making options named D1, D2, D3 which operate very differently.

- **D1** simply uses the features angle and range and classifies each stroke. The resulting sequence of strokes is then the assumed unlock pattern of the user.

- **D2** uses only the angle feature of strokes. The sequence of directions reveals a set of candidate patterns. The set of candidate patterns is likely to contain the user's unlock pattern.

- **D3** combines D2 and D1. First, a set of candidates is determined by investigating the angle feature of strokes. Then within the candidate set angle and range are used to classify strokes and identify the user's pattern.

Users do change unlock patterns infrequently and once malicious software is deployed on a phone multiple unlock procedures can be observed. For each observed unlock procedure the decision making process provides one or more candidate pattern. All proposed candidate patterns are ordered according to the number of times they were suggested. The position of the user's pattern in the list of suggested patterns determines the effectiveness of the side channel attack.

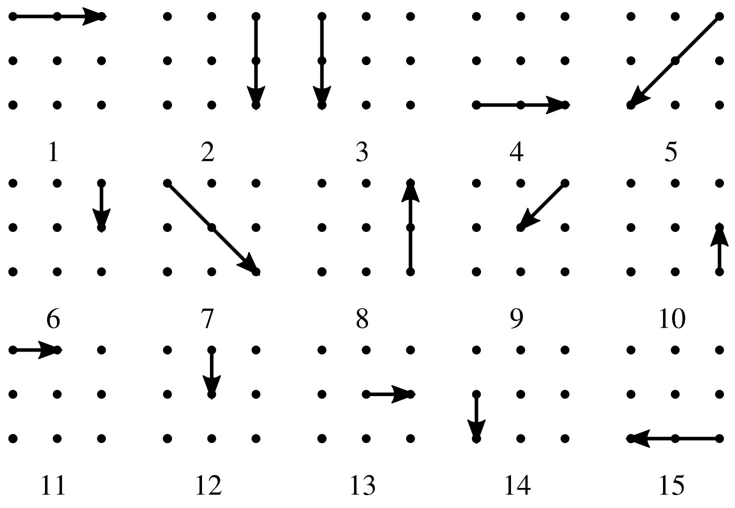


Fig. 7.8 The 15 strokes used to compose the 12 unlock patterns shown in Figure 7.1.

D1 - Classifying Strokes Using Angle and Range

We classify the strokes using machine learning. The sequence of classified strokes reveals the unlock pattern. We use the 12 most likely unlock pattern as shown in Figure 7.1 which decompose into 15 unique strokes as shown in Figure 7.8. Sample data for each of the 15 strokes from 2 individuals (trainers) is used to train the classifier. Trainers are not subjects of the user study. We use the direction (angle) and distance (range) of the strokes obtained from the echo data of both microphones.

There are 3 variants of this decision making process: **(D1.1)** using data from both microphones; **(D1.2)** using data from the bottom microphone only; **(D1.3)** using data from the top microphone only.

D2 - Grouping Patterns Using Angle

Table 7.1 Groups of patterns that have the same number of strokes with the same behaviours when using one microphone. Behaviours are shown as *A* if the stroke is moving away from the microphone, and *T* if the stroke is moving towards to the microphone.

Patterns	**Bottom Mic.**	**Patterns**	**Top Mic.**
1, 4, 7, 8	A - T	1,2,4,5,8,11	A - A
2, 5, 6, 11	T - A	3, 10	A - A - A
3, 10	A - T - A	6, 7	A - T
9	A - T - A - A	9	A - A - A - T
12	A - T - T	12	A - A - T

Table 7.2 Groups of patterns that have the same number of strokes with the same behaviours when using both microphones. Behaviours are shown as *A* if the stroke is moving away from the microphone, and *T* if the stroke is moving towards to the microphone.

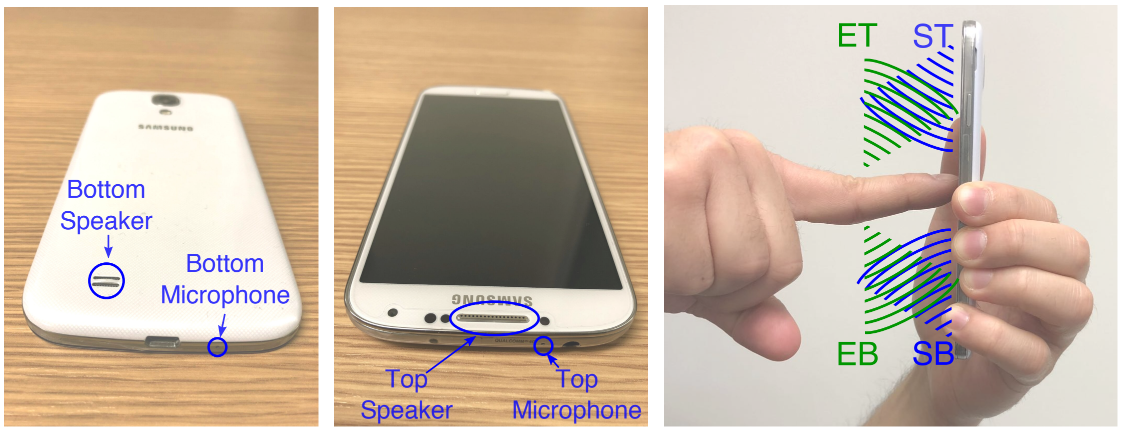
Patterns	**Bottom Microphone**	**Top Microphone**
1, 4, 8	A - T	A - A
2, 5, 11	T - A	A - A
3, 10	A - T - A	A - A - A
6	T - A	A - T
7	A - T	A - T
9	A - T - A - A	A - A - A - T
12	A - T - T	A - A - T

For this method we use only the direction (angle) of a stroke, whether it is moving towards a microphone or away. Then we look at the combination of the strokes to guess the pattern. For example, the first stroke of the Pattern 1 in Figure 7.1 is *moving away* from the bottom microphone and the second stroke is *moving towards* the bottom microphone. Pattern 4 , Pattern 7 and Pattern 8 have the same behaviour from the perspective of the bottom microphone. Therefore, using only angle information, a group of patterns is identified. Table 7.1 shows pattern groups for each microphone that have the same stroke behaviour when considering patterns as shown in Figure 7.1.

The group sizes can be reduced by considering data from both microphones together. For example, the first stroke of the Pattern 1 in Fig 7.1 is *moving away* from the bottom microphone and the second stroke is *moving towards* the bottom microphone; considering the top microphone the first and second stroke are *moving away* from microphone. Only Pattern 4 and Pattern 8 have this same behaviour. Table 7.2 shows pattern groups that have the same stroke behaviour.

It may happen that the analysis of stroke patterns using angle information of both microphones is inconclusive. For example, the strokes from the top microphone are reported as *moving towards* and *moving towards*, and the strokes from the bottom microphone are reported as *moving away* and *moving towards*. In this case, no group mapping exists as the combination cannot be mapped to any entry in Table 7.2. In such situation where no match is possible we choose to fall back on data collected from one microphone.

We use four strategies to operate this decision making process: **(D2.1)** using data from both microphones, using only the bottom microphone in inconclusive situations; **(D2.2)** using data from both microphones, using only the top microphone in inconclusive situations; **(D2.3)** using data from the bottom microphone only; **(D2.4)** using data from the top microphone only.



(a) (b) (c)

Fig. 7.9 Galaxy S4 used in the experiments. (a) Location of the bottom speaker and the bottom microphone. (b) Location of the top speaker and the top microphone. (c) Simplified reflection paths of the bottom speaker (SB), the top speaker (ST), the echo coming to the bottom microphone (EB), and the echo coming to the top microphone (ET).

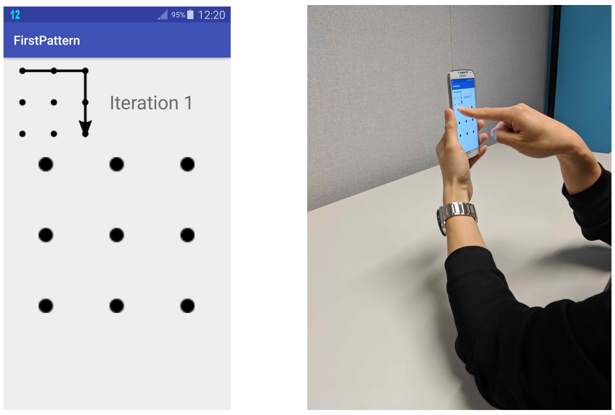
D3 - Grouping Patterns Using Angle and Classifying Strokes Using Angle and Range

We combine the first two approaches to improve the overall accuracy. We first use method D2 to identify a pattern group, then we select a specific pattern from this group using method D1. This approach improves on using D1 alone as the pool of candidate patterns is reduced before machine learning is applied. We train machine learning models for the strokes of each group using corresponding microphone's data.

Similar to method D2, four different operation modes can be used: **(D3.1)** using data from both microphones, using only the bottom microphone in inconclusive situations; **(D3.2)** using data from both microphones, using only the top microphone in inconclusive situations; **(D3.3)** using data from the bottom microphone only; **(D3.4)** using data from the top microphone only.

7.4 Experimental Evaluation of SonarSnoop

We evaluate SonarSnoop using a Samsung Galaxy S4 running Android 5.0.1. A dedicated evaluation App is used for a user study to prompt users to input unlock patterns which we then aim to reveal using SonarSnoop.



(a) (b)

Fig. 7.10 (a) Screenshot of the App used for our user study. (b) An example demonstration of the user study.

7.4.1 User Study

The Samsung Galaxy S4 provides two speakers and two microphones as shown in Figure 7.9. We execute the data collection component of SonarSnoop on the phone. Signal generation, signal processing and decision making are executed on a dedicated PC.

For the experiments we develop a dedicated evaluation App instead of using the built-in pattern unlock mechanism of Android. The App replicates the pattern unlock mechanism and provides additional features to simplify evaluation. The App provides the user with a 9-point matrix to enter unlock patterns. In addition, the user interface shows the pattern we expect the user to draw. The user interface is shown in Figure 7.10a. The evaluation App guides a user through the experiment and ensures that echo data recorded by SonarSnoop can be matched with the pattern the user was asked to draw (ground truth).

We ran a user study with 10 volunteers (we obtained approval for the study from the University Ethics Committee). Each volunteer was asked to draw the 12 unlock patterns as shown in Figure 7.1 five times. The evaluation App guided the volunteers through this process which took up to 30 minutes.

The participant is asked to hold the phone with one hand and to draw the pattern with the other. The participants sat at a table and rest the arm holding the phone with their elbow on the table (see Figure 7.10b).

All experimentation was carried in an open plan office without restrictions on the environment noise. During our experiment usual office noise was present (people moving, chatting, moving chairs, opening doors). The results shows that our approach is fairly robust against such environment noise.

7.4.2 Evaluation Metrics

Using the data collected in our user study we evaluate the three different variants of our decision making process. To judge performance we use five key metrics:

- Pattern guess rate per user **(M1)**: For each user we calculate the ratio of successfully retrieved patterns to the number of patterns in the pool (i.e., 12). A pattern is successfully retrieved if the decision making suggests a set of patterns which contains the correct one.

- Pattern candidates per user **(M2)**: Without the aid of our side-channel attack, the attacker must perform a random guess (i.e. selecting randomly from a pool of 12 patterns in our case). This metric describes the average size of the pattern pool after the decision making per user. If the size of the remaining pattern candidate pool is reduced, it will improve pattern guess rate.

- Pattern guess rate per pattern **(M3)**: This metric is similar to M1. However, the rate of success is per pattern instead of per user. The metric shows how successful a specific pattern is retrieved across all users within our study.

- Pattern candidates per pattern **(M4)**: This metric is similar to M2. Here the average size of the pattern pool after decision making is calculated per pattern across all users in the study.

- Attack attempts **(M5)**: This metric is based on M2. However, the unlock patterns are ordered by the number of times they were suggested. This gives the sequence of patterns an attacker will try. M5 is the position of the user's pattern in this ordered pattern pool.

7.4.3 Decision Making Option D1

With this decision making variant we classify the individual strokes of the patterns and then deduce the pattern from the result (see Section 7.3.4). Figure 7.8 shows the 15 unique strokes of the 12 patterns shown in Figure 7.1 which are elements of our study. We train our machine learning model using data obtained from 2 trainers. We collect between 30 and 40 samples in total for each stroke. We test various algorithms using 5-fold cross validation on the training data, and pick *Medium Gaussian SVM* algorithm with *kernel scale* parameter 2.2, as it performs best among other algorithms in terms of accuracy.

Figure 7.11 shows the classification accuracies for the 15 strokes shown in Figure 7.8 for 10 users that participated in our study. The figure shows accuracies for each user with

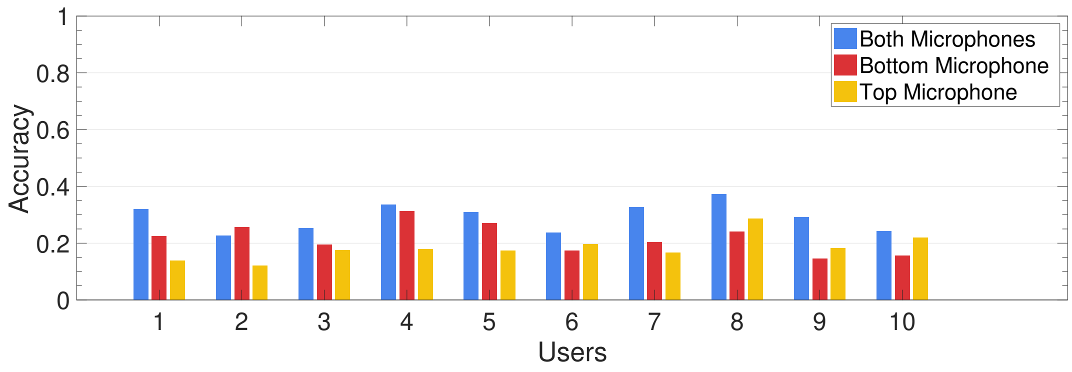
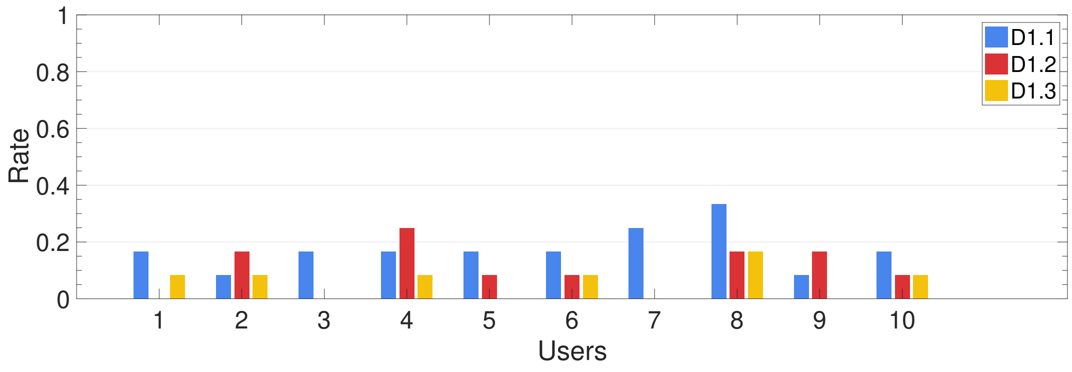
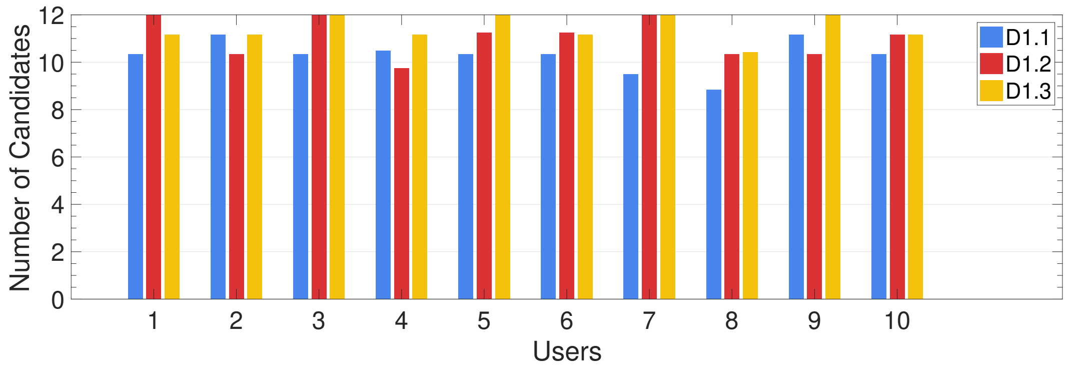


Fig. 7.11 Classification accuracies for 15 strokes shown in Figure 7.8 for 10 users.



(a)

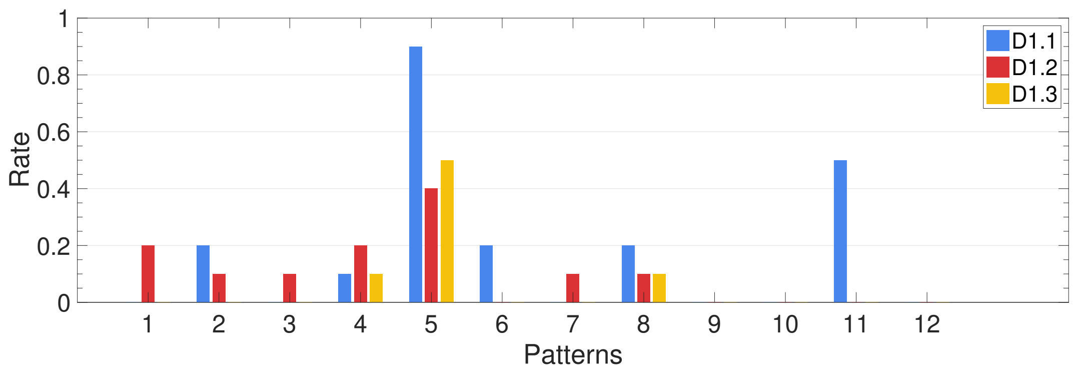


(b)

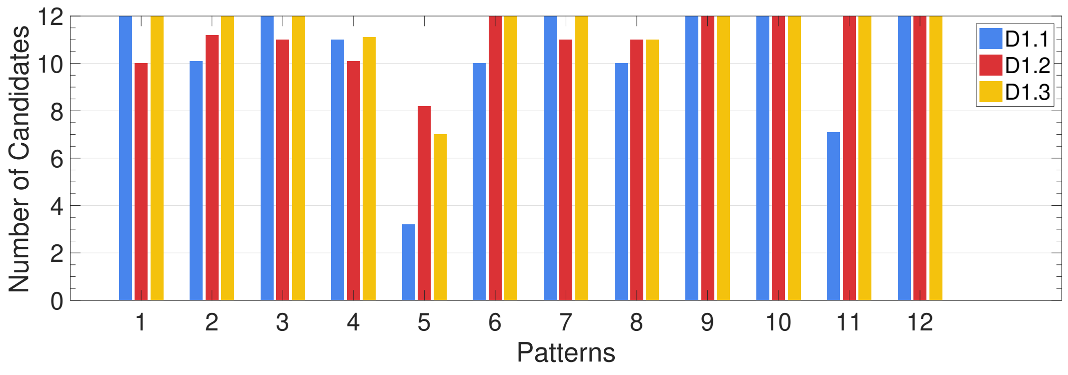
Fig. 7.12 Results per user with decision method D1. (a) Pattern guess rate (Metric M1). (b) Average number of pattern candidates remained after predictions (Metric M2).

different combinations of feature sets. Using both microphones gives the best performance as we would expect. The highest accuracy value of 0.37 is achieved with User 8 when using both microphones. We obtain the best overall average accuracy when using both microphones (Accuracy of 0.29).

Next we combine the classified strokes to guess the users' pattern. Figure 7.12a shows the pattern guess rate per user (Metric M1). The rate is shown using variation D1.1, D1.2 and



(a)



(b)

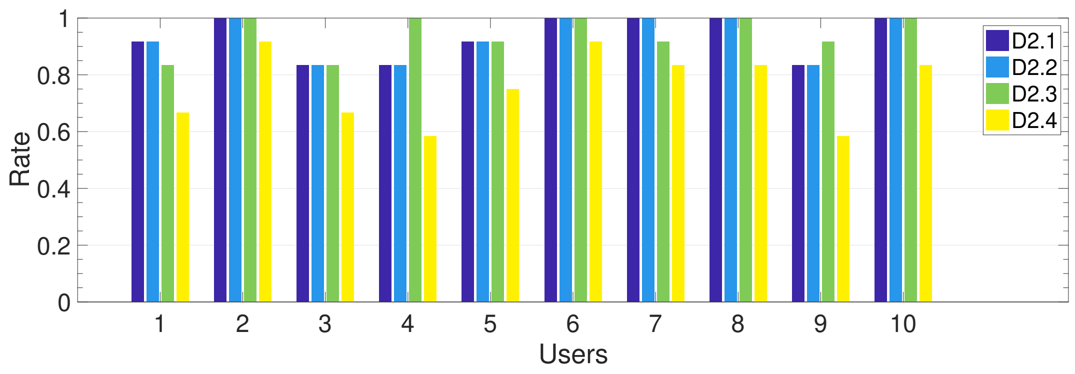
Fig. 7.13 Results per pattern with decision method D1. (a) Pattern guess rate (Metric M3). (b) Average number of pattern candidates remained after predictions (Metric M4).

D1.3 of our decision making method D1. As a reflection of the results shown Figure 7.11, we obtain the best rate of 0.33 for User 8 when using data from both microphones, and the best average value of 0.18 across all users is also achieved when using both microphones.

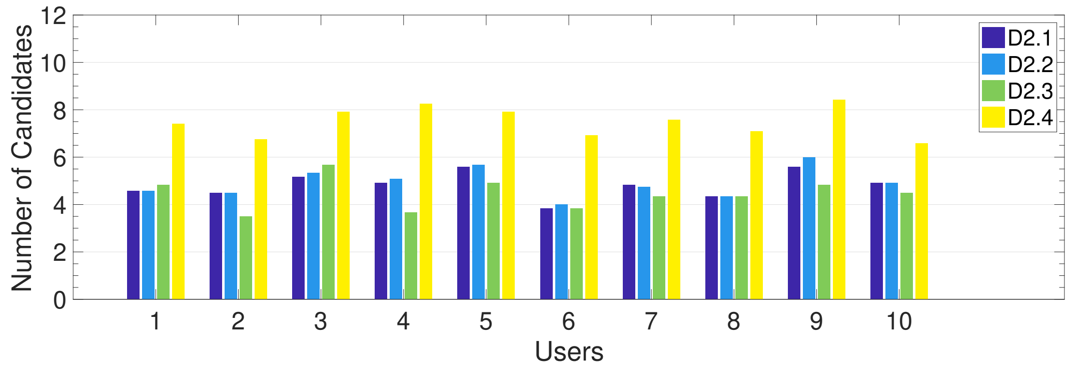
Figure 7.12b shows the average number of candidate patterns remained after predictions for each user (Metric M2). A minimum number of candidates of 8.83 is achieved when using both microphones for User 8, and we obtain the minimum average value of 10.28 when using both microphone across the user population.

Figure 7.13a shows pattern guess rate across all users for each pattern (Metric M3). Although the average rate of 0.18 is achieved when using both microphones, Pattern 5 is revealed for 9 users within 5 iterations.

Figure 7.13b shows the average number of candidates remained after predictions for each pattern (Metric M4). The minimum value of 3.20 is achieved with Pattern 5 when using both microphones. We obtain the minimum average value of 10.28 when using both microphones.



(a)



(b)

Fig. 7.14 Results per user with decision method D2. (a) Pattern guess rate (Metric M1). (b) Average number of pattern candidates remained after predictions (Metric M2).

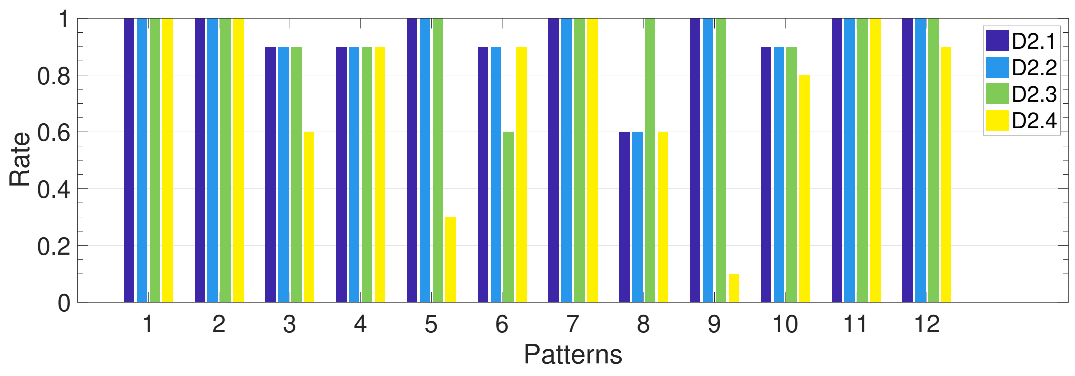
Summary The results show that method D1 reduces the candidate pool of patterns (Metrics M2, M4). Thus, we show that the acoustic side-channel is generally useful to an attacker. However, the improvement is not very significant. The average number of candidate patterns for the attacker to try is reduced from 12 to 10.28 (Metrics M2, M4).

7.4.4 Decision Making Option D2

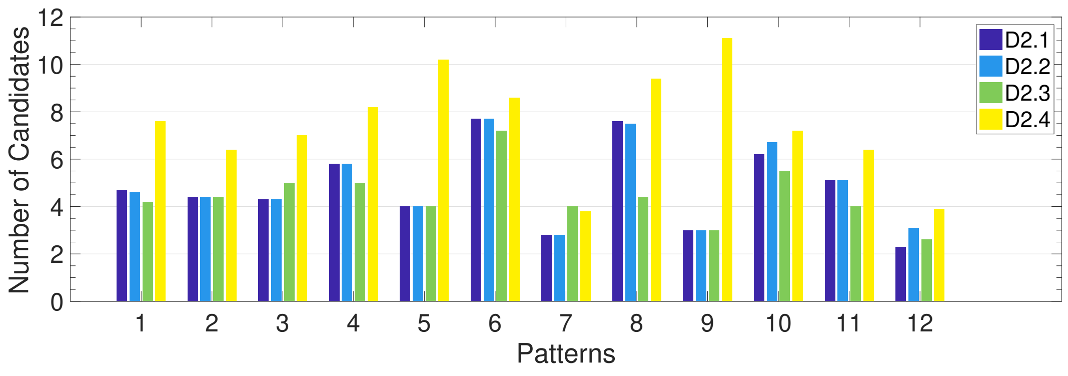
With this decision making variant we identify candidate groups based on the angle information. Some patterns share the same number of movements with the same behaviours (moving away or moving towards), and we cannot narrow the decision down to a single pattern.

Table 7.2 shows groups of patterns that have the same number of movements with same behaviours when using both microphones.

The rates (Metric M1) as shown in Figure 7.14a are above 0.83 for all users when using both microphones (D2.1 and D2.2). The patterns of Users 2, 6, 7, 8 and 10 are mapped to their groups shown in Table 7.2 with 100% success when using both microphones, and the best average value of 0.93 for M1 is achieved when using both microphones (D2.1 and D2.2).



(a)



(b)

Fig. 7.15 Results per pattern with decision method D2. (a) Pattern guess rate (Metric M3). (b) Average number of pattern candidates remained after predictions (Metric M4).

The minimum number of candidates as shown in Figure 7.14b (Metric M2) of 3.50 is achieved for User 2, and we obtain the minimum average value of 4.44 when using only bottom microphone's data (D2.3).

Most of the patterns are mapped to their correct groups (Metric M3 shown in Figure 7.15a), and an overall average value of 0.93 is achieved when using the data from both microphones (D2.1 and D2.2).

The average number of candidates remained after predictions for each pattern are shown in Figure 7.15b (Metric M4). A minimum value of 2.30 is achieved for Pattern 12 when using both microphones and the bottom microphone is dominant (D2.1). We obtain the minimum average value when using only the bottom microphone (D2.3) which is 4.44.

Summary D2 performs significantly better than D1. The number of pattern candidates is significantly reduced from 12 to 4.4 (Metrics M2, M4). This is an interesting result as this method uses only one feature (angle) for the decision making. However, the attacker still has more than one candidate due to similarities of patterns and their grouping.

Table 7.3 Groups of unique strokes to be trained for each pattern group when using both microphones, only bottom microphone, and only top microphone.

Both Microphones		**Bottom Microphone**		**Top Microphone**	
Patterns	**Strokes**	**Patterns**	**Strokes**	**Patterns**	**Strokes**
1, 4, 8	2, 5, 6	1, 4, 6, 8	2, 5, 6, 9	1, 2, 4, 5, 8, 11	1, 3, 5, 14
2, 5, 11	3, 5, 14	2, 5, 6, 11	3, 5, 7, 14	1, 2, 4, 5, 8, 11	2, 4, 5, 6
3, 10	1, 11	2, 5, 6, 11	4, 8	3, 10	1, 11
3, 10	5, 12	3, 10	1, 11	3, 10	5, 12
3, 10	4, 13	3, 10	5, 12	3, 10	4, 13
		3, 10	4, 13	6, 7	1, 7
				6, 7	8, 9

7.4.5 Decision Making Option D3

Here we first map a pattern into a group of patterns by looking at the directions using method D2. The results are then narrowed down further by classifying the unique strokes of the patterns within a group. For this classification machine learning models for each group are required. For method D1 we had 15 different strokes that need training for the decision making method D1. For the method D3, the number of unique strokes within each group is less than 15, which will be less challenging for the machine learning models. Therefore, we expected that the algorithm performance will be better than the one in Section 7.4.3.

The groups of patterns that have the same number of strokes with the same behaviours are shown in Table 7.2. These groups are created using the data of both microphones. For the method D3, we sometimes use only one of the microphone's data as fall-back. Therefore, we need to take into account the patterns that are grouped together when using only one of the microphone's data, which are shown in Table 7.1. We analyse the patterns within each group shown in Tables 7.1 and 7.2, then filter out the common strokes of all patterns within this group and only list the unique strokes within each group. For instance, Strokes 2, 5, and 6 are unique for each pattern (Pattern 1, 4, and 8) of the first group shown in Table 7.2. Unique stroke groups to be trained for each pattern group when using both microphones, only bottom microphone, and only top microphone are shown in Table 7.3. We create machine learning models for the strokes of each group using the corresponding microphone's data. Various machine learning algorithms are applied to training data using 5-fold cross validation, and *Medium Gaussian SVM* algorithm with *kernel scale* parameter 2.2 is chosen for decision making over users' data.

D3 is an extension of D2 and therefore the rates achieved (Metric M1 and M3) are the same for D2 and D3. However, the additional processing after identifying candidate sets reduces the pattern candidate sets further (Metric M2 and M4). We therefore present next the results regarding M2 and M4.

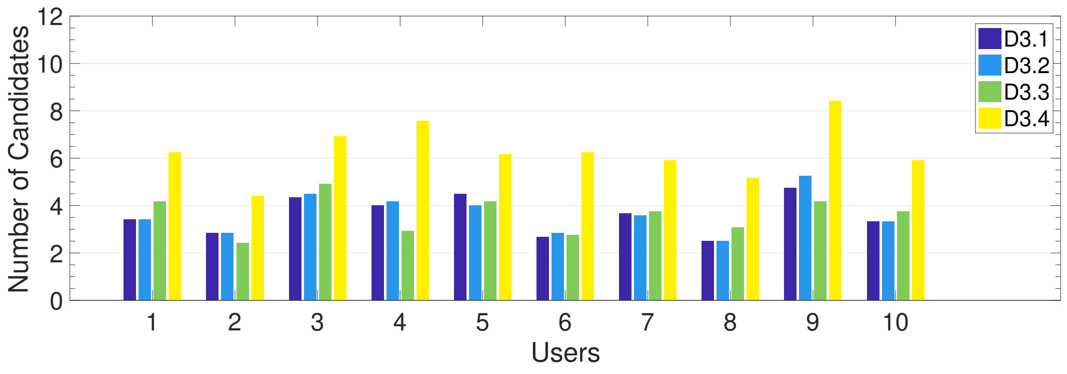


Fig. 7.16 Average number of pattern candidates remained after predictions for each user (Metric M2) with decision method D3.

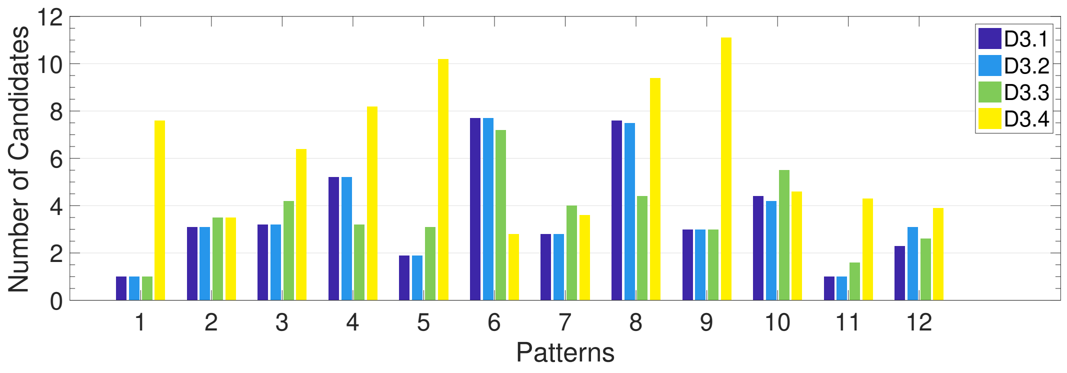


Fig. 7.17 Average number of pattern candidates remained after predictions for each pattern (Metric M4) with decision method D3.

The average number of candidates remained after predictions for each user (Metric M2) are shown in Figure 7.16. A minimum number of candidates of 2.5 is achieved for User 8. We obtain the minimum average value of 3.6 for D3.1 when looking across all users.

Figure 7.17 shows the average number of candidates remained after predictions for each pattern (Metric M4). The minimum number of candidates of 1 is achieved for Patterns 1 and 11 when using the data from both microphones (D3.1 and D3.2), which means we just need one attempt to guess these two patterns. We obtain the minimum average value of 3.6 using method D3.1.

Summary Using this method we can reduce the pattern candidate pool in some cases from 12 to 1 (Metric M4). When looking across all patterns, this method D3 improves on D2. The number of pattern candidates is reduced from 4.4 to 3.6 (Metric M2 and M4). Moreover, the average attack attempt value of 2.71 is achieved when using method D3.2 (Metric M5).

7.5 Discussions

Our experimental evaluation shows that the acoustic side-channel is in principle a useful instrument for revealing user interaction patterns on phones. However, our study and the components of SonarSnoop have limitations and improvements are possible.

7.5.1 Algorithm Performance

D1 is the most generic method, identifying individual strokes and composing these into patterns. The method helps to reduce an attackers effort in guessing the unlock pattern. However, the method does not yield very good results. The average number of candidate patterns the attacker has to try is reduced from 12 to 10.28.

D2 is much better and provides an average reduction from 12 to 4.4 patterns. This result is achieved by analysing less features from the collected sound data than method D1 (based only on direction of finger movement). However, D2 requires us to decide on a pool of patterns beforehand. This is a limitation D1 is not bound to; however, in practice this may not be problematical as the pool of likely patterns is known [32].

D3 improves on D2 by combining D1 and D2. Patterns are grouped and thereafter the method used in D1 is applied to narrow down the pattern candidate pool further. D3 provides an average reduction from 12 to 3.6 pattern (D2 reduces these from 12 to 4.4). Although D3 requires reasonably more computational effort, it gives better results than D2.

7.5.2 Limitations and Improvements

The acoustic signal generation can be improved. We believe it is possible to reduce the silence period in between pulses to achieve better resolution. The current gap size between pulses ensures reflections can be received from up to 1m distance before the next pulse is emitted. Given that we are interested in movement on the screen in close proximity we can reduce this gap. Also, different signal shapes might be possible that improve system performance.

For convenience and simplicity, we do not implement the system to cope with different users interaction speeds. We use a fixed column width of the echo profile matrix to determine if there is movement. We calibrated the system to work well with most users. However, if a user draws a pattern very slowly, the differential echo profile matrix may not reveal movements, since the rate of change is too slow to be detected. An improved implementation could support an adaptive feature to adjust with vastly different interaction speeds. Nevertheless, our method is applicable for practical scenarios since we have observed that people draw patterns consistently fast.

We proposed three decision making strategies. The algorithms sufficiently demonstrate that the active acoustic side-channel attack is feasible. However, we believe it is possible to design better decision making strategies. For example, additional features could be extracted from the recorded sound data to provide a better basis for decisions. Also, different methods for analysing the existing features (angle and direction) are possible.

SonarSnoop in its current form relies on clear separation of strokes within a pattern. When users do not pause at an inflection it is currently impossible to distinguish individual strokes. In our user study we asked users to pause at inflections. We aim to extend the system with methods for automatic separation of strokes. This can be achieved by analysing angle changes within individual CC.

Our user study has limitations. We had 10 users that were asked to draw 12 patterns 5 times. While the study provided sufficient data to analyse SonarSnoop, it would be useful to expand the data set, e.g. with a greater variety of patterns and with these entered more than 5 times by the users.

7.6 Attack Generalisation and Countermeasures

SonarSnoop can do more than stealing unlock patterns on phones. This approach can be applied to other scenarios and device types, and SonarSnoop represents a new family of security threats.

7.6.1 Attack Generalisation

SonarSnoop can be expanded to support different interactions and device types.

SonarSnoop can be extended to observe different user interactions such as gestures or typing on a phone's virtual keyboard. Recognising simple gestures (such as swipe left or right as used for Tinder) would be relatively simple to discern while identifying different key presses on a keyboard is more challenging. Adaptation to different interaction types will enable new side-channel attacks on specific applications.

Our experiment observes user interactions with a touch screen. However, SonarSnoop can be extended to observe user behaviour in some distance to the phone. For example, FingerIO has used a similar approach to observe gestures a meter away from the speaker/microphone. Thus, a phone could be used to observe user interaction with a device (e.g. an ATM) other than the phone itself.

In our study, acoustic emitter and receiver are located in the same device, and situations where these two components are separate should be considered. It is not uncommon that phones are just put aside people's laptops when they work. In this case, speakers on the phone can act as emitter while microphones on the laptop can work as receiver, or vice versa. Similarly, devices do not need to be limited only to phones and laptops. Any devices with microphones and speakers such as tablets and phones, smart watches, cameras or voice assistants are candidates.

7.6.2 New Attack Scenarios

We envisage a number of new attack scenarios that extend our experiment.

7.6.2-1 Stealing personal preferences: Tinder, the popular social search App, helps strangers to socialise with each other. It supports a filter mechanism that two people can only start chatting if they both like each other's profile picture. Tinder treats a user's 'right swipe' actions as like, and 'left swipe' as dislike. These swipe actions can be easily differentiated by SonarSnoop.

More and more human gestures are incorporated into the so-called natural user interaction with various computing devices. Our Tinder attack suggests numerous new possibilities for stealing people's sensitive personal preferences via spying on their gestures.

7.6.2-2 Combo attacks: SonarSnoop can be extended to use additional sensor inputs to boost performance. The combination of multiple sensing inputs has been used successfully in the past. For instance, Simon et al. make use of the front phone camera and microphone recording to infer PINs [124]. They use the front camera to record a video when people input, by tapping on the screen, PINs. The recorded acoustic signal helps to identify frames in which a PIN is entered. Machine learning is used to identify the pressed number in the identified frames. Narain et al. combine a gyroscope and microphone to unveil PINs [95]. Sound and gyroscopes data is used to detect finger tap location on the virtual keyboard or PIN pad.

SonarSnoop can be augmented similarly. For example, data from sensors such as gyroscopes, accelerometers or cameras could be combined with the active sonar approach. It is also possible to use a combination of approaches based on the acoustic channel. Specifically, active and passive approaches can be combined. If passive and our active acoustic side-channel analyses are combined, tapping information (timing and location) and finger movement information (movement distance and movement direction between taps) can be extracted. Such more fine grained data collection will allow us to infer user interaction with greater detail.

7.6.2-3 Espionage: Installing hidden acoustic bugs say in an embassy has been a common practice in the intelligence community. This old-fashioned eavesdropping method, when combined with SonarSnoop, will have new advantages. First, the combined use turns a passive eavesdropping into an active one. Second, cautious people know the necessity of playing loud music or otherwise introducing background noise to mitigate the eavesdropping bugs. However, this common countermeasure does little to defend against SonarSnoop, since it is robust to ambient noise.

7.6.3 Countermeasures

The main feature that enable SonarSnoop is the transmission and reception of inaudible sound. Different hardware and software solutions are possible to interfere with this feature and to prevent an acoustic active side-channel attack.

7.6.3-1 Sound System Design: Devices could be constructed such that transmission of inaudible signals is simply impossible. After all, the intended purpose of a speaker system is to communicate with people who should be able to hear the sound. Supporting the very high frequency range might be useful for high-quality sound systems (e.g. concert halls) but is perhaps unnecessary for simple appliances (e.g. phones or smart TVs). The frequency range that the hardware supports can be restricted to mitigate the threat of SonarSnoop, but this is not viable for already existing systems.

7.6.3-2 Sound Notification: Software or hardware can be used to notify users of a present sound signal in the high frequency range. Users can be alerted by an LED or by a pop-up notification. This can enable users to realise an active side-channel's presence.

7.6.3-3 Jamming: Another option is to actively disable side channels. Jamming can actively render side channels useless to an attacker. For example, Nandakumar et al. proposed to jam acoustic signals in the inaudible range [94]. A device can be designed to monitor acoustic channel activities and, once a threat situation is detected, enable jamming. Alternatively, application software can actively generate noise within the acoustic channel when sensitive tasks are executed (e.g. when a banking App requests a PIN).

7.6.3-4 Sound System Off Switch: A sound system (or either microphones or speakers individually) could be disabled during sensitive operations. Either the device provides features that allows software to disable the sound system when needed or a method is provided that allows the user to disable it. For example, a device could provide a switch (the equivalent to a mechanical camera cover) to enable users to control the capability of the device.

Among these countermeasures, no single method fits in all situations. Probably a standalone appliance which can jam in the inaudible frequency range has a best defence capability. However, this approach might not be very user friendly as people need carry an extra device with them.

7.6.4 Wider Impact

A core attacker activity is to study user interaction with systems. The simplest approach here is to follow a victim and observe their actions, for example, to observe a victim entering a PIN code at an ATM. However, people are quite aware that this might happen and take precautions. An attacker therefore may use a more covert approach and may use a camera for observation. Either a camera is deployed for this purpose or an existing camera is re-purposed for this task. For example, the attacker places a camera on the ATM or uses existing CCTV equipment. However, people have also become aware of this attacker approach and are cautious. It is common practice to cover camera lenses on a laptop with a sticker and to be aware of cameras when performing sensitive tasks.

Most devices, including numerous IoT systems have nowadays a high-end acoustic system. Phones, smart TVs, voice assistants such as Alexa and Google Home have multiple high-end speakers and microphones incorporated. As our study has demonstrated, it is possible to use these systems to gather very detailed information on user behaviour. The information is not yet as detailed as what is possible with optical systems but sufficient to obtain very detailed behaviour profiles. Users are not aware of the capability of sound systems. You would not consider that the presence of a sound system is problematic when carrying out sensitive tasks. People may be wary that conversations are recorded but they certainly lack awareness that the sound system can be used for observation of movements.

Clearly, this type of threat should be considered. People need to be made aware and the threat should be considered when designing systems.

7.7 Chapter Conclusion

We have developed a novel acoustic side-channel attack. Unlike the prior approaches where acoustic signals are passively created by a victim user or computer, we re-purpose a computer device's acoustic system into a sonar system. Thereby, an attacker actively beams human inaudible acoustic signals into an environment. The echo stream received not only allows the attacker to stealthily observe a user's behaviour, but also creates a side-channel that leaks her security secrets.

With this active acoustic side-channel, our attack could significantly reduce the number of trials required to successfully guess a victim's unlock pattern on an Android phone. We have noted that attackers do not have to limit themselves to use only smartphones. Instead, our attack appears to be applicable in any environment where microphones and speakers can interact in a way that is similar to our experimental setting.

Thus, our work starts a new line of inquiry, with fertile grounds for future research. For example, it is interesting to investigate and qualify the effectiveness of our attack in different scenarios, and to explore the best countermeasures for each of the scenarios. We also expect our work to inspire novel attacks in the future.

While it helps to improve user experience by tracking human movements and gestures via sound waves or the like, this approach can have a significant security consequence. Unfortunately, this lesson had been largely ignored in previous research for long. Because of the growing popularity of these invisible 'sensing' technologies, the lesson we have learned here is significant.

Chapter 8

Conclusion and Future Work

PVAs are pervasive in people's daily life. It has rich functionality for providing user convenience, and it plays a critical role as the major interface for users to interact with computing systems (smart devices, smart home appliances or back-end cloud services). PVAs have direct access to user private and secret information, can easily affect the normal operations of smart appliances and can even influence user personal safety (e.g., PVAs being integrated into cars and NHS applying PVAs for advices). Therefore, potential PVA security threats accompanying these capabilities desire comprehensive study. Classical security challenges such as PVA hacking, being utilised as a node of a botnet, and PVA cloud breach are out of the scope of this thesis. This thesis focuses on PVA security and privacy issues related to the acoustic channel because it is the major and most important interface between users and a PVA. This thesis clearly defines what these security and privacy issues are, and contributes to improving understanding of them via proposing attack and defence studies. Contributions of the work in this thesis are summarised next, followed by a discussion of limitations and possible future work.

8.1 Contributions

Chapter 3 presents a taxonomy of PVA security and privacy challenges related to the acoustic channel and introduces the state-of-the-art research work according to this taxonomy framework. This taxonomy shown as Figure 3.1 categorise relevant study topics into four main domains: Access Control (C1), Privacy (C2), DoS (C3) and Sensing (C4), then more specific sub-domains are built so that existing studies all fit in these groupings. This taxonomy shows what the focus of existing studies is, provides clear guidance on areas which require more attention, and outlines newly emerging research topics worthy of additional research effort. The four work presented in later chapters of this thesis are clearly highlighted in the taxonomy, which illustrates how the work in this thesis contribute to PVA acoustic security and privacy. This chapter introduces relevant existing studies in each category in detail. For some areas where multiple related works exist, comparisons are made to pinpoint their overlap and differences (e.g., Adversarial Commands (C1.2.3) and Hardware Non-linearity (C1.2.1)). Summaries for each specific research domain are made to outline the current progress and to identify promising yet largely unexplored research directions. Finally, conclusions and discussion are presented from two perspectives: public perception/industry efforts and research directions. This chapter is a research map which improves the understanding of the current research status of acoustic-channel PVA security and privacy. It is a guide showing which areas require more study, and it shows how the work in this thesis contribute to these areas.

Chapter 4 proposes a novel defence method against adversarial attacks targeting ASR as introduced in Section 2.4.3. As pointed out in Chapter 3, there are extensive studies on adversarial commands targeting ASRs, but defence studies on detecting these attacks or on protecting the ASR from being manipulated barely exist. The proposed defence method utilises a parallel second ASR working together with the major one to detect potential adversarial commands. The commands are classed as attack if the decoding difference between these two ASR is beyond a threshold. The assumption behind this methodology is that it is infeasible for an attacker to craft an adversarial command which is recognised by two different ASR. Practical experiments were carried out to show the feasibility and efficacy of this defence method.

Chapter 5 proposes a privacy control method for PVA users to embed additional information together with their voice in the case when their voice is recorded by nearby PVAs. Due to the popularity of PVAs and its various forms as smart devices, it is common that we are always in the range of a PVA. People usually have little control over these PVAs recording their voice if they are activated. Although possible privacy violation is the prominent concern for users in regarding of PVA usage, there are not enough studies focusing on this area as pointed out in Chapter 3. The proposed acoustic tagging is a novel method which gives users a mechanism to control their private data. If the back-end PVA service provider is cooperative, this tag can signal that the voice owner does not give a recording consent and the corresponding recording will not be kept and analysed. This tag can also be designed to represent multi-bit information identifying when and where a recording has taken place to enable recording tracing. This chapter discusses different tagging techniques and application scenarios, then it shows a tagging prototype device based on PocketSphinx. The demonstration shows the feasibility of tagging spoken voice commands recorded by a

Google Home Mini device. This tag can be easily retrieved from conversations stored in the back-end Google online service.

Chapter 6 presents a DoS attack called reactive jamming to prevent a PVA from being activated and recording people's voice unintentionally. DoS is a classic research topic in the cybersecurity area, however, there are barely studies looking at PVA scenarios from a DoS perspective. This chapter creatively applies a DoS attack to protect users from pervasive PVA recording. Even though some PVAs (e.g., Google Home Mini) provide a switch to disable their microphones, users may not be aware of it, may have forgotten to switch the microphone off, and they may not be able to control the PVAs owned by others in their surroundings. The reactive jamming method enables user to disable activation of nearby PVAs. To our best knowledge, this is the first work investigating reactive jamming towards PVAs. The reactive acoustic jamming for PVAs utilises a PJD to observe conversations (this is similar to the always listening feature as a PVA). Upon detecting the same wake word as the PVA, the PJD emits an acoustic jamming signal. The PJD must recognise the wake word faster than the PVA such that the jamming signal prevents the wake word detection on the PVA. This chapter describes necessary work towards a mature and robust reactive acoustic jamming solution on PVAs. This chapter presents an evaluation of the effectiveness of various jamming signals. The impact of jamming signal, wake word overlap on jamming success, and jamming false positive rate in dependence of the overlap are all evaluated quantitatively. The evaluation shows 100% successful jamming is assured with an overlap of at least 60% with a negligible false positive rate. The work in this chapter shows that reactive jamming on PVAs is a feasible approach (without creating a system perceived as a noise nuisance).

Chapter 7 proposes the first active acoustic side-channel attack on a smartphone. This attack utilises speakers on the smartphone to emit human inaudible pre-defined sound waves, and echoes are recorded via smartphone microphones. The echo signal can be profiled to infer user interaction with the device. In this way, the smartphone or other smart devices (i.e., PVAs) with necessary acoustic components is turned into a sonar system. This system is named as SonarSnoop in this chapter, and the proposed attack reveals a new family of security threats emerging due to sophisticated active sensing a PVA can accomplish. In Chapter 7, an empirical study of stealing Android unlock patterns via inferring victim's finger movements is presented to exemplify the proposed active acoustic attack. In our experiment, the number of candidate unlock patterns that an attacker must try to authenticate herself to a Samsung S4 phone can be reduced by up to 70% using this novel acoustic side-channel. The attack is entirely unnoticeable to victims. A detailed discussion is included in this chapter about potential other application scenarios of this type of attack. The work in this chapter has attracted attention from technology related media such as Motherboard, ZDNet and Naked Security of Sophos. It was a finalist for the Pwnie Awards [17] for the most innovative research in 2019.

8.2 Limitations and Future Work

In this section I discuss some limitations of the work presented in each chapter and outline possible future work for each of the key contributions presented in this thesis.

Chapter 4 tested the feasibility of the proposed adversarial commands defence framework with 20 adversarial examples and 20 normal human spoken commands on 4 sets of main ASR and protection ASR pairs. Compared to the main ASR (Kaldi nnet2), the choices of the protection one includes both more advanced one with more sophisticated architecture (Kaldi nnet3), less advanced one (Kaldi GMM-HMM) and another GMM-HMM PocketSphinx designed by a different team (Carnegie Mellon University). I would like to continue this line of research by implementing more comprehensive experiments. Specifically, I would use larger amount of samples for both the adversarial examples and benign human spoken commands; I followed the study from Schönherr et al [118] to generate adversarial examples in this chapter and used WSJ dataset for the training process. I would like to test adversarial examples generated in other fashion, applying larger scale training data, and incorporating more experiment subjects (more types of main and protection ASR) to verify the proposed defence method more comprehensively). From another research perspective other than defence, the work in Access Control (C1) are mostly on Hidden Voice Commands (C1.2), especially Adversarial Commands (C1.2.3). All of these work generate adversarial commands in an iterative optimisation way, but I would like to try if it is possible to add perturbations to original speech commands on the fly to force an ASR to transcribe these commands to a specified target text.

Chapter 5 proposes acoustic tagging as a consent management method. Our prototype demonstrates the feasibility of the tagging approach. I would like to advance to implement inaudible acoustic tagging, and to design the tagging protocol for the scenario where there are multiple tagging devices for multiple users. The acoustic tag I have implemented in the chapter only contains one-bit information on recording dissent, and expanding the tag to include more information referring when and where the conversation happened is the next step. Technologies in acoustic watermarking area could be used for this improvement.

Chapter 6 proposes the reactive jamming method rather than constant jamming to prevent a PVA from being activated. This is the first step towards a complete PVA reactive jamming solution. Future work will carry out a practical evaluation with COTS PVAs and play the jamming signal over the air instead of direct feeding it to a PVA. I would like to advance reactive jamming to be more sophisticated by inducing inaudible jamming signal. Although inaudible jamming signal is in a very high frequency range which cannot survive the low pass filter of the PVA, the non-linearity feature could be exploited (as described in Section 2.4.1) to make sure the jamming signal is recorded and overlaps with the wake word speech segment. Furthermore, I would like to try if it is possible to jam via adding inference very targeted to introduce ASR transcription errors.

Chapter 7 exemplifies the proposed acoustic side-channel attack by revealing Android unlock patterns via inferring finger movements, but it is only one application scenario out of a new family of threats potentially posed by SonarSnoop. I would like to apply SonarSnoop methodology to inspect other user interactions with smart devices as described in Section 7.6.1. For instance, I would like to observe user behaviours in some distance to the smartphone (e.g., inferring user interaction with an ATM using SonarSnoop on a smartphone). Another interesting scenario desire exploring is where sound wave emitter and receiver are not on the same device. For instance, I would like to apply SonarSnoop in the situation that a user is working on her laptop keyboard while her phone is placed besides the laptop. This is a common scenario, and I could use speakers of the phone to emit inaudible sound waves, and use microphones on the laptop as the receiver. Note that devices do not need to be limited as phones or laptops. Any smart device with speakers and microphones are fit for SonarSnoop. PVAs are perfect candidate for SonarSnoop and I look forward to apply SonarSnoop on a PVA (e.g., a smartphone, a smart speaker etc.) for long-distance (or even through the wall) user complex behaviour (not only single-bit status such as standing still, moving or acting repetitively) observation.

References

[1] (2009). Kaldi. http://kaldi-asr.org. Online; accessed 3rd July, 2020.

[2] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I. J., Harp, A., Irving, G., Isard, M., Jia, Y., Józefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D. G., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P. A., Vanhoucke, V., Vasudevan, V., Viégas, F. B., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467.

[3] Abdel-Hamid, O., Mohamed, A., Jiang, H., Deng, L., Penn, G., and Yu, D. (2014). Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(10):1533–1545.

[4] Abdi, N., Ramokapane, K. M., and Such, J. M. (2019). More than smart speakers: Security and privacy perceptions of smart home personal assistants. In *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*, Santa Clara, CA. USENIX Association.

[5] Abdullah, H., Garcia, W., Peeters, C., Traynor, P., Butler, K. R. B., and Wilson, J. (2019a). Practical hidden voice attacks against speech and speaker recognition systems. In *Proceedings of the 2019 Network and Distributed System Security Symposium (NDSS '19)*.

[6] Abdullah, H., Rahman, M. S., Garcia, W., Blue, L., Warren, K., Yadav, A. S., Shrimpton, T., and Traynor, P. (2019b). Hear" no evil", see" kenansville": Efficient and transferable black-box attacks on speech recognition and voice identification systems. *arXiv preprint arXiv:1910.05262*.

[7] AI, G. (2019). An All-Neural On-Device Speech Recognizer. https://ai.googleblog.com/2019/03/an-all-neural-on-device-speech.html. Online; accessed 10th Oct, 2019.

[8] Alegre, F., Vipperla, R., Evans, N., and Fauve, B. (2012). On the vulnerability of automatic speaker recognition to spoofing attacks with artificial signals. In *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pages 36–40.

[9] AlexaPi (2016). Alexa client for all your devices! https://github.com/alexa-pi/AlexaPi.

[10] Allen, J. B. and Berkley, D. A. (1979). Image method for efficiently simulating small-room acoustics. *The Journal of the Acoustical Society of America*, 65(4):943–950.

[11] Alzantot, M., Balaji, B., and Srivastava, M. B. (2018). Did you hear that? adversarial examples against automatic speech recognition. *CoRR*, abs/1801.00554.

[12] Amin, T. B., German, J. S., and Marziliano, P. (2013). Detecting voice disguise from speech variability: Analysis of three glottal and vocal tract measures. In *Proceedings of Meetings on Acoustics 166ASA*, volume 20, page 060005. ASA.

[13] Amin, T. B., Marziliano, P., and German, J. S. (2014). Glottal and vocal tract characteristics of voice impersonators. *IEEE Transactions on Multimedia*, 16(3):668–678.

[14] Apple (2018). Personalized Hey Siri - Apple. In *Apple Machine Learning Journal*, volume 1. Apple.

[15] Aviv, A. J., Gibson, K., Mossop, E., Blaze, M., and Smith, J. M. (2010). Smudge Attacks on Smartphone Touch Screens. In *Proc. Usenix WOOT'10*.

[16] Aviv, A. J., Sapp, B., Blaze, M., and Smith, J. M. (2012). Practicality of Accelerometer Side Channels on Smartphones. In *Proc. ACSAC'12*.

[17] Awards, P. (2019). the Pwnie Awards. https://pwnies.com/nominations/. Online; accessed 7th Sep, 2020.

[18] Bloomberg (10-Apr-2019). Is anyone listening to you on Alexa? a global team reviews audio. https://www.bloomberg.com/news/articles/2019-04-10/is-anyone-listening-to-you-on-alexa-a-global-team-reviews-audio. Online; accessed 3rd Aug, 2019.

[19] Blue, L., Vargas, L., and Traynor, P. (2018). Hello, is it me you're looking for?: Differentiating between human and electronic speakers for voice interface security. In *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, WiSec '18, pages 123–133, New York, NY, USA. ACM.

[20] Brown, J., Bagci, I., King, A., and Roedig, U. (2013). *Defend Your Home! Jamming Unsolicited Messages in the Smart Home*, pages 1–6. ACM Press.

[21] Cai, C., Zheng, R., and Hu, M. (2019). A survey on acoustic sensing. *CoRR*, abs/1901.03450.

[22] California Legislature (2019). Information privacy: other connected device with a voice recognition feature. "AB-1395".

[23] Carlini, N., Mishra, P., Vaidya, T., Zhang, Y., Sherr, M., Shields, C., Wagner, D., and Zhou, W. (2016). Hidden voice commands. In *Proceedings of the 25th USENIX Security Symposium (USENIX Security'16)*, pages 513–530, Austin, TX. USENIX Association.

[24] Carlini, N. and Wagner, D. (2018). Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 1–7.

[25] Champion, C., Olade, I., Papangelis, K., Liang, H., and Fleming, C. (2019). The smart^2 speaker blocker: An open-source privacy filter for connected home speakers. *CoRR*, abs/1901.04879.

[26] Chen, G., Parada, C., and Heigold, G. (2014). Small-footprint keyword spotting using deep neural networks. In *ICASSP*, volume 14, pages 4087–4091. Citeseer.

[27] Chen, S., Ren, K., Piao, S., Wang, C., Wang, Q., Weng, J., Su, L., and Mohaisen, A. (2017). You can hear but you cannot steal: Defending against voice impersonation attacks on smartphones. In *Proceedings of the 2017 37th IEEE International Conference on Distributed Computing Systems (ICDCS '17)*, pages 183–195. IEEE.

[28] Chen, T., Shangguan, L., Li, Z., and Jamieson, K. (2020). Metamorph: Injecting inaudible commands into over-the-air voice controlled systems. In *Proc. NDSS'20*.

[29] Cheng, P., Bagci, I., Yan, J., and Roedig, U. (2019). Smart speaker privacy control - acoustic tagging for personal voice assistants. In *Proceedings of the IEEE Workshop on the Internet of Safe Things (SafeThings '19)*. IEEE.

[30] Cheng, P., Bagci, I. E., Roedig, U., and Yan, J. (2018a). Sonarsnoop: Active acoustic side-channel attacks. *CoRR*, abs/1808.10250. Later accepted by International Journal of Information Security, Jul 2019. Available: https://doi.org/10.1007/s10207-019-00449-8.

[31] Cheng, P., Bagci, I. E., Yan, J., and Roedig, U. (2018b). Towards reactive acoustic jamming for personal voice assistants. In *Proceedings of the 2nd ACM International Workshop on Multimedia Privacy and Security (MPS '18)*, pages 12–17, New York, NY, USA. ACM.

[32] Cho, G., Huh, J. H., Cho, J., Oh, S., Song, Y., and Kim, H. (2017). SysPal: System-Guided Pattern Locks for Android. In *Proc. IEEE Symposium on S&P'17*.

[33] CMUSphinx (2006). Basic concepts of speech recognition. https://cmusphinx.github.io/wiki/tutorialconcepts/. Online; accessed 19th July, 2020.

[34] CMUSphinx (2017). Deep Neural Networks in PocketSphinx. https://cmusphinx.github.io/2017/05/gsoc-2017-accepted-projects-announced/. Online; accessed 22nd Aug, 2020.

[35] Consortium, A. (2019). Asvspoof 2019: Automatic speaker verification spoofing and countermeasures challenge evaluation plan. https://www.asvspoof.org/asvspoof2019/asvspoof2019_evaluation_plan.pdf. Online; accessed 28th Jan, 2020.

[36] Dahl, G. E., Yu, D., Deng, L., and Acero, A. (2012a). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42.

[37] Dahl, G. E., Yu, D., Deng, L., and Acero, A. (2012b). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42.

[38] De Leon, P. L., Pucher, M., Yamagishi, J., Hernaez, I., and Saratxaga, I. (2012). Evaluation of speaker verification security and detection of hmm-based synthetic speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(8):2280–2290.

[39] De Leon, P. L., Pucher, M., Yamagishi, J., Hernaez, I., and Saratxaga, I. (2012a). Evaluation of speaker verification security and detection of hmm-based synthetic speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(8):2280–2290.

[40] De Leon, P. L., Pucher, M., Yamagishi, J., Hernaez, I., and Saratxaga, I. (2012b). Evaluation of speaker verification security and detection of hmm-based synthetic speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(8):2280–2290.

[41] Delgado, H., Todisco, M., Sahidullah, M., Evans, N., Kinnunen, T., Lee, K., and Yamagishi, J. (2018). Asvspoof 2017 version 2.0: meta-data analysis and baseline enhancements.

[42] Diao, W., Liu, X., Zhou, Z., and Zhang, K. (2014). Your voice assistant is mine: How to abuse speakers to steal information and control your phone. In *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices (SPSM '14)*, pages 63–74, New York, NY, USA. ACM.

[43] Diapoulis, G., Kropp, W., and Rosas Pérez, C. (2018). Person identification from walking sound on wooden floor.

[44] Euronews (11-Jul-2019). Experts raise security concerns surrounding Amazon Alexa and NHS partnership. https://www.euronews.com/2019/07/10/amazon-alexa-to-provide-health-advice-after-teaming-up-with-the-nhs. Online; accessed 3rd Aug, 2019.

[45] Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., and Wagner, D. (2012). Android Permissions: User Attention, Comprehension, and Behavior. In *Proc. SOUPS'12*.

[46] Feng, H., Fawaz, K., and Shin, K. G. (2017). Continuous authentication for voice assistants. In *Proceedings of the 23rd ACM Annual International Conference on Mobile Computing and Networking (MobiCom '17)*, pages 343–355, New York, NY, USA. ACM.

[47] Festival (2004). The Festival Speech Synthesis System (version 2.4). http://www.cstr.ed.ac.uk/projects/festival/.

[48] Gaida, C., Lange, P., Petrick, R., Proba, P., Malatawy, A., and Suendermann-Oeft, D. (2014). Comparing open-source speech recognition toolkits. *Tech. Rep., DHBW Stuttgart*.

[49] Gao, C., Chandrasekaran, V., Fawaz, K., and Banerjee, S. (2018). Traversing the quagmire that is privacy in your smart home. In *Proceedings of the 2018 Workshop on IoT Security and Privacy*, IoT S&#38;P '18, pages 22–28, New York, NY, USA. ACM.

[50] Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G., and Pallett, D. S. (1988). Getting started with the darpa timit cd-rom: An acoustic phonetic continuous speech database. *National Institute of Standards and Technology (NIST), Gaithersburgh, MD*, 107:16.

[51] Genkin, D., Pattani, M., Schuster, R., and Tromer, E. (2018). Synesthesia: Detecting screen content via remote acoustic side channels.

[52] Genkin, D., Shamir, A., and Tromer, E. (2014). RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis. In *International Cryptology Conference*, pages 444–461. Springer.

[53] Gong, Y. and Poellabauer, C. (2017). Crafting adversarial examples for speech paralinguistics applications. *CoRR*, abs/1711.03280.

[54] Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples.

[55] Graves, A., Mohamed, A., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649.

[56] Gruhn, R. E., Minker, W., and Nakamura, S. (2011). *Statistical pronunciation modeling for non-native speech processing*. Springer Science & Business Media.

[57] Hannun, A. Y., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., and Ng, A. Y. (2014). Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567.

[58] Hautamäki, R. G., Kinnunen, T., Hautamäki, V., Leino, T., and Laukkanen, A.-M. (2013). I-vectors meet imitators: on vulnerability of speaker verification systems against voice mimicry. In *Interspeech*, pages 930–934.

[59] Hautamäki, R. G., Kinnunen, T., Hautamäki, V., and Laukkanen, A.-M. (2015). Automatic versus human speaker verification: The case of voice mimicry. *Speech Communication*, 72:13 – 31.

[60] He, Y., Bian, J., Tong, X., Qian, Z., Zhu, W., Tian, X., and Wang, X. (2019). Canceling inaudible voice commands against voice control systems. In *The 25th Annual International Conference on Mobile Computing and Networking*, MobiCom '19, New York, NY, USA. Association for Computing Machinery.

[61] Help, G. A. (2018). Link your voice to your google assistant device with voice match. https://support.google.com/assistant/answer/9071681. Online; accessed 25th Jun, 2020.

[62] Help, G. S. (2017). Google stores your voice input. https://goo.gl/7w5We1. Online; accessed 23rd Aug, 2020.

[63] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

[64] Huang, X., Acero, A., and Hon, H.-W. (2001). *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.

[65] Huggins-Daines, D., Kumar, M., Chan, A., Black, A. W., Ravishankar, M., and Rudnicky, A. I. (2006). PocketSphinx: A free, real-time continuous speech recognition system for hand-held devices. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 1, pages I–I. IEEE.

[66] iFLYTEK (2020). iflytek. http://www.iflytek.com/en/index.html. Online; accessed 24th July, 2020.

[67] Iter, D., Huang, J., and Jermann, M. (2017). Generating adversarial examples for speech recognition. *Stanford Technical Report.*

[68] Karmann, B. and Knudsen, T. (30-Jun-2019). Project Alias. https://github.com/bjoernkarmann/$project_alias$. Online; accessed 5th Aug, 2019.

[69] Khare, S., Aralikatte, R., and Mani, S. (2019). Adversarial black-box attacks on automatic speech recognition systems using multi-objective evolutionary optimization. *Proc. Interspeech 2019*, pages 3208–3212.

[70] Kim, N. and Park, K. (2016). Speech-to-text-wavenet. https://github.com/buriburisuri/speech-to-text-wavenet. Online; accessed 2nd July, 2020.

[71] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

[72] Kinnunen, T., Sahidullah, M., Delgado, H., Todisco, M., Evans, N., Yamagishi, J., and Lee, K. A. (2017). The ASVspoof 2017 challenge: Assessing the limits of replay spoofing attack detection. In *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH '17)*, Stockholm, Sweden.

[73] Kinnunen, T., Sahidullah, M., Kukanov, I., Delgado, H., Todisco, M., Sarkar, A., Thomsen, N. B., Hautamäki, V., Evans, N., and Tan, Z.-H. (2016). Utterance verification for text-dependent speaker recognition: a comparative assessment using the reddots corpus.

[74] Kinnunen, T., Wu, Z., Lee, K. A., Sedlak, F., Chng, E. S., and Li, H. (2012). Vulnerability of speaker verification systems against voice conversion spoofing attacks: The case of telephone speech. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4401–4404.

[75] Kreuk, F., Adi, Y., Cisse, M., and Keshet, J. (2018). Fooling end-to-end speaker verification with adversarial examples. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1962–1966.

[76] Kumar, D., Paccagnella, R., Murley, P., Hennenfent, E., Mason, J., Bates, A., and Bailey, M. (2018). Skill squatting attacks on amazon alexa. In *Proceedings of the 27th USENIX Security Symposium (USENIX Security '18)*, pages 33–47, Baltimore, MD. USENIX Association.

[77] Lange, P. and Suendermann-Oeft, D. (2014). Tuning Sphinx to outperform Google's speech recognition API. In *Proc. of the ESSV 2014, Conference on Electronic Speech Signal Processing*, pages 1–10.

[78] Lau, J., Zimmerman, B., and Schaub, F. (2018). Alexa, are you listening?: Privacy perceptions, concerns and privacy-seeking behaviors with smart speakers. *Proc. ACM Hum.-Comput. Interact.*, 2(CSCW):102:1–102:31.

[79] Lee, Y., Zhao, Y., Zeng, J., Lee, K., Zhang, N., Shezan, F. H., Tian, Y., Chen, K., and Wang, X. (2020). Using sonar for liveness detection to protect smart speakers against remote attackers. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 4(1).

[80] Lei, X., Tu, G., Liu, A. X., Li, C., and Xie, T. (2017). The insecurity of home digital voice assistants - amazon alexa as a case study. *CoRR*, abs/1712.03327.

[81] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

[82] Lindberg, J. and Blomberg, M. (1999). Vulnerability in speaker verification-a study of technical impostor techniques. In *Sixth European Conference on Speech Communication and Technology*.

[83] Liopa (12-Jul-2019). Liveness detection. https://liopa.ai/liveness-detection. Online; accessed 3rd Aug, 2019.

[84] Liu, J., Wang, Y., Kar, G., Chen, Y., Yang, J., and Gruteser, M. (2015). Snooping Keystrokes with mm-level Audio Ranging on a Single Phone. In *Proc. MobiCom'15*.

[85] Love, R., Dembry, C., Hardie, A., Brezina, V., and McEnery, T. (2017). The Spoken BNC2014. *International Journal of Corpus Linguistics*, 22(3):319–344.

[86] Malekesmaeili, M. and Ward, R. K. (2014). A local fingerprinting approach for audio copy detection. *Signal Process.*, 98:308–321.

[87] Martinovic, I., Pichota, P., and Schmitt, J. B. (2009). Jamming for Good: A Fresh Approach to Authentic Communication in WSNs. In *Proceedings of the Second ACM Conference on Wireless Network Security*, WiSec '09, pages 161–168, New York, NY, USA. ACM.

[88] McLoughlin, I. (2009). *Applied Speech and Audio Processing: With Matlab Examples*. Cambridge University Press, New York, NY, USA.

[89] McLoughlin, I. V. (2016). *Speech and Audio Processing: A MATLAB-based Approach*. Cambridge University Press.

[90] Mohamed, A. (2014). *Deep Neural Network Acoustic Models for ASR*. PhD thesis, University of Toronto.

[91] Mukhopadhyay, D., Shirvanian, M., and Saxena, N. (2015). All your voices are belong to us: Stealing voices to fool humans and machines. In Pernul, G., Y A Ryan, P., and Weippl, E., editors, *Computer Security – ESORICS 2015*, pages 599–621, Cham. Springer International Publishing.

[92] Naika, R. (2018). An overview of automatic speaker verification system. In Bhalla, S., Bhateja, V., Chandavale, A. A., Hiwale, A. S., and Satapathy, S. C., editors, *Intelligent Computing and Information and Communication*, pages 603–610, Singapore. Springer Singapore.

[93] Nandakumar, R., Iyer, V., Tan, D., and Gollakota, S. (2016). FingerIO: Using active sonar for fine-grained finger tracking. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*, pages 1515–1525, New York, NY, USA. ACM.

[94] Nandakumar, R., Takakuwa, A., Kohno, T., and Gollakota, S. (2017). CovertBand: Activity Information Leakage Using Music. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(3):87:1–87:24.

[95] Narain, S., Sanatinia, A., and Noubir, G. (2014). Single-stroke Language-agnostic Keylogging Using Stereo-microphones and Domain Specific Machine Learning. In *Proc. WiSec'14*.

[96] News, B. (03-Jul-2018). Gavin williamson interrupted by siri during commons statement. https://www.bbc.com/news/av/uk-politics-44701007/gavin-williamson-interrupted-by-siri-during-commons-statement. Online; accessed 3rd Aug, 2019.

[97] NPR (2018). The smart audio report. https://www.nationalpublicmedia.com. Online; accessed 3rd Aug, 2019.

[98] Olson, C. and Kemery, K. (2019). 2019 Voice report: Consumer adoption of voice technology and digital assistants.

[99] Ouali, C., Dumouchel, P., and Gupta, V. (2014). A robust audio fingerprinting method for content-based copy detection. In *2014 12th International Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 1–6.

[100] Panchapagesan, S., Sun, M., Khare, A., Matsoukas, S., Mandal, A., Hoffmeister, B., and Vitaladevuni, S. (2016). Multi-Task Learning and Weighted Cross-Entropy for DNN-Based Keyword Spotting. In *INTERSPEECH*, pages 760–764.

[101] Panjwani, S. and Prakash, A. (2014). Crowdsourcing attacks on biometric systems. In *Proceedings of the Tenth USENIX Conference on Usable Privacy and Security*, SOUPS '14, pages 257–269, Berkeley, CA, USA. USENIX Association.

[102] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., et al. (2011). The Kaldi speech recognition toolkit. In *Proceedings of the 2011 IEEE workshop on automatic speech recognition and understanding*, number CONF. IEEE Signal Processing Society.

[103] Pradhan, S., Sun, W., Baig, G., and Qiu, L. (2019). Combating replay attacks against voice assistants. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 3(3):100:1–100:26.

[104] Projects, Z. (2017). Zamia Speech. https://github.com/gooofy/zamia-speech#asr-models. Online; accessed 26th July, 2020.

[105] Pu, Q., Gupta, S., Gollakota, S., and Patel, S. (2013). Whole-home Gesture Recognition Using Wireless Signals. In *Proc. MobiCom'13*.

[106] Qi Li, Biing-Hwang Juang, and Chin-Hui Lee (2000). Automatic verbal information verification for user authentication. *IEEE Transactions on Speech and Audio Processing*, 8(5):585–596.

[107] Qian, J., Du, H., Hou, J., Chen, L., Jung, T., and Li, X.-Y. (2018). Hidebehind: Enjoy voice input with voiceprint unclonability and anonymity. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems (SenSys '18)*, pages 82–94, New York, NY, USA. ACM.

[108] Qin, Y., Carlini, N., Goodfellow, I., Cottrell, G., and Raffel, C. (2019). Imperceptible, Robust, and Targeted Adversarial Examples for Automatic Speech Recognition. *arXiv e-prints*, page arXiv:1903.10346.

[109] Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

[110] Rahman, T., Adams, A. T., Zhang, M., Cherry, E., Zhou, B., Peng, H., and Choudhury, T. (2014). BodyBeat: A mobile system for sensing non-speech body sounds. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '14)*, pages 2–13, New York, NY, USA. ACM.

[111] Roy, N., Hassanieh, H., and Roy Choudhury, R. (2017). BackDoor: Making microphones hear inaudible sounds. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '17)*, pages 2–14, New York, NY, USA. ACM.

[112] Roy, N., Shen, S., Hassanieh, H., and Choudhury, R. R. (2018). Inaudible voice commands: The long-range attack and defense. In *Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI '18)*, pages 547–560, Renton, WA. USENIX Association.

[113] Ruhr-Universität Bochum (2018). Adversarial Attacks. https://github.com/rub-ksv/adversarialattacks. Online; accessed 26th July, 2020.

[114] Sainath, T. N. and Parada, C. (2015). Convolutional neural networks for small-footprint keyword spotting. In *Sixteenth Annual Conference of the International Speech Communication Association*.

[115] Sak, H., Senior, A. W., Rao, K., and Beaufays, F. (2015). Fast and accurate recurrent neural network acoustic models for speech recognition. *CoRR*, abs/1507.06947.

[116] Schinkel-Bielefeld, N., Lotze, N., and Nagel, F. (2013). Audio quality evaluation by experienced and inexperienced listeners. In *Proceedings of Meetings on Acoustics ICA2013*, volume 19, page 060016. Acoustical Society of America.

[117] Schlegel, R., Zhang, K., Zhou, X.-y., Intwala, M., Kapadia, A., and Wang, X. (2011). Soundcomber: A stealthy and context-aware sound trojan for smartphones. In *Proc. NDSS'11*, volume 11, pages 17–33.

[118] Schönherr, L., Kohls, K., Zeiler, S., Holz, T., and Kolossa, D. (2019). Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. In *Proceedings of the 2019 Network and Distributed System Security Symposium (NDSS '19)*.

[119] Schweppe, D. (10-Apr-2019). Mycroft - open source voice assistant. https://mycroft.ai. Online; accessed 5th Aug, 2019.

[120] Schönherr, L., Zeiler, S., Holz, T., and Kolossa, D. (2019). Imperio: Robust over-the-air adversarial examples for automatic speech recognition systems.

[121] Shen, J., Nguyen, P., Wu, Y., Chen, Z., Chen, M. X., Jia, Y., Kannan, A., Sainath, T. N., Cao, Y., Chiu, C., He, Y., Chorowski, J., Hinsu, S., Laurenzo, S., Qin, J., Firat, O., Macherey, W., Gupta, S., Bapna, A., Zhang, S., Pang, R., Weiss, R. J., Prabhavalkar, R., Liang, Q., Jacob, B., Liang, B., Lee, H., Chelba, C., Jean, S., Li, B., Johnson, M., Anil, R., Tibrewal, R., Liu, X., Eriguchi, A., Jaitly, N., Ari, N., Cherry, C., Haghani, P., Good, O., Cheng, Y., Alvarez, R., Caswell, I., Hsu, W., Yang, Z., Wang, K., Gonina, E., Tomanek, K., Vanik, B., Wu, Z., Jones, L., Schuster, M., Huang, Y., Chen, D., Irie, K., Foster, G. F., Richardson, J., and et al. (2019). Lingvo: a modular and scalable framework for sequence-to-sequence modeling. *CoRR*, abs/1902.08295.

[122] Shen, S., Chen, D., Wei, Y.-L., Yang, Z., and Choudhury, R. R. (2020). Voice localization using nearby wall reflections. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, MobiCom '20, New York, NY, USA. Association for Computing Machinery.

[123] Shumailov, I., Simon, L., Yan, J., and Anderson, R. (2019). Hearing your touch: A new acoustic side channel on smartphones. *CoRR*, abs/1903.11137.

[124] Simon, L. and Anderson, R. (2013). PIN Skimmer: Inferring PINs Through the Camera and Microphone. In *Proc. SPSM'13*.

[125] Spreitzer, R., Moonsamy, V., Korak, T., and Mangard, S. (2018). Systematic Classification of Side-Channel Attacks: A Case Study for Mobile Devices. *IEEE Communications Surveys & Tutorials*, 20(1):465–488.

[126] Stafylakis, T., Alam, M. J., and Kenny, P. (2016). Text-dependent speaker recognition with random digit strings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(7):1194–1203.

[127] Stallings, W. (2009). *Operating systems: internals and design principles*. Upper Saddle River, NJ: Pearson/Prentice Hall,.

[128] Stylianou, Y. (2009). Voice transformation: A survey. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3585–3588.

[129] Szurley, J. and Kolter, J. Z. (2019). Perceptual based adversarial audio attacks.

[130] Taori, R., Kamsetty, A., Chu, B., and Vemuri, N. (2018). Targeted adversarial examples for black box audio systems. *CoRR*, abs/1805.07820.

[131] Turner, M. R. (1986). Texture discrimination by gabor functions. *Biological Cybernetics*, 55(2):71–82.

[132] Uellenbeck, S., Dürmuth, M., Wolf, C., and Holz, T. (2013). Quantifying the Security of Graphical Passwords: The Case of Android Unlock Patterns. In *Proc. CCS'13*.

[133] Vadillo, J. and Santana, R. (2020). On the human evaluation of audio adversarial examples.

[134] Vaidya, T., Zhang, Y., Sherr, M., and Shields, C. (2015). Cocaine noodles: Exploiting the gap between human and machine speech recognition. In *Proceedings of the 9th USENIX Workshop on Offensive Technologies (WOOT '15)*, Washington, D.C. USENIX Association.

[135] Verge, T. (11-Jul-2019). Yep, human workers are listening to recordings from google assistant, too. https://www.theverge.com/2019/7/11/20690020/google-assistant-home-human-contractors-listening-recordings-vrt-nws. Online; accessed 3rd Aug, 2019.

[136] Verge, T. (11-Oct-2017). Amazon's alexa can now recognize different voices and give personalized responses. https://www.theverge.com/circuitbreaker/2017/10/11/16460120/amazon-echo-multi-user-voice-new-feature#:~:text=Youcansetupvoice,partypartyAlexaenableddevices. Online; accessed 25th Jun, 2020.

[137] Villalba, J. and Lleida, E. (2011). Detecting replay attacks from far-field recordings on speaker verification systems. In Vielhauer, C., Dittmann, J., Drygajlo, A., Juul, N. C., and Fairhurst, M. C., editors, *Biometrics and ID Management*, pages 274–285, Berlin, Heidelberg. Springer Berlin Heidelberg.

[138] Voicebot.ai (2019). Australia smart speaker consumer adoption report 2019.

[139] voicebot.ai (2019). Nest secure's control hub has a microphone – users only found out when it became google assistant enabled this week. https://voicebot.ai/2019/02/07/nest-secures-control-hub-has-a-microphone-users-only-found-out-when-it-became-google-assistant-enabled-this-week/. Online; accessed 16th Sep, 2019.

[140] Wang, C., Anand, S. A., Liu, J., Walker, P., Chen, Y., and Saxena, N. (2019). Defeating hidden audio channel attacks on voice assistants via audio-induced surface vibrations. In *Proceedings of the 35th Annual Computer Security Applications Conference*, ACSAC '19, pages 42–56, New York, NY, USA. ACM.

[141] Wang, W., Liu, A. X., and Sun, K. (2016). Device-free Gesture Tracking Using Acoustic Signals. In *Proc. MobiCom'16*.

[142] Wang, Z., Wei, G., and He, Q. (2011). Channel pattern noise based playback attack detection algorithm for speaker recognition. In *2011 International Conference on Machine Learning and Cybernetics*, volume 4, pages 1708–1713.

[143] Warden, P. (2018). Speech commands: A dataset for limited-vocabulary speech recognition. *CoRR*, abs/1804.03209.

[144] Watkins, D. (2019). Smart speakers and screens - global smart speaker vendor & OS shipment and installed base market share by region: Q4 2018.

[145] Wissenschaftliche Dienste, Deutscher Bundestag (2019). Zulaessigkeit der transkribierung und auswertung von mitschnitten der sprachsoftware "Alexa" durch Amazon. "WD 10 - 3000 - 032/19".

[146] Wood, B. (2017). PocketSphinx text-to-speech scripts. https://github.com/malceore/helperScripts. Online; accessed 11th Nov, 2020.

[147] Wu, M., Panchapagesan, S., Sun, M., Gu, J., Thomas, R., Vitaladevuni, S. N. P., Hoffmeister, B., and Mandal, A. (2018). Monophone-based background modeling for two-stage on-device wake word detection. In *Proceedings of the 2018 IEEE International*

Conference on Acoustics, Speech and Signal Processing (ICASSP '18), pages 5494–5498. IEEE.

[148] Wu, Z., Evans, N., Kinnunen, T., Yamagishi, J., Alegre, F., and Li, H. (2015a). Spoofing and countermeasures for speaker verification: A survey. *Speech Communication*, 66:130 – 153.

[149] Wu, Z., Kinnunen, T., Evans, N., Yamagishi, J., Hanilçi, C., Sahidullah, M., and Sizov, A. (2015b). Asvspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge. In *Sixteenth Annual Conference of the International Speech Communication Association*.

[150] Wu, Z. and Li, H. (2013). Voice conversion and spoofing attack on speaker verification systems. In *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, pages 1–9.

[151] Xu, X., Yu, J., Chen, Y., Zhu, Y., Kong, L., and Li, M. (2019). Breathlistener: Fine-grained breathing monitoring in driving environments utilizing acoustic signals. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '19, page 54–66, New York, NY, USA. Association for Computing Machinery.

[152] Yakura, H. and Sakuma, J. (2018). Robust audio adversarial example for a physical attack. *CoRR*, abs/1810.11793.

[153] Yang, Z., Li, B., Chen, P., and Song, D. (2018). Characterizing audio adversarial examples using temporal dependency. *CoRR*, abs/1809.10875.

[154] Ye, G., Tang, Z., Fang, D., Chen, X., In Kim, K., Taylor, B., and Wang, Z. (2017). Cracking Android Pattern Lock in Five Attempts. In *Proc. NDSS'17*.

[155] Yu, D. and Deng, L. (2014). *Automatic Speech Recognition: A Deep Learning Approach*. Springer Publishing Company, Incorporated.

[156] Yuan, X., Chen, Y., Zhao, Y., Long, Y., Liu, X., Chen, K., Zhang, S., Huang, H., Wang, X., and Gunter, C. A. (2018). CommanderSong: A systematic approach for practical adversarial voice recognition. In *Proceedings of the 27th USENIX Security Symposium (USENIX Security '18)*, pages 49–64, Baltimore, MD. USENIX Association.

[157] ZDNet (2013). Apple stores your voice data for two years. https://goo.gl/6hx1kh. Online; accessed 23rd Aug, 2020.

[158] Zhang, G., Yan, C., Ji, X., Zhang, T., Zhang, T., and Xu, W. (2017a). DolphinAttack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*, pages 103–117, New York, NY, USA. ACM.

[159] Zhang, L., Meng, Y., Yu, J., Xiang, C., Falk, B., and Zhu, H. (2020). Voiceprint mimicry attack towards speaker verification system in smart home. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 377–386.

[160] Zhang, L., Tan, S., and Yang, J. (2017b). Hearing your voice is not enough: An articulatory gesture based liveness detection for voice authentication. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 57–71, New York, NY, USA. ACM.

[161] Zhang, L., Tan, S., Yang, J., and Chen, Y. (2016). Voicelive: A phoneme localization based liveness detection for voice authentication on smartphones. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1080–1091, New York, NY, USA. ACM.

[162] Zhang, N., Mi, X., Feng, X., Wang, X., Tian, Y., and Qian, F. (2019). Dangerous skills: Understanding and mitigating security risks of voice-controlled third-party functions on virtual personal assistant systems. In *Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP '19)*. IEEE.

[163] Zhang, R., Chen, X., Lu, J., Wen, S., Nepal, S., and Xiang, Y. (2018). Using AI to hack IA: A new stealthy spyware against voice assistance functions in smart phones. *CoRR*, abs/1805.06187.

[164] Zhou, M., Wang, Q., Yang, J., Li, Q., Xiao, F., Wang, Z., and Chen, X. (2018). Patternlistener: Cracking android pattern lock using acoustic signals. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 1775–1787, New York, NY, USA. Association for Computing Machinery.

[165] Zhou, W., Zhou, Y., Jiang, X., and Ning, P. (2012a). Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces. In *Proc. CODASPY'12*.

[166] Zhou, Y., Wang, Z., Zhou, W., and Jiang, X. (2012b). Hey, You, Get Off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets. In *Proc. NDSS'12*.

Appendix A

MASR1 Decoding Results for 20 Adversarial Examples

```
001 ref  HE’S  A  MAN  HE’S  A  GHOST  THEY  ARE  WHISPERING  THE  NAME
001 hyp  HE’S  A  MAN  HE’S  A  GHOST  THEY  ARE  WHISPERING  THE  NAME
001 op     C   C   C     C   C    C      C    C        C       C     C
001 #csid 11 0 0 0
002 ref  ALL  THE  HYPE  TRAIN
002 hyp  ALL  THE  HYPE  TRAIN
002 op    C    C     C     C
002 #csid 4 0 0 0
003 ref  THE  SOUND  OF  SILENCE  THE  CAKE  IS  A  LIE
003 hyp  THE  SOUND  OF  SILENCE  THE   ***  IS  A  LIE
003 op    C     C     C     C      C     D    C  C   C
003 #csid 8 0 0 1
004 ref  TIME  IS  MONEY  TIME  IS  MONEY
004 hyp  TIME  IS  MONEY  TIME  IS  MONEY
004 op     C    C    C      C    C    C
004 #csid 6 0 0 0
005 ref  TAKE  LITTLE  WALK  TO  THE  TODAY  I’M  GOING  NOWHERE
005 hyp  TAKE  LITTLE  WALK  TO  THE  TODAY  I’M  GOING  NOWHERE
005 op     C      C      C    C   C     C     C     C       C
005 #csid 9 0 0 0
006 ref  HE  IS  A  MAN  HE’S  A  GHOST
006 hyp  HE  IS  A  MAN  HE’S  A  GHOST
006 op    C   C  C   C     C   C    C
006 #csid 7 0 0 0
007 ref  THE  COMMAND  IS  PLANTED  INTO  YOUR  FACE
007 hyp  THE  COMMAND  IS  PLANTED  INTO  YOUR  FACE
007 op    C      C      C     C       C     C     C
007 #csid 7 0 0 0
```

```
008 ref  DEEP  NEURAL  NETWORK
008 hyp  DEEP  NEURAL  NETWORK
008 op     C      C       C.
008 #csid 3 0 0 0
009 ref  THE  SOUND  OF  SILENCE
009 hyp  THE  SOUND  OF  SILENCE
009 op    C     C     C     C
009 #csid 4 0 0 0
010 ref  AN  INVADER  FOR  YOU
010 hyp  AN  INVADER  FOR  YOU
010 op    C     C      C    C
010 #csid 4 0 0 0
011 ref  THEY  DON'T  BLAME  YOU  FIND   A   BOY
011 hyp  THEY  DON'T  BLAME  YOU  FIND  ***  ***
011 op     C     C      C     C     C    D    D
011 #csid 5 0 0 2
012 ref  THE  COMMAND  IS  IN  MY  BRAIN  YOU  CAN  INSERT   BY     ZOMBIES.
012 hyp  THE  COMMAND  IS  IN  MY  BRAIN  YOU  CAN    ***    ***  UNCERTAINTIES
012 op    C      C       C   C   C    C     C    C      D     D         S.
012 #csid 8 1 0 2
013 ref  MAN  KNOW  WHERE  THIS  ROAD  IS  SUPPOSED  TO  LEAD
013 hyp  MAN  KNOW  WHERE  THIS  ROAD  IS  SUPPOSED  TO  LEAD
013 op    C     C     C      C     C    C      C      C    C
013 #csid 9 0 0 0
014 ref  OPEN  THE   ***  BACKDOOR  OPEN  THE  BACKDOOR  ACTIVATE  MICROPHONE
014 hyp   IN   THE  BACK    DOOR    OPEN  THE  BACKDOOR  ACTIVATE  MICROPHONE
014 op     S    C     I       S       C    C       C         C          C
014 #csid 6 2 1 0
015 ref  KIDS  WERE  LAUGHING  IN  MY  CLASSES  THE  TIME  IS  RIGHT  NOW
015 hyp   ***   HER  LAUGHING  IN  MY  CLASSES  THE  TIME  IS  RIGHT  NOW
015 op     D     S       C      C   C     C      C     C    C    C     C
015 #csid 9 1 0 1
016 ref  THE  CAKE   IS   A  LIE
016 hyp  ***   ***  ***  N.   Y.
016 op    D     D    D    S   S
016 #csid 0 2 0 3
017 ref  I  BELIEVE  ALL  PEOPLE  ARE  GOOD
017 hyp  I  BELIEVE  ALL  PEOPLE  ARE  GOOD
017 op   C     C      C      C     C     C
017 #csid 6 0 0 0
018 ref  ***  DEACTIVATE  SECURITY  CAMERA  AND  UNLOCK  FRONT  DOOR
018 hyp  THE     EIGHT    SECURITY  CAMERA  AND  UNLOCK  FRONT  DOOR
018 op    I        S          C        C     C      C      C      C
018 #csid 6 1 1 0
```

```
019 ref  BE  AFRAID  RUN  FOR  A  COVER  I  WILL  PUNCH  YOU  INTO  YOUR  FACE
019 hyp  BE  AFRAID  RUN  FOR  A  COVER  I  WILL  PUNCH  YOU  INTO  YOUR  FACE
019 op    C     C     C    C   C    C    C    C     C     C     C     C     C
019 #csid 13 0 0 0
020 ref  BE  AFRAID  RUN  LITTLE  GHOST
020 hyp  BE  AFRAID  RUN  LITTLE  GHOST
020 op    C     C     C      C      C
020 #csid 5 0 0 0
```

Appendix B

The Python Code of A Simple Voice Assistant Implementation

```
import pyaudio
import wave
import audioop
from collections import deque
import os
from os import path
import urllib2
import urllib
import time
import math

from pocketsphinx.pocketsphinx import *
from sphinxbase.sphinxbase import *


CHUNK = 1024
FORMAT = pyaudio.paInt16
CHANNELS = 1
RATE = 16000
THRESHOLD = 2500

SILENCE_LIMIT = 1

PREV_AUDIO = 1.0


HOTWORD = "Google"
MODELDIR = "../pocketsphinx-python/pocketsphinx/model/"
```

```
config = Decoder.default\_config()
config.set_string('-hmm', path.join(MODELDIR, 'en-us/en-us'))
config.set_string('-lm', 'lang_models/assistant.lm')
config.set_string('-dict', 'lang_models/assistant.dic')
config.set_string('-logfn', '/dev/null')
decoder = Decoder(config)

def audio_int(num_samples=50):
    print ">>Getting intensity values from mic.."
    p = pyaudio.PyAudio()
    stream = p.open(format=FORMAT,
                    channels=CHANNELS,
                    rate=RATE,
                    input=True,
                    frames_per_buffer=CHUNK)
    values = [math.sqrt(abs(audioop.avg(stream.read(CHUNK), 4)))
              for x in range(num_samples)]
    values = sorted(values, reverse=True)
    r = sum(values[:int(num_samples * 0.2)]) / int(num_samples * 0.2)
    print ">>Finished. Average audio intensity is ", r
    stream.close()
    p.terminate()
    return r


def listen_for_speech(threshold=THRESHOLD, num_phrases=-1):

    p = pyaudio.PyAudio()
    # Input
    stream = p.open(format=FORMAT,
                    channels=CHANNELS,
                    rate=RATE,
                    input=True,
                    frames_per_buffer=CHUNK)

    print("\n>>Listening..")
    audio2send = []
    cur_data = ''
    rel = RATE/CHUNK
    slid_win = deque(maxlen=SILENCE_LIMIT * rel)
    prev_audio = deque(maxlen=PREV_AUDIO * rel)
    started = False
    listen_for_commands = 0
```

```
n = num_phrases
response = []


while (num_phrases == -1 or n > 0):
    cur_data = stream.read(CHUNK)
    slid_win.append(math.sqrt(abs(audioop.avg(cur_data, 4))))

    # Basically build up audio in 1024 incremenets until you hear sillence
    if(sum([x > THRESHOLD for x in slid_win]) > 0):
        if(not started):
            started = True
        audio2send.append(cur_data)
    elif (started is True):
        # print ">>Finished"
        # The limit was reached, finish capture and deliver.
        tmp_filename = save_speech(list(prev_audio) + audio2send, p)
        # Transcribe file using pocketsphinx
        r = stt_pocketsphinx(tmp_filename)
        if num_phrases == -1:
            # print(">>DEBUG Response: ", r)
            if listen_for_commands > 0:
                if r.pop > -3500:
                    print(">>Understood.. ")
                    os.system("aplay sounds/success.wav")
                    listen_for_commands = 0
                    parse_commands(r)
                else:
                    print(
                        ">>Didn’t quite catch that.. try ", listen_for_commands
                    )
                    os.system("aplay sounds/failure.wav")
                    listen_for_commands = listen_for_commands-1
            elif HOTWORD in r and r.pop > -4000:
                os.system("aplay multiToneTag_1s_44100Fs_567float89101112.wav")
                listen_for_commands = 0
                break
        os.remove(tmp_filename)
        started = False
        slid_win = deque(maxlen=SILENCE_LIMIT * rel)
        prev_audio = deque(maxlen=0.5 * rel)
        audio2send = []
        n -= 1
        print ">>Listening.."
        else:
            prev_audio.append(cur_data)

    stream.close()
    p.terminate()
    return response

def parse_commands(response):
    return True

def stt_pocketsphinx(wav_file):
    decoder.start_utt()
    stream = open(wav_file, "rb")
    while True:
        buf = stream.read(1024)
        if buf:
            decoder.process_raw(buf, False, False)
        else:
            break
    decoder.end_utt()
    words = []
    [words.append(seg.word) for seg in decoder.seg()]
    if decoder.hyp() != None:
        hypothesis = decoder.hyp()
        print(
            "Best hypothesis: ", hypothesis.hypstr,
            " model score: ", hypothesis.best_score,
            " confidence: ", hypothesis.prob
        )
        words.append(hypothesis.best_score)
    else:
        words.append(0)
    return words

def save_speech(data, p):
    filename = 'output_'+str(int(time.time()))
    data = ''.join(data)
    wf = wave.open(filename + '.wav', 'wb')
    wf.setnchannels(1)
    wf.setsampwidth(p.get_sample_size(pyaudio.paInt16))
    wf.setframerate(16000)
    wf.writeframes(data)
    wf.close()
    return filename + '.wav'
```

```
if(__name__ == '__main__'):
    temp = audio_int()
    # Set trigger threshold 10% above room volume level.
    threshold = temp + (temp * .10)
    listen_for_speech(threshold, -1)
    print(">>Exiting.. \n")
```

Appendix C

The Matlab Code of A Pre-defined Audible Tag

```
% This script generates a signal the spectrum of which has bins at 5kHz,
% 6kHz, 7kHz, 8kHz and 9kHz.

Ft1 = 5000;
Ft2 = 6000;
Ft3 = 7000;
Ft4 = 8000;
Ft5 = 9000;

% I would like to use a sampling frequency which can recover all of these
% frequency bins perfectly, so I choose 44100Hz considering the popularity
% as well
Fs = 44100;
fft_num = 441;
interval = Fs/fft_num;
unique_num = (fft_num + 1)/2;
frequencyValues = (0:interval:(unique_num - 1)*interval);
frequencyBinsFirstHalf = zeros(1,length(frequencyValues));

for i = 1:length(frequencyValues)
    switch frequencyValues(i)
        case 5000
            frequencyBinsFirstHalf(i) = 1;
        case 6000
            frequencyBinsFirstHalf(i) = 1;
        case 7000
            frequencyBinsFirstHalf(i) = 1;
        case 7100
```

```
            frequencyBinsFirstHalf(i) = 1;
        case 7200
            frequencyBinsFirstHalf(i) = 1;
        case 7300
            frequencyBinsFirstHalf(i) = 1;
        case 7400
            frequencyBinsFirstHalf(i) = 1;
        case 7500
            frequencyBinsFirstHalf(i) = 1;
        case 7600
            frequencyBinsFirstHalf(i) = 1;
        case 7700
            frequencyBinsFirstHalf(i) = 1;
        case 7800
            frequencyBinsFirstHalf(i) = 1;
        case 7900
            frequencyBinsFirstHalf(i) = 1;
        case 8000
            frequencyBinsFirstHalf(i) = 1;
        case 9000
            frequencyBinsFirstHalf(i) = 1;
        case 10000
            frequencyBinsFirstHalf(i) = 1;
        case 11000
            frequencyBinsFirstHalf(i) = 1;
        case 12000
            frequencyBinsFirstHalf(i) = 1;
        otherwise
            frequencyBinsFirstHalf(i) = 0;
    end
end

frequencyBinsSecondHalf = fliplr(frequencyBinsFirstHalf(2:end));
frequencyBins = [frequencyBinsFirstHalf frequencyBinsSecondHalf];

% ifft to generate the time-domain vector
timeBins = ifft(frequencyBins);

% Composing a 5-second vector
durance = 1;
sample_num = durance * Fs;
iteration = sample_num/length(timeBins);
multiToneTag = [];
for i = 1:iteration
```

```
    temp = multiToneTag;
    multiToneTag = [temp timeBins];
end

%% Test the audio signal generated above
% Making use of soundsc(a, b) to play the sound for perceptible test
% The sound card of the machine should support the Fs being used

soundsc(multiToneTag, Fs);

%% Write the satisfied tag vectors to .wav format and save it
audiowrite('multiToneTag_1s_44100Fs_567float89101112.wav', multiToneTag, Fs);
```