2014

Secure and Reliable Wireless Communication through End-to-End-based Solution

Hossen Asiful Mustafa
University of South Carolina - Columbia

Secure and Reliable Wireless Communication through End-to-End-based Solution

by

Hossen Asiful Mustafa

Bachelor of Science
Bangladesh University of Engineering and Technology 2005

Master of Engineering
University of South Carolina 2011

Submitted in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy in

Computer Science and Engineering

College of Engineering and Computing

University of South Carolina

2014

Accepted by:

Wenyuan Xu, Major Professor

Csilla Farkas, Committee Member

Manton M. Matthews, Committee Member

Srihari Nelakuditi, Committee Member

Jacob Sorber, Committee Member

Lacy Ford, Vice Provost and Dean of Graduate Studies

Acknowledgments

There are many people who made this dissertation possible by their direct or indirect help, unwavering guidance, moral support, and blind friendship and I am greatly indebted to them.

Firstly and most importantly, I would like to express my utmost gratitude to Dr. Wenyuan Xu, who is not only my advisor but also my mentor. She taught me how to become a researcher from a coder. Her demands for high standards have helped me immensely to improve my presentation and writing skills. I am greatly indebted to her continuous inspiration, and guidance, constant encouragement and absolute support during the last five years.

I am very grateful to many professors at the Department of Computer Science and Engineering in University of South Carolina (USC) who have been instrumental in my theoretical and professional developments. Specifically, I would like to thank Dr. Srihari Nelakuditi who set "the bar" much higher which motivated me a lot. I also express my gratitude to Dr. Csilla Farkas, Dr. Manton M. Matthews, and Dr. Jacob Sorber for their valuable advice and encouragements.

It would be a crime not to acknowledge the members of Dr. Xu's research group. Thanks to Zhenhua, Miao, Ishtiaq, Jing, and Aniqua for bearing with me during this period. Without them, this journey would not have been so much fun!

Finally, I would like to express my gratitude my father Dr. Muhammed Mustafa, my mother Jahanara Hasnat, My sisters - Hasna Mustafa, and Jakiun Mustafa, and my wife Jahan Arju; thank you for always being there for me; thank you for your inspiration and support; my achievements would not be possible without you!

Abstract

In the past few decades, network architectures and protocols are often designed to achieve a high throughput and a low latency. Security was rarely considered during the initial design phases. As a result, many network systems are insecure by design. Once they are widely deployed, the inherent vulnerabilities may be difficult to eliminate due to the prohibitive update cost. In this dissertation, we examine such types of vulnerabilities in various networks and design end-to-end-based solutions that allow end systems to address such loopholes.

The end-to-end argument was originally proposed to let end hosts implement application-specific functions rather than letting intermediate network nodes (i.e., routers) perform unneeded functions. In this dissertation, we apply the end-to-end principle to address three problems in wireless networks that are caused by design flaw with following reasons: either because integrating solutions into a large number of already deployed intermediate nodes is not a viable option or because end hosts are in a better position to cope with the problems. First, we study the problem of jamming in a multihop wireless network. Jamming attacks are possible because wireless networks communicate over a shared medium. It is easy to launch a jamming attack but is difficult to defend against it. To ensure the end-to-end packet delivery, we propose a jamming-resilient multipath routing algorithm that maximizes end-to-end availability based on the availability history between sources and destinations. Second, we investigate caller ID spoofing attacks in telephone networks in which an attacker can send a fake caller ID to a callee rather than her real one to impersonate as someone else. Such attacks are possible because there is no caller ID authentication

mechanism in operator interconnection protocols. Modifying current protocols to verify caller ID between operators may be infeasible due to the scale of deployed systems. So, we propose two schemes to detect caller ID spoofing attacks based on end-to-end verification. Finally, we examine evil twin access point attacks in wireless hotspots. In such attacks, an adversary sets up a phishing access point that has the same Service Set IDentification (SSID) as the legitimate ones in the hotspot. Such attacks are easy to launch because of how 802.11 standards are designed. Existing solutions take away convenience from the user while providing security. Our aim is to detect evil twin access point attacks in wireless hotspots without modifying how access point works in hotspots and without additional infrastructure support. We propose an end-to-end-based mechanism that can effectively detect evil twin access point attacks in wireless hotspots.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Wireless technologies are becoming increasingly popular, because they can offer pervasive services at an affordable price. In the past few decades, a variety of wireless technologies have been proposed to cover almost all possible communication scenarios. For instance, Bluetooth [17] technologies are designed for short-distance communication (several meters), 803.11 (Wi-Fi) [58] for medium-distance communication (up to 250 meters), LTE [105] for high-speed long-distance communication (up to 70 miles), etc. Apart from providing last-hop Internet services to end users, wireless networks have been applied to various situations to improve the quality of our life like never before, including smart meters for improving the efficiency of power grids, medical devices for continuous treatment outside hospitals, vehicular ad hoc networks for improving road safety, etc. Often wireless networks were designed with the goal of achieving a high throughput and a low latency; security was not always considered in the initial design phase. Consequently, security and reliability problems of the wireless communication may appear after wireless systems are deployed.

Since systems were often designed without security in-mind, many network systems have been widely deployed with inherent vulnerabilities, i.e., such systems may fail to satisfy one or more properties of secure and reliable wireless communication: (a) **confidentiality**, which means that the content of messages is accessible only by authorized users, i.e., an attacker can eavesdrop on the communication but cannot

read the content, (b) **integrity**, which means that a receiver can verify whether a message has been altered or corrupted, (c) **authentication**, which means that a receiver can verify the identity of a sender, and (d) **availability**, which represents quality of service. Failure to satisfy the above properties may lead to different types of attacks, e.g., evil twin access point attack in Wi-Fi networks. Although these properties may be achieved using traditional cryptographic solutions [103][53][55][54][124][69], many deployed systems were not designed to ensure all of them, e.g., Wi-Fi networks, telephone interconnection networks, etc. As a result, inherent vulnerabilities may be difficult to eliminate either because the update cost may be prohibitive for a large-scale system, or a cryptographic solution may take away convenience and degrade usability. In some cases, even if we could redesign the protocol, we may not eliminate all vulnerabilities. In this dissertation, we address such vulnerabilities. Rather than modifying the core network components in existing infrastructures, we design *end-to-end-based solutions* that enable end systems to address such loopholes.

End-to-end principle [102] is one of the well-known principles of computer networking. This principle suggests that end hosts should implement application-specific functions rather than letting intermediate network nodes (i.e., routers) perform functions that are unnecessary to all applications, provided that the application-specific functions can be *completely and correctly* implemented in end hosts. Transmission Control Protocol (TCP) is an example that utilizes such a principle. End-to-end principle allows us to design solutions that can be applied/installed in end hosts without modifying the core network infrastructure or without taking away the convenience of existing systems. Additionally, end-to-end solutions ensure that end-users are at liberty to utilize solutions, and not forced to use the existing vulnerable systems.

1.2 Problem Overview

In this dissertation, we examine problems in wireless networks that are caused by how the networks were designed. Our problem domain covers from multi-hop ad hoc networks, widely popular Wi-Fi networks to several types of widely deployed telephone networks. We apply the end-to-end principle to address the problems either because integrating solutions into a large number of already deployed intermediate nodes is not an option or because end hosts are in a better position to cope with the problems. In particular, we investigate three attacks in three different types of networks and address these vulnerabilities through end-to-end-based solutions. We discuss these attacks in the following.

Jamming Attacks. Jamming attacks are especially harmful to the reliability of wireless communication, as they can effectively disrupt communication between any node pairs. Jamming attacks are possible because wireless networks communicate over a shared medium. It is easy to launch jamming attacks using off-the-shelf devices, but is difficult to defend against them. Existing jamming defense techniques primarily focus on repairing connectivity between adjacent nodes. We take a different point of view; rather than repairing network communication locally, we aim to achieve end-to-end availability at the network layer. Our basic idea is to choose multiple fault-independent paths between a pair of communication nodes; as long as all paths do not fail concurrently, the end-to-end availability is maintained. Existing multipath routing algorithms choose multiple disjoint paths, i.e., paths do not share common nodes or links. However, through our experiments using MicaZ nodes, we found that disjointness is not sufficient for selecting fault-independent paths in the presence of jamming attacks. Instead, we choose multiple paths that maximize the `end-to-end availability` based on the historic availability measurements of wireless links between nodes.

Caller ID Spoofing Attacks. Caller ID (caller identification) is a service provided by telephone operators, where the phone number and/or the name of the caller are transmitted to inform the callee who is calling. Today, most people trust the caller ID information, and it is increasingly used to authenticate customers (e.g., by banks or credit card companies). However, with the proliferation of smartphones and VoIP, it is easy to spoof caller ID by installing corresponding Apps on smartphones or by using fake ID providers. This vulnerability has already been exploited with crucial consequences such as faking caller IDs to emergency services (e.g., 9-1-1) or to commit fraud. As telephone networks are fragmented between enterprises and countries, modifying existing protocols to add caller ID verification mechanisms may not be financially feasible due to the cost of upgrading large deployments. No mechanism is available today to easily detect such spoofing attacks. In this dissertation, we propose an `end-to-end caller ID verification` mechanism `CallerDec` that works with existing combinations of landlines, cellular and VoIP networks. CallerDec can be deployed at the liberty of users, and does not require any modification to the existing infrastructures.

Evil Twin Access Point Attacks. Internet usage through wireless networks, Wi-Fi in particular, has gained rapid popularity in the last few years. Wi-Fi is now de facto network interface for many state of the art devices, e.g., smartphones, tablets, laptops, smart televisions, etc. To take advantage of such popularity, many shops, cafés, airports, and parks provide wireless hotspot services to users free of cost. Wireless hotspots allow users to use Internet via Wi-Fi interface. However, there is no authentication mechanism available in such hotspots which makes them vulnerable to evil twin access point (AP) attacks. To launch such attacks, an adversary sets up a phishing AP that has the same Service Set IDentification (SSID) as a legitimate AP in the victim hotspot. The phishing AP could be a laptop, a smartphone, or any types of Wi-Fi-enabled devices. Such an attack can be harmful to users when

the attack is used for stealing sensitive data from the wireless users. Cryptographic solutions could be used to counter evil twin AP attacks. However, these solutions would take away convenience of the hotspot, and hotspot providers are less likely to adopt them. Thus, end-user mechanism that can effectively detect evil twin AP attacks in wireless hotspots is necessary but not available today. Instead of changing how an AP of a hotspot operates, we focus on detecting evil twin access point attacks by executing an `end-to-end-based solution` at the hotspot client.

1.3 Overview of the Dissertation

In this dissertation, we analyze three security vulnerabilities in wireless networks and present end-to-end-based solutions to address all of the vulnerabilities. The organization of the dissertation is as follows:

In Chapter 2, we briefly overview the problem of wireless interference or jamming attacks in wireless mesh networks. We propose to select multiple paths based on the knowledge of a path's *availability history*. Using Availability History Vectors (AHVs) of paths, we present a centralized AHV-based algorithm to select fault-independent paths, and a distributed AHV-based routing protocol that is built on top of Ad hoc On-Demand Distance Vector (AODV) routing protocol. Our extensive simulation results validate that both AHV-based algorithm and AHV-based routing protocol can effectively overcome the jamming impact by maximizing the end-to-end availability of the selected paths.

In Chapter 3, we examine caller ID spoofing attacks in telephone networks and identify the underlying factors that allow an adversary to launch such attacks. We propose two end-to-end caller ID verification schemes which we call `CallerDec`. CallerDec requires no modification of the existing telephone network infrastructure. We design an SMS-based CallerDec scheme for phones that have SMS services and a timing-based CallerDec for landlines, cell phones and VoIP phones. We imple-

mented both CallerDec schemes as Apps for Android based phones and validated their effectiveness in detecting caller ID spoofing attacks in various scenarios.

In Chapter 4, we overview the problem of evil twin access point attacks in wireless hotspots and discuss the potential effects of such attacks to end users. We propose an end-user mechanism, `CETAD`, that can effectively detect evil twin access point attacks in wireless hotspots without any infrastructure support or without prior training. Our proposed mechanism leverages public servers and only requires installing an App at the client device without changing the hotspot APs. Through our implementation and evaluation, we show that CETAD can detect evil twin AP attacks in various scenarios effectively.

Finally, we conclude the dissertation with a summary of the research and outline directions for future work in Chapter 5.

Chapter 2

Multipath Routing Maximizing End-to-End Availability

2.1 Background

Wireless networks communicate through a shared medium and thus are vulnerable to both malicious jamming attacks and radio interference. With little effort, an adversary can purchase an off-the-shelf programmable radio (e.g., a software-defined radio) and build a jammer to continuously disturb wireless communication. Even without adversaries, the tension between the proliferation of wireless technologies and the limited number of unlicensed bands will continue to make the radio environment crowded. Already, an increasing number of companies are manufacturing various types of wireless devices using the 2.4 GHz radio spectrum, such as Wi-Fi network adapters, 2.4 GHz cordless phones, Bluetooth headsets, ZigBee-enabled appliances, microwave ovens, etc. Those devices can experience throughput degradation from unintended radio interference among co-existing devices. Whether intentional or not, both jamming attacks and radio interference will continue to be one of the most serious threats to the dependability of wireless communication. Since both threats can prevent networks from delivering information, we use the term *jamming* to refer to both in the rest of the chapter.

To cope with jamming, much research effort has focused on local repairing, i.e., restoring communication between adjacent nodes. Those anti-jamming measures include the conventional physical-layer techniques that rely on advanced transceivers [98]

(e.g., frequency hopping) and MAC-layer mechanisms [88][136] that adjust error correcting codes, channel adaptation [136], or physical location [76]. Although those techniques are important to defend against jamming, we take a different viewpoint and focus on defending against jamming at the network level, i.e., restoring reliability of end-to-end data delivery.

In this study, we examine multipath routing protocols that will react to communication disturbance on-demand. In particular, a source node selects multiple different paths for reaching the destination in advance. When one of the paths fails, other working paths will be used to deliver packets and thereby maintain end-to-end availability, *as long as not all paths between the source and the destination fail concurrently.* Such end-to-end availability provided by multiple paths between a source-destination pair is referred to as *multipath availability.* A crucial component of multipath routing is *multipath selection* (i.e., the decision process of determining which paths to use), as the selection tactic and resulting path qualities will directly influence the effectiveness of multipath routing. In this study, we design multipath selection algorithms that will optimize multipath availability even when one or more jammers may disrupt network communication occasionally or continuously.

Most existing multipath selection algorithms in both wireless and wired networks [83][127][2][139][142] choose node-disjoint paths or link-disjoint paths, i.e., paths without common nodes or shared links, in an attempt to minimize the probability that all the paths simultaneously fail. While such an approach is simple and intuitive, it relies on the assumption that the disjointness among multiple paths is sufficient to guarantee failure-independence. However, in a wireless network, disjoint paths can still be failure-correlated, especially in the presence of multiple interference sources. We illustrate this by the following two scenarios. In both scenarios, a wireless ad hoc network with each node equipped with an 802.11b/g network adaptor is deployed in a square area. Consider three *disjoint* paths between the source node

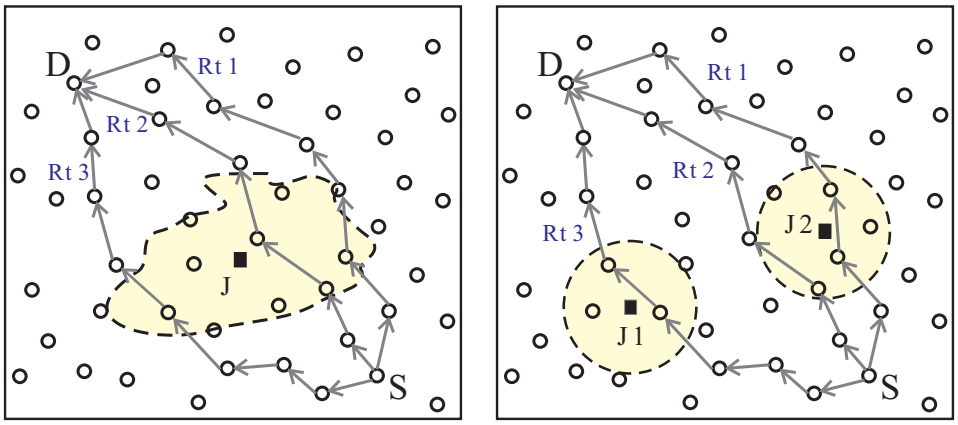


Figure 2.1 An illustration that disjoint paths are still correlated with regard to jamming (shaded areas represent jamming regions): (a) one non-isotropic jamming area and (b) two jamming areas far apart.

S and the destination node D, as illustrated in Figure 2.1. In the first example, a stationary interferer J with an irregular jamming area (e.g., a mounted microwave) becomes active occasionally, as shown in Figure 2.1(a). Upon turning on, J disturbs all three disjoint routes ($Rt1$, $Rt2$, $Rt3$) and causes them to fail concurrently. As another example, two sets of 2.4 GHz cordless phones call each other from time to time, as denoted by J_1 and J_2 in Figure 2.1(b). They start to interfere with the network communication whenever they are connected. As a result, J_1 and J_2 turn all three routes to be fault-correlated. In both cases, the disjointness is necessary but not sufficient to guarantee fault-independence between paths.

To address the failure correlation between disjoint paths in the presence of jamming, one natural way is to mathematically model the impact of jamming on the network links. However, electromagnetic signals propagate in complex environments full of absorption, reflection, scattering and diffraction, and the resulting jamming impact on the network is highly irregular [73]. Figure 2.2 shows packet delivery contours of a sender in the presence of a jammer obtained using MicaZ nodes. The pink (dark-shaded) areas within which a receiver can successfully receive messages exhibit

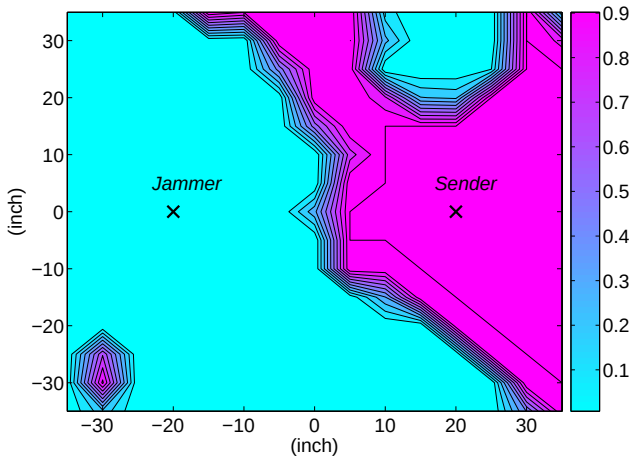


Figure 2.2 PDR contours of a sender located at (20,0) in the presence of a jammer located (-20,0) to illustrate the irregularity of jamming effect in a real system. To obtain the PDR contours, a receiver was placed at the grid with a grid size of 5 inches in an indoor environment. The sender, receiver, and the jammer are all implemented on MicaZ nodes with the same transmission power levels.

high irregularity. This indicates that even given accurate information of jammers' locations and jamming power levels, it is still difficult, if even possible, to quantify their impacts with reasonable accuracy.

Rather than relying on inaccurate models, our key insight is to address multipath selection based on the knowledge of a path's *availability history*, granted that failure correlation between paths can be automatically derived from their availability history. Specifically, if two paths tend to exhibit a history of concurrent failures, we regard them as failure-correlated, otherwise as failure-independent. Admittedly, two failure-independent paths may also present certain concurrent failures by chance, but such a *random correlation* will not harm the multipath selection as it will not lead to choosing failure-dependent paths. Additionally, by using the historical failure correlation of links to predict future correlation, our scheme is most resilient to the types of failures that can repeat in the future, while we will show that it is still effective in improving network reliability when the failures happen only once or new types of failures occur in the future.

Our underlying rationale for leveraging availability history to exploit failure-correlation lies in the following: The intricate factors that cause failure-correlation between different paths (Section 2.3) are difficult to identify, but their impacts on a set of links may be deterministic and long-lasting. Thus, the best we can do is to derive such correlation *after* the failures take place, while it is an open challenge to detect failure-correlation *before* failures happen. In our prior work [143], we have proposed the availability history approach for the Internet. In this chapter, we demonstrate that availability history is particularly effective to defend against jamming attacks in wireless networks and also present how to integrate it with classic ad hoc network routing protocols.

The rest of this chapter is organized as follows. We specify our network and threat model in Section 2.2 and overview the Availability History Vector (AHV)-based approach in Section 2.3. In Section 2.4, we present an AHV-Based Link-State (ALS) algorithm that selects multiple paths based on the global network information, and evaluate the ALS algorithm in our customized simulator. In Section 2.5, we discuss a distributed AHV-based routing algorithm built on top of AODV and present our evaluation effort using ns-3. Finally, we discuss the related work in Section 2.6 and summarize the chapter in Section 2.7.

2.2 Network and Threat Models

In this section, we summarize the network model and the threat model for our study.

Network Model

To focus our effort in examining the resilience of multipath selection against jamming and radio-interference, we consider ad hoc wireless networks with limited mobility, e.g., wireless mesh networks. That is, the link state is primarily affected by jamming and interference but not by the mobility of network nodes. Furthermore, we assume

that each node will maintain a neighbor table recording the link states between its neighbors and itself. Such a neighbor table is supported by most routing protocols and can be easily implemented by periodically broadcasting beacons.

Threat Model

Besides jamming, our scheme is resilient to other types of failures that can change disjoint paths into fault-dependent ones. For instance, network nodes belonging to the same carrier may be far apart but leave/join the network at the same time. Nevertheless, we focus on studying the multipath selection problem under the threat of jamming. In particular, we examine the following representative jammers to mimic radio interference sources and malicious jammers.

- **Stationary Jammers.** One or more stationary jammers alternate between on and off mode but do not move around. This type of jammers can be a radio interferer that becomes active at times, e.g., a mounted microwave or a ZigBee-enabled appliance. Specifically, when interferers are active, they emit energy to the channel without following the MAC protocol implemented by the network. When multiple interferers are present, they can start to emit signals simultaneously, independently, or in a manner such that at least one of them is active at any time instant. Regardless of their activation patterns, they will disturb the network communication inside the same regions occasionally.

- **Mobile Jammers.** A mobile jammer will move around in the network while emitting signals continuously, and it will disrupt the network communication within its vicinity. Such a jammer can either be a malicious jammer or an interferer with mobility, e.g., a Bluetooth device in a moving vehicle. Whether intentional or malicious, a mobile jammer can travel following a specific pattern or can move randomly.

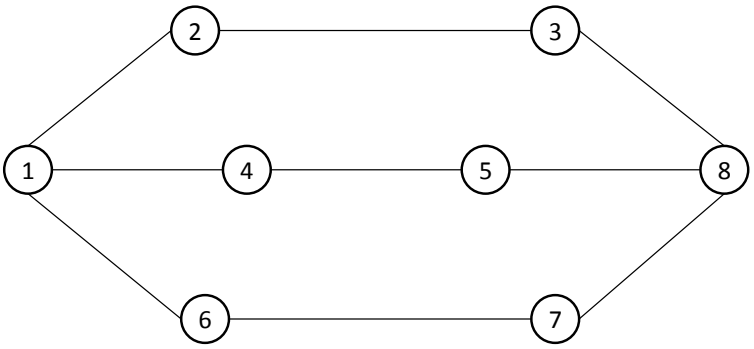


Figure 2.3 A simplified topology of a wireless network with three paths.

The distinction between the stationary and mobile jammer is that the latter, especially the jammer moving randomly, affects a wider range of links but not simultaneously. As a result, it is challenging to predict future link states with the history information when a mobile jammer is present.

2.3 AHV-Based Routing Overview

The success of multipath selection algorithm necessitates two components, namely, (a) a *metric* that can accurately reflect failure correlation between different paths, and (b) a *selection algorithm* that can effectively leverage the metric to rule out failure-correlated paths from being selected together.

In this Section, we first present a mechanism which can not only evaluate individual path availability, but can also derive a multipath availability metric even in the presence of failure correlation between links. Then, we sketch how our mechanism helps to select multiple paths based on the derived availability metric.

We use the following standard notations: "$\wedge$" is the logical AND bit-operation; "$\vee$" stands for the logical OR bit-operation; "$|\mathcal{X}|$" operation returns the cardinality of the set $\mathcal{X}$; and "$\|\mathcal{X}\|$" operation returns the norm of the vector $\mathcal{X}$.

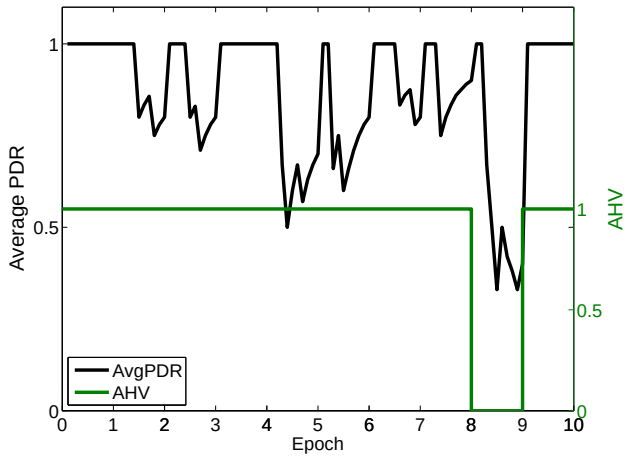


Figure 2.4 An illustration of converting PDR to AHV for the link 1→2 by applying a threshold value of $\gamma_0 = 0.6$.

Availability History Vector

Given the increasing possibility of cross-platform interference and jamming in wireless networks, as well as the overwhelming complexity of the wireless propagation environment that a wireless network relies on, it is difficult (if not impossible) to precisely predict or analyze the correlation between different paths. To bypass such complexity while still exploring the failure correlation between different paths, we propose a mechanism called an *Availability History Vector* (AHV) [143], to record path availability histories, from which the failure correlation between different paths can be learned. We first define an AHV on a per-link basis, from which path (multipath) availability can be then easily derived.

AHV of a Single Link

One natural metric to determine the availability of a wireless link is the Packet Delivery Ratio (PDR), i.e., the percentage of packets successfully delivered to the destination node over the link. Recording the PDR time series directly requires at least 1 byte for each data point, and calculating the aggregated PDR of a path requires

multiplication. Thus, direct PDR recording is not an efficient option considering the packet and computation overhead it will incur. To store and compute availability history efficiently, we utilize a binary vector for recording, and bitwise operations for calculating path availability.

In particular, we map a PDR to a 0-1 value, where '1' corresponds to the time instant when the link is *available* (acceptable PDR), while '0' corresponds to the time instant when the link is *unavailable* (unacceptable PDR). A threshold γ_0 is predefined to determine whether a PDR is acceptable, and γ_0 should be sufficiently high to ensure acceptable end-to-end PDRs. Furthermore, we divide time into epochs with a fixed duration. The length of each epoch can be chosen based on the network characteristics. At the lth epoch, let PDR_{ij}^l be the average PDR between node i and j, then the availability record of the link between node i and node j at the lth epoch is calculated as:

$$r_{ij}^l = \begin{cases} 1 & \text{if } PDR_{ij}^l \geq \gamma_0, \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

The AHV of this link for e epochs is:

$$a_{ij} = [r_{ij}^1, r_{ij}^2, \ldots, r_{ij}^e]. \tag{2.2}$$

To facilitate an observation, we depict an AHV as a continuous line and illustrate an example of converting the PDR into the AHV with γ_0 being 0.6 in Figure 2.4; except for Epoch 9, the availability of other epochs is '1'.

So far, AHV is used to characterize individual links between a pair of nodes. Now we present how to derive an AHV for an entire path consisting of concatenating links or sub-paths. We achieve this by using a *series combination* operation as discussed in the following.

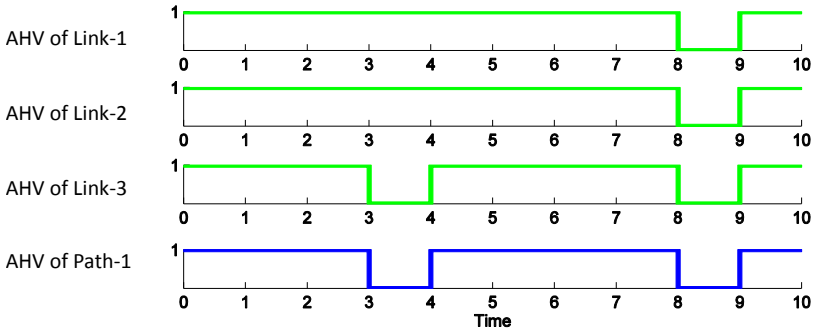


Figure 2.5 The calculation of AHVs for Path-1 in Figure 2.3. Path-1 has three links: 1➞2, 2➞3, and 3➞8.

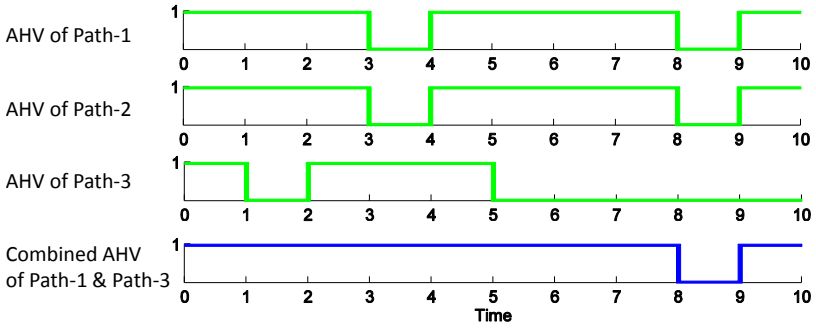


Figure 2.6 The calculation of multipath AHVs for three paths from Figure 2.3.

AHV of One Path

The AHV of a complete path is computed as the logical bitwise AND of all AHVs of the links or sub-paths. The AHV of path p_i can be formulated as

$$A_i = \mathbf{a}_{I_1 I_2} \wedge \mathbf{a}_{I_2 I_3} \wedge \ldots \wedge \mathbf{a}_{I_q I_{q+1}}, \tag{2.3}$$

where I_q is the qth node ID on the path p_i.

For example, recall that Path-1, shown in Figure 2.3, consists of links 1➞2➞3➞8. Figure 2.5 illustrates the series combination for calculating its AHV. The top three lines present the AHVs of links 1➞2, 2➞3, and 3➞8, and the last line is the AHV of Path-1, computed as the bitwise AND of the first three AHVs.

AHV of Multiple Paths

Recall that in multipath routing, we aim at selecting multiple paths that provide the highest multipath availability; thus we derive the AHV of a given set of k paths using the following parallel combination operation.

Let M be the set of k paths between a source-destination pair. The AHV of M is computed as the logical bitwise OR of all AHVs of the paths, denoted as

$$A_M = A_1 \vee A_2 \vee \ldots \vee A_k. \tag{2.4}$$

Figure 2.6 shows an example of the AHVs of three paths along with the combined AHV of two paths. The top three lines present the AHVs of Path-1, Path-2, and Path-3 respectively. The last line is the combined AHV of Path-1 and Path-3, obtained by a logical bitwise OR of the Path-1 and Path-3's AHVs.

Multipath Availability Metric θ

From the availability history carried by AHVs, we can infer that two paths are highly correlated if they tend to fail at the same time, and similarly, two paths are failure-independent if they do not fail at the same time. To facilitate selecting failure-independent routing paths, we define an availability metric θ, which is computed as the number of 1-epochs (i.e., availability bit equals '1' in that epoch) in the AHV of multiple paths between a source-destination pair. Specifically, the availability of a multipath set M is

$$\theta(M) = \|A_M\|. \tag{2.5}$$

Consider the example shown in Figure 2.6, θ of the set of Path-1 and Path-3 is 9.

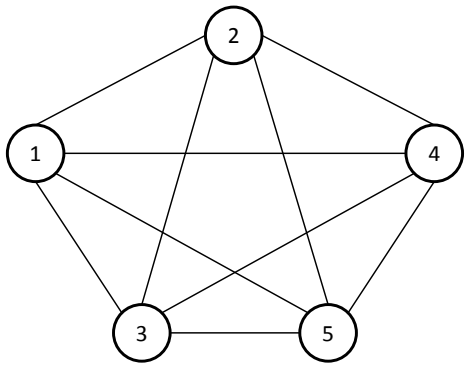


Figure 2.7 A 5-by-5 Network where each pair has sixteen (16) disjoint loop-free paths to each other.

Multipath Selection

The goal of our multipath selection scheme is to select k AHVs that can produce the largest θ, to ensure that failure-correlated paths are bound to be less likely chosen together. Formally, given the set H containing h candidate paths, the multipath selection problem can be defined as the following:

Definition 1.

$$\underset{M}{maximize} \quad \theta(M)$$

$$subject\ to \quad |M| \leq k, M \subseteq H.$$

Multipath Selection Framework

Obtaining the optimal M containing k paths encounters two challenges. First, the multipath selection problem in Definition 1 is NP-complete, according to our prior work [143]. Second, a huge number of possible paths may exist between nodes in a multi-hop wireless network. Consider a 5-by-5 grid network with each node connected with the other four nodes, as shown in Figure 2.7. Sixteen disjoint, loop-free paths exist from 1 to 4: 1→4, 1→3→4, 1→2→4, 1→2→3→4, 1→3→2→4, etc. It is foreseeable that as the number of nodes increases, the number of paths will increase

exponentially. Solving the NP-Complete problem of Definition 1 with a large number of candidate paths can be computationally prohibitive.

To address both issues, we propose a two-stage framework to select multiple paths,

1. the *path pre-selection* stage
2. the *greedy multipath selection* stage,

as shown in Algorithm 1. In particular, we model the network as a weighted graph $G = (N, E, W)$ with N being the node set, E being the link set, and W being the map from edges to weights. Given the network graph G, the function `PreSelectPath()` selects h paths as the candidates in the *path pre-selection* stage. Then, in *greedy multipath selection* stage, the function `AHVSelect()` selects k paths using an approximation algorithm (as shown in Algorithm 2).

Greedy Multipath Selection

Essentially, `AHVSelect()` (Algorithm 2) is a greedy algorithm to choose k paths out of h candidate paths that produce a high level of availability according to the AHVs. In each iteration, the algorithm greedily selects the path that can maximize the multipath availability accumulated so far, and it has a time complexity of $O(hk)$.

To illustrate, recall the example topology shown in Figure 2.3 with the AHV of each path given in Figure 2.6. Suppose Path-1 has already been selected where $\theta(\text{Path-1}) = 8$. If we further select Path-2 which is failure-correlated with Path-

Algorithm 1 AHV-Based Multipath Selection Framework

Require: INPUT:
$G = (N, E, W)$, h, k, $\{A_i\}$;
OUTPUT:
M;
PROCEDURES:
1: $H = \text{PreSelectPath}(G, h)$
2: $M = \text{AHVSelect}(H, k, \{A_i\}_{i \in H})$

Algorithm 2 AHVSelect: AHV-Based Multipath Selection

Require: INPUT:

$H, k, \{A_i\}_{i \in H}$

OUTPUT:

M;

PROCEDURES:

1: $M = \emptyset$, $\theta(M) = 0$
2: **while** $|M| \leq k$ **do**
3: *select* a path $p \in H$ that maximizes $\theta(M \cup p)$
4: *add* p to M, *update* $\theta(M) = \theta(M \cup p)$
5: **end while**

1, the resulting combined AHV gains no increase in the total number of 1-epochs, i.e., θ(Path-1 V Path-2) is also 8. In contrast, if we parallel-combine Path-3 with Path-1, the resulting combined AHV θ(Path-1 V Path-3) is 9. Since Path-3 is failure independent with Path-3, this combination benefits from a significant increase in θ; thus Path-3 is chosen over Path-2 in our selection mechanism even though Path-2 is more stable than Path-3.

2.4 AHV-Based Link-State Algorithm

In this section, we present an AHV-Based Link-State (ALS) algorithm that selects multiple fault-independent paths utilizing the individual AHV information. Specifically, each node maintains a history table recording the AHVs and PDRs between its neighbors and itself, and such history tables for every link are accessible by the ALS algorithm. In practice, collecting the history tables across the entire network might be prohibitive. Nevertheless, the discussion of the ALS algorithm serves as a means to understand and validate the theoretical underpinning of AHV-based multipath selection strategies. In Section 2.5, we present a distributed algorithm that selects multiple fault-independent paths without global knowledge.

Algorithm Description

The AHV-based Link-State algorithm follows the two-stage framework: path pre-selection and greedy multipath selection.

Path Pre-Selection

The combined quality of the h candidate paths directly affects the achievable multi-path availability to be obtained in the greedy multipath selection stage. To improve the combined quality, we defined a few pre-selection rules. Formally, we denote the quality of path p_i as $w(p_i)$ and N_i as the node set on the path p_i. The candidate path set H will satisfy the following requirements:

1. $|H| \leq h$,
2. $\forall p_j, p_i \in H, \frac{|N_i \cap N_j|}{\min(|N_i|,|N_j|)} \leq \rho$,
3. $\forall p_u \notin H, p_i \in H, \quad w(p_i) \geq w(p_u)$ OR
   $\exists p_j \in H, \frac{|N_j \cap N_u|}{\min(|N_j|,|N_u|)} \geq \rho$,

where ρ is a threshold and $\rho \in [0, 1]$.

The first condition requires that the size of set H must not exceed h.

The second condition requires that any pairs of paths belonging to H should have less than ρ percent of shared nodes. The parameter ρ controls the level of overlapping between paths in H. When $\rho = 0$, H only contains node-disjoint paths, while when $\rho = 1$, H consists of any paths without the disjointness restriction. Setting $\rho = 0$ may sound appealing at first glance, but strictly choosing h disjoint paths can filter out 'good' candidates, some of which in combination can be highly failure-independent. At the other extreme, setting $\rho = 1$ can include several candidate paths that are highly correlated with each other, reducing the diversity of H. Thus, ρ serves as a

tunable value to strike the balance between those extreme cases. In our study, we choose $\rho = 0.8$ based on our empirical analysis (Section 2.4).

The third condition requires that a top-ranked candidate path should be selected, unless many of its nodes overlap with a higher ranking one, where the paths are ranked with regard to link quality. We continue to use PDR as a link quality indicator. Considering that the $PDRs$ of links are independent, the PDR of a path from node i to node j with q links is,

$$PDR_{ij} = \prod_{i=1}^{q} PDR_{I_i I_{i+1}}$$

To obtain h paths, we utilized Yen's ranking algorithm [79], a classic algorithm to determine the K shortest paths. A few issues arise when applying Yen's ranking algorithm to obtain H: (a) it assumes the end-to-end weight of two consecutive links equals the *sum* of individual link weight while the end-to-end PDR is the *product* of individual $PDRs$; (b) it returns paths with the top *minimum* weight while we are interested in top *maximum* weighted ones, and (c) it returns paths regardless of the number of shared nodes among them. To address those issues, we define the weight of a path from nodes i to j as

$$w_{ij} = -\log(PDR_{ij}) = -\sum_{i=1}^{j} \log(PDR_{I_i I_{i+1}})$$

and simply discard a path from H if it shares more than ρ percent of nodes with a higher ranking path.

Greedy Multipath Selection

In this stage, we use Algorithm 2 to select k paths from the candidate path set H obtained at the path pre-selection stage. Here, we first select the path with the highest availability, and then we iteratively select the remaining $k-1$ path maximizing combined availability.

Algorithm Evaluation

In this section, we evaluate the performance of the AHV-based Link-State algorithm in our customized simulator that was implemented in Java. Our customized simulator has the flexibility of adopting various physical propagation models and hardware models for decoding packets. The focus is to validate the algorithm performance without the influence of network traffic.

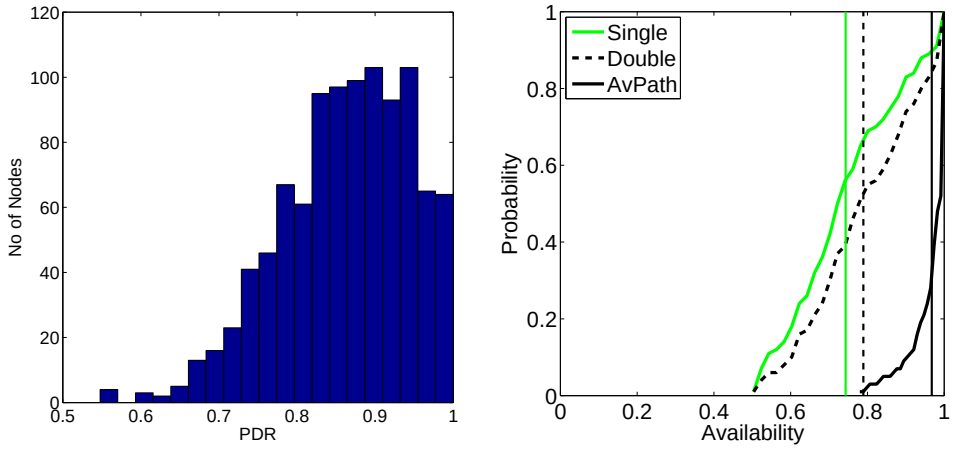
Simulation Methodology

Propagation Model. To prepare for the extensive performance study, we chose a simple yet representative model that captures the essence of signal propagation without using computer-aided modeling tools, i.e., log-normal shadowing model [44]. Our model captures both path loss versus distance and the random attenuation due to blockage from objects in the signal path, and it has the following form,

$$PL(d) = PL(d_0) - 10 \cdot \eta \cdot \log(\frac{d}{d_0}) + X_\sigma,$$

where $PL(d)$ is the path loss at distance d, $PL(d_0)$ is the known path loss at a reference distance d_0, η is the Path Loss Exponent, and X_σ is a Gaussian zero-mean random variable with standard deviation σ. To emulate a real environment, we tune the variables obtained from our prior empirical study [73]: $\eta = 2.11$, $\sigma = 1.8$, and $PL(d_0) = 33$dB.

Essentially, the link quality (PDR) between a pair of directly connected nodes is determined by the physical-layer metric, i.e., signal-to-noise ratio (SNR) at the receiver. When the SNR is larger than a threshold value ξ_o, a message can be decoded and will be received successfully, otherwise it will not. We measured the PDRs by examining the SNR for each link periodically while setting ξ_o to 0dB. After obtaining the PDR between node i and j for the l^{th} epoch, the availability, r_{ij}^l, can be derived according to Equation 2.1.



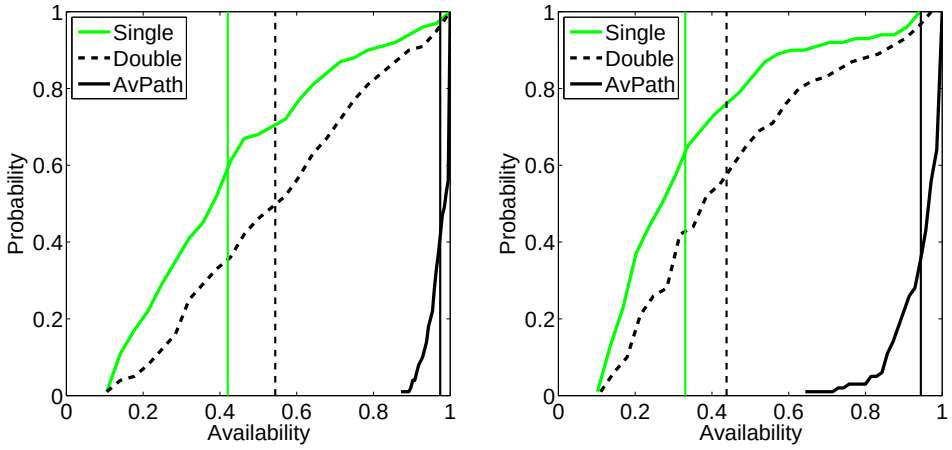
(a) histogram of PDR_{SD} (b) CDF

Figure 2.8 The end-to-end PDR and availability of ALS in no-jammer cases.

Network Setup and Scenarios. We simulated a random wireless network consisting of 1000 nodes in a $700m$-by-$700m$ square. The nodes were deployed with a uniform density and each node had a transmission range of about $40m$, which resulted in approximately 10 neighbors per node.

We evaluated our ALS algorithm with $k = 2$ and compared it with two other baseline algorithms: (a) single path: selecting the path with the highest average end-to-end PDR; (b) double disjoint paths: selecting two paths that are disjoint and have the top $PDRs$. We denote those three algorithms as `AvPath, single,` and `double`, respectively.

We studied the following scenarios: no jammer, one stationary jammer, two stationary jammers, and a mobile jammer with two types of moving patterns. For each scenario, we ran our experiment 100 times to collect the statistical characteristics. For each simulation run, the nodes built their neighbor tables and history tables prior to time $t_1 = 600s$. At time t_1, all algorithms selected paths based on the link information observed so far. Then the average availability between a pair of nodes (S and D) that were approximately 20 hops apart was measured for 600 seconds, and the normalized availability is depicted.



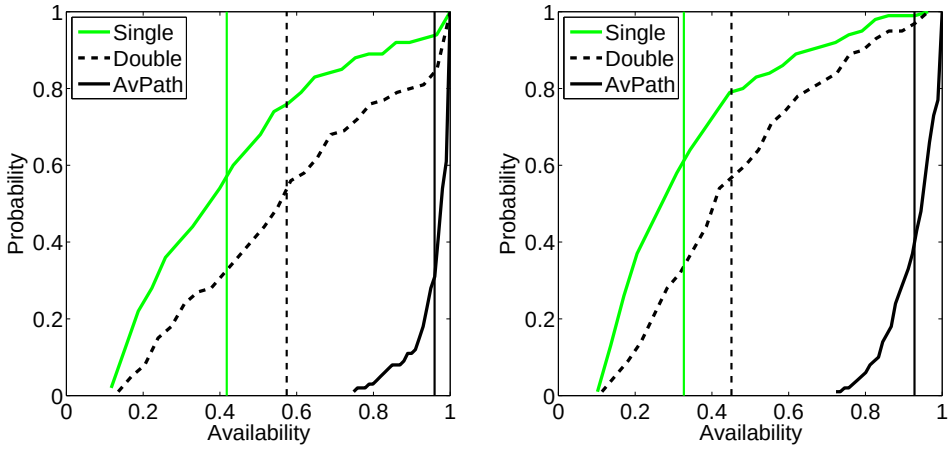
(a) JTR is the same (b) JTR is 50% higher

(c) JTR is 100% higher (d) JTR is 200% higher

Figure 2.9 The CDF of end-to-end availability of ALS for one jammer scenarios where Jammer Transmission Range (JTR) is the same or larger than regular network nodes.

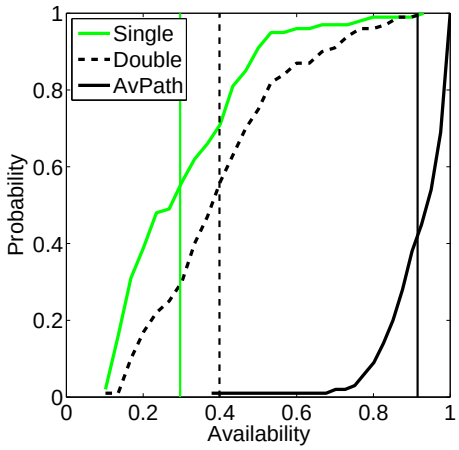
Evaluation Results

No Jammer. Figure 2.8 shows the histogram of the end-to-end PDR and Cumulative Distribution Function (CDF) of the normalized availability between node S and node D with no active jammers. The vertical lines are the average normalized availability for the three cases. Overall, selecting two disjoint paths did increase the average availability slightly: by 4%. However, by selecting two fault-independent paths, our ALS algorithm boosted the average availability more: from 75% to 98%.

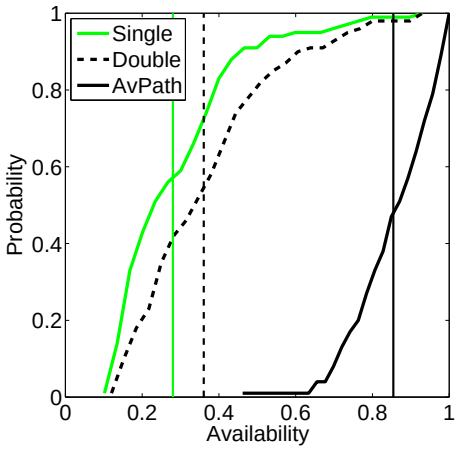


(a) JTR is the same

(b) JTR is 50% higher



(c) JTR is 100% higher



(d) JTR is 200% higher

Figure 2.10 The CDF of end-to-end availability of ALS algorithm in two-jammer cases with various Jammer Transmission Range (JTR).

This result suggests that the AHV-based algorithm can improve network communication even without the disturbance of jamming.

Stationary Jammers. In this set of experiments, we studied two scenarios: a *one-stationary-jammer scenario* and a *two-stationary-jammer scenario*. In both scenarios, the stationary jammers were present at the beginning of the simulations, and they alternated between ON and OFF for random amount of time, e.g., a random duration uniformly distributed between 5 and 20 seconds. The jammers were placed somewhere on the shortest path between nodes D and S, so that it would affect the

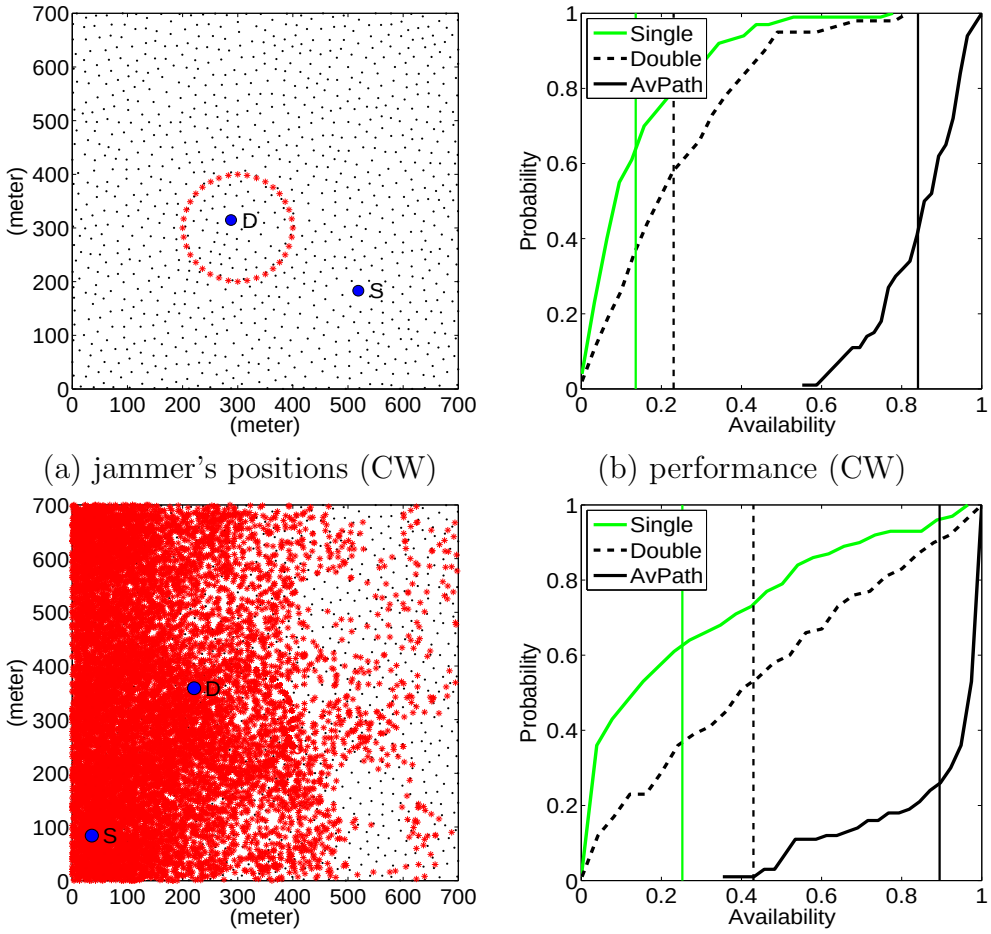
shortest path between the source and the destination. We assumed that the jammers were capable of transmitting at a higher power level than the network nodes, and we evaluated cases when the jammer had (i) the same, (ii) 50% more, (iii) 100% more, and (iv) 200% more transmission range than the network nodes. The average availability between S and D for the one jammer case is depicted in Figure 2.9, and two jammer case is depicted in Figure 2.10. In all cases, the ALS algorithm outperforms the other two baseline algorithms by $60\% - 70\%$.

Additionally, the average availability of all algorithms decreases as the transmission range of the jammer increases, since a larger jamming range will affect more paths and will reduce the end-to-end availability.

Mobile Jammers. In our experiment, we studied two types of mobile jammers: one traveling in a circle (Circular Walk or CW jammer) and the other moving randomly (Random Walk or RW jammer), as illustrated in Figure 2.11(a) and (c). Regardless of their moving patterns, both mobile jammers' transmission range is 100% more than network nodes, and both jammers remained active throughout the simulation.

Specifically, the circular-walk jammer constantly disturbed communication between node D and node S, as it hovered around the destination node D (affecting node D about 40% of the time). As a result, the single-path algorithm generated a path that is only available 18% of the time, on average. The double-disjoint-path algorithm selected two paths that in combination have slightly higher availability than the single path, even though twice the number of paths were used. In comparison, the `AvPath` also selected two paths but two fault-independent paths, and thus the availability of those paths is 60% more than the other two algorithms.

Finally, the RW-mobile jammer represents a jammer whose behavior is not fully captured in the availability history prior to multipath selection, i.e., it creates 'future' failures. The simulation results in this case show that the ALS algorithm can improve



(a) jammer's positions (CW) (b) performance (CW)

(c) jammer's positions (RW) (d) performance (RW)

Figure 2.11 The two types of mobile jamming scenarios: a circular-walk jammer (CW) and a random-walk jammer (RW). Here in (a) and (c), the red (shaded) dots denote the positions of the mobile jammer; (b) and (d) show the CDF of end-to-end availability of ALS for CW and RW jammer respectively.

the availability even when unexpected new faults may appear after multipath selection is done. This is because the failure-dependence caused by jamming and other factors is also affected by the positions of network nodes, radio propagation environments, etc. Although the 'future' failure has not occurred yet, the influencing factors have already been partially encoded in the historical failure correlation implicitly.

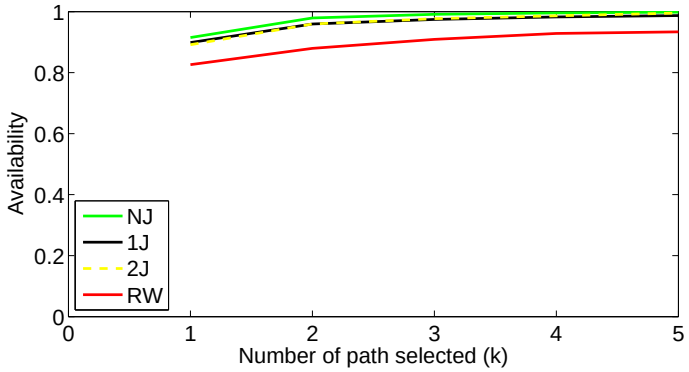
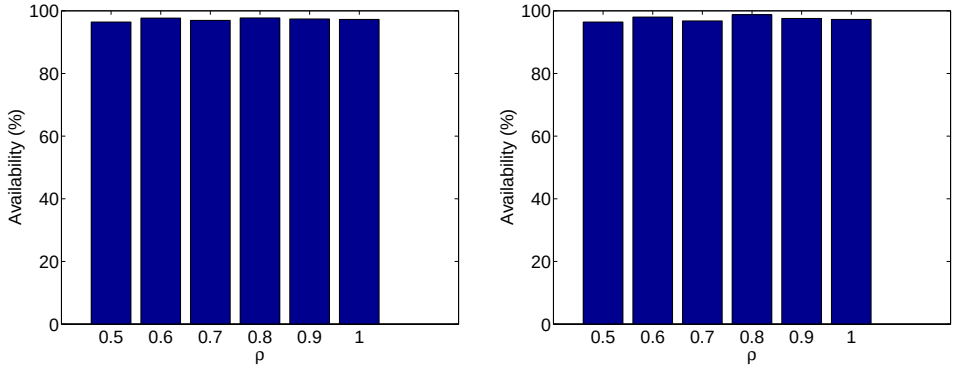


Figure 2.12 The average end-to-end availability of ALS algorithm for different jamming scenarios as number of paths selected by the algorithm increases.

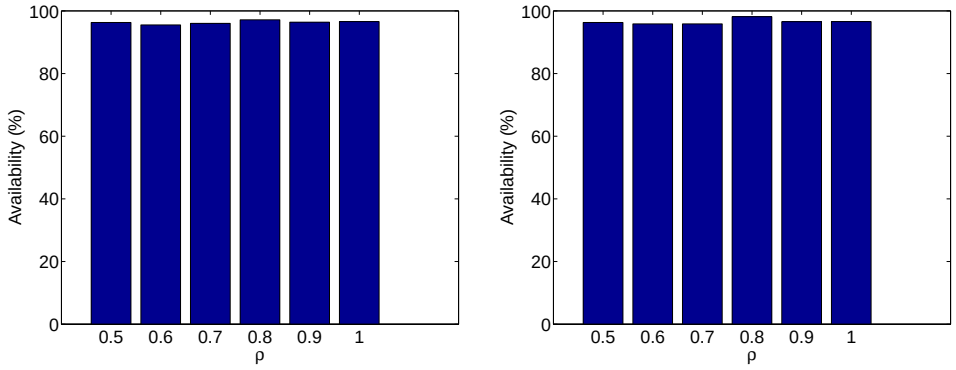


(a) 2 paths selection (b) 3 paths selection

Figure 2.13 The average end-to-end availability of ALS algorithm in no jammer cases for different values of ρ where 2 paths were selected in (a), and 3 paths were selected in (b).

Availability with Additional Paths.

We did most of our simulation setting $k = 2$, i.e., two failure-independent paths are chosen by the ALS algorithm. To find out the relation between availability and k, we ran some more simulations varying k. The result is depicted in Figure 2.12. From the figure, we can see that the end-to-end availability increases with k while it is evident that setting $k = 2$ is enough as the higher values of k does not gain significant increase in end-to-end availability.



(a) 2 paths selection (b) 3 paths selection

Figure 2.14 The average end-to-end availability of ALS algorithm in one jammer cases for different values of ρ where 2 paths were selected in (a), and 3 paths were selected in (b).

Analysis of ρ

As we discussed in Section 2.4, when multiple paths are selected by ALS algorithm, the parameter ρ controls the level of overlapping between paths. Thus, the selected paths are node-disjoint when $\rho = 0$, and do not have any disjointness restriction when $\rho = 1$. To find the appropriate value of ρ, we ran a set of experiments for no jammer and one jammer scenarios by varying the value of ρ. In particular, we ran experiments for six discrete values of ρ which are 0.5, 0.6, 0.7, 0.8, 0.9, and 1.0. We depict the average end-to-end availability for the no jammer cases in Figure 2.13. As we can see, our ALS algorithm performs well for all the values of ρ, however it worked best when $\rho = 0.8$. The average end-to-end availability for the one jammer scenarios is depicted in Figure 2.14. As we can see, the same observation holds for one jammer scenarios as well. Consequently, we choose $\rho = 0.8$ in ALS algorithm. Even though we use $\rho = 0.8$ in all our experiments, we note that ρ is a tunable parameter and can be set depending on the network size and density.

The evaluation results for the AHV-based multipath selection are promising and encourage us to design a distributed routing algorithm utilizing Availability History Vectors (AHVs). Note that the AHV-based multipath selection can be applied to a wide variety of routing algorithms for wireless ad hoc networks, such as Dynamic Source Routing (DSR) [67] and Ad-Hoc On-Demand Distance Vector Routing (AODV) [36]. Because of the source routing feature, the integration to DSR is less involved than AODV. Nevertheless, we illustrate the key requirements of applying AHVs, using AODV as a case study; we call the integrated algorithm the AHV-Enhanced AODV (AvAODV) routing. We briefly review the key elements of AODV and then present the AvAODV routing.

AODV

AODV is one of the classic on-demand routing protocols for wireless ad hoc networks. It specifies two operations for routing: route discovery and route maintenance. To discover a route to a specific destination, the source node broadcasts a Route Request (`RREQ`) packet with a unique broadcast-ID and its own address. Upon receiving the `RREQ` packet for the first time, the intermediate node forwards the packet, and the destination node replies with a Route Reply (`RREP`) packet. In particular, the `RREP` packet is unicast back to the source via the reverse path that was set up when the `RREQ` packet traveled from the source to the destination. Meanwhile, each receiving node updates its routing table to keep track of the latest route to that destination. As part of the route maintenance, when a link is no longer available, a special Route Error (`RERR`) packet is sent to the affected source nodes. Once receiving the `RERR` packets, the source node initiates another round of route discovery. Additionally, `HELLO` packets are broadcast locally to enable nodes to keep track of its neighbors.

Algorithm Description

After integrating the availability-based multipath routing with AODV, AHV-enhanced AODV (AvAODV) routing protocol will have the following capabilities.

- **AHV Recording.** To utilize availability histories, the AHV of each link will be measured and collected distributedly.

- **Path Pre-Selection.** When a source node wants to discover routes to a specific destination, AvAODV will discover h candidate paths and will report their path AHVs to the source node distributedly. To coordinate the path-AHV collection and to ensure that the AHVs of different nodes are roughly aligned, loose-synchronization is required [95].

- **Multipath Selection.** The source node will select k paths out of the h candidate paths so that those k paths are correlated least in terms of failures, i.e., failure-independent. Moreover, those k paths will be ranked according to their availability histories.

- **Active-Duty Route Switching.** Although multiple paths are selected, only one path will be chosen to deliver packets at a given time; we call this path the *active-duty path*. Initially, the most available path (e.g., top-ranked one) will be used as the active-duty path. After a while, the active-duty path may be unavailable. To maintain the end-to-end availability with small disturbance, the source node will execute a fast route switching strategy to quickly select another available path out of the remaining $k-1$ paths, without another round of route discovery.

Table 2.1 Control packet formats for AHV-Enhanced AODV derived from AODV.

AODV RREQ
AHV
Intermediate node 1
Intermediate node 2
...
Intermediate node i

AODV RREP
AHV
Intermediate node 1
Intermediate node 2
...
Intermediate node i

AHV Recording

To facilitate multipath selection leveraging AHVs, the AHV of each link is measured and recorded locally, and the AHV of each path is calculated on-demand and accumulatively, as the AHV is propagated from one end to the other.

Measure the AHV of a link. In AODV, each node can broadcast `HELLO` packets and create a *neighbor table* collecting the local connectivity information. The AvAODV re-uses the `HELLO` packets to derive the link availability. In particular, a node records an m-bit AHV for each of its neighbors in the neighbor table, and each bit of AHV maps to the availability in one epoch. Whenever an epoch has passed, the node shifts the AHVs to the left by one bit to include the latest availability in the least-significant bit.

Calculate the AHV of a Path. The AHVs of paths is calculated distributedly when a route discovery is preformed. To facilitate AHV calculation, two additional entries are added to the original AODV routing control packets and routing tables: the AHV accumulated so far and the IDs of the nodes on the path. Table 2.1 illustrates sample packet formats for `RREQ` and `RREP` packets, and Table 2.3 shows an example routing table of node 1 in the topology depicted in Figure 2.3[1].

As `RREQ` packets travel from the source to the destination, each node will update the AHV field by performing a bitwise AND with the immediate upstream link AHV,

[1]Note that the routing table includes one more entry, i.e., the priorities of the paths, which is used for fast active-duty route switching.

Table 2.2 An illustration of calculating AHV accumulatively, as RREQ packets travel along Path 1 (Figure 2.3).

Link	Link AHV	Path	Path AHV
-	-	-	0xFFFFFFFF
1-2	0xFFFFFFF7	1→2	0xFFFFFFF7
2-3	0xFFFFFFEF	1→2→3	0xFFFFFFE7
3-8	0xFFFFFFFB	1→2→3→8	0xFFFFFFE3

and will append its address to the intermediate node list. Algorithm 3 summarizes the forwarding operation at each intermediate node. In addition, when the source node receives an RREP packet, it will add the path reported in this RREP packet to its routing table.

To walk through an example, consider Path 1 in the topology shown in Figure 2.3. The AHVs for each link and intermediate AHV calculation are listed in Table 2.2. After receiving an RREQ packet from node 1 with AHV=$0xFFFFFFFF$, node 2 will calculate the AHV of path 1→2 as the bitwise AND of the received AHV and the AHV of the link 1→2, i.e., $0xFFFFFFFF \wedge 0xFFFFFFF7 = 0xFFFFFFF7$. Then, node 3 will derive the AHV of path 1→2→3 as $0xFFFFFFF7 \wedge 0xFFFFFFEF = 0xFFFFFFE7$. Finally, node 8 will update the AHV of path 1→2→3→8 as AHV = $0xFFFFFFE3$.

Algorithm 3 RREQForward: AVH-Based RREQ Forwarding

Require: INPUT:
$P : RREQ\_PACKET$
PROCEDURES:
1: **if** ((ForwardingCount($P.id$) $\leq MaxForward$) &&
($|P.AHV| > \gamma_o$)) ||
($P.destination \in this.neighbors$) **then**
2: **if** $this.IP \cap P.nodeList = \emptyset$ **then**
3: *update* $P.AHV = P.AHV \wedge AHV_{ReceivingLink}$
4: *update* P = AddIPAddress($P, this.IP$)
5: *forward* P
6: IncreaseCount($P.id$)
7: **end if**
8: **end if**

Table 2.3 The routing table for node 1 (Figure 2.3) containing paths to node 8.

Destination	Path	AHV	Priority
8	1→2→3→8	0xFFFFFFE3	1
8	1→4→5→8	0xFFFCFFBD	3
8	1→6→7→8	0xFDFFFAE0	2

Path Pre-Selection

Since AODV aims to select the shortest path, forwarding `RREQ` packets with the same broadcast-ID once suffices for discovering one path. However, such restriction cannot be applied to AvAODV since the goal of AvAODV is to choose several paths and to record their end-to-end availability. Therefore, each node has to forward `RREQ` packets with the same broadcast-ID several times, provided the intermediate nodes are different.

Essentially, the path pre-selection algorithm is performed distributedly as each node determines whether to forward an `RREQ` packets or not. For instance, on one extreme, if the intermediate node always forwards an `RREQ` packet containing a new path, then the source node will receive all possible routes, at the cost of excessive `RREQ` packets. To balance the communication overhead and performance, we apply the following heuristics.

- To reduce the total number of broadcast packets for each route discovery, each node is only allowed to forward the `RREQ` packets with the same broadcast-ID at most `MaxForward` times. As a result, AvAODV tends to select paths with reletively small hop counts.

- Each node will forward an `RREQ` packet only if the availability of the AHV reported in the packet is greater than a threshold, γ_o. Thus, AvAODV reduces unnecessary control overhead by actively discarding paths at each intermediate node if the discovered paths have low availability.

- A node always forward RREQ packets regardless of the aforementioned two restrictions, if the destination node is one of its neighbors. The motivation is that such an RREQ packet has collected almost the entire information of a path, and the benefit of forwarding it exceeds the concern of increasing the control overhead.

The path pre-selection process ends at the destination node. After receiving the RREQ packet that contains a new path, the destination node creates an RREP packet by copying the intermediate node list and the end-to-end AHV from the RREQ packet, and then unicasts it back to the source.

Greedy Multipath Selection

After receiving h RREP packets, the source node will start the greedy multipath selection by executing Algorithm 2. It will select k paths that in combination produce high availability to the destination. Additionally, it will rank the k paths based on their individual availability, θ.

Active-Duty Route Switching

Out of k selected paths, only one path will be chosen as the *active-duty path*. Right after performing the greedy multipath selection, the path with the highest θ will be the active-duty route. To cope with cases where the active-duty path becomes unavailable, AvAODV contains a fast route switching strategy that can quickly select another active-duty route, without initiating the route discovery. In particular, the AvAODV utilizes RERR packets to inform the source about the broken path and to report the corresponding AHVs. In particular, once a node on the active-duty path detects that its downstream link is no longer available, it initiates an RERR packet containing the latest AHV. Each node that receives the RERR packet updates and forwards the RERR packet. Upon receiving the RERR packet, the source updates the

routing table and immediately selects the highest priority path from the remaining paths. As such, a route discovery is avoided, and it is foreseeable that AvAODV can reduce the route discovery overhead when links tend to fail frequently.

Algorithm Evaluation

In this section, we evaluate the performance of AvAODV protocol. Without loss of generality, we studied the performance of AvAODV when choosing at most two paths.

Simulation Methodology

Network Setup. We studied the AvAODV in ns-3 [89], a packet-level network simulator. In particular, we simulated a network of 100 nodes placed in a $2000m$-by-$2000m$ square, where each node had a transmission range of $300m$. Similar to the setup for evaluating the ALS algorithm, each node was positioned randomly yet resulted in a uniform density, i.e., roughly 10 neighbors per node.

In total, we compared the performance of the following three algorithms: (a) the original `AODV`, (b) the `AvAODV` algorithm with $k = 1$, (c) the `AvAODV` algorithm with $k = 2$. The parameters selected for simulation setup are summarized in Table 2.4. For all algorithms, every node sent one `HELLO` packet per second to build up the neighbor table. At the 200^{th} second, a source node S started to unicast UDP messages to a destination node D at a constant rate (CBR traffic) until the 500^{th} second. The locations of nodes S and D are illustrated in Figure 2.15. Unless specified, we focused on the scenarios of one data stream between S and D. In the later section, we examined multiple streams between multiple node pairs.

For both `AvAODV` protocols, in every epoch (5 seconds), each node calculated the average PDR and converted it to availability using a threshold value of 0.6. When the source node S discovered that it did not have a route to the destination node D, it would initiate a route discovery. As the `RREQ` packets flood the network, each

Table 2.4 Simulation parameters and values used in NS-3 for evaluation.

Variable Name	Value
`MaxForward`	10
`HELLO_INTERVAL`	1s
`Availability Threshold,` γ_o	0.6
`AHV Epoch`	5s
`Simulation time`	500s
`WiFi:FragmentationThreshold`	2200 bytes
`WiFi:RtsCtsThreshold`	0 bytes

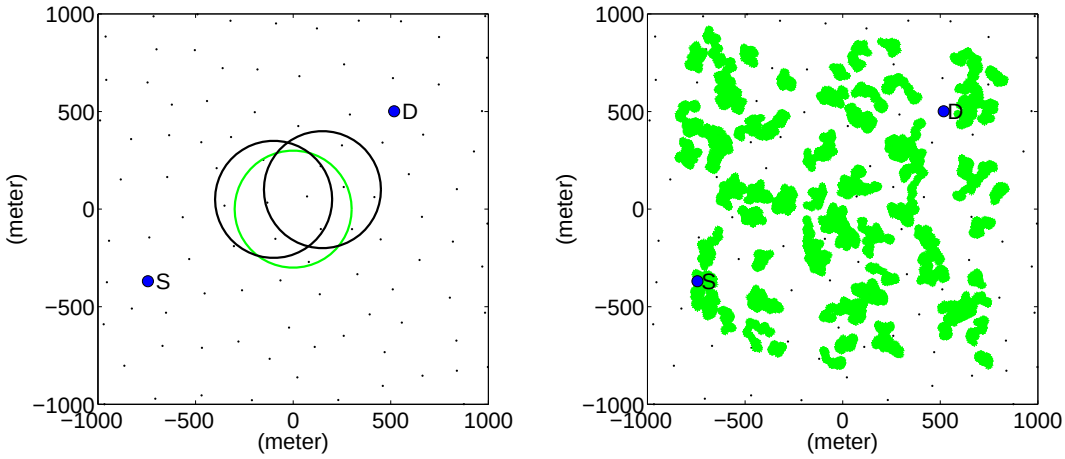
intermediate node was allowed to forward up to 10 `RREQ` packets for the same route discovery. Thus, the node S will receive more than 10 candidate paths.

Similar to the evaluation setup of ALS algorithm, we studied the following scenarios: no jammer (NJ), one stationary jammer (1J), two stationary jammers (2J), and a mobile jammer walking randomly (RW). In all scenarios, the transmission power of the jammers is the same as the regular network nodes. We customized those jammers by using the ns-3 jamming package developed by the Network Security lab at the University of Washington [86]. For each scenario, we ran our experiment 50 times to collect statistical characteristics.

Evaluation Metrics. We examined both the performance and the overhead of all three routing algorithms.

Performance. Since the responsibility of networks is to deliver packets, we used the average end-to-end PDR between the source node S and the destination node D to evaluate the performance of routing algorithms. If node D received no messages in a measurement interval, then the end-to-end PDR was considered as 0.

Control Overhead. Collecting information of multiple paths may impose additional overhead compared to the baseline `AODV` protocol. To evaluate the overhead for each algorithm, we measured all routing control packets (`RREQ`, `RREP`, and `RERR`) broadcasted across the entire network, and calculated the normalized control overhead in terms of packet counts and packet sizes, e.g., the average packet counts.



(a) stationary jammers (b) a RW jammer

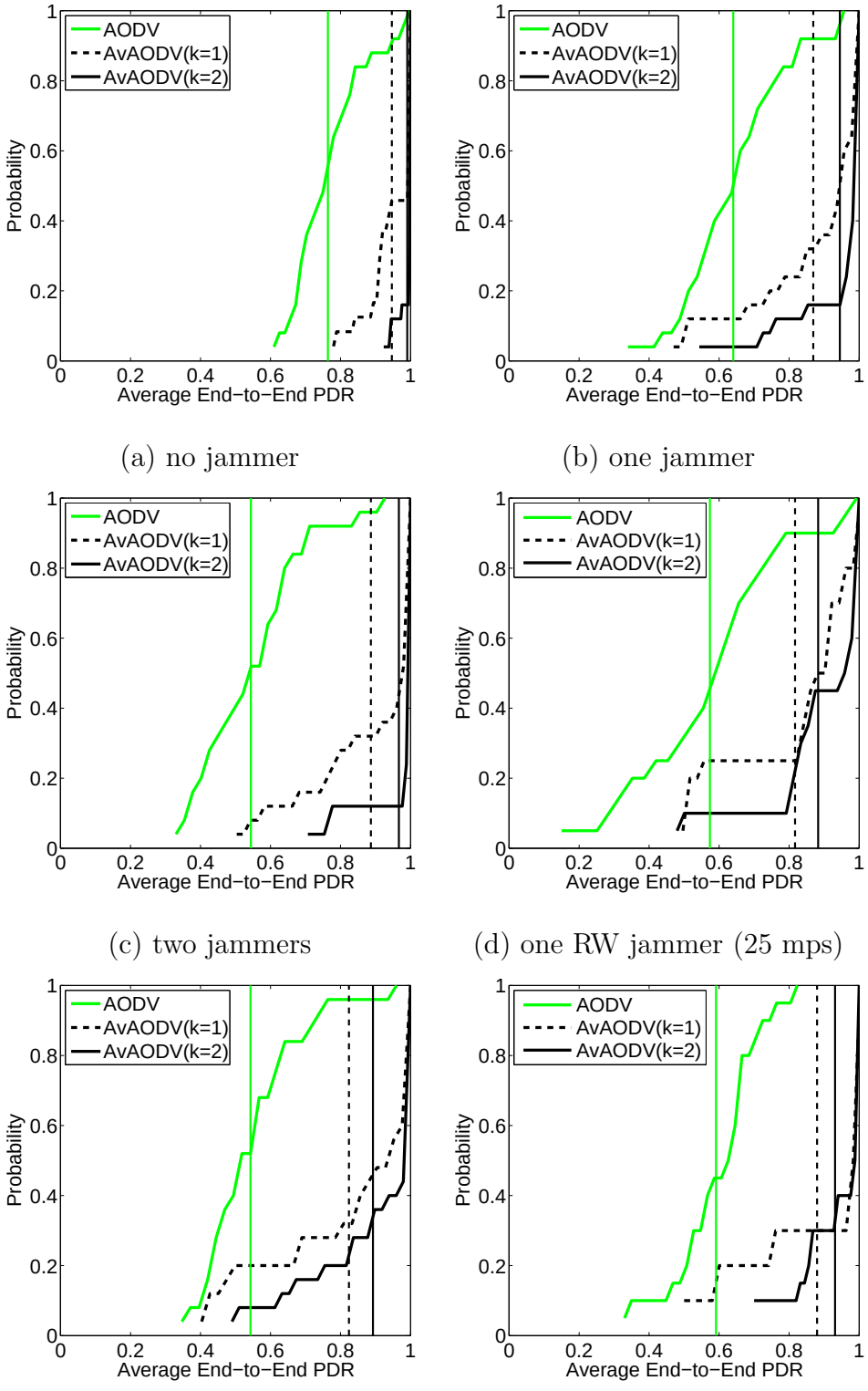
Figure 2.15 The positions of jammers for the stationary jammers and the mobile jammer: (a) Stationary jammers. The green (shaded) circle shows the approximate area covered by a single stationary jammer and the two black circles shows the areas affected by two stationary jammers; (b) A mobile jammer. The green (shaded) dots denote the positions of the mobile jammer.

Simulation Results for Performance

The average end-to-end PDR between a pair of nodes S and D for various scenarios are depicted in Figure 2.16, and the results of multiple pairs of nodes are summarized in Figure 2.17. We discuss the cases for no jammer, stationary jammers, mobile jammers, and multiple concurrent streams in the following.

No Jammers. As expected, even in cases when no jammer was present, `AvAODV` outperformed `AODV`, as shown in Figure 2.16 (a). Although `AvAODV (k=1)` selected one path, it achieved an average PDR that is 17% better than AODV. This is because `AvAODV (k=1)` selects the path that has the highest availability in the past rather than the path with least hop count. Moreover, selecting two paths, `AvAODV (k=2)`, can boost the average PDR by 22% over `AODV`.

Note that compared with the results of the ALS algorithm in Figure 2.8, `AvAODV` achieved similar performance, which is promising. The ALS algorithm has a global view of the network connectivity and can pick k paths out of a larger candidate pool.



(a) no jammer

(b) one jammer

(c) two jammers

(d) one RW jammer (25 mps)

(e) one RW jammer (50mps)

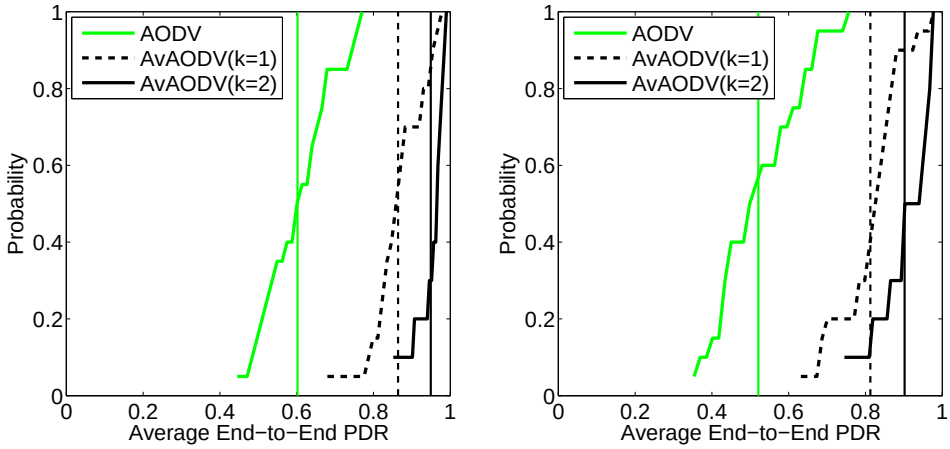
(f) one RW jammer (75mps)

Figure 2.16 Performance comparison between `AODV` and `AvAODV`: the CDF of end-to-end PDRs for various jamming scenarios.

In comparison, AvAODV relies on the route discovery to search for a few candidate paths. The concern of prohibitive overhead and packet collisions restricts the total number of paths discovered. Despite the limited number of discovered paths, AvAODV can still perform as good as the centralized algorithm.

Stationary Jammers. For stationary jamming scenarios, we considered a one-jammer case and a two-jammer case. In either case, one jammer alternated between ON and OFF state every 20s, while a second jammer was always ON in the two-jammer case. The stationary jammers were placed between node S and node D so that they woulld disturb the shortest path in between, as illustrated in Figure 2.15 (a). Figure 2.16 (b-c) depicts the corresponding results and demonstrates that both AvAODV protocols can improve the average PDR by at least 32% in the presence of jamming. Furthermore, since a larger area of the network was disrupted in two-jammer cases than in one-jammer cases, the average PDR obtained by AODV in the two-jammer case was 9% lower than the one-jammer case. In contrast, AvAODV can accomplish a higher PDR in the two-jammer case. This is because a stronger jammer reduced control overhead of AvAODV (which will be explained in Section 2.5) and helped to choose paths with high availability. This suggests that the stronger the jammer is, the higher resilience the AvAODV protocols may achieve.

Mobile Jammers. In the stationary-jammer scenarios, we studied the cases where future faults were already embedded in the AHVs. To examine the situation that new faults may appear after the multipath selection, we implemented a mobile jammer that moved randomly within the network area, and called it a random-walk (RW) jammer. The moving speed of the RW jammer was set to $\{25, 50, 75\}$ meter per second (mps). Figure 2.15 (b) shows a position trace of a RW-jammer in one of the simulation runs, which shows that the RW-jammer is highly unpredictable. Even in such a case, the AvAODV (k=2) provides an average PDR 27% higher than AODV for the RW jammer with various speeds, as shown in Figure 2.16 (d-f).



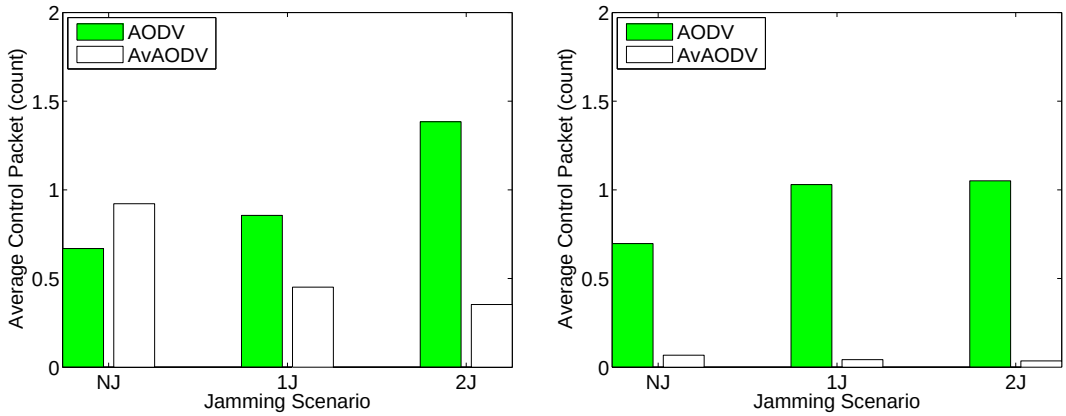
(a) two jammers (b) one RW jammer (50mps)

Figure 2.17 Performance comparison between `AODV` and `AvAODV` for two concurrent data streams between two pairs of source-destination nodes.

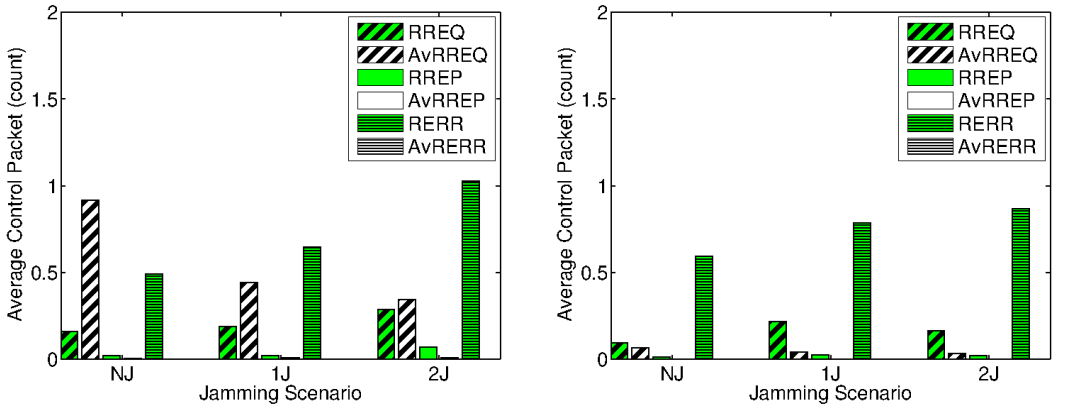
Interestingly, we found that the performance of AvAODV increased slightly as the speed of the RW jammer increased. This is because given the same amount of time, a faster jammer affects a larger number of links than a slower one. Thus, its jamming impact is likely to be recorded in a larger number of AHVs, which enables AvAODV to select multiple paths with a higher level of failure-independence based on AHVs of the pre-selected paths.

Multiple Concurrent Streams. In this set of experiments, two pairs of source-destination nodes maintained two streams (CBR traffic) simultaneously, and the locations of the second node pair are randomly chosen such that the candidate paths for both streams may overlap.

The experimental result for multiple concurrent streams is depicted in Figure 2.17, where (a) shows the two-jammer scenarion and (b) shows one RW-jammer scenario. As we can see, the `AvAODV` ($k = 2$) outperforms `AODV` by at least 32%, similar to the improvement in one-stream scenarios.



(a) all control packets (**500s**) (b) all control packets (**3200s**)

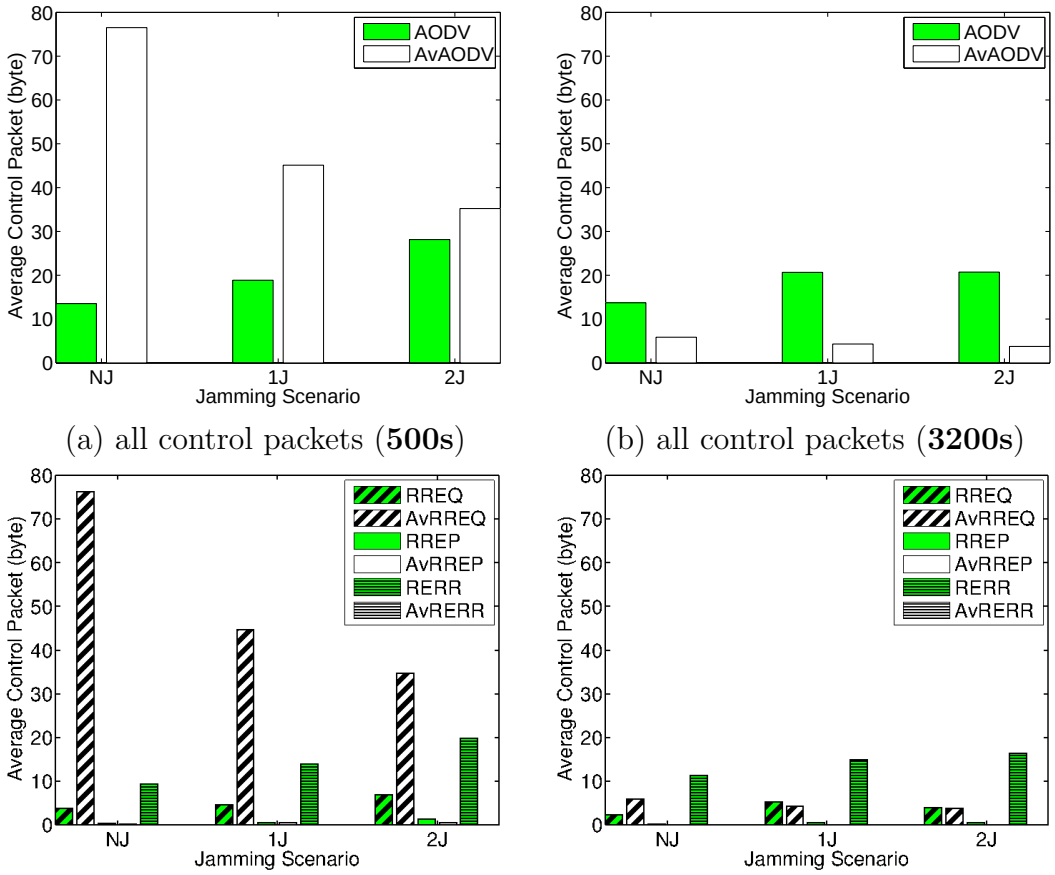


(c) categorized packets (**500s**) (d) categorized packets (**3200s**)

Figure 2.18 Control overhead comparison between `AODV` and `AvAODV` (k = 2) for the simulation time 500s and 3200s. It shows the control overhead by packet counts (e.g. the numbers of control packets per second per node). `Rxxx` stands for the `Rxxx` packets in `AODV`, and `AvRxxx` stands for the `Rxxx` packets in `AvAODV`.

Simulation Results for Control Overhead

To evaluate the control overhead, we measured the total number of `RREQ`, `RREP`, and `RERR` packets sent by every node across the entire network, when `AODV` and `AvAODV` algorithms (k = 2) were used, respectively. Additionally, to understand whether the duration of a communication session affects the control overhead, we measured the routing overhead for both short-lived and long-lived sessions, i.e., S kept sending UDP messages to D until the 500^{th} second and until the 3200^{th} second, respectively. Figure 2.18 depicts the total numbers of all control packets, and individual numbers of



(a) all control packets (**500s**) (b) all control packets (**3200s**)

(c) categorized packets (**500s**) (d) categorized packets (**3200s**)

Figure 2.19 Control overhead comparison between `AODV` and `AvAODV` (k = 2) for the simulation time 500s and 3200s. It shows the control overhead by packet sizes (e.g., (e.g. the bytes of control packets per second per node). `Rxxx` stands for the `Rxxx` packets in `AODV`, and `AvRxxx` stands for the `Rxxx` packets in `AvAODV`.

packets in each category (i.e., `RREQ`, `RREP`, and `RERR` packets) for both types of sessions. We also analyzed the control overhead in terms of packet size. Figure 2.19 depicts the total size of all control packets, and individual sizes of packets in each category (i.e., `RREQ`, `RREP`, and `RERR` packets) for both types of sessions. In the no-jammer cases, for the short-lived sessions, `AODV` sent 27% less control packets than `AvAODV`. However, for the long-lived sessions, `AODV` sent 932.7% more control packets than `AvAODV`. Examining the categorized control overhead reveals the underlying causes: For each round of route discovery, `AvAODV` sent a larger amount of `RREQ` packets than

AODV as a result of allowing each node to forward multiple RREQ packets. Over the entire duration of a session, less numbers of route discoveries were performed and less RERR packets were generated for AvAODV than for AODV, because AvAODV can switch the active-duty route quickly without initiating route discovery when a link failure occurs. Thus, on average, AvAODV incurs a less amount of overhead, when node pairs tend to communicate for a long period of time.

In the presence of one or more jammers, AODV generated 89% (one jammer) or 295% (two jammers) more control packets than AvAODV for the short-lived sessions and incurred 23 times more or 29 time more overhead for the long-lived sessions. Interestingly, for both types of sessions, the numbers of control packets of AODV increased as an increasing number of jammers interfered the network, while the numbers of control packets of AvAODV decreased. When the jammers disturbed the network communication, the delivery failures will incur RERR packets. As a result, the numbers of RERR in AODV increased as the level of disturbance increased. In comparison, AvAODV chose paths that have failed the least in the past. It excluded those that have been and would be affected by jamming heavily, greatly reducing the number of RERR packets. Finally, AvAODV produced less RREQ packets with the increasing amount of interference from jammers, because intermediate nodes will not forward a RREQ packet if its availability was lower than the threshold value. Considering that we have added the AHV and complete paths into control packets for AvAODV, we compared the total bytes sent by both AODV and AvAODV algorithms. As shown in Figure 2.19(a)-(d), although in the short-lived sessions, AvAODV sent more bytes for routing, in the long-lived sessions, the overhead of AODV are 13 times, 38 times, or 45 times more than AvAODV for no jammer, one jammer, and two jammer cases.

In summary, by choosing multiple fault-independent paths, AvAODV achieves a much higher PDR than AODV at a modest cost for short-lived session and at a lower cost for long-lived sessions.

While we present a jamming-resilient multipath routing scheme maximizing end-to-end availability between source-destination pair, there has also been extensive study using multipath routing in benign scenarios without jamming attacks. We discuss some multipath routing schemes in the following.

Multipath Selection

The challenges of employing multipath routing have been recognized [83], and a number of protocols focus on how to efficiently discover and select multiple paths. For example, CHAMP [115] uses cooperative packet caching and shortest multipath routing to reduce packet loss due to frequent route breakdowns. However, CHAMP simply selects the shortest multiple paths with equal length. Wu et al. [127, 128] proposed to use IP-layer topological disjointness and connectivity as an indicator of the potential interference across paths, and select multiple paths that are the 'least connected' at the IP-layer topology. Similarly, TORA [94] uses multiple topological disjoint paths to increase network adaptivity; Nasipuri et al. [85] extended DSR with multipath routing by which each source node can learn a set of topologically disjoint paths; and Marina et al. [78] proposed AOMDV which is a multipath extension for AODV that computes multiple IP-layer link-disjoint paths. Furthermore, Ye et al. [140] proposed to strategically deploy some "reliable" nodes to maximize path diversity in the topology. However, as argued by Zhang et al. [143], this IP-layer disjointness cannot reflect interference and failure correlation between routes, such as physical-layer correlation due to jamming attacks.

Multipath Utilization

As a separate line of research, various approaches have been proposed to minimize congestion or achieve security via multipath routing, assuming multiple paths are already selected. For example, Zhang et al. [141] studied how to distribute traffic along the already selected multiple paths to achieve load balancing. Tsirigos et al. [114] developed a theoretical model and coding scheme to spread traffic across the given disjoint paths in order to maximize the packet delivery ratio. Setton et al. [106] proposed an analytical model to minimize network congestion based on the global real-time traffic distribution in the network. Similarly, MR-CS [26] relies on centralized scheduling to achieve load balancing, QoS, and throughput optimization on the given multiple paths. In addition, SPREAD [75] uses multipath routing to protect data confidentiality; it transforms a secret message into multiple shares, and then delivers the shares via multiple paths to the destination so that even if a certain number of message shares are compromised, the secret message as a whole is not compromised.

Sensor Networks

Researchers have proposed multipath routing schemes tailored for sensor networks, where the base station is the common destination, sensor nodes are grouped in clusters and are extremely energy and storage constrained. For example, M-MPR [37] leverages a meshed multipath routing together with selective forwarding and forward-error correction to increase end-to-end throughput. Baek et al. [11] and Liant et al. [71] propose different multi-pathing schemes to minimize energy consumption and storage expense. Finally, I2MR [112] uses zone-disjoint paths to minimize cross-path interference in a sensor network which is divided into multiple zones.

Jamming Resilience

Jamming and radio interference are known threats and have attracted much attention. Jamming detection was studied by Xu et al. [138] in the context of commodity wireless devices, and was also studied in the context of sensor networks [24].

Countermeasures for coping with jamming have been intensively investigated. Traditional PHY-layer techniques provide resilience to interference [98] at the expense of advanced transceivers. For commodity wireless networks, defense strategies include the use of error correcting codes [88] to increase the likelihood of decoding corrupted packets, channel hopping [136] to adapt the working channel to escape from jamming, anti-jamming timing channels [137], wormhole-based anti-jamming techniques [22], and binary-key-tree-based mitigation schemes [29]. Additionally, the combination of mask framing, frequency hopping, packet fragmentation, and redundant encoding techniques is proposed to cope with multiple types of jammers [126].

In the area of routing, to the best of our knowledge, few multipath routing proposals addressed the jamming resilience. Recent work by Tague et al. [109][110] studied the optimal traffic allocation scheme over already selected multiple paths and aimed at maximizing the throughput in the presence of jammers; while our scheme effectively incorporates jamming-resilience into multipath selection.

2.7 Summary

In this chapter, we first addressed the problem of multipath selection with the goal of improving jamming resilience in wireless networks. Our key insight is to select multiple paths that are unlikely to fail concurrently, based on the knowledge of paths' availability histories. The availability histories of paths are efficiently recorded and calculated via availability history vectors (AHVs). Leveraging AHVs, we presented two AHV-based multipath selection algorithms: one selects multiple paths with the full knowledge of AHVs in the network, and the other computes the path in a dis-

tributed manner. Our simulation results have validated that AHV-based algorithms can effectively identify multiple paths that provide high end-to-end availability, even in the presence of a new jammer that did not affect the network before path selection. Additionally, the proposed distributed AHV-based algorithm accomplishes higher availability than AODV at a smaller communication cost for long-lived communication sessions. Compared with schemes utilizing jamming models, our AHV-based algorithms work in a plug-and-play fashion and are resilient to a wide variety of jammers.

Chapter 3

CallerDec: End-to-End Detection of Caller ID Spoofing Attack

3.1 Introduction

In a caller ID service, a telephone carrier transmits the phone number and/or the name of a caller to the recipient (callee) side as caller ID. While a call is being dialed, the caller ID is automatically extracted and transmitted to the callee during the signalling phase, and is displayed at the receiving phone. The caller ID service was first introduced in 1969 in several patents [93][92]. Since then leading telephone companies started to design their own caller ID protocols. For instance, Bellcore [14] and SIN227 [20] are the de facto caller ID standards for landline services in America and Europe. In addition, dominating cellular standards (e.g., GSM and CDMA) and VoIP have designed their own built-in caller ID protocols.

Regardless of the protocol types, caller ID service is intended to provide informed consent to the receiver before answering calls. Recently, caller ID has also been used to authenticate users in several automated systems: (a) In the *9-1-1 emergency service*, the authority relies on the caller ID to obtain the physical location of the caller; (b) In *automatic telephone banking systems,* caller ID is used by the authentication process to grant customers access to their accounts[1]; (c) In *voicemail service*, some telephone providers grant users access to their voicemail boxes based on their caller ID [7].

[1]For example, Bank of America only requires a customer to enter a debit/credit card number to access account information when the caller ID matches their records.

However, existing caller ID protocols lack authentication mechanisms and hence are vulnerable to spoofing attacks; i.e., an attacker can send a fake caller ID to a callee. As such, caller ID is untrustworthy for authenticating callers' locations or identities. This vulnerability has already been exploited in variety of misuse and fraud incidents: (a) In 2010, a group of police officers arrived at a residence in response to a 9-1-1 call, prepared to rescue hostages from armed criminals. However, the call was the result of *swatting*, in which pranksters spoof the caller ID and appear to be calling near the house of the victim [35]. As a consequence, the unaware victims were at risk of injury, and the police were tied up responding to a prank; (b) Spoofers were reported to have stolen sensitive personal and financial information using fake caller ID [5]; (c) In the US, thousands of people were victimized by credit card fraud with the help of caller ID spoofing [100, 5], causing a loss of more than $15 million dollars annually; (d) Drugs were misused as a result of spoofed pharmacists' phone numbers [100]; (e) Other incidents include identity theft, dangerous fire prank at several hotels [6], purchase scams [104], etc.

Caller ID spoofing has become a real threat to the phone user and needs to be addressed [47]. The existing caller ID systems were designed based on strong assumptions that (a) the telephone infrastructure is tightly controlled, and no intruders could tap into the infrastructure to create an arbitrary caller ID, and (b) the telephone service providers are trustworthy and will not manipulate caller IDs. Due to the assumed mutual trust between carriers, caller ID is not authenticated while a call is routed between different carriers; and a callee's carrier will simply accept the caller ID claimed by a caller's carrier. In early days, since the phone network used dedicated lines operated by a monopoly, this trust model was reasonable. However, today with current converging phone/data networks, trust relations are weak, and equipment hacking has become affordable. Hence an authentication mechanism is required and necessary, but not in place yet. Moreover, leveraging the lack of authentication, a

special type of service provider has emerged allowing their customers to claim their chosen caller ID; a user only needs to dial a special phone number, and then enter the callee's phone number and a chosen fake number. Alternatively, a smartphone user can utilize readily available apps to spoof caller ID (e.g., Caller ID Faker [25]). Finally, caller ID spoofing is trivially possible in VoIP, where many VoIP providers allow users to claim any caller ID through VoIP client software (e.g., x-lite [130]).

Obviously a fundamental solution against caller ID spoofing attacks is to redesign the entire telephone infrastructure and to add built-in caller ID verification mechanisms. However, in today's practice this would not be an option since the telephone infrastructure comprises a variety of technologies that are owned by several telephone carriers with their own trust domains. One proprietary commercial solution, TrustID, offers to detect caller ID spoofing attacks for business customers, but it is not designed for end users [113]. To the best of our knowledge, no mechanism is currently available to end users for detecting caller ID spoofing without answering the call first or without a special interface provided by the carrier, as in TrustID.

We thus focus on detecting caller ID spoofing attacks at end users in an *end-to-end fashion*, i.e., at the caller and callee sides without altering the existing protocols. Such detection mechanisms are challenging to realize: First, only limited information and resources are available at end users. The route of call signalling is unknown. Second, compatibility to different protocols (GSM, VoIP, PSTN) limits the design space. Third, any deviation from the regular calling procedure is unlikely to be accepted by most people. Thus, naive solutions such as rejecting an incoming call and then calling back, are not an option. The detection mechanisms shall be automated and should require few inputs from users. Fourth, a few legitimate services provided by telephone companies allow the caller IDs to be different from the calling numbers, making those caller IDs appear to be spoofed. However, those scenarios should not be classified as caller ID spoofing attacks. We address all these requirements and design

an end-to-end caller ID verification scheme which we call `CallerDec`. We summarize our contributions as follows:

- We analyze the problem of caller ID spoofing and survey available means to launch caller ID spoofing attacks. The key weakness in this context concerns the interconnection protocols between operators.
- We propose CallerDec, an end-to-end caller ID verification scheme that requires no modification to existing telephone infrastructure and is applicable to calling parties using any telephone services.
- We present two use cases of CallerDec, one for an emergency call scenario (e.g., 9-1-1 call) and the other for a regular call scenario. In both cases, the end users, (e.g., a 9-1-1 service or an individual customer) can utilize CallerDec to verify caller IDs.
- We implement CallerDec for Android-based smartphones and study its performance in various scenarios. Our results show that CallerDec can detect spoofed caller ID effectively and efficiently.

While we implemented CallerDec in Android-based phones as a case study, it can also be integrated in any other telephone devices.

The rest of the chapter is organized as follows. In Section 3.2, we discuss the background of caller ID and describe various caller ID spoofing attacks in Section 3.3. In Section 3.4, we introduce the system model, summarize the requirements, and discuss underlying design principles of CallerDec. Then, we present and evaluate SMS-based CallerDec in Section 3.5. After presenting and evaluating Timing-based CallerDec in Section 3.6, we discuss related works in Section 3.7 and summarize the chapter in Section 3.8.

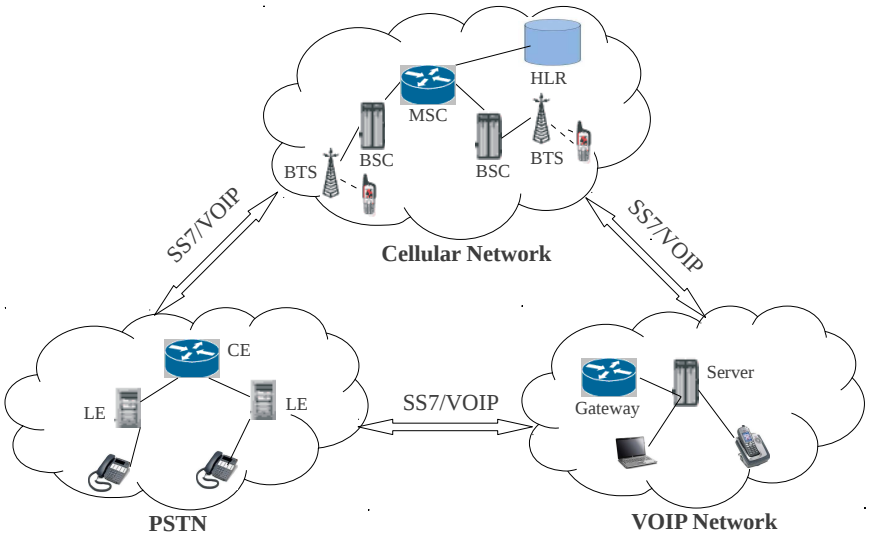


Figure 3.1 An example telephone network architecture, where different carriers are connected using peering architecture. Here, each telephone network follows there own protocol for internal communication, but uses SS7 or VoIP for inter-network communication.

3.2 Background

Three categories of telephone carriers are in service: Public Switched Telephone Network (PSTN), cellular networks, and Voice over Internet Protocol (VoIP) providers. In all these telephone networks, creating a phone call typically involves two types of channels: an end-to-end *control channel* for signalling, and an end-to-end *voice channel* for transmitting voice data. In addition, all telephone carriers support caller ID which works as follows. When a caller dials a number, the carrier first authenticates the caller, and then generates or looks up the associated caller ID. Finally, the caller ID is forwarded to the callee, possibly from one carrier to another.

In the following, we give an overview of the popular caller ID standards used within each type of carrier and between different carriers with the goal of understanding the feasibility of injecting spoofed caller IDs.

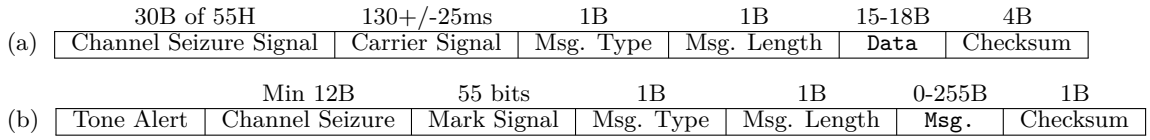


Figure 3.2 Message formats for Bellcore and SIN227 where (a) shows Bellcore and (b) shows SIN227 format. For Bellcore, the caller ID is in the `Data` field and for SIN227, it is in the `Message` field.

Public Switched Telephone Network

Architecture

The PSTN is a circuit-switched telephone network, known as landline telephone. The PSTN generally has a hierarchical architecture [74] with Central Exchanges (CEs) at the top level of the hierarchy, and Local Exchanges (LEs) that provide service in a local neighborhood. Figure 3.1 shows a simple PSTN architecture with one CE and two LEs.

LEs play the key role in the caller ID service. Each LE consists of several PSTN switches and after a customer subscribes for the telephone service, a switch port in the LE is assigned to him/her with the corresponding caller ID, i.e., the customer's phone number, possibly with a name. When a customer dials a number, the LE sends the pre-configured caller ID in the outgoing call.

Protocols

There are several caller ID standards in PSTN, e.g., Bellcore FSK, SIN227, DTMF, V23, ETSI FSK, etc. We introduce two popular standards while skipping the rest, since they work in a similar manner.

`BellCore FSK.` Developed by *Bell Communications Research, Inc.*, Bellcore FSK [14] is the most widely-used caller ID standard (e.g., North America and Asia). At the signal level, the caller ID information is modulated using Frequency Shift Keying (FSK), and the signal is transmitted between the first and second phone ring, while

the telephone unit is still in an onhook state (i.e., still ringing). Figure 3.2(a) shows the message format of Bellcore FSK, whereby the `Data` field contains the caller ID in an ASCII format. The first 8 bytes of the `Data` field record the time of the day, and the remaining 7-10 bytes are used for caller ID.

`SIN227 (Suppliers Information Note).` Designed by *British Telecommunication PLC*, SIN227 [20] is mostly used in Europe. SIN227 also uses an FSK-based analog signal to deliver caller ID. However, unlike Bellcore, a caller ID message in SIN227 may contain the name of a caller in addition to the phone number. As shown in Figure 3.2(b), the `Message` field in a SIN227 message contains the caller ID and has a length up to 255 bytes.

Feasibility of Caller ID Spoofing

In addition to the aforementioned standards in PSTN, all other standards transmit the caller ID information in plaintext. However, it is not easy to launch the attack inside PSTN because the caller ID signal is generated automatically by the LE, based on the pre-configured information. Such information cannot be changed by unauthorized entities, because the switches (LEs) are generally kept in secured cabinets, inaccessible to the general public. However, we identified one particular scenario where an attacker can manipulate the standards to spoof the caller ID, which we will discuss in detail in Section 3.3.

Cellular Network

Architecture

Universal Mobile Telecommunication System (UMTS) [41] and Wide-band Code-Division Multiple Access (W-CDMA) [42] are the two most popular technologies for providing cellular telephone services. Despite which technology is used, a cellular telephone network follows a hierarchical structure. As illustrated in Figure 3.1, the

simplest cellular network consists of the following entities for voice services (from the top to bottom levels): Mobile Switching Centers (MSC), Base Station Controllers (BSC), and Base Transceiver Stations (BTS). The upper level entities control the lower level ones, and the bottom ones (BTS) directly interact with nearby Mobile Stations (MS), i.e., mobile phones. One important entity that assists caller ID services is the Home Location Register (HLR), which interfaces with MSC directly. The HLR stores all necessary data for caller ID services, authentication and billing purposes.

Upon subscription, each customer gets a Subscriber Identity Module (SIM) and inserts it in a mobile station (MS). A mobile carrier authenticates an MS based on the SIM information. When an MS makes a phone call, the call setup process always goes through BTS, BSC, and MSC. Then, the MSC obtains the caller ID associated with the MS from the HLR and encodes it in a control packet for call setup.

Protocols

In UMTS and W-CDMA, the caller ID is encoded in call setup packets (in the `Calling Party BCD Number` field) using the Binary Coded Decimal (BCD) format and has a variable length of 3-14 bytes [3]. It is possible that a call is set up without caller ID, and the `Presentation Indicator` field is used to indicate whether caller ID is present in the packet.

Feasibility of Caller ID Spoofing

3GPP specification has security mechanisms which include MS authentication, random session keys and SIM security [87]. Although the communication between MS and BTS is encrypted with a session key, cracking the session key [18] and man-in-the-middle attacks [80] have been reported. An attacker can take advantage of such vulnerabilities to spoof caller IDs. However, to the best of our knowledge, no caller ID spoofing attacks that take advantage of the 3GPP protocol vulnerabilities have been reported.

Voice over Internet Protocol

Architecture

VoIP technology takes advantage of IP where both voice and control data are transmitted in IP packets. Unlike PSTN or cellular networks, VoIP usually follows a flat/p2p architecture [50] where all the clients and servers must be connected to the Internet. The control channel always follows the client-server model but the voice channel between a caller and a callee may use a direct connection. For the call setup, VoIP uses protocols such as Session Initiation Protocol (SIP) [59] or H.323 [63]. Irrespective of the used protocol, each customer is assigned a user name and a password for authentication and billing purposes, but the customer can often set up any caller ID for an outgoing call.

Protocols

Both SIP and H.323 have built-in support for caller ID. SIP uses the `From` field in the `INVITE` packet to send the caller ID, and the caller ID can be any ASCII characters with an arbitrary length. For instance, the caller ID in SIP has the form `sip:callerID@ip_address` and is typically encapsulated in SIP packets in plaintext. Although secured SIP is available to encrypt caller IDs, they are not authenticated [91]. In H.323, during the call setup, the caller ID is encoded in the `Information Element (IE)` field of a signaling packet in a binary format and it may have variable length. Similar to SIP, caller ID is also transmitted in plaintext in H.323.

Feasibility of Caller ID Spoofing

Unlike PSTN or cellular network scenarios, in VoIP the caller ID originates at the client end; i.e., the clients could generate control packets with arbitrarily chosen caller IDs. Most VoIP software provides an interface allowing a caller to specify his/her

caller ID for each phone call, making caller ID spoofing trivial; e.g. x-lite [130], sipdroid [32], etc. In fact, many VoIP carriers manipulate caller ID to avoid long distance charge [40].

Network Interconnection Protocols

Different telephone networks are interconnected using a peering architecture, as shown in Figure 3.1. In the following, we discuss how a call is routed and caller ID is forwarded between carriers.

Call Routing

In telephone systems, phone numbers are assigned based on geographic locations to discriminate between local and long-distance calls. In the US, each carrier is assigned a unique prefix in each geographic location by the North American Numbering Plan Administration (NANPA) [133] and the call routing is done based on prefix-matching. For example, in Washington, the 360-269 prefix is assigned to *AT&T* and 360-270 to *Sprint*. When a customer calls 360-269-XXXX, the originating carrier must forward the call to AT&T[2].

Caller ID Forwarding

Signaling System No.7 (SS7) [66] is the de facto standard for interconnecting carriers, even though many regulators have suggested using VoIP [116]. When a caller and a callee have subscribed to different carriers, the call has to go through either an SS7 or VoIP connection. The originating carrier sends the caller ID as part of the control packets. In both cases, the receiving carrier passes the caller ID data to the callee without any modification or validation. Hence, caller ID is not verified in either case.

[2]Unless the phone number is ported, i.e., a customer of carrier A switches service to carrier B and still uses the same phone number. In this case, the call will be routed to the carrier B instead of carrier A.

Feasibility of Caller ID Spoofing

Since there is no verification mechanisms between carriers, it is possible for an attacker to get connected with a carrier using either SS7 or VoIP, and then exploit the lack of authentication between carrier networks to spoof caller ID. Such an attack, while valid, is costly and complex to carry out because the attacker has to establish an SS7/VoIP connection with a carrier, which requires the attacker to complete an interconnection agreement with the carrier, to install necessary hardware and software, to pay a premium, etc. Thus, it is not meant for casual adversaries with a limited budget.

Summary

It is difficult to launch caller ID spoofing attacks by exploiting the caller ID protocol within PSTN or cellular networks, but it is possible by exploiting VoIP. Additionally, an adversary can spoof caller ID by exploiting the lack of caller ID authentication between carriers.

3.3 Caller ID Spoofing Attacks

Caller ID spoofing is defined in the US legislation act titled *Truth in Caller ID Act of 2009* [33] as follows:

> "A *caller ID spoofing attack* is a malicious action that causes any caller identification service to knowingly transmit misleading or inaccurate caller identification information with the intent to defraud, cause harm, or wrongfully obtain anything of value."

This definition makes it difficult to detect caller ID spoofing, since there are a few standard, non-malicious telecommunication services that result in a mismatch of the displayed number and should not be classified as caller ID spoofing. In this section,

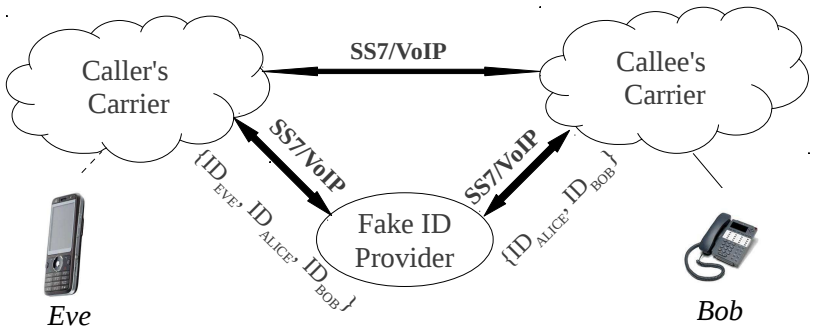


Figure 3.3 An illustration of how existing fake caller ID service provider spoofs a caller ID leveraging the loophole in network interconnection protocols.

we first discuss spoofing attacks that can be carried out in different telephone setups, and then discuss scenarios that should not be identified as spoofing attacks.

Spoofing During Call Signalling

As mentioned before, caller ID spoofing attacks are possible because VoIP protocols and network interconnection protocols lack caller ID validation mechanisms. While it takes an extra endeavor to establish an SS7 or VoIP connection with a telephone carrier for spoofing attacks, adversaries could carry out spoofing attacks with little effort in several ways. We discuss three such attacks in the following.

Spoofing via Fake ID Providers.

A special service provider, which we refer to as *Fake ID Provider*, provides caller ID spoofing services by exploiting the lack of authentication in caller ID protocols. A Fake ID Provider establishes SS7/VoIP connections with various telephone carriers. Such interconnection options are supported by all the leading carriers in the US for business customers (e.g., AVOICS [10]). Then, the Fake ID Provider acts as the middle man between attackers and victims to relay caller IDs specified by its customers (attackers in this case). Figure 3.3 illustrates an example, where an attacker (Eve) tries to call the victim (Bob) faking Alice's caller ID. First, Eve calls a Fake

ID Provider, and supplies Bob's phone number as the destination number and Alice's phone number as the desired spoofed caller ID. Then, the Fake ID Provider establishes a call to Bob with Alice's caller ID, and finally connects Eve with Bob once the call is answered. Since the Fake ID Provider is connected to the *callee's carrier* via an SS7/VoIP link, the *callee's carrier* simply accepts and forwards the caller ID of the incoming call to Bob, making spoofing attacks simple and effective.

An attacker can subscribe to a Fake ID Provider and carry out spoofing attacks towards any victim from any type of phone, provided that the Fake ID Provider is connected to the victim's network.

Spoofing via VoIP Services.

Many VoIP carriers allow their customers to specify their own caller ID, and will forward the caller ID to the callee's carrier without modifications. To launch an attack, an adversary can subscribe to a VoIP carrier that allows caller ID manipulation and can either use VoIP client software or a VoIP phone to claim arbitrary caller IDs.

Spoofing via Automated Phone Systems.

An increasing number of businesses use automated phone systems to provide Interactive Voice Response (IVR) services, so that they can computerize phone calls for purposes of marketing, survey collection, appointment reminders, etc. To simplify the development of automated phone systems, a group of emerging service providers (e.g., Voxeo [118], Nuance Cafe [15]) allow their subscribers to customize automated phone systems by using a scripting language, such as VoiceXML [119] and select their own caller IDs. These providers connect to major telephone carriers via SS7 or VoIP protocols so that their customers can call their target callee [39]. Such providers allows their customer to configure desired caller ID for outgoing calls and will deliver preconfigured caller IDs for their subscribers regardless of their intentions. Because of

the loophole of the network interconnection protocols (Section 3.2), the downstream telephone carriers will simply accept any caller IDs, including the spoofed ones. An adversary can subscribe to such a service to launch caller ID spoofing attacks.

Summary

Regardless of which types of the aforementioned spoofing attack is being launched, our proposed solution is capable of detecting all of them. In this work, we only evaluate our caller ID spoofing detection schemes utilizing Fake ID Provider, and we believe that our experimental results will provide important insight to other types of spoofing attacks since our detection scheme is independent of how caller ID spoofing attacks are launched.

Spoofing After Call Establishment

When a caller and a callee belong to the same PSTN carrier, it is virtually impossible to spoof a caller ID during signalling processes, as we discussed in Section 3.2. It is, however, feasible to spoof the caller ID *after* a call is established: An attacker can transmit a fake caller ID for a second call (call waiting). After the first call is answered, an end-to-end voice channel is established between the calling parties, and if there is a new call, the caller ID of that call is transmitted over this voice channel. The attacker has no access to the control channel during the call setup, but can manipulate information transmitted over the voice channel. To launch such an attack, the adversary makes a phone call to the victim and once the call is answered, she transmits a packet according to the caller ID protocol of the carrier, e.g., Bellcore FSK, with the desired fake caller ID. Immediately after receiving the caller ID packet, the displayed caller ID in the victim's phone will change to the fake one. An adversary can install already available software, e.g., the Software Orange Box, to launch such an attack.

A disadvantage of this technique is that the spoofed ID can only be transmitted once the call is answered. As a result, unless the call is answered right after the first ring[3], the callee would observe the original caller ID until the call is answered and the FSK packet with the spoofed caller ID is received. Additionally, the callee will also hear a call waiting beep when the spoofed ID appears on the phone. Although this attack may not always be practical, we included it in the discussion for completeness, and our detection scheme can detect such an attack anyway.

A Mismatched Caller ID but not Spoofing

The caller ID blocking services and Primary Rate Interface (PRI) lines generate a mismatched caller ID, but should not be classified as caller ID spoofing. For caller ID blocking service, a carrier will transmit the text BLOCKED or UNAVAILABLE instead of the real caller ID to the callee.

PRI lines are designed for business organizations that want to support multiple simultaneous calls (i.e., 32 channels for an E1 line [111]) while sharing one single caller ID for all their phone lines. In a PRI system, each phone line inside an organization is connected to the PRI line through a Private Branch Exchange (PBX), which assigns the same caller ID to all outgoing calls. The mismatched caller IDs in PRI lines are different from caller ID spoofing because the caller ID associated with a PRI line is officially owned by the business organization and once assigned, the caller ID cannot be changed without the permission from telephone carriers. Our CallerDec will mechanism will recognize both scenarios as non-spoofing cases.

[3]The caller ID is transmitted between first and second ring; thus if an incoming call is answered before the caller ID is received, the caller ID cannot be shown in the display.

In this section, we specify the system model and discuss design requirements and give an overview of our end-to-end detection scheme.

System Model

For the rest of the chapter, we will refer to Alice as the caller, Bob as the callee, and Eve as the attacker who tries to spoof Alice's caller ID while calling Bob. We note that, Alice may not be in Bob's contact list (unknown), and Alice's number could be invalid (unreachable). Since the verification operation is preformed automatically, we expand our definition of the names and refer Alice, Bob, and Eve to their devices as well. We envision that Alice, Bob, and Eve can be a smartphone, a mobile phone, a PSTN phone, a VoIP phone, or an automated system (e.g., bank), etc. Regardless of the type, we assume that Bob has a strong incentive to verify the caller ID of a caller, e.g., he can be a bank that needs to verify the caller ID of a customer. Thus, Bob integrates CallerDec in his device (e.g., by installing an app in a smartphone, or by upgrading the firmware of a PSTN phone, or by updating the software of a Private Branch Exchange (PBX)[4], etc). In comparison, Alice may or may not integrate CallerDec.

We consider that telephone carriers are trusted; they route outgoing calls to dialed numbers and do not collude with Eve in any way. Thus, Eve cannot capture or inject any type of packets into the telephone networks. Neither can she answer or reject a call unless she is the callee. Additionally, we assume that Alice does not collude with Eve and will not help Eve with caller ID validation. Otherwise, we consider that Eve is authorized to use Alice's caller ID.

[4]Business organizations use PBX as phone exchanges which offer internal phones service, multiple simultaneous calls with the same caller ID, etc.

Requirements

Security

The detection scheme should guarantee that an honest caller can prove the validity of his/her caller ID, and an adversary cannot pretend to be calling from an arbitrary number.

Compatibility

The detection solution should only change telephone terminals but not the existing telephone infrastructure, because adding any extra hardware to the existing infrastructure or introducing new protocols to the core telephone networks would be a great expense to all telephone carriers. Additionally it should be compatible to various telephone networks (e.g., GSM, VoIP, PSTN).

Usability

The detection strategies should be user-friendly, i.e., they should be automated, require almost no effort from either a caller or a callee, and should not change common procedures of phone calls. Otherwise, the callee could just dial the displayed caller ID and verify verbally.

Efficiency

The detection scheme should have low computational overhead so that it can be integrated into telephone terminals that have limited resources, e.g., PSTN phones, mobile phones, etc.

End-to-End Detection Scheme Overview

How can CallerDec, a program running on a telephone, verify who is indeed calling? Similar to the design principle of TCP in Internet, CallerDec considers a telephone network as a black box and rely on the feedback of end-to-end communication services supported by this black box for verification. In total, we have identified two services as the basis for CallerDec.

i. **Short Message Service (SMS).** SMS enables a user to send short text messages to another user with a mobile, VoIP [65], or a PSTN line [43], even though not all PSTN phones support SMS. Given a recipient number, the telephone operator will route SMS to the intended destination via control channels, unless the destination is unreachable. In such a case, the operator can notify the sender of the failed SMS delivery.

ii. **Traditional phone calls.** Given a number, the operator will try to establish a phone call for the caller over a control channel, and create a voice channel after the call is answered. We choose to use control channels for caller ID verification, since a caller cannot manipulate the control channels in a traditional telephone network but can acquire the status of the phone calls, e.g., through distinguishable ringback tones (e.g., busy) or voicemail greetings.

Leveraging either SMS or the call setup procedure, we design two types of CallerDec for caller ID verification, an `SMS-based CallerDec` and a `Timing-based CallerDec`. Overall, CallerDec works as follows. When Bob receives a phone call, CallerDec will automatically initiate the caller ID verification by sending a challenge to Alice over one of the end-to-end communication services, e.g., either SMS or a phone call. Then, the challenge will be delivered to Alice if it is reachable. Once the challenge reaches Alice, the CallerDec at Alice's end will respond to Bob whether she has indeed made the phone call. Collaboratively, CallerDec on both ends can achieve automated caller

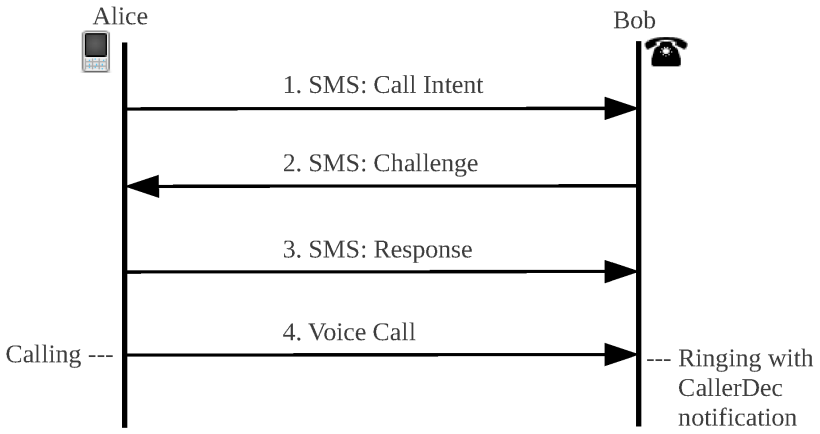


Figure 3.4 SMS-based CallerDec involves performing challenge-response between a caller and a callee before a call initiation.

ID verification. There are, however, several challenges to be addressed: How should Alice respond? How should Bob infer the response based on the feedback of the end-to-end communication channel? Is it possible to automate the verification process? Is it possible to use a second end-to-end communication channel while there is an incoming call? We address all these challenges in the design of CallerDec. We will discuss details of SMS-based CallerDec in Section 3.5 and timing-based CallerDec in Section 3.6.

3.5 SMS-based CallerDec

Our first scheme is SMS-based CallerDec (hereafter SMS-CallerDec) and is designed for devices with SMS capability. We will discuss timing-based CallerDec that does not require SMS services in Section 3.6. The idea is that in parallel to a phone call, Bob (the callee's phone) will send a *Challenge SMS* to Alice (the owner of the claimed caller ID), who will reply with a *Response SMS*, indicating whether she has made the phone call. The goal of the verification mechanism is to inform a user the validity of the displayed caller ID and to ensure that a spoofed caller ID will not be marked as valid by mistakes.

Although it is natural to let Bob initiate the *SMS challenge* right after he receives a call and before answering it, we found that it is impossible to implement for some telephone networks. For instance, in CDMA mobile networks, it is impossible to send an SMS while there is a pending incoming call because the mobile will be in `System Access State` where channel assignment is performed. No SMS can be sent until the channel assignment is complete [4]. Thus, mobile phone OS, such as Android [46], does not allow SMS transmission while a call is waiting to be answered. To overcome this restriction, SMS-CallerDec lets the caller initiate the challenge-response procedure prior to making a phone call. As long as the challenge-response SMS exchange is performed sufficiently close to a phone call, we consider the verification bound to this phone call.

Protocol Description

As shown in Figure 3.4, dialing a phone call with the SMS-CallerDec consists of 4 steps, and the verification output can be one of the three results: `VALID`, `SPOOFED`, and `NOTSUPPORTED`. `VALID` means that the caller ID has been successfully verified, `SPOOFED` indicates that the caller does not own the caller ID, and `NOTSUPPORTED` indicates that the caller ID cannot be verified because the caller does not have CallerDec.

(1) **Call Intent.** When a user (Alice) wants to make a voice call, she sends a special call-intent SMS to Bob, which contains the caller ID of Alice and the intended destination phone number (e.g. Bob's number):

SMS_Intent = IntentHeader; CallerNumber; CalledNumber

(2) **SMS Challenge.** Bob, upon receiving an *SMS_Intent*, sends a *Challenge SMS* to Alice. A *Challenge SMS* consists of a random nonce of m bits ($m = 128$ in our systems). This SMS is formatted as follows.

SMS_Challenge = ChallengeHeader; CallerNumber; CalledNumber; Nonce;

If the caller ID is unreachable, e.g., it is an invalid phone number, the phone is off or out of service region, then the *SMS_Challenge* will fail to be delivered. In this case, Bob will consider the caller ID `SPOOFED`, since it contradicts to the fact that Bob just received an *SMS_Intent* from 'Alice'. If the SMS delivery report confirms a successful delivery, Bob will start a timer, waiting for the response message. The timer ensures that the verification process does not wait indefinitely for the response SMS, and the verification session will be terminated after a while.

We note that telephone operator returns SMS delivery reports upon requests. For instance, Android phone users can select this option in the SMS settings or insert a special header in every SMS, e.g., `*noti#` for T-Mobile.

(3) **SMS Response.** After receiving the *SMS_Challenge SMS*, Alice replies an *SMS Response* acknowledging or dis-acknowledging the challenge. In cases that Alice does try to call Bob, she generates a *positive-response SMS* with the following format.

SMS_Response_ACK = ACK; F (SMS_Challenge, ACK)

Here F is a hash function that will return `n`-bit hash generated based on nonce, CallerNumber, and ACK/NACK. For instance, F can be MD5 [101] or SHA-1 [48].

In cases that Alice does *not* intend to call Bob, she generates a *negative-response SMS* with the following format:

SMS_Response_NACK = NACK; F (SMS_Challenge, NACK)

(4) **Voice Call.** After sending the *Response SMS*, Alice checks SMS delivery status using SMS delivery report. If SMS is delivered successfully, she starts the voice call to Bob. When Bob's phone starts to ring, the notification from SMS-

Algorithm 4 SMS-CallerDec

Require: INPUT:
CN = CallerNumer
OUTPUT:
$verified \in \{VALID, SPOOFED, NOTSUPPORTED\}$
PROCEDURES:

1: $verified = NOTSUPPORTED$
2: $C = GenerateChallenge(CN)$
3: $s = SendSMS(C, CN)$
4: **if** $s = failed$ **then**
5: $verified = SPOOFED$
6: **return** $verified$
7: **end if**
8: $StartTimer(t)$
9: **while** $true$ **do**
10: **if** $timeout$ **then**
11: **return** $verified$
12: **else if** $ReceivedResponse$ **then**
13: $verified = VerifyResponse(C, CN)$
14: **return** $verified$
15: **end if**
16: **end while**

CallerDec will be displayed so that Bob can make an informed decision, e.g., either answer or reject the call.

Algorithm 4 shows the pseudo code of SMS-CallerDec running at the callee's (Bob) end. In particular, if the *response SMS* is received before a timeout, then Bob checks the correctness of the response by comparing the received response with locally calculated hash. He considers caller ID `VALID` for a correct *SMS_Response_ACK* and caller ID `SPOOFED` for *SMS_Response_NACK* or an incorrect *response SMS*. If no *response SMS* is received prior to a timeout or ringing, then caller ID is considered `NOTSUPPORTED`.

Security Analysis

The SMS-CallerDec requires no secret exchange beforehand. Instead, its correctness is ensured by the observation that the telephone operator is trusted and the challenge SMS will be delivered to the intended recipient of the SMS, not to or via attacker. In a telephone system, when a user requests to send an SMS, the SMS first goes to an SMS Controller (SMSC) of the operator through BTS, BSC, and MSC. Then, it is in turn routed through the destination MSC, BSC, BTS, and finally to the destination. Since the network devices are tightly controlled by operators and it is difficult for attackers to compromise, when Bob verifies Alice, the challenge SMS from Bob will be delivered to Alice, instead of Eve.

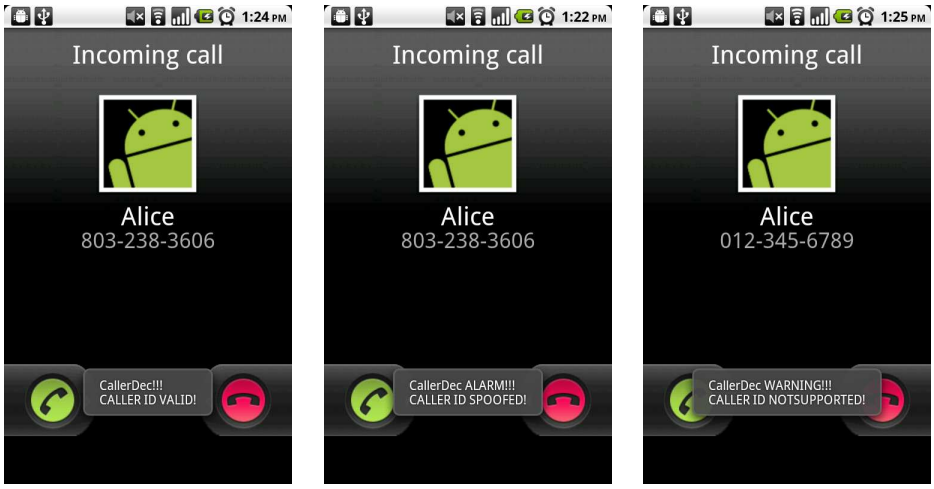
In this section, we show that leveraging the SMS-CallerDec, an honest caller can always prove the validity of her caller ID, and an attacker cannot pretend to be calling from random numbers. We divide the cases into normal and attacking scenarios.

Normal Scenario: Alice is calling Bob

We envision that Alice is motivated to prove that her caller ID is valid. Thus, she will install the SMS-CallerDec. When Alice calls Bob, she will participate in the challenge-response procedure, and make Bob conclude that her caller ID is `VALID`. In cases where Alice does not support the SMS-CallerDec, SMS-CallerDec at Bob's end can identify that the caller ID verification is `NOTSUPPORTED` at Alice's end, since neither *SMS_Intent* nor *SMS_Response* is received.

Attack Scenario: Eve is calling Bob spoofing Alice's caller ID

The goal of Eve is to fool Bob, and she can perform a few operations, trying to achieve it. We will show that none of Eve's attempt will work because of the SMS-CallerDec. For instance, Eve can choose not to send any *SMS_Intent* or *SMS_Response*, which will make Bob conclude that the caller ID verification is `NOTSUPPORTED`. Alterna-



(a) (b) (c)

Figure 3.5 Three outcomes of the SMS-based verifier: the caller ID is (a) VALID, (b) SPOOFED, and (c) NOTSUPPORTED.

tively, Eve can send a *SMS_Intent* with a colluder's phone number and let the colluder generate a *SMS_Response*. In this case, Bob will still conclude `NOTSUPPORTED`, since this challenge-response exchange is not associated with the incoming caller ID, i.e., Alice's number. Finally, Eve can send an `SMS_Intent` with Alice's phone number, and send a spoofed *SMS_Response_ACK*. Given that *SMS_Response_ACK* is an n-bit hash generated based on *SMS_Challenge*, the chance of creating a correct *SMS_Response_ACK* is only $\frac{1}{2^n}$, not to mention that Bob may receive an *SMS_Response_NACK* from Alice in response to his *SMS_Challenge*. Thus, Bob will consider this caller ID `SPOOFED`, and it is impossible for Eve to convince Bob that Alice's number is her `VALID` caller ID.

If Alice's caller ID is `BLOCKED`, SMS-CallerDec considers it a non-spoofing case and will not initiate verification process.

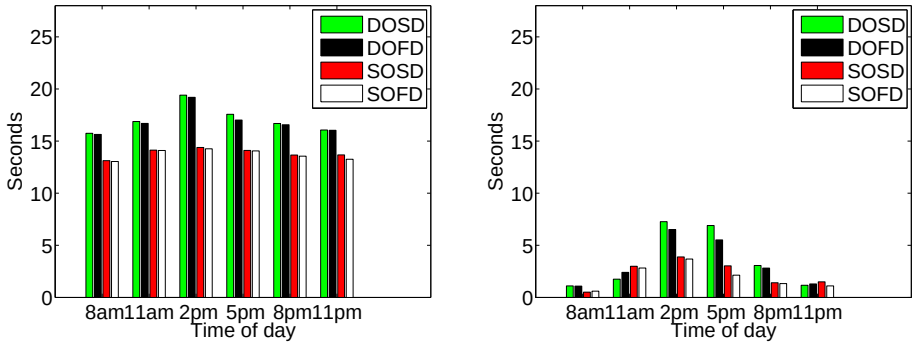
Implementation

We implemented the proposed protocol in Android (version 2.3.3) and automated all steps. In particular, we used the standard `Android SMS API` which includes an option to obtain a delivery report for each SMS. To avoid interfering users with our control SMSs, we added a special HEADER field in each verification SMS so that regular SMSs and SMS-CallerDec SMSs are distinguishable. We set the priority of our SMS-CallerDec app higher than built-in SMS app to make sure that a new SMS is delivered to it first. Upon matching the header, we suppress user notification for verification SMS and delete these SMSs from user inbox. We generated the hash using SHA-1 and converted it to SMS encoding before transmission. Fig. 4.17 shows three screenshots SMS-CallerDec, where (a), (b), and (c) show the cases when the caller ID is `VALID`, `SPOOFED`, and `NOTSUPPORTED` respectively. At this point, the callee can make an informed decision to accept or reject the call.

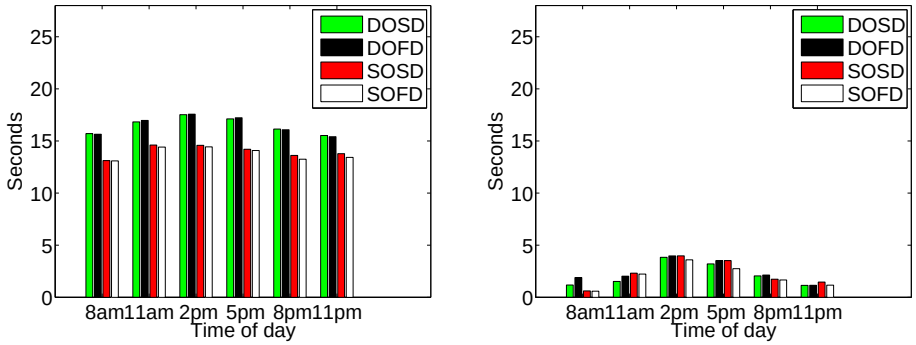
Performance

We evaluated the performance of SMS-CallerDec by measuring the delay of end-to-end verification, and studied the impact of the type of phones, the operators, and the time of the day. We selected three Android devices and classified them as fast devices or slow devices based on the processor speeds, and the device specifications are summarized in Table 3.1. We chose some common telephone operators in the USA, which are AT&T, T-Mobile, and SimpleMobile. We used two cases in the experimental setup: (a) The caller and the callee belong to the same operator, i.e., T-Mobile, and (b) the caller and the callee belong to different operators, i.e., a T-Mobile user calls an AT&T user, or a T-Mobile user calls a SimpleMobile user.

In total, we have measured the delay of end-to-end verification in four scenarios: (a) *DOFD:* Different Operators and using two Fast Devices. (b) *DOSD:* Different Operators and using one fast and one Slow Device. (c) *SOFD:* The Same Operator



(a) total time during weekdays (b) variance during weekdays



(a) total time during weekends (b) variance during weekends

Figure 3.6 Timing analysis for SMS-CallerDec.

and using two Fast Devices, and (d) *SOSD:* The Same Operator and using one fast and one Slow Device. We measured the end-to-end delay every three hours from 8AM to 11PM, on both weekdays and weekends. At each measuring hour, we ran the whole verification process at least 20 times for each scenario.

From the experiment results depicted in Figure 3.6, we observed that the end-to-end verification delay was not affected by the specification of devices or the day of the week, rather it is affected by whether the SMS have to traverse different operators and the network traffic load.

Table 3.1 Configurations of Android devices used in the experiments.

Device Name	**Processor**	**RAM**	**Class**
Google Nexus One	1 GHz	512 MB	Fast
HTC Sense	1 GHz	576 MB	Fast
MyTouch	528MHz	192 MB	Slow

When the caller and callee belong to the same operator, the verification process was completed within 13.8 seconds on average both on weekdays and weekends. When the caller and the callee belong to different operators, the process took 16.8 seconds on average on weekdays and 16.5 seconds on weekends. In addition, the average end-to-end delay peaked at 2PM for both weekdays and weekends. We note that SMS delivery delays dominate the end-to-end verification delay. Typically, telephone operators allocate a limited number of channels for SMS services, and thus besides the regular network delays, an SMS may be queued in the SMS controller for an extended period of time, resulting in a long delay.

We acknowledge that the SMS delay could be higher than expected. However, the end-to-end delay can be hidden if SMSs are transmitted in concurrence to a phone call. After the channel assignment is completed, one can answer the phone call and perform SMS-based caller ID verification simultaneously. For instance, in a 9-1-1 call, the caller ID can be verified while the emergency situation is reported.

In summary, despite the short delay overhead, SMS-CallerDec can be used effectively to detect caller ID spoofing attacks.

3.6 Timing-based CallerDec

An End-to-End Covert Channel

Forming an end-to-end covert channel is difficult as CallerDec considers a telephone network as a black box and hence, only the services that are available to end systems can form a covert channel, e.g., SMS, control channel during traditional phone call, etc. We utilize SMS channel in SMS-CallerDec (Section 3.5). To make CallerDec independent of telephone networks and SMS compatibility, we utilize the *traditional phone call* service and design Timing-based CallerDec (hereafter T-CallerDec). Nevertheless, similar services can be chosen to form a covert channel.

Utilizing traditional phone call service, we form an end-to-end *timing channel* between Alice and Bob for spoofing detection. Essentially, the timing channel is built on top of the control channel that is used for call signalling in a traditional telephone network. Even though Alice and Bob *cannot manipulate* control channels directly, they can *acquire the status* of the phone call between them, for instance, through distinguishable ringback tones (e.g., busy) or call status (e.g., the call is answered or rejected). Since Eve cannot control the calls between Alice and Bob, they form a trusted timing channel by initializing, answering, or rejecting phone calls between them.

When Bob receives a call from Alice, he will initiate a new call to Alice after an interval τ_{sv} and Alice will respond to the new call according to whether she is indeed calling Bob. We refer to the first call from Alice to Bob as the *original call* denoted by $C^o_{A\to B}$ and the second call from Bob to Alice as the *verification call* denoted by $C^v_{B\to A}$. Bob determines whether the original call $C^o_{A\to B}$ is indeed from Alice by examining two information that is sent over the control channel: (a) how Alice responds to the verification call $C^v_{B\to A}$, and (b) how long Alice waits before responding. For instance, if Alice is calling Bob, Bob will observe that she rejects the verification call $C^v_{B\to A}$ after a pre-defined interval τ_v. Both τ_v and τ_{sv} are security parameters used to differentiate whether Alice has installed T-CallerDec or not, and these parameters are selected and agreed upon by all users in advance. Because timing estimation of Alice's waiting time τ_v is performed at Bob's side and its accuracy depends on the packet delivery delays inside telephone networks, we use a probabilistic classifier instead of a threshold-based approach to improve the estimation accuracy. As we discuss in Section 3.6, we use a light-weight Bayesian classifier [49] that is suitable to resource constrained phone terminals.

The idea of forming a covert timing channel between Alice and Bob is simple. However, several open problems remain, such as: (a) Bob must be able to estimate

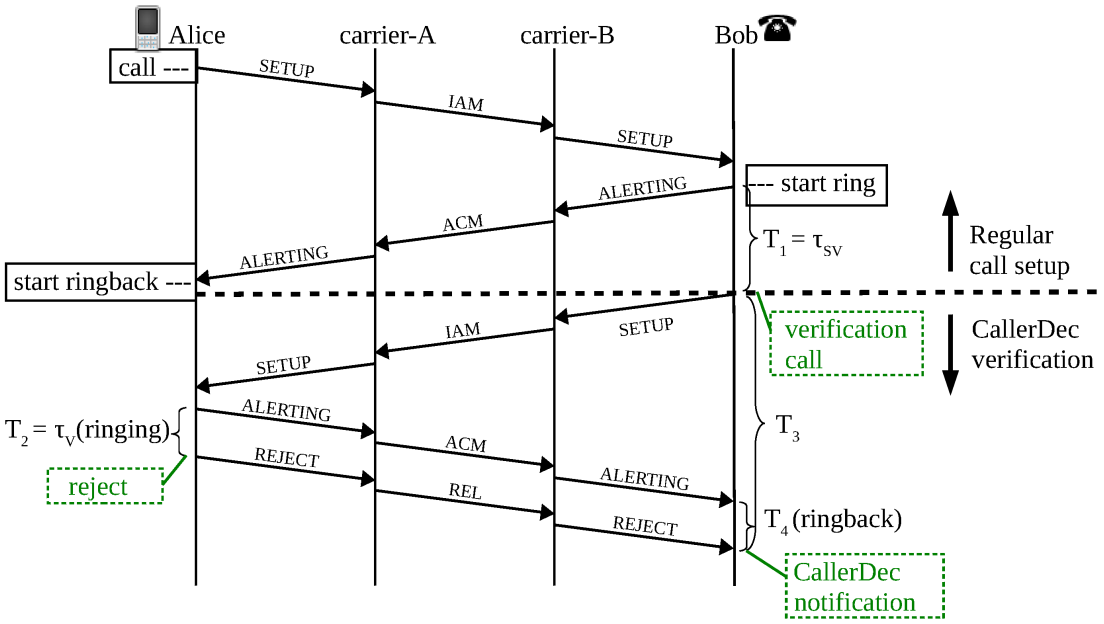


Figure 3.7 Call establishment and verification process when Alice is calling Bob: Bob initiates a verification call after τ_{sv} interval and Alice rejects the verification call after τ_v interval to prove her caller ID.

Alice's waiting time τ_v at his end, (b) the protocol must handle all possible scenarios, i.e., caller ID is valid, caller ID is spoofed, Alice does not support T-CallerDec, Bob's verification call goes to Alice's voicemail, etc. We address all these issues in the design of T-CallerDec protocol.

Protocol Description

T-CallerDec protocol is depicted in Figure 3.7 where Alice and Bob belong to carrier-A and carrier-B respectively. Assuming that the two carriers communicate through SS7, we describe regular call setup process and T-CallerDec protocol in the following.

Regular Call Setup: $C^o_{A\to B}$

In a telephone network where carriers use SS7 for communication, when Alice dials Bob's number, a `SETUP` request is sent to carrier-A. Then, carrier-A sends carrier-B

an `Initial Address Message(IAM)`, which is equivalent to `SETUP`. After carrier-B sends a `SETUP` to Bob, he responds with an `ALERTING` message and starts the ringing tone at the device. The `ALERTING` message indicates that Bob is available and the ringing has started. At this point, carrier-B sends carrier-A an `Address Complete Message(ACM)`. Subsequently, carrier-A sends Alice an `ALERTING` message, and Alice starts to play the ringback tone.

T-CallerDec Verification Protocol

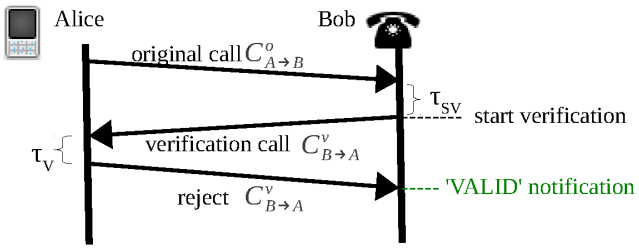
In this section, we introduce the protocol using the following scenarios: (a) *normal scenario:* Alice is indeed calling Bob and both of them installed T-CallerDec, (b) *attack scenario - spoof a reachable user:* Eve is spoofing Alice's number, and Alice is reachable with T-CallerDec installed, (c) *attack scenario - spoof an unreachable user:* Eve is spoofing Alice's number, and Alice is unreachable, and (d) *not-supported scenario*: Alice does not install T-CallerDec. For simplicity, assume for now that Bob can make two concurrent phone calls. Later, we will relax this assumption.

Normal Scenario. As shown in Figure 3.7, after receiving a phone call $C^o_{A\to B}$ from Alice, Bob will start the verification process according to the following steps.

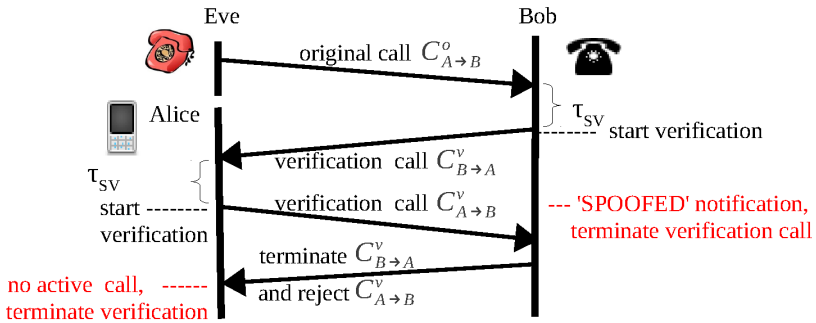
1. After an interval $T_1 = \tau_{sv}$, Bob initiates a verification call $C^v_{B\to A}$ to Alice that triggers a sequence of six messages: `SETUP, IAM, SETUP, ALERTING, ACM`, and `ALERTING`.

2. When Alice receives the verification call $C^v_{B\to A}$, she will reject it after an interval $T_2 = \tau_v$. As a result, Bob will receive a `REJECT` message from carrier-B indicating that $C^v_{B\to A}$ has been rejected.

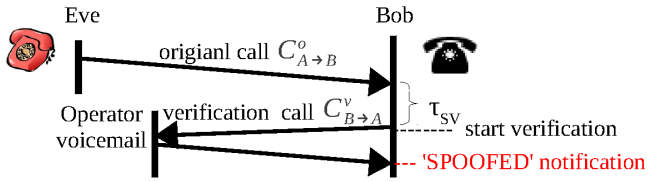
3. After receiving a `REJECT` message, Bob will measure the time difference T_3 between the moment of sending the `SETUP` message and receiving the `REJECT` message (Figure 3.7). Examining T_3 using the classifier, Bob will verify whether



(a) a simplified version of Figure 3.7 in a normal scenario



(b) attack scenario when Alice is reachable



(c) attack scenario when Alice is unreachable

Figure 3.8 Simplified T-CallerDec verification protocol and outcomes in normal and attack scenarios.

Alice has waited for the expected time τ_v before rejecting verification call $C^v_{B\to A}$. If the original call $C^o_{A\to B}$ is still active, then Bob will conclude that the caller ID is `VALID`.

As we discussed in Section 3.4, Eve cannot inject packets to the traditional telephone networks, neither can she reject or answer the verification call directed to Alice. Thus the verification process between Bob and Alice is protected, and the response from Alice is trusted. We show a simplified version of this scenario in Figure 3.8(a) and discuss how to identify Alice's response, and why Bob does not use T_4 (Figure 3.7) for estimating τ_v in Section 3.6.

Attack Scenario - Spoof a Reachable User. In this scenario, Eve is calling Bob and Alice is reachable, as shown in Figure 3.8(b). Similar to the normal scenario, Bob will first initiate a verification call $C^v_{B\to A}$ to Alice once he receives a call from Eve who pretends to be Alice. Alice will treat Bob's verification call $C^v_{B\to A}$ as a regular call $C^o_{B\to A}$ since she is not calling Bob. As a result, instead of rejecting it, she will initiate a new verification call $C^v_{A\to B}$ after an interval τ_{sv}. When Bob identifies that Alice has initiated a verification call $C^v_{A\to B}$, he concludes that Alice was not calling him. Instead, Alice is trying to verify Bob's verification call $C^v_{B\to A}$. After confirming that Alice's verification call $C^v_{A\to B}$ was initiated after a duration of τ_{sv}, Bob will conclude that the caller ID is `SPOOFED`. He will terminate his verification call $C^v_{B\to A}$ ($C^o_{B\to A}$ for Alice) and reject Alice's verification call $C^v_{A\to B}$ after an interval τ_v. On the other hand, Alice detects that her verification call $C^v_{A\to B}$ has been rejected and the original call $C^o_{B\to A}$ she received is terminated. As a result, she concludes that Bob may have received a call with spoofed caller ID and terminates her own verification process.

Attack Scenario - Spoof an Unreachable User. In this scenario, Eve is calling Bob, and Alice is unreachable, e.g., her phone can be powered off, out of range, or Alice is an invalid number. In such cases, as shown in Figure 3.8(c), the verification call from Bob $C^v_{B\to A}$ will be directed immediately to either Alice's voicemail or carrier's voicemail. When $C^v_{B\to A}$ goes straight to voicemail, it contradicts to the fact that "Alice" was calling Bob, and Bob will conclude that the caller ID is `SPOOFED`.

Not-supported Scenario. Now, we discuss the case when Alice does not support T-CallerDec, i.e., T-CallerDec is not installed in Alice's phone. In this case, the verification call $C^v_{B\to A}$ will be considered as a regular call. Since T-CallerDec is not installed, Alice herself may reject the call after a random interval, answer the call, or even not respond to the call. Regardless of the response, T-CallerDec verification can handle the following cases correctly:

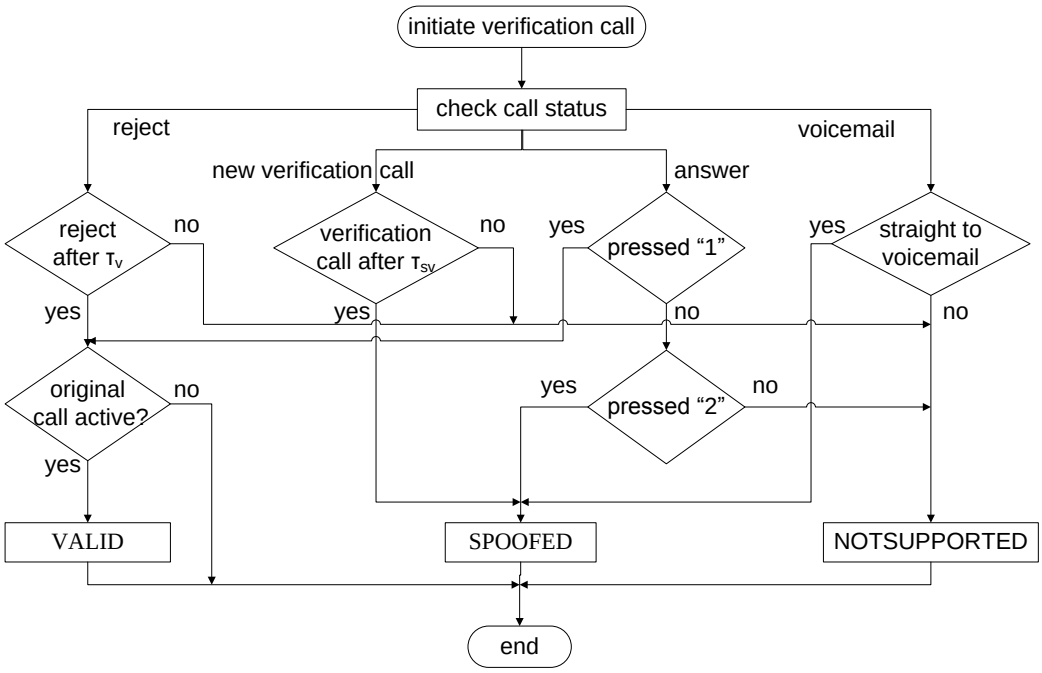
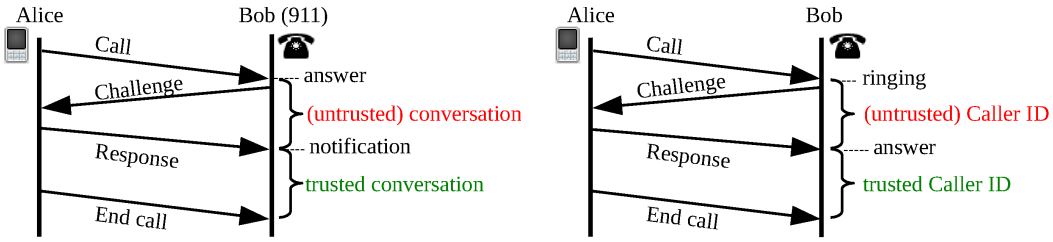


Figure 3.9 This flowchart shows how T-CallerDec handles different cases at callee's end to detect caller ID spoofing. Verification process is initiated as soon as there is a new incoming call.

(a) *Alice rejects the verification call after a random interval.* Once the call is rejected, Bob will measure T_3 and will use the classifier to verify whether Alice has waited for an interval τ_v before rejecting the call. In this case, it is unlikely that Alice happens to wait for τ_v, and thus T-CallerDec concludes `NOTSUPPORTED`.

(b) *Alice answers the verification call.* To leverage Alice's knowledge, T-CallerDec will play a pre-recorded voice instruction to advise Alice so that she can confirm whether she is indeed calling by pressing "1" or pressing "2" to reject the verification. Based on the input from Alice, Bob can conclude that the caller ID is either `VALID` or `SPOOFED`. Of course, Alice may refuse to enter a proper digit, and Bob will conclude `NOTSUPPORTED`.

(c) *No answer.* If the call was not answered at all, then the verification call will go to Alice's voicemail after ringing timeout period. In such cases, Bob makes a



(a) emergency call scenario

(b) regular call scenario

Figure 3.10 Two use cases of T-CallerDec. T-CallerDec follows the same verification protocol for both cases, because the verification procedure is independent to whether the original call is answered or not.

conservative decision and concludes that Alice does not support T-CallerDec, i.e., `NOTSUPPORTED`. Bob cannot conclude `SPOOFED` because Alice (the person) could be indeed calling and does not want to answer another incoming phone call.

The overall decision process is illustrated in Figure 3.9.

Use Cases

We envision two use cases of T-CallerDec: *emergency call*, where calls need to be answered immediately, and *regular call*, where Caller IDs are expected to be verified before calls are answered. In both cases, T-CallerDec follows the same verification protocol, because verification relies on an end-to-end covert channel that is independent to whether the original call is answered or not. However, caller ID will be verified *before* call is answered in a regular call scenario, and *after* call is answered in an emergency call scenario, as we discuss in the following.

Emergency Calls

In emergency call cases, such as 9-1-1 services, caller ID verification is performed in parallel to the voice call. As shown in Figure 3.10(a), the callee (e.g., an 9-1-1 service) answers the call from Alice upon ringing, and T-CallerDec starts the

verification process in the background. After the caller ID is verified, T-CallerDec will notify the callee, and sensitive information may be exchanged thereafter. Since the duration of 9-1-1 calls are reported to be between 1-2 minutes on average [72][34], the verification results shall be returned before the call is terminated.

Regular Calls

Eve may spoof Alice's caller ID with the goal of winning a chance talking to Bob, who would refuse otherwise (e.g., Eve may be an unknown number or is on Bob's block list). Thus, in regular call scenarios, T-CallerDec performs the verification before a call is answered. As shown in Figure 3.10(b), once Bob receives an incoming call, T-CallerDec starts to verify the caller ID and notifies Bob after the verification completes. While T-CallerDec may introduce delay in answering phone calls, it allows users to answer or reject spoofed calls. ■

The verification protocol is independent of both the above use cases, however, the number of required concurrent calls for T-CallerDec depends on the type of use cases. For regular call cases, Bob only requires one call control channel for verification. On the other hand, for emergency call cases, two concurrent call control channels are required since the verification shall be performed in parallel to the original call.

Security Analysis

The security of this mechanism relies on the observation that the verification call from Bob to Alice will be routed to Alice if she is available, and Eve cannot manipulate the verification call. Based on the choice of use cases, Bob can determine when to answer a call, e.g., before the caller ID is verified or after. We stress that when a call is answered is independent to the caller ID verification process. Hence, regardless of the use cases (Section 3.6), Bob can utilize the same T-CallerDec to fulfil the security requirement discussed in Section 3.4. The type of use cases do require different num-

ber of concurrent calls. For regular call cases, Bob only requires one call. However, for emergency call cases, two concurrent calls are required since the verification shall be performed in parallel to the original call. We analyze the security of T-CallerDec by considering the normal scenarios and attacking scenarios.

Correctness: Alice is Calling Bob

As we discussed in Section 3.6, utilizing T-CallerDec, Alice will reject the verification call after an interval τ_v to prove her caller ID, and Bob will conclude that the caller ID is `VALID`. Without T-CallerDec, Alice may answer the call from Bob, and listen to the pre-recorded voice instruction. With the proper input from Alice, Bob is able to confirm her caller ID. In cases that Bob receives no input or his verification call gets rejected by Alice at a random time, Bob can conclude that T-CallerDec is `NOTSUPPORTED`. We note that Alice is motivated to protect her caller ID and is likely to install T-CallerDec to detect that Eve spoofs her caller ID.

Security: Eve is Spoofing Alice's ID to Bob

As we discussed in Section 3.6, Alice will treat Bob's verification call as a new call and will initiate a new verification call to Bob. Consequently, Bob concludes that the caller ID is `SPOOFED` and Alice will conclude that Bob received a `SPOOFED` call and will terminate her verification process. Without T-CallerDec, when Alice receives the verification call from Bob, she may answer the call and enter proper input which leads Bob to conclude that the caller ID is `SPOOFED`. If Alice rejects the call after a random amount of time or does not respond, then Bob will conservatively conclude `NOTSUPPORTED`. The bottom line is that Eve cannot send any signal to convince Bob that Alice is calling.

Discussion

Special Cases

In this section, we discuss three special cases whereby the claimed caller ID is different from the true caller ID. Yet they should not be tagged as spoofed caller ID as we discussed in Section 3.3.

i. *Blocked caller IDs.* T-CallerDec depends on the caller ID of an incoming call for verification and T-CallerDec cannot initiate verification process if caller ID is *BLOCKED* or *UNAVAILABLE*. However, if Bob supports the mechanism to uncover caller ID of such calls, then T-CallerDec can be integrated seamlessly. For instance, the commercial service `TrapCall`[5] claims to have the ability to unmask blocked numbers, and T-CallerDec can utilize such a service to first unmask the blocked number, and then perform caller ID verification if it is needed. The 9-1-1 service also has such capability, and if integrated, T-CallerDec can perform caller ID verification effectively.

ii. *PBX systems.* T-CallerDec can be integrated easily in a PBX system of an organization, e.g., a bank. Since such systems generally have resources for multiple concurrent calls, they can adopt parallel verification, as discussed in use case 2 (Section 3.6). Furthermore, if Alice is reachable via an extension number of the PBX system and she calls Bob, Bob can verify the caller ID as usual.

iii. *Legitimate caller ID 'spoofing'.* It is possible that Alice intentionally spoofs her own caller ID when calling Bob, e.g., Alice uses skype to call Bob, while pretending to call from her cell phone. In this case, she can control T-CallerDec on her cell phone, and thus can proof her identity. We consider this scenario as a legitimate caller ID 'spoofing,' and T-CallerDec will conclude `VALID`.

[5] www.trapcall.com

Race Conditions

In a regular call scenario, both Alice and Bob may try to call each other simultaneously. In such cases, both calls will go straight to the voicemail. This is because most standards allow to signal at most one call at a time for each user, i.e., neither Alice nor Bob can receive an incoming call while making an outgoing one [62]. In this scenario, T-CallerDec is not triggered. In the case where one of the calling parties starts the call earlier than the other, at most one call will go through[6] and T-CallerDec handles the case as usual. The party that successfully receives the call will trigger its T-CallerDec to initiate the verification process, and validate the caller ID.

When Eve tries to spoof both Alice and Bob simultaneously, both Alice and Bob will initiate verification call to each other. But these calls would go straight to the voicemail and T-CallerDec will correctly conclude `SPOOFED`.

Denial of Service (DoS) Attack

The online forum [120][51] has reported several "DoS attacks" caused by the spoofed caller ID. In particular, attackers have spoofed a victim's caller ID (hereafter Alice). As a result, Alice received repeated calls from strangers who thought to have received calls from her. T-CallerDec can effectively address this problem. As the attacker calls Bob while spoofing Alice's caller ID, verification calls will be directed to Alice. With the help of T-CallerDec, the verification process will be carried out automatically as a background process. Without answering the call, Bob can learn that he received a spoofed call and shall ignore it; Alice's T-CallerDec automatically confirms that she didn't call. This way, the spoofed calls will not disturb Alice or Bob. Furthermore, Alice can collect the number of verification calls and thus detect the "DoS attacks" launched by the attacker.

[6] Alice can receive a second call while she is on another active call because the call signaling for the active call has ended and the call waiting protocol kicks in.

Advanced Configuration

T-CallerDec can be configured to verify only pre-selected numbers and balance the trade off between delays and trust. Similarly, verification can be disabled for certain numbers, e.g., premium rate numbers.

Implementation Challenges

When implementing T-CallerDec in Android, we encountered several challenges. Particularly, T-CallerDec requires to estimate the ringing duration at the other end, to obtain the status of that call, and to automatically initiate a verification call. However, Android does not contain APIs for identifying the status of an outgoing phone call or estimating the ringing duration at the other end. Neither does it allow two concurrent phone calls, and it hides the APIs for automating phone calls. We discuss how we overcome these challenges in the following.

Verify Caller ID Using Timing Estimation

One key issue to verify Alice's caller ID is to estimate her ringing duration (denoted by T_2). As shown in Figure 3.7, the ringing duration (T_2) is the time difference between the moments when Alice sends an `ALERTING` message, and a `REJECT` message.

In an ideal scenario, when the end-to-end transmission latency for `ALERTING` and `REJECT` messages is the same, T_2 equals T_4, where T_4 is the time difference between the moment when Bob receives an `ALERTING` and the one when he receives an `ANSWER` or `REJECT` message (shown in Figure 3.7). This makes T_4 a perfect candidate for estimating T_2. However our analysis on T_4 shows that it can vary from 50ms to several seconds. This is because some carriers start playing the ringback tone before receiving an `ALERTING` message. For instance, AT&T starts the ringback tone even when the callee is unavailable. Due to such a high variance in the time difference, we ruled out using T_4 to estimate T_2.

We found that T_3 (as shown in Figure 3.7), which is roughly the sum of T_2 and the round trip time from Bob to Alice, is independent of the types of carriers, since it does not depend on when the ringback tone starts. Hence we chose T_3 for estimating T_2 and consequently use it to classify caller ID as `VALID`, `SPOOFED`, or `NOTSUPPORTED`. Unlike the Internet where the round trip time varies, the round trip time in a telephone network is relatively stable due to the quality of service (QoS) requirements of telephone standards [41][42] and the circuit-switched nature of communication. This makes it possible to estimate T_2 using T_3. Our experiments also confirm this hypothesis. To make T-CallerDec compatible to devices with low computational power, e.g., mobile phones, we choose `Bayesian Classifier` [49]. The Bayesian classifier is an efficient method for calculating posterior probability based on prior probability and likelihood in the training data. Although the classifier needs prior training, our experimental results involving various geographic locations and time of the day show that the same trained model can be used on different phones for effective classification.

For the training dataset, we recorded the values of T_2, T_3, time of day (T_{day}), and status of the verification call (S_{call}), i.e., rejected, answered or voicemail. We label each dataset with appropriate class: `VALID`, `SPOOFED`, or `NOTSUPPORTED`. For each test sample, we employ the following Bayes equation [49] to calculate the probability of each class, C_i.

$$p(C_i \Big| T_3, T_{day}, S_{call}) = \frac{p(T_3, T_{day}, S_{call} \Big| C_i)\, p(C_i)}{p(T_3, T_{day}, S_{call})} \tag{3.1}$$

Here, $p(C_i)$ is the probability of C_i in the training dataset. T-CallerDec classifies the test sample as the class with the highest probability. Thus, based on the estimated duration of T_2 and Alice's action, T-CallerDec detects caller ID spoofing attacks.

Identify the Status of the Verification Call

T-CallerDec scheme requires Bob to identify the status of the verification call, i.e., whether the call has been answered, rejected, or directed to the voicemail. This

task poses several challenges. Android does not allow users to access call signalling messages during call setup. As a result, we cannot identify call status directly from call setup messages, e.g., `REJECT` message. Neither does Android provide any API that returns whether the callee's phone is ringing or the call is answered. The status of an outgoing call is always `OFFHOOK`[7]. So, we seek alternatives to identify the status of an outgoing call.

To identify the status of the verification call, we utilized system logs. Logs of each Android app are printed in the system shell and T-CallerDec continuously monitors real-time logs using `Runtime` APIs. In particular, T-CallerDec monitors logs of three built-in system apps: `CallNotifier`, `AudioService`, and `Ringer`. Once a `DISCONNECT` log is printed by `CallNotifier`, T-CallerDec concludes that the verification call is *rejected*. To identify the *answer* or *voicemail* status, T-CallerDec searches for an `audioOn` entry from `AudioService` and a `stopRing()` entry from `Ringer`. To differentiate between answered and voicemail, T-CallerDec can record voice data using the microphone and identify the patterns of voicemail greeting using available tools [27]. If the pattern matches, T-CallerDec has reached the *voicemail*, otherwise the verification call is *answered*. This identification process is summarized in Table 3.2.

[7]OFFHOOK traditionally indicates that the handset of a PSTN phone is off the base and the user could be dialing a number or on an active call. It is used in the same context in Android.

Table 3.2 Identifying call status using Android system logs and matching voicemail patterns.

Call Status	**System App & Search String**			**VM Pattern?**
	CallNotifier	**AudioService**	**Ringer**	
Rejected	DISCONNECT	-	-	-
Answered	-	audioOn	stopRing	No
Voicemail	-	audioOn	stopRing	Yes

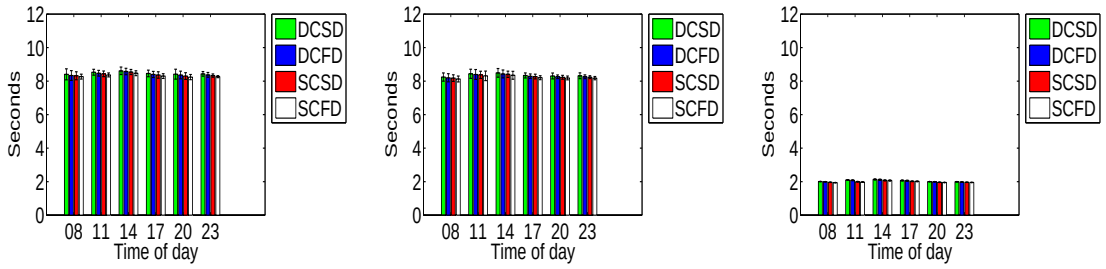
Initiate the Verification Call

Depending on the number of concurrent phone calls, two categories of phone services exist: (a) primary rate interface (PRI) [64] lines, and (b) regular lines (e.g., a mobile phone or a residential landline). PRI supports multiple concurrent phone calls using the same caller ID. Thus, a second line can be used to initiate the verification call while the first call may be in progress. Regular end users can dial a secondary phone call but at most one call can be active at a time. For instance, UMTS requires to put an incoming call on hold before initiating a new call [62] and Android enforces this requirement. As a result, when implementing T-CallerDec in Android, we have to put the incoming call from Alice on hold before initiating the verification call. Furthermore, Android provides no official APIs for putting a call on hold. To overcome the problem, we leverage Android hidden APIs of `ITelephony` interface using java reflection. We created an interface ITelephony in T-CallerDec App with the package name set as `com.android.internal.telephony` and added the function definition from the original `ITelephony` interface with an empty body. As a result, T-CallerDec is able to call the hidden functions from ITelephony at runtime and can perform call control operations (e.g., initiate a new call).

After overcoming the challenges, we implemented T-CallerDec on Android. Figure 4.17 shows three screenshots of T-CallerDec when the caller ID is `VALID` and `SPOOFED`. and `NOTSUPPORTED` respectively.

Performance

To evaluate the performance of T-CallerDec, we measured time of day and end-to-end delay of completing caller ID verification for the following scenarios which we discussed in Section 3.6: (a) normal, (b) spoof a reachable user, (c) spoof an unreachable user, and (d) not-supported scenarios. Additionally, we studied the impact of the type of phones, the carriers, and the time of the day in the verification



(a) VALID (b) SPOOFED (reachable) (c) SPOOFED (unreachable)

Figure 3.11 End-to-end verification delay in (a) the normal scenario when caller ID is VALID, and in the attack scenarios when caller ID is SPOOFED with Alice is (b) reachable and (c) unreachable.

delay. We selected three Android devices and classified them as fast devices or slow devices based on the configurations, and the device specifications are summarized in Table 3.1. We chose some common telephone carriers in the USA, which are AT&T, T-Mobile, and SimpleMobile. We used two cases in the experimental setup: (a) Alice and Bob belong to the same carrier, e.g., T-Mobile, and (b) Alice and Bob belong to different carriers, e.g., a T-Mobile user calls an AT&T user. In total, we measured data at six different times of the day in four experimental setup: (a) *DCFD:* Different Carriers and using two Fast Devices, (b) *DCSD:* Different Carriers and using one fast and one Slow Device, (c) *SCFD:* The Same Carrier and using two Fast Devices, and (d) *SCSD:* The Same Carrier and using one fast and one Slow Device. We set $\tau_v = \tau_{sv} = 0$ seconds in our implementation to minimize verification delay. Note that other threshold values can be used depending on the network parameters.

End-to-end Verification Delay

We measure end-to-end verification delay as the time difference between the moment when Bob receives an incoming call and the one when he identifies Alice's action. In the *normal scenario*, when caller ID was valid and the verification call was rejected after τ_v seconds, (Figure 3.11(a)), the verification was done in 8.40 seconds on average

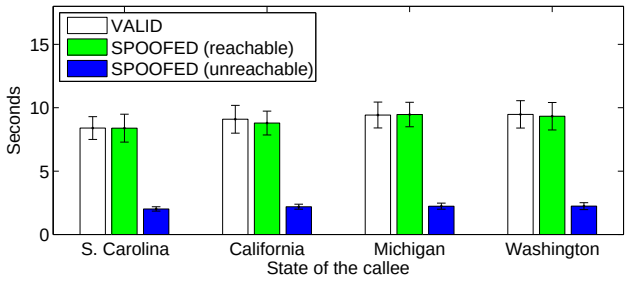


Figure 3.12 End-to-end verification delay of T-CallerDec based on geographic locations. The caller was always in State-1 and the callee was in one of the four states.

with variance between 0.1 to 0.3 seconds. In the worst case, when the caller and the callee were under different carriers, and one of them was using a slow device, the delay was 8.61 seconds. In all cases, the verification delay peaked at 2PM, which indicates that the verification delay of T-CallerDec is affected by network load but not much. The call setup delay dominates the delays. For instance, a recent study reported that call setup in 3G networks is between 4-7 seconds on average for various scenarios [99].

In the *spoofing a reachable user* scenario, Alice initiated a verification call after τ_{sv} seconds in response to Bob's verification call. As shown in Figure 3.11(b), the verification was done in 8.35 seconds on average. with a variance of 0.1 to 0.3 seconds, and in 8.49 seconds in the worst case. Similar to regular scenarios, the call setup delay dominates the end-to-end delay.

In the *spoofing an unreachable user* scenario, Alice's phone was turned-off and the verification call went straight to the voicemail. As shown in Figure 3.11(c), the verification delay was less than 2 seconds on average with a variance less than 0.04 seconds and 2.13 seconds in the worst case. Note that verification delay is low in this scenario because the call is not routed to its destination since the caller is unreachable.

Although T-CallerDec takes a few seconds for end-to-end verification, our analysis shows that such delay is mainly caused by telephone networks, and end devices or

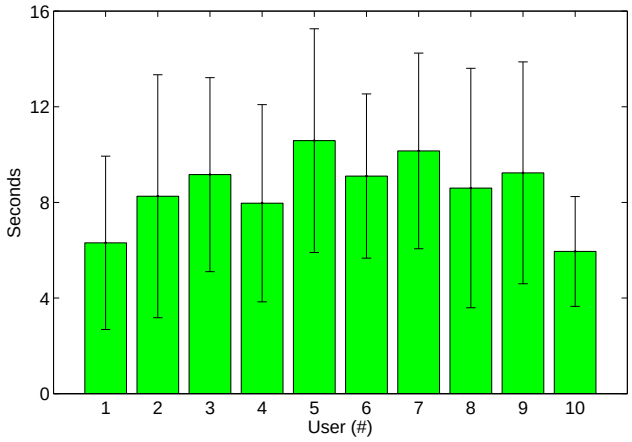
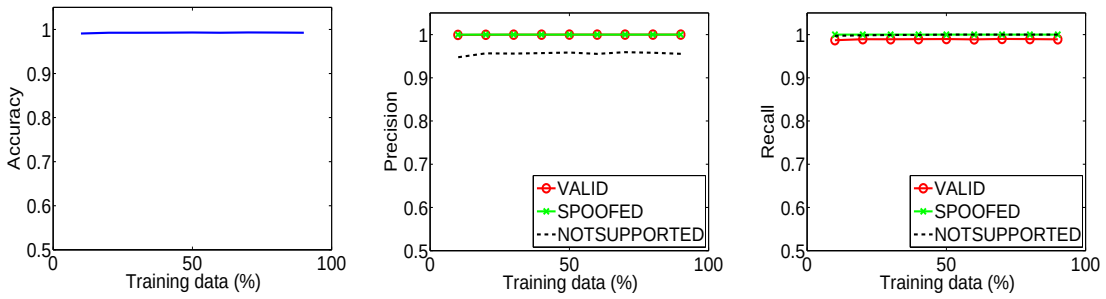


Figure 3.13 Mean and standard deviation of the respond time for 10 volunteers to reject or answer incoming calls, which represents the respond time in `NOTSUPPORTED` scenarios.

network loads have minor effects. However, the verification delay can be hidden in case of emergency calls (Figure 3.10(a)) because the verification is done in parallel to the phone call. Although T-CallerDec adds delay overhead before a user may answer calls in case of regular calls (Figure 3.10(b)), the actually experienced overhead should be lower since it generally takes a few seconds to answer a call [38].

Impact of Geography

We also analyzed the latency of T-CallerDec based on the geographic locations of the caller and the callee, as depicted in Figure 3.12. We selected four states located across the US which are California, Michigan, South Carolina, and Washington. In our experiments, the caller was always in South Carolina and the callee was in one of the four states. The result indicates that geographic locations of the caller and the callee have minor effects on the delay with variance in delay less than 1.1 seconds in all cases.



(a) accuracy (b) precision (c) recall

Figure 3.14 Performance of our Bayesian spoof detection classifier where (a) shows the accuracy, (b) shows the precision and (c) shows the recall of the 1classifier.

Timing Estimation

To verify caller ID, Bob estimates Alice's waiting time τ_v using a Bayesian classifier to decide whether a call is `VALID`, `SPOOFED` or `NOTSUPPORTED`. To analyze the performance of our classifier, we collected more than 3000 instances of calls labelled with appropriate class, e.g., approximately 1100 `VALID`, 1100 `SPOOFED`, and 800 `NOTSUPPORTED` instances. Both the `VALID` and `SPOOFED` instances were collected by T-CallerDec. In particular, the `VALID` samples were collected when Alice was indeed calling Bob, and the `SPOOFED` samples were collected when Eve spoofed her caller ID. For the `NOTSUPPORTED` samples, we asked 10 volunteers to reject or answer incoming calls at their will. No T-CallerDec was installed, but we wrote a customized app to collect the delay between the start of ringing and the moment of the users' operation. From the mean and standard deviation of the user's response time (Figure 3.13), we observed that the average response time of an incoming call varied from 5.9 seconds to 10.6 seconds, indicating that it is unlikely to confuse the user operation with T-CallerDec responses that are triggered with no delay.

We divided the dataset into training and test sets at various proportions p, where p = [0.1 - 0.9]. For instance, with p = 0.1, 10% (approximately 300 instances) of the dataset was used for training and the rest 90% (approximately 2700 instances) was

used for testing. We use the following metrics for evaluating T-CallerDec classifier where 100% is the desired outcomes for each metric:

i. *Accuracy* which is the percentage of correct outcomes of T-CallerDec,

ii. *Precision* which is the percentage of correct outcome for a class out of all the T-CallerDec outcomes for that class (e.g., correct `VALID` outcomes out of all the `VALID` outcomes), and

iii. *Recall* which is the percentage of correct outcome out of all the correct outcomes for a class (e.g., correct `VALID` outcomes of T-CallerDec out of all the outcomes that should be `VALID`).

As depicted in Figure 3.14(a), the accuracy of the classifier is more than 99% even when the percentage of the training dataset is only 10%, and 99.26% on average. Furthermore, the precision and recall are fairly constant: a 99.98% precision and a 98.91% recall when caller ID is `VALID`, a 100% precision and recall when caller ID is `SPOOFED`, and 95.62% precision and 99.93% recall when T-CallerDec is `NOTSUPPORTED`. The results also suggest that a small number of training data is sufficient for efficient classification.

In summery, T-CallerDec can be used effectively to detect caller ID spoofing. It provides high accuracy in caller ID spoofing detection.

Power Consumption Overhead

To evaluate the power consumption of T-CallerDec, we measured the millivolts consumed per hour of the entire phone. In this setup, apart from the default apps comes with Android OS, the phone only executed one battery logger app. Additionally, the Wi-Fi interface was turned-off in all cases.

Since T-CallerDec is triggered on demand by incoming calls, we measured the power consumption of T-CallerDec in two experiment setups: no incoming calls;

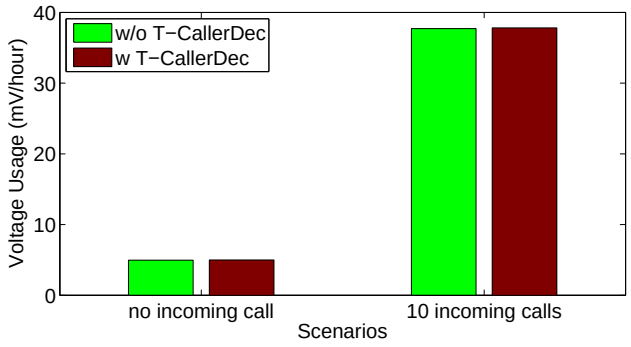


Figure 3.15 Analysis of power consumption overhead for T-CallerDec in four scenarios: no incoming call, with incoming calls, T-CallerDec is installed, and T-CallerDec is active. In all the experiments, the phone operated on batteries and the average millivolts used per hour was measured on average.

10 incoming calls per one hour with each call lasted for approximately 1 minute. For both setups, we compared the scenarios that T-CallerDec was installed with *not* installed; In the case that T-CallerDec was not installed, the user answered the calls at a random moment during ringing. In the case that T-CallerDec was installed, T-CallerDec became active right after receiving an incoming call, and the user answered the calls after he received the verification results. Each experiment lasted for six hours. We depict the experimental results in Figure 3.15, which shows that T-CallerDec almost consumed no extra power when T-CallerDec remained idle. This is because T-CallerDec is implemented as a background process with intent `android.intent.action.PHONE_STATE` [46] and the android operating system wakes T-CallerDec only when there is a new incoming call. In the case of receiving 10 incoming calls, T-CallerDec consumes little power, i.e., approximately 0.01 mV per call, compared to the case when T-CallerDec was not installed. In summary, our T-CallerDec ensures caller ID spoofing detection with almost no extra power consumption.

3.7 Related Work

While the problem of caller ID spoofing is generally known, previously proposed solutions typically require the cooperation and modification of phone provider networks, which imposes extra cost. For instance, TrustID is a corporate service offered to banks or large institutions [113] and it leverages special interfaces with the carriers to detect caller ID spoofing.

Cai [23] proposes a system to validate caller ID information based on "originating node information" of the respective call. In his design, a phone network provider should validate the claimed caller ID information based on additional meta-data associated with the call. Several ways of comparing the call's meta-data with the claimed caller ID are proposed, however, it does not address the problem of fake ID providers, which have both incentive and means to fake most of this meta-data. Additionally, customers must rely on their respective phone providers to verify the claimed caller ID.

Another approach is the RealName Registry by Chow et al. [30]. They propose that telephone providers establish authenticated name registries within their respective jurisdictions. Specifically, customers should register to their providers and be issued cryptographic certificates that can then be validated by the respective callee's. Note that this mechanism also allows the reverse authentication of the callee's towards the caller. However, while spoofing is a serious issue in many transactions, the cost imposed by introducing cryptographic authentication and a PKI is significant. This is aggravated by the fact that the proposed authentication scheme is only useful if widely deployed, so that unauthenticated calls can be rejected.

PinDrop by Balasubramaniyan et al. [13] determines the "provenance" of a call by evaluating audio artifacts introduced by digital encoding and analog noise patterns. As the call is routed through the network, the different deployed types of network technology succinctly manipulate the audio signal, creating a characteristic water-

mark that can be used to reconstruct and recognize the employed techniques, and thus also the path taken by a call. The approach is closest to our work as it does not require cooperation of the involved network providers and can be realized on an on-demand basis, by unilaterally modifying the callee's device software. However, PinDrop requires that the receiver answers the call before the source can be verified, and cannot by itself validate the origin of a previously unseen caller ID. As such, PinDrop is complementary to our scheme. Their approach could be integrated with ours to increase the detection certainty once the call was answered, or warn if a known caller ID originates from an unusual network location.

Piotrowski et al. [96] consider voice spoofing as an extension of caller ID spoofing and propose a watermarking mechanism for mitigating the threat. However, their approach requires modification of both the caller's and callee's devices and is therefore mainly useful for closed environments. In such scenarios, where arbitrary modifications to the caller as well as callee can be assumed, a wide range of well-known cryptographic and steganographic approaches can be considered (e.g., CryptoPhone [1]).

3.8 Summary

In this chapter, we investigated caller ID spoofing attacks and identified that it is the network interconnection protocols that make caller ID spoofing possible. There is no evidence that telephone carriers will change their networks to support caller ID verification. Thus, we seek an end-to-end solution to detect a spoofed caller ID. We designed CallerDec end-to-end verification mechanisms: SMS-CallerDec and T-CallerDec, implemented both CallerDec in Android-based phones. Our experimental analysis validates that CallerDec can effectively verify caller ID. Although the end-to-end delay for completing a verification takes a few seconds, such delay can be hidden when a verification is performed in parallel to the voice call. Moreover, we studied CallerDec on Android-based phones as a case study, but CallerDec can be

integrated to other types of phone terminals to effectively protect end users from caller ID spoofing attacks.

Chapter 4

CETAD: Detecting Evil Twin Access Point using End-to-End-based Solution

4.1 Problem Overview

Internet has become a part of our everyday life and Wi-Fi has gained much popularity in the last few years for accessing Internet. Wi-Fi market reached 6.4 billion in 2011 [57] and a rapid growth is forecasted in the upcoming years [84] as most the of mobile devices, e.g., smartphones, tablets, etc., have Wi-Fi capability. Due to such popularity, many shops, cafés, airports, etc., provide free Wi-Fi hotspot services to attract customers. JiWire reports that there are more than 840,000 Wi-Fi hotspots worldwide with more than 150,000 in the US [52] and these numbers are growing rapidly. Since the goal of the hotspots is to provide convenience and to attract customers, in the US, few or no security mechanism is in place. For instance, McDonalds, Starbucks, etc., provide free , open, and zero liability[1] access to customers.

The openness of Wi-Fi hotspot makes it vulnerable to evil twin access point (AP) attack [97]. In an evil twin AP attack, the adversary sets up a phishing AP that pretends to be the legitimate one as it uses the same Service Set IDentification (SSID) as a legitimate AP. Evil twin AP attacks can be harmful to users and have been used to launch a Man-in-the-Middle (MITM) attack [56][82] because all the packets from the client to any web server must go through the AP. An adversary can use MITM

[1]Customers have to agree to terms and conditions that the provider has no liability in case of security issues during hotspot use.

attacks for hijacking a session [82], or stealing sensitive data from the user by message falsification [90].

An evil twin AP attack is easy to launch, especially in a Wi-Fi hotspot due to the lack of security mechanisms of hotspots. An adversary can launch an evil twin AP attack using a laptop, a smartphone, or other Wi-Fi enabled devices by exploiting the loopholes of Wi-Fi client software implementation: Existing built-in Wi-Fi client implementation assumes that all the APs with the same SSID are legitimate and automatically connects to the AP with the maximum Received Signal Strength Indication (RSSI) value. As a result, if the RSSI of the evil twin AP is higher than that of a legitimate AP, a client will associate with the evil twin AP. An adversary can setup an evil twin AP with the same SSID in the following methods:

- She can configure the SSID of a mobile Wi-Fi AP with 3G/4G data connection, e.g., a Linksys mobile router, to same one as the legitimate Wi-Fi hotspot.
- She can install an app [31] in her smartphone to impersonate the legitimate Wi-Fi hotspot.
- She can use two Wi-Fi interfaces; one to connect to a legitimate Wi-Fi hotspot and configure the other in ad hoc mode pretending to be the legitimate Wi-Fi hotspot AP.
- She can install a software, e.g., Virtual Wi-Fi Router [117], to use a single Wi-Fi interface to setup an evil twin AP as well as connect to the legitimate Wi-Fi hotspot.

Much work is focused on detecting rogue AP[2], however most solutions are designed for infrastructure network rather than for client devices. For instance, several mechanisms have been proposed to monitor packets at the network gateways

[2]Evil twin AP is one type of rogue AP. Additionally, rogue AP can be an unauthorized AP connected to a network.

or routers [122][16][107][121] or to install extra custom devices for monitoring the network infrastructure, e.g., mobile agents [8], distributed radios [12], trusted wireless clients [19], etc. Those solutions are not applicable to Wi-Fi hotspots. A Wi-Fi hotspot provides free Internet service to attract customers; it has little motivation to guarantee no attack nor will setup additional devices or install detection software in the router to detect an evil twin AP attack. Moreover, most hotspot providers free themselves of any responsibilities from any damages due to security vulnerabilities of their system through "Terms and Conditions". Thus, it is the responsibility of the Wi-Fi clients to look out for themselves and *a client side* scheme to detect evil twin attack is necessary to ensure security of the Wi-Fi access. To the best of our knowledge, existing client side mechanisms for detecting evil twin AP attacks requires to install additional hardware in each hotspot [108] or needs prior training [45]. Thus, we focus on designing a plug-and-play mechanism to detect evil twin AP attacks that only requires to install software at the client device.

Designing a client side mechanism to detect evil twin AP attack is challenging for several reasons. First, the client has limited resources. Neither he has access nor prior information about the hotspot architecture. Second, hotspots use various Wi-Fi setup and the mechanism must consider all of them. Third, adding custom hardware, e.g., routers or servers, is not an option as it would limit the applicability and universal acceptance. We overcome these challenges in our detection mechanism which we call `CETAD` (Client end Evil Twin Access point Detector). We design CETAD based on the following observation: when multiple APs are legally configured to form a hotspot, besides the same SSID, they also share similar network parameters such as ISP names, Global IP addresses, Round Trip Time (RTT), temporal network behavior, etc. However, when an evil twin AP is present, discrepancies can be found among the APs. Our detection mechanism explores such similarities and discrepancies to verify that all available APs belong to the same group. In particular, we use three

techniques in our detection mechanism: similarity of ISP information, difference in RTT values, and standard deviation of RTT values. Using these techniques, we can identify multiple groups of APs during an evil twin AP attack with one group containing the legitimate AP(s) and the other group(s) containing evil twin AP(s); we identify such scenario as an evil twin AP attack. Even though CETAD is designed for a client device in a hotspot, it can be extended to to detect evil twin APs in an infrastructure network as well.

The rest of the chapter is organized as follows. In Section 4.2 we give an overview of the network architectures used in a Wi-Fi hotspot. We specify the threat model in Section 4.3 and overview our solution in Section 4.4. Then we present CETAD in Section 4.5 and implementation details, and results in Section 4.6. Finally, we discuss related works in Section 4.7 and summarize the chapter in Section 4.8.

4.2 Background

We consider the scenarios of Wi-Fi hotspots because hotspots generally allow open access and are vulnerable to evil twin AP attacks. Since our detection scheme leverages network elements and characteristics, we give a brief overview of Wi-Fi hotspot architecture in the following for better understanding of the scheme.

A Wireless Local Area Network (WLAN) is generally implemented using IEEE 802.11 standards [58], popularly known as Wi-Fi. IEEE 802.11 standards defines the communication protocol between a client and an AP. To access Internet, a client first associates with an AP by selecting the desired SSID. If multiple APs have the same SSID, the client associates with the AP that has the highest RSSI. Once associated, the client gets network parameters for the hotspot using Dynamic Host Configuration Protocol (DHCP) [61](details to follow) and [3]. then, can start surfing Internet.

[3]Most routers support built-in DHCP server capability by default.

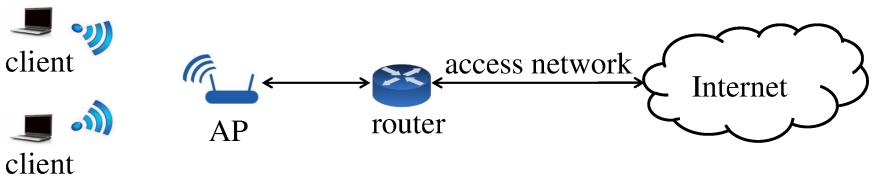


Figure 4.1 A simple WLAN network which is connected to Internet via an access network with a single access point and a router.

The popularity of Wi-Fi has made it one of the must-have properties for laptops, tablets, etc., for accessing Internet. A typical Wi-Fi network has four components: an access network, a router, a Wi-Fi access point (AP), and one or more wireless clients, as shown in Figure 4.1. An AP communicates with users in wireless channels and is connected to the router via wires. Many times APs are mistakenly considered as a router that connects an access network. In fact, an AP works in Open Systems Interconnect (OSI) layer 1-2, and connects wireless clients to a Local Area Network (LAN). Then, a router, a layer 3 device with Network Address Translation (NAT) capability, serves multiple LAN clients and connects to a Internet Service Provider (ISP) through one of the access networks. A wireless router has the functionalities of both a router and an AP, e.g., Cisco Linksys E100 Wireless-N Router. In the rest of the chapter, we use the terms AP and wireless router interchangeably. In the following, we discuss different types of access networks, possible hotspot architecture, and automatic network configuration in detail.

Type of Access Networks

A router can connect to an ISP through one of the following access networks:

i. **Cable network.** In this case, ISPs use cable television wired network to provide Internet service to households, e.g., Time Warner Cable [134] in the US.

ii. **2G/3G/4G network.** Mobile operators provide Internet service to mobile customers using their own networks. This type of access network is completely

wireless. All leading operators in the US, e.g., AT&T, Verizon [135], etc., offer mobile Internet services to customers.

iii. **Ethernet.** Some ISPs also provide Internet service through high speed Ethernet cable, e.g., Comcast [21]. Such services are generally available for business customers.

iv. **Digital Subscriber Line (DSL) network.** In this case, ISPs use landline telephone infrastructure to provide Internet service to users. For instance, AT&T [129] offers DSL Internet service in the US.

v. **Optical Fiber.** In this case, ISPs set up or utilize custom optical fiber network to provide ultra-fast Internet service to customers. Google Fiber [131] offers this service in Kansas, USA.

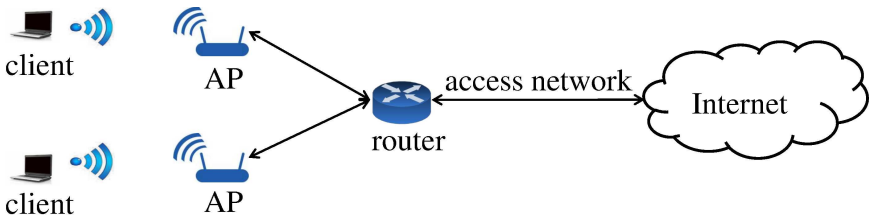
Among these access networks, cable network has the largest market share in wireless hotspots in the US. CableWiFi initiative[4] has 165,000 Wi-Fi hotspots in the US that uses cable networks [70]. AT&T has more than 32,000 Wi-Fi hotspots in the US that mostly uses DSL and cable networks [9]. Regardless of which access network is used, a wireless hotspot adopts one of two network architectures as discussed in the following.

Hotspot Architecture

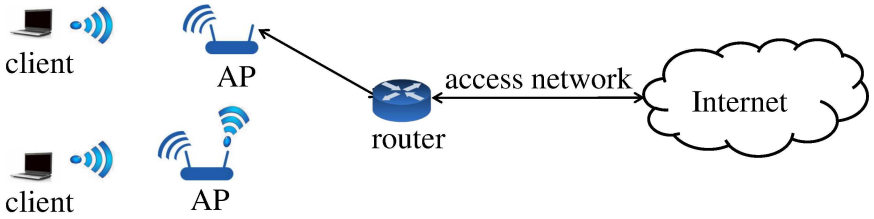
Single AP Architecture

In this setup, there is one AP that supports multiple wireless clients using a Wi-Fi standard, e.g., 802.11b/g. The AP is connected to a modem or a router through a Wide Area Network (WAN) interface. The modem/router is connected to the ISP via one of the possible access networks. We show a single AP network in Figure 4.1.

[4]The partner companies are Cablevision Systems, Comcast, Time Warner Cable, Cox Communications and Bright House Networks.



(a) multiple AP architecture



(b) multiple AP architecture with WDS

Figure 4.2 A simple WLAN network with multiple APs which is connected to Internet via an access network.

Multiple AP Architecture

The range of a Wi-Fi network is limited. For instance, 802.11b has range of 120 ft indoors and 300 ft outdoors [58]. As a result, multiple APs are required to cover a large area. The APs form a set named `Extended Service Set (ESS)` and all the APs in the ESS have the same SSID and similar configurations so that a user can automatically switch to another AP with a higher RSSI value when he moves towards the new AP. There are two possible architectures a WLAN with multiple APs can use.

- In one architecture, the APs are not required to have routing capabilities, instead they are connected to a router that is connected to the ISP via an access network. In this architecture, the neighbouring APs are configured to use *different channel frequencies* to avoid packet collision and to increase network throughput. A simple WLAN network for this architecture with two APs is shown in Figure 4.2(a).

- In an alternative architecture, the APs can be configured in wireless distribution system (WDS) [58] mode. In WDS mode, only one AP is connected to the router and all the APs can communicate between themselves over wireless channels. WDS APs are statically configured and use MAC layer for communication. WDS mode allows the users to roam from one AP to another without terminating any ongoing sessions, i.e., supports *wireless handover*. In WDS, all the APs use the *same channel frequency* so that they can communicate with each other. A simple WDS network with two APs is shown in Figure 4.2(b). We note that WDS architecture has not been standardized yet and most WDS capable APs do not support inter-operability. Thus, in this work, we do not consider multiple AP architecture with WDS mode.

A hotspot can implement either single or multiple AP architecture based on the area it plans to cover. However, there is no way for a wireless client to determine the architecture of a hotspot. Thus, evil twin AP attacks are possible in both single AP and multiple APs architectures. For multiple AP architecture, most Wi-Fi client software show only AP that has the highest RSSI value and allow the users to associate with it without validation.

Automatic Network Configuration

When a Wi-Fi client associates with an AP, it requires to configure network interface for using Internet and most WLANs have at least one DHCP server for automatic network configuration. The DHCP protocol works as follows. Upon associating with an AP, the client broadcasts a `DHCP DISCOVER` packet. In response, each DHCP server in the WLAN sends a `DHCP OFFER` to the client. If there are multiple DHCP offers, the client accepts one offer by broadcasting a `DHCP ACK`. A DHCP offer contains at least 5 fields: (a) Host IP address, (b) Subnet mask, (c) Router IP address, (d) Domain Name System (DNS) [60] servers, and (e) Server identifier. The client uses these

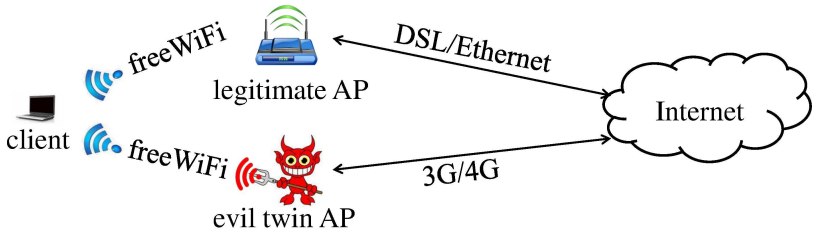


Figure 4.3 Mobi attack model where an adversary uses a Wi-Fi interface to create an evil twin AP and connects to Internet through 3G/4G.

information to configure his W-Fi interface automatically. At this point, the client can start surfing Internet.

In a WLAN with multiple APs, typically the router acts as a DHCP server so that the local IP address of the user remains same when he moves from one AP to another. However, each AP can act as a DHCP server instead of the router. In this scenario, the client will loose active network connections while roaming because he will receive different DHCP information from each AP.

4.3 Evil Twin AP Attacks

In a Wi-Fi network, the AP periodically broadcasts SSID, which allows a Wi-Fi client to discover the existence of the AP and associate with it. In an evil twin AP attack, the adversary sets up her AP using the SSID of the targeted Wi-Fi network. As a result, a client receives SSID broadcast from both the legitimate AP and the evil twin AP, but it cannot differentiate between these APs. The client simply assumes that both the APs are legitimate and associates with the one that has a higher RSSI value. For a successful attack, an adversary can increase the transmission power of the evil twin AP, or set it closer to the client to ensure that the client gets a higher RSSI value for the evil twin AP. There are several ways an adversary can launch an evil twin AP attack as we discuss below.

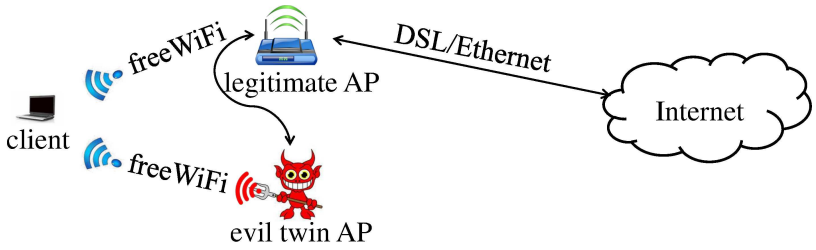


Figure 4.4 Multihop attack model where an adversary uses a laptop's Wi-Fi interface to create an evil twin AP and connects to Internet through the legitimate AP.

Launching Evil Twin AP Attacks

Attacks using Mobile Internet Access

For this attack, the adversary uses mobile Internet, e.g., 2G/3G/4G, as the access network for connecting to the Internet. We denote these attacks as `Mobi Attacks` and depict this attack model in Figure 4.3. There are two types of devices an adversary can use to launch Mobi attacks:

i. **Hotspot Router.** The adversary can use an off-the-shelf hotspot router to create an evil twin AP, e.g., Linksys WRT54G-TM hotspot router. Such routers require mobile Internet subscription for Internet connectivity. The adversary can simply configure the SSID of the AP to the SSID of the Wi-Fi hotspot to launch the attack.

ii. **Smartphone.** A smartphone can act as an AP and thus can allow Wi-Fi clients to use mobile Internet service of the smartphone. Sharing mobile Internet connection via Wi-Fi interface is also known as tethering [31]. The adversary can configure a tethering app with the SSID of the hotspot and turn it on, e.g., Android Wi-Fi Tether app.

Attacks utilizing the Victim AP's Internet Access

This attack eliminates the requirement of a device with mobile Internet service. In this attack, the adversary associates with a victim AP as a Wi-Fi client and shares this Internet connection as an AP. We denote these attacks as `Multihop Attacks`, as shown in Figure 4.4. An adversary can launch this attack using either one or two Wi-Fi interfaces as we discuss in the following.

i. **Single Wi-Fi Interface (`Si-Fi Attack`).** The adversary can install a software to use a laptop as an AP with victim hotspot's SSID, e.g., `Virtual Router` [117]. This software creates two virtual Wi-Fi interfaces, one for associating with the legitimate AP and the other for acting as an evil twin AP, using a single physical interface. In this case, the evil twin AP must use the same wireless channel as the legitimate AP.

ii. **Dual Wi-Fi Interfaces (`Du-Fi Attack`).** In this model, the adversary has two Wi-Fi interfaces. For instance, she can use the built-in Wi-Fi interface and use an additional USB Wi-Fi interface. The adversary can configure one Wi-Fi interface in ad hoc mode using the SSID of the Wi-Fi hotspot. In the ad hoc mode, the device can communicate with other Wi-Fi devices directly and thus pose as an evil twin AP. The adversary can use the other Wi-Fi interface to associate with the legitimate AP and share its Internet connection. In this case, the evil twin AP can use any wireless channel.

The Consequence of Evil Twin AP Attacks

An evil twin AP attack can be used to for stealing sensitive data from a user, or for identity theft, etc. For instance, session hijacking can be used to hack an email account, facebook account, etc [125]. This attack is feasible because most users enable *automatic login* to their online accounts to avoid repeated password entry, e.g., email

accounts. In such a configuration, a session key is saved both on the client side and server side for each account. To steal the session key, the adversary simply injects `<img>` element in the HTTP response message for a user. For instance, let the victim has automatic login setup for `mail.yahoo.com` and he is associated with an evil twin AP. When the victim visits a webpage, e.g., `www.bing.com`, the adversary will inject a `<img src="http://mail.yahoo.com">` element in the HTTP response message. As a result, the victim's browser will transmit yahoo cookies. Thus, the adversary will be able to sniff the cookies and use these to log in to victim's `mail.yahoo.com` account. MITM attacks are possible even when the victim uses Wi-Fi Protected Access (WPA) [82]. For instance, the adversary can be an insider and have the WPA key, e.g., hotel customer.

Threat Model

It is easy for an adversary to create an evil twin AP in a Wi-Fi hotspot using a Wi-Fi-enabled device, e.g., laptop, smartphone, etc. The adversary may have her own access network, e.g., mobile Internet, for Internet connectivity or she may connect her device to a legitimate AP for accessing the Internet. We envision that the evil twin AP may have the following capabilities to avoid detection: it can (a) control the timing for sending out a packet from the client, (b) reply the client with locally generated spoofed packet, (c) modify packet content while forwarding a packet to the client, or (d) use Virtual Private Network (VPN) for tunneling. Irrespective of the above capabilities, the adversary should forward the packets from an associated client for successful attacks. We assume that hotspot providers are trustworthy as they are bounded by several laws [81] and do not collude with the adversary in any way.

4.4 CETAD Overview

The focus of our mechanism is on detecting evil twin AP attacks in wireless hotspots from a client end without any infrastructure support, i.e, the client should be able to use our mechanism to detect an evil twin AP in a known environment as well as in an environment without any prior knowledge. In this section, we identify design requirements, specify the assumptions, give an overview of CETAD, and discuss our observations from several hotspots.

Design Requirements

A client-side mechanism for detecting evil twin AP attacks must fulfil the following requirements.

1. The mechanism must not require any administrative access to the hotspot routers or APs. The client may have such administrative control in a controlled environment, e.g., home network, but he is unlikely to have such control in a hotspot.

2. It must be able to verify an AP in a hotspot and thus cannot assume any custom infrastructure support, e.g., other wireless devices or wireless sensors. Designing a solution with infrastructure support would require hotspot owners to modify hotspots, which is unlikely to happen because hotspots are free services for attracting customers.

3. It must not use any sensors or interfaces available in smartphones so that it can be integrated in any type of Wi-Fi enabled devices. Using a resource that is not universal to all Wi-Fi devices will limit it's use.

4. It must be automated with minimal intervention from users to ensure usability.

Assumptions

We assume the following while designing CETAD.

- The wireless hotspot supports a DHCP server which dynamically assigns network parameters to the clients, e.g., IP address, DNS address, etc. The DHCP server eliminates manual network configuration for Internet access.
- The wireless hotspot does not use WDS architecture.
- The built-in Wi-Fi application associates with the AP that has the strongest RSSI signal.
- The adversary can associate with a legitimate AP similar to a client, but cannot gain administrative access to the infrastructure of the hotspot.
- The hotspot uses one ISP for Internet connectivity.
- A wireless client does not have any prior knowledge about the hotspot.

CETAD Framework Overview

CETAD is designed based on the idea that the ISP information, public IP address of the routers, RTT values of packets travelling through two legitimate APs are similar, but these are different for a legitimate AP and an evil twin AP. This holds for the following cases:

i. *No Attack.* All the legitimate APs will have same ISP and IP address because an organization generally purchase Internet service from one local ISP. Additionally, since the APs will share the same access network, they will have similar RTT values.

ii. *Mobi Attacks.* Since an adversary uses her own access network for these attacks, the global IP address, and ISP of the legitimate AP and the evil twin AP will be

different. Additionally, since the access networks of the legitimate AP and the evil twin AP are different, the RTT values are likely to be different as well.

iii. *Multihop Attacks.* In these attacks, an adversary utilizes the access network of the wireless hotspot; thus the ISP information will not reveal the attack. However, there will still be dissimilarity between a legitimate AP and an evil twin AP in RTT values as the evil twin AP will use a legitimate AP as the next hop.

While the idea is simple, there are several challenges to be addressed: In what degree the IP address and ISP information differ among APs? Can we measure RTT without custom server? How effectively can RTT be used for detecting an attack? Can an adversary manipulate packets to avoid detection? We address these challenges in CETAD and answer the questions in Section 4.4.

As outlined in Algorithm 5, CETAD works in two phases: In (a) `secure data collection phase`, data for all possible APs in the hotspot are collected, and in (b) `detection phase`, detection techniques are applied on all the AP data for an evil twin AP attack detection.

Algorithm 5 Evil Twin Attack AP Detection

Require: INPUT:
$\delta : rssi\_threshold$
PROCEDURES:
/* data collection phase */
1: $ap.list = Scan()$
2: **for each** a in $ap.list$ **do**
3: **if** $a.rssi \geq \delta$ **then**
4: $data[a] = CollectData(a)$
5: **end if**
6: **end for**
/* detection phase */
7: $detect = DetectEvilTwinAP(data)$
8: **return** $detect$

Secure Data Collection Phase

In this phase, CETAD communicates with public servers leveraging widely used Hypertext Transfer Protocol Secure (HTTPS) protocol (details in Section 4.5). First it scans the wireless hotspot to detect available APs with the desired SSID. From the AP list, it eliminates APs with low RSSI value by applying a threshold δ. Then, for each of the available APs with RSSI value greater than δ, the mechanism collects data in the following 3 steps.

- **Step 0:** It associates with an AP and collects DHCP information for the Wi-Fi network. This is the first step because for collecting ISP and RTT data, the client must connect to Internet by associating with an AP.
- **Step 1:** It collects ISP information of the AP by contacting a public server. This information is used in ISP-based scheme to find correlation in ISP information of the APs.
- **Step 2:** It collects multiple RTT values by creating an HTTPS connection to a public server. The RTT values are used in the timing-based scheme.

Detection Phase

CETAD detects evil twin AP attacks by classifying multiple APs in a hotspot. After the data collection phase completes, CETAD uses a multi-level approach to classify the APs based on data of all APs. First, it applies ISP-based scheme which utilizes ISP data to identify whether two APs are using the same ISP for Internet access and this scheme is enough to detect *Mobi attacks* (Section 4.3). In the next level, CETAD uses a timing-based scheme utilizing the RTT values to detect *Si-Fi attacks* and *Du-Fi attacks* (Section 4.3). We discuss the details of ISP-based and timing-based detection schemes in Section 4.5.

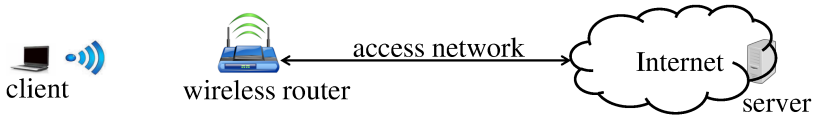


Figure 4.5 Experimental setup for data connection.

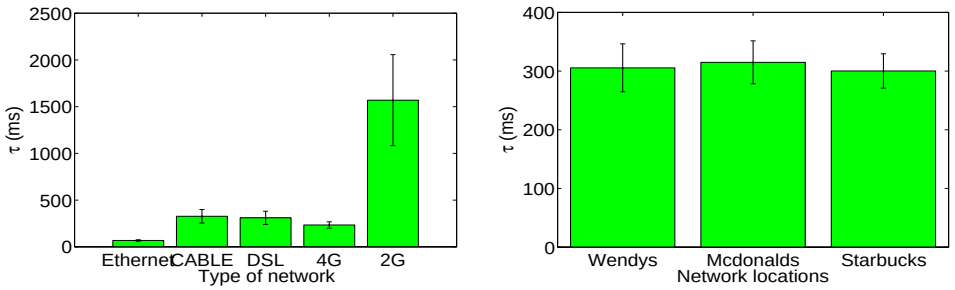
Observations

To address the design challenges, we studied 30 hotspots which includes Mcdonalds, Wendys, Starbucks, university, airport, etc. For each hotspot we collected the ISP information, e.g., public IP address, zip code, ISP name, etc., and timing data by setting up an HTTPS [111] connection to public servers (Section 4.5). Use of public server allows our design to be independent of any custom server. Since it is hard to measure packet level RTT using public server, we redefine RTT as the time to setup an HTTPS connection to a public server and denote it as τ. We show the experimental setup for data collection shown in Figure 4.5. To understand attack scenarios, we set up Si-Fi attacks and Du-Fi attacks, as discussed in Section 4.3, for data collection. We collected data from several locations in the USA and in China. We ran the experiments on both weekdays and weekends, and collected approximately 2000 hours of data. In total we collected more than $5,000,000$ instances of data for the five types of access network. We summarize our observations in the following.

Network Analysis

Based on the data of all hotspots we observe the following.

- Most of the hotspots have two or more APs.
- Each hotspot is connected to one ISP and supports DHCP.
- None of the hotspots utilizes a WDS architecture or uses mobile Internet (3G/4G).



(a) (b)

Figure 4.6 Timing analysis: (a) shows average τ values for different types of access network, and (b) shows average τ values for different hotspots where all hotspots were using DSL as the access network.

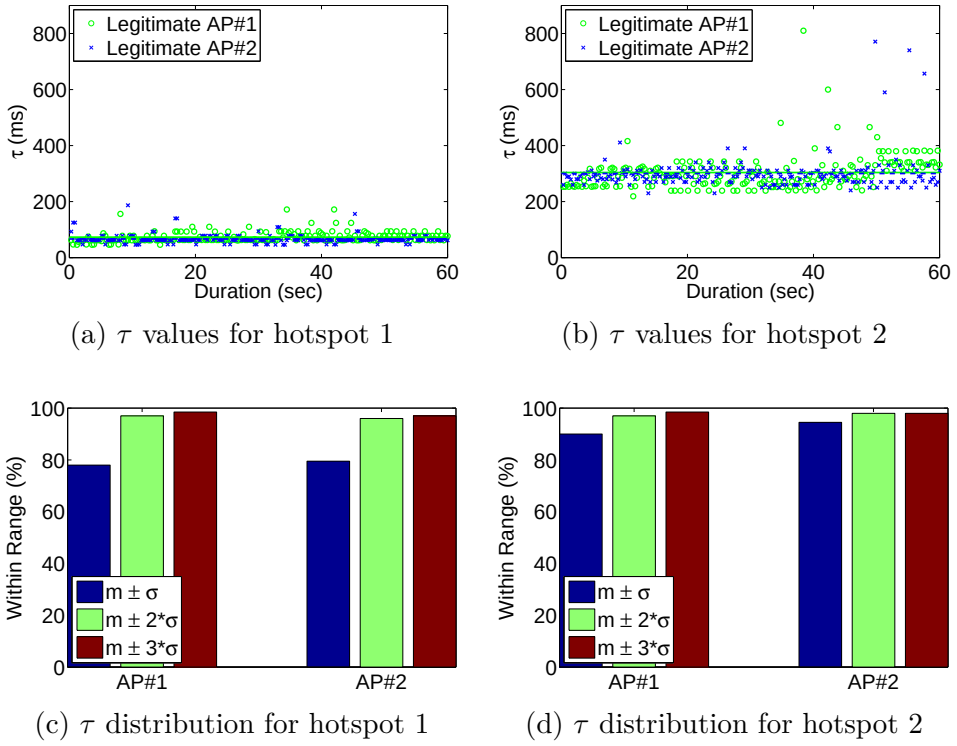
- All the APs in a hotspot use similar configurations, e.g., shared SSID, global IP address, DNS, etc., which is a standard practice to ensure efficient service [58][68], i.e., to allow users to associate with another AP smoothly when he changes his location.
- The ISP information of a legitimate AP, and an evil twin AP is different.
- The built-in Wi-Fi drivers of all our devices associate with the AP that has the highest RSSI value.

These observations validate our assumptions and indicate that ISP information can be useful for attack detection.

Timing Analysis

In this section, we discuss our findings from the timing values (τ) we collected from several hotspots.

Comparison among different access networks and hotspots. Here, we compare the τ values based on the type of access networks and location of hotspots for the same type of access network. From the timing data, we observe that the average value of τ varies for different access networks but cannot be used to identify or



(a) τ values for hotspot 1

(b) τ values for hotspot 2

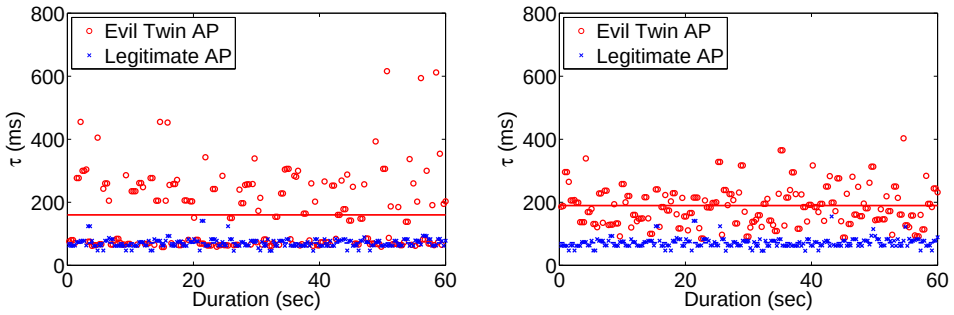
(c) τ distribution for hotspot 1

(d) τ distribution for hotspot 2

Figure 4.7 Timing data analysis of 2 legitimate APs in 2 different hotspots: hotspot 1 uses Ethernet and hotspot 2 uses DSL as access network. Each hotspot consists of 2 APs.

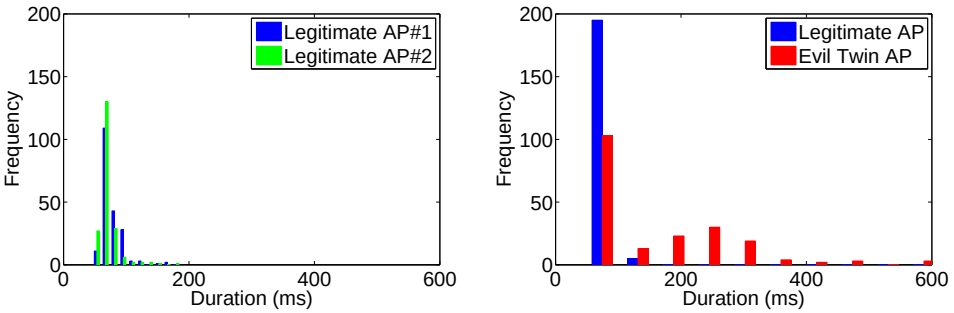
differentiate an access network. Figure 4.6(a) depicts the average values and standard deviation of τ for five different access networks. According to our experiments, the average τ is the lowest (68.3 ms) for a high speed Ethernet network, and highest (1.57 seconds) for 2G network. The average values of τ for DSL and Cable networks are similar, approximately 300 ms. This rules out the option that one can use τ values to identify the type of an access network.

Interestingly, we also observe that for the same type of access network, e.g., DSL, the average values of τ are similar even though the network locations were different. In Figure 4.6(b), we show average values of τ for three different hotspots all of which were using DSL as the access network. As we can see, the average value of τ for all the three hotspots are approximately 300 ms. However, this does not help us in detecting evil twin AP attacks.



(a) τ values for Si-Fi attack (b) τ values for Du-Fi attack

Figure 4.8 Timing data analysis for Si-Fi and Du-Fi attack scenarios where the hotspot uses Ethernet as access network.



(a) no attack (b) Si-Fi attack

Figure 4.9 Histogram analysis of τ value for no attack and Si-Fi attack scenarios where the hotspot uses Ethernet as access network.

Comparison among different APs in the same hotspot. We depict τ values for two different hotspots in Figure 4.7(a)-(b), where one hotspot uses Ethernet and the other uses DSL as the access network. As we can see, the average of τ for one-minute period is approximately 70 ms for Ethernet and 300 ms for DSL network. In no attack scenarios, τ values vary over time due to network load, number of users, etc., but the variance is small within a short time interval. Additionally, τ values for different APs in the same hotspot are similar and the mean values are close. As we see in Figure 4.9(a), τ values for both the legitimate APs are centered around 70ms mark. Even though there are some random spikes in τ values, majority of the values can be grouped within a range using mean (m) and standard deviation (σ) of the

values. For instance, in all cases, more than 98% of the values are within a range of $m \pm 2 * \sigma$. We depict the percentage of τ values within range for different ranges in Figure 4.7(c)-(d).

On the other hand, in Si-Fi and Du-Fi attack scenarios, the τ values through an evil twin AP varies widely compared to a legitimate AP, as depicted in Figure 4.8. Additionally, the mean of τ is much larger for an evil twin AP than for a legitimate AP for both attack scenarios. The histogram in Figure 4.9(b) shows that τ values are distributed over a long range with high variance for an evil twin AP in Si-Fi attack scenario. We observe similar distribution in Du-Fi attack scenario as well. Several factors may cause such high variance of τ values in the attack scenarios. In Si-Fi attacks, there are additional software processing delay. Additionally, since two virtual interfaces use the only physical interface and single wireless channel for communication, there could be additional delay due to added collisions in the Medium Access Control (MAC) [58] layer. In Du-Fi attacks, the increase in delay could be caused by packet forwarding delay from one interface to another.

Summary

Our network analysis shows that ISP data are similar for different legitimates APs in the same hotspot. From our timing analysis, we find that timing values vary for different types of access networks, different time of a day, different locations, etc. However, for the same AP in a hotspot, the timing values does not vary much within a short interval. Our analysis on τ values for different APs in the same hotspot suggests that it is feasible to use τ values effectively for detecting evil twin AP attacks. In Section 4.5, we discuss the design methodology of CETAD based on these observations.

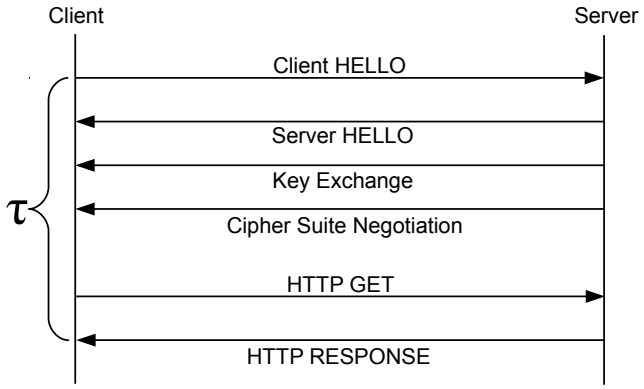


Figure 4.10 Simplified HTTPS connection setup protocol between the client and the server. τ is the HTTPS connection setup time between the client and the server.

4.5 CETAD Description

In this section, we discuss secure data collection phase, and detection phase of CETAD and perform security analysis.

Secure Data Collection

After associating with an AP in a hotspot, CETAD leverages public HTTPS servers to find ISP information of the AP and to measure τ. HTTPS allows us to ensure the integrity of the packets sent and received by CETAD, since an evil twin AP can try to change or manipulate the content of the packets to avoid detection when a packet goes through it. In the following, we first give an overview of HTTPS and then discuss data collection steps.

HTTPS

HTTPS is a secure communication protocol which is designed by adding the security capabilities of SSL/TLS[5] protocol [111] over HTTP. HTTPS is an application layer protocol and uses public key infrastructure (PKI) to exchange short term session keys

[5]Secured Socket Layer (SSL)/Transport Layer Security (TLS)

between the client and the server. Session keys are used to encrypt the data packets that are transmitted over HTTP. A simplified HTTPS connection setup process is depicted in Figure 4.10. First, the client sends a `Hello` message to the server and the server replies with a `Hello` message. Then the client and the server exchange their public keys upon validation, and agree on cipher suites to use for communication. In these steps, the server creates a unique hash and encrypts it using both the client's public key and its private key, and sends this back to the client. This ensures that only the client is able to read the hashed data. At this point, the connection setup is complete, and the client and the server can securely exchange information.

Collecting Data

ISP information for an IP address is publicly available through several HTTPS servers and CETAD uses HTTPS GET to collect the data for the AP. To calculate τ, CETAD measures the difference between initiating an HTTPS connection and establishing the connection. In particular, CETAD measure the difference between the moments when the `Client HELLO` is sent and the one when `HTTP RESPONSE` is received. We choose this difference as τ because it ensures that secure connection is established properly. Additionally, this will also simplify implementation process as most HTTPS library APIs hide the key negotiation from the user. CETAD records n values of τ for each AP to improve detection accuracy (Section 4.5).

Algorithm 6 Function: DetectEvilTwinAP()

Require: **INPUT:** $data : isp\_data, timing\_data$
PROCEDURES:
1: $n = 0$
2: $n = ISP\_Detection(data)$
3: **if** $n == 1$ **then**
4: $n = Timing\_Detection(data)$
5: **end if**
6: **if** $n > 1$ **then**
7: **return** *true*
8: **end if**
9: **return** *false*

Detection Phase

As we show in Algorithm 6, our detection mechanism works in two phases; when ISP-based detection scheme does not detect an attack, timing-based detection schemes kicks in. We discuss these schemes in the following.

ISP-based Detection.

A wireless hotspot is generally a small local network of a few computers. In order to provide Internet service, each network must have a gateway with a global IP address. Each ISP gets a block of IP addresses and provides at least one unique global IP address to the customers upon subscription. Internet Assigned Numbers Authority (IANA) [132] is the authority to manage global IP addresses all over the world. The information related to each global IP address is public, e.g., the name of the organization it is assigned to, assignment date, location, etc. These information are publicly available in various website, e.g., `www.arin.net, www.ip2location.com`. CETAD sends an HTTP GET request to such a server and from the HTTP REPLY packet, it gets several information including the source IP address of that AP's network, ISP information of the source IP address, location, etc. We show some public information obtained by this method for two random IP addresses in Table 4.1.

In wireless hotspots, the legitimate APs are connected to the same router sharing the same global IP address or at least the same ISP. CETAD uses these public IP

Table 4.1 Sample public information for two random global IP addresses.

Info	**208.54.44.158**	**171.65.95.202**
Location	Doraville, GA	Palo Alto, CA
ZIP Code	30340	94301
Net Speed	DSL	COMP
Usage Type	MOB	EDU
Domain	t-mobile.com	stanford.edu

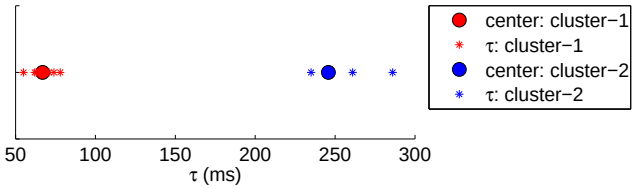


Figure 4.11 Example of clustering scheme where MSC algorithm returns two clusters in a Si-Fi attack scenario.

information to differentiate between APs, i.e., to detect whether two APs use the same IP address block from a ISP. Using this scheme, CETAD can easily detect an evil twin AP attack when the adversary launches a Mobi attack. CETAD matches one or more information from Table 4.1 to classify the APs and thus detects an evil twin AP attack.

Timing-based Detection.

In this section, we discuss how we can use τ values for evil twin AP attack detection. As we discussed in Section 4.4, in non attack scenarios, the values of τ for the same hotspot are similar within a short interval. On the other hand, when an evil twin attack is launched utilizing the victim's Internet (Section 4.3), τ varies significantly. We utilize the temporal values of τ to detect an evil twin attack in two ways: one is by using unsupervised clustering to cluster data in more than one group, and the other is by investigating the standard deviation of the data. We discuss these approaches in the following.

Unsupervized Clustering. In this scheme, we try to cluster APs in a hotspot based on the similarity in the τ values. Since the τ values can vary significantly based on time, hotspot location, network load, etc., a naive threshold-based approach would be ineffective. So, we employ *unsupervised clustering* algorithm for effective detection of evil twin AP attacks. In particular, we use Mean Shift Clustering (MSC) algorithm [28] for clustering the APs which does not require any prior training. We

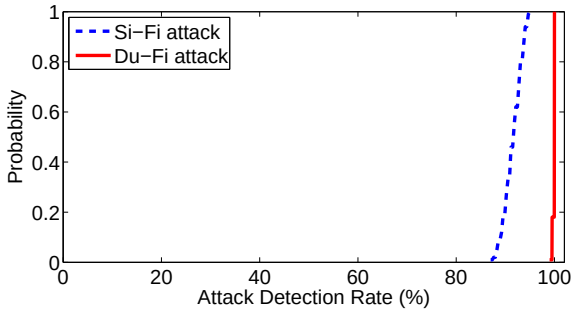
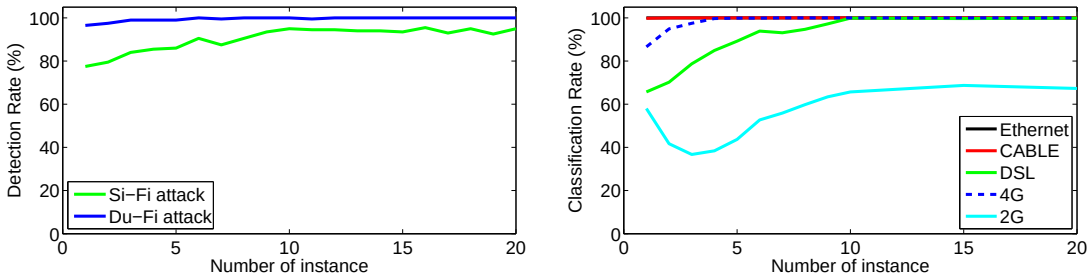


Figure 4.12 CDF of attack detection rate when clustering technique is used.

observed some random spikes in τ values (Figure 4.7) for different hotspot. Since such noisy data can affect the clustering algorithm, in our scheme, we first remove noisy τ values for each AP by applying a filter of $m \pm k * \sigma$, where m is the mean, and σ is the standard deviation of τ and k is a tunable factor. After removing noisy data, we supply AP list with corresponding τ values as input to MSC which returns one or more clusters with similar APs grouped together in the same cluster. When the clustering algorithm returned more than one clusters, we identify the scenario as an evil twin AP attack. Thus, MSC returns one cluster in no attack scenarios and two, or more clusters in attack scenarios. We show an example output of clustering scheme for a Si-Fi attack scenario in Figure 4.11.

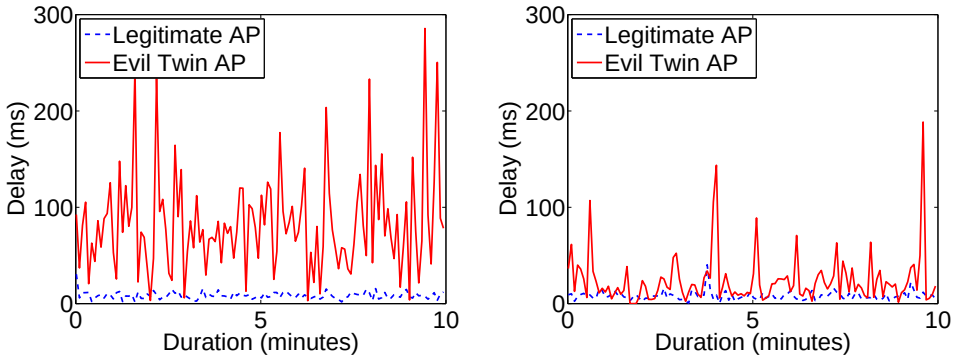
To analyze the algorithm, first we created a dataset by randomly selecting $n = 10$ values of τ from each of the legitimate APs, and evil twin APs for the same time period. Then, we ran the MSC algorithm to generate clusters. When the clustering algorithm returned more than one clusters, we identified the scenario as an evil twin AP attack. We ran the experiment 1000 times, and calculated the accuracy of our algorithm in detecting the attack. We depict the Cumulative Distribution Function (CDF) of the detection accuracy in Figure 4.12. As we can see, in the case Si-Fi attack, this technique can achieve 95% detection rate with the lowest detection rate of 87%. On the other hand, for Du-Fi attack, the detection rate is 99.5% on average with the lowest detection rate of 99%.



(a) attack scenario (b) no attack scenario

Figure 4.13 A comparison of average accuracy of unsupervised clustering as the number of instance for each cluster grows: (a) shows the attack scenario where the accuracy reached 95% for Si-Fi attack when $n = 10$ and 100% for Du-Fi attack when $n = 7$; (b) shows no attack scenarios where each dataset only contains instances from one class and in this case accuracy reached 100% when $n = 10$.

To find the optimum value for n, we ran another set of experiment by varying the values of `n`. As depicted in Figure 4.13(a), the clustering accuracy was 77.5% for the Si-Fi attack when $n = 1$. The accuracy gradually increased as the value of n increased and it achieved the maximum of 95% accuracy when $n = 10$. For Du-Fi attack, the accuracy was 96% for $n = 1$ and the maximum accuracy of 100% was reached with $n = 7$. The result indicates that 10 instances are necessary to achieve good result. To simulate a regular scenario (no attack), we generated dataset with instances of the legitimate APs only and ran the clustering algorithm. In this case, the algorithm should return only one cluster as there is no attack and the dataset only contains legitimate APs. We ran the experiment for different values of n in this scenario as well. We depict the clustering accuracy for each access network type as the number of instance in the dataset increases in Figure 4.13(b). Except for 2G network, we find that we can achieve a 100% clustering accuracy when we have 10 instances in the dataset. For Ethernet, Cable and 4G network, 100% can be achieved with using 4 instances. Even though the accuracy for 2G network is not good (approximately 65%), it is highly unlikely that a wireless hotspot will use 2G network as an access network and we did not find any hotspot with 2G access network.



(a) Si-Fi attack (b) Du-Fi attack

Figure 4.14 Comparison of standard deviation in HTTPS connection setup time between an evil twin AP and a legitimate AP where the evil twin AP utilizes a legitimate AP as it's next hop; (a) shows Si-Fi attack scenario and (b) shows Du-Fi attack scenario.

Standard Deviation Analysis. In Section 4.4, we observed that τ values vary significantly in attack scenarios. This indicates that the standard deviation of τ would be larger for an evil twin AP compared to a legitimate AP. On the other hand, the standard deviation would be similar for two legitimate APs in the same hotspot. We depict the standard deviation of τ for every 10 sec interval in Figure 4.14, where (a) shows the standard deviation for Si-Fi attack, and (b) for Du-Fi attack. As we can see, the standard deviation of τ fluctuates a lot in the attack scenarios compared to regular (no attack) scenarios. Based on this observation, we devised a detection technique where we compare the standard deviation of τ for different APs. In this scheme, we use a threshold γ to detect attack. We calculate the difference of standard deviation between a pair of APs in a hotspot; if difference of standard deviation exceeds the threshold γ, then we identify the case as an attack scenario.

Using the same dataset used in the clustering technique, we analyzed the effectiveness of this technique. We depict the CDF of detection rate for this technique in Figure 4.15. As we can see, this technique achieved the maximum of 95.5% detection rate for Si-Fi attack scenario and average detection rate was 91.7%. However, for the Du-Fi attack scenario, this technique did not perform well; it achieved approximately

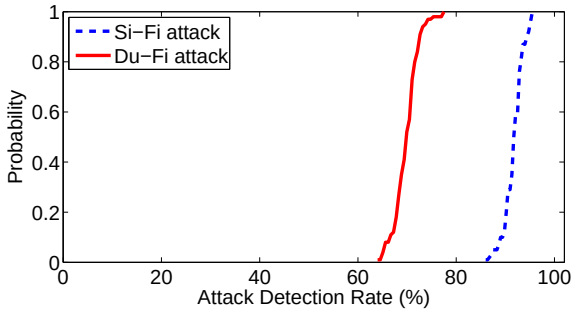


Figure 4.15 CDF of attack detection rate when standard deviation technique is used as the detection technique.

70% detection rate on average. The performance is not good for the Du-Fi attack scenario compared to Si-Fi attack scenario because the standard deviation of τ for Du-Fi attack scenario does not vary much compared to the Si-Fi attack scenario.

Combined Detection Technique. Both the clustering technique and standard deviation technique performed relatively well. However, since the techniques are independent of each other, we wanted to see how they perform in combination. In the combined technique, CETAD detects an attack when either the clustering or standard deviation technique detects an attack. To analyze the performance, we created dataset containing data for regular scenario, Si-Fi attack scenario, and Du-Fi attack scenario. Then we applied the three techniques, i.e., clustering, standard deviation, and combined, on the same dataset. We depict the accuracy, precision, and recall of these techniques in Figure 4.16. As we can see, the accuracy, precision and recall increased when both the clustering and standard deviation are combined. Consequently, we use both these techniques in our timing-based detection algorithm.

The experimental results show that the timing-based scheme is very effective in detecting an evil twin AP attack that utilizes a legitimate AP's Internet. This is because CETAD captures temporal network behavior in the dataset because it collects data for all APs within a short period of time. Even though τ may vary at different time of day or at different locations, CETAD automatically adapts to that.

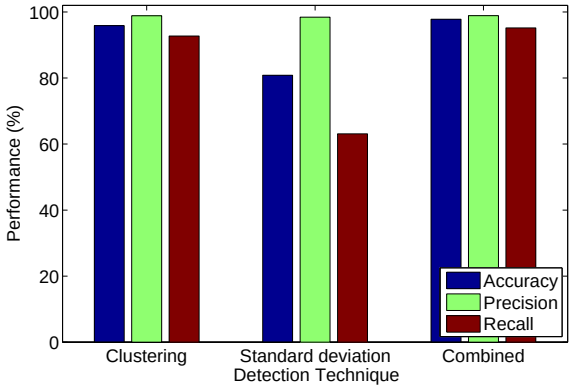


Figure 4.16 Performance of clustering and standard deviation techniques when used independently and in combination. Here, the dataset contains τ values collected in the lab environment.

Security Analysis

The security of CETAD depends on secure data collection and the combination of the three techniques it uses in two phases, i.e., ISP data comparison, unsupervized clustering technique, and standard deviation analysis. The use of HTTPS in the data collection phase ensures that an adversary cannot change source IP address or packet contents, generate a fake message, or shorten τ values to avoid detection. The adversary can at most buffer a packet which will increase τ, but cannot reduce it any way. Thus, adversary cannot evade CETAD by falsifying data. In the following, we analyze security for detection phase for three scenarios.

No Attack

The ISP-based scheme correctly detects no attack scenarios because the ISP information of multiple APs are similar. The timing-based techniques also detect this case correctly due to the similarity in τ values of the legitimate APs. However, there could be false detection in the case when the τ values vary abruptly for two legitimate APs.

Mobi Attacks

The ISP-based technique ensures that the adversary will not be able to use his own access network to setup an evil twin AP attack. This also eliminates the possibility that the adversary can use an access network with faster τ compared to the hotspot and thus control the τ values to evade the timing-based technique.

Multihop Attacks

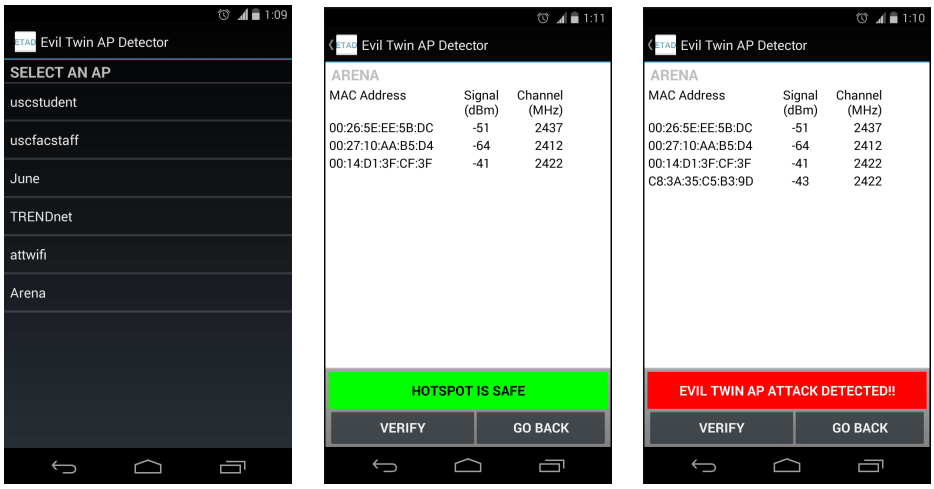
In multihop attacks, the adversary sets up Si-Fi attacks or Du-Fi attacks. Since both these attacks use a legitimate AP as the next hop, intuitively she cannot control the value τ to be similar to that of a legitimate AP. On top of the delay from a legitimate AP, the τ value will also depend on the processing delay of software or hardware, Wi-Fi MAC layer collisions, access control, etc. Thus, the adversary cannot control the value of τ at the client for evading timing-based techniques of CETAD.

4.6 Implementation and Results

In this section, we discuss implementation details of CETAD and present experimental results from several hotspots.

Implementation

We implemented CETAD as an app for Android based phones as a case study, however it can be implemented for other Wi-Fi devices as well. The app first scans and shows a list of available unique SSIDs. When the user chooses one of the SSIDs from the list, the app creates an AP list with the selected SSID and removes the APs with signal strength below threshold δ, where we used $\delta = -75dBm$. Then for each AP in the list, the app first collects ISP data by using HTTP GET connection to `www.ip2location.com`, and then it collects n instances of τ values by opening an



(a) Home Screen (b) Regular Scenario (c) Attack Scenario

Figure 4.17 Screenshots of CETAD app in regular and attack scenarios.

HTTPS connection to a public HTTPS server, e.g., `www.google.com`. We used $n = 10$ in our implementation based on our analysis (Section 4.5). Note that some hotspots require users to submit a web form by agreeing to their *terms and conditions*. To automate this step, we use `automatic HTML form submission` technique. However, this technique might fail if the form has protection against automatic submission, e.g., captcha.

After collecting ISP and timing data for all the APs in the list, the app first uses ISP-based detection technique. If ISP-based detection technique cannot detect an attack, then timing-based detection technique is used, which first filters the data using a range of $m \pm k * \sigma$ (Section 4.5). We use $k = 2$ in our implementation. Then, it uses mean shift clustering algorithm to cluster the τ values for different APs. Apart from the dataset, MSC algorithm requires only one input parameter, window size, h. We used $h = 30ms$ in our implementation. Our algorithm detects an attack when the number of clusters in the input is more than one. Then the mechanism calculates the standard deviation of τ values and applies a threshold value γ to detect an attack. We used $\gamma = 30ms$ in our implementation. When either of the above techniques

detects an attack, the app identifies such a scenario as an evil twin AP attack and notifies the user. We depict the screenshots of our app in Figure 4.17.

Experimental Results

To measure the effectiveness of our mechanism, we launched two types of evil twin AP attack in several hotspots[6]:

i. **Mobi attacks:** We used a Google Nexus 4 Android phone with 3G data subscription to launch smartphone-based attacks with our own access network, as we discussed in Section 4.3. We used `android-wifi-tether` to setup the evil twin AP for our attack.

ii. **Multihop attacks:** We used a laptop with Windows-7 operating system to launch a Si-Fi attack. We used `Virtual Router version 1.0` for this attack. To launch a Du-Fi attack, we used an additional USB Wi-Fi interface and a laptop with Windows-7 operating system. We used `Medialink Wireless-N USB adapter` for our attack.

In each hotspot, we ran CETAD 10 times for each type of the attacks. The hotspot providers include McDonalds, Starbucks, Wendys, University, etc. For the performance analysis, we use the following standard metrics: (a) *Accuracy* indicates how accurately CETAD detects evil twin AP attacks, (b) *Precision* is the fraction of positively detected attacks to all positively detected attacks (correctly or incorrectly), and (c) *Recall* is the fraction of positively detected attacks to all attacks that should be positively detected. We use True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) to calculate these metrics. Here TP and TN

[6]Disclaimer: Several unknown Wi-Fi clients were associated with our evil twin AP during the attack, but we did not record any kind of data from these clients during the attacks.

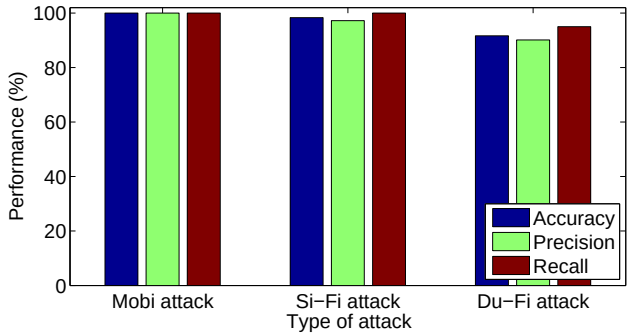


Figure 4.18 Performance of CETAD for three different types of evil twin AP attacks in several Wi-Fi hotspots in different locations. The hotspots include McDonalds, Starbucks, Wendys, University, etc.

represent correct classification, and FP and FN represents incorrect classification. The equations for calculating these metrics are as follows.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

$$precision = \frac{TP}{TP + FP} \tag{4.2}$$

$$recall = \frac{TP}{TP + FN} \tag{4.3}$$

Performance

Our experimental results show that CETAD is highly effective in detecting evil twin AP attacks in wireless hotspots as discussed below.

Mobi attacks. CETAD detected Mobi attacks in all cases, i.e., with 100% accuracy, precision, and recall. For this type of attack, our ISP-based scheme was enough to identify the attacks as the ISP information of a legitimate AP, and an evil twin AP were different. For instance, in all cases they had different public IP addresses, zip codes, and usage types. (Table 4.1).

Multihop attacks. For Si-Fi and Du-Fi attacks, our timing-based scheme was useful because ISP-based scheme cannot detect such attacks as a legitimate AP is used as a gateway by the evil twin AP in these attacks. We depict the performance

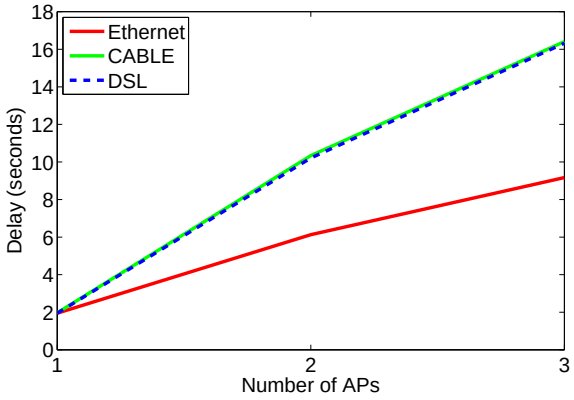


Figure 4.19 Overhead of CETAD in different types of wireless hotspots. With one AP, our mechanism returns after DHCP discover which indicates that DHCP requires approximately 2 sec for each AP.

of CETAD in detecting Si-Fi attacks and Du-Fi attacks in Figure 4.18 which shows that the accuracy, precision, and recall of our timing-based scheme for Si-Fi attack is 98.33%, 97.22%, and 100.0% respectively; and for Du-Fi attack, it achieved 91.67%, 90.15%, and 95.0% accuracy, precision, and recall respectively.

In summary, CETAD can effectively detect different types of evil twin AP attacks in wireless hotspots.

Delay Overhead

Admittedly, our mechanism comes with some delay overhead, since the client application needs to associate with the candidate APs and collect ISP data, and n values of τ. Then after the attack detection phase, we allow the user to associate with a legitimate AP. All these steps incur delay overhead. To analyze the delay overhead, we measure the time difference between the time when the user starts verification and the time when the application shows notification to the user. We depict the delay overhead in Figure 4.19. When there is only one AP in the AP list, our mechanism does not use the detection scheme because it requires at least 2 APs for detection. In this case, the mechanism returns after DHCP operation is completed. As we can see,

the DHCP configuration takes approximately 2 seconds for each AP. As the number of APs increase, the delay overhead increases monotonically. This delay mainly consists of the DHCP delay and delay for collecting n values τ, where $n = 10$. For the three AP scenario, the detection mechanism was complete within 9.2 seconds for Ethernet network and it took approximately 16.4 seconds for DSL network. Despite the small delay overhead, our mechanism is able to detect evil twin AP attacks effectively.

4.7 Related Works

In this section, we discuss rogue AP detection schemes in general rather than limiting to evil twin AP detection only. The current rogue AP detection schemes can be classified into two categories; (i) Solution for infrastructure networks, e.g., corporate networks, school , and (ii) Solution for client devices.

Solutions for Infrastructure Networks

The goal of these solutions is to help the network administrators to detect a rogue AP and possibly locate it. These can be divided into two categories:

Hybrid Approach

Branch et al. [19] designed a Distributed Wireless Security Auditor (DWSA) to provide continuous wireless network assessments. The proposed system uses available, trusted wireless clients as distributed anomaly sensors throughout a company's network infrastructure to collect periodic activity report on AP. A back-end server detects rogue and misconfigured APs and subsequently locates them via 3D trilateration. The back-end server utilizes a list of authorized access points to determine rogue AP. Athawale et al. [8] proposed a solution that uses mobile agents to quickly scan all possible rogue APs, without generating additional network loads on the network, hence reducing scan and reporting time. Bahl et al. [12] proposed Dense Array of

Inexpensive Radios (DAIR), a framework for detecting rogue APs attached to corporate networks. DAIR can be built using commodity USB-based wireless adapters connected to PC. In this framework, wireless and wired data are collected and saved in a database. The inference engine of the framework compares the data with known AP list and thus detects rogue AP. Ma et al. [77] proposed a system where wireless data is collected in promiscuous mode and passed to rogue AP detection engine, which uses AP probing, and OS fingerprint to detect rogue AP.

Wired Approach

This type of approaches monitor wired traffic at the gateway and determine whether the client uses a wired or wireless connection. Wei et al. [123] detects rogue AP by analyzing packet header at the routers. They proposed two algorithms that uses TCP ACK-pair technique [122] to differentiate wired and wireless LAN TCP traffic, and exploit the fundamental properties of the 802.11 CSMA/CA MAC protocol as well as the half duplex nature of wireless channels. Beyah et al. [16] proposed the use of temporal traffic characteristics to detect rogue APs at a central location (a switch). The hypothesis used in the proposed solution is that a wireless link in a network path of multiple links would cause a more random and temporally different spreading of packets, as compared to a path that has only wired links. Watkins et al. [121] proposed a similar approach where they used round trip time (RTT) to distinguish between wired and wireless nodes. This information coupled with a standard wireless AP authorization policy allows the differentiation between wired nodes, authorized APs, and rogue APs. Shetty et al. [107] identifies unauthorized WLAN hosts connected to rogue access points by analyzing traffic characteristics at the edge (router or gateway) of a network. The scheme is based on the frequency of access of a particular port and the increase in cross-port communication. It analyzes the traffic to compute the frequency of straight-access (AP accesses the port it is physically connected to) and

crossing-access (AP accesses the port it is not physically connected to) attempts. A rogue AP is detected when the frequency values of these access attempts exceeds a threshold.

Summary

Both the above approaches require administrative access to the network switch or routers. Additionally, the hybrid approaches require to install additional devices. Hotspot providers are unlikely to utilize these solutions as there is no incentive for them. CETAD does not fall into the infrastructure category because we aim at detecting attack from a client device. However, we can extend CETAD as infrastructure solution as well by deploying client devices with CETAD installed to notify the administrators.

Solutions for Client Devices

Gonzales at el. [45] proposed a scheme that first trains a model in a hotspot by creating an AP map based on the user location. Subsequently, the trained model is used to detect evil twin AP by detecting change in the AP map. This scheme cannot detect an evil twin attack if it is ongoing during the training. Moreover, it is dependent on user's location and cannot be used in a new hotspot without training. In comparison, CETAD can be used in any hotspot immediately without any training.

Song et al. [108] proposed two algorithms to detect rogue AP at the client. The solutions are based on server Inter-packet Arrival Time (IAT) which is the time interval between two consecutive data packets sent from the server to the client. The solutions requires to setup a server within the LAN with their software installed to measure server IAT. Based on the server IAT, the paper proposes a Trained Mean Matching Algorithm (TMM) and a Hop Differentiating Technique (HDT) for detecting an evil twin AP. Our work solves a similar problem, but we do not have the

rigid requirement of having a custom server in the LAN, rather we leverage public web server in our mechanism. Additionally, the proposed work cannot detect an evil twin when an adversary uses Mobi attack, whereas CETAD can detect evil twin AP attacks in various threat models with high accuracy.

4.8 Summary

Wi-Fi clients using wireless hotspots are vulnerable to evil twin AP attacks and are in the danger of loosing sensitive information to adversaries. To the best of our knowledge, no mechanism is currently available that can detect evil twin AP attacks without prior training or installing additional hardware. In this chapter, we proposed CETAD, an end-to-end mechanism that can detect evil twin AP attacks in wireless hotspots and does not require to install any hardware or software in the hotspot infrastructure. We implemented CETAD as an app for Android-based phones as a case study and showed that it can be used effectively in wireless hotspots to detect evil twin AP attacks.

Chapter 5

Concluding Remarks

In this dissertation, we applied *end-to-end* principle to solve three different problems in wireless networks. First, we used it as a metric to maximize the availability of paths in multipath selection. We show that end-to-end availability can perform better in selecting fault-independent paths instead of node disjoint or link disjoint paths and thus improve *network reliability*. Then, we applied end-to-end principle to detect caller ID spoofing attacks in telephone networks by passively *authenticating* the caller. Without end-to-end mechanism, the telephone networks may remain vulnerable to caller ID spoofing attacks even with network/protocol redesign. Finally, we utilized end-to-end principle to detect evil twin AP attacks in wireless hotspots. Cryptographic solutions, while promising, will take away usability of the wireless hotspots, and thus end-user solution is necessary to ensure *secure communication*.

Through the above solutions, in this dissertation, we showed that *end-to-end* principle is an important method that can be effectively applied to solve various real-world problems and we believe that it will continue to be important in the current complex networked world. In an ideal world, it is better to design a complete secure system at the very beginning, but is unlikely to be feasible in the real world because of business decisions, budget limitations, backward compatibility, etc. Thus, end-to-end-based solutions will remain useful to solve loopholes in currently deployed systems as well as in future systems. In the future, we plan to investigate other practical problems in different networks that can be solved by applying end-to-end based solutions.

Bibliography

[1] *GSMK CryptoPhone*.

[2] *R-BGP: Staying Connected In a Connected World*, 2007.

[3] 3GPP, *Ts 24.081*, www.quintillion.co.jp/3GPP/Specs/, 2004.

[4] 3GPP2, *Short message service*, www.3gpp2.org, 1998.

[5] ABCNews, *Caller id scam solicits personal info, money.*, abcnews.go.com/GMA/Consumer/story?id=3305916, 2007.

[6] _____, *Fire! hotels in 4 states hit by prank calls*, http://abclocal.go.com/wpvi/story?section=news/national_world&id=6860704, 2009.

[7] Byron Acohido, *Hacking into voicemail is easy, experts say*, www.usatoday.com/tech/news/ (2011).

[8] S.V. Athawale and S.B. Vanjale, *Detection of rouge access point in 802.11g using MA*, International Journal of Computer Science and Communication (2011).

[9] AT&T, *2.7 billion connections made in 2012; wi-fi network growth doubles*, www.att.com/gen/press-room?pid=23824&cdvn=news&newsarticleid=36066.

[10] _____, *Voice networking solutions*, www.business.att.com.

[11] Seung Jun Baek and Gustavo de Veciana, *Spatial energy balancing through proactive multipath routing in wireless multihop networks*, IEEE/ACM Transactions on Networking **15** (2007), no. 1, 93–104.

[12] Paramvir Bahl, Ranveer Chandra, Jitendra Padhye, Lenin Ravindranath, Manpreet Singh, Alec Wolman, and Brian Zill, *Enhancing the security of corporate Wi-Fi networks using DAIR*, Proceedings of the 4th international conference on Mobile systems, applications and services, ACM, 2006, pp. 1–14.

[13] Vijay A Balasubramaniyan, Aamir Poonawalla, Mustaque Ahamad, Michael T Hunter, and Patrick Traynor, *Pindr0p: using single-ended audio features to determine call provenance*, Proceedings of the 17th ACM conference on Computer and communications security, ACM, 2010, pp. 109–120.

[14] Bell Communication Research, *Bellcore technical specification*, www.morehouse.org/hin/blckcrwl/telcom/callerid.txt, 1984.

[15] Bevocal Cafe, *Supercharge your portal*, cafe.bevocal.com.

[16] Raheem Beyah, Shantanu Kangude, George Yu, Brian Strickland, and John Copeland, *Rogue access point detection using temporal traffic characteristics*, Global Telecommunications Conference (GLOBECOM), vol. 4, IEEE, 2004, pp. 2271–2275.

[17] Pravin Bhagwat, *Bluetooth: technology for short-range wireless apps*, Internet Computing, IEEE **5** (2001), no. 3, 96–103.

[18] Alex Biryukov, Adi Shamir, and David Wagner, *Real time cryptanalysis of A5/1 on a PC*, Fast Software Encryption, Springer, 2001, pp. 1–18.

[19] Joel W Branch, David Safford, Nick L Petroni Jr, and Leendert Van Doorn, *Autonomic 802.11 wireless lan security auditing*, IEEE Security & Privacy **2** (2004), no. 3, 56–65.

[20] British Telecomm, *Sin 227 issue 3.5*, www.btwebworld.com, 2008.

[21] business.comcast.com, *Comcast Cable Communication*.

[22] Mario Cagalj, Srdjan Capkun, and Jean-Pierre Hubaux, *Wormhole-based anti-jamming techniques in sensor networks*, IEEE Transactions on Mobile Computing (TMC) (2007).

[23] Yigang Cai, *Patent application: Validating caller id information to protect against caller id spoofing*, US Patent and Trademask Office, 2008.

[24] Murat Cakiroglu and Ahmet Turan Ozcerit, *Jamming detection mechanisms for wireless sensor networks*, Proceedings of the 3rd international conference on Scalable information systems, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, p. 4.

[25] CallerIDFaker, *Caller id faker-fake a call!*, www.calleridfaker.com.

[26] Yang Cao, Zhimin Liu, and Yi Yang, *A centralized scheduling algorithm based on multi-path routing in wimax mesh network*, International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM), IEEE, 2006, pp. 1–4.

[27] Carnegie Mellon University, *CMU Sphinx*, cmusphinx.sourceforge.net.

[28] Yizong Cheng, *Mean shift, mode seeking, and clustering*, IEEE Transactions on Pattern Analysis and Machine Intelligence **17** (1995), no. 8, 790–799.

[29] Jerry T Chiang and Yih-Chun Hu, *Dynamic jamming mitigation for wireless broadcast networks*, The 27th conference on computer communications (INFOCOM), IEEE, 2008.

[30] Stanley T. Chow, Christophe Gustave, and Dmitri Vinokurov, *Authenticating displayed names in telephony*, Bell Labs Journal (2009).

[31] code.google.com/p/android-wifi-tether/, *android-wifi-tether*.

[32] code.google.com/p/sipdroid/, *Sipdroid*.

[33] US Congress, *Truth in caller id act of 2009*, www.gpo.gov.

[34] Dauphin County, *Dauphin County Emergency Management Agency Year Yearly Statistics*, www.dauphincounty.org, 2011.

[35] Dann Cuellar, *Pranksters terrorize delco family in "swatting" call.*, WPVI-TV, Philadelphia, PA, 2010.

[36] Samir R Das, Elizabeth M Belding-Royer, and Charles E Perkins, *Ad hoc on-demand distance vector (AODV) routing*, IETF RFC 3561 (2003).

[37] Swades De, Chunming Qiao, and Hongyi Wu, *Meshed multipath routing with selective forwarding: an efficient strategy in wireless sensor networks*, Computer Networks **43** (2003), no. 4, 481–497.

[38] F.P. Duffy and R.A. Mercer, *A study of network performance and customer behavior during direct-distance-dialing call attempts in the usa*, Bell System Technical Journal **57** (1978).

[39] Jonathan Eisenzopf, *What you need to know about voice asps*, www.datamation.com, Datamation, 2001.

[40] Jerry Ellig, *Regulatory status of voip in the post-brand x world*, bepress Legal Series, 2006.

[41] ETSI, *UMTS*, www.etsi.org.

[42] ———, *W-CDMA*, www.etsi.org.

[43] ———, *ETSI ES 201 912*, www.etsi.org, 2011.

[44] Andrea Goldsmith, *Wireless communications*, Cambridge University Press, New York, USA, 2005.

[45] Harold Gonzales, Kevin Bauer, Janne Lindqvist, Damon McCoy, and Douglas Sicker, *Practical defenses for evil twin attacks in 802.11*, IEEE Global Telecommunications Conference (GLOBECOM), IEEE, 2010, pp. 1–6.

[46] Google, *Android OS*, developer.android.com/index.html.

[47] Slade E. Griffin and Casey C. Rackley, *Vishing*, Information Security Curriculum Development, 2008.

[48] Network Working Group, *US Secure Hash Algorithm 1 (SHA1)*, tools.ietf.org/pdf/rfc3174.pdf.

[49] Jiawei Han and Micheline Kamber, *Data mining: Concepts and techniques*, Elsevier Inc, 2006.

[50] Olivier Hersent, David Gurle, and Jean-Pierre Petit, *IP Telephony: packet-based multimedia communications systems*, Addison-Wesley, 2000.

[51] Howard Forums Mobile Community, *Can my phone number be hijacked to make calls?*, http://www.howardforums.com/showthread.php/1466560-Can-my-phone-number-be-hijacked-to-make-calls.

[52] http://v4.jiwire.com/search-hotspot-locations.htm, *Jiwire*.

[53] Yih-Chun Hu, David B Johnson, and Adrian Perrig, *SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks*, Ad Hoc Networks **1** (2003), no. 1, 175–192.

[54] Yih-Chun Hu, Adrian Perrig, and David B Johnson, *Packet leashes: a defense against wormhole attacks in wireless networks*, 22nd Annual Joint Conference of the IEEE Computer and Communications (INFOCOM), vol. 3, IEEE, 2003, pp. 1976–1986.

[55] _____, *Ariadne: A secure on-demand routing protocol for ad hoc networks*, Wireless networks **11** (2005), no. 1-2, 21–38.

[56] Hyunuk Hwang, Gyeok Jung, Kiwook Sohn, and Sangseo Park, *A Study on MITM (Man in the Middle) Vulnerability in Wireless Network Using 802.1X and EAP*, Information Science and Security, 2008.

[57] IDC, *Worldwide wlan market reaches nearly 6.4 billion in 2011*, www.idc.com, IDC, 2012.

[58] IEEE, *IEEE 802.11 Wireless Local Area Network*, www.ieee802.org/11/.

[59] IETF, *SIP: Session Initiation Protocol*, www.ietf.org/rfc/rfc3261.txt.

[60] _____, *Domain names - implementation and specification*, www.ietf.org/rfc/rfc1035.txt.

[61] _____, *Dynamic Host Configuration Protocol*, www.ietf.org/rfc/rfc2131.txt.

[62] _____, *ETSI TS 122 083*, www.etsi.org.

[63] ITU, *H.323 : Packet-based multimedia communications systems*, www.itu.int/rec/T-REC-H.323/e.

[64] _____, *Q.431 primary rate interface*, www.itu.int/rec/T-REC-I.431-199303-I/en.

[65] ITU-T, *Session Initiation Protocol (SIP) Extension for Instant Messaging*, www.ietf.org/rfc/rfc3428.txt.

[66] ITU-T, *Q.700*, www.itu.int/rec/T-REC-Q.700-199303-I/en, 1994.

[67] David B Johnson and David A Maltz, *Dynamic source routing in ad hoc wireless networks*, Mobile computing, Springer, 1996, pp. 153–181.

[68] Zaib Kaleem, *Multiple SSIDs*, www.wlanbook.com, 2008.

[69] Chris Karlof and David Wagner, *Secure routing in wireless sensor networks: Attacks and countermeasures*, Ad hoc networks **1** (2003), no. 2, 293–315.

[70] Kevin C. Tofel, *Who has the largest Wi-Fi network in the US? Cable companies say they do*, www.gigaom.com/2013/06/10/who-has-the-largest-wi-fi-network-in-the-us-cable-companies-say-they-do/.

[71] Q. Liang and Q. Ren, *Energy and mobility aware geographical multipath routing for wireless sensor networks*, Wireless Communications and Networking Conference, IEEE, 2005.

[72] Lincoln Emergency Communications Center, *Lincoln emergency communications center annual report*, www.lincoln.ne.gov.

[73] Zhenhua Liu, Hongbo Liu, Wenyuan Xu, and Yingying Chen, *Exploiting jamming-caused neighbor changes for jammer localization*, IEEE Transactions on Parallel and Distributed Systems (TPDS) **23** (2012), no. 3, 547–555.

[74] Daniel Livengood, Jijun Lin, and Chintan Vaishnav, *Public switched telephone networks: A network analysis of emerging networks*, ocw.mit.edu, 2006.

[75] Wenjing Lou, Wei Liu, and Yuguang Fang, *SPREAD: enhancing data confidentiality in mobile ad hoc networks*, 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), vol. 4, IEEE, 2004, pp. 2404–2413.

[76] Ke Ma, Yanyong Zhang, and Wade Trappe, *Mobile network management and robust spatial retreats via network dynamics*, IEEE International Conference on Mobile Adhoc and Sensor Systems Conference, IEEE, 2005, pp. 8–pp.

[77] Liran Ma, Amin Y Teymorian, Xiuzhen Cheng, and Min Song, *RAP: protecting commodity wi-fi networks from rogue access points*, The Fourth International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness & Workshops, ACM, 2007, p. 21.

[78] Mahesh K Marina and Samir R Das, *On-demand multipath distance vector routing in ad hoc networks*, Ninth International Conference on Network Protocols, IEEE, 2001, pp. 14–23.

[79] Ernesto QV Martins and Marta MB Pascoal, *A new implementation of Yen's ranking loopless paths algorithm*, Quarterly Journal of the Belgian, French and Italian Operations Research Societies **1** (2003), no. 2, 121–133.

[80] Ulrike Meyer and Susanne Wetzel, *A man-in-the-middle attack on UMTS*, Proceedings of the 3rd ACM workshop on Wireless security (WiSe), ACM, 2004, pp. 90–97.

[81] Mike Perry, *365-Day: HTTPS Cookie Stealing*, http://fscked.org/talks/ActiveHTTPSCookieStealing.pdf.

[82] Arunesh Mishra and Willian Arbaugh, *An initial security analysis of the IEEE 802.1X standard*, Tech. report, University of Maryland, 2002.

[83] Stephen Mueller, Rose P Tsang, and Dipak Ghosal, *Multipath routing in mobile ad hoc networks: Issues and challenges*, Performance Tools and Applications to Networked Systems, Springer, 2004, pp. 209–234.

[84] Rik Myslewski, *Wireless devices to break one-billion barrier in 2011*, www.theregister.co.uk, The Register, 2011.

[85] Asis Nasipuri and Samir R Das, *On-demand multipath routing for mobile ad hoc networks*, Eight International Conference on Computer Communications and Networks, IEEE, 1999, pp. 64–70.

[86] University of Wasington Network Security Lab, *Jamming framework for ns-3*, Jamming Framework, University of Wasington, 2010.

[87] Valtteri Niemi and Kaisa Nyberg, *UMTS Security*, John Wiley & Sons, 2003.

[88] Guevara Noubir and Guolong Lin, *Low-power DoS attacks in data wireless LANs and countermeasures*, ACM SIGMOBILE Mobile Computing and Communications Review **7** (2003), no. 3, 29–30.

[89] NS-3 Consortium, *Network Simulator 3*, www.nsnam.org.

[90] Toshihiro Ohigashi and Masakatu Morii, *A practical message falsification attack on WPA*, Cryptography and Information Security Conference System, 2009.

[91] Kumiko Ono and Shinya Tachimoto, *SIP signaling security for end-to-end communication*, The 9th Asia-Pacific Conference on Communications (APCC), vol. 3, IEEE, 2003, pp. 1042–1046.

[92] Theodore Paraskevakos, *Decoding and display apparatus for groups of pulse trains*, U.S. Patent No. 3,727,003/4-10-1973, US Patent and Trademark Office, 1973.

[93] _____, *Apparatus for generating and transmitting digital information*, U.S. Patent No. 3,812,296/5-21-1974, US Patent and Trademark Office, 1974.

[94] Vincent Douglas Park and M Scott Corson, *A highly adaptive distributed routing algorithm for mobile wireless networks*, Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), vol. 3, IEEE, 1997, pp. 1405–1413.

[95] Adrian Perrig, Ran Canetti, Dawn Song, and Doug Tygar, *The TESLA broadcast authentication protocol*, RSA Cryptobytes, 2002.

[96] Zbigniew Piotrowski and Piotr Gajewski, *Voice spoofing as an impersonation attack and the way of protection*, Journal of Information Assurance and Security **2** (2007), no. 3, 223–225.

[97] Bruce Potter, *Wireless hotspots: petri dish of wireless security*, Communications of the ACM **49** (2006), no. 6, 50–56.

[98] John G Proakis, *Digital communications*, McGraw-Hill, Columbus, OH, 2000.

[99] QUALCOMM, *Circuit-switched fallback. the first phase of voice evolution for mobile LTE devices*, Tech. report, www.qualcomm.com, 2012.

[100] Rep. Elliot L. Engel, *Rep. Engel Anti-Spoofing Bill Passes House*, http://engel.house.gov/latest-news1/rep-engel-anti-spoofing-bill-passes-house.

[101] Ronald Rivest, *The MD5 Message Digest Algorithm, Request for Comments (RFC)1321*, (1992).

[102] Jerome H Saltzer, David P Reed, and David D Clark, *End-to-end arguments in system design*, ACM Transactions on Computer Systems (TOCS) **2** (1984), no. 4, 277–288.

[103] Akbar Sayeed and Adrian Perrig, *Secure wireless communications: Secret keys through multipath*, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2008, pp. 3013–3016.

[104] Bruce Schneier, *Caller id spoofing*, http://www.schneier.com/blog/archives/-2006/03/caller_id_spoof.html.

[105] Stefania Sesia, Issam Toufik, and Matthew Baker, *LTE: the UMTS long term evolution*, Wiley Online Library, 2009.

[106] Eric Setton, Xiaoqing Zhu, and Bernd Girod, *Congestion-optimized multi-path streaming of video over ad hoc wireless networks*, IEEE International Conference on Multimedia and Expo (ICME), vol. 3, IEEE, 2004, pp. 1619–1622.

[107] Sachin Shetty, Min Song, and Liran Ma, *Rogue access point detection by analyzing network traffic characteristics*, IEEE Military Communications Conference (MILCOM), IEEE, 2007, pp. 1–7.

[108] Yimin Song, Chao Yang, and Guofei Gu, *Who is peeping at your passwords at starbucks?—to catch an evil twin access point*, IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), IEEE, 2010, pp. 323–332.

[109] Patrick Tague, Sidharth Nabar, James Ritcey, and Radha Poovendran, *Jamming-aware traffic allocation for multiple-path routing using portfolio selection*, IEEE/ACM Transactions On Networking **19** (2011), no. 1, 184–194.

[110] Patrick Tague, Sidharth Nabar, James A Ritcey, David Slater, and Radha Poovendran, *Throughput optimization for multipath unicast routing under probabilistic jamming*, IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), IEEE, 2008, pp. 1–5.

[111] Andrew S. Tanenbaum, *Computer networks*, Barnes & Noble, 2003.

[112] Jenn-Yue Teo, Yajun Ha, and Chen-Khong Tham, *Interference-minimized multipath routing with congestion control in wireless sensor network for high-rate streaming*, IEEE Transactions on Mobile Computing **7** (2008), no. 9, 1124–1137.

[113] TrustID, *Automated caller authentication*, www.trustid.com.

[114] Aristotelis Tsirigos and Zygmunt J Haas, *Analysis of multipath routing-part i: The effect on the packet delivery ratio*, IEEE Transactions on Wireless Communications **3** (2004), no. 1, 138–146.

[115] Alvin Valera, Winston Khoon Guan Seah, and SV Rao, *Cooperative packet caching and shortest multipath routing in mobile ad hoc networks*, Twenty-Second Annual Joint Conference of the IEEE Computer and Communications (INFOCOM), vol. 1, IEEE, 2003, pp. 260–269.

[116] Rudolf van der Berg, *The future of interconnection*, 9th Global Symposium for Regulators (GSR), ITU, 2009.

[117] virtualrouter.codeplex.com/, *Virtual-router*.

[118] Voxeo, *Prophecy IVR Platform Software*, www.voxeo.com.

[119] W3C, *Voice extensible markup language (voicexml) version 2.0*, www.w3.org/TR/voicexml20/.

[120] Warrior Forum, *Someone is using my phone number*, http://www.warriorforum.com/off-topic-forum/659448-need-help-someone-using-my-phone-number.html.

[121] Lanier Watkins, Raheem Beyah, and Cherita Corbett, *A passive approach to rogue access point detection*, IEEE Global Telecommunications Conference (GLOBECOM), IEEE, 2007, pp. 355–360.

[122] Wei Wei, Sharad Jaiswal, Jim Kurose, Don Towsley, Kyoungwon Suh, and Bing Wang, *Identifying 802.11 traffic from passive measurements using iterative bayesian inference*, IEEE/ACM Transactions on Networking (TON) **20** (2012), no. 2, 325–338.

[123] Wei Wei, Kyoungwon Suh, Bing Wang, Yu Gu, James Kurose, Don Towsley, and Sharad Jaiswal, *Passive online detection of 802.11 traffic using sequential hypothesis testing with tcp ack-pairs*, IEEE Transactions on Mobile Computing **8** (2009), no. 3, 398–412.

[124] André Weimerskirch and Gilles Thonet, *A distributed light-weight authentication model for ad-hoc networks*, Information Security and Cryptology (ICISC 2001), Springer, 2002, pp. 341–354.

[125] WiFi Foundation, *Legal*, http://www.wififoundation.org/legal.

[126] A Wood, John A Stankovic, and Gang Zhou, *DEEJAM: defeating energy-efficient jamming in IEEE 802.15. 4-based wireless networks*, 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), IEEE, 2007, pp. 60–69.

[127] Kui Wu and Janelle Harms, *On-demand multipath routing for mobile ad hoc networks*, Proceedings of European Personal Mobile Communicatons Conference (EPMCC), Citeseer, 2001, pp. 1–7.

[128] _____, *Performance study of a multipath routing method for wireless mobile ad hoc networks*, Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOT), IEEE, 2001, pp. 99–107.

[129] www.att.com, *AT&T*.

[130] www.counterpath.com/x-lite.html, *X-lite*.

[131] www.fiber.google.com, *Google-fiber*.

[132] www.iana.org, *Internet assigned numbers authority (iana)*.

[133] www.nanpa.com, *North american numbering plan administration*.

[134] www.timewarnercable.com, *Time-warner-cable*.

[135] www.verizonwireless.com, *Verizon-wireless*.

[136] Wenyuan Xu, Wade Trappe, and Yanyong Zhang, *Channel surfing: defending wireless sensor networks from interference*, Proceedings of the 6th international conference on Information processing in sensor networks, ACM, 2007, pp. 499–508.

[137] _____, *Anti-jamming timing channels for wireless networks*, Proceedings of the first ACM conference on Wireless network security, ACM, 2008, pp. 203–213.

[138] Wenyuan Xu, Wade Trappe, Yanyong Zhang, and Timothy Wood, *The feasibility of launching and detecting jamming attacks in wireless networks*, Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing, ACM, 2005, pp. 46–57.

[139] Xiaowei Yang and David Wetherall, *Source selectable path diversity via routing deflections*, ACM SIGCOMM Computer Communication Review, vol. 36, ACM, 2006, pp. 159–170.

[140] Zhenqiang Ye, Srikanth V Krishnamurthy, and Satish K Tripathi, *A framework for reliable routing in mobile ad hoc networks*, Twenty-Second Annual Joint Conference of the IEEE Computer and Communications (INFOCOM), vol. 1, IEEE, 2003, pp. 270–280.

[141] Lianfang Zhang, Zenghua Zhao, Yantai Shu, Lei Wang, and Oliver WW Yang, *Load balancing of multipath source routing in ad hoc networks*, IEEE International Conference on Communications (ICC), vol. 5, IEEE, 2002, pp. 3197–3201.

[142] Xin Zhang, Hsu-Chun Hsiao, Geoffrey Hasker, Haowen Chan, Adrian Perrig, and David G Andersen, *SCION: scalability, control, and isolation on next-generation networks*, IEEE Symposium on Security and Privacy, IEEE, 2011, pp. 212–227.

[143] Xin Zhang and Adrian Perrig, *Correlation-resilient path selection in multi-path routing*, IEEE Global Telecommunications Conference (GLOBECOM), IEEE, 2010, pp. 1–6.