ABSTRACT

Title of Dissertation: PRIVATE COMMUNICATION DETECTION VIA SIDE-CHANNEL ATTACKS

Chang-Han Jong
Doctor of Philosophy, 2012

Directed By: Professor Virgil D. Gligor
Department of Electrical and Computer Engineering

Professor Gang Qu
Department of Electrical and Computer Engineering

Private communication detection (PCD) enables an ordinary network user to discover communication patterns (e.g., call time, length, frequency, and initiator) between two or more private parties. Analysis of communication patterns between private parties has historically been a powerful tool used by intelligence, military, law-enforcement and business organizations because it can reveal the strength of tie between these parties. Ordinary users are assumed to have neither eavesdropping capabilities (e.g., the network may employ strong anonymity measures) nor the legal authority (e.g. no ability to issue a warrant to network providers) to collect private-communication records. We show that PCD is possible by ordinary users merely by sending packets

to various network end-nodes and analyzing the responses. Three approaches for PCD are proposed based on a new type of side channels caused by resource contention, and defenses are proposed. The Resource-Saturation PCD exploits the resource contention (e.g., a fixed-size buffer) by sending carefully designed packets and monitoring different responses. Its effectiveness has been demonstrated on three commercial closed-source VoIP phones. The Stochastic PCD shows that timing side channels in the form of probing responses, which are caused by distinct resource-contention responses when different applications run in end nodes, enable effective PCD despite network and proxy-generated noise (e.g., jitter, delays). It was applied to WiFi and Instant Messaging for resource contention in the radio channel and the keyboard, respectively. Similar analysis enables practical Sybil node detection. Finally, the Service-Priority PCD utilizes the fact that 3G/2G mobile communication systems give higher priority to voice service than data service. This allows detection of the busy status of smartphones, and then discovery of their call records by correlating the busy status. This approach was successfully applied to iPhone and Android phones in AT&T's network. An additional, unanticipated finding was that an Internet user could disable a 2G phone's voice service by probing it with short enough intervals (e.g., 1 second). PCD defenses can be traditional side-channel countermeasures or PCD-specific ones, e.g., monitoring and blocking suspicious periodic network traffic.

PRIVATE COMMUNICATION DETECTION VIA SIDE-CHANNEL ATTACKS

By

Chang-Han Jong

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2012

Advisory Committee:
Professor Virgil D. Gligor, Chair
Professor Gang Qu, Co-Chair
Professor A. Yavuz Oruc
Professor Richard J. La
Professor Katherine J. Stewart

Dedication

To my wife Wenjing, my parents and my grandmother

Acknowledgements

First, I would like to thank Dr. Virgil Gligor for helping me to complete this dissertation. He guided me through this exciting knowledge exploration process. I also need to thank Dr. Gang Qu for his advising and mentoring.

I would like to thank my other committee members, Dr. Richard La, Dr. A. Yavuz Oruc, and Dr. Katherine Stewart for their time and feedback. I am especially glad to have worked with Dr. Stewart and her open source research group in the R.H. Smith School of Business.

I wish to thank my Maryland and Carnegie Mellon colleagues for friendship and inspiring discussions. I am grateful for the support from classmates, choir folks and friends.

I feel so lucky to be raised by parents who allowed me to develop my own interests. My mother's sacrifices were most notable. My sister and brother-in-law were also very supportive, so that I could concentrate on studying. I wish my dear father and grandmother could have seen the completion of this dissertation.

Finally, I would like to thank my wife for her love and support during this journey. It is not scientifically proven, but things are just becoming better and better since I met her.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Privacy has different meanings in different contexts. In communications, a system may provide anonymity, a quantifiable form of privacy, to its users. Side channels are communicational channels that leak information unintentionally. This dissertation will investigate the interaction between side channels and communication anonymity.

1.1. Motivation

Ordinary phone call records - even in the absence of voice/data content - have proven to be exceedingly useful for discovering relationships between private parties. Such records are so useful that both law-enforcement and business organizations have even broken US law to surreptitiously collect these records [49, 50, 69, 72]. Yet one does not actually need to surreptitiously collect 1.9 trillion call records, as the NSA was reported to have done between 2001 and 2004 [10, 49], in order to discover the communication patterns (e.g., call time, length, frequency, and initiator) that could reveal the strength of relationships between private parties [25]. A few thousand records, or less, would suffice, as was the cases in the reported illegal collection of records in incidents by the FBI, and by HP's former Chairman of the Board [50, 69]. Whether collected legally or not, the capacity of relatively few call records to enable detection of private relationships is clearly illustrated by law-enforcement successes worldwide; e.g., the discovery of a drug ring in New Zealand, drug smuggling in Minnesota prisons, and pornography groups on Moldova [49, 72]. Sometimes, call-records analysis is even more favorable than call-content analysis, for example in the

initial phase of surveillance, because the transcription effort of call contents is considerable [72].

Such success in utilizing communication records inspired us to rethink the privacy attacks that are possible in current and future communications networks, which are able to apply strong secure and privacy measures.

1.2. Side Channels and Covert Channels

Side channels [2, 4, 7, 8, 40, 41, 43, 51] and covert channels [26, 29, 42] are two highly-related terms about information leakage in cyber security. The difference is illustrated in Figure 1. These channels are not designed for communication, but they both may be used to obtain or convey information in violation of a network's security policy.

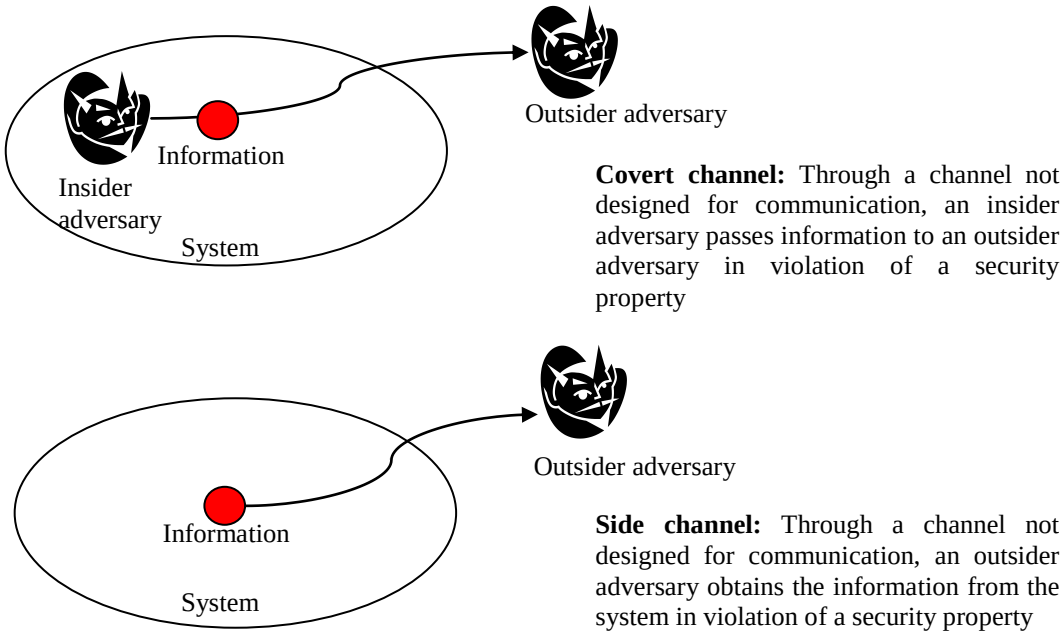


Figure 1. Covert channel and side channel

Suppose a system contains certain information that is desirable by an outsider adversary (or external adversary) and the security policy does not allow such information to be obtained by those outside of the system. Roughly speaking, a side channel is one that an outsider adversary can use to obtain inside information, using it in ways that are not designed for communication. In contrast, a covert channel is one that an insider adversary (i.e., Trojan horse) can use to convey the private information to another outsider adversary by a method that is not designed for communication. The major difference between a side channel and a covert channel lies in that the side channel does not have an insider adversary.

1.2.1. Examples

We show examples for both channels. Huskamp's quantum-time channel is a covert channel that conveys information from a process (insider adversary) in an operating system to another process (outsider adversary) by exploiting the CPU scheduler [29]. The security policy prohibits this information flow. The CPU is a shared resource which is accessible to both processes. Specifically, a process can assign jobs to the CPU (i.e., via write privileges on the shared resource) and check the waiting time of itself (i.e., via read privileges to the shared resource). To execute this convert channel, a process (insider adversary) that wants to send out information, encodes the information to its CPU time use. For example, 0 and 1 are transmitted by using 1 unit and 2 units of CPU time, respectively. Another process (outsider adversary) is able to decode the information by measuring its waiting time. In this case, CPU scheduling is not designed for communication but is used to transmit information.

Kocher's timing attack uses a timing side channel to break public-key cryptography systems [40]. Public-key cryptography systems require substantial computation in calculating exponents. Kocher found that different keys result in different computation times. Therefore, the adversary, who would like to obtain the protected key inside a cryptography device, can use the device to perform cryptography operations (e.g., encryption) and measure the computation times. The adversary can further use the timing information to infer the key stored in the cryptography device. In this case, the time used for computation is not designed for communication, but can be exploited by an adversary.

1.2.2. Terminology

However, the early uses and corresponding definitions of both channels in the context of a single machine do not reflect the advances of communication technologies. Among many different definitions, Wagner and Murdoch provide a new working definition, as described below, for the context of this highly mobilized and wireless networked world [52].

Side channel: A communication channel which violates a security property, but where the sender unintentionally leaks information and only the receiver wishes the communication to succeed.

Covert channel: A means of communicating on a computer system, where both the sender and receiver collude to leak information, over a channel not intended for the communication taking place, in violation of a mandatory access control security policy.

Although the distinction between the side channel and covert channel has been identified [52], the criteria for converting between a side channel and a covert channel has not been discussed in the extant literature. For simplicity, we still use the insider and outsider adversary terms in the rest of this paper. For the side-channel-to-covert-channel conversion, if an insider adversary is added to a side channel, a covert channel is formed if the new insider adversary can decide the information to be received by outsider adversary. For the covert-channel-to-side-channel conversion, if insider adversary of a covert channel is removed, a side channel is formed if the information received by the outsider adversary is of his/her interest.

We demonstrate this conversion through the example of the quantum-time channel, a covert channel. In the quantum-time channel, a process (outsider adversary) can monitor its waiting time to infer the program behavior of another program (information of interest). This is actually a side channel where an adversary, in the form of one process, gets information about another process' behavior. However, this side channel is only meaningful when the adversary regards the program behavior of another process as the information of interest. Moreover, in a system with the given side channel, the quantum-time channel cannot be established if processes cannot affect the CPU scheduler (e.g., round-robin scheduling).

1.3. Anonymity

The term privacy has many definitions, such as "the right to be alone" by Judge Cooley in 1888, and "the condition of not having undocumented personal knowledge about one possessed by others" by Parent in 1983 [19, 58, 75]. As the topic of this

dissertation is on communication systems, we focus on the features of anonymity, a property or subset of the conception of privacy, instead of the broader definition of privacy. This limitation is helpful, since anonymity is quantifiable.

Informally, anonymity is the ability to conduct communication without being identified [52]. Pfitzmann and Hansen gave the following definitions for anonymity and related terms [61].

> **Anonymity of a subject** means that the subject is not identifiable within a set of subjects, the anonymity set.

Anonymity can be defined with respect to different kinds of entities, including the message sender (sender anonymity) and message recipient (recipient anonymity). If the sender of a message cannot be identified among a set of possible senders, sender anonymity is maintained. Similarly, we say that recipient anonymity is provided if the recipient of a message cannot be identified among a set of possible recipients. Also, the concept of anonymity is defined with respect to the messages, based on the definition of Unlinkability [61].

> **Unlinkability** of two or more items of interest (IOIs, e.g., senders, recipients, messages) from an attacker's perspective means that within the system (comprising these and possibly other items), the attacker cannot sufficiently distinguish whether these IOIs are related or not.

In communication systems, the relationship anonymity of two messages refers to the fact that two messages are unlinkable [61]. In other words, if an adversary eavesdrops on messages in multiple locations and can determine whether two messages in

different places are related (e.g., an encrypted message passes through the two places), then there is no relationship anonymity for these messages. In the broader sense, if communication records, including telephone call records, are disclosed, the corresponding relationship anonymity of caller and callee is infringed upon.

1.3.1. Anonymity Systems

Encryption provides data confidentiality such that only the users who have the correct key can decrypt the encrypted messages. In communication networks, whether encryption can be applied to a whole network packet, including headers and payload, depends on the type of the network. In broadcast networks, a whole packet can be encrypted because all users will receive this packet and only those who have the correct key can decrypt it. This also provides anonymity, since the address headers (e.g., a destination address header) are encrypted, meaning that the adversary is unable to identify the sender or the recipient from the encrypted packet. However, in unicast networks, the destination address header cannot be encrypted; otherwise the packet cannot be delivered. An adversary who captures such a packet is able to identify the recipient of this packet, and sometimes even the sender, if the sender address header is not encrypted. In this case, the recipient anonymity (or the sender anonymity when the sender address header is not encrypted) cannot be maintained. To address this problem, anonymity systems, which deliver a message through multiple links, were introduced.

In addition to applying encryption on communication links, an anonymity system consists of the network end-nodes and multiple relaying hosts [17]. The simplest

anonymous system uses a trusted third party that relays the messages between the message sender and the message recipient [15, 17]. If an adversary can only eavesdrop on traffic on one of the two links between the end-node and the relaying host, sender anonymity or the recipient anonymity is maintained. However, if the single relaying host is compromised, the anonymity provided disappears. Hence, other anonymity systems that utilize multiple relaying hosts were introduced [17, 20]. These systems use multiple relaying hosts, such that the partial compromise of relaying hosts would not necessarily threaten anonymity. Packet encryption in anonymity systems is done according to the onion routing, namely a packet is encrypted many times by the sender. When a relaying host receives such a packet, it decrypts and forwards the packet to the next hop [20]. By using onion routing, relaying hosts' views are localized; i.e., a relaying host only knows the previous hop and next hop of the packets, but not the source or ultimate destination. Consequently, the information provided by a compromised relaying host to the adversary is reduced.

1.4. Anonymity, Covert-Channel, and Side-Channel Attacks

Current attacks against anonymity in communication systems can be categorized into five types: malware injection, passive traffic analysis, network watermarking, remote node analysis, and remote traffic analysis. The last three share the property that the adversary needs to insert or modify the network traffic, and therefore they can also be categorized as active traffic analysis. Except malware injection, anonymity attacks are either side-channel or covert-channel attacks, depending on the existence of an insider adversary. The differences between these attacks lie in three aspects: 1) the type of

target information (i.e., information about a network node or the traffic on a communication link), 2) the access privileges of the adversary (i.e., insert traffic to the network, capture traffic from the network, and read node internal states), and 3) constraints (e.g., blocked by the firewall). To simplified the following discussion, we denote the above three kinds of access privileges as a **write** privilege to the network, a **read** privilege to the network, and a **read** privilege to the node. As illustrated in Figure 2, the five types of anonymity attacks will be briefly described followed by a comparison.

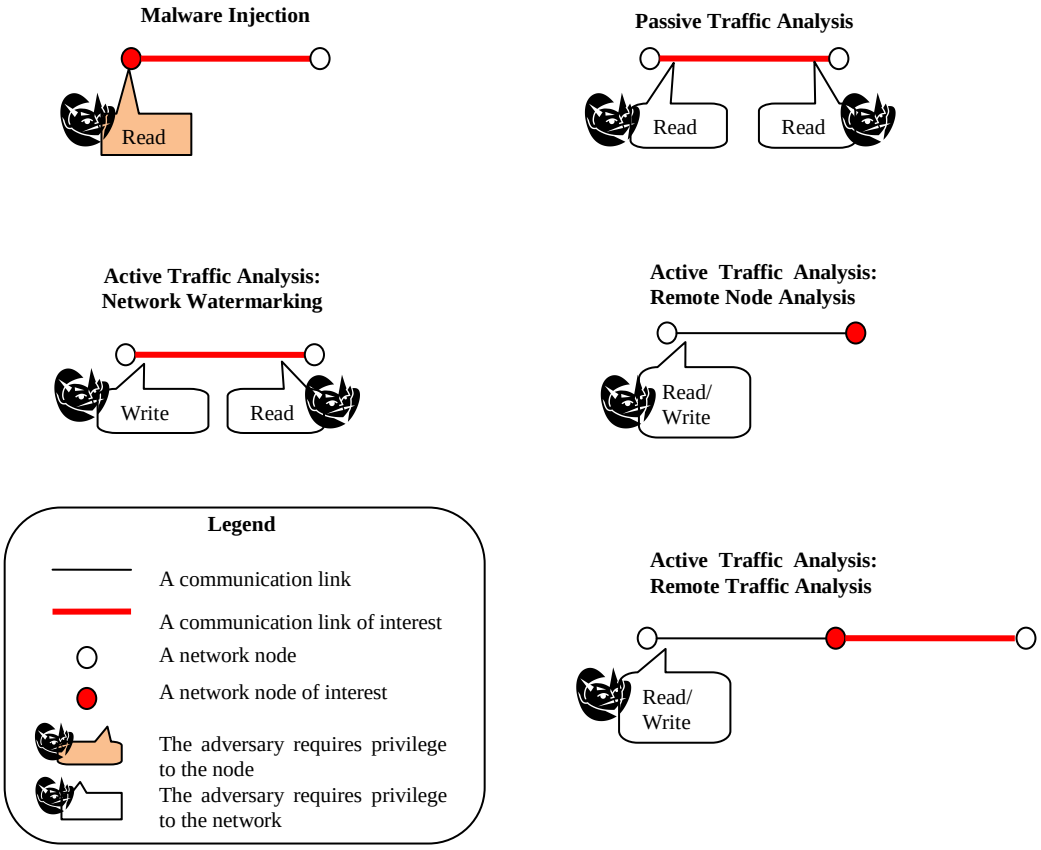


Figure 2. Adversary models of current anonymity attacks

1. *Malware Injection*. An adversary injects malware to a target network end-node so that the traffic of any communication link connected to this node will be revealed. For a communication link of interest, the adversary should have a read privilege for one end-node. But malware injection has limitations on the software protection mechanisms and scalability [47]. Modern operating systems in the network end-nodes may use a confined environment, such as a sandbox or virtual machine, to provide access control. Therefore, installed malware may not get enough system privileges (e.g., phones which are not "jail-broken"), to obtain communication records. In addition, the system administrator may apply integrity checks for the network end-nodes software such that only the authenticated and authorized software can be installed in the network nodes. On the other hand, a firewall or practically-equivalent network address translation (NAT) devices reduces the chance of remote malware injection.
2. *Passive Traffic Analysis*. This attack monitors the traffic of a network and extracts information from the (encrypted) traffic [20, 66, 70, 71, 76, 80]. Depending on the adversary model, the adversary may be able to capture traffic in every communication link as a global eavesdropper (equivalently, the adversary may capture traffic on both ends of a communication link of interest), or more realistically, the adversary may capture only the traffic in some of the communication links. In any case, the adversary needs to have read privilege on both ends of the communication link for a communication link of interest. Note that the communication links may be protected by an

anonymity system and therefore the traffic captured at one end cannot be directly associated with the traffic captured on the other end. From the captured traffic, the adversary can correlate the traffic captured in the two ends of a communication link (e.g., via packet counting, or frequency), to confirm that the traffic is actually linked. For example, Srivasta *et al*. confirm the existence of VoIP calls by correlating flows, and require a global adversary who knows the number of flows between nodes (including VoIP nodes) to attack an anonymous network [71]. The adversary can also use probabilistic models (e.g., Hidden Markov Models) to extract the content of communications from the captured traffic [76]. However, the assumption that the adversary has read privileges to both ends of the communication link is not always practical. For example, mobility brings high dynamics to the network end-node addresses and a network end-node may have multiple access channels to connect to the network (e.g., a smartphone can connect to the Internet by 3G or WiFi). These multiple channels make correlations difficult.

3. *Network Watermarking*. The adversary encodes information in the traffic at one end of a communication link and decodes signals at the other end, in order to confirm the traffic on this communication link. For a communication link of interest, the adversary should have a read privilege to one end and a write privilege to the other end. In Wang, Chen and Jajodia's watermarking attack, the adversary perturbs and monitors the timing of the VoIP flows between a VoIP node and a corresponding anonymous network end-point [74]. The

perturbed packet transmission time enables the adversary to link one VoIP call with another. In Murdoch's attack, watermarking is used to reveal the identity of a hidden server in the Tor anonymous network [53]. Through the Tor network, a node, as a hidden server, can provide a service without revealing its IP address and other nodes can access the service through a pseudonym. The adversary probes the hidden server with watermarked packets (i.e., write privilege on the network); meanwhile, the adversary probes possible relaying hosts (i.e., read privilege on the network) and if the responses from a relaying host match the watermark, this relaying host is determined to be included on the path between the adversary node and the hidden server. The network watermarking attack has the limitation that the adversary must have read privileges to one end of the communication link and write privileges to the other end.

4. *Remote Node Analysis*. An adversary probes a remote network end-node to obtain this node's type or internal state. Such information about a node can be used to degrade or break the sender or recipient anonymity by giving the adversary the ability to link a pseudonym for this node to its real identity. The adversary needs to have both read and write privileges to one end of a communication link to obtain the private information about a network end-node at the other end of this link. This attack can obtain the private information for a remote node, but cannot, at least directly, confirm the traffic of a communication link. In Murdoch's temperature channel, the adversary, at one end of the communication link, sends packets to the other end, which is a

hidden server in the Tor anonymous network [54]. The adversary then analyzes the timing of the response packets. By varying the packet rate, the target end-node will have different CPU loads that result in different clock skews because of the temperature changes. Clock skews can be calculated from the timing of response packets. Meanwhile, the adversary can measure the clock skew of a set of candidate hosts using the same probing mechanism. The adversary can further obtain the identity of the target end-node by comparing clock skews. Operating-system fingerprinting is another attack of this type [24, 57]. The adversary sends carefully-designed packets to the target network end-nodes, and different types of responses can be used to distinguish different kinds of operating systems or TCP/IP protocol stacks.

Another type of attack discovers the parameters of remote hosts. Barbuzzi, Ricciato, and Boggia conjectured that probing phones can obtain the state machine parameters of radio resource management in the 3G mobile communication system [5]. Their conjecture was then confirmed by Perala *et al*. who compared empirical results with a telecommunication traffic analyzer [60]. This technique was then systematized by Qian *et al*. [64]. Discovered parameters are used to improve the mobile application performance [3, 45], improve the simulation accuracy, or to optimize the paging attack, which is a denial of service attack [5]. In another attack, Fu *et al*. probed network end-nodes and used the response time to infer the system load for the target end-nodes [23]. The constraint of this type of attacks is that the obtained private information may not directly deanonymize the communication links.

5. *Remote Traffic Analysis*. This type of attack is a special case for remote node analysis when the adversary probes the target end-node and infers the traffic from the probing responses. Blond *et al.*'s attack concurrently probed a host through both the Skype (VoIP) and BitTorrent (P2P file sharing) protocols. The target host should have a public IP address or the open ports in its network address translation (NAT) devices [6]. When a host is behind NAT, it may share the same public IP address with others. Based on the assumption that some TCP/IP implementations still use a predictable identification field in IP headers, if the IP headers in Skype and BitTorrent probing response packets contain identification fields with close values, the Skype user is detected as the BitTorrent user. This attack is limited due to the requirements that the identification field is predictable and the fact that the NAT Port is open. The former requirement is already well-known as a security vulnerability and the latter may be disabled by those who use strict security policies.

   In another remote traffic analysis attack by Gong, Kiyavash, and Borisov, the adversary probes a DSL router with ICMP packets and uses the time series of response time as the fingerprint for the website accessed by a computer served by the router [27]. However, their approach does not deal with noise and randomness. They used Dynamic Time Warping (DTW) distance to match the observed time series and multiple time series of different websites which were recorded previously. Although DTW can identify shifted and stretched signals, it cannot deal with stochastic signals. In other words, two time series

that are the realization of the same random process are regarded as different, even though they have the same probabilistic properties. Furthermore, if two computers access the Internet using the same DSL router, the pattern in time series data is likely to be heavily distorted and not recognizable. Moreover, ICMP blocking is a standard function in modern network devices and therefore such attacks can be avoided. Another constraint of this attack is that when two websites have the same communication content, the adversary is unable to distinguish which one is actually involved in the communication. Therefore, this approach cannot be extended to VoIP, whose traffic is alike.

These above attacks, except the malware injection, are either side-channel attacks or covert-channel attacks. The passive traffic analysis is a side-channel attack because the adversary only gets information from a position outside the system and there is no insider adversary. The network watermarking is a covert-channel attack because an adversary needs to encode and hide information into the traffic and the other adversary needs to decode this hidden information. The remote node analysis and remote traffic analysis are side-channel attacks because an adversary needs no inside help to obtain the inside information.

By comparing these attacks, we can find that remote traffic analysis, while still exhibiting some limitations, is a promising attack for breaching the anonymity of communication records because it analyzes the traffic "remotely". Malware injection can be defeated by software integrity checks. Passive traffic analysis and watermarking can both confirm the traffic of a communication link, but the adversary

needs to have read or write privileges to both ends of the communication link of interest (i.e., not remote attacks). Remote node analysis can reveal the node identity but cannot confirm the traffic on communication links. Remote traffic analysis enables the adversary to obtain the traffic information where the adversary does not have direct read or write access to the traffic or the corresponding nodes. However, the remote traffic analysis schemes cannot go beyond a firewall/NAT and has a problem in identifying the communication pattern when two communication links associated with the same end-node have the same type of content. In contrast, our approach is to combine remote traffic analysis and passive traffic analysis so that the adversary can obtain the "remote" read privileges to the both ends of the communication link of interest.

1.5. The Private Communication Detection (PCD) Problem

Based on the analysis of extant research and the use of covert channels and side channels, we propose the private communication detection (PCD), a communication anonymity attack whose adversary has the same capabilities as the remote traffic analysis but can perform powerful traffic correlation among all communication links in the same way as a passive global eavesdropper. In the remote traffic analysis' attack model, the adversary does not need to have any special capability above an ordinary network user (i.e., although the adversary needs to have read and write privileges on the network, the adversary does not need to have *direct* access privileges to the communication link of interest). However, since the remote traffic analysis only indirectly observes one end of the communication link of interest, if two

communication links associated with the same target end-node may transmit the same content (e.g., VoIP calls have the same call-setup procedures and constant-rate streaming), the adversary is unable to identify which communication link actually has traffic. If the adversary can observe both ends of the communication link of interest, via the passive traffic analysis, she will overcome the above problem because she can correlate the traffic on both ends of the communication link of interest. In short, this correlation enables an adversary to detect private communication between the two ends of a communication link, and hence to breach communication privacy without any special wiretapping capabilities.

1.5.1. The Problem

We first consider the simplest form of Private Communication Detection (PCD), which contains only two targets and one adversary. PCD with more than two targets will be discussed in Section 1.6. As illustrated in Figure 3, the PCD adversary model composes an adversary, goal, and capabilities.

Adversary Goal: Can adversary Eve obtain the communication pattern (e.g., call time, length, and frequency) of two target users, Alice and Bob?

Adversary Capabilities: Eve is an ordinary network user without special capabilities, such as malware injection, control of infrastructure, or eavesdropping. Eve can only send packets to Alice and Bob, and receive timely responses; i.e., via probing.

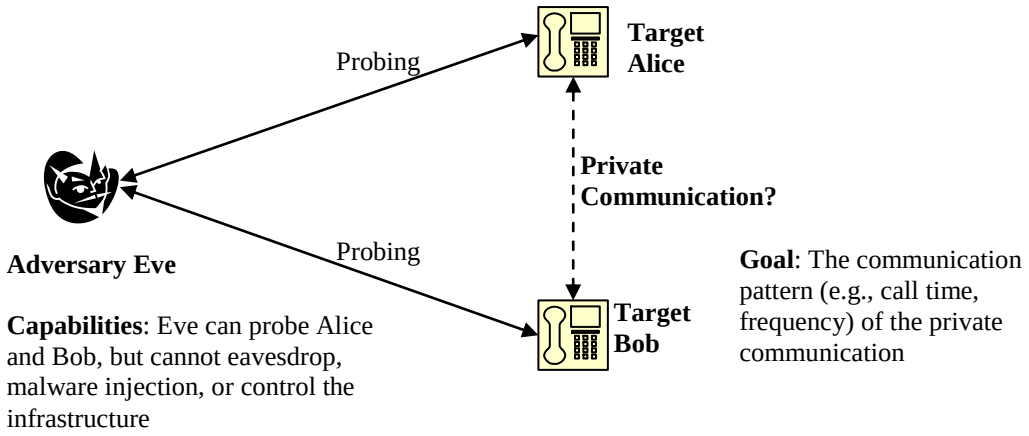


Figure 3. The definition of PCD adversary

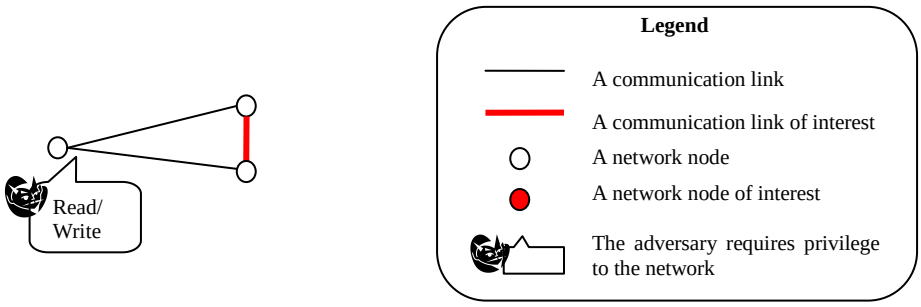


Figure 4. The simplified PCD adversary model on the read/write privilege

In this setting, the private communications between Alice and Bob can be based on an anonymity system, if such a system is able to satisfy the requirements for this communication (e.g., VoIP requires low latency low jitter). The inaccessibility (e.g., physically-secured) of traffic to adversary Eve makes the communication between Alice and Bob private to Eve. Therefore, we assume the communications between Alice and Bob are a type of "private communication", which is a more general case than the anonymity system case.

In order to make a more explicit comparison with the other anonymity attacks defined in Figure 2, the PCD problem is illustrated in a similar fashion in Figure 4. The major

difference between PCD and remote traffic analysis (e.g., Gong *et al.*'s remote traffic analysis) is that the PCD adversary infers the private communication between two target network end-nodes. It gives PCD a network-wide view. For example, in Gong *et al.*'s remote traffic analysis, the node to which the target end-node connects is actually inferred by the fingerprint of the communication contents. In that case, the target end-node may connect one of two nodes with the same contents. Hence the adversary is unable to distinguish which node is actually connected by the target end-node. We illustrate the relationship between PCD and other anonymity attacks in Figure 5. This figure illustrates the role of PCD in active remote traffic analysis, along with remote node analysis, network watermarking, and remote traffic analysis.

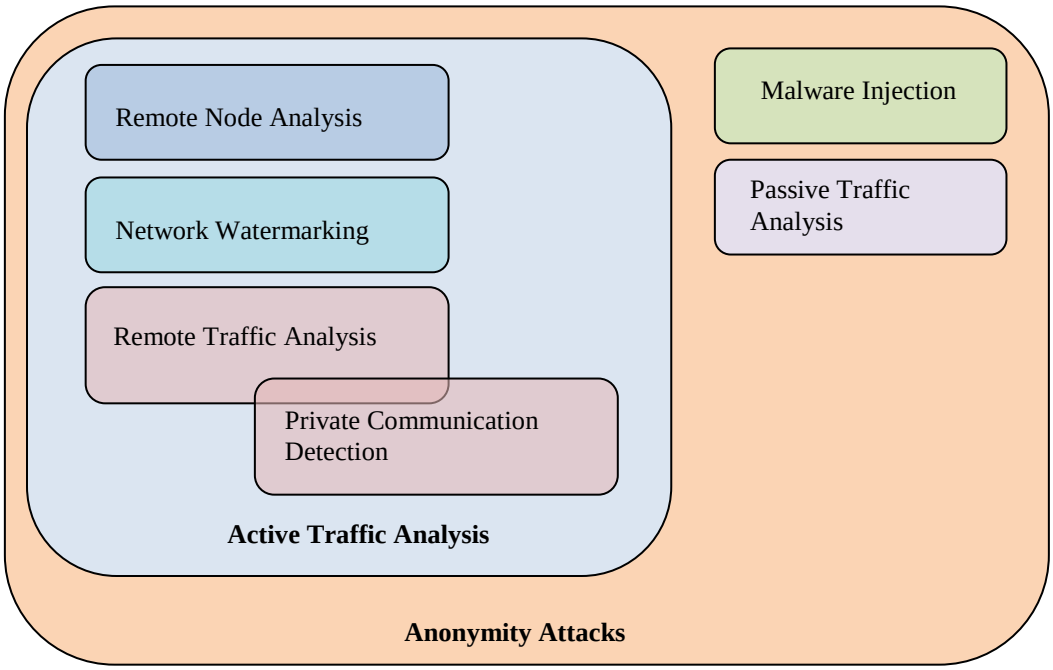


Figure 5. Relationship of PCD and other anonymity attacks

PCD has two unique properties as follows.

1. PCD can monitor any link in the network without direct access. Other attacks, including passive traffic analysis and network watermarking attacks, require the adversary to have read or write privileges on at least one end of the communication link of interest.
2. The information that is of interest in PCD, the communication pattern between two end-nodes, implies that PCD needs to have a high time resolution. In other words, PCD concerns the short-term relationships (e.g. a voice call may last only tens of seconds) between targets, but not the long-term relationships (e.g., friendship). However, the knowledge of short-term communications can allow us to infer the long-term relationship as well. Current privacy attacks in anonymous networks cannot, and are not intended to be used for call detection since they require longer data acquisition times than an entire call, which may only last for seconds or minutes [18, 53, 54]. By using the obtained communication patterns of short-term communications, an adversary Eve can also perform link analysis (Viz., Figure 3) and determine the strength of ties between Alice and Bob. The tie strength can be accurately measured by interaction frequency, evidence of recent communication, communication reciprocity, and the existence of at least one mutual friend in linking the two target parties [25].

Whenever PCD is achievable, it enables an ordinary network user without special abilities to obtain private information from remote hosts (or equivalently, an ordinary

user of a perfectly anonymous network [18, 61, 62]). For example, a foreign agent might want to collect call records of secure (e.g., malware-free) smartphones of targeted US government officials or a spouse may want to discover if his/her significant other is developing an undesirably strong relationship (tie) with a third party. The question of whether ordinary users can collect call records gains further relevance in anticipating future secure networks, which will undoubtedly provide anonymity despite wiretapping, as well as secure communication devices, which rule out the remote insertion of malware that could operate in an undetectable manner.

1.5.2. A New Type of Side Channels and Relationship Inference

The achievability of PCD relies on two techniques, namely a new type of side channels and the ability to infer relationships. In the NSA's Rainbow series (i.e., the series of security guidelines for trusted computing systems of U.S. Department of Defense), Gligor categorized a class of covert channels, including the abovementioned quantum-time channel, that exploit the shared resources [26]. Once an adversary can use certain shared resource (e.g., CPU, bus, and disk) and another adversary can observe the resource use, there exists a convert channel between the two adversaries.

Although the covert channels caused by a shared resource have been extensively studied, only few side channels caused by a shared resource have been discussed in

the literature[1] outside cryptography. Surprisingly, we found that the side channels caused by a shared resource are the key to achieving PCD.

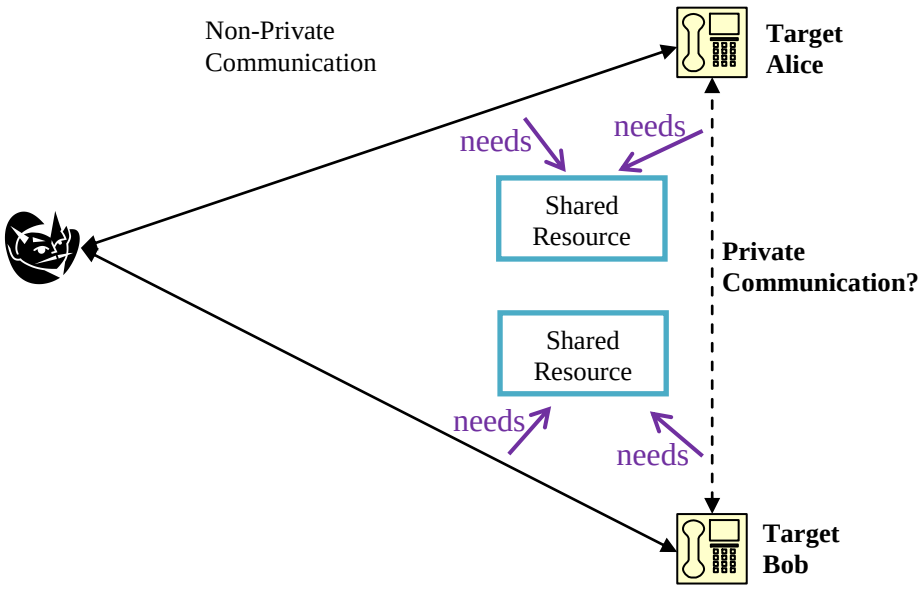


Figure 6. Resource-contention side channels in PCD

As shown in Figure 6, if certain (exclusive) shared resources are required for communication, the resources will both affect and be affected by the communication channel. Resource contention associates the private communication (between Alice and Bob) and the non-private communication (between Eve and Alice, or Eve and Bob). One example of a resource-contention side channel is provided when the radio channel is the shared resource under contention in a WiFi network. When a WiFi

[1] It is easy to find a side channel in daily life. For example, imagine an apartment where a bathroom with two doors is located between the two bedrooms. In this case, a tenant living in one bedroom can indirectly observe the behavior of the other tenant with ease, because they both share the bathroom. Use of the shared resource, in this case, the light under the door or sound of the shower, provide easily accessible clues to the other occupant that the bathroom is in use.

node intends to send a packet, it must use the radio channel according to the CSMA/CA protocol, such that if another node is transmitting or expected to transmit, the sending node must wait and will retry access (back-off) after a random amount of time. Therefore, the timing of one WiFi node's traffic is influenced by another WiFi node's traffic.

The side channels caused by resource contention allow the PCD to distinguish whether there is traffic on a communication link. As shown in Figure 7, given the probing responses, the PCD adversary acts as a *distinguisher* that identifies whether there is traffic on a remote communication link at the time of probing.

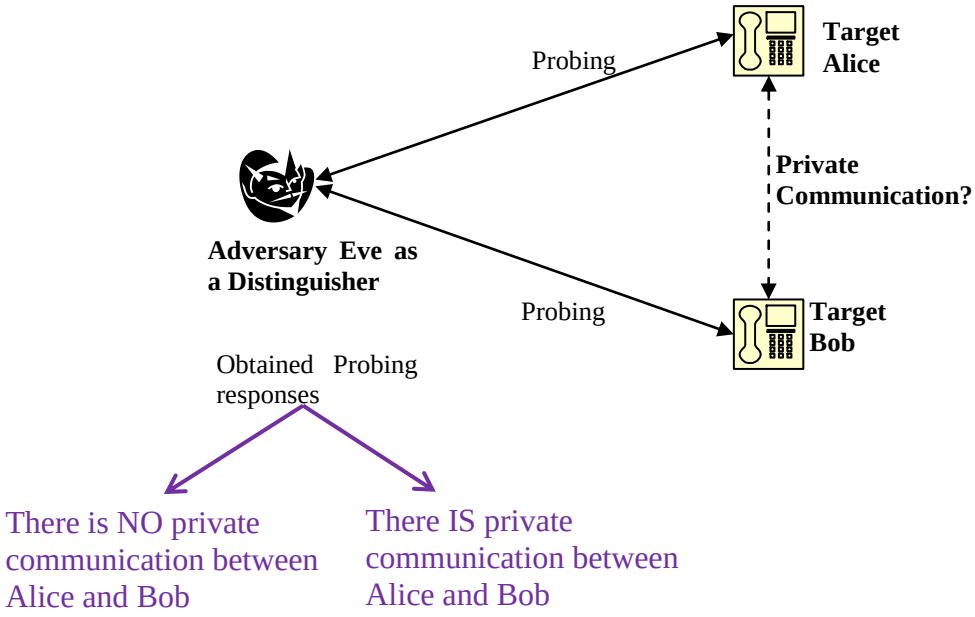


Figure 7. The PCD adversary is a distinguisher

Due to the fact that a PCD adversary can probe (via read privileges on the network) the two ends of the private communication, PCD can actually infer the relationship between two end-nodes from the behavior of these two end-nodes. This idea is similar

to Resig *et al.*'s work, where the "friendship" is inferred by the online/offline status of Instant Messaging users [68]. They hypothesized that if two users go online and go offline at about the same time, the two users may be friends. The difference between PCD and their work is that 1) in their scheme the user status (online/offline) is explicitly provided by the users, while in PCD the adversary must detect the status of end-nodes (which will be discussed in the next section), and 2) the "friendship" relationship in their work is long-term while the call-records under study here are short-term.

1.6. Overview of the Proposed PCD Approaches

We propose three PCD approaches using various probing techniques that exploit resource contention. The Resource-Saturation PCD exploits the resource contention inside network end-nodes; the Stochastic PCD utilizes the resource contention in the network; the Service-Priority PCD makes use of resource contention in different services. The three approaches are briefly described below and then compared in Table 1.

Resource-Saturation PCD Approach (Chapter 2). In some VoIP phones, especially for those implemented with dedicated hardware (rather than software), a fixed-size buffer (the resource under contention) is used to store the context for VoIP protocol negotiation. By sending carefully designed packets[2] to the phones, the adversary is able to determine whether a phone is busy because certain exceptions (e.g., no or

[2] The probe packets should ensure that the phones behave as usual (e.g., no alerting).

different responses) are signaled when the buffer is full. The adversary can then use the following principle to infer the call records: When two phones become busy during the same period, it is likely[3] that they could communicate during that period. We analyzed this approach successfully on Cisco, LinkSys, and Grandstream VoIP phones, despite the fact that *source code is unavailable*. However, this approach requires an extensive, time-consuming protocol analysis and applying it to a new protocol/device may require redesigning the probe packets.

Stochastic PCD Approach (Chapter 3). The shortcomings of Resource-Saturation PCD are overcome by the Stochastic PCD approach, which periodically probes the targets and classifies the time series of the response times using stochastic models. While stochastic models deal with the randomness in the random-access networks (e.g., a WiFi node's back-off time is a random variable), the challenge lies in removing the noise (network jitter) introduced by the Internet switching, application server, and other nodes in the same network. By reducing the noise through signal processing, the performance in terms of detection accuracy is improved significantly. We employed this approach in two applications: Instant Messaging and WiFi. The shared resources exploited are the keyboard and radio channel, respectively. The Stochastic PCD can also be used for the practical detection of Sybil attacks [46, 55].

Service-Priority PCD Approach (Chapter 4). The last and most powerful PCD attack is performed against smartphones. Mobile communication systems, such as the 3G

[3] False detection will be discussed in Section Appendix I.

UMTS and 2G GSM/GPRS, provide voice and data services through circuit and packet switching, respectively. Due to the limited radio resource (the resource under contention), voice service is given a higher priority than data service. Therefore, the adversary can repeatedly probe a phone and detect whether it is busy according to the delayed or lost responses. The adversary can then correlate the busy status to infer the call records in the same way as the Resource-Saturation PCD. We employed this attack in AT&T's 3G and 2G networks. In addition, sending probes to smartphones can not only surreptitiously discover the call records but also can perform a denial of service attack on 2G smartphones. Probing 2G phones with short intervals (e.g., 1sec) will block the paging channel and the incoming/outgoing service becomes inactive, without providing an explicit error that the user can detect.

Our proposed PCD approaches not only achieve anonymity breach in four separate applications (WiFi, Instant Messaging, VoIP, smartphones) but also suggest other candidate applications for PCD. Resource-Saturation PCD is suitable for communication systems where each network end-node (e.g., phone) uses a certain limited-quantity resource (e.g., fixed-size buffer, semaphores, or threads) to establish a communication session. The used resource is released upon the end of the communication session. Stochastic PCD works most effectively when the chosen stochastic models can accurately model random-access networks (e.g., Markov Chains can model CSMA/CA networks) and when signal-processing techniques can eliminate much of the noise in the probe responses. The Service-Priority PCD will also work on a communication system that provides two or more kinds of services whose priorities are not equal.

Table 1. Summary of three proposed PCD approaches

		Resource-Saturation PCD	Stochastic PCD	Service-Priority PCD
Evaluated application	Type of service(s)	VoIP	WiFi; Instant Messaging (Two applications)	Smartphones
	Resource under contention	A fixed-size buffer in the VoIP phone	Radio channel; Keyboard	Radio channel
	Time resolution	High (three phones have 2, 3, and 34 seconds minimum intervals, respectively)	Low (1000 seconds intervals for WiFi ; 300 seconds intervals for Instant Messaging)*	High (1 second intervals or less)
	Methodology	Sending special probe messages to saturate the remote fixed-size buffer	Classifying response time series by stochastic models and signal processing techniques	Exploiting the fact that voice service has higher priority than data service
Candidate applications		Network end-nodes need certain resource with limited quantity (e.g., semaphore; thread) for a communication session	The chosen stochastic models can accurately model the random-access networks and the signal-processing techniques can eliminate much of the noise	The communication system has two or more types of service whose service priorities are not equal

*: These numbers are simply the data collecting interval for samples. The intervals can be lower (i.e., higher time resolution)

The *time resolution*, namely how often the adversary can obtain the busy status or the communication pattern, is an important parameter for PCD. The higher the time resolution, the more accurate the communication pattern readings. The Resource-Saturation PCD and Service-Priority PCD can achieve a very fine-grained time resolution (1-2 seconds). In that case, the number of probe messages required to successfully obtain the busy status in both PCD approaches will range between 1 and 7. The time resolution of the Stochastic PCD is much lower because more responses are required by the stochastic models. Another important feature of the time

resolution is that the higher the time resolution, the lower the false detection rate, as shown in the next section.

1.7. PCD with More Targets

The two-target PCD can easily be extended to monitor more targets. This can be implemented by performing two-target PCD on each pair of the monitored targets. The downside of this increased coverage is that one requires a greater awareness of the false detection problem. When Alice and Bob communicate with third parties, false detection occurs if the adversary falsely identifies that there was a call between Alice and Bob when they actually communicated with third parties. Fortunately, false detection rates are insignificant in the Resource-Saturation PCD and Service-Priority PCD approaches; the Stochastic PCD approach has a special case solution.

The rate of false detection due to third-party communications in all three approaches depends on the PCD time resolution. In Resource-Saturation PCD and Service-Priority PCD, if Alice and Bob both have a single call during a one-hour period and the time resolution is 10 seconds, the false detection rate is 1.8% when the start time and the end time of the calls are uniform-distributed random variables (viz., Appendix I for details). The false-detection rate drops to 0.24% when the time resolution is 1 second. For all practical purposes these false-detection rates are insignificant for call-record collection. This is the case not only because these rates are extremely small but also because most call-record analyses require multiple instances of detected calls; e.g., strength-of-tie analysis requires call frequency and reciprocal initiation, and timing patterns. The more instances of detected calls in an analysis, the greater the

statistical confidence in the results obtained.

Stochastic PCD approach typically needs a much longer probing interval than the other two approaches. Fortunately, the Stochastic PCD approach has a solution for the special case when the type of traffic between the targets and the third parties is different from that between the targets. This would be the case if, for example, the communication types among all the targets and the third party included web surfing, video streaming and SSH, whereas the traffic between the targets is VoIP. In this case, signal-processing techniques can help remove the effects of non-VoIP communications in performing the Stochastic PCD.

1.8. Contributions

The main contribution of this dissertation is the definition and realization of a new, powerful attack against call-record privacy, namely private communication detection (PCD). Not only can PCD be launched with only ordinary-user capabilities, but it succeeds with very high probability. Through PCD, an ordinary network user can obtain the communication pattern (e.g., call time, length, and frequency) without any of the typically-assumed special abilities, such as malware injection, control of the infrastructure, or eavesdropping. This new privacy attack is enabled by a new type of side channels caused by resource contention. The three approaches and the four applications illustrated herein show the generality of PCD and the resource-contention side channels. Among various countermeasures, we describe a general defense against PCD toward active probing by monitoring and blocking periodic suspicious network traffic.

The Resource-Saturation PCD approach not only indicates that a fixed-size buffer in a VoIP phone can result in private information leakage, but also implies that other shared resources, such as operating system resources (e.g., semaphores, number of threads) may enable privacy breaches in communication systems. Moreover, we tested our attack in closed-source commercial VoIP phones, which do not provide any implementation documentation, and the attack success here provides a fact that provides further testimony for the practicality of the attacks.

In Stochastic PCD, we show that PCD is enabled by timing side channels caused by different types of resource contention in end nodes. Exploitation of these timing channels from PCD is effective despite network and proxy-generated noise (e.g., jitter, delays). The Stochastic PCD approach also suggests that the traditional countermeasure to side-channel attacks, adding noise, would not be effective when the adversary is able to eliminate this added noise through well-developed, existing signal-processing techniques. We use a stochastic analysis to demonstrate how PCD exploits indirectly accessible end-node resources, such as WiFi radio channels and computer keyboards in Instant Messaging. Similar analysis enables practical Sybil node detection.

The Service-Priority PCD is *adversary-friendly*: simple application programs can perform automatic probing and relatively unskilled users can visually recognize the pattern of probe responses returned when a smartphone is engaged in a call. The attack is illustrated in a real environment where an iPhone and a Google Nexus One Android phone are connected to AT&T's 3G/2G network. In addition, we illustrate a

denial of service attack on phones that use a 2G network, where frequent probing is able to prevent the voice service signals (e.g., paging) from being delivered. Without any explicit error a user would recognize as an attack, outgoing calls fail to connect and incoming calls will fail to ring through to the user's device.

1.9. Dissertation Organization

In the rest of this dissertation, we describe the PCD context, methodology, and results. We illustrate the Resource-Saturation PCD approach in Chapter 2, Stochastic PCD approach in Chapter 3, and Service-Priority PCD approach in Chapter 4. Each PCD approach is applied to one or two applications, including the protocol analysis and the evaluation. Chapter 5 discusses the countermeasures to PCD. Finally, the conclusion s and future directions are given in Chapter 6.

Chapter 2: Resource-Saturation PCD

In this chapter, we propose the Resource-Saturation PCD approach that performs a PCD attack against VoIP phones by exploiting the fixed-size buffer that stores the protocol negotiation context. This approach was developed by analyzing both the protocol specification and implementation. The main challenge is to perform the attack without alerting the users, because the probe messages that can be used to saturate the fixed-size buffer may also be the messages used to set up calls. The work in this chapter was previously reported in [35].

2.1. Overview

Problem Statement. In Figure 8, attack targets Alice and Bob establish a VoIP session. The session can be delivered over one or more forwarding proxies to provide anonymity, if its latency satisfies the needs of VoIP (i.e., <400ms latency recommended by ITU-T G.114; longer latency tolerance for tactical environments) [17, 20, 34, 39]. The forwarding proxies can be implemented in the network layer, transport layer, or application layer. Adversary Eve's goal is to discover whether Alice and Bob call each other using the VoIP network, the duration of the conversation, and possibly the call initiator.

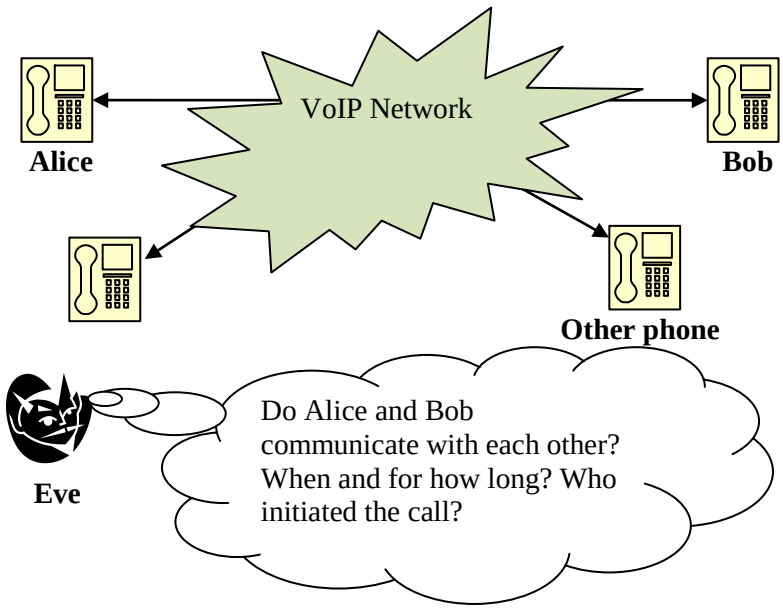


Figure 8. Revealing call records in a private VoIP network

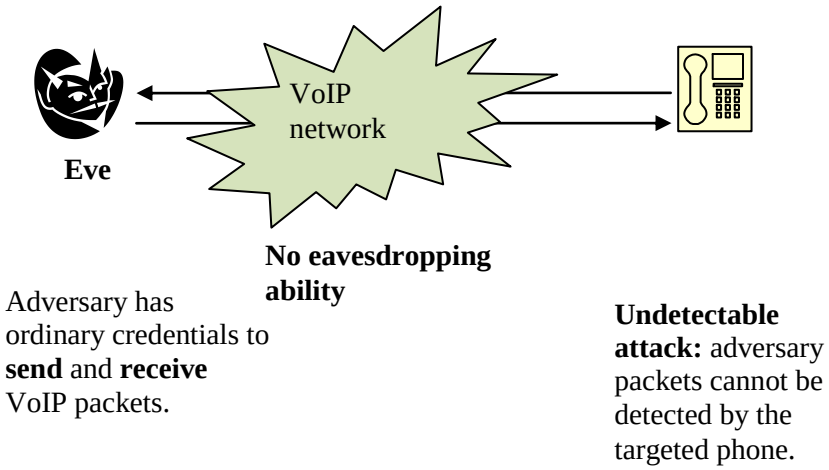


Figure 9. Adversary capabilities

Our adversary Eve does not need any added capabilities beyond those of an ordinary user. As shown in Figure 9, she can send probe VoIP packets to the target phones and receive response VoIP packets in return, which does not require any eavesdropping

capabilities. Yet her probing is undetectable[4] by the end-users of the target phones unless the phone logs or network traffic are analyzed; i.e., her packets alert neither Alice nor Bob that their phones are being probed remotely.

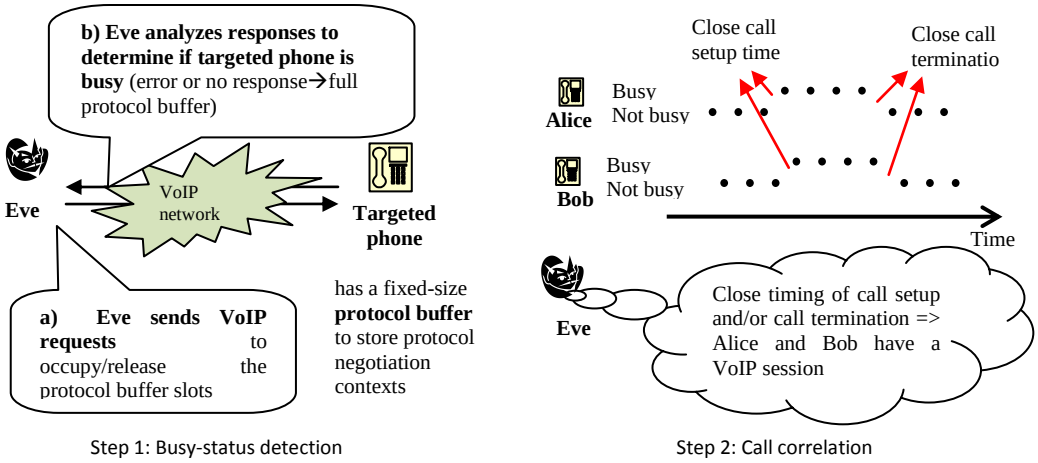


Figure 10. Attack steps

Our Approach. The simplest attack consists of the two steps illustrated in Figure 10. In the first step, Eve detects whether both Alice's phone and Bob's are busy; i.e., she performs *busy-status detection* for both phones. In the second step, Eve verifies whether Alice's phone and Bob's are busy, or not busy, at around the same time; i.e., she performs *call correlation*. If the phones' busy statuses correlate, then Alice and Bob probably have a VoIP call during Eve's probing period. Note that the busy-status detection should be undetectable (not alerting the phone end-users), otherwise the end-users of the target phones can easily recognize the anomaly and stop using these phones.

[4] We assume the unsuspecting end-user of a target phone accesses the phone via the handset, hook, buttons, ringer, and a display.

Resource contention makes it possible to detect a phone's busy status in a VoIP network. The resource, named the *protocol buffer*, is used to store the contexts of the VoIP protocol negotiation. Certain VoIP packets (e.g., call-setup packets) create a state machine in both the caller and callee phones. The context of the state machine, including the status and timer, is stored in a slot of the protocol buffer. When the desired action is completed or times out, the buffer slot occupied by the call is released. Call-termination packets can expedite buffer slot release if the buffer slots are occupied due to call-setup packets.

In some VoIP phones, including those implemented with dedicated hardware, the protocol buffer is a fixed-size array. As expected, fixed-size protocol buffers can cause resource contention. For example, in a protocol buffer containing **N** slots, the call setup occupies 1 slot and the remaining **N**-1 slots are available for other protocol instantiations. This implies that if the adversary can detect the number of available protocol buffer slots, she can determine a phone's busy status. She can do this by periodically sending VoIP packets, which would overflow all available protocol buffer slots of the target phone, and detecting whether a full-buffer condition is signaled back. By examining the response to a full-buffer condition, the adversary can count the number of available protocol buffer slots in a target phone. This enables the adversary to detect the phone's busy status in a matter of seconds – much faster than with any of the current eavesdropping/flow-analysis methods. Rapid busy-status detection makes our attack feasible in any VoIP networks. For our privacy attack experiments, we selected the IETF Session Initiation Protocol (SIP), which has been widely adopted by the telecommunication industry, the military, cable operators, and consumers at large.

2.2. The Attack

In this section, we provide an overview of SIP with an emphasis on the mechanisms related to SIP resource-contention side channels. Then we show how busy-status detection is enabled by four side channels that we found by testing closed-source commercial VoIP phones. (The reader who is familiar with the IETF SIP specification can skip Section 2.2.1. below.)

2.2.1. Call Setup and Termination Overview

SIP is a HTTP-like application-layer protocol designed for VoIP signaling and other applications that require devices to setup/terminate sessions to exchange information Figure 11. SIP uses a *transaction* as its basic message exchange component. A transaction is composed of a *request* message, optional *provisional response* messages, and a *final response* message. In this chapter we use the terms transaction and request interchangeably in the case where the request represents the transaction itself. The communication initiated by the caller to a callee's phone can be relayed by one or more SIP proxy servers. Relaying, which is typically used for billing, redirection and many other telephony functions, does not affect our attacks.

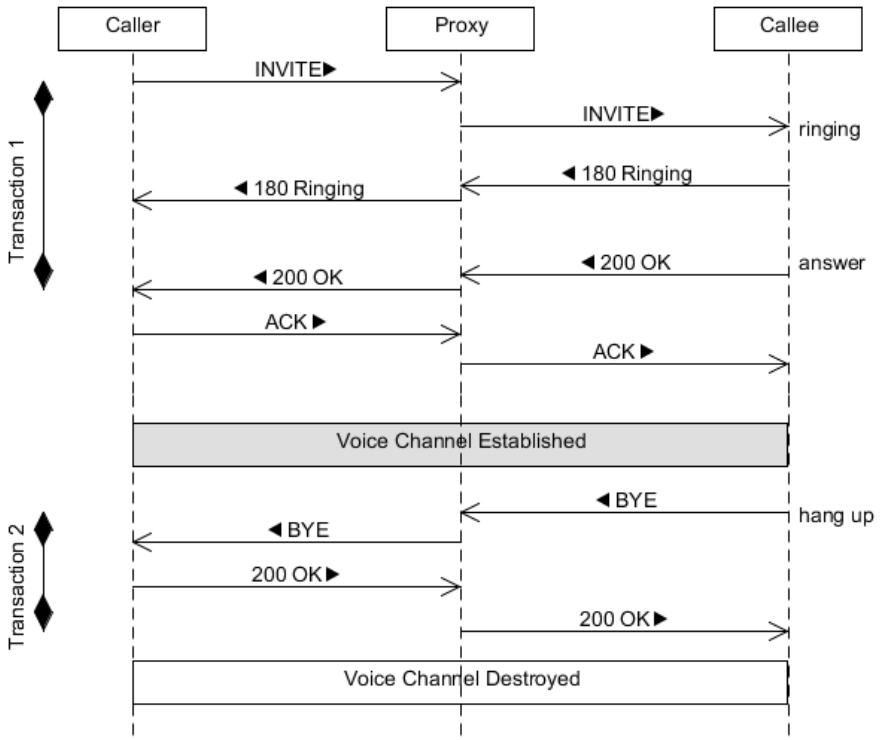


Figure 11. SIP call setup and termination

In Figure 11, we illustrate a call setup (Transaction 1) and termination (Transaction 2) in SIP. In Transaction 1, the caller first sends an INVITE request to the callee. A SIP request message includes three parts. The first part contains the method name, such as INVITE, to describe the main purpose of this request. The second part contains headers, which specify attributes (e.g. To: sip:wj@iptel.org). The third, and optional, part is the payload, which usually includes media parameters encoded by the Session Description Protocol (SDP). When the INVITE request is received, the callee's phone rings, and immediately sends a provisional response, such as "180 RINGING," to the caller. (SIP responses are identified by a 3-digit number.) Responses with a hundreds digit of value 1 are called the "provisional responses" (denoted 1xx in SIP specification) and provide status information to the caller in the middle of the

transaction. In Transaction 1, the callee decides to answer this call by picking up the handset, so the callee's phone sends back "200 OK" in response to the original INVITE request. "200 OK" is among the "final responses", which are specified by codes with digits that do not begin with a 1 (i.e., 200-699 in SIP specification).

After the caller sends an ACK request, the caller and the callee have already reached an agreement on the media parameters for establishing a voice channel by a streaming protocol such as RTP. Eventually one of the two parties (in this case, the callee) will terminates the call by initiating Transaction 2 with a BYE request. The other party replies using a "200 OK", which terminates the voice channel.

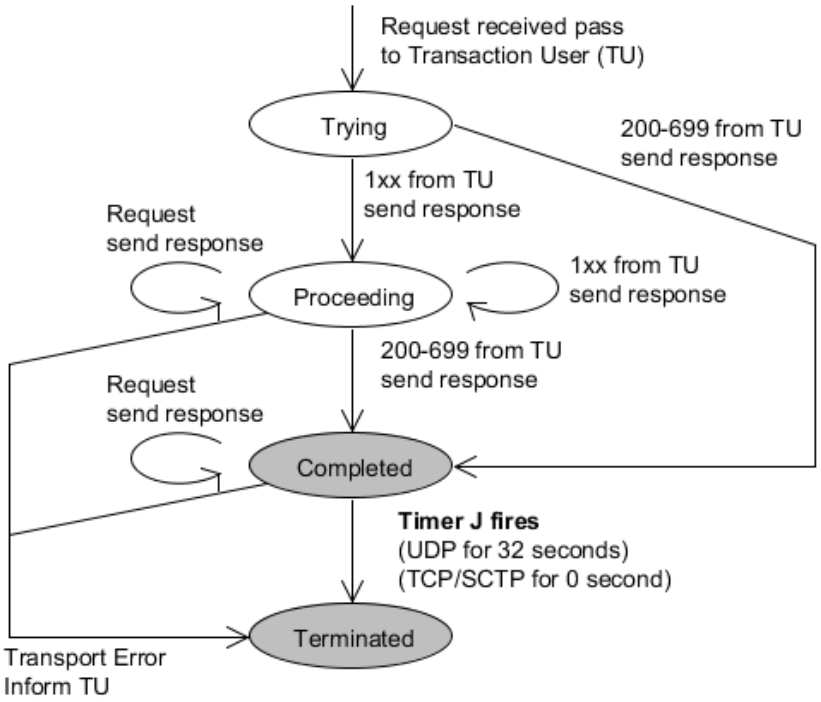


Figure 12. A non-INVITE transaction at the server side

SIP defines the roles of server and client. When a SIP protocol side receives SIP requests and sends SIP responses, it acts as a server; when it sends SIP requests and

receives SIP responses, it acts as a client. Four state machines are defined to describe the general behavior of INVITE and non-INVITE transactions for SIP clients and server roles. Method-specific behavior is based on the four state machines. In Figure 12, we illustrate one of the four state machines for a non-INVITE transaction on the server side. When the SIP phone (a Transaction User) receives a non-INVITE transaction, such as BYE or OPTIONS, it enters the *Trying* state and creates a transaction instance. After sending the provisional responses (1xx), it enters the *Proceeding* state. If the SIP phone (i.e., user) decides to send out a final response (200-699), it enters the *Completed* state. When Timer J signals a time-limit exceeded event or there is a transport layer error, the state machine ends up in the *Terminated* state and the transaction expires. By default, Timer J is set to 32 seconds for UDP and 0 seconds for TCP/SCTP.[5] Timer J also helps SIP to handle packets lost in UDP via retransmission.

2.2.2. Busy-status Detection

Side Channel. Like other stateful protocols, SIP needs a protocol buffer for storing the protocol negotiation context for each transaction. In some SIP phones, especially for those implemented through dedicated hardware (i.e., hardware SIP phones), the protocol buffer is implemented as a fixed-size array for two reasons. First, SIP phones generally do not need a large protocol buffer since they are not expected to receive dozens of calls per second. Second, if the protocol buffer is a simple fixed-size array,

[5] SIP standards require both UDP and TCP to be implemented, but UDP is generally more popular than the TCP because TCP has a longer call-setup time due to a three-way hand-shake [21].

the SIP phone can recover from buffer flooding automatically, after the flooding packets are gone. In contrast, if the protocol buffer is allocated on demand, recovery is more complex, as the SIP phone may enter unrecoverable states (e.g., kernel panic) due to unreleased, used-up memory. The disadvantage is that, this fixed-size array enables an adversary to count the number of available slots in the protocol buffer and tell whether a VoIP session exists; i.e., the adversary is able to tell whether a buffer slot is used and the phone is busy. The adversary exploits this side channel to detect calls surreptitiously.

Detection Algorithm. Let **N** be the size of the protocol buffer. To count the number of available buffer slots, the adversary sends **N**+1 SIP request (i.e., probes) to the target phone, one every *d* time units. *d* is chosen to be small enough so that no request could expire and free up a slot before the last request is received. In some SIP implementations, when the buffer becomes full, the phone ignores the next request, whereas in others it returns an error. Suppose the protocol buffer size has *h* available slots before the adversary begins sending requests. If the phone ignores the request arriving after the protocol buffer is full, the adversary will not receive the h+1st response, as shown in Figure 13 (a). In the other case, the adversary will receive an error response for the h+1st request, such as "486 Busy Here," as shown in Figure 13 (b). In either case, the adversary discovers the value of *h*.

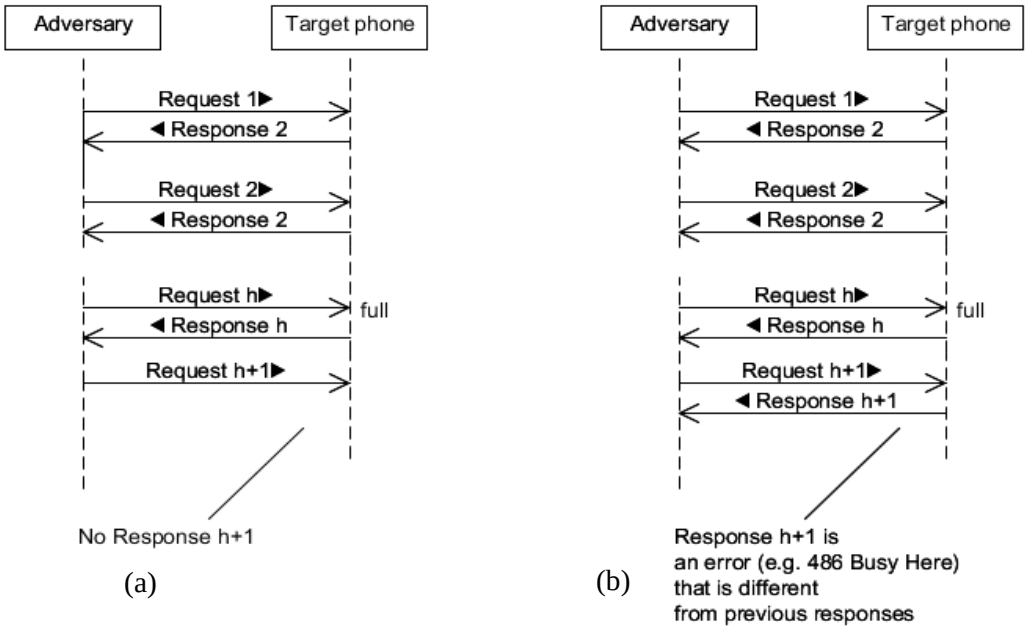


Figure 13. Possible target-phone responses when the protocol buffer is full

Detection Side Effects. If the SIP phone, or proxy in the transmission path, supports retransmission, the adversary will get a delayed positive response to the h+1st request after the protocol buffer slot becomes free. However, the delay is several seconds or less, and cannot affect busy-status detection.

Another side effect arises because whenever the protocol buffer becomes full, the phone is disabled for a period of time, which we call the *disabled period*. For example, our experiments with two phones show that the disabled period is close to 30 seconds (with parameters d=80ms, **N**=6 for 7940G, **N**=32 for PAP2). The details of the calculation are shown in Appendix III. However, during the disabled period, the target phone does not answer a caller's setup request. Only after the disabled period ends the target phone can receive a call setup request issued by the retransmission mechanism of the caller phone or proxy. The disabled period causes

the target-phone user to perceive a small ring delay. However, as described later, it is possible to shorten the disabled period in some phones such that the target-phone user will experience almost no ring delay.

Target-Undetectable Probe Requests. Since we used closed-source hardware phones, we had to find SIP methods and parameters that could be used as probes for busy-status detection, experimentally. Suitable SIP requests (i.e., probes) must fill the protocol buffer and yet must *not* alert the targeted user (e.g., by phone rings) that an attack is in progress. We experimented with all SIP methods on the closed-source4 phones and found that OPTIONS, INVITE, NOTIFY, and UPDATE methods are suitable for implementing adversary probes. In describing the suitable SIP requests, we use the naming convention "METHOD-type," such as "INVITE-require" or "OPTIONS-ordinary" below. These SIP requests are discussed below. Examples of actual SIP-request messages are given in Appendix II.

1) INVITE-require and INVITE-SDP requests. As described in Section 2.1, the INVITE transaction performs the call setup. By default, upon receiving an ordinary INVITE request, the SIP phone alerts (rings) the user. If this were always the case, the INVITE transaction could not be used to perform busy-status detection. However, an INVITE request that is invalid for making a call will not ring, and hence could be used for busy-status detection. We found two types of INVITE requests with this capacity, namely INVITE-require and INVITE-SDP.

We use an INVITE-require request with Require header whose functionality is not supported by the callee [31]. For example, the header "Require: 100rel" in an

INVITE request requires the callee to support an extension named 100rel. If the callee does not support the requested function, the callee responds immediately without alerting the user, but a protocol-buffer slot still remains allocated in the callee's phone. Hence, an INVITE request with a header "Require:xx" is suitable for busy-status detection. Similarly, the INVITE-SDP request can also attach an invalid SDP message in the INVITE request (e.g., an incomplete/invalid IP address in SDP). The INVITE with SDP is a commonly used request since a call setup needs to exchange media parameters through several SDP messages attached in the call-setup messages [32].

Disabled-Period Shortening. For busy-status detection using the INVITE method, there is a way to shorten the disabled period dramatically; i.e., to less than 2 seconds. To do this we use ACK requests, which terminate the INVITE transactions. Call setup is a three-way handshake, including an INVITE request (caller to callee), INVITE response (callee to caller), and ACK request (caller to callee), as shown in Figure 4. Since an INVITE request occupies a protocol buffer slot, an ACK request would release that buffer slot. Hence, as soon as the adversary receives all INVITE responses from the callee, it can send the ACK requests to force the freeing of the occupied protocol-buffer slots.

2) OPTIONS-ordinary request. An ordinary (no special header needed) OPTIONS request can also serve as busy-status detection probe. An OPTIONS request is used to query the protocol capability of a SIP device, and can be used as a trace route tool. The adversary sends the OPTIONS requests to the phone and they will occupy the protocol buffer slots.

3) NOTIFY-check-sync and NOTIFY-refer requests. SIP provides an asynchronous event-notification scheme for signaling events (e.g. voice mail available) [33]. An event can be subscribed in advance and then will be delivered to a SIP phone. (The Internet's IANA organization maintains the database of valid event parameters [30].) The events are sent via NOTIFY requests. Thus, the adversary simply sends NOTIFY requests with specific event identifiers (e.g. with the header "Event:refer") to occupy the protocol buffer slots.

4) UPDATE-ordinary request. The UPDATE method changes the media parameter during a call. The UPDATE method is seldom implemented, and thus it is possible that unsupported UPDATE requests (and other unsupported methods) can still occupy the SIP protocol-buffer slots and allow busy-status detection.

2.2.3. Call Correlation

Busy-status detection can obtain the busy status of a phone at a specific time. However, depending on the phone type, the busy-status detection can detect whether the phone is busy either 1) at any time during the call, or 2) only at the beginning and/or the end of the call.

Hence, we define three tests for correlating busy status to infer a call. When both of the two target phones belong to the first type, we use the *continuity* test: the busy period of one phone (from time i to time j) should be close the busy status of another phone (from time x to y) such that $|x-i|+|y-j|<\varepsilon$, where ε is a constant indicating the upper-bound of measurement error. Otherwise, we use the weak or strong tests. The *weak* test is that the two phones are busy at the beginning or at the end of the call

such that $|x-i|<\varepsilon$ or $|y-j|<\varepsilon$. The *strong* test is that two target phones are both busy at the beginning and the end of the call such that $|x-i|<\varepsilon$ and $|y-j|<\varepsilon$.

2.3. The Attack Experiments

We set up a simple VoIP network including three (hardware) SIP phones to perform the busy-status detection and call-correlation steps. For the busy-status detection, we show how the phones respond to the probe requests. For call correlations, we derive the detection rate.

2.3.1. Environment

We implemented a program named *Voice Pulser*[6] to perform busy-status detection. Since the proposed attack does not require eavesdropping, we experimented on an ordinary LAN. Voice Pulser sent attack messages to the three phones, which were a Linksys PAP2, a Cisco 7940G, and a Grandstream BT-100. The PAP2 is a consumer-level product which was bundled by Vonage, a major VoIP service provider. The 7490G is a medium-level product that provides business phone features. Although Linksys is a division of Cisco, the PAP2 and 7940G use different protocol stacks and operating systems. The Grandstream BT-100 is an entry-level VoIP phone. Communication between the testing phones was relayed by a SIP proxy server.

[6] This program is hosted in google code, http://code.google.com/p/voice-pulser/

2.3.2. Busy-status Detection

2.3.2.1. Attacks based on INVITE

We examined the phone responses under INVITE-require (PAP2 and 7940G) and INVITE-SDP (BT-100) attacks with two conditions, namely 1) when the phone did not receive or make a call (Not Busy), and 2) when the phone made or received a call and was still active on the line (Busy). Voice Pulser sent 33 and 7 attack SIP messages to the PAP2 and 7940G, respectively, according their protocol buffer sizes. We sent 23 attack SIP attack messages to the BT-100. Figure 15 shows how the PAP2 received all 33 responses when it was not busy and missed the 33rd response when it was busy. Figure 14 shows how 7940G responded with an error '486 Busy Here' when the protocol buffer was full. Figure 16 shows that the BT-100 response, which was slightly different. In the middle of the 23 probes, we made a call from BT-100. A SIP phone should add a tag to the To field when responding to a request. E.g., "To: wj@iptel.org" in a request becomes "To:wj@iptel.org;tag=aa69981" in the response [31]. The tag must be cryptographically random and globally unique (i.e., a nonce). When the BT-100 was not busy, the To tag was a nonce consisting of digits and letters. However, when the BT-100 was busy, the To tag was fixed.

Based on the above observation, the adversary is able to determine the busy status of the PAP2 by finding out whether the 33rd response is received, the busy status of 7940G by finding out whether 6th response is a "486 Busy Here", and the busy status of the BT-100 by finding out whether two consecutive response have the same value for the To tag.

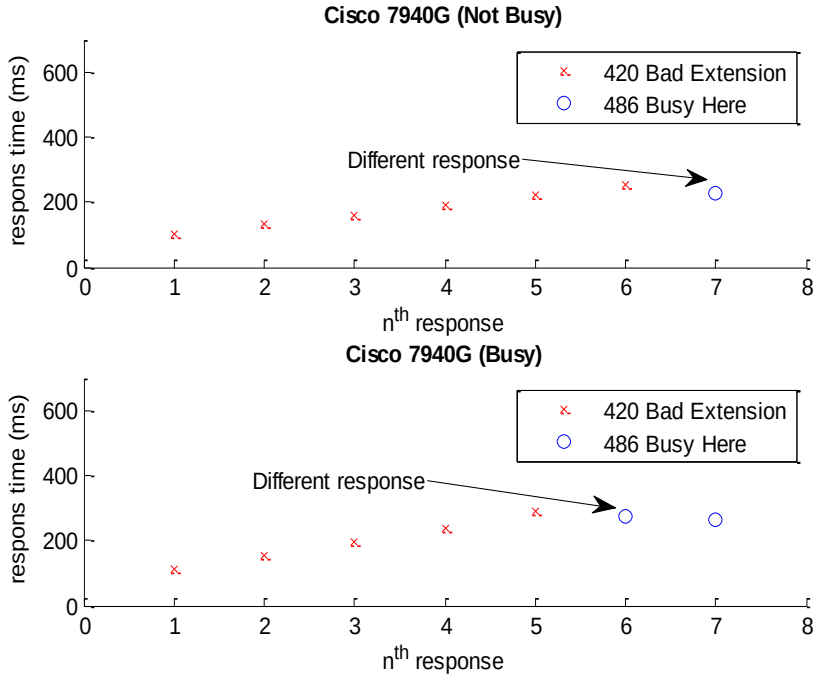


Figure 14. Responses of 7940G under INVITE-require attack (as a callee)

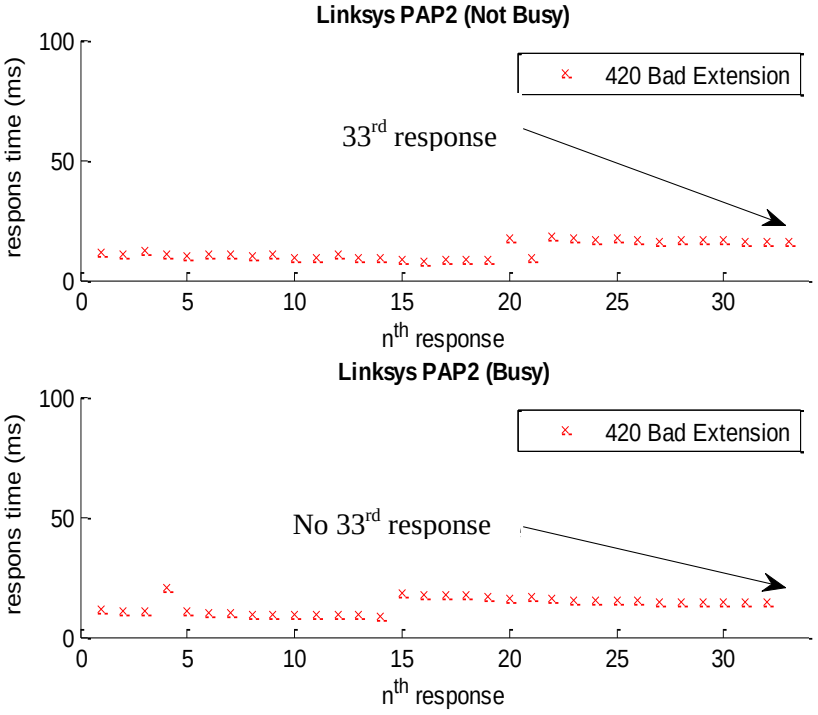


Figure 15. Responses of PAP2 under INVITE-require attack (as a callee)

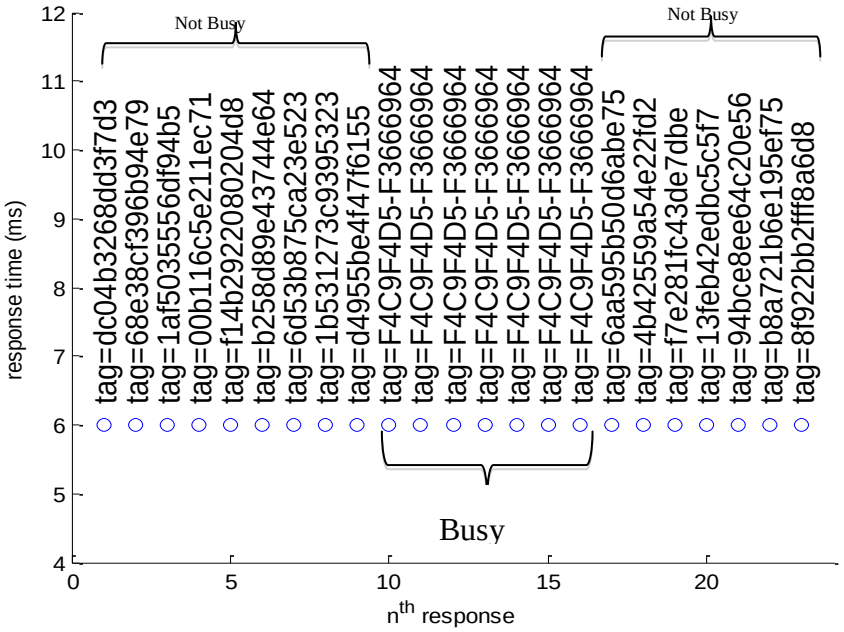


Figure 16. Responses of BT-100 under INVITE-SDP attack (as a caller)

2.3.2.2. Attacks

We evaluated all types of attack messages listed in previous section and show the results in Table 2 and Table 3. Table 2 indicates whether the busy-status detection was successful. Table 3 provides details for the phones responses to the probe requests and how the busy status can be identified.

Table 2. Phones against the attacks

	Linksys PAP2	Cisco 7940G	Grandstream BT-100
OPTIONS-ordinary	Success(S1)	Fail(F2)	Fail(F2)
INVITE-require	Success(S2)	Success(S4) *	Fail(F1)
INVITE-SDP	Fail(F1)	Success(S5) *	Success(S6)#
NOTIFY-refer	Success(S1)	Fail(F2)	Fail(F3)
NOTIFY-check-sync	Success(S1)	Fail(F2)	Fail(F3)
UPDATE-ordinary	Success(S3)	Fail(F3)	Fail(F3)

*: Disabled period shortening is supported
#: No disabled period

Table 3. Explanation of Table 2

Type	How the SIP phone responds
S1	"**200 OK**" response for first *h* requests No response for other requests *h*=32 if not busy; *h*=31 if busy
S2	"**420 Bad Extension**" response for first *h* requests No response for other requests *h*=**33** if not busy; *h*=32 if busy
S3	"**501 Not Implemented**" response for first *h* requests No response for other requests *h*=32 if not busy; *h*=31 if busy
S4	"**420 Bad Extension**" response for first *h* requests "**486 Busy here**" response for other requests *h*=6 if not busy; *h*=5 if busy
S5	"**500 Internal Server Error**" response for first *h* requests "**486 Busy here**" response for other requests *h*=6 if not busy; *h*=5 if busy
S6	The To tag in the response is a **nonce** if not busy; The To tag in the response is a **fixed** value if busy *h*=1; no disabled period When the phone is busy as the **caller**, To tag consists of digits and **upper-case** letters; When the phone is busy as the **callee**, To tag consists of digits and **lower-case** letters.
F1	The phone rings
F2	"**200 OK**" response for all requests
F3	Other responses (415, 401, 501, etc.)

The PAP2 phone had different responses to different request probes. It responded with "200 OK" to OPTIONS-ordinary, NOTIFY-refer and NOTIFY-check-sync attacks. Even though refer and check-sync events are not supported by PAP2, for NOTIFY-refer and NOTIFY-check-sync, it responded "200 OK" as well. For the INVITE-require probe, PAP2 rejected the request with message "420 Bad Extension" because the requested extension "xx" is not supported. Since the UPDATE method is not supported by the PAP2, for UPDATE-ordinary, it responded with "501 Not Implemented." It is noteworthy that the protocol buffer size **N** of PAP2 has a different value (**N**=33 instead of 32) in the INVITE-require request probe. This due to the fact that INVITE and non-INVITE transactions use different state machines.

The 7940G phone allows busy-status detection for only INVITE-based attack messages. We believe that the designers of the 7940G tried to avoid any unnecessary resource usage so that all requests, except those of the INVITE method, avoid filling the protocol buffer. Ironically, their conservative resource-management design enables disabled-period shortening.

The BT-100 phone is only vulnerable to INVITE-SDP. In contrast with the other phones, the adversary can easily identity whether a BT-100 phone is the caller or callee. The To tag consists of upper-case letters characters when the phone is the caller; the To tag consists of lower-case letters when the phone is the callee. Another difference is that the BT-100 phone does not have a disabled period such that attack messages do not occupy the protocol buffer slots but busy status is still revealed.

2.3.2.3. Response-Time Behavior

It is also important to know how often the busy-status detection can be performed. We used several different probing intervals and found the minimum values as shown in Table 4. The adversary can probe the PAP2 phone with a 34 seconds interval, the 7940G phone with a 2 seconds interval (when the disabled period is shortened), the BT-100 with a 3 seconds interval (no disabled period). These values are the time resolution of the Resource-Saturation PCD.

The disabled period also affects the ring delay, which is the time between call setup and ring. Table 4 also shows the ring delay. While the PAP2 phone has a longer ring delay under attack, the 7940G and the BT-100 have no significant ring delay under attack. For all three phones, the ring delay is essentially negligible for end-users unless they analyze the network traffic or phone logs.

Table 4. Timing behavior of busy-status detection

	LinkSys PAP2	Cisco 7940G	Grandstream BT-100
Minimum Probing Interval (Time resolution)	34s	2s	3s
Ring delay under attack* (mean/stddev)	4.8s/6.3s	12.3s/0.47s	0.1s/#
Regular ring delay* (mean/stddev)	1.41s/0.51s	12.4s/0.50s	0.1s/#

*:17 samples

#: The phone rings immediately such that the investigator cannot differentiate the ring delay between different tests

2.3.3. Call Correlation

A phone has four possible roles. For a call setup, a phone can either send the setup request (call-setup initiator) or receive it (call-setup recipient). Similarly, for call termination, a phone can either send the termination request (call-termination initiator) or receive it (call-termination recipient).

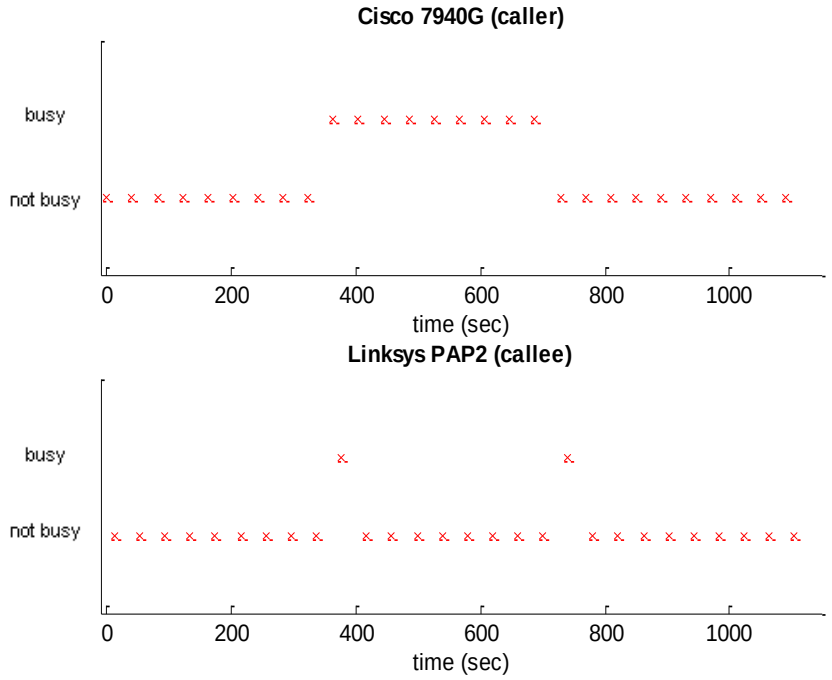


Figure 17. A call correlation experiment: 7940G calls PAP2 at 360sec and terminates at 700sec (7940G as call setup initiator and call termination initiator)

First, we used the following steps to demonstrate a scenario of call correlation in Figure 17. The 7940G called the PAP2 after Voice Pulser had started, at 360 seconds, and the 7940G terminated the call at 700sec. For the 7940G phone (call-setup initiator and call-termination initiator), the phone revealed its busy status during the entire call. In contrast, the PAP2 phone (call-setup recipient and call-termination recipient) only showed it was busy when the call setup and call termination were performed.

We then performed experiments where the three phones had different roles. The experimental steps are as follows:

1. Use Voice Pulser to probe the PAP2 and the 7940G (or BT-100) concurrently for their busy status with a 34 seconds interval.
2. Use one of the phones called the other.
3. Pickup the callee phone after 10 seconds (before ring), or after it rings, if the call setup requires more than 10 seconds.
4. Hang up the caller phone.

Table 5 shows the results. For the detection rate, there was an outstanding case when the PAP2 received the call setup request. The rate was 53%, instead of either 0% or 100%. The reason is that the PAP2 phone has a non-negligible disabled period. In that case, whether the adversary could detect the call setup depends on the time when the call setup was performed. We performed 17 experiments, varying the call setup; i.e., the call setup was performed at 5 seconds increments (60, 65,…, 140) after the Voice Pulser started monitoring. The timing spread over two Voice Pulser monitoring intervals (2×34 seconds).

The 7940G and BT-100 phones achieved a 100% detection rate in all four roles. This means that if the adversary monitors 7940G or BT-100 phones, she will detect all VoIP calls between them via the continuity test. In contrast, the adversary will not be able to detect calls if a PAP2 phone is busy when it initiates or terminates a call. However, this finding does not imply that the PAP2 is invulnerable to our attack. If the adversary monitors a PAP2 phone and a 7940G (or BT-100) phone, she can still detect the VoIP calls between them, except in the case when the PAP2 calls the 7940G (or BT-100) and the PAP2 terminates the call.

Table 5 also shows that the false alarm rate was extremely low. To estimate the false alarm rates, we probed the phones continuously for over 13 hours (1459 probes). The PAP2 phone only had a 0.2% false alarm rate whereas the 7940G and BT-100 had a rate of 0%.

Table 5. Performance of busy-status detection

Type	Role	PAP2	7940G	BT-100
Detection rate	Call-setup initiator(caller); sending INVITE request	0%	100%	100%
	Call-termination initiator; sending BYE request	0%	100%	100%
	Call-setup recipient(callee); receiving INVITE request	53%*	100%	100%
	Call-termination recipient; receiving BYE request	100%	100%	100%
False alarm rate	No actions	0.2%	0%	0%

*: Tested by 17 samples

We present the detection rates of call correlation for the 7940G and PAP2 in Table 6. Since the 7940G and BT-100 had the same detection rates in all roles, they are identical in deriving the detection rates of call correlation. In general, the detection rates of the 7940G (and the equivalent BT-100) are better than the detection rates of

PAP2. If the two target phones are 7940G (or BT-100) in case 6 of Table 6, then detection rate is 100%. In contrast, if the two target phones are both PAP2 (case 5 of Table 6), then the detection rate is 0%. If the two target phones are 7940G (or BT-100) and PAP2, the detection rates depend on phones' type and their roles in a call.

Table 6. Detection rate of call correlation derived from Table 5

	Call setup		Call termination		Detection rate (*continuity* test)	Detection rate (*weak* test)	Detection rate (*strong* test)
	initiator	recipient	Initiator	recipient			
Case 1	PAP2	7940G*	PAP2	7940G	N/A	0%	0%
Case 2	PAP2	7940G	7940G	PAP2	N/A	100%	0%
Case 3	7940G	PAP2	PAP2	7940G	N/A	53%	0%
Case 4	7940G	PAP2	7940G	PAP2	N/A	100%	53%
Case 5	PAP2	PAP2	PAP2	PAP2	N/A	0%	0%
Case 6	7940G	7940G	7940G	7940G	100%	100%	100%

*: 7940G in this table can be replaced by BT-100 since they have same detection rate as shown in Table 5

2.4. Summary

We proposed an attack for discovering call records in a VoIP service. We analyzed the SIP protocol and discovered that array-based buffers of three commercially available closed-source hardware phones, which are used for storing protocol negotiation contexts, leak the busy status of a SIP phone. Through the leaked busy status of a callee's phone, an adversary can easily detect the VoIP communication between phones. This attack is general enough to hold in other types of hardware phones.

Chapter 3: Stochastic PCD

In this chapter, we propose the Stochastic PCD approach that performs PCD attack against WiFi and Instant Messaging through the concept of classifying the response time series. The main challenge is to deal with the noise in the response time series introduced by Internet switching, application proxies, and other nodes in the same network. The proposed approach can also be used to detect the Sybil attacks. The work in this chapter was previously reported in [36].

3.1. Overview

Problem Statement. Our network setting, abstractly shown in Figure 18 and Figure 19, illustrates the challenge faced by an ordinary user in performing PCD against two targeted network users. Ordinary user Eve sends repeated probe messages to Alice and Bob's nodes and analyzes the timing of their responses to determine whether these nodes communicate with each other, or alternatively whether their nodes are implemented by the same physical network node (as in a Sybil node[7] instance). The type of messages Eve sends will not alert Alice and Bob of Eve's repeated probing. However, Eve faces two Stochastic PCD challenges. First, the network may have other third-party nodes that communicate with Alice and Bob's nodes separately. As a consequence, the timing of Alice and Bob's node response to Eve's probe messages might be similar to those when Alice and Bob communicate with each other, and Eve

[7] Sybil nodes are undesirable because they negatively affect resource allocation, bandwidth utilization and routing [47, 56].

must be able to distinguish the former from the latter case. Second, network routers and application-layer proxies[8] located between Eve's node and those of Alice and Bob, respectively, can introduce jitter and delays. Since most interactive WiFi applications require low latencies (e.g., ITU G.114 recommends under 400ms latency for voice communication), the jitter and delay incurred by such networks would not prevent an ordinary user from performing the Stochastic PCD[9].

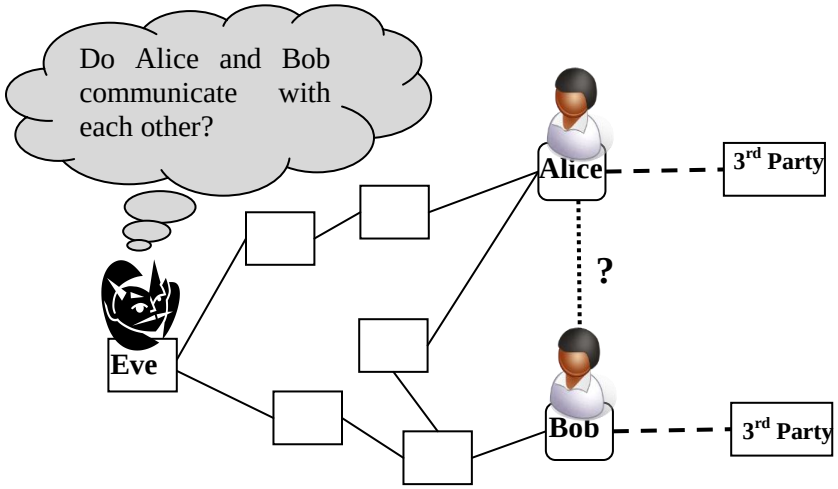


Figure 18. Private communication detection

[8] Application-layer proxies enable mobility and provide some degree of privacy and security; e.g., a smartphone registering with a server ensures that messages will be forwarded to the phone (mobility) and the phone's IP address is hidden (privacy). Moreover, a phone receives a packet only through a pre-established connection initiated by the phone to the server (security). This limitation exists when the node is behind a network address translation router (e.g., AT&T's data plan)

[9]However, we note current low-latency anonymous networks, such as Tor and I2P [17, 20, 29], cannot be used for real-time WiFi applications since their latency in the order of seconds.

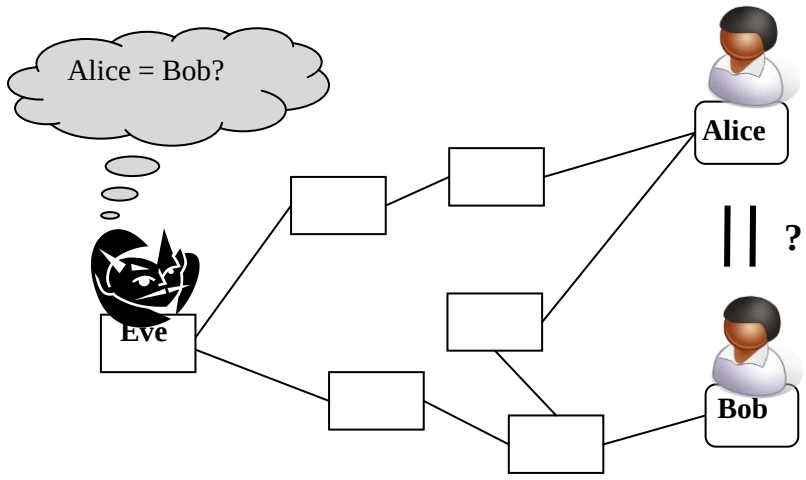


Figure 19. Remote Sybil detection

Our Approach. In the simplest form of the Stochastic PCD, ordinary user Eve is presented with two hypotheses, which she must test in the presence of separate third-party communication with Alice and Bob's nodes, and network jitter and delay.

For private communication detection,

H_0: Alice and Bob do no communicate with each other
H_1: Alice and Bob communicate with each other

For remote Sybil detection,

H_2: Alice and Bob are not physically the same node (non-Sybil)
H_3: Alice and Bob are physically the same node (Sybil)

During a period of probing, Eve sends packets to Alice and Bob's nodes and receives a (time) series of response times. Given the time series, she can use a classifier to detect which hypothesis (e.g., H_0 or H_1) is more likely to be true.

The reason why classifying the time series of response times enables the detection of the private communication is that the timing responses caused by resource contention in

Alice's, or respectively Bob's, node (e.g., back-off in CSMA/CA) are different depending on the type of application running on that node. For example, if Alice and Bob's nodes communicate with each other via a WiFi network in a VoIP application, the WiFi radio channel contention caused by Eve's probe messages would differ from that caused by Alice's or Bob's communication with third parties in web browsing or file transfer applications. Hence, ordinary user Eve can construct two models of timing responses, obtained when two remote nodes communicate with each other via a VoIP application and when two remote nodes do not communication with each other via a VoIP application. Then Eve can apply supervised classification to the time series of responses obtained from Alice and Bob's nodes in response to her probes, in order to detect their communication over VoIP, if any. In short, Eve can exploit a *timing side channel* that is caused by the different types of resource-contention responses when end nodes run different applications.

Eve can use Markov Chain models for classification and leverage its success in stochastic time series classification of automatic speech recognition [37, 44, 65]. She can also apply signal-processing techniques to filter out the noise from the time series of responses; i.e., we use a high-pass filter to eliminate the noise in certain frequency components. Finding good filters to fit the different time series of device responses is a routine design problem in signal processing, and thus any ordinary user can use standard tools (as illustrated later in this chapter) to analyze responses generated from the different devices and protocols. In performing such analysis for PCD, ordinary user Eve can merge the time series of different probe responses obtained from the targeted nodes into a single time series, such that the interactivity between the two targeted

nodes can be established in a specific application, such as VoIP.

Limitation. The proposed approach is most effective when the stochastic models chosen to represent the time series of probe responses can accurately model random-access networks (e.g., Markov Chains can model CSMA/CA networks well), and when signal-processing techniques can eliminate much of the noise introduced by the third-party nodes. This is important because high-frequency target probing during the detection period may also cause adversary detection by the targets. Hence, the adversary must strike a delicate balance between detection accuracy and the possibility of attack detection.

Sybil Attacks. There are a number of studies of Sybil detection in wireless, peer-to-peer networks, and other areas. Levine *et al*. provide a comprehensive survey [46]. They conclude that Sybil attack defenses can be grouped into four categories: trusted certification, resource testing, recurring costs and fees and trusted devices. Our remote Sybil detection is a variation of resource testing, which implicitly asks the nodes to access certain shared resources (e.g., radio channels) to show they are not Sybil nodes [55]. Unlike other resource testing methods, however, our method does not require the target or its neighbor to perform specific resource testing actions (i.e., that a node send packets to the radio channel at a designated time). In other words, our Stochastic PCD approach can utilize the original networking environments without modifying the software or protocols.

3.2. Practical PCD Applications

We will examine two applications for the Stochastic PCD as summarized in Table 7. The first application is WiFi where nodes compete for the radio channel. The second application is Instant Messaging, where the user switches between different chat windows, making the keyboard the resource under contention. The time series of probe responses for WiFi application has strong noise while time series data for Instant Messaging has negligible noise. For each of these two applications, we describe the application, experiment environments, and procedure of experiments.

Table 7. Two PCD applications

	WiFi	Instant Messaging
Resource under contention	Radio channel	Keyboard
Noise due to non-targeted nodes	Yes	No*
Noise due to internet and application-layer proxy	Yes	Negligible#

*:For this experimental setting
#: Human actions in Instant Messaging are measured in seconds, while network delay and jitters rarely exceed 1 second

3.2.1. WiFi

WiFi provides multiple access to radio channels through IEEE 802.11 protocol suite as shown in Figure 20. In a radio channel of the selected frequency, nodes compete for channel usage according to CSMA/CA protocol. When a node wants to send a frame, if another node is using the channel or is expected to use the channel (virtual carrier sensing), this node will be back off and try to re-access the channel after a random period; otherwise, the node sends it immediately. In this way, a node's traffic affects other nodes in the same WiFi network.

Because of this structure, if an ordinary user Eve periodically sends probe packets

remotely to a targeted node in a WiFi network, the response time will be influenced by the traffic of the WiFi network. If two targeted nodes Alice and Bob in the same WiFi network are communicating with each other under constant-rate traffic (e.g., Voice), the impact on the transmission time is retained during this probing period. Suppose the traffic of other nodes (not Alice nor Bob) in the WiFi network are web browsing video streaming, or SSH, but not VoIP, the remote ordinary user Eve can distinguish whether VoIP is in session by the time series of the probe responses.

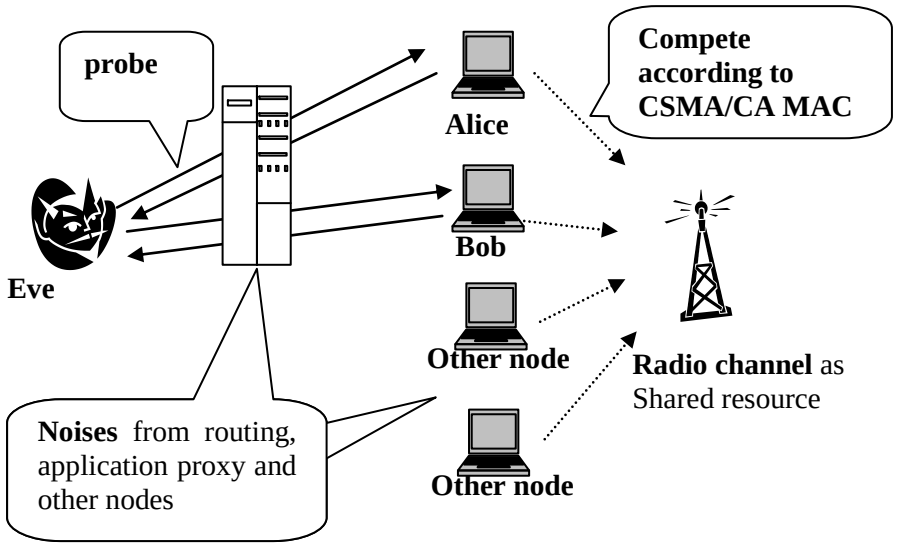


Figure 20. Radio channel is the shared resource in WiFi

3.2.1.1. Experimental Environment

Two nodes Alice and Bob are in the same WiFi network provided by a WiFi Access Point, which is connected to the Internet. Since PCD depends on the stochastic time signals, not the protocols, we would like to know PCD's performance in different network environments. The experiments are performed in two different locations. The

university café is in the lobby of a large computer science building where hundreds of WiFi users can access the internet through over about 10 WiFi access points. Hence, the university café is a wireless environment with high dynamics. In the author's apartment, WiFi user dynamics are limited to only one to two WiFi users, in additional to the Alice and Bob.

The configuration for the private communication detection is shown in Figure 21. Nodes Alice and Bob are two laptops, a MacBook Air (Mac OS X version Lion) and a MacBook Pro (Mac OS X version Snow Leopard), respectively. Both have the VoIP client CounterPath X-Lite installed. Eve has the VoIP IDs of node Alice and node Bob, and therefore is able to send probe VoIP packets to the targets through a VoIP proxy and get responses. The VoIP proxy is an OpenSips server running on a Linux server located in a university computer cluster room [56].

The configuration for the remote Sybil detection is shown in Figure 22. The target is a HP laptop which is running Windows 7 and has an Intel WiFi NIC. This Intel NIC supports “My WiFi”, a wireless NIC virtualization technology that enables the wireless NIC to serve as different roles in the same time [12, 77]. When “My WiFi” is enabled, an Apple iPhone 4 can access the Internet through the laptop. In this case, the laptop has two virtual identities, fulfilling hypothesis H_3 for remote Sybil detection. One identity connects to the WiFi Access Point and gains Internet access while the other identity acts as a WiFi Access Point, providing WiFi service to iPhone. The laptop is also equipped with VoIP client X-Lite so that Eve can probe the laptop through the VoIP proxy.

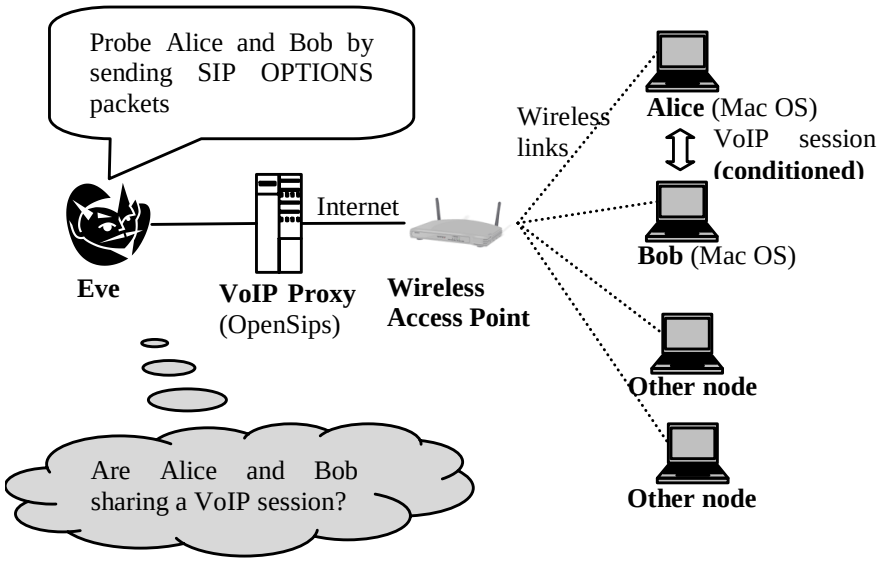


Figure 21. Experimental environment for WiFi private communication detection

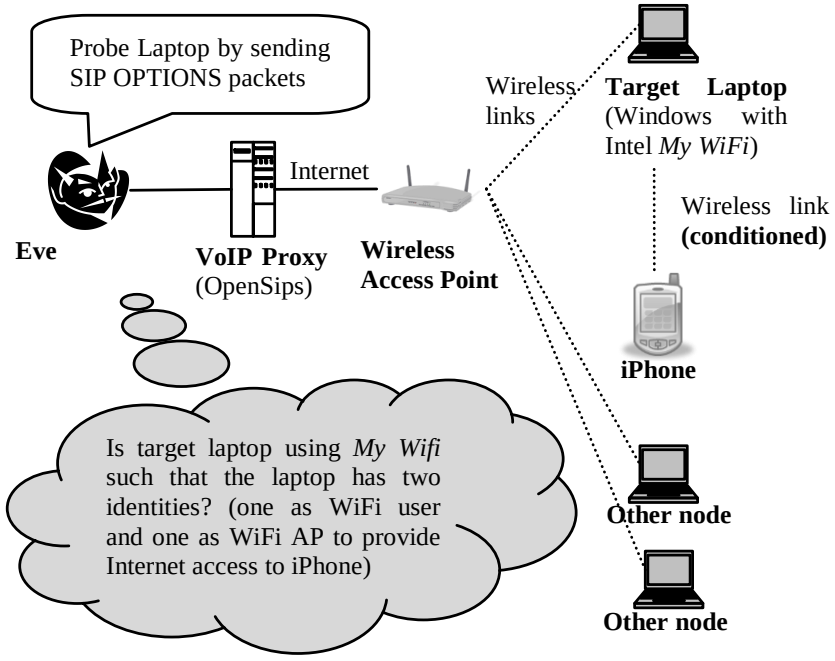


Figure 22. Experimental environment for WiFi remote Sybil detection

3.2.1.2. Procedure

In both of the settings, Alice or Bob will browse the web and use SSH to connect to remote site as the communication with third parties. For the private communication

detection, we condition upon whether Alice and Bob share a VoIP communication (H_0 and H_1). Eve uses the program Voice Pulser (as seen in Chapter 2) to probe both Alice and Bob. The program will send SIP OPTIONS requests, VoIP probe packets in the Session Initiation Protocol (SIP), to targeted nodes with a 100ms interval (~10 Hz) [31]. For each condition, the program probes the targeted nodes 120000 times. In Section 3.4.2. , we will split the results in each condition to 12 samples.

For the remote Sybil detection, we condition upon whether the laptop enables "My WiFi" and shares the Internet with the iPhone (H_2 and H_3). However, Eve only probes the laptop, not iPhone, due to the fact that the IP address obtained by the iPhone is private and not routable (e.g., 192.168.0.0/16). For each condition, the program probes the targeted nodes for 120000 times as well.

3.2.2. Instant Messaging

The Instant Messaging (IM) service is an interactive social medium that enables person-to-person real-time text chatting. Common systems include Facebook, AOL AIM, Yahoo Messenger, Microsoft MSN Messenger, Google Talk, and Skype.

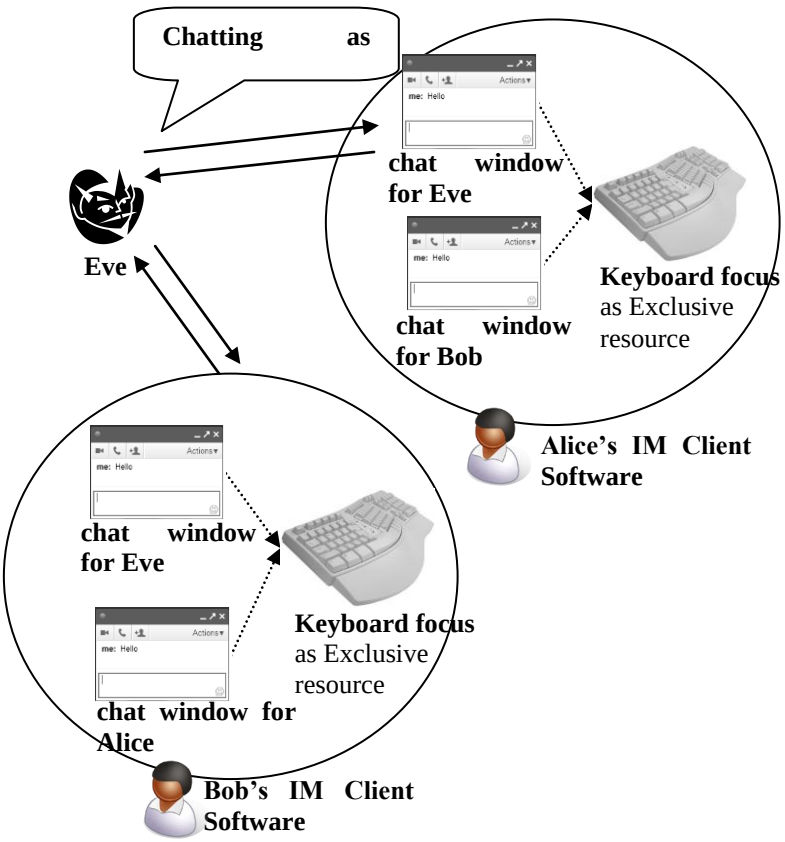


Figure 23. Keyboard is the shared resource in Instant Messaging

On the user interface, IM has one contact status window that shows a list of contacts. The IM user can select a contact and begin a one-to-one text chat with the contact via another window. Due to human limitations and the design of the user interface, the IM user can only type in only one window at one time (i.e. only one window gains the keyboard input focus). Therefore, keyboard, or the corresponding end-user attention, is the resource under contention. Chatting with one contact affects chatting with a different contact because the end user needs to switch between chat windows.

Hence, the shared keyboard causes side channels as illustrated in Figure 23. When Eve chats with a targeted user, the timing of the interactions will be affected by the chatting between the targeted user and another user, if one exists. Moreover, besides

the regular text messages, all major IM systems support composing notification messages[10], which are sent periodically when the IM user is typing on the keyboard, but has not yet sent the message. Therefore, Eve can use the received regular text messages and composing notification messages to obtain the private information of interest.

3.2.2.1. Experimental Environment

Human participants use IM client software named pidgin to chat with their contact [63]. The communications are relayed through an IM server. The pidgin is modified to log received regular text messages and composing notification messages. The investigator (Eve in this case), who also participates in chatting, records the timing of received regular text messages and composing notification messages. In addition, although IM allows conference chatting, we only focus on one-on-one text chatting with one or more contacts.

It is usually a challenge to recruit human subjects for research, and our experiment is no exception. We use the following approaches to ease the recruitment and make maximum use of the data:

1. Virtualization. The participants do not need to come to the designated site. Instead, they use their own computers to connect to the virtual machine where pidgin is installed and investigators' instructions will be displayed. The

[10] There are different names for composing notification messages. We use the name composing notification from XMPP, an IETF open standard.

investigator's computer hosts these virtual machines so that the investigator can monitor whether the participants follow the instructions.

2. Cross-validation: Cross-validation is used to exhaustively utilize the limited number of experiment samples [73]. Given all the samples, a certain number of samples are picked as the training set and the rest are used as the testing set. By iterating combinations of the training vs. sample set, the result utilizes all samples for both training and testing.

3.2.2.2. Procedure

For private communication detection, we have 5 groups of participants (2 participants/group) who perform 4 sessions of a 5-minutes experiment. In two of the sessions, two participants both chat with the investigator (H_0). In another two sessions, two participants both chat with the investigator as well as chat to each other (H_1). Each session generates a sample of 5-minutes time series and totally 10 samples are available for each hypothesis.

For remote Sybil detection, we can also use the data for private communication detection. In private communication detection, we have 20 existing samples that correspond to H_2 (non-Sybil) data in remote Sybil detection. Therefore, we need only to perform experiments where H_3 (Sybil) is hold. To this end, 5 groups of participants (1 participant/group) perform 4 sessions of experiments where each participant uses two IM accounts to concurrently chat with the investigator. Therefore, we have the samples for H_3 in remote Sybil detection. Combining these 20 samples with the 20 samples obtained in private communication detection as H_2 in remote Sybil detection,

we have 20 samples available for both hypotheses for remote Sybil detection.

Before a group of participants begins the experiments, they remotely login to the virtual machine and the investigator gives instructions for how to use the environment. Besides pidgin, there is a program on the virtual machine that displays the investigator's instructions. The investigator gives instructions to the participants for when to chat and whom to chat with.

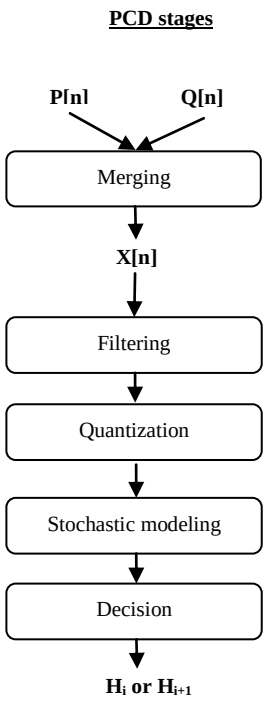
During the experiments, the pidgin program logs the timing of regular text messages and composing notification messages. The event times are then converted to a time series with the resolution in seconds. Data points in the converted time series indicate whether or not the participant has been busy typing to the investigator at each second. The converting algorithm is described as follows.

The *estimated busy period* is when the contact is busy typing to the investigator. This begins when a composing notification is received, and ends when a regular text message is received. However, the period could be over-estimated in the following scenario. An IM user types something to a contact but does not hit send, but instead switches to chat with another contact. Eventually the user switches back to send the unsent message to the first contact. A simple and effective way is to set an upper bound of the estimated period, say 15 seconds. We assume that 15 seconds is a reasonable period for someone to type and send one message. If the estimated period is larger than 15 seconds, we assume that the estimated busy period is 15 seconds long and ends at the time when the text message is received.

3.3. The Attack

We show the architecture of PCD followed by its components and the design choices we have made.

3.3.1. Architecture



PCD stages	Purpose	Our Implementation choices
Merging	Converting 2D input time series to 1D	Cartesian product or Hadamard product
Filtering	Reducing noises	Butterworth high-pass filter and log()
Quantization	Reducing state space	Equally quantized
Stochastic modeling	Representing time series by model	1st order Discrete time Markov Chains
Decision	Giving final results by comparing conditional probability of time series	Maximum Likelihood

Notation:
X[n]: The input time series or a certain data point of the input time series indexed by n.
P[n] and Q[n]: If the input time series is 2-dimensional, then P[n] and Q[n] represent the time series of the first and the second dimension, respectively.

Figure 24. Stochastic PCD

PCD inputs a time series of response times and outputs which hypothesis is more likely to be true. The input time series is either 1-dimesional or 2-dimensional, is discrete in time, and can be discrete or continuous in value.

As illustrated in Figure 24, PCD analysis has a 5-stage design, which is partly

inspired by Automatic Speech Recognition [37]. The input time series are processed through each applicable stage and the private information of interest is generated in the end.

1. The first stage, *merging,* is only applied when the PCD input time series is 2-dimensional. The merging stage will convert P[n] and Q[n] to a new time series, denoted as X[n] for convenience. This stage reduces dimensionality for ease of classification but retains the information contained in both P[n] and Q[n].

2. The next stage, *filtering,* eliminates the noise contained in series X[n]. If X[n] is noisy and this stage is skipped, it is likely Eve cannot gain an information advantage (i.e., accuracy is not much higher than 50%). The filtering stage inputs a time series and outputs another time series. Through filtering, the frequency components where noise resides will be eliminated.

3. The third stage is *quantization*, which reduces the number of possible values of the time series (i.e., size of the sample space). When there are more possible values, a larger the number of parameters will be needed in the next stage.

4. The fourth stage, *stochastic modeling*, which has two stochastic models for the two hypotheses. A stochastic model represents a variable-length time series through a fixed and small set of parameters. Through stochastic models, two operations are available: 1) to estimate the parameters of the stochastic models by given time series and 2) to evaluate the probability of a time series

given the model.

5. The final stage, *detection,* is to make the detection decision based on the probability of a time series given the pre-trained models trained.

Assumptions and limitations. PCD assumes that the private information of interest (i.e., whether two targets are communicating; whether nodes Sybil nodes) does not change during the period that PCD probing is performed. This assumption indicates that the time resolution of PCD may be limited and application-dependent. Furthermore, high-frequency target probing may also cause the detection of the attack by the targets. In contrast, in passive probing, such as Instant Messaging, adversary detection is irrelevant since PCD is only applicable when there is an active conversation between the adversary and the targets.

Stochastic PCD can be applied effectively in network protocols where stochastic models capture the time series of probe responses accurately, and when signal processing can eliminate most of the noise introduced by active third-party nodes. The former suggests that stochastic PCD is well suited in random access networks (e.g., CSMA/CA), since Markov Chains can model them properly. In contrast, stochastic PCD is not easily applicable to non-random-access networks such as TDMA. The latter suggests that if the traffic between the target and the third-party nodes (e.g., web browsing, video streaming, and SSH) is different from the traffic between the targets (e.g., VoIP communication), the filtering component of PCD can effectively remove the noise brought by the third-party nodes. However, if the traffic between the targets and third-party nodes is the same with the traffic between the

targets, PCD may experience false detection. Despite this, by repeatedly performing PCD, the adversary can still determine the long-term relationship between the targets [25]. Detection confidence increases as the number of communication instances increase.

Optional stages. Some of the stages are optional or unnecessary in certain situations. If the input is 1-dimensional, the merging stage is skipped. If the input time series is not noisy, the filtering stage can be skipped. If the input time series are discrete-valued and the number of distinct values is small enough, the third stage (quantization) can be skipped. If Eve is training models, the last stage is skipped.

3.3.2. Components

We will describe the choices of each component of PCD. Although our choices are not optimal due to the large design space, our choices do show the achievability of PCD with reasonable accuracy as seen later in the evaluation section. Finding better component implementations, such as trying other filters or stochastic models, would be among the important future work.

3.3.2.1. Merging Stage

We have two choices, Cartesian and Hadamard products, to implement the merging stage. The former is suitable for discrete input time series (e.g., IM application) while the latter is suitable for continuous input time series (e.g., WiFi application).

Cartesian Product. If P[n] and Q[n] are discrete in values, the Cartesian product is the simplest way to merge the two time series. For example, if two input series have

sample space (possible values) {0,1}, then the output time series will a sample space {{0,0}, {0,1}, {1,0}, {1,1}} which contains four difference values. The formula of the Cartesian product merging can be written as

X[n]={P[n], Q[n]} for every n.

The size of the new state space is the square of the size of the original state space. However, the Cartesian product has the potential problem of state space explosion, i.e. when the new state space becomes so large that the amount of training data for the stochastic models becomes unwieldy. Therefore, if the input time series is continuous in value, the Cartesian product is inappropriate and the following Hadamard product should be used.

Hadamard Product. The Hadamard product merge two (continuous-valued or discrete-valued) time series into one, while retaining the information of P[n] and Q[n].

X[n]=P[n]Q[n] for every n

The idea comes from the fact that the Hadamard product is a way to measure the *similarity* between two vectors. For example, the Fourier transform is a function of frequency that converts a time series to the magnitude of a given frequency. The value of a Fourier transform at frequency *f* is the sum of Hadamard product of the time series and a sine/cosine function. Similarly, then the new X[n] will go up or down when both P[n] and Q[n] go up or down and hence X[n] keeps the information of P[n] and Q[n].

3.3.2.2. Filtering Stage

The filtering stage uses digital filters to improve the quality of the input time series data to get a better final PCD accuracy. A digital linear filter of input X[n] is a mechanism to process a time series by calculating the difference equation as shown below. The parameters a_i and b_i decide the behavior of a causal filter and N is the order of the filter.

$$Y[n] = \sum_{i=1}^{N} b_i Y[n-i] + \sum_{i=0}^{N} a_i X[n-i]$$

High-Pass Filtering. A high-pass filter is a filter that attenuates the low frequency components of a signal. The frequency components with frequency lower than a given cut-off frequency will have lowered magnitude. Similarly, a low-pass filter attenuates the high frequency components of a signal. Modern filter design use a prototype filter (e.g., the Butterworth filter) and convert it to be low-pass, high-pass, or an even band-pass filter with different cut-off frequencies. When applying high-pass or low-pass filters on the filtering stage, Eve should choose the type of filter (e.g. Butterworth, Elliptic filters), the order (the value of N), and the cut-off frequency such that the final PCD accuracy is maximized. Our empirical experience on the WiFi application shows that the type and the order of the filter do not have much impact, but the cut-off frequency dominates the performance.

Log. log() can be applied before or after the filtering such that Eve can emphasis the signal change in small values. This is because log() is an increasing and concave

function. Other increasing and concave functions, such as the square root, should have a similar effect. Note that if the log transformation is applied before filtering, the Hadamard product of merging stage becomes an addition.

3.3.2.3. Quantization Stage

If the input time series is continuous or is discrete but the set of possible values (sample space) is large, its value should be converted to a smaller discrete set for efficient stochastic modeling. Stochastic models have parameters which need to be trained before use. In general, a stochastic model with more parameters needs to be trained with a larger set of training data.

We use the simplest quantization scheme which divides the interval between the maximum value and the minimum value into equal size. However, if log() is applied in the previous stage, the equal-quantization can be regarded as log-scale quantization.

3.3.2.4. Stochastic Modeling Stage

Our choice of stochastic model is 1^{st} order discrete-time Markov Chains due to the fact that it is simple, easy to analyze, and the number of parameters is relatively small. The more parameters included, the more data that is required to train the model. However, it is a common practice in signal processing to model signals by 1^{st}

order Markov Chains first and then use more complex stochastic models [11] if necessary.

Markov Chain. A discrete-time Markov Chain is a sequence of random variables $Y_1,Y_2,...$ with the Markov property, that is the state transition from Y_n to Y_{n+1} will not be affected by any of the previous states $Y_{n-1},...,Y_1$. Or we can write as $Pr\{Y_{n+1}|Y_n\}=Pr\{Y_{n+1}|Y_n,Y_{n-1},...,Y_1\}$. We can think of Markov Chains as capturing the relationship between a data point and the next data point in a time series. As a Markov Chain characterizes all pairs of adjacent data points, it characterizes a time series. The parameters of a Markov Chain are the transition probabilities from one state to another state. Therefore, for a state space of size *k*, a Markov Chain has k^2 parameters, namely a transition matrix.

3.3.2.5. Detection Stage

Maximum likelihood. The previous stage gives the conditional probability of a time series, given the stochastic models. In this stage, the conditional probabilities will be used to make the final detection result. One of the most popular methods is maximum likelihood (ML), where the detection result belongs to the hypothesis that has larger conditional probability. For the private communication detection, suppose M_0 and M_1 are models corresponding to H_0 and H_1, the result is

H_i, where $i=1\{Pr\{Y|M_0\}<Pr\{Y|M_1\}\}$

[11] For example, semi-Markov models and N-order Markov chains are more general and have more parameters (more training data). Hence, it is not practical to deal with the small set of data in the IM application.

For the remote Sybil detection, suppose M_2 and M_3 are two models corresponding to H_2 and H_3, the result is

H_i, where $i=1\{Pr\{Y|M_2\}<Pr\{Y|M_3\}\}+2$

3.4. The Attack Experiments

First we will show what the time series in both applications looks like. For WiFi application where the signal is noisy, we also show why we would need filtering due to the signal characteristics. We then show the accuracy of both applications as the performance index. The accuracy is defined as the number of correct detections divided by the total number of detections. Since we have half samples in H_0 and another half samples in H_1, a blind guess (50% accuracy) becomes the baseline comparison rate. If the accuracy is much higher than 50%, we can say Eve gains information advantages through PCD.

Table 8. PCD settings in two applications

		WiFi	Instant Messaging
Time series		Continuous-valued; Discrete-time	Binary-valued; Discrete-time
1.Merging stage	PCD (2-D)	Hadamard product	Cartesian product
	remote Sybil detection (1-D)	N/A	N/A
2. Filtering stage		Butterworth high-pass filter with order 10; log() used	N/A
3. Quantization stage		Equally quantized by 12	N/A
4. Stochastic modeling stage		1st order discrete time Markov Chain	1st order discrete time Markov Chain
5. Decision stage		Maximum Likelihood	Maximum Likelihood

The performance of both applications is evaluated under the settings as shown in Table 8. It is notable that since the time series of IM is a binary-valued (whether a

targeted node is busy typing to Eve), we can use the Cartesian product in merging, and skip the filtering and quantization stages.

3.4.1. Signal characteristics

Two samples of time series for WiFi application for private communication detection are shown in Figure 25. We have two samples of 10000 data points (~1000 seconds) under the hypotheses H_0 and H_1. We can see that values are mostly close to 0ms and around 500ms. The 500ms values are due to the packet lost. Because Eve sends VoIP packets through a VoIP proxy, the proxy will retransmit if the responses are not received in 500ms as defined by Timer E in RFC 3261[31]. In this figure, it is not easy to visually find the difference of these two signals, except the two spikes in H_0.

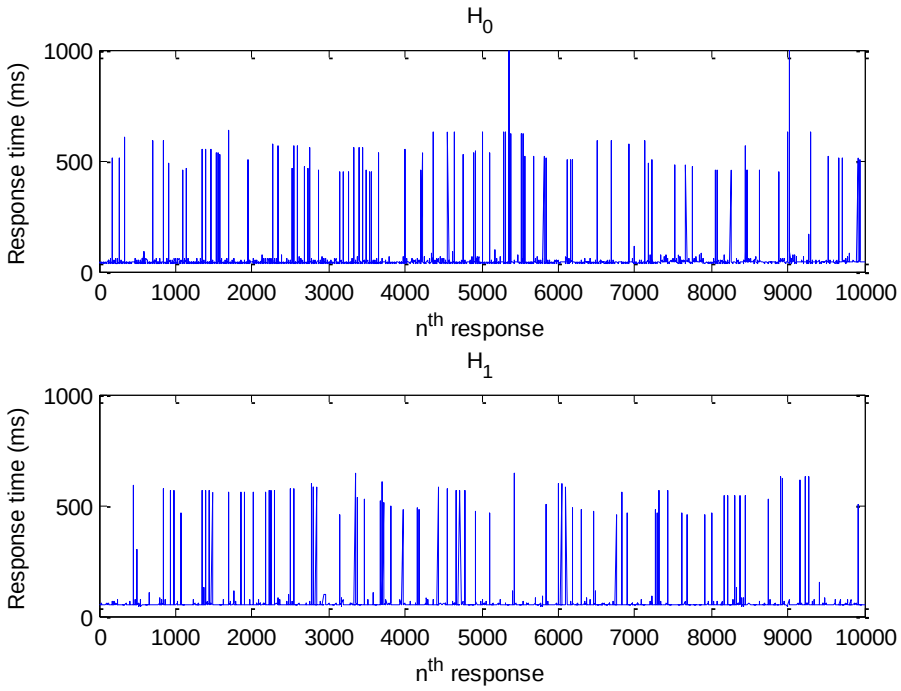


Figure 25. Two examples (H_0 and H_1) of WiFi time series for private communication detection

Frequency analysis helps to analyze time series in another aspect. A signal can be presented in the time domain as a time series, as well as in the frequency domain. A signal in the time domain is a function of time to show the magnitude at different time. On the other hand, a signal in the frequency domain is a function of frequency to show the magnitude at different frequencies. The Fourier transform can translate a signal from the time to frequency domain. Therefore, by applying the Fourier transform, we can examine a signal in the frequency domain, allowing us to explore the properties that are not easily observed in the time domain. However, the original Fourier transform has the limitation of being a global analysis. For example, if a sine wave appears only in the beginning of the time series, when in the frequency domain, we do not know whether this wave appears in the beginning or in the end of the time series. Such limitation bring us the problem that we will not know if a pattern shown in the frequency domain appears in the whole time series, or just in part of it.

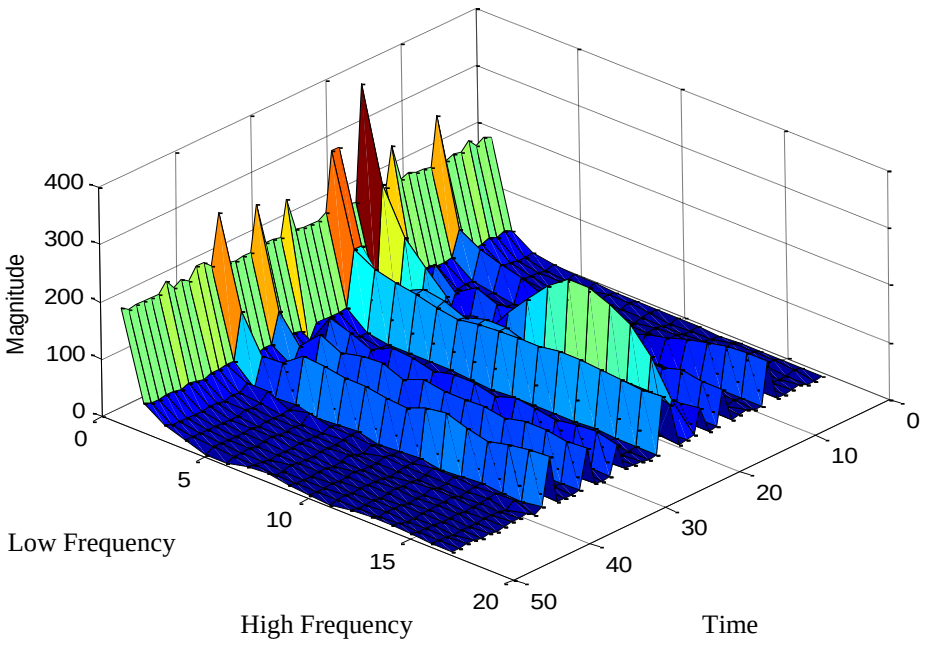


Figure 26. Frequency-time analysis of the H_0 signal in Figure 25 shows strong frequency component in high frequency

To overcome the drawback of frequency analysis, time-frequency analysis can be used to look at the time and frequency simultaneously. For example, Short Time Fourier Transform (STFT) does Fourier transform on a portion of the signal, repeated many times so that the whole signal is covered. Figure 26 shows the frequency-time analysis of the signals in Figure 25. It is particularly noticeable that the low frequency components remains high at all time. It is easy to say that the low-frequency components are either very important or not wanted at all (noise). From this observation, we tested high-pass and low-pass filters with different cut-off frequency and found that that removing the low-frequency components via high-pass filters helps to improve the performance of the Stochastic PCD in WiFi significantly.

But why are the high-frequency components important and why are the low-frequency components noisy? In Figure 27, we zoom in and see the first 500 data points of the Figure 25 signals. Except for the spikes caused by packet lost, the response time is about 50ms with small fluctuations. The fluctuation is network jitter caused by Internet switching and the WiFi back-off. Since the side channel we are interested in is the WiFi back-off behavior, the fluctuation of the signals may contain the useful information we are looking for. Since log(x) is a strictly increasing and concave function when $x>1$, it can emphasis the signals in small values, where the fluctuations are. We applied log() before and after high-pass filtering (twice), finding that it works.

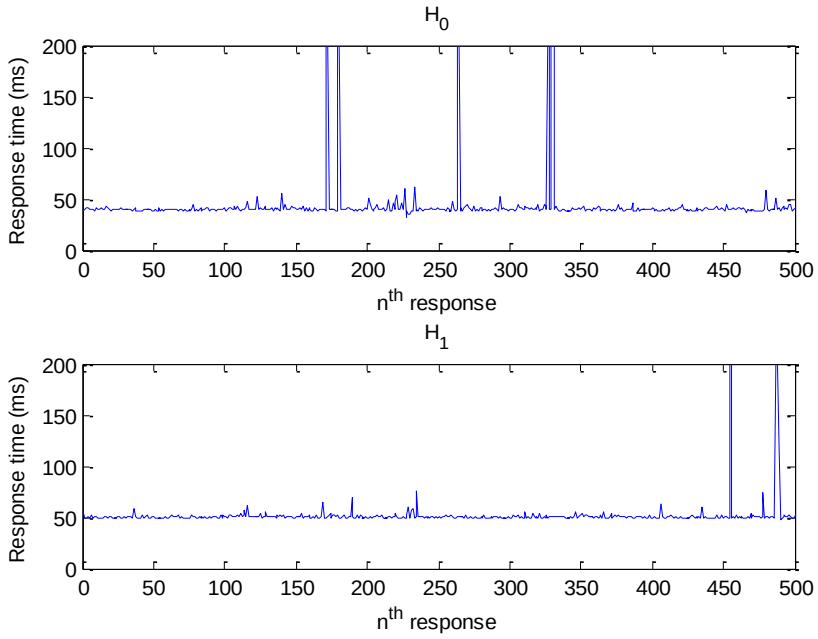


Figure 27. Detail version of Figure 25

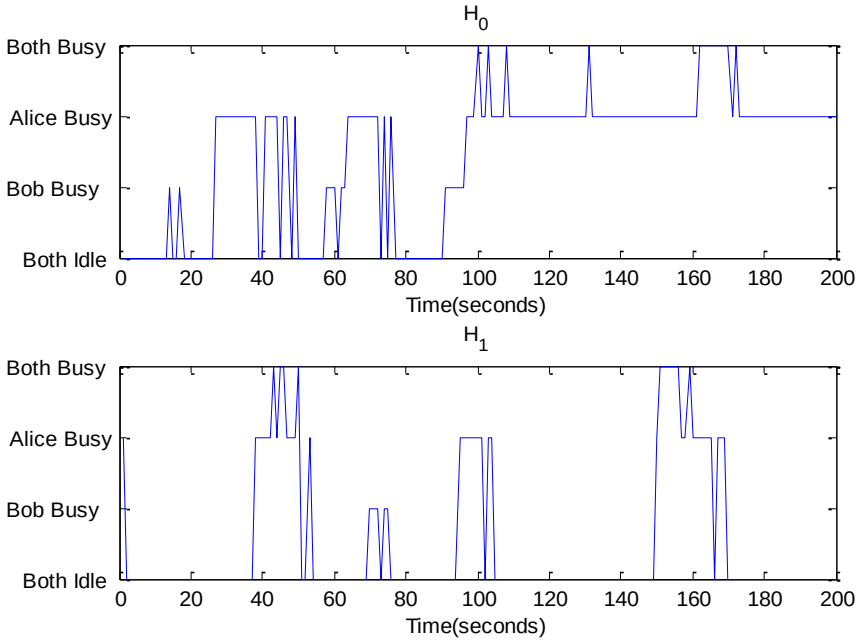


Figure 28. Two examples (H_0 and H_1) of IM time series for private communication detection

We now look at the signals of IM in Figure 28. These are two samples of H_0 (no communication between Alice and Bob) and H_1 (communication between Alice and

Bob) in the form of the busy status of Alice and Bob according to the estimated busy period. In the experiments we have samples of 300 seconds, but here we only show the first 200 seconds. Using the Cartesian product to merge the busy status of both targeted nodes, the time series will have four states. These refer to both nodes being busy, only Alice as busy, only Bob as busy and both nodes not busy, as perceived by Eve. Unlike the signals in WiFi which are quite noisy, the signals here seem simpler and there seem to be some patterns.

Through these patterns in the IM time series, we created the first version of PCD which contains only the last two stages. This idea works and then we apply it to WiFi but encounter difficulties because the WiFi signals are very noisy, such that the accuracy is very close to the baseline (50%; blind guess) or even worse than the baseline. To deal with the noise, we brought the first three stages to PCD, finding that it improves the results considerably.

3.4.2. WiFi

The accuracy of PCD and remote Sybil detection in the WiFi application is shown in Figure 29 and Figure 30, respectively. The square mark shows the average accuracy using the samples obtained in the university café for training and testing. We use one sample (10000 data points) to train the system and use the other 11 samples for testing. We again use cross-validation so that any one sample used as the training sample will produce an accuracy measure. Thus, averaging of 12 accuracy measures becomes the average accuracy for our results.

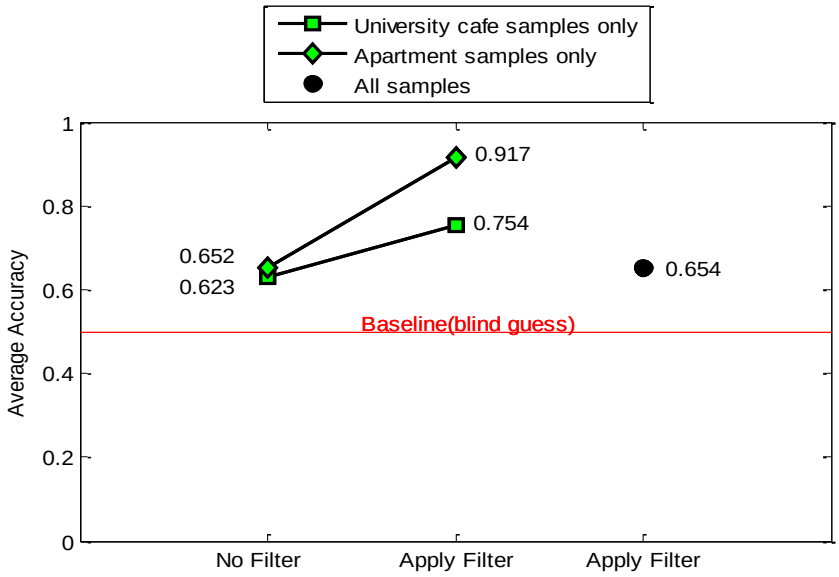


Figure 29. Performance of private communication detection in WiFi (high-pass Butterworth filter with normalized cut-off normalized frequency f=0.3, No log() applied)

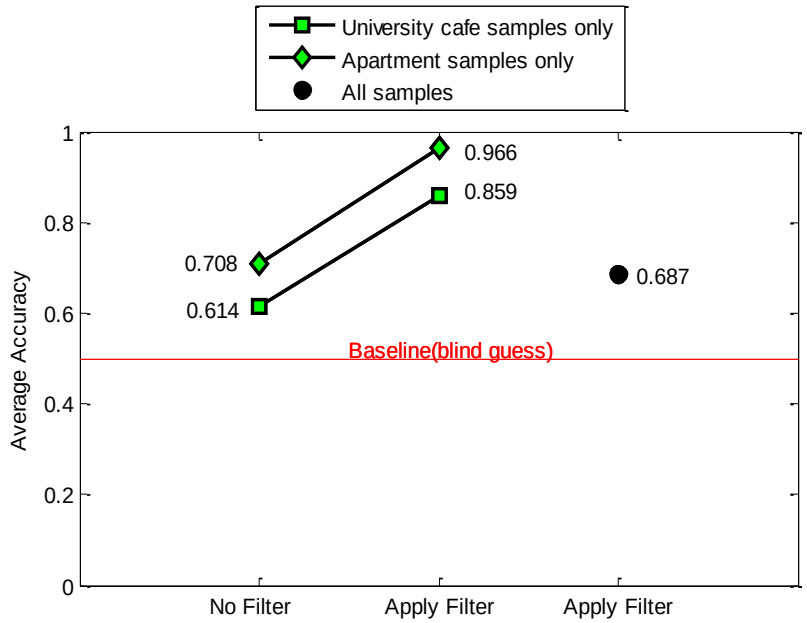


Figure 30. Performance of remote communication Sybil in WiFi (high-pass Butterworth filter with normalized cut-off normalized frequency f=0.3, log() applied before and after high-pass filtering)

The diamond marks show the average accuracy when we use the samples obtained in the apartment setting for training and testing. In both of the figures the average accuracy also rises significantly after applying filtering. In addition, we can see that the performance for the apartment samples is better than the performance for the university café samples. The reason is obvious: the university café environment is noisier (with more WiFi users) than the apartment.

We then consider the case where the system is trained by one sample but is tested on all other samples from both university café and the apartment. This is a way to evaluate the impact of environment variation to the detection. The results are showed by the circle marks in both of the figures. It is not surprising that the performance is lower because the trained models do not fit the testing samples closely. We will discuss about the environment variation in Section 3.5.

We have the following findings:

1. **High-frequency components are useful**. When high-pass filters are applied, the accuracy improves significantly. The reason is that the transmission jitter due to WiFi back-off is subtle in the time scale.

2. **Small values in signals are more important**. It is human nature to notice the spikes (large values) in a time series. However, in this application, small values are more important, as we observed that applying Log() helps to improve accuracy. The reason is that the small fluctuations contain more useful information.

In other words, PCD is successful in this WiFi application because it extracts the useful information from the small time scale (high frequency) and small values from time series. It is different from most signals where high frequency components and visually-recognizable patterns have the most useful information.

3.4.3. Instant Messaging

The number of samples in the IM condition is extremely limited. Therefore we use a version of cross-validation in which k (k<10 for private communication detection and k<20 for remote Sybil detection) samples are randomly selected to train models and the rest are used to evaluate accuracy. The average accuracy conditioned in the number of training samples is the performance of interest.

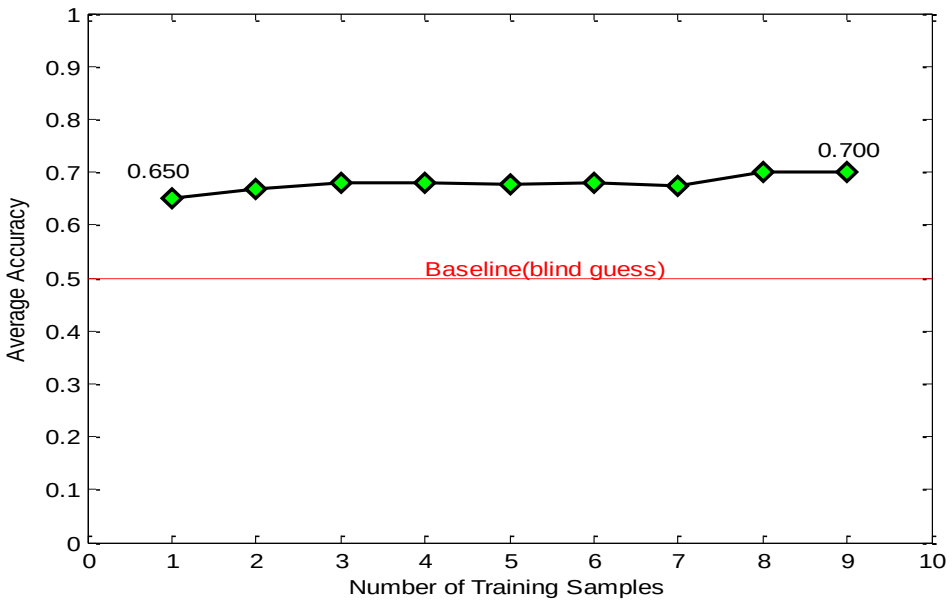


Figure 31. Performance of private communication detection in IM (total 10 samples per hypothesis)

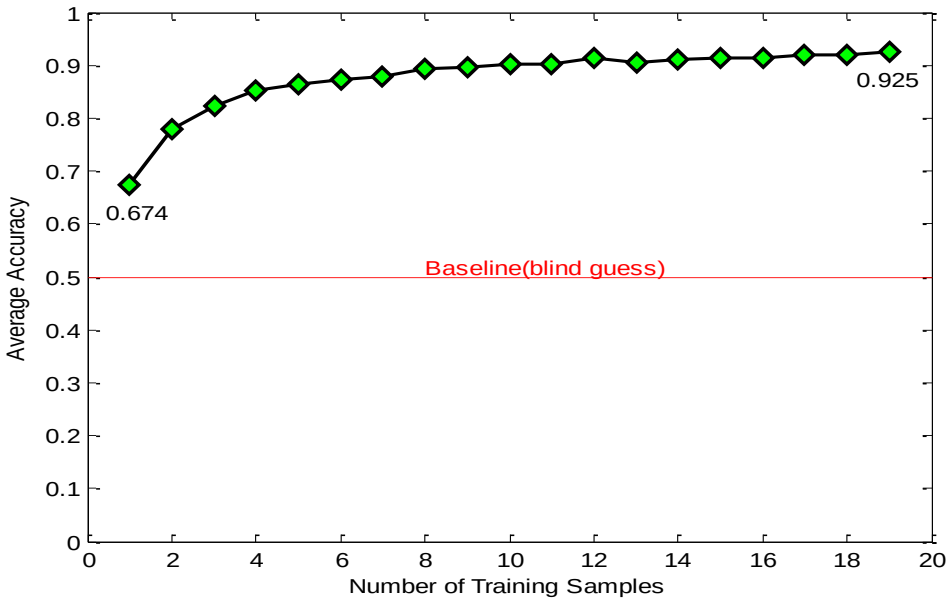


Figure 32. Performance of remote Sybil detection in IM (total 20 samples per hypothesis)

Both goals show that average accuracy is above 50% which is the baseline when a blind guess is applied instead of PCD. In addition, both goals also suggest that more samples would give higher average accuracy. For private communication detection, as shown in Figure 31, the average accuracy grows from 65.0% to 70.0% as the number of training samples increases from 1 to 9. Similarly, for remote Sybil detection, as shown in Figure 32, the average accuracy grows from 67.4% to 92.5% as number of training samples increases from 1 to 19.

Therefore we have the following findings for the Instant Messaging application. 1) PCD is achievable in IM since the average accuracy is much better than the baseline. 2) There is a trend for IM application to gain higher performance when more samples are available.

3.5. Environmental Variation

The WiFi application shows environment variation reduces the accuracy. In WiFi, both goals have lower accuracy when we use all samples from the university café and apartment for training and testing.

Does it mean that PCD can only work if Eve can obtain many training samples from the place where the targeted nodes reside? From the experiences of speech recognition, from which our architecture is inspired, it may not be true. Speech recognition initially supports only speaker-dependent recognition, meaning that the training and testing samples should come from the same source. However, later speaker-independent recognition is achieved by adjusting the variance, such as using a linear transform of existing models from many speakers [23, 37, 44]. Therefore, for PCD, once we have enough data from different network environments, similar things can be done to adapt the trained models to the variation of network environment. In other words, training may be achieved from small samples over a variety of situations, rather than just the specific targeted network environment. Also, Eve may use a probe time series to find the optimal way of adjusting trained models.

3.6. Summary

This solution to the PCD problem is a stochastic analysis that extracts non-traditional side-channel information caused by resource contention. As network jitters and delays make the side channels very noisy, our approach treats the noise with signal processing and other techniques to boost the performance. The experience of using

signal processing techniques in this side-channel attack suggests that other signal processing techniques, such as Wiener filters and Wavelet Packet Decomposition, have the potential to enhance the side-channel attacks.

Chapter 4: Service-Priority PCD

We propose the Service-Priority PCD approach that performs a PCD attack against mobile communication systems, particularly the 3G and 2G systems. Similar to the Resource-Saturation PCD approach, this approach was developed by analyzing the mobile communication protocols. The main challenge is the complexity of these mobile communication protocols. The chapter also illustrates the vulnerability of 2G phones to denial of service attacks whose perpetrators cannot be easily identified.

4.1. Overview

Our Approach. We start with a simple example where only two phones are targeted. (Generalizing this example to multiple phones poses no significant obstacle to an ordinary user attempting to collect call records.) Suppose that ordinary user Eve, illustrated in Figure 33, aims to collect the call records generated by two smartphones, which belong to users Alice and Bob. First, adversary Eve is able to surreptitiously *probe* the target phones used by Alice and Bob to discover whether they are busy; i.e., she can send data packets to the target phones over the Internet, which would remain undetected by Alice and Bob. (Various probing techniques that satisfy this requirement are discussed in Section 3.) Eve can then analyze the timing of received responses. Eve can correlate the busy-status of Alice and Bob's phones and determines whether they are communicating.

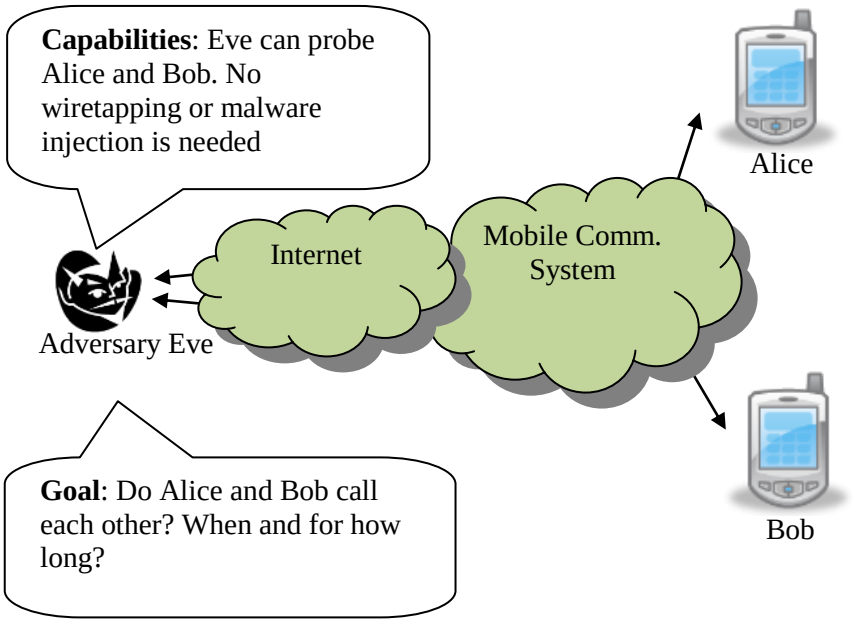


Figure 33. Adversary goal and capabilities

The key enabler for surreptitious busy-status monitoring by remote probes is the *radio resource management policy* of the 2/3G mobile communication system, which gives priority to voice over data services. If an adversary repeatedly probes the target phone via the data service, the response time spikes when the phone is busy, due to the lower priority given to the data service. This is a classic *side channel* for detecting the busy status of a smartphone: an adversary is able to determine whether a phone is busy by simple timing analysis of responses to probe messages [40]. Whenever two phones become busy during the same probing period, it is likely that the users of these two phones are having a conversation during that period.

False Detection. The false detection of calls between two targeted smartphones has only two possible sources in our given adversary setting. First, when two target phones both talk to other, non-target phones during the same period, their responses

to an adversary's probes become indistinguishable from those obtained when the phones talk to each other. A second possible source of false detection occurs if at least one of the target smartphones is accessing the Internet (e.g., Internet radio streaming or web surfing) during the probing period. Then, at least theoretically, it might be possible that its probing response could be confused with the response received when the voice service is in use.

The false detection rate for the first case depends on the probing interval, namely the shorter the interval the lower the false detection rate. For example, if Alice and Bob both have a single call during a one-hour period and the probing interval is 10 seconds, the false detection rate is 1.8% when the start time and the end time of the calls are uniform distributed random variables (viz., the Appendix I for details). The false-detection rate drops to 0.24% when the probing interval is 1 second. For all practical purposes these false-detection rates are insignificant for call-record collection. This is the case not only because these rates are extremely small but also because most call-record analyses require multiple instances of detected calls; e.g., strength-of-tie analysis requires call frequency and reciprocal initiation, and timing patterns. The more instances of detected calls in an analysis the greater the statistical confidence in the results obtained.

The false detection rate for the second case above is zero. Although it would seem possible, in practice, the response pattern of a smartphone engaged in Internet use is always very different from that obtained when voice services are used. The experimental results illustrated in Section 4.4. (viz., Figure 45 and corresponding

experiment Scenario E) confirm this general observation.

Missed Detection. Missed detection of real calls during the probing period for two targeted smartphones occurs only when the data service of at least one of the phones fails to work properly. Because the data and voice services share the same radio resources, , the use of the voice service can never be missed unless the faulty data service prevents probing; e.g., unsuccessful handover or poor radio signal. The rate of data service failure can be approximated by the typical call-drop rates since both voice and data services use the same radio facilities. Recent market research [13] suggests that these rates are fairly insignificant; i.e., the call drop rates range between 1.4% and 4.6% for four major U.S. carriers during a 90-day study, which would suggest a low false negative response rate.

4.2. Introduction to 3G Mobile Communication Systems

This section briefly introduces the 3rd generation mobile communication system (3G) with emphasis on topics related to the proposed attack. Readers who are already familiar with 3G communication systems can skip this section.

There are several approved 3G air interfaces, namely the wireless protocol suite. Among these air interfaces, in this chapter we study UMTS (Universal Mobile Telecommunication System) and GERAN (GSM EDGE Radio Access Network). The former is widely deployed and the latter is based on 2G technology. GERAN usually co-exists with UMTS and is used as a backup because of better coverage. In the rest of this chapter, we refer to 2G as the 3G system using the GERAN air interface.

4.2.1. Preliminaries

3G provides data and voice service through packet switching and circuit switching, respectively [16]. 3G is an evolving technology, meaning that it has an initial release, and provides new functionality or modification in later releases with consideration of backward compatibility. The telecommunication carrier can choose to upgrade their 3G services to newer releases for more and faster services. The first release is named after the release year (Release 99, or simply R99), and the following releases are named by integers starting from 4. In the rest of this chapter, we will discuss R99 and R6 because the former is the first release and the latter provides the faster data service HSPA (High Speed Packet Access). UMTS Release 6 or later is quite commonly deployed, as of 2012. HSPA includes downlink data service HSDPA (High Speed Download Packet Access) and uplink data service HSUPA (High Speed Upload Packet Access).

Architecture. As illustrated in Figure 34, 3G provides data and voice services through an architecture consisting of three major parts: the core network (CN), the radio access network (RAN), and the user equipment (UE). UE is a phone or computer with 3G access. UE and RAN are connected through an air interface, such as UMTS and GERAN. RAN and CN are connected via wired networks. When a UE wants to communicate with another UE or an Internet node, the UE first connects wirelessly to the RAN and then becomes accessible to the CN. Then the CN will connect the UE with a destination node. Many UEs support both 3G and 2G, which allows the use of 2G as a backup, enabling better coverage.

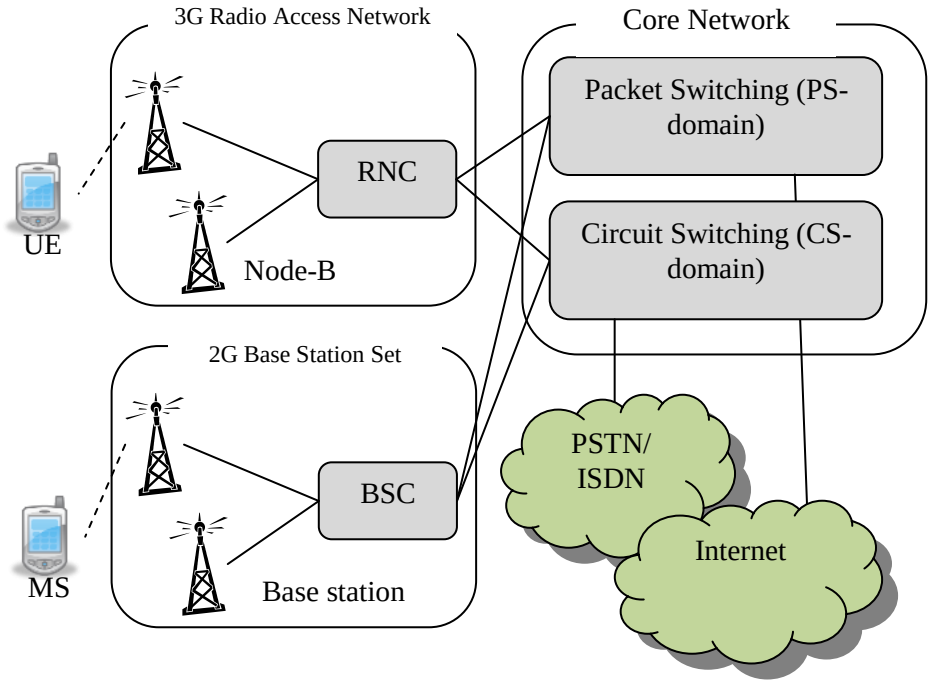


Figure 34. 3G Architecture

User Equipment. In 2G and 3G, UE has three classes. Class A has highest ability to provide circuit and packet switching simultaneously. For example, in a 2011 TV commercial, AT&T advertised that on its network the iPhone can provide simultaneous data and voice services (i.e., surfing while talking) [1]. Class B is the most common phone that can provide both circuit and packet switching, but in contrast to the iPhone, provides only one at a time. Class C can only provide circuit switching or packet switching. For example, a laptop with 3G access is of this class.

Core Network. The CN consists of a subsystem to support packet switching (PS-domain) and a subsystem to support circuit switching (CS-domain). Both of them are connected to the RAN. The RAN needs to satisfy the bandwidth demand from both circuit switching and packet switching. Therefore, RAN needs to provide radio

resource management to utilize the radio efficiently and provide quality of service (QoS), which is extremely important for the voice service.

Radio Access Network. RAN has two types of elements, RNC (Radio Network Controller) and Node-B (the base station). One RNC connects to multiple Node-B's. RNC and Node-B together perform operations for the MAC Layer (Layer 2) and Node-B solely performs the Physical Layer (Layer 1). Radio resource management is managed by RNC prior to release 5 and is co-managed by RNC and Node-B in later releases, to improve performance. Nevertheless, we can regard RAN as the component that performs the radio resource management.

4.2.2. UMTS Radio Resource Management (3G)

The mechanism of radio resource management is specified in the protocols, but the policy is open to change [9, 11, 14]. The number of users able to simultaneously access the radio channel is limited. UMTS uses a 5Mhz band for uplink and another 5Mhz band for downlink. UMTS provides multiple users access to the radio channel via CDMA and TDMA. Each 5Mhz radio channel is divided into 10ms or 2ms time slots (TDMA) and multiple users can use different codes to access the radio channel (CDMA). Although CDMA allows multiple users to use different codes to simultaneously access the radio channel, the number of codes (and the corresponding number of users) is limited.

UMTS defines several *transport channels* which transmit information. Transport channels are layered between MAC and PHY. There are different ways to classify transport channels. First, they can be categorized according to the transmission

direction: transport channels can transmit information only for uplink, only for downlink, or both uplink and downlink. Second, users employing the transport channel may use the same code (i.e., *common channel*), or different codes (i.e., *dedicated channel*). Common channels require a random access mechanism like slotted ALOHA to work, and therefore messages cannot be delivered as quickly as by dedicated channels.

Transport channels related to the proposed attack are listed in Table 9. The *Dedicated Transport Channel* (DCH) is used to deliver voice service because it is a dedicated channel which has stable throughput and supports both uplink and downlink. The *Forward Access Channel* (FACH) is used to deliver information from the network to the UE (downlink) and the *Random Access Channel* (RACH) is used to deliver information from the UE to the network (uplink). Both FACH and RACH are common channels that use less radio resources and consume less power than DCH. FACH and RACH are used to deliver information when the data rate requirement is low. The Broadcast Channel (BCH) and Paging Channel (PCH) are used to broadcast information from the network to the UE and are less relevant to our attack.

In Release 5, UMTS added *High Speed Downlink Packet Access* (HSDPA) to improve the downlink speed in packet switching. Specifically, a new transport channel named *High-Speed Downlink Shared Channel* (HS-DSCH) was introduced. In Release 6, UMTS added *High Speed Uplink Packet Access* (HSDPA) which improved the uplink speed in packet switching domain. The corresponding new transport channel is the *Enhanced Dedicated Channel* (E-DCH).

Table 9. UMTS transport channels related to the proposed attack

Transport Channel	Direction	Dedicated or Common	Use
DCH	Uplink/Downlink	Dedicated	Voice or mid-rate data
RACH	Uplink	Common	Low-rate data
FACH	Downlink	Common	Low-rate data
BCH/PCH	Downlink	Common	Sleeping mode
HS-DSCH (R5)	Downlink	Common	High-rate data
E-DCH (R6)	Uplink	Dedicated	High-rate data

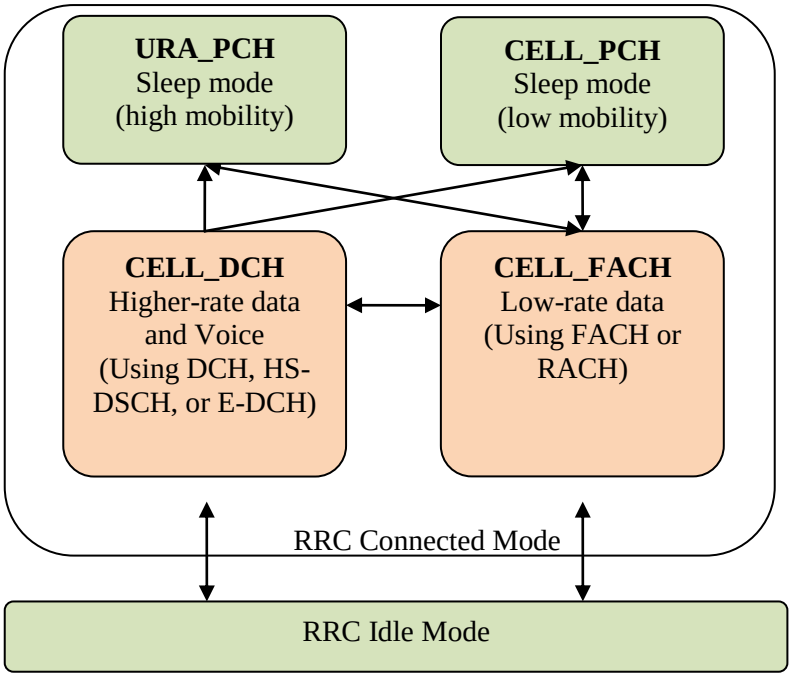


Figure 35. RRC state machine

UMTS uses the protocol Radio Resource Control (RRC) to establish, maintain and terminate the radio connection between UE and RAN. Each pair of UE and RAN maintains a state machine as shown in Figure 35. The state in which that the RRC machine is indicates which transport channels are used for transporting packets.

When a UE powers on, the initial state is *RRC Idle Mode*, which means no RRC connection is established. If a RRC connection is established, the state machine enters

the *RRC Connected Mode*, which consists of four sub-states. If there is a voice call or the demand for data service is high, the state machine will enter the CELL_DCH state. In this state, transport channels supporting high speed transmission, such as DCH or HS-DSCH, are used. If the demand for data service is low, the state machine will enter the CELL_FACH state. In CELL_FACH, common transport channels such as FACH and RACH are used because they use less radio resources and save power.

When there is no data or voice service demand, the state machine may enter the CELL_PCH or URA_PCH state. Both of these are called sleep mode, because the power consumption is low without active Internet access or voice calls. The difference between CELL_PCH and URA_PCH is the mobility. In CELL_PCH, the system maintains a more accurate location of the UE but these two states are not of interest in our experiment.

4.2.3. GERAN Radio Resource Management (2G)

GERAN consists of GSM and GPRS technologies. GSM is a 2G technology that provides data and voice services via circuit switching [67]. The data service in circuit switching is slow because circuit switching is not as efficient as packet switching. To increase the data rates and improve radio efficiency, GPRS was introduced as an extension to GSM. GPRS provides data service through packet switching, but only requires a modified BSC (Base Station Controller), which is the GSM version of RNC, and a new packet switching subsystem (PS-domain) [38, 48]. The minimal design change saves the prior investments on the GSM network. GSM uses TDMA and FDMA such during a particular time slot, multiple nodes can access the radio simultaneously through different frequencies.

The radio resource is completely controlled by the Base Station Set (BSS), which is the equivalent or close-to-equivalent component of RAN in terms of 3G. Therefore, BSS knows exactly which slots are not used. GPRS can ask BSS to assign these unused slots to it and provide data service via packet switching. In this case, voice service (GSM) has absolute priority over data service (GPRS), because GPRS can only use the slots that GSM does not fill. On the other hand, GPRS may suffer from low service quality. Later, an enhanced version was proposed, to allow BSS to reserve some slots for GPRS. For example, each frequency in GSM has 8 time slots and one or two may be reserved for GPRS. This version provides starvation prevention for GPRS, but GPRS still has lower priority than GSM.

If accessing GPRS and GSM simultaneously (i.e., surfing while talking) is desired, the UE either needs to be class A or to support Dual Transfer Mode (DTM) [59]. DTM is a feature starting from R99. When data service is ongoing and a voice call comes in, data service is disconnected and then voice service is established. Finally, the phone uses DTM to re-establish the data connection. A class A GSM/GPRS phone (not DTM) implies that two sets of hardware (e.g., transceivers) are needed and therefore the design is not favored by phone vendors for the additional cost.

4.3. The Attack

Since the proposed attack is based on remote probing for busy-status detection, we first will describe the possible ways for the adversary to probe the target phones. Then we will show how and why the probing response time reveals the busy status of smartphones. Finally, we will present the experiment design.

4.3.1. Probing

Probing a smartphone is not as easy as probing an Internet server with ICMP, UDP or TCP. There are two concerns in the environment of mobile communications. First, mobility may make the phone IP address dynamic. It is impossible to probe a smartphone in the network or transport layer (IP/TCP/UDP) without knowing its IP address. Second, if the assigned IP address is private (i.e., certain IP addresses which are not routable), the traffic back and forth to the smartphone is processed by a Network Address Translation (NAT) device. NAT devices usually deny inbound IP/UDP/TCP packets if there is no previously associated connection established by the smartphone. For this reason, NAT devices actually act as a firewall to prevent IP/TCP/UDP probing.

However, some apps running on smartphones make it possible to probe smartphones in the application layer. For example, VoIP and Instant Messaging apps need to run in the background and receive paging or messages. They may not listen to a port to accept a message from the outside, but they can establish a TCP/UDP session to a server that relays messages between smartphones and other VoIP/Instant Messaging users. For example, *Session Initiation Protocol* (SIP) is a widely-used protocol for VoIP and Instant Messaging [31]. A user Eve can send a SIP OPTIONS request message over TCP or UDP to the SIP server and the message will be forwarded to the destination user Alice. Alice will send back a corresponding SIP OPTIONS response message to Eve via the SIP server. Such probing can be launched by our Voice Pulser program as described in Chapter 2. Other apps may provide probing as well if the

adversary can find a way to utilize the communication channel between the apps and the server.

In addition to these methods, there are at least two other ways to probe the phones. When the adversary has a Private Branch Exchange (PBX) facility that can exchange telecommunication signals with the carrier of the target phone, the adversary could probe the target phone using signaling protocols (e.g., SIP). Moreover, when the phone is continuously using streaming service, such as Internet radio, the adversary, acting as the streaming service provider, can use the streaming traffic to obtain the timing of messages. The latter method can be regarded as a passive means of probing, because although streaming is logically a one-way transmission from the streaming server to the phone, the phone still needs to provide feedback (i.e., ACK) to the streaming server for flow control. We use the similar *passive* probing in the Instant Messaging application of the Stochastic PCD approach.

4.3.2. Busy-Status Detection

The radio resource management in both 3G and 2G requires that the voice service has priority over the data service. In UMTS, a voice call will force the RRC state machine to enter the CELL_DCH state where dedicated transport channels will be used. In contrast, a data transmission may enter the CELL_DCH state only if the data rate is high enough, instead entering CELL_FACH for lower data rates. In the Dual Transfer Mode (DTM) mode with GSM/GPRS, when the data service is continuously provided and a voice call comes in, the data service is first disconnected and then attached to the dedicated channel for the voice service. In that case, the data service may use the

voice channel where the voice service only runs at half rate.

These state differences matter because if the adversary can estimate the priority of the data service, she can detect if there is a call underway with the target phone. More specifically, when a UE is using data service, an incoming or outgoing voice call would likely lower the priority of this data service. The priority of the service is actually the state of the state machine inside the radio resource management (e.g., UMTS RRC state machine). Therefore, detecting whether a phone is busy is reduced to the problem of determining the state of state machine in the radio resource management.

Identification of the radio resource management state (viz., Figure 35) by probing the target phone is possible because different channels are assigned in different radio resource management states [5]. In UMTS, if the RRC state machine moves from CELL_FACH to CELL_DCH state, the traffic transmission time is reduced because of the change of the corresponding transport channels. The transport channels used in CELL_FACH are common channels, i.e., multiple users share the same code to access the channel. Hence, transmission time is influenced by other UE under a random access method. On the other hand, voice service is carried by a DCH dedicated transport channel in the CELL_DCH state. With a dedicated transport channel, the UE is able to transmit without the delay brought by the random access method.

This means that detection of the busy status of a phone can easily be done by probing the target phone; delayed (queued) response or lost responses would indicate that the

phone is busy at that moment. After obtaining the time series of probe responses from the target phones, the adversary can infer the a call takes place between the targeted phones based on the following principle: If two phones become busy during the same period, they are detected as having a call with each other.

As explained in the Introduction, the false detection rate depends on the time resolution of the busy-status detection, with coarser time-resolution leading to a higher false detection rate. (See Appendix I for the details of calculation.) For our experiments, we chose a probing rate that makes false detection of calls negligible.

4.4. The Attack Experiments

4.4.1. Experiment design

We evaluate the proposed attack in a real-world environment as shown in Figure 36. Two phones are equipped with a VoIP app which can use 3G, 2G, or WiFi to connect to a public VoIP server. An attack program Voice Pulser running on a university server will send SIP OPTIONS probe messages to the two phones via the VoIP server. This setting, aside from the attack program, is essentially a real-world environment of smartphone use.

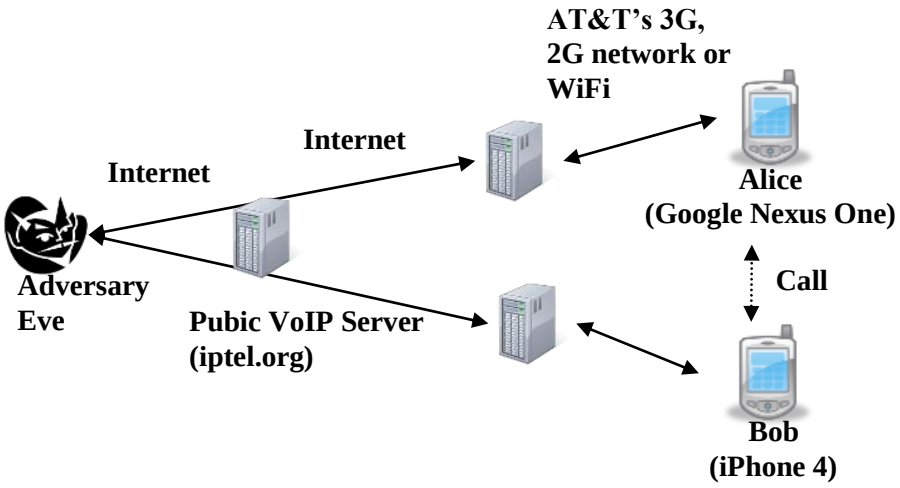


Figure 36. Experiment environment

The experiment equipment and environment are listed in Table 10. The attack target phones are an iPhone 4 and a Google Nexus One (android 2.1). These two phones are connected to the Internet by AT&T's 3G network, 2G network or WiFi. The attack program sends VoIP requests (SIP OPTIONS) to the two phones and records the time of the responses. The VoIP client app is installed in both phones. Both of the phones have set up a VoIP account from the VoIP server. The VoIP app will periodically connect to the VoIP server for updating information and for keeping a communication channel open between the phone and the VoIP server.

Table 10. Experiment equipment/environment

Equipment	Parameters
Alice	Google Nexus One (hTC G1) android 2.1; 3G Rel. 6 (HSPA)
Bob	iPhone 4 iOS 5.0; 3G Rel. 6 (HSPA)
Mobile Network	AT&T (UMTS/GPRS/GSM) 3G Rel. 8 or later
VoIP App	Bria by Counterpath Corp.
VoIP Server	iptel.org, a public/free SIP service
Attack Program	Probing by SIP OPTIONS

In order to test different mobile network configurations, the experiments are performed in one of four locations, as shown in Table 11. Two locations are in a major city and the other two are in the suburban area of that city. These locations are miles or tens of miles away from each other to ensure that the phones will connect to different base stations.

Table 11. Locations for experiments

Location	Type	Description
1	Urban	A Carnegie Mellon University building in Pittsburgh
2	Suburban	A mall (Waterfront) outside Pittsburgh
3	Suburban	Monroeville, a satellite city of Pittsburgh
4	Urban	Highland park in Pittsburgh

As shown in page 107, we design six scenarios to evaluate the effectiveness of the proposed attack. The specific question and steps of each scenario are listed below. Scenarios A to D test whether the proposed attack is effective in different environments. Scenario E tests whether the proposed attack is affected if the phones are using the Internet. Scenario F tests whether the adversary can detect whether the target phones are in the 3G network, instead of WiFi, such that the proposed attack is feasible.

The following scenarios, except Scenario B, were performed in Location 1. In all scenarios, except Scenarios E and F, the experiment is done during a 5-minute session. The probing interval is 1 second, except in Scenario D and F.

In Scenario A, we examine whether the attack is effective in 3G. Each session lasts 300 seconds and begins with probing at 1 second intervals. Each session assigns

different roles to the phones. A phone can be either the caller, or the callee. In closing the call, a phone can be either the one which hangs up the call, or which receives a hang-up signal.

In Scenario B, we examine whether the attack is effective when the phones connect to different base stations. The reason is that the carrier may decide to use different parameters for radio resource management in different locations.

In Scenario C and D, we examine whether the attack is effective in 2G. The two scenarios are different with respect to the probing interval. Scenario C has 1-second intervals whereas Scenario D has 8-second intervals. The reason for using the larger, 8-second, interval is that higher rate probing causes denial of service in 2G phones (i.e., Scenario C) as described in Section 4.4.2.

In Scenario E, we examine the false detection caused by the smartphone app's Internet access. As a call might cause a response time spike, would Internet usage result in the same or a similar response patterns as with voice calls? To answer this question, we perform three 1-hour test sessions. In the first session, there is no Internet access for foreground apps, in order to set a baseline condition. In the second session, the iPhone calls the android phone every 5 minutes and terminates after 1 minute. In the third session, both phones are listening to Internet radio streaming, which has a constant rate traffic of about 64Kbps. There are no calls or other foreground app activities during the third session.

The last Scenario F, we test whether an adversary can distinguish if the target phones are in 3G or WiFi. If the target phone is in WiFi, the proposed attack is neither

feasible nor of interest to our experiments. However, the Stochastic PCD can still be applied to WiFi as described in Chapter 3.

Scenario A: Does the attack work in 3G?

1. Turn on 3G and turn off WiFi
2. Start to probe the both phone (1 second interval)
3. At 120 seconds, one phone calls the other (note that dialing needs time)
4. After 60 seconds, one phone terminates the call
5. Keep probing for 120 seconds
6. The above steps are performed repeatedly using different roles for the phones (which phone makes the call; which phone terminates the call)

Scenario B: Does the attack work in 3G in different locations (with different base stations)?

1. Go to one of the *four locations*
2. Turn on 3G in both phones and turn off WiFi
3. Start to probe both phones (1 second intervals)
4. At 120 seconds, one phone calls the other (note that dialing needs time)
5. After 60 seconds, one phone terminates the call
6. Keep probing for 120 seconds

Scenario C: Does the attack work in 2G?

1. Turn off 3G and WiFi and *use 2G*
2. Start to probe both phones (1 second intervals)
3. At 120 seconds, one phone calls the other (note that dialing needs time)
4. After 60 seconds, one phone terminates the call
5. Keep probing for 120 seconds

Scenario D: Does the attack work in 2G with longer probing interval?

1. Follow the same steps in scenario C except use an *8 second interval*

Scenario E: Is the attack affected when the phones are accessing the Internet?

1. Turn on 3G and turn off WiFi
2. Start to probe both phones (1 second interval) for 1hour
3. Perform one of the following actions
    - *No action*
    - Every 5 minutes, Alice *calls* Bob and then Alice terminates after 1 minute
    - Two phones both listen to *Internet radio*

Scenario F: Can the adversary distinguish if the phones are in 3G or in WiFi?

1. Turn on *3G* and turn off WiFi
2. Start to probe both phones (20 second interval) for *2000 seconds* (100 probes per phone)
3. Turn off 3G and turn on *WiFi*
4. Start to probe both phones (20 second interval) for *2000 seconds* (100 probes per phone)

4.4.2. Experiment Results

The results of the experiments are summarized in Table 12. In brief, the attack is effective. The attack can be performed in 3G and 2G and it will not be affected when the phones are using the Internet. The adversary can distinguish whether the target phones are in 3G or WiFi.

Table 12. Summary of experiment results

Scenario	Question	Answer
A	Does the attack work in 3G?	Yes
B	Does the attack work in 3G in different locations (with different base stations)?	Yes
C	Does the attack work in 2G (1 second probing interval)?	No*
D	Does the attack work in 2G (8-second probing interval)?	Yes
E	Is the attack affected when the phones are accessing the Internet?	No.
F	Can the adversary distinguish if the phones are in 3G or in WiFi?	Yes.

*:The phone's voice service is disabled (no rings for incoming or outgoing calls)

The results of Scenario A are shown in Figures 31-34. The four figures show the probing results in the different assigned roles. All four figures suggest that there is a response time spike when there is a call between the two phones. The pattern is a triangle or trapezoid because of the priority change of the data service. When a call is established, the priority of the data service is lowered but the packets will still be delivered. This results in the response time increasing. However, when a SIP VoIP message cannot get a response in 30 seconds, the VoIP server that relays the messages will respond with a "408 Request Timeout" [31]. Therefore, we would see a 30 second cap in the response time. Finally, when the call is over, all queued probe messages are resumed. The response time shows a decreasing trend because these probe messages are sent at different times but received all at about the same time.

It is likely, but we will need to perform further testing, that an adversary could also distinguish between the callee and the caller. In Figure 37 and Figure 38 where the iPhone is the caller, the response time spike happened a little earlier in the iPhone than in the android phone. In Figure 39 and Figure 40, where the android phone is the caller, the response-time spike of the iPhone and android are much closer. (Again, whether the caller role is distinguishable requires more experiment data to support this assertion.)

In Figure 38, the android phone shows a small spike around time t=50 seconds. The reason is that packets are lost during that period but retransmitted by the VoIP server.

The results of Scenarios B are shown in Figure 37[12], Figure 41, Figure 42, and Figure 43. These four figures show that the proposed attack is effective in the four locations where different base stations are used because the response time spike patterns of the iPhone and android do match. In Figure 41 and Figure 43 we can see there are small increases, stemming from the packets lost and retransmission.

The testing performed for Scenario C actually failed. When the iPhone called the android phone, the android phone did not ring. When the android phone called the iPhone, the iPhone did not ring either. This is actually a denial of service attack. We believe it is a design or implementation flaw of 2G that could be readily exploited by an adversary. However, the next scenario shows that the proposed attack is still effective whenever the probing interval is large enough.

The results of Scenario D are shown in Figure 44. The data points are less dense because of the long probing interval. The response time spike of the android and iPhone here are matched with a time shift. With the iPhone as the caller and as the call terminating phone the response time spikes and then falls back earlier than the android phone. By considering the shift and using a long enough probing interval, the proposed attack is still effective in 2G.

The results of Scenario E are shown in Figure 45. The upper subfigure shows the response time when the two phones have no calls nor Internet access by foreground apps. The middle subfigure shows the response time when the iPhone calls the

[12] Since Scenario B in place 1 is the same with Scenario A when iPhone calls and iPhone terminates, we did one experiment and used the data for both cases.

android phone every 5 minutes. The lower subfigure shows the response time when the two phones are listening to the Internet radio streaming. For the first case, the response time is almost flat, except that the android phone has a 30-second response time between time t=1500 seconds and t=2000 seconds. (The reason is that the android lost Internet access during that period and the VoIP server responded "408 Request Timeout" instead.) Note that this pattern is different from the pattern of the calls. Therefore, the adversary could easily distinguish the pattern of calls from the pattern of lack of Internet access.

The results of the final Scenario F are shown in Figure 46. The upper subfigure shows the response time when the phones uses 3G and the lower subfigure shows the response time when the phone uses WiFi. The response time in both cases are quite visually different. The response time in 3G ranges between 0 to 1 seconds while the response time in WiFi rates below 0.4 seconds. The adversary may apply classification techniques, such as the Support Vector Machine (SVM) or Artificial Neural Networks, to detect if the target phone is in 3G by one or more probes.

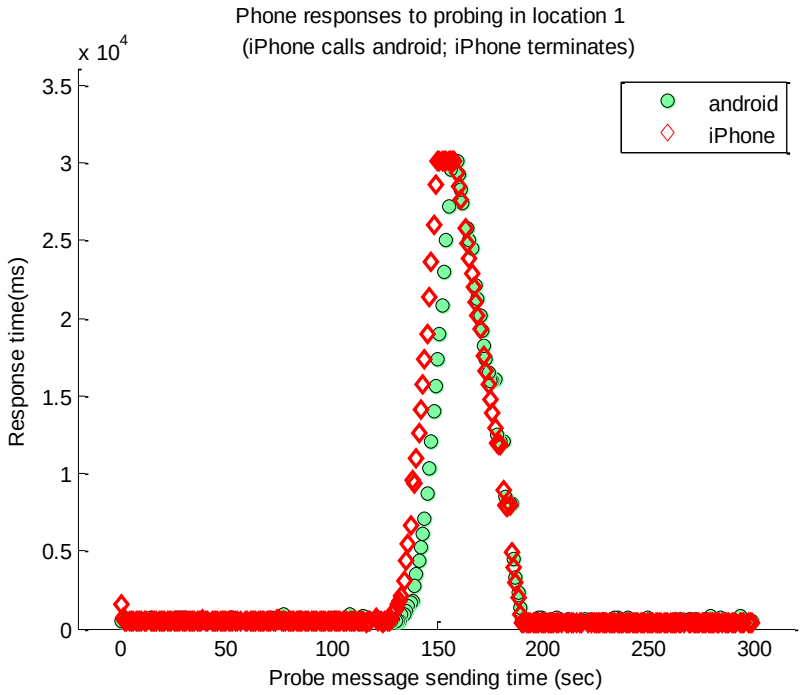


Figure 37. iPhone calls android and iPhone terminates the call in location 1 (Scenarios A, B)

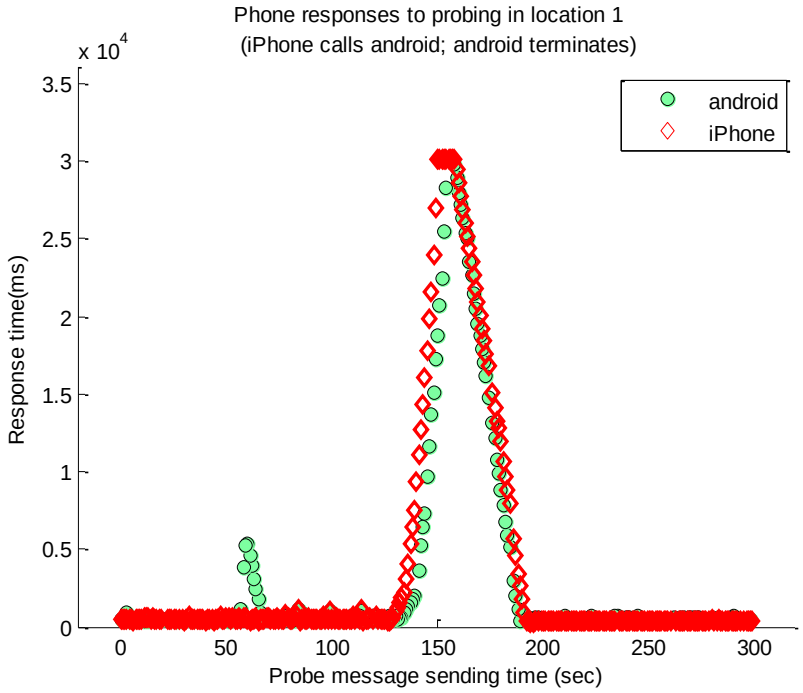


Figure 38. iPhone calls android and android terminates the call in location 1 (Scenario A)

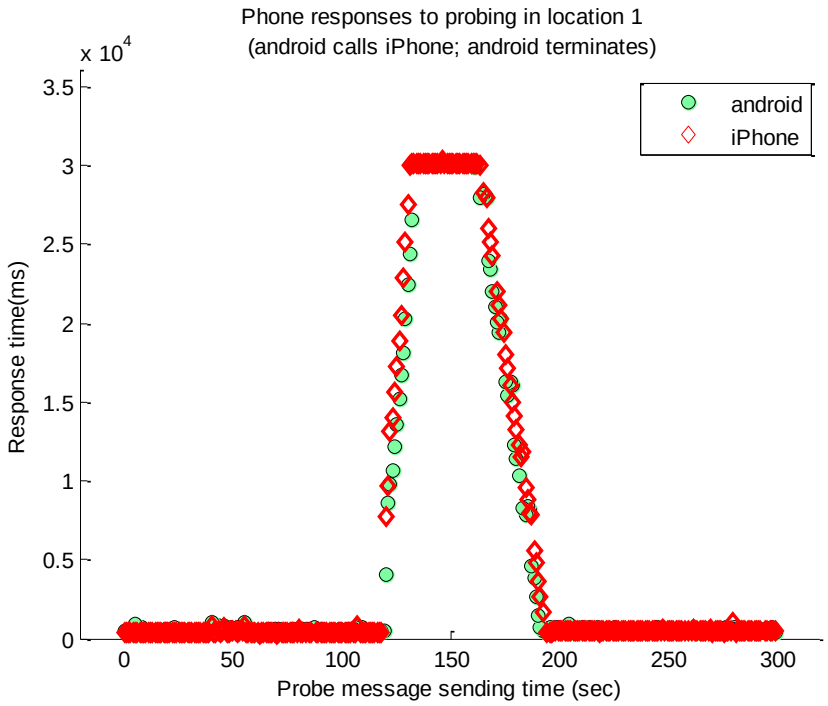


Figure 39. Android calls iPhone and android terminates the call in location 1 (Scenario A)

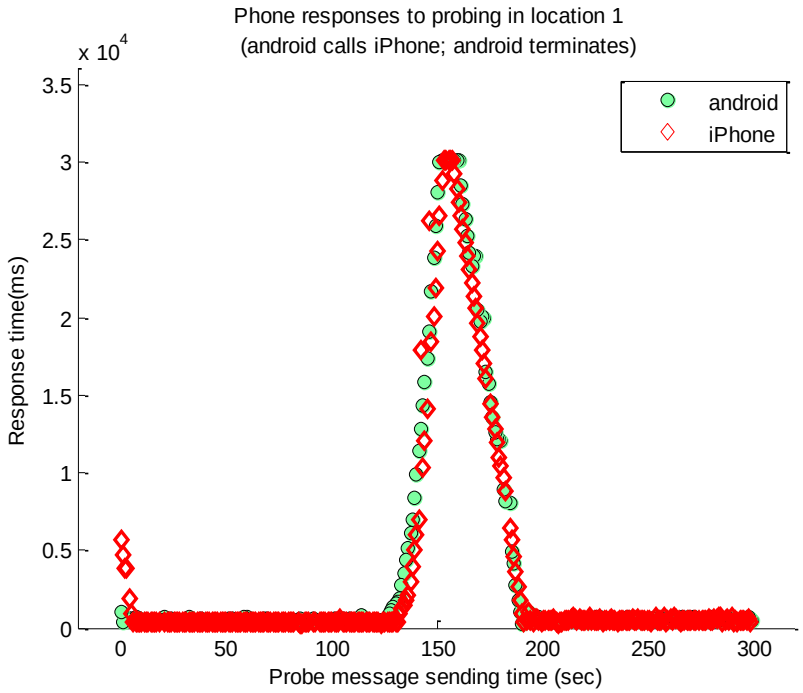


Figure 40. Android calls iPhone and iPhone terminates the call in location 1 (Scenario A)

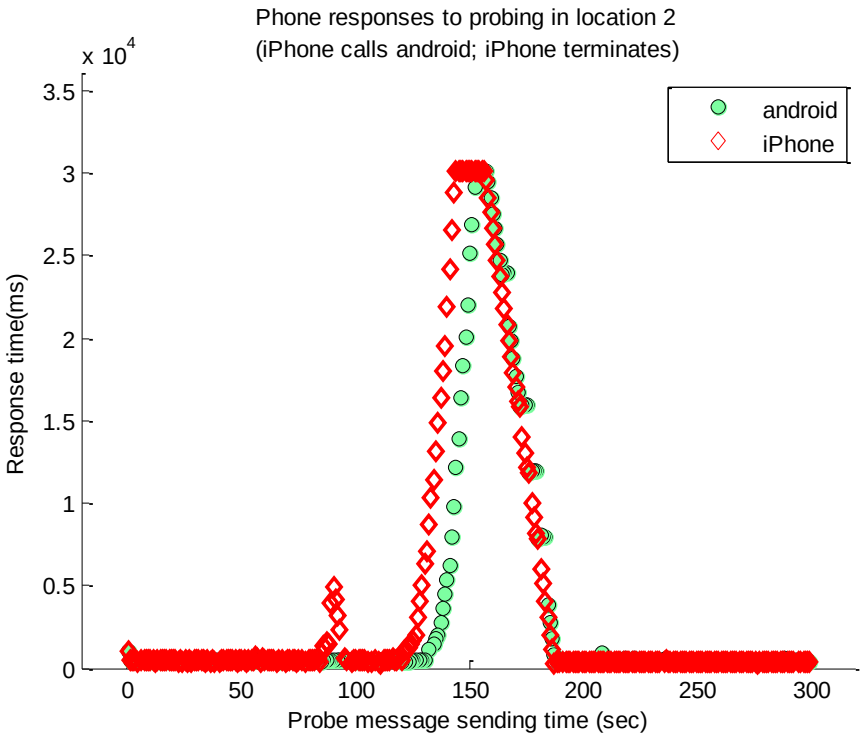


Figure 41. 3G attack in location 2 (Scenarios B)

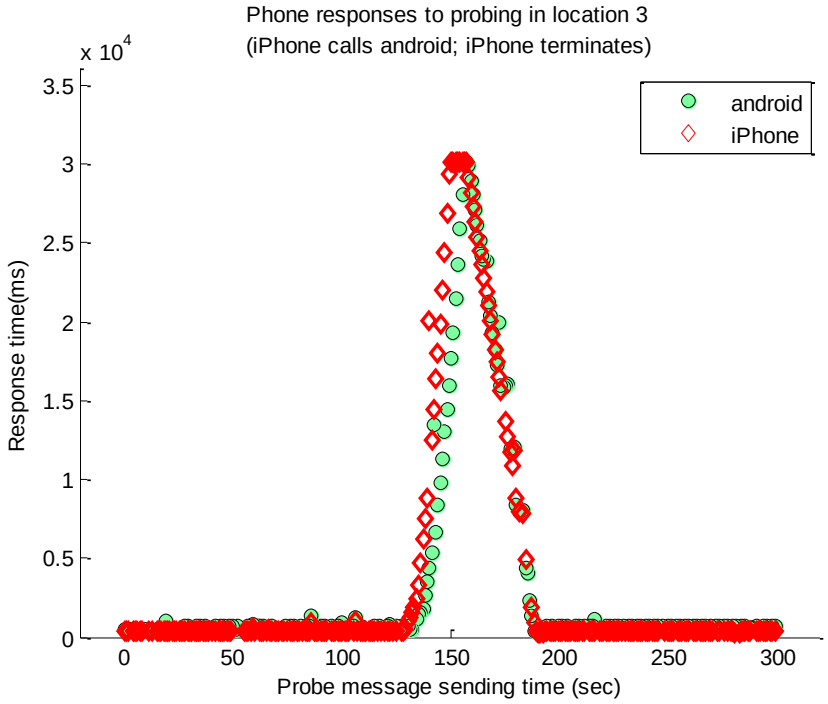


Figure 42. 3G attack in location 3 (Scenarios B)

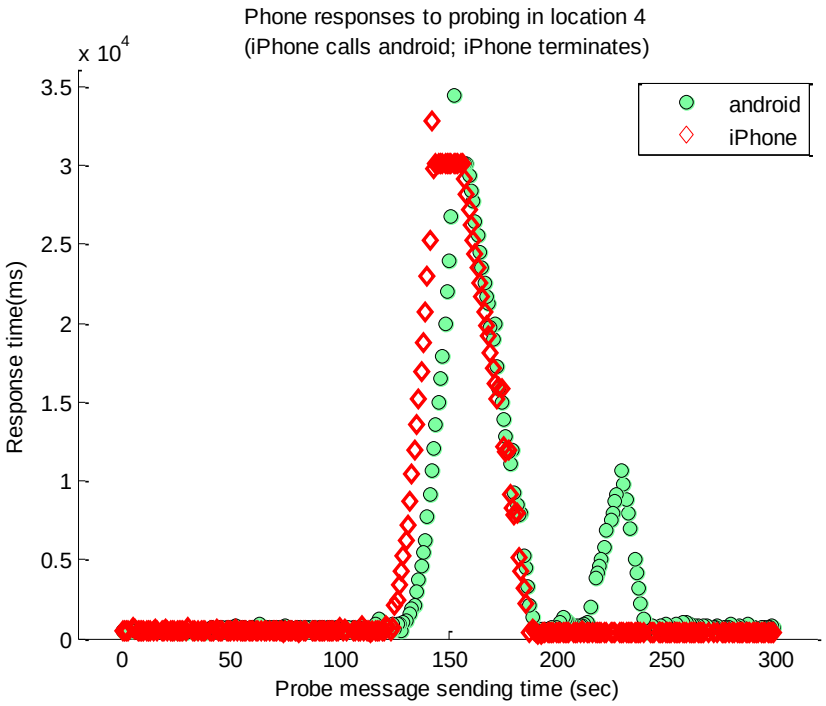


Figure 43. 3G attack in location 4 (Scenarios B)

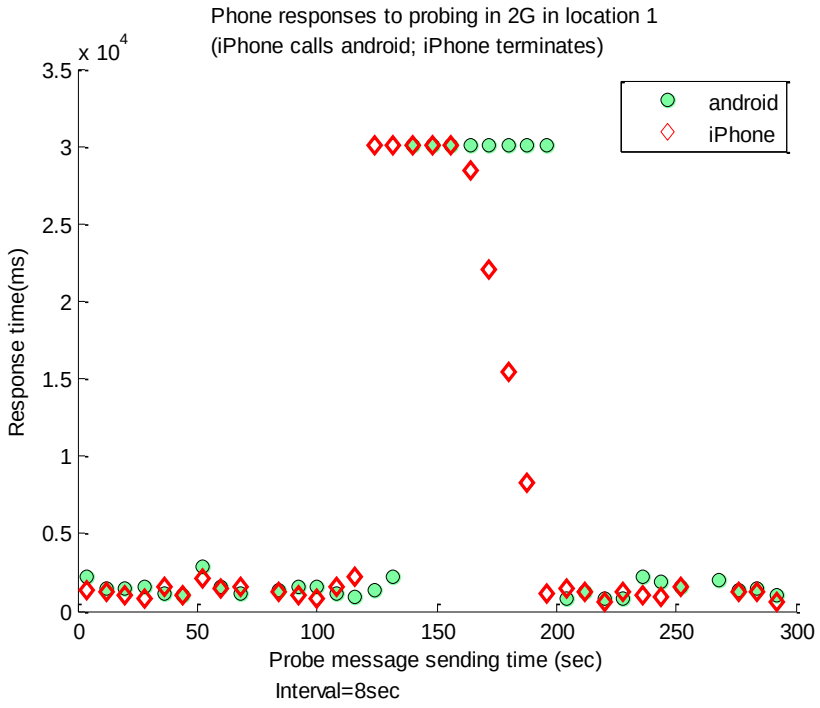


Figure 44. 2G attack in location 1 (Scenario D)

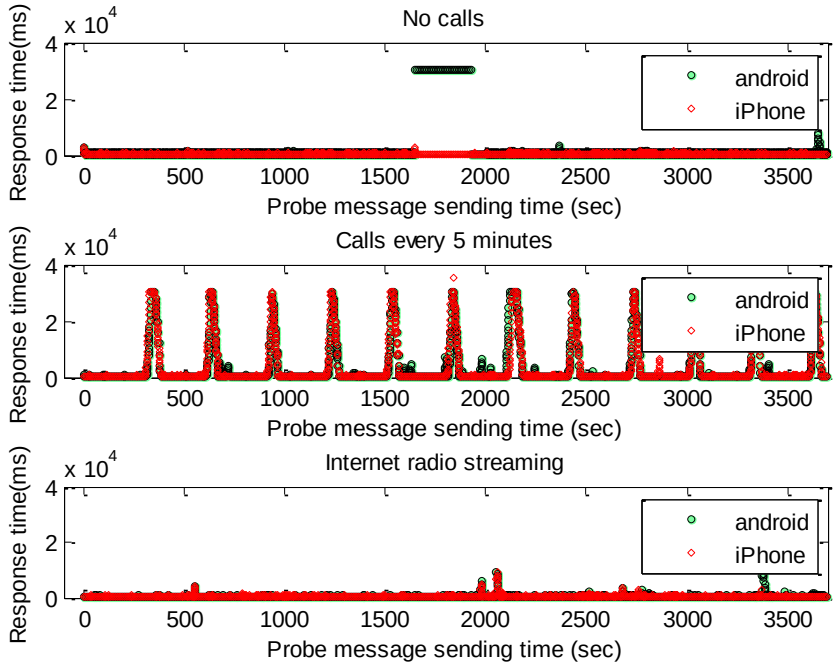


Figure 45. Probing responses with no calls (upper figure), with calls every 5 minutes (middle figure), and with phones using Internet radio streaming (Scenario E)

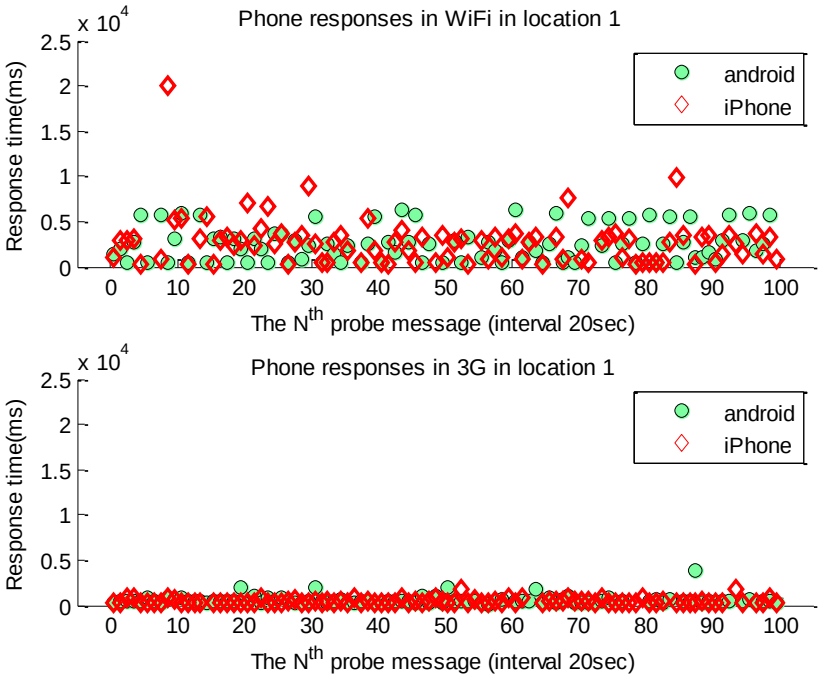


Figure 46. Probing responses in 3G and WiFi (Scenario F)

4.5. Summary

This chapter describes a new and powerful privacy attack against smartphones by exploiting a fundamental design goal of mobile communication systems, namely to make radio resource management efficient by prioritizing traffic. This leads to a side channel, which an adversary can exploit to obtain private call record information. By probing the mobile phones via their data service interface, the adversary is able to distinguish whether the target phone is busy and to infer the occurrence of a call via correlation of phone busy status.

Chapter 5: Countermeasures

We will discuss general countermeasures against side-channel attacks and then the application-specific countermeasures for the four aforementioned PCD applications.

5.1. Countermeasures against Side-Channel Attacks

As summarized in Table 13, the three traditional ways to prevent side-channel attacks (which are not PCD-specific) are 1) blocking the side-channels, 2) adding noise to the side-channels, and 3) removing the source of side-channels. For the first one, since PCD is based on repeated probing, blocking such probing can block PCD. For example, a network end-node may use a firewall that blocks the suspicious periodic packets. Since probing can be performed in different layers (e.g., mobile phone probing can be done in the application layer or in the telecommunication signaling layer), the firewall should take care of packets in different layers. However, this countermeasure can only work when PCD is performed by active probing, namely sending probing packets explicitly, instead of analyzing existing communication such as the Instant Messaging application of the Stochastic PCD. Moreover, if blocking is done according to some packet signatures (e.g., SIP INVITE messages with an invalid IP address), attack packets which are encrypted or without known signatures will not be identified.

This first method is more viable than the second and third countermeasures. For the second countermeasure, adding noise, we discovered that signal-processing techniques can be use to eliminate the noise in the response time series, so this

countermeasure is not preferred unless the type and the magnitude of added noise is guaranteed to effectively prevent the Stochastic PCD. It would be desirable to see future work that would determine the necessary and sufficient conditions for adding noise effectively to side channels. The last countermeasure suggests that removing the resource contention can eliminate the side-channel information. However, removing resource contention may imply unlimited resource usage, which is not usually practical.

Table 13. General countermeasures to PCD

	Blocking side channels	Adding noise to side channels	Removing the source of side channels
Meaning in PCD	Monitoring and blocking suspicious periodical network traffic	Adding jitters to probing responses	Removing the resource contention
Achievability	Yes (subject to application constraints; not for passive probing)	Unknown (Signal-processing techniques may eliminate added noise)	Difficult (no resource contention implies unlimited resource)

Table 14. Application-specific PCD countermeasures

		Blocking side channels	Adding noise to side channels	Removing the source of side channels	Other
Resource-Saturation PCD (VoIP)		Applying the general countermeasure	Randomizing the protocol buffer size or SIP transaction timer		Detecting full protocol-buffer
Stochastic PCD	WiFi	Applying the general countermeasure	Adding jitters to probe responses (may not be feasible)		
	Instant Messaging				
Service-Priority PCD (Smartphones)		Applying the general countermeasure			Using data service on demand

5.2. Application-Specific PCD Countermeasures

As summarized in Table 14, we show the application-specific PCD countermeasures in the following subsections.

5.2.1. Resource-Saturation PCD

Randomization is a classic, if somewhat impractical, technique for defending against both covert- and side-channel attacks. In our case, phone firmware can be modified to provide randomization in the available size of protocol buffer or the transaction life. A limitation of this technique is that it may be incompatible with the manufacture's firmware updates. A simpler countermeasure, though not a permanent solution, is for the VoIP administrator to use Voice Pulser to send "attack" packets at random intervals to the phones. This would introduce "noise" and reduce the adversary's ability to estimate the number of available protocol buffer slots. To maintain the phones' operability, the "attack" packets should be sent slowly enough that they would not fill the protocol buffer. This approach is scalable in the sense that one installation could protect many phones. One limitation of this approach is the availability of the server hosting Voice Pulser. If the server is unavailable, call privacy protection will also be unavailable.

Another practical countermeasure is alerting the user when the protocol buffer is full or the SIP message rate is too high. This constitutes a simple way to remove the cover for otherwise undetectable busy-status probes. The SIP phone or the server could simply display a message on the screen, leave a voice message, or send an email as a

way to alert the user of a potential attack.

5.2.2. Stochastic PCD

We do not have a practical countermeasure that would work specifically for the Stochastic PCD. Adding noise to the side channels is one possible remedy for this attack, but it requires that we consider where to perform this and what noise to add. In the WiFi application, if noise is added in the WiFi network, we need to modify the MAC protocol, which is very undesirable. Noise added in the application proxy is a possible solution, but the WiFi user would likely need to convince the application proxy's administrator or designers.

Removing resource contention is difficult in WiFi. In the WiFi application, if CSMA/CA is replaced by TDMA or CDMA, then there is no radio-channel contention. However, replacing a radio technology is often difficult due to non-technical reasons, such as deployment and costs.

In the IM application, one trivial countermeasure is to disable or block the composing notification messages. However, this approach may not work because ordinary messages still provide information about the timing of the messages being sent. Eve can estimate the period that the contact spends typing to the investigator with a constant value, such as 3 seconds. In addition, adding artificial jitters to messages may break the semantics of Instant Messaging, where the goal is that *instant* messaging should be provided.

5.2.3. Service-Priority PCD

Possible defenses against the Service-Priority PCD are not technically difficult to implement. One trivial countermeasure is to use a phone's data service only when needed. For example, a user can simply turn off the data service and restore it only when certain apps need Internet access. This approach is simple and requires no modification of either software or hardware. It is already a common practice for users who are conscious of high fees when roaming abroad or exceeding data caps (e.g., Using 301 MB or more in AT&T's 300MB/month plan would cause a double data charge). However, the disadvantage is that it destroys most of the convenience of a mobile smartphone, since it requires the user to deliberately turn Internet access on and off, and makes continuous-access applications like VoIP/Instant Messaging impractical.

A better approach is to use a firewall embedded in the mobile phone, one that can be used to detect periodical traffic, and only allowing such traffic to proceed with the user's authorization. The disadvantage of this approach is that it requires modification of the mobile phone operating system. The advantage is that once the firewall is provided, it applies to all apps. However, like other anomaly detection systems, there may be false detection of "suspicious periodic network traffic". Therefore, the firewall should give end-user the ability to grant or deny access for each app.

Chapter 6: Conclusion and Future Directions

In exploiting a novel side channel caused by resource contention, this dissertation constructed a powerful new attack, private communication detection (PCD), which allows an ordinary network user to obtain remote information without the typically-assumed capabilities of adversaries, such as malware injection, eavesdropping or control of a part or whole of a communication system. The information obtained by monitoring the communication pattern (e.g., call time, length, and frequency), does not merely provide records of communication, but can also be used to construct the tie strength of social networks. Such information has historically been used by intelligence, military, law-enforcement and business organizations as a powerful tool to monitor persons of interest. Three PCD approaches based on probing the targets were proposed and evaluated in four applications. Among PCD countermeasures, monitoring and blocking periodic suspicious network traffic is a general solution that could work in most cases.

The Resource-Saturation PCD discovers the busy status of network end-nodes by sending special probing messages that can saturate the shared resource in the network end-nodes, then analyzing the exceptions in the responses caused by the saturated resource. The call records can be inferred by correlating the busy status of the network end-nodes. The Resource-Saturation PCD successfully attacked three different commercial closed-source VoIP phones, without requiring access to source code.

Unlike the Resource-Saturation PCD, which requires time-consuming protocol

analysis, the Stochastic PCD takes a classification approach, such that the time series of probing responses is categorized, allowing for a probabilistic test of which hypothesis of the communication pattern is more likely to be true. Through signal-processing techniques, this approach can also deal with the noise in the time series brought by Internet switching, application proxies, and other nodes in the network. This attack has been applied to two applications, WiFi and Instant Messaging, in experimental environments.

The Service-Priority PCD takes advantage of one inherent design principle in many communication systems: services are given different priority when the network usage may overwhelm the network capacity. This is especially true for mobile communication networks. In 3G/2G mobile communication systems where data and voice services are both supported, a spike in the probing response time will indicate that a voice call is taking place, taking advantage of the network design feature that the service level of data service is lowered during a voice call.

This dissertation successfully bridges communication and mobile privacy research with the idea of side-channels, which were originally used in cryptanalysis. The proposed attacks suggest that the design of future communication systems should consider resource contention as a major potential vulnerability. The vulnerability could exist in application or operating system resources (e.g. a fixed-size buffer), communication resources (e.g., radio channels), and even in the interaction with human themselves, in the form of user interfaces (e.g., keyboard).

6.1. Future Directions

The PCD attack and its implications suggest several directions for promising future work. Other aspects of the PCD methods and different kinds of shared resources (and the corresponding side channels) that would be worthy for investigation are listed below.

1. *Large-scale PCD*. We used a simple call model (i.e., uniform-distributed beginning time and end time of a call) in Appendix I to calculate the false detection rate when PCD is applied to more than two targets. However, telephone use really depends on the time, the date, and the user. Therefore, we can use the real call records to find a more accurate model and to estimate its parameters. The new model will allow us to better calculate the false detection rate in large-scale PCD. Looking at this from the other side, the adversary could also apply some heuristics to reduce the false detection rate. For example, if three PCD targets show that they are busy during the same time, it is unlikely that they are having a conference call because a conference call is established either by adding the third party to the existing two-party call or by calling to the conference bridge. Instead, it is more likely that two of them are talking to each other and the other target is communicating with someone else, who is not a target.
2. *The Power of Stochastic PCD*. In Stochastic PCD, we found that existing signal-processing techniques can reduce the noise in the side channels. On the other hand, a general defense to side-channel attacks is adding noise. These

two facts suggest that the theoretical limit of both adding noise and noise reduction capacity should be investigated; otherwise, we do not know how much added noise would be sufficient to defend against the Stochastic PCD.

3. *4G mobile communication systems*. One of the major differences between the 4G and 3G mobile communication systems is that 4G uses packet switching to deliver both voice and data service while 3G uses packet switching for data service and circuit switching for voice service. Therefore, whether Service-Priority PCD is applicable to 4G is not yet known.
4. *Satellite Communication*. Communication satellites are another possible target for PCD, because they usually use solar cells and batteries as their power sources. Therefore, the energy resource is limited. While multiple components of a satellite are using this limited resource, contention takes place. Therefore, we suspect that the energy resource contention may cause a side channel. An adversary who has access right to use the communication satellites may indirectly observe the communication pattern between these satellites or between a satellite and a ground station. This attack may have values in military or business intelligences. For example, a captured node or weapons platform (e.g., a tank) that has military satellite network access may help the adversary to monitor the communications in this military satellite network. For business intelligence, a satellite network operator may want to get the traffic information of its competitor to understand its competitor's business model. In that case, the operator can employ the PCD attack against the competitor's network.

5. *Smartphone Apps*. Although mobile phones provide a confined environment (e.g., a sandbox, virtual machine) to isolate apps running in the same phone, it may be possible for one malicious app to obtain another app's internal state through some shared system resources under contention (e.g., file descriptors, and semaphores). Furthermore, the malicious Apps in different smartphones can cooperate to correlate the observed information (i.e., behavior of other apps) to infer the communication pattern between these smartphones. This provides a new set of potential targets for the PCD attack, such as monitoring a banking application and the corresponding money flows.
6. *Power Grid*. The smart power grid is regarded as another life-changing networking technology, similar to the widespread use of the modern Internet. We suspect that these smart devices may bring more privacy issues because they now have communication abilities. For example, smart meters can measure the household electricity usage and report back to the electricity provider. It is possible that these smart meters have resource-contention side channels and the adversary use them to observe the activities inside a building without physically being inside.

Appendices

Appendix I. False Detection Rate for PCD

We use a simple model to derive the false call-detection rate when the monitored phones also communication with third-party phones. We assume that calling time is uniformly distributed. If we have telecommunication traffic data, a more accurate model can be built. Our model is as follows.

Model assumption

1. During T seconds, two phones Alice and Bob are probed for *n* times with a fixed time interval. Phones are probed at time $t_1,t_2,\ldots,t_n$.
2. During the T seconds, Alice and Bob both have one call. Alice becomes busy during the t_X and t_Y probes ($X \leq Y$ and $X,Y \in \{1,2,\ldots,n\}$); Bob becomes busy during the t_P and t_Q probes ($P \leq Q$ and $P,Q \in \{1,2,\ldots,n\}$).
3. X~Uniform(1,*n*) , Y~Uniform(X,*n*)
4. P~Uniform(1,*n*) , Q~Uniform(P,*n*)
5. X and P are i.i.d.
6. Y and Q are i.i.d.

The probability when Alice and Bob's call have the same start time and end time is Pr(F).

$$\Pr(F) = \Pr(X = P \text{ and } Y = Q)$$

$$= \sum_{x=1}^{n} \Pr(X = x) \Pr(X = P \text{ and } Y = Q | X = x)$$

$$= \frac{1}{n} \times \frac{1}{n} + \frac{1}{n} \times \frac{1}{n-1} + \frac{1}{n} \times \frac{1}{n-1} + \cdots + \frac{1}{n} \times \frac{1}{1}$$

$$= \frac{1}{n} \sum_{k=1}^{n} \frac{1}{k}$$

Suppose there is a call on both phones during a one-hour period, what is the probability that these two calls are not the same call? (T=3600 seconds)

If the time resolution of the busy-status detection is 1 second (n=3600), Pr(F)=0.0024.

If the time resolution of the busy-status detection is 10 second (n=360), Pr(F)=0.018

We can easily prove Pr(F) decreases as n increases. It means the higher the probing resolution, the lower the false detection rate.

$$\Pr(F)_n - \Pr(F)_{n+1}$$

$$= \frac{1}{n} \sum_{k=1}^{n} \frac{1}{k} - \frac{1}{n+1} \sum_{k=1}^{n+1} \frac{1}{k}$$

$$= \sum_{k=1}^{n} \left(\frac{1}{n} \times \frac{1}{k} - \frac{1}{n+1} \times \frac{1}{k} \right) - \frac{1}{n+1} \times \frac{1}{n+1}$$

$$= \sum_{k=1}^{n} \left(\frac{1}{n(n+1)} \times \frac{1}{k} \right) - \frac{1}{n+1} \times \frac{1}{n+1}$$

$$= \frac{1}{n(n+1)} \times \left(\sum_{k=1}^{n} \frac{1}{k} - \frac{n}{n+1} \right) > 0$$

Appendix II. Probe Messages of Resource-Saturation PCD

Several attack SIP messages are listed here. Note that the adversary should use different Call-ID in different requests so they are not being identified as duplicates.

Message 1. OPTIONS probe message

```
OPTIONS sip:wj@iptel.org SIP/2.0
Via: SIP/2.0/UDP 127.0.0.1:5000;branch=z9hG4bK.1770151725;rport;alias
From: sip:voicepulse@127.0.0.1:5000;tag=1236798926
To: sip:wj@iptel.org
Call-ID: 1811209364@127.0.0.1
CSeq: 1 OPTIONS
Contact: sip:voicepulse@127.0.0.1:5000
Content-Length: 0
Max-Forwards: 70
User-Agent: Voice Pulser version 1
Accept: text/plain
```

Message 2. INVITE-require probe message

```
INVITE sip:wj@iptel.org SIP/2.0
Via: SIP/2.0/UDP 127.0.0.1:5000;branch=z9hG4bK.899244477;rport;alias
From: sip:voicepulse@127.0.0.1:5000;tag=524046249
To: sip:wj@iptel.org
Call-ID: 111248654@127.0.0.1
CSeq: 1 INVITE
Contact: sip:voicepulse@127.0.0.1:5000
Content-Length: 0
Max-Forwards: 70
User-Agent: Voice Pulser version 1
Accept: text/plain
Require:xx
```

Message 3. INVITE-SDP probe message for Cisco 7940G

```
INVITE sip:wj@iptel.org SIP/2.0
Via: SIP/2.0/UDP 127.0.0.1:5000;branch=z9hG4bK.600541343;rport;alias
From: sip:voicepulse@127.0.0.1:5000;tag=1552686011
To: sip:wj@iptel.org
Call-ID: 1436339943@127.0.0.1
CSeq: 1 INVITE
Contact: sip:voicepulse@127.0.0.1:5000
Content-Length: 181
Content-type:  application/sdp
Max-Forwards: 70
User-Agent: Voice Pulser version 1
Accept: text/plain

v=0
o=alice 2890844526 2890844526 IN IP4 192.168.1
s=
c=IN IP4 192.168.1a
t=0 0
m=audio 49170 RTP/AVP 0 8 97
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
```

Message 4. INVITE-SDP probe message for Grandstream BT-100

```
INVITE sip:wj@iptel.org SIP/2.0
Via: SIP/2.0/UDP 127.0.0.1:5000;branch=z9hG4bK.600541343;rport;alias
From: sip:voicepulse@127.0.0.1:5000;tag=1552686011
To: sip:wj@iptel.org
Call-ID: 1436339943@127.0.0.1
CSeq: 1 INVITE
Contact: sip:voicepulse@127.0.0.1:5000
Content-Length: 181
Content-type:  application/sdp
Max-Forwards: 70
User-Agent: Voice Pulser version 1
Accept: text/plain

v=0
o=- 3530196336 3530196336 IN IP4 128.2.13a
s=pjmedia
c=IN IP4 128.2.13.1A
t=0 0
a=X-nat:0
xm=audio 4000 RTP/AVP 103 102 104 117 3 0 8 9 101
a=rtcp:4001 IN IP4 128.2.13.168
a=rtpmap:103 speex/16000
a=rtpmap:102 speex/8000
a=rtpmap:104 speex/32000
a=rtpmap:117 iLBC/8000
a=fmtp:117 mode=30
a=rtpmap:3 GSM/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:9 G722/8000
a=sendrecv
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
```

Message 5. NOTIFY-refer probe message

```
NOTIFY sip:wj@iptel.org SIP/2.0
Via: SIP/2.0/UDP 127.0.0.1:5000;branch=z9hG4bK.1592896042;rport;alias
From: sip:voicepulse@127.0.0.1:5000;tag=499998901
To: sip:wj@iptel.org
Call-ID: 1599885472@127.0.0.1
CSeq: 1 NOTIFY
Contact: sip:voicepulse@127.0.0.1:5000
Content-Length: 0
Max-Forwards: 70
User-Agent: Voice Pulser version 1
Accept: text/plain
Event:refer
Subscription-State: active;expires=180
```

Appendix III. Disabled Period of Resource-Saturation PCD

We illustrate the timing of busy-status detection and calculate the disabled period, as shown in Figure 47. Consider the three relevant time points of a transaction. The black diamond and the white diamond show the time that the SIP phone receives the request and responds the request, respectively, whereas the black square denotes the end of a transaction. The interval between the first two time points is the processing time *r*, a small time period measured to be between 10ms and 100ms. After this interval, a transaction waits for L seconds; e.g., L is 32 seconds (Timer J) for non-INVITE UDP transactions. For INVITE UDP transactions, L is 32 seconds (Timer H) if no ACK request is received, and is 5 seconds (Timer I) if an ACK request received. These timers are standard [31].

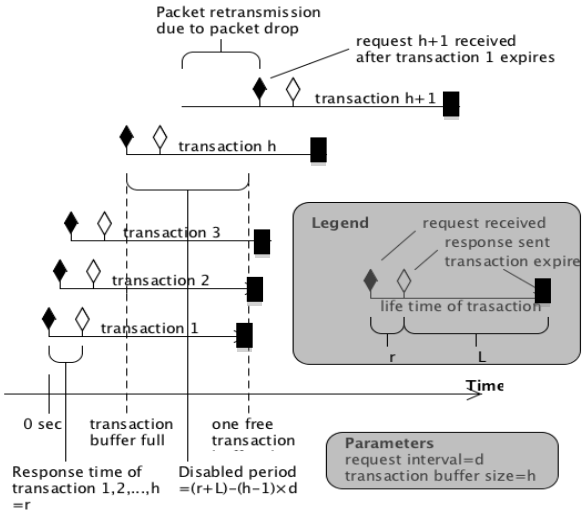


Figure 47. Disabled Period

In calculating the disabled period, we omit the network transmission time since it is negligible compared to the disabled period. In Figure 10, time 0 denotes the receipt of the 1st request. For the first *h* requests, the SIP phone takes *r* seconds to respond to each request. The phone will become available to accept a new request after the 1st request expires, at time *r*+L. Thus, during time (*h*-1)**d* and time *r* +L, the phone is unable to accept a new request for (*r*+L)-(*h*-1)×*d* seconds. The response to an INVITE request is given after the disabled period.

Given parameters of PAP2 and 7940G along with an estimated parameter *r*=20ms, we are able to calculate the disabled period of both phones. PAP2 phone's disabled period is 29.54 seconds with parameters L=32sec, *d*=80ms, and *h*=32. 7940G phone's disabled period (not shortened) is 31.62 seconds with a different parameter *h*=6. However, by applying disabled-period shortening, the shortened disabled period of 7940G is smaller than 2 seconds.

Bibliography

[1] AT&T's advertisement on simultaneous surfing and talking, http://www.youtube.com/watch?v=kYjBg_r114o, retrieved on Feb 7, 2012

[2] Agrawal, D., Archambeault, B., Rao, J., and Rohatgi, P. *The EM Side-Channel(s).* In Proceeding of Workshop on Cryptographic Hardware and Embedded Systems (CHES), 2002, Redwood, CA

[3] Alvarez-Campana, M., Vázquez, E., and Vinyes, J. *Performance Modeling of Web Access over HSPA Networks*. In Proceeding of International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2010, Moscow, Russia

[4] Asnov, D., and Agrawal, R. *Keyboard Acoustic Emanations*. In Proceeding of the IEEE Symposium on Security and Privacy, 2004, Oakland, CA

[5] Barbuzzi, A., Ricciato, F., and Boggia, G. *Discovering Parameter Setting in 3G Networks via Active Measurements*. IEEE Communication Letters, Oct 2008, Vol. 12 Issue 10, Page 730-732

[6] Blond, S. L., Zhang, C., Legout, A., Ross, K., and Dabbous, W. *I Know Where You are and What You are Sharing: Exploiting P2P Communications to Invade Users' Privacy*. In Proceeding of IMC 2011, Berlin, Germany

[7] Boneh, D., DeMillo, R. A., and Liptonm, R. J. *On the Importance of Checking Cryptographic Protocols for Faults.* In Proceeding of EUROCRYPT, 1997, Konstanz, Germany

[8] Brier, E., Clavier, C., and Olivier, F. *Correlation Power Analysis with a Leakage Model*. In Proceeding of Workshop on Cryptographic Hardware and Embedded Systems (CHES), 2004, Cambridge, MA

[9] Brueck, S., Jugl, E., Kettschau, H.-J., Link, M., Mueckenheim, J., and Zaporozhets, A. *Radio Resource Management in HSDPA and HSUPA*. Bell Labs Technical Journal, Winter 2007, Volume 11, Issue 4

[10] Cauley, L. *NSA has Massive Database of Americans' Phone Calls*. USA Today, May 11, 2006.

[11] Cesana, M., and Capone, A. *Impact of mixed Voice and Data Traffic on the UMTS-FDD Performance*. In Proceeding of IEEE GLOBECOM, 2002, Politecnico di Milano, Italy

[12] Chandra, R., Bahl, P., and Bahl, P. *MultiNet: Connecting to Multiple IEEE 802.11 Networks Using a Single Wireless Card*. In Proceeding of IEEE INFOCOMM, 2004, Hong Kong

[13] ChangeWave Research. *ChangeWave Survey Results: AT&T iPhone 4 Owners vs. Verizon iPhone 4 Owners*. http://www.changewaveresearch.com/articles/2011/att_verizon_iphone4_20110405.html, retrieved on Feb. 16, 2012

[14] Chao, H., Liang, Z.C., Wang, Y.G., and Gui, L.N. *A Dynamic Resource Allocation Method for HSDPA in WCDMA System*. In Proceeding of 5th IEE International Conference on 3G Mobile Communication Technologies, 2004, London, UK

[15] Chaum, D. *Untraceable Electronic Email, Return Addresses, and Digital Pseudonyms*. Communications of the ACM, 1983, Vol. 24, No. 2

[16] Cox, C. *Essentials of UMTS*. Cambridge University Express, 2008, ISBN 978-0-521-88931-5

[17] Danezis, G. and Diaz, C. *A survey of anonymous communication channels*. Microsoft Research Technical Report (MSR-TR-2008-35), Jan. 2008

[18] Danezis, G. *Statistical Disclosure Attacks: Traffic Confirmation in Open Environments*. In Proceeding of Security and Privacy in the Age of Uncertainty (SEC), 2003, Athens, Greece

[19] DeCew, J. *Privacy*. The Stanford Encyclopedia of Philosophy (Fall 2008 Edition), Edward N. Zalta (ed.), URL = <http://plato.stanford.edu/archives/fall2008/entries/privacy/>.

[20] Dingledine, R., Mathewson, N., and Syverson, P. *Tor: The Second-Generation Onion Router*. In Proceeding of the 13th USENIX Security Symposium, 2004, San Diego, CA

[21] Fathi, H., Chakraborty, S. S., and Prasad, R. *Optimization of SIP Session Setup Delay for VoIP in 3G Wireless Networks*. IEEE Transaction on Mobile Computing, Vol. 5, No. 9, Sep. 2006

[22] Furui, S. *Speaker-Independent Isolated Word Recognition Using Dynamic Features of Speech Spectrum*. IEEE Transaction of Acoustics, Speech and Signal Processing, Vol. ASSP-34, No. 1, Feb 1986

[23] Fu, X., Graham, B., Bettati, R., and Zhao, W. *Active Traffic Analysis Attacks and Countermeasures*. In Proceeding of IEEE ICCNMC, 2003, Shanghai, China

[24] Fyodor. *Remote OS Detection via TCP/IP Stack FingerPrinting*. Phrack 54, 8, Dec 1998. URL http://nmap.org/nmap-fingerprinting-article.txt

[25] Gilbert, E., and Karahalios, K. *Predicting Tie Strength With Social Media*. In Proceeding of ACM CHI 2009, April 2009, Boston, MA

[26] Gligor, V. D. *Covert Channel Analysis of Trusted Systems*. Light Pink Book of the Rainbow Series by U.S. Department of Defense National Computer Security Center, NCSC-TG-030

[27] Gong, X., Kiyavash, N. and and Borisov, N. *Fingerprinting Websites Using Remote Traffic Analysis*. In Proceeding of ACM CCS, 2010, Chicago, IL

[28] Herrmann, M., and Grothoff, C. *Privacy-Implications of Performance-Based Peer Selection by Onion-Routers: A Real-World Case Study Using I2P*. In Proceeding of Privacy Enhancing Technologies Symposium (PETS), 2011, Waterloo, Canada

[29] Huskamp, J. C. *Covert Communication Channels in Timesharing Systems.* Ph.D. Dissertation of University of California, Berkeley, 1976

[30] IANA-defined SIP Parameters. http://www.iana.org/assignments/sip-parameters

[31] IETF RFC 3261. *SIP: Session Initiation Protocol*

[32] IETF RFC 3264, *An Offer/Answer Model with the Session Description Protocol (SDP*)

[33] IETF RFC 3265. *Session Initiation Protocol (SIP)-Specific Event Notification*

[34] ITU-T G.114. *One-way transmission time*

[35] Jong, C.-H., and Gilgor, V. D. *Discovering Records of Private VoIP Calls without Wiretapping*. In Proceeding of ACM AsiaCCS, 2012, Seoul, Korea

[36] Jong, C.-H., and Gilgor, V. D. *Private Communication Detection: A Stochastic Approach*. In Proceeding of ACM WiSec 2012, Tucson, AZ

[37] Jurasky, D. and Martin, J. H. *Speech and Language Processing*. Pearson Prentice Hall, 2nd Edition, 2008

[38] Kalden, R., Meirick, I., and Meyer, M., *Wireless Internet Access Based on GPRS*. IEEE Personal Communications, April, 2000

[39] Kazatzopoulos, L., Delakouridis, C., and Marias, G.F. *Providing Anonymity Services in SIP*. In Proceeding of IEEE PIMRC, 2008, Cannes, France

[40] Kocher, P.C. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Proceeding of CRYPTO, 1996, Santa Barbara, California

[41] Kocher, P.C., Jaffe, J., and Jun, B. *Differential Power Analysis*. In Proceeding of CRYPTO, 1999, Santa Barbara, CA

[42] Lampson, B. W. *A Note on the Confinement Problem*. Communications of ACM, 1973, Vol. 16, Issue 10

[43] Le, T.-H., Clédière, J., Servière, and C., and Lacoume, J.-L. *Noise Reduction in Side Channel Attack Using Fourth-Order Cumulant*. IEEE Transaction of Information Forensics and Security, Vol. 2, No. 4, December 2007.

[44] Lee, K.-F. *On Large-Vocabulary Speaker-Independent Continuous Speech Recognition*. Speech Communication, Elsevier Science Publishers, 1988

[45] Lee, K.-H., Park, J.-H., and Koh, J.-S. *User Experience Analysis of Smartphone*

Web Surfing in UMTS Networks. In Proceeding of IEEE VTC, 2010, Seoul, South Korea

[46] Levine, B. N., Shields, C., and Margolin, N. B. *A Survey of Solutions to the Sybil Attack*. Tech report 2006-052, University of Massachusetts Amherst, October 2006.

[47] Lin, Y.-B., and Tsai, M.-H. *Eavesdropping Through Mobile Phone*. IEEE Transaction on Vehicular Technology, Vol 56, Issue 6, Nov 2007

[48] Litjens, R., and Boucheri, R.J. *Radio Resource Sharing in a GSM/GPRS Network*. In Proceeding of the ITC Specialist Seminar on Mobile Systems and Mobility, 2000, Lillehammer, Norway

[49] Markoff, J. *Taking Spying to a Higher Level.* New York Times, Feb. 2006, http://www.nytimes.com/2006/02/25/technology/25data.htm, (accessed Dec 2, 2011)

[50] McKeay, M. *Taking Corporate Spying to a Higher Level. Computerworld*. 2006, http://blogs.computerworld.com/node/3396 (accessed Dec 2, 2011)

[51] Messerges, T.S., and Dabbish, E. A. *Investigations of Power Analysis Attacks on Smartcards*. In Proceeding of USENIX Workshop on Smartcard Technology, 1999, Chicago, IL

[52] Murdoch, S. J. *Covert Channel Vulnerabilities in Anonymity Systems*. University of Cambridge Technical Report, UCAM-CL-TR-706

[53] Murdoch, S. J. and Danezis, G. *Low-Cost Traffic Analysis of Tor*. In Proceeding of the IEEE Symposium on Security and Privacy, 2005, Oakland, CA

[54] Murdoch, S. J. *Hot or Not: Revealing Hidden Services by their Clock Skew*. In Proceeding of ACM CCS, 2006, Alexandria, VA

[55] Newsome, J., Shi, E., Song, D. and Perrig, A. *The Sybil Attack in Sensor Networks: Analysis & Defenses.* In Proceeding of the ACM IPSN, 2004, Berkeley, CA

[56] OpenSips software, http://opensips.org

[57] Padhye, J. and Floyd, S. *On Inferring TCP Behavior*. In Proceeding of ACM SIGCOMM, 2001, San Diego, CA

[58] Parent, W. A. *Privacy, Morality, and the Law*. Philosophy and Public Affairs, 1983, Vol. 12, No. 4

[59] Pecen, M., and Howell, A. *Simultaneous Voice and Data operation for GPRS/EDGE: Class A Dual Transfer Mode.* IEEE Personal Communication, April, 2001

[60] Perala, P.H.J., Barbuzzi, A., Boggia, G., and Pentikousis, K. *Theory and Practice of RRC State Transitions in UMTS Networks*. In Proceeding of IEEE GLOBECOM, 2009, Honolulu, Hawaii

[61] Pfitzmann, A., and Hansen, M. *A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management*. Version 0.34 Aug. 10, 2010, available on http://dud.inf.tu-dresden.de/Anon_Terminology.shtml

[62] Pfitzmann, A., Pfitzmann, B., and Waidner, M. 1991. *ISDN-MIXes: Untraceable Communication with Very Small Bandwidth Overhead*. In Proceeding of Communication in Distributed Systems. Springer-Verlag

[63] Pidgin software, http://www.pidgin.im

[64] Qian, F., Wang, Z., Gerber, A., Mao, Z. M., Sen, S., and Spatscheck, O. *Characterizing Radio Resource Allocation for 3G networks*. In Proceeding of ACM IMC 2010, Melbourne, Australia

[65] Rabiner, L. *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*. In Proceeding of the IEEE, vol. 77, pp. 257–286, Feb 1989.

[66] Raymond, J.-F. *Traffic analysis: Protocols, attacks, design issues and open problems*. In Proceeding of International Workshop on Design Issues in Anonymity and Unobservability, 2000, Berkeley, CA

[67] Redl, S. M., Oliphant, M. W., Weber, M. K. *An Introduction to GSM*. Artech

house, 1995, ISBN 9780890067857

[68] Resig, J., Dawara, S., Homan, C. M., and Teredesai, A. *Extracting Social Networks from Instant Messaging Populations*. In Proceeding of LinkKDD, 2004, Seattle, Washington

[69] Solomon, J., Johnson, C. *FBI Broke Law for Years in Phone Record Searches*. Washington Post, Jan. 19, 2010

[70] Song, D. X., Wagner, D., and Tian, X. *Timing Analysis of Keystrokes and Timing Attacks on SSH.* In Proceeding of USENIX Security, 2001, Washington, DC

[71] Srivatsa, M., Iyengar, A., Liu, L. and Jiang, H. *Privacy in VoIP Networks: Flow Analysis Attacks and Defense*. IEEE Transaction on Parallel and Distributed Systems, Vol. 22, No. 4, April 2011

[72] Superstructure Group. *SiD Case Study in Drug Intelligence.* rel. 1.1, February 2011, www.superstructuregroup.com/Resources/SiDCaseStudy_DrugIntell.pdf (accessed Aug. 20, 2011)

[73] Tan, P.-N., and Kumar, V. *Introduction to Data Mining.* Addison-Wesley, 2006

[74] Wang, X., Chen, S., and Jajodia, S. *Tracking Anonymous Peer-to-Peer VoIP Calls on the Internet.* In Proceeding of ACM CCS, 2005, Alexandria, VA

[75] Warren, S., and Brandeis, L. *The Right to Privacy*. Harvard Law Review, Vol. IV, December 15, 1890, No. 5

[76] Wright, C. V., Ballard, L., Coull, S. E., Monrose, F., and Masson, G. M. *Spot me if you can: Uncovering Spoken Phrases in Encrypted VoIP Conversations*. In Proceeding of IEEE Symposium on Security and Privacy, 2008, Oakland, CA

[77] Xia, L., Kumar, S., Yang, X., Gopalakrishnan, P., Liu, Y., Schoenberg, S., and Guo, X. *Virtual WiFi: Bring Virtualization from Wired to Wireless*. In Proceeding of ACM International Conference on Virtual Execution Environments (VEE), 2011, Newport Beach, CA

[78] Zhang, F., He, W., Liu, X., and Bridges, P. G. *Inferring Users' Online Activities*

Through Traffic Analysis. In Proceeding of ACM WiSec, 2011, Hamburg, Germany