Discovering Records of Private VoIP Calls without Wiretapping

Chang-Han Jong
ECE Department
University of Maryland
College Park, Maryland 20742, USA
chjong@umd.edu

Virgil D. Gligor
ECE Department and CyLab
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213, USA
gligor@cmu.edu

ABSTRACT

In this paper we show that any user of a VoIP service over a private network, even one without a special attack capability (e.g., wiretapping), can discover calls between two targeted individuals, with high probability. We conducted this privacy attack in an experimental setting using three types of commercially available closed-source phones that implement the standard IETF Session Initiation Protocol (SIP) in hardware. We show that private call records can be probabilistically derived by using a new class of side-channels caused by resource contention. By sending carefully designed VoIP packets and analyzing the responses, an ordinary user can detect the busy status of SIP phones without alerting either the caller or the callee. Hence an ordinary user can correlate the busy status of two given phones, or more, can detect calls between them. We demonstrate the effectiveness of our remote attack on three commercial closed-source phones, and discuss countermeasures.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General – *security and protections.*

General Terms

Experimentation, Security

Keywords

Privacy; Anonymity; Side-Channel Attacks; Protocol

1. INTRODUCTION

On January 10, 2010, the Washington Post reported that the FBI broke US law by illegally obtaining over 2,000 records for phone calls made by US citizens (call logs, but not call content) without obtaining a "national security letter," as required by the Patriot Act of 2001 [28]. Since a call record does not contain information about the call (e.g., voice patterns, data, content), one cannot help but wonder: What makes the collection of call records such an attractive data source as to cause the FBI to break US law? Furthermore, why would the FBI break the law for so few records, as compared to similar actions by the NSA, which collected over 1.9 trillion call records between 2001 and 2004 [1, 19] ? These few records must provide important information.

Call-record analysis is one of the oldest tools used in defense, law-enforcement, and business intelligence. Call records enable an analyst to discover the social milieu of targeted individuals, or groups. For example, a call-record database allows both single link (e.g., time, initiation, duration, frequency of a call) and cluster analysis of calls in the temporal, spatial, and frequency domains. It can also indicate overlaps among different clusters, such as those obtained from different investigations, and similarity of clusters, such as those obtained when a group of targets changes their phone numbers (to avoid tracking) but not their communication habits. Relatively small call-record sets, well under 10,000 records, have been sufficient to discover a variety of law-breaking operations worldwide, ranging from drug trafficking in New Zealand [29], to drug smuggling in a Minnesota prison, to surreptitiously charging unsuspecting pornography clients with expensive and unwanted 900-number phone calls in Moldova [19]. Even smaller sets of call records have been used to (illegally) detect boardroom leaks in a major US corporation [20].

The question that motivates the privacy attack reported in this paper is the following: would call-record analysis be possible in a private VoIP network? Specifically, a private VoIP provides anonymity for callers and callees, and for their relationship [24]. Informally, this means that the caller and callee's phones are indistinguishable from other phones in the VoIP network, and neither the caller nor the callee can be linked to each other. In such a setting, end-to-end encryption would force end-point invasive wiretapping, or even end-host malware/Trojan insertion, for call-record collection. This would be an expensive proposition, which would undoubtedly require additional legal work, such as court orders, and would face increased odds of detection by the owners of the targeted phones. While a private VoIP network is yet to be fully achieved in practice, a low-latency encrypted channel using with one or more (trusted) forwarding proxies is a reasonable approximation[1] of such a network and could become the medium of choice for the private calls in the rapidly growing VoIP communications [3, 17]. In this paper, we answer our motivating question affirmatively, by showing that any user of a private VoIP network *without any special attack capability* – not just a powerful government agency equipped with a national security letter – can discover private calls between two or more

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee
ASIACCS'12, May 2–4, 2012, Seoul, Korea.
Copyright 2012 ACM 978-1-4503-1303-2/12/05…$10.00.

[1] Low-latency anonymous networks provide anonymity through multiple forwarding proxies where anonymity is still maintained even if some of the proxies are compromised. Current deployed systems such as Tor and I2P cannot provide low enough latency for VoIP to become practical (<400ms delay). However, commercial services such as anonymous.com provide anonymity through a trusted proxy (VPN) where latency is low enough.

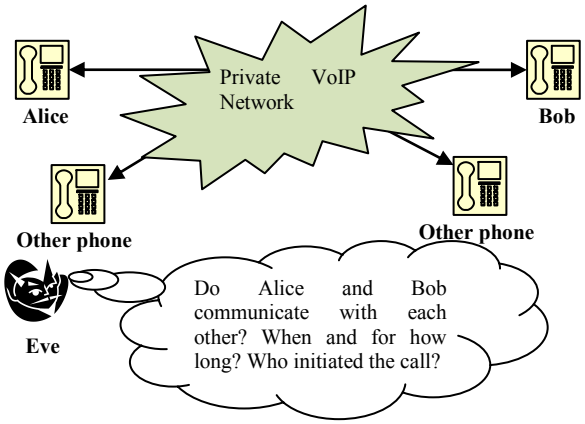

Figure 1. Revealing call records in a private VoIP network

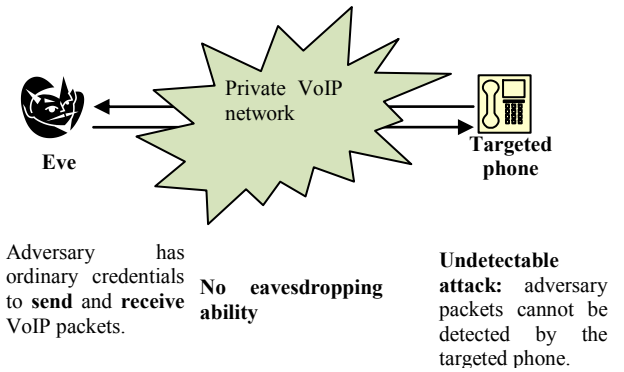

Figure 2. Adversary capabilities

specified targeted phones. Discovery of private calls can breach the relationship privacy of two or more phone users and can have unpredictable effects, as we argue below.

Adversary Definition. In Figure 1, 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) [3, 4, 13, 17]. The forwarding proxies can be implemented in the network layer, transport layer, or application layer. Adversary Eve's goals are to discover whether Alice and Bob communicate with each other using the private VoIP network, the duration of the conversation, and if possible, the call initiator. [2] Eve can also perform link analysis and determine the strength of ties between Bob and Alice. 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 targeted parties [7].

Our adversary Eve does not need any added capabilities beyond those of an ordinary user. As shown in Figure 2, she can send probe VoIP packets to the targeted phones and receive response VoIP packets in return, and need not have any eavesdropping capabilities. Yet her probing is undetectable[3] by the end-users of the targeted 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.

Attack Overview. The simplest attack consists of the two steps illustrated in Figure 3. In the first step, the Eve detects whether both Alice's phone and Bob's phone 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 almost the same time; i.e., she performs *call correlation*. If the phones' busy statuses correlate, then Alice and Bob probably share a VoIP call session 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 targeted phones can easily recognize the anomaly and stop using these phones.

Resource contention is what makes it possible to detect a phone's busy status in a private VoIP network. The resource, namely 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 targeted 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 targeted 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 private VoIP network. 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. Theoretically, the proposed attack may apply to landline phones if the adversary has access to the telecommunication signaling network (e.g., having a Private Branch Exchange).

Contribution. The main contribution of this paper is the definition of a new, powerful attack against call-records privacy, which is launched with ordinary-user capability and succeeds with very high probability in private VoIP networks. This attack not only indicates that a fixed-size buffer in a SIP phone can result in private information leakage, but also implies that other exclusive resources, such as operating system resources (e.g., semaphore; number of threads) may enable privacy breaches in SIP or other VoIP phones. We tested our attack in closed-source commercial

[2] In addition to the hypothetical role of an eager FBI agent lacking a national security letter, Eve could alternatively be a jealous girlfriend wanting t to determine whether her boyfriend Bob is cheating with her best friend Alice.

[3] We assume the unsuspecting end-user of a targeted phone accesses the phone via the handset, hook, buttons, rings, and a display.

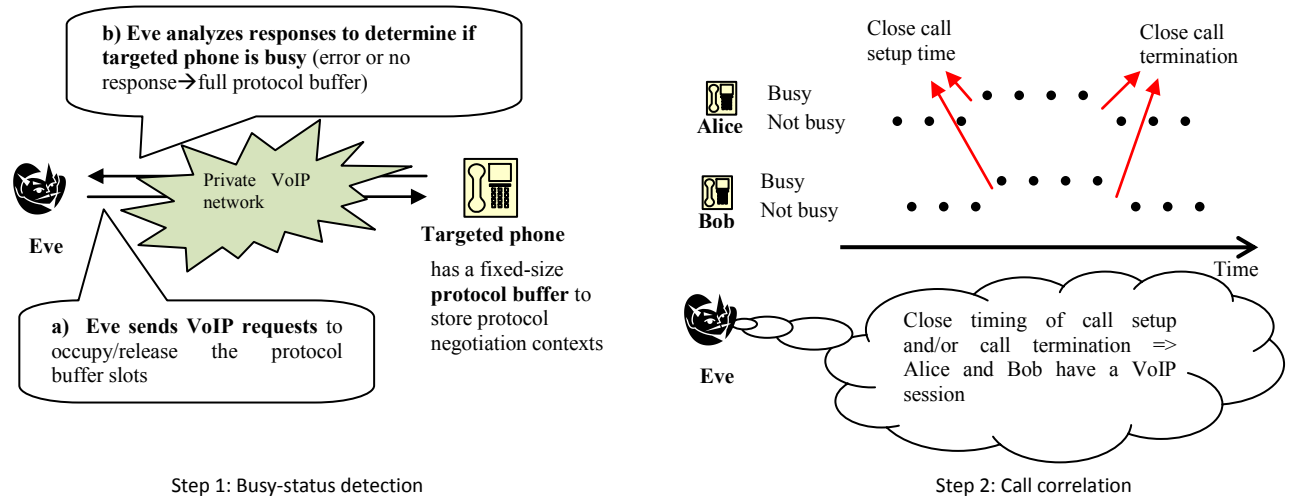

Figure 3. Attack steps

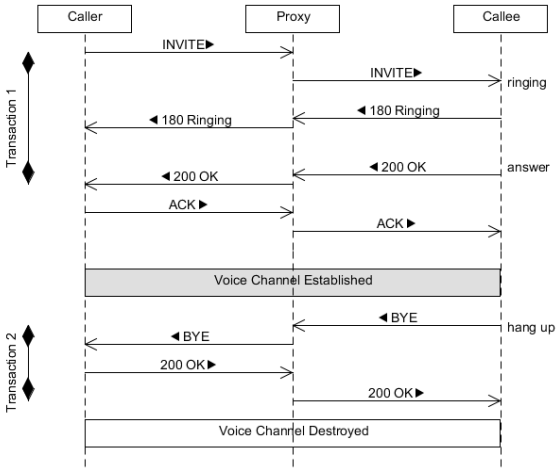

Figure 4. SIP call setup and termination

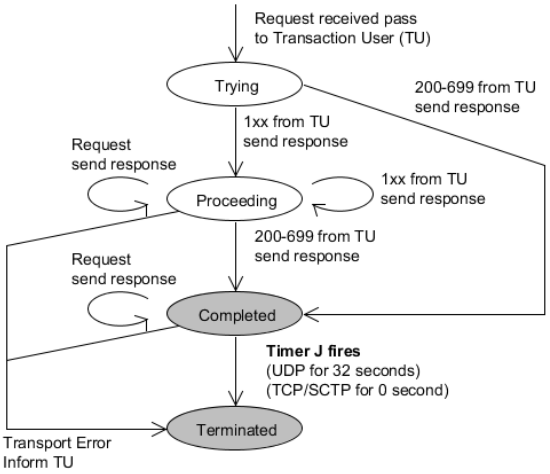

Figure 5. A non-INVITE transaction at the server side

SIP phones, which do not provide implementation documentation, a fact that provides further testimony for the practicality of this attack. We propose countermeasures against this type of attack, including protocol-buffer manipulation to reduce call-detection probability, attack-discovery mechanisms, and firewalls.

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 six 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 2.1 below.)

2.1 Call Setup and Termination Overview

SIP is an HTTP-like application-layer protocol designed for VoIP signaling and other applications that require devices to setup/terminate sessions to exchange information [10]. 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 paper 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, generally does not affect our attacks.

In Figure 4, 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 of 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 also 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 (as the final acknowledgement), the caller and the callee will have already reached an agreement on the media parameters for establishing a voice channel, using a streaming protocol such as RTP. Eventually one of the two parties (in this case, the callee) will terminate the call by initiating Transaction 2 with a BYE request. The other party replies using a "200 OK", which terminates the voice channel.

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 5, 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.[4] Timer J also helps SIP to handle packets lost in UDP via retransmission.

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, 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 targeted phone, sending 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

[4] 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 [5].

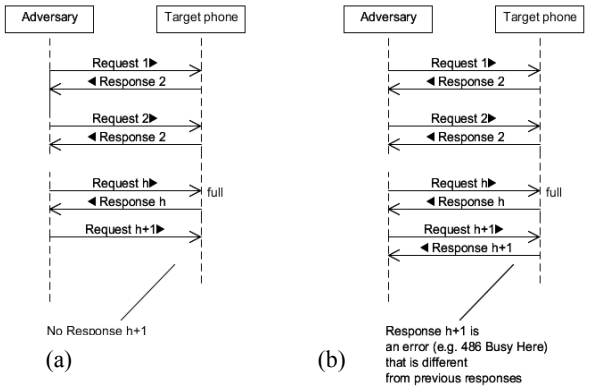

Figure 6. Possible target-phone responses when the protocol buffer is full

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 response number *h*+1, as shown in Figure 6 (a). In the other case, the adversary will receive an error response for the $h+1^{st}$ request, such as "486 Busy Here," as shown in Figure 6 (b). In either case, the adversary discovers the value of *h*.

Detection Side Effects. Note that if the SIP phone, or proxy in the transmission path, supports retransmission, the adversary will get a delayed positive response to the $h+1^{st}$ request after the protocol buffer slot becomes free. However, this 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 this calculation are shown in Appendix II. During the disabled period, the targeted phone does not answer a caller's setup request. Only after the disabled period ends is the targeted phone able to receive a call setup request issued by the retransmission mechanism of the caller phone or proxy. This 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-source 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 I.

1) INVITE-require and INVITE-SDP requests. As described in Section 2.1, the INVITE transaction performs the call setup. By

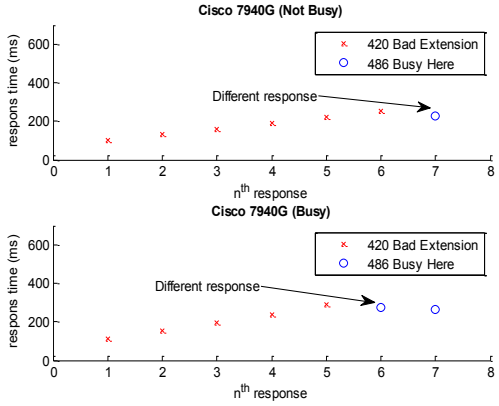

Figure 7. Responses of 7940G under INVITE-require attack (as a callee)

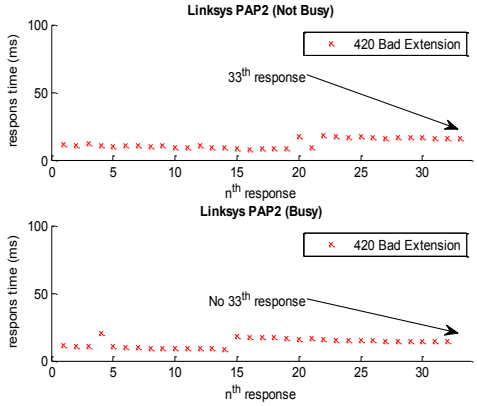

Figure 8. Responses of PAP2 under INVITE-require attack (as a callee)

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 a Require header whose functionality is not supported by the callee [10]. 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 [11].

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 is available) [12]. An event can be subscribed to in advance and then will be delivered to a SIP phone. (The Internet's IANA organization maintains the database

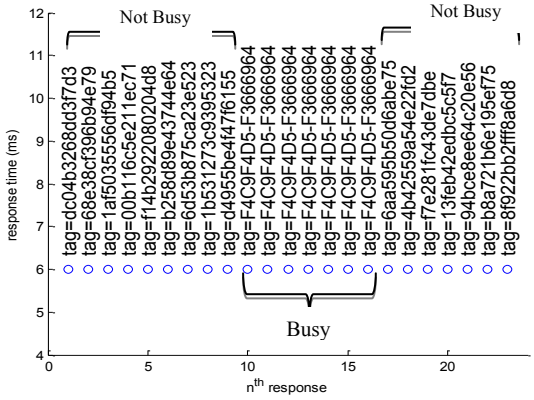

Figure 9. Responses of BT-100 under INVITE-SDP attack (as a caller)

of valid event parameters [9].) 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 parameters 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.

2.3 Call Correlation

Busy-status detection can detect the busy status of a phone at a specific time. However, depending on the phone type, the busy-status detection can determine 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 targeted 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 to 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 targeted

phones are both busy at the beginning and the end of the call such that $|x-i|<\varepsilon$ and $|y-j|<\varepsilon$.

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 busy-status detection, we show how the phones respond to the probe requests. For call correlations, we derive the detection rate.

3.1 Environment

We implemented a program named *Voice Pulser* to perform busy-status detection. Since the proposed attack does not require eavesdropping, we experimented on an ordinary LAN [14]. 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.

3.2 Busy-status Detection

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 8 shows how the PAP2 received all 33 responses when it was not busy and missed the 33rd response when it was busy. Figure 7 shows how the 7940G responded with an error '486 Busy Here' when the protocol buffer was full. Figure 9 shows the BT-100 response, which was slightly different. In the middle of the 23 probes, we made a call from the BT-100. According to RFC 3261, 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. 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 observations, the adversary is able to determine the busy status of the PAP2 by finding out whether the 33rd response is received, while the busy status of the 7940G is shown when the 6th response is a "486 Busy Here", and the busy status of the BT-100 is found by checking whether two consecutive responses have the same value for the To tag.

3.2.2 Attacks

We evaluated all types of attack messages listed in previous section and show the results in Table 1 and Table 2. Table 1 indicates whether the busy-status detection was successful. Table 2 provides details for the phones responses to the probe requests and how the busy status can be identified.

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.

Table 1. 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 2. Explanation of Table 1

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.)

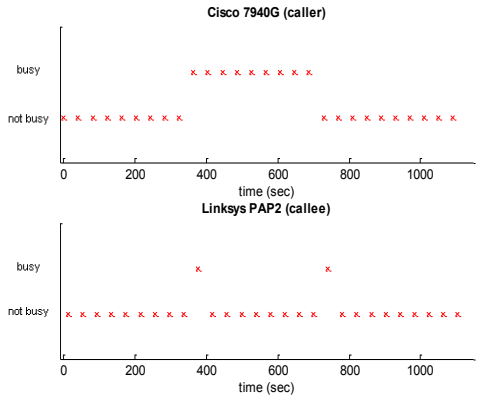

Figure 10. A call correlation experiment: 7940G calls PAP2 at 360sec and terminates at 700sec (7940G as call setup initiator and call termination initiator)

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: attack messages do not occupy the protocol buffer slots yet the busy status is still revealed.

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 3. 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).

The disabled period also affects the ring delay, which is the time between the call setup and ring. Table 3 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 this attack. For all three phones, the ring delay is essentially negligible for end-users unless they analyze the network traffic or phone logs.

3.3 Call Correlation

A phone has four possible roles. For 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).

First, we used the following steps to demonstrate a scenario for call correlation in Figure 10. 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 that it was busy when the call setup and call termination steps were performed.

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

Table 3. Timing behavior of busy-status detection

	LinkSys PAP2	Cisco 7940G	Grand-stream BT-100
Minimum Probing Interval	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

1. Use Voice Pulser to probe the PAP2 and the 7940G (or BT-100) concurrently for their busy status with a 34 second interval.
2. Use one of the phones to call the other.
3. Pick up 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 4 shows the results. For the detection rate, there was an outstanding case when the PAP2 received the call setup request. The detection 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 depended 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 4 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%.

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

Table 4. 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

Table 5. Detection rate of call correlation derived from Table 4

	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 4

4. COUNTERMEASURES

Busy-status detection relies on the protocol-buffer contention and undetectability of probing requests by a callee. Therefore, mechanisms that could prevent or neutralize protocol-buffer contention or enable probe detection by a callee would serve as effective defenses.

Protocol-buffer randomization. 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 the 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.

Firewall. An application-layer firewall could filter packets that exhibit probing patterns and limit the maximum probing rate. For example, the INVITE-require attack could be avoided by filtering INVITE requests with unsupported extensions. Such an application-layer firewall has an inherent disadvantage: it can only recognize busy-status detection attacks with known patterns. Another drawback is that application-level firewalls cannot handle encrypted communications such as the S/MIME used for SIP end-to-end encryption. Furthermore, the state maintained in the application-level firewall may become another side-channel. Another approach is to monitor and block suspicious periodic network traffic. Although it can be easily done, false detection is an issue.

Full protocol-buffer alert. Alerting the user when the protocol buffer is full is a simple way to "unblind" the previously undetectable busy-status detection probes. The SIP phone or the server could simply display a message on the screen, leave a voice message, or send an email to alert the user of a potential attack.

5. RELATED WORK

Current VoIP privacy attacks against low-latency anonymous network differ from our attack since they require much stronger, and largely impractical, adversary capabilities; e.g. global eavesdropping, full control of end-points, and malware insertion capabilities. Wright *et al*. showed that the text can be extracted from encrypted VoIP traffic by probabilistic methods [31]. Srivasta *et al.* confirmed the existence of VoIP calls by correlating flows, and required a global adversary who knows the number of flows between nodes (including VoIP nodes) of an anonymous network [26]. 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 [30]. Their attacks require the control of anonymous network end-points. In other attacks, the adversary has to resort to installing malware in the targeted phones [18]. However, such attacks could be defeated by firmware integrity checks (e.g. use of Trusted Platform Modules in phones) and would not scalable to more than very few targets. More interestingly, 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 phone call, which may only last for seconds or minutes [2, 22].

The idea of correlating the busy status of SIP phones with call records is also similar to that of associating users' online/offline status in Instant Messaging to infer friendship relations, as shown in the work of Resig *et al.* [25]. They hypothesized that if two users go online and go offline at about the same time, the two

users may be friends. In contrast with our work, Resig *et al*. assume that the user status (online/offline) was explicitly provided to the adversary by the users, while we must find the phone status (busy/not busy) in a manner that is undetectable by the targeted phone users.

The idea of obtaining remote host information by sending packets and analyzing responses has been used in the past, but in different ways with different goals. Operating-system fingerprinting can identify the operating-system versions based on how TCP/IP subsystem responds to attack packets [23]. For example, the widely-used tool *nmap* [6] sends out packets in TCP, UDP, and ICMP and analyzes the types of responses it gets, but not their timing. While some of these techniques do not alert the remote hosts in standard TCP/IP subsystems, they have very different goals than ours. Our attacks discover the internal state (busy status) of a network node, whereas operating system fingerprinting only distinguishes different types of network nodes.

In another form of fingerprinting, Gong, Kiyavash, and Borisov analyze the round-trip time to extract private information from remote hosts [8]. They sent ICMP packets to a DSL router and used the time series of response times to fingerprint the websites accessed through that router. Our attack also obtains remote private information through active probing, but differs from the technique of Gong *et al*., in three ways. First, the goal of our attacks is to obtain the call-records for targeted phones, whereas their goal was to identify the website accessed by a host behind a DSL router. Second, our attacks can penetrate application-level proxies while their ICMP packets cannot; i.e., the target's IP address is necessary for their attack, but not for ours. Third, we assume that every entry in the protocol buffer expires independently, whereas they use a FIFO queue to model how ICMP packets are processed.

Although we use resource exhaustion in a similar way as research on covert channels, ours is a side channel, *not* a covert channel, since we do not need a Trojan Horse program installed on a targeted phone to leak its busy status [18]. However, in contrast with traditional side-channel attacks used in cryptography [15, 16] and non-traditional ones used to detect clock skew in remote hosts [21], our attack correlates the side channels of multiple network nodes, rather than operating on a single node.

Our side-channel is based on resource-use detection and is somewhat reminiscent of the resource exhaustion attack in the TCP SYN flooding analyzed by Schuba *et al.* [27]. Both TCP SYN flooding and our attack send packets to fill a buffer.[5] However, these attacks differ in their execution and goals: our attack determines the current size of the buffers, whereas TCP SYN flooding fills a host buffer to disable the host.

[5] TCP is a three-way hand-shake protocol. The client sends a SYN packet to the server; the server responds with a SYN/ACK packet; finally the client sends ACK packets to the server. The TCP protocol stack needs to use or allocate a buffer for recording the context of each connection. When the adversary sends numerous SYN packets to a host, the host will run out of buffer slots eventually. Therefore the host cannot accept any new connection. In other words, TCP SYN flooding is a denial of service attack that disables a host by sending numerous TCP SYN packets.

6. SUMMARY

We proposed an attack for discovering call records in a VoIP service over a private network. We analyzed the SIP protocol and discovered that the array-based buffers of three commercially available closed-source hardware phones, which are used for storing protocol negotiation contexts, can be exploited to 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. To defend against such attacks, we suggested several countermeasures based on manipulating of the buffer, detecting full buffers and using firewalls.

7. ACKNOWLEDGMENTS

This research was supported in part by CyLab at Carnegie Mellon under grant DAAD19-02-1-0389 from the US Army Research Office. The first author was also partially supported by the MURI grant W 911 NF 0710287 from the Army Research Office. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government, or any other entity.

8. APPENDIX

Appendix I: Probe Messages

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:voicepulser@127.0.0.1:5000;tag=1236798926
To: sip:wj@iptel.org
Call-ID: 1811209364@127.0.0.1
CSeq: 1 OPTIONS
Contact: sip:voicepulser@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:voicepulser@127.0.0.1:5000;tag=524046249
To: sip:wj@iptel.org
Call-ID: 111248654@127.0.0.1
CSeq: 1 INVITE
Contact: sip:voicepulser@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

```
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:voicepulser@127.0.0.1:5000;tag=1552686011
To: sip:wj@iptel.org
Call-ID: 1436339943@127.0.0.1
CSeq: 1 INVITE
Contact: sip:voicepulser@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. 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:voicepulser@127.0.0.1:5000;tag=499998901
To: sip:wj@iptel.org
Call-ID: 1599885472@127.0.0.1
CSeq: 1 NOTIFY
Contact: sip:voicepulser@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 II: Calculation of Disabled Period

We illustrate the timing of busy-status detection and calculate the disabled period, as shown in Figure 11. 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 [10].

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

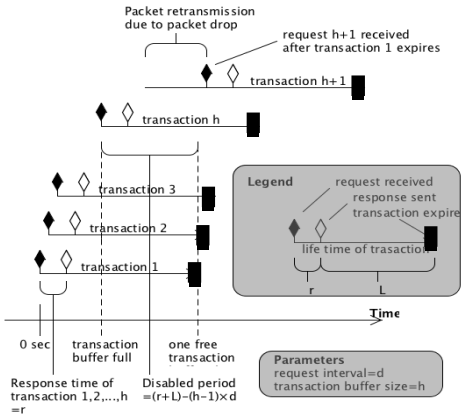

Figure 11. Disabled period

(*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.

9. REFERENCES

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

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

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

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

[5] 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

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

[7] Gilbert, E., and Karahalios, K. "Predicting Tie Strength With Social Media," In Proc. of ACM CHI 2009, April 2009, Boston, MA

[8] Gong, X., Kiyavash, N. and Borisov, N., "Fingerprinting Websites Using Remote Traffic Analysis," In Proc. of ACM CCS, 2010, Chicago, IL

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

[10] IETF RFC 3261, "SIP: Session Initiation Protocol"

[11] IETF RFC 3264, “An Offer/Answer Model with the Session Description Protocol (SDP)”

[12] IETF RFC 3265, “Session Initiation Protocol (SIP)-Specific Event Notification”

[13] ITU-T G.114, One-way transmission time

[14] Jong, C.-H., Voice Pulser SIP attack program, https://code.google.com/p/voice-pulser/

[15] Kocher, P.C. “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems,” In Proc. Of CRYPTO, 1996, Santa Barbara, CA

[16] Kocher, P.C., Jaffe, J., and Jun, B. “Differential Power Analysis,” In Proc. of CRYPTO, 1999, Santa Barbara, CA

[17] Kazatzopoulos, L., Delakouridis, C., and Marias, G.F. “Providing Anonymity Services in SIP,” In Proc. of IEEE PIMRC, 2008, Cannes, France

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

[19] Markoff, J. “Taking Spying to a Higher Level,” *New York Times*, Feb. 2006.

[20] McKeay, M. ‘Taking Corporate Spying to a Higher Level,” Computerworld, 2006.

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

[22] Murdoch, S. J. and Danezis, G. “Low-Cost Traffic Analysis of Tor,” In Proc. of the IEEE Symposium on Security and Privacy, 2005, Oakland, CA

[23] Padhye, J. and Floyd, S. “On Inferring TCP Behavior,” In Proc. of ACM SIGCOMM, 2001, San Diego, CA

[24] 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

[25] Resig, J., Dawara, S., Homan, C. M., and Teredesai, A. “Extracting Social Networks from Instant Messaging Populations, “ In Proc. Of LinkKDD, 2004, Seattle, WA

[26] 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

[27] Schuba, C. L., Krsul, I. V., Kuhn, M. G., Spafford, E. H., Sundaram, A., and Zamboni, D. “Analysis of a denial of service attack on TCP, “ In Proc. of IEEE Symposium of Security and Privacy, 1997, Oakland, California

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

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

[30] Wang, X., Chen, S., Jajodia, S. “Tracking Anonymous Peer-to-Peer VoIP Calls on the Internet,” In Proc. of ACM CCS, 2005, Alexandria, VA

[31] Wright,C., Ballard, L., Coull, S., and Monrose, F. "Spot Me If You Can: recovering spoken phrases in encrypted VOIP conversations," In Proc. of IEEE Symposium on Security and Privacy, May, 2008, Oakland, CA