ABSTRACT

ZHOU, YUZHENG. Mitigating Voice over IP Spam Using Computational Puzzles. (Under the direction of Associate Professor Peng Ning).

Voice over IP (VoIP) has gained increasing popularity in recent years. With the growing of its deployment, VoIP will become the target of hackers and crackers. As organizations plan and deploy VoIP networks, VoIP spam should be considered as a very real threat and proactively addressed as part of the overall security strategy. However, preventing VoIP spam is a complex problem and requires many techniques working in combination to reduce spam to acceptable levels while still allowing efficient communication. White/black list is a popular and important technique of dealing with spam. A system would first check whether an incoming request is from someone on the white/black list before establishing the communication between the two parties. However, there still needs a way for callers who are not on the white/black list to communicate with the callee. To address this problem, this thesis develops a method combining white/black list with the computational puzzle mechanism, which permits anonymous call establishment while mitigating undesirable contacts. Compared with other VoIP anti-spam methods, this method has three main properties. Firstly, user disturbance by the spam is light because this method is applied before the phone rings. Secondly, few user maintenance is required since the maintenance could be located at the service provider only. Thirdly, it increases the costs for the spammer, making spam less profitable. A prototype system is developed on the basis of Mjsip (an open source VoIP software) for performance evaluation. Our experiments show that our method can effectively slow down the spammers with only a light cost to innocent anonymous callers and proxy servers.

Mitigating Voice over IP Spam Using Computational Puzzles

by

Yuzheng Zhou

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Computer Science

Raleigh, North Carolina

2007

Approved By:

Dr. S. Purushothaman Iyer

Dr. Ting Yu

Dr. Peng Ning
Chair of Advisory Committee

Dedication

I dedicate this thesis to my parents and my wife without whom none of this would have been even possible.

Biography

Yuzheng Zhou received his Bachelor's degree in Computer Science from Tianjin University, Tianjin, China in 2001. Since 2004, he has been a graduate student in the Department of Computer Science at North Carolina State University.

Acknowledgements

With a deep sense of gratitude, I wish to express my sincere thanks to my advisor, Dr. Peng Ning, for the stimulating advice and patience that he had for me during my graduate study. The training I received from him is priceless. I would also like to thank my committee members Dr. S. Purushothaman Iyer, and Dr. Ting Yu for devoting time and providing valuable inputs.

My former colleagues in the Cyber Defense Laboratory supported me in my research work. I want to thank Dingbang Xu, Pai Peng, Kun Sun, Yai Zhai, Pan Wang, Fang Feng, Qing Zhang, Yi Zhang, Qinghua Zhang, An Liu and Ting Wang for their help during my study in the lab.

Finally, I am deeply grateful to my parents, my aunt as well as my lovely wife. This thesis work would not be possible without their continued support and encouragement during the past years in my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

IP telephony, commonly known as Voice over IP (VoIP), is emerging as a viable alternative to traditional telephone systems. With the increasing popularity of its deployment, VoIP cannot avoid becoming the target of hackers and crackers. VoIP may suffer threats from different protocol layers. From bottom to up, malicious attackers may exploit the misconfiguration of devices, the vulnerability of the underlying operating systems, and the protocol implementation flaws to break in. Well-known attacks of data networks such as worms, viruses, Trojan horse, denial-of-service (DoS) attacks can also plague VoIP network devices.

Among the various attacks, VoIP spam would be one of the most annoying ones. Most email users are familiar with receiving spam on a daily basis. They have to use filtering software to block the messages as well as check junk mail folders periodically for false positive. But imagine your VoIP phone rings all day, only to have dozens of messages offering you discounted prescription drugs or an opportunity to start a home-based business. Unfortunately, it is now possible, with sophisticated spammers looking toward VoIP phones as their latest targets.

VoIP spams incidents has already been reported in Japan, where VoIP is more highly developed. In one case, a major VoIP provider, SoftbankBB, found three incidents of VoIP spams within its network, which included unsolicited commercial messages for an adult Web site and illegitimate requests for personal information [3].

Moreover, compared to email spam, VoIP spam is more obtrusive [20]. With email, the user decides when he is going to read his messages. But with VoIP, the phone will ring with every spam message, even in the middle of the night when you are asleep. In addition, we should note that launching VoIP spam is naturally easier and cheaper. VoIP uses Internet instead of traditional PSTN (Public Switched Telephone Network) networks as its carrier, making it easier for spammers to utilize automated tools to deliver their spam to the users [17]. VoIP is also a lot cheaper compared to traditional PSTN telephony, in terms of both the monthly fee as well as the costs per call. The authors of [25] claim that the costs per call for VoIP are roughly three orders of magnitude cheaper than traditional PSTN calls, making it a lot cheaper for a spammer to get his message out into the world.

As VoIP deployment continues to accelerate and technology is now available to block unwanted emails, it is reasonable to expect that VoIP spams will quickly follow. As organizations plan and deploy VoIP networks, VoIP spam should be considered as a very real threat and proactively addressed as part of the overall security strategy.

1.2 Motivation

In recent years, numerous techniques and technologies have been developed to fight email-based spam including anti-spam appliances, client based filters, white/black list technologies, and social networks, all of which provide a strong starting point for combating VoIP spam. However, the significant differences between voice and other data indicate existing email anti-spam solution could not be directly applied to combating VoIP spam.

A voice call consists of two phases, signaling phase and media session phase. In the signaling phase, information is exchanged between the VoIP network and end-users that establish a call. In the media session phase, voice conversation is carried over media stream, in most cases, directly between the end-users.

In fact, preventing VoIP spam is usually performed in the signaling phase. This can be accounted to two reasons. Firstly, obtaining caller identity is easier in the signalling phase. Basic information about the identity of the end-users could be relatively easily obtained by analyzing the content of the signalling protocols including source addresses, country of origin and call patterns. White/black list is a representative anti-spam technology based on caller identities. It allows calls to pass through when callers are on the white list whereas reject calls when callers are on the black list. However, this kind of method alone is not sufficient to combat VoIP spams because it cannot handle the callers not in white/black list. Secondly, unlike email, it is hard to analyze voice content on the fly, and it is infeasible to use content filtering method on voice. In the case of email spam, additional information such as subject and content of the email are available for analysis, and spam can be identified before it reaches end-users. In the world of VoIP spam, that information is not easily obtainable. Theoretically, it may be possible to collect all the VoIP packets carrying the conversation, reassemble them into speech and then analyze the content. However, this is a very difficult technical problem that cannot be addressed by the existing technology. Even if there is an effective method to analyze voice content, the call has to be established first to get the voice content and the user has already paid attention before the content is delivered. In a word, VoIP spam cannot be analyzed before a user answers the call.

To address this problem, this thesis develops a method combining white/black list with computational puzzle, which permits anonymous call establishment while mitigating undesirable contacts. Compared with other VoIP anti-spam methods, this method has three main properties. Firstly, user disturbance by the spam is light because this method is applied before the phone rings. Secondly, few user maintenance is required since the maintenance could be located at the service provider only. Thirdly, it increases the costs for the spammer, making spam less profitable. With the help of white/black list, our method can efficiently mitigate VoIP spams. We believe this method can be easily integrated into current VoIP products working as either a separate VoIP anti-spam solution or a component in other VoIP anti-spam solutions.

1.3 Contributions

The scientific contributions of this thesis are threefold:

1. The thesis describes the design and implementation of a VoIP spam mitigating method based on computational puzzles. Using computational puzzle is not novel, but to our knowledge, we are the first to complete both the design and the implementation of a computational puzzle based VoIP anti-spam method.

2. The thesis compares and classifies existing VoIP anti-spam methods, and analyzes their advantages and disadvantages.

3. The thesis evaluates the effectiveness of the integration of white/black list and the proposed VoIP anti-spam method via several metrics, including the added cost for proxy server and normal caller, the call generating ability of VoIP spammers, spam call block rate, spam ratio (percentage of successfully accomplished VoIP spam calls in all the VoIP calls). We also compare the performance of our system using different puzzle cost assignment methods and using different spammer strategies. The experiments show that our method can effectively slow down the spammers with only a light cost to innocent anonymous caller and proxy server.

1.4 Thesis Organization

In Chapter 2, we review Voice over IP architecture and the existing VoIP anti-spam methods. In Chapter 3, we describe in details our computational puzzle based VoIP anti-spam method. In Chapter 4, we explain the implementation of our method in details. We present our experimental results in Chapter 5, followed by our conclusion and discussion of future work in Chapter 6.

Chapter 2

Background Study

2.1 Voice over IP Architectures

In a simplistic view, the Voice over IP technology has the goal of establishing and managing communication sessions for transmitting voice data over standard IP networks. A stable and reliable transmission has to be maintained all throughout the conversation, and the media session needs to be ended when either of the parties decides to terminate the call.

Figure 2.1 illustrates the protocol stack for a VoIP system. In the application layer, two types of protocols are used by VoIP technology, in a similar manner to standard telephony: signaling protocols (such as H.323 [26] and SIP) and media transport protocols (such as RTP [10]). Quality of Service (QoS) protocols can also be used by VoIP technology to guarantee the service performance, but they are not required. In the transport layer, both TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) can be used by signaling protocols. And the media transport protocols, such as RTP, are based on UDP. As for the network and lower layers, VoIP uses the same protocol suites as current IP networks.

Currently, two VoIP architectures have emerged as standards and are widely deployed throughout the world. They are ITU-T Recommendation H.323 and the IETF SIP (Session Initiation Protocol) [13] architecture. Their names reflect the signaling protocols

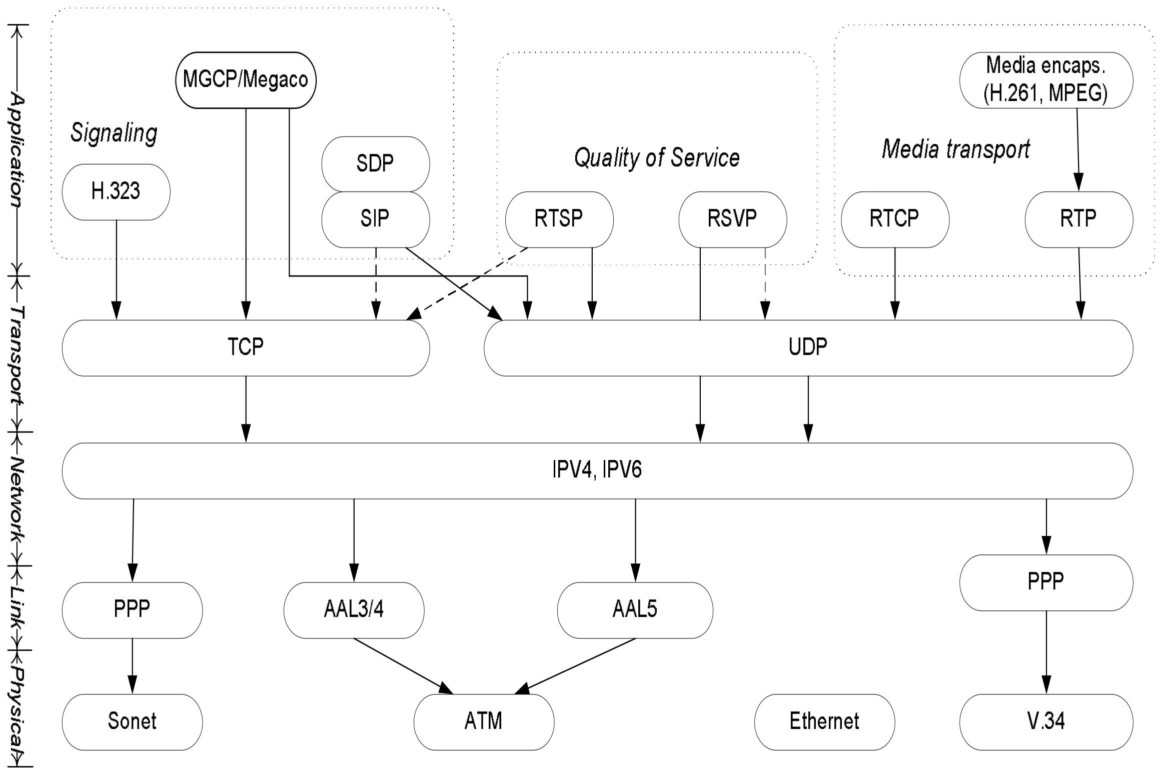

Figure 2.1: VoIP Protocols Stack

they use.

H.323 is the International Telecommunication Unit (ITU) standard for audio and video transmissions over packet based networks. It is actually a wrapper standard, encompassing several other protocols, including H.225, H.245, and etc. Each of these protocols has a specific role in the call setup process. The components of an H.323 network are: several endpoints, a gateway, and possibly a gatekeeper (for address resolution and bandwidth control), Multipoint Control Unit (for multi point conferencing), and Back End Service (for storing and maintaining configuration data about endpoints). H.323 uses RTP as standard protocol for media transport.

SIP is the Internet Engineering Task Force (IETF) protocol dedicated to initiating two-way communication sessions. SIP is not specific to VoIP and it could be used in any session driven technology/application. SIP is text based, similar to HTTP. It can be carried by TCP, UDP, or SCTP. TCP and UDP are the two most common transport layer

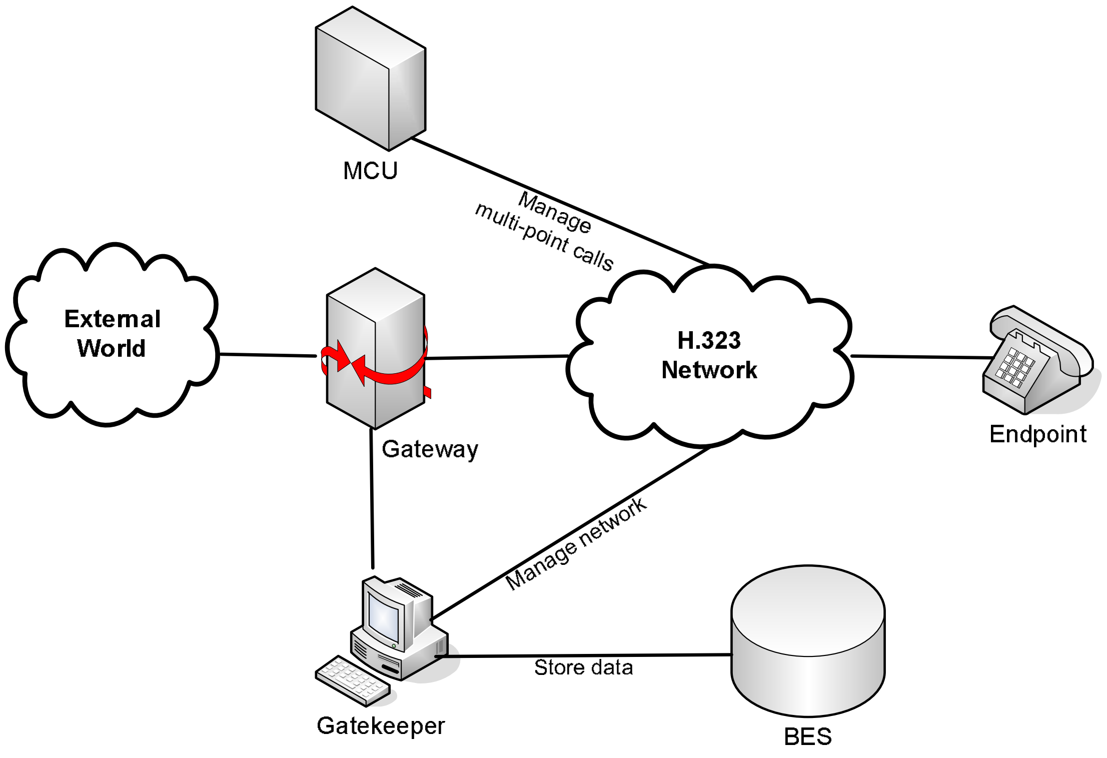

Figure 2.2: H.323 Architecture

protocols. TCP is connection oriented and provides reliable transmission, and it could be used if SSL/TLS2 [7] is incorporated for security services. UDP is connectionless, but it may be used to decrease overhead and increase speed and efficiency. SIP uses one network port with the default value 5060.

Different from the architecture of the H.323 network, a SIP network is made up of endpoints (also called User Agents), a proxy and/or redirect server (for endpoint message transmission), a location server (for locating users), and a registrar (for registering location information). The registrar and the location server may be integrated into the proxy server.

SIP architecture diagram is provided in Figure 2.3. In the SIP model, a user is not bound to a specific host. The users initially report their location to a registrar, which may be integrated into a proxy or redirect server. This information is in turn stored in the external location server. Messages from endpoints must be routed through either a proxy or a redirect server. The proxy server intercepts messages from endpoints or other services, inspects their "To:" field, and then contacts the location server to resolve the username into an address and forwards the message along to the appropriate endpoint or another server. Redirect servers perform the same resolution functionality, but the bonus is placed on the endpoints to perform the actual transmission. That is, redirect servers obtain the actual address of the destination from the location server and return this information to the original sender, which then must send its message directly to this resolved address.

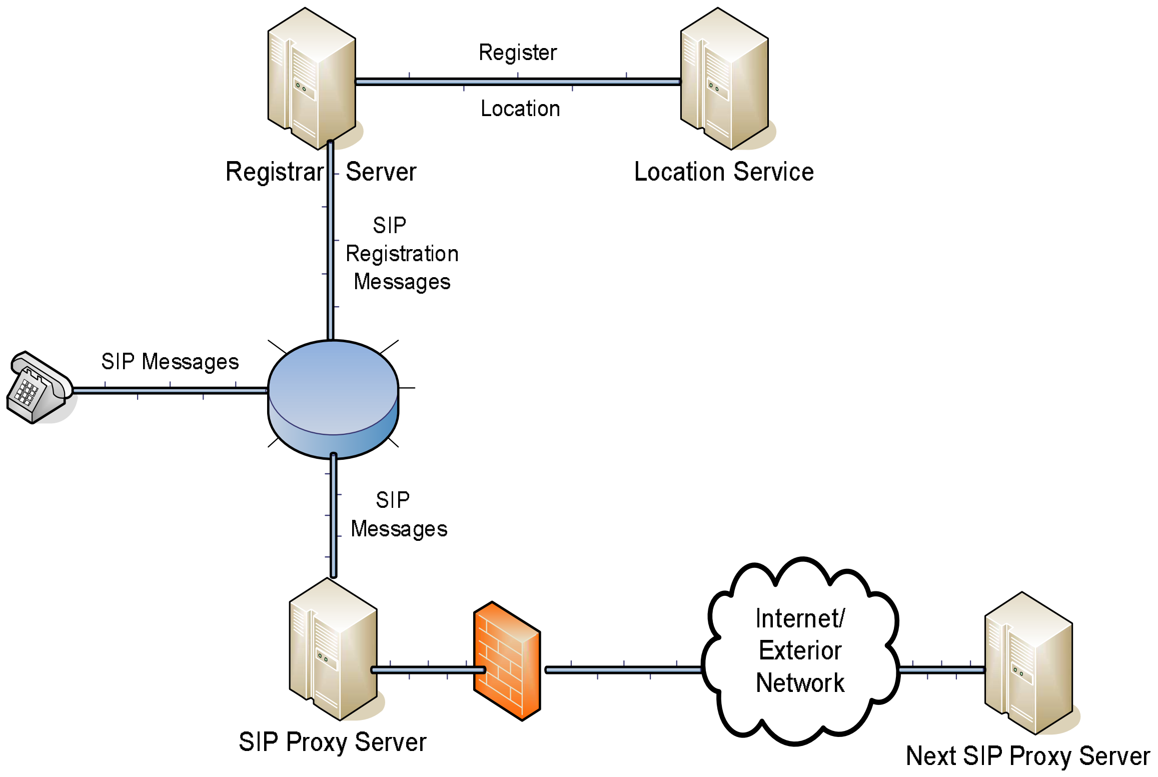

Figure 2.3: SIP Architecture

The two VoIP architectures are quite different, and both signaling protocols are equally vulnerable for spam because of implementation errors and protocol features that are exploited [9]. It is reasonable to choose only one of them as the target architecture. In the rest of this thesis, the terms VoIP architecture or system represents the SIP VoIP architecture, unless explicitly stated otherwise.

2.2 Voice Over IP Spam

Spam, defined as the transmission of bulk unsolicited email, has been a plague on the Internet email system. Similar to email systems, VoIP systems are susceptible to abuse by malicious parties who initiate unsolicited and unwanted communications. As the popularity of VoIP and its deployment grows, VoIP spam, also known as SPIT (for "Spam over Internet Telephony"), has been received a great deal of attention from marketers and the trade press. Figure 2.4 shows a typical example of VoIP spam.

According to the different intensions of the attacks, VoIP spams can be classified into three categories: Call spam, IM spam and Presence spam [25].

- Call spam is defined as a bulk unsolicited set of session initiation attempts (i.e., INVITE requests), attempting to establish a voice, video, instant messaging [2] or other types of communication sessions. This is the classic telemarketer spam applied to SIP.
- IM spam is defined as a bulk unsolicited set of instant messages, whose contents contain the messages that the spammer is seeking to convey.
- Presence spam is similar to IM spam. It is defined as a bulk unsolicited set of presence requests (i.e., SUBSCRIBE requests [22] for the presence event package [23]), in an attempt to get on the "buddy list" or "white list" of a user in order to send them IM or initiate other forms of communications.

IM spam is quite similar to email spam. And all the effective email anti-spam methods could be possibly used for IM spam. Presence information is important for users, and the presence requests are not as popular as calls and IM requests. Therefore, expensive authentication methods, such as digital signature, are good solutions for presence spam. Although the thread of call spam is similar to email spam, its impact, in terms of interference with VoIP users' daily life, is more severe than email spam. Without any prevention from call spam, the normal VoIP user may receive hundreds of calls with publicity messages, and the phone may ring all day. Also, the unwanted contact may be subjective. You may be easily get irritated when you receive a spam call, maybe a business advertisement, at 3:00 am. Furthermore, the significant differences between voice and other data indicate that currently effective email anti-spam solutions may not be useful for call spam. For example, due to the inmature voice recognition technology, the content-based filtering method cannot be easily used as an anti-call spam method. Therefore, in this thesis, we will focus on anti-call spam solutions only. For the rest of this thesis, VoIP spam has the equivalent meaning of call spam, unless explicitly stated otherwise.

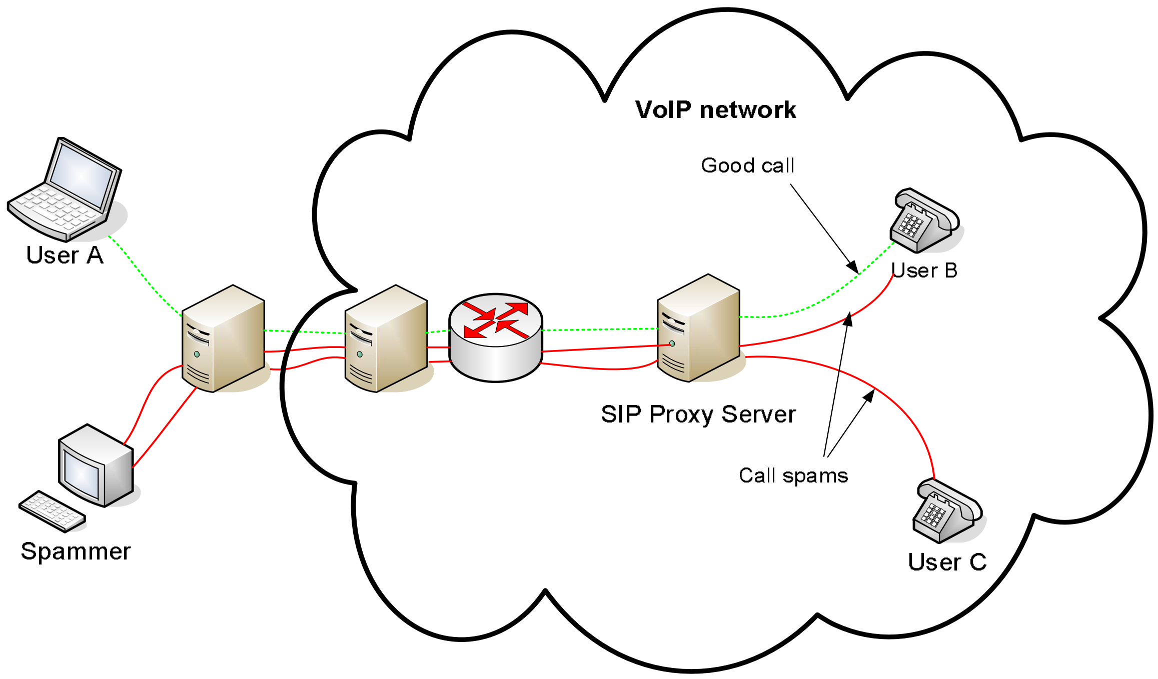

Figure 2.4: Voice over IP spam

2.3 Voice over IP Spam Prevention Methods

Currently, email anti-spam solution is still an open research area. The difficulty of voice analysis makes VoIP anti-spam even more difficult than email anti-spam. Since the VoIP spam prevention problem is complex, it will require many techniques working in combination to reduce spam to acceptable levels while still allowing efficient voice communication for normal users. In this section, we will first discuss the criteria for evaluating VoIP anti-spam methods. Then we classify and review existing VoIP anti-spam methods, and discuss their advantages and disadvantages.

To evaluate a VoIP spam prevention method, we need a number of criteria. Firstly, user disturbance should be reduced as much as possible. Unlike email spam, VoIP spam will not lie down there silently, instead, it causes interruptions to users' current activities. It would be a loss of productivity if spam has to be identified after interruptions. Secondly, it is desirable to have the least user intervention, since users' participation itself is a kind of interruptions. Thirdly, the cost added by the VoIP anti-spam method for the users is also an important aspect. It is preferable that a spam prevention method involves as less costs as possible for users. In contrast, it is also preferable for a protection technique to introduce as much as possible costs for spammers, then spamming will be less profitable and eventually becomes less [11]. Fourthly, the number of false positives and false negatives should be as less as possible, preferably even zero, for a spam protection technique. VoIP phone users would not want to lose any potential customers of their home businesses just because of spam. Fifthly, it is ideal that a spam protection technique would cause the least delay. And last but not least, a spam prevention method should be resistant to circumventing. The method would be totally ineffective if it can be easily circumvented by spammers. An good spam protection technique should be both effective and very difficult to circumvent.

According to the different ways of interactions between the anti-spam methods and callers/callees, VoIP anti-spam methods can be classified into four categories, which are illustrated in Figure 2.5. We will discuss these methods in the following subsections.

2.3.1 No Interaction With Call Participants

Because there are no interaction with both the caller and callee, this kind of VoIP spam prevention methods are very convenient and highly encouraged. White/black list is a typical example. Calls from identities in the white list are allowed, whereas, calls with callers from the black list are rejected. Based on the white/black list, the VoIP system could easily accept/deny an incoming call.

Although the idea of white/black list is simple and efficient, it is not an ideal method to solve the VoIP spam problem. Usually, white/black list is used as the first barrier for VoIP spams. There are three disadvantages of the white/black list, which would degrade the spam filtering effects. Firstly, a powerful identity mechanism is required to guarantee that calls are really from the caller as declared in the incoming call. Currently, IETF is working on the SIP identity standardization [21] and this will help identify callers. Secondly, white/black list only works for known identities. VoIP system cannot use white/black list

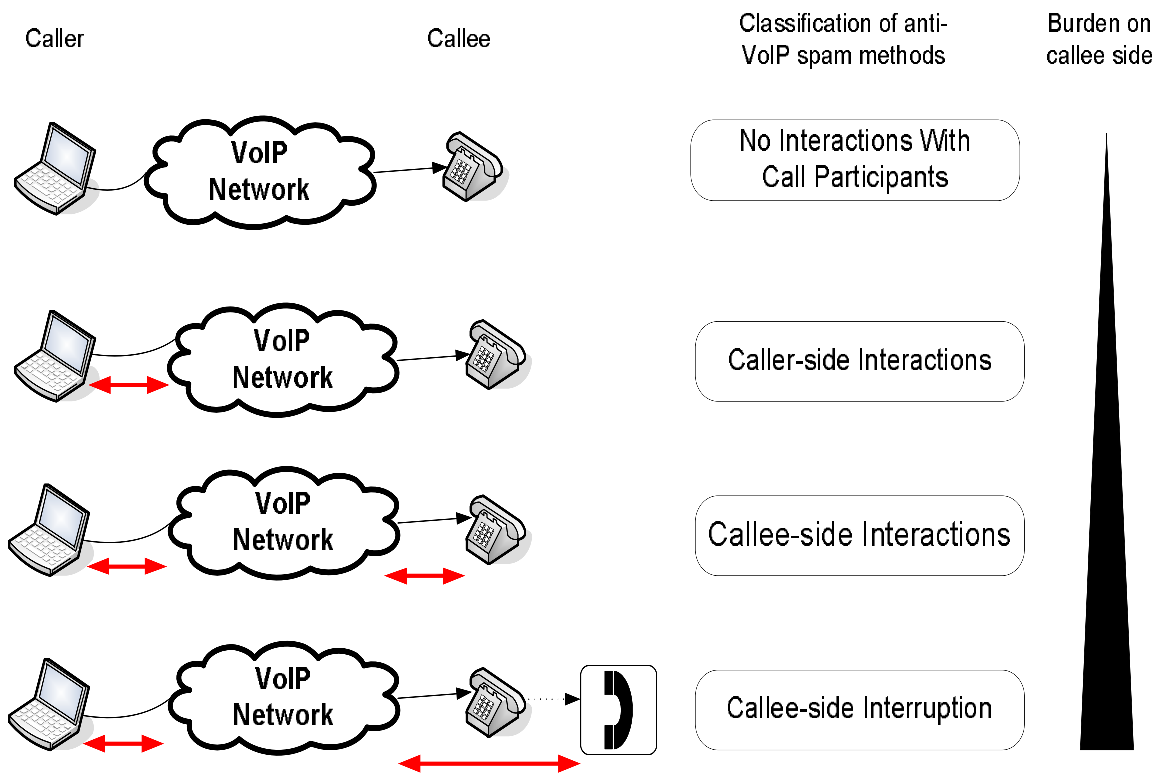

Figure 2.5: Classification of VoIP anti-spam methods

to allow or deny a call from an anonymous caller, since the caller is neither on the white list or on the black list. Moreover, this does not meet the criterion that the technique needs to work before the phone rings, because before the user can add somebody to his blacklist he first needs to answer the call to decide if it is spam. Thirdly, white/black list can be easily circumvented. Whenever the spammer obtains a new phone number that is not on the blacklist, it can continue to launch spams until the new number is added to the blacklist.

2.3.2 Caller-side Interactions

The methods in this category involve interactions with callers. For example, the caller may be required to answer some simple questions before the call gets through. Although these methods are not as perfect[1] as those in the previous subsection, they are good supplements and will make the whole VoIP anti-spam system stronger. Also, concerning

[1]For perfect, we mean neither caller or callee is bothered by the method.

that VoIP spam prevention is used to protect callee, certain amount of caller-side interaction would be acceptable.

Reverse Turing test is a typical caller-side interaction method. In the world of VoIP, reverse Turing tests are those solutions whereby the caller is given some kind of challenges, which only a human can answer. These tests are also known as captchas (Completely Automated Public Turing test to tell Computers and Humans Apart). These challenges frequently take the form of answering a question or repeating numbers while background music or noise is playing, making it difficult for an automated speech recognition system to be applied to the media. Only by entering the correct numbers with their keypad, the users can make their call accepted.

Reverse Turing tests fulfill most of the criteria defined previously. They are taken before the phone rings at the callee side. They cost spammers a huge amount of computer resources to circumvent. Even in the situation that a spammer is able to circumvent the reverse Turing test, in the way that he can provide the correct answer to the asked question, reverse Turing tests can be easily upgraded with more difficult questions. However, reverse Turing tests expect a certain level of knowledge for the users. For example, some reverse Turing tests require the caller to do some arithmetics and return the result. If the question is too difficult, a child would not make a call due to not being able answer the question. However, if the question is too simple, spammers would be able to answer the question with a voice recognition system, making the technique ineffective. Another problem with the voice reverse Turing test is that: instead of having an automata process to take the test, a spammer may afford paying cheap workers to take the tests. Thirdly, the language used for the reverse Turing tests may become a problem for large international companies, since not all customers speak the same language.

2.3.3 Callee-side Interactions

Some anti-spam methods require the participation of callees. Identifying a spam has to be at the cost of callee interaction. However, once the callee reacts to a spam call, the spammers might have partially achieved their goals. Therefore, methods in this category are not recommended as separate solutions for VoIP spam prevention, but they can be good supplement for other methods.

Consent-based communication [21] [24] is a representative for this category. It is used in conjunction with white/black lists. Suppose caller A is not in callee B's white/black list, and caller A attempts to communicate with callee B. Then caller A's attempt will be initially rejected, and caller A is told that consent is being requested. Next time callee B connects, callee B is informed that caller A had attempted communications. Callee B can then authorize or reject caller A. It would seem this method can help the VoIP spam prevention a lot. However, it might just change the nature of the spam. Instead of being bothered by content, users are bothered by consent requests.

2.3.4 Callee-side Interruption

The methods falling into this category do not meet the criterion that the technique works before the phone rings. In fact, when the callee accepts the spam calls, the spammers have achieved their goals. However, even at this time, the callee can still protect himself from VoIP spams in the future.

Reputation filtering [5] is a typical representative for this category. Reputation systems can be implemented in highly centralized VoIP identification system. Each user who receives a spam call could report the caller to the VoIP spam identification system. Individual reputation will be calculated based on users' feedback. Although the input of a single user is insufficient to ruin one's reputation, but consistent negative feedback would give the abusive user a negative reputation score. Reputation filtering also does not fulfill minimal user intervention criterion, since for every anonymous caller, the user has to decide if he wants to allow the call based on the reputation of the caller. Once the caller is accepted, he has to be added to the user's white list. This might be plenty of work for users who receive a lot of calls from anonymous callers.

2.4 Computational Puzzle

Computational puzzle, as an email anti-spam method, is a challenge response method that requires a sender (or a sender's computer) to assist authentication of a given request by solving a simple puzzle. While this activity takes little effort for a regular sender, it would overwhelm the computing cycles for someone sending large amounts of bulk emails.

Our VoIP anti-spam system is based on this technology. Here we only briefly introduce the basic idea and related application of computational puzzles, and leave the detailed explanation of our method in chapter 3.

The basic idea of computational puzzle is: "If I do not know you and you want to send me a message, then you must prove that you spent, say, ten seconds of CPU time, just for me and just for this message" [4]. This "proof of effort" is mainly cryptographic, which is hard to compute but very easy to check. The idea of using computation to restrict abuse dates back to Dwork and Naor [8], who suggested using it to fight spam. It works as follows: to send a message to a destination, an email sender takes a hash of the message, the current time, as well as the address of the destination, and then feed it as input to a moderately hard pricing function: a function that is moderately difficult to compute but the result of which is easy to verify. The output of the function would be included with the message, along with the time used in the function. An email recipient has all the information for verifying the result, and would only accept a message passing the verification. By requiring a per message pricing function computation, this method imposes a little cost on legitimate email users but a heavy one on spammers. Juels and Brainard [16] proposed a similar scheme for preventing denial of service attack. They suggested issuing client puzzles to anyone requesting service and requiring a solution before the service is provided. The puzzle includes a random input to the pricing function, so the method is an interactive puzzle protocol. Computational puzzle based method is an efficient email anti-spam solution. For example, in [27], computational puzzle is used to throttle outgoing email spams at the email server provider side.

However, one defect of computational puzzle would undermine its application in email anti-spams. Computational puzzle system cannot tell whether the source address is the real sender or a forged address. Sending puzzles to forged or unknown address is equivalent to sending messages to an innocent third party. This is called Joe Job [1]. Forging the "From" address field is not rare. Spammers do not put their addresses into either the From or other fields of their emails. So they never get puzzles at all. However, the difference between VoIP and email communication mechanisms makes VoIP system immune from this defect. Spammers in email systems may not care the responses from receivers. Therefore, spammers can use the email addresses of innocent third parties as the forged sender address. In contrast, three-way handshake (via SIP protocol) is used to build connection between callers and callees in VoIP system. If spammers use an innocent address which is out of their control as a forged caller, they would not receive the SIP response from the callee. Then the media connection for this call would not be established. This characteristic makes computational puzzles more suitable and more efficient for VoIP systems.

Chapter 3

Computational Puzzle Based VoIP Anti-spam System

VoIP spam prevention is a complex problem and requires many techniques working together to reduce spam to acceptable levels while still allowing efficient communication. To mitigate the VoIP spams, we propose a mechanism that reduces VoIP spams by adaptively assigning computational costs to the callers based on their previous behavior. This mechanism can work either as a separated VoIP anti-spam solution, or as a supplement for other VoIP anti-spam solution.

In this chapter, we describe this computational puzzle based VoIP mitigation mechanism. We first review the SIP message flow of the VoIP systems, followed by the description of the system architecture of our VoIP anti-spam method. Next, we present the puzzle generation and verification procedure in details. Then we describe our adaptive cost assigning method used to adjust costs for different callers.

3.1 SIP Call Flow Diagram

SIP (Session Initiation Protocol) is a signaling protocol used for establishing sessions in an IP network. SIP protocol incorporates elements of two widely used Internet protocols: Hyper Text Transport Protocol (HTTP) for Web browsing and Simple Mail Transport Protocol (SMTP) for email. From HTTP, SIP borrowed a client-server design and the use of URLs and URIs. From SMTP, SIP borrowed a text-encoding scheme and header style. SIP reuses SMTP headers such as To, From, Date, and Subject.

In this section, we will briefly describe the call flows in SIP invitation dialog, which is used for making and accepting call requests. It is useful for understanding section 3.2. Usually the best way to learn a protocol is to look at examples of its use, because an example message flow can give a snapshot of some of the key concepts of a protocol. We will use two typical examples [15] to explain how SIP protocol works as a signaling mechanism in VoIP system. The examples will be explained using call flow diagrams between a callee and a caller, along with the explanations of each message. Each arrow in the figures represents a SIP message, with the arrowhead indicating the direction of message transmission. The thick lines in the figures indicate the media stream. In both examples, the media will be assumed to be RTP packets containing audio, but it could be another protocol.

3.1.1 SIP Call Without Proxy Server

Figure 3.1 shows a simple SIP session establishment process. In this example, all SIP messages are exchanged between two SIP-enabled devices. These two devices could be SIP phones, laptops, or cell phones, and both devices are connected to an IP network such as the Internet and they know each other's IP addresses.

Firstly, Alice, as the caller, initiates the message exchange by sending a SIP INVITE message to the callee, Bob. INVITE is an example of a SIP request message. The INVITE message contains the detailed information about the type of call that is requested. It could be a simple voice session, or a multimedia session such as a video conference. The INVITE message header contains the following fields:

```
INVITE sip:bob@receiver.org SIP/2.0
Via:SIP/2.0/UDP sender.org:5060; branch=z9hG4bKfw19b
Max-Forwards:  70 To:  Bob <sip:bob@receiver.org>
From:  Alice <sip:alice@sender.org>;tag=76341
Call-ID: 123456789@receiver.org
CSeq:  1
Contact:  <sip:alice@sender.org>
```

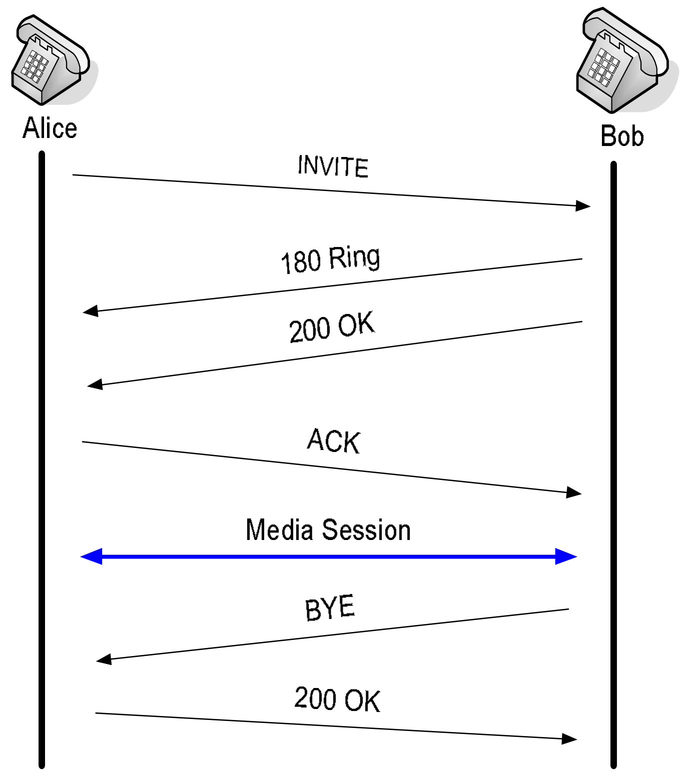

Figure 3.1: Simple SIP call flow

```
Content-Type:  application/sdp
Content-Length:  158
```

The fields listed in the INVITE message are called header fields. The first line of the request message, called the start line, lists the method, which is INVITE, the Request-URI, followed by the SIP version number (2.0), all separated by spaces. The Call-ID header field is a globally unique identifier and used to keep track of a particular SIP session. The originator of the request creates a locally unique string, then usually adds an “@” as well as its host name to make it globally unique. In addition to the Call-ID, each party in the session also contributes a random identifier, unique for each call. These identifiers, called tags, are included in the To and From header fields as the session is established. The shown initial INVITE message contains a From tag but no To tag.

After callee Bob received the INVITE message, he will send a 180 Ringing message back to the caller Alice. When Bob decides to accept the call (i.e., the phone is answered), a 200 OK response is also sent to Alice. This response also indicates that the type of media session proposed by the caller Alice is acceptable. Both the 180 Ringing and 200 OK are examples of SIP response messages. Responses are numerical and are classified by the first digit of the number. A 180 Ringing response is an "informational class" response, identified by the first digit being a 1. "informational class" response is used to convey noncritical information about the progress of the call. A 200 OK is a "success class" response, identified by the first digit being a 2.

The 200 OK message header fields are similar to INVITE message. It is created by copying many of the header fields from the INVITE message, including the Via, To, From, Call-ID, and CSeq, then adding a response start line containing the SIP version number, the response code, and the reason phrase. Also, the To header field now contains a tag that was generated by Bob. All future requests and responses in this session will contain both the tag generated by caller Alice and the tag generated by callee Bob.

```
SIP/2.0 200 OK
Via:  SIP/2.0/UDP sender.org:5060;branch=z9hG4bKfw19b ;received=100.101.102.103
To:  Bob <sip:bob@receiver.org>;tag=a53e42
From:  Alice <sip:alice@sender.org>;tag=76341
Call-ID: 123456789@sender.org
CSeq:  1 INVITE
Contact:  <sip:bob@receiver.org>
Content-Type:  application/sdp
Content-Length:  155
```

Then caller Alice can confirm the media session with an ACK request. The confirmation means that caller Alice has successfully received Bob's response. After the media session is established, either the caller or the callee can originate the BYE request to terminate the session. And the originator acts as the SIP client, while any other call participant acts as the SIP server when he responds.

3.1.2 SIP Call With Proxy Server

In the SIP message exchange of Figure 3.1, Alice knows the IP address of Bob and is able to send the INVITE directly to that address. Generally, an IP address cannot be used like a telephone number because IP addresses are often dynamically assigned using DHCP

(Dynamic Host Configuration Protocol) due to the shortage of IPv4 addresses. Another reason is an IP address does not uniquely identify a user, but identifies a node on a particular physical IP network. A user may share an IP address with other users at his office, by logging in as different users, and a user may also have multiple IP addresses at different locations. Ideally, there would be one address that would identify a user wherever he is. Therefore, SIP uses email-like names for addresses. The addressing scheme is part of a family of Internet addresses known as URIs. SIP URI is a name that is resolved to an IP address by using SIP proxy server and DNS lookups at the time of the call.

Figure 3.2 shows an example of a more typical SIP call with a type of SIP server called a "proxy server". In this example, the caller Alice calls Bob through a SIP proxy server. A SIP proxy server operates in a similar way to a proxy in HTTP and other Internet protocols. A SIP proxy server does not set up or terminate sessions, but sits in the middle of a SIP message exchange, receiving messages and forwarding them.

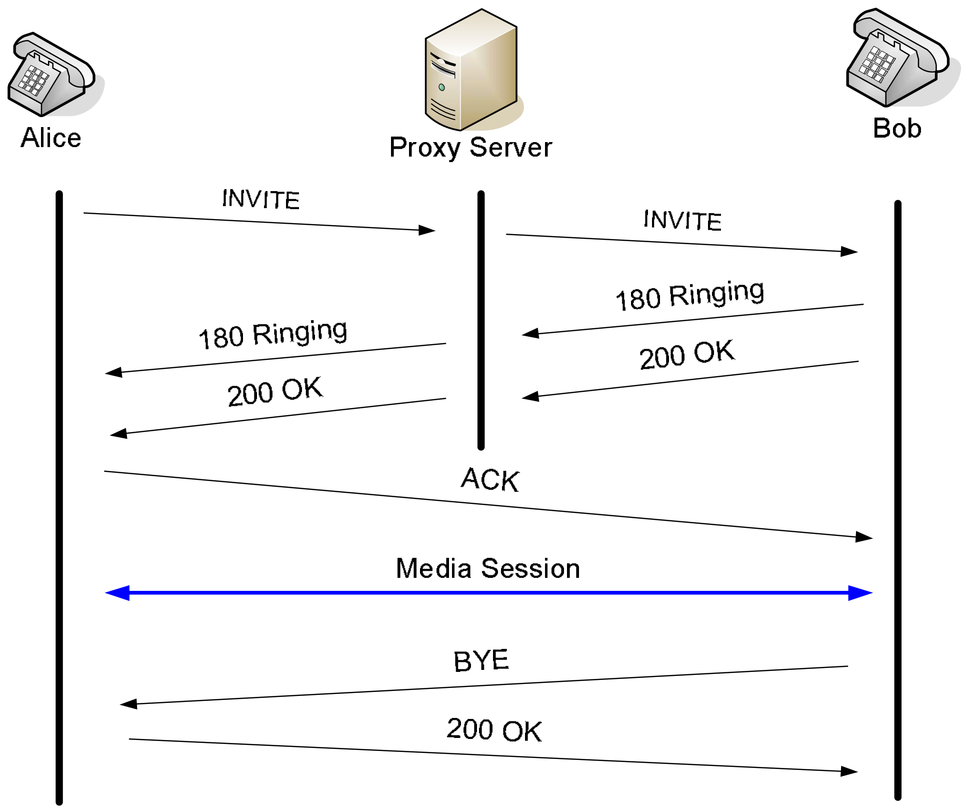

Figure 3.2: SIP call flow with proxy server

This example shows one proxy server, but there can be multiple proxy servers in a signaling path. Because Alice does not know exactly where Bob is currently logged on and which device they are currently using, a SIP proxy server is used to route the INVITE. A DNS lookup of Bob's SIP URI domain name (example2.org) is performed, which returns the IP address of the proxy server proxy.example2.org. And the INVITE is then sent to that IP address.

Usually, a user will tell its contact URI (current device and its IP address) to a specific server in a domain. This procedure is called registration, and the server is called registrar server. So the proxy server looks up the SIP URI in the Request-URI (sip:bob@example2.org) in the registrar server's database and locates Bob. The INVITE is then forwarded to Bob's IP address with the addition of a second Via header field stamped with the address of the proxy server.

From the presence of two Via header fields, Bob knows that the INVITE has been routed through a proxy server. Having received the INVITE, a 180 Ringing response is sent by Bob to the proxy server. The proxy server receives the response, checks that the first Via header field has its own address (proxy.example2.org), uses the transaction identifier in the Via header, then removes that Via header field, forwards the response to Alice. Similarly, when the call is accepted by Bob, he sends a 200 OK response to Alice via the proxy server. In SIP, the path of the signaling messages is totally independent of the path of the media. And the media session is ended when Bob sends a BYE message.

3.2 System Overview

The purpose of our computational puzzle based VoIP anti-spam method is to consume spammers' computation resources so that spammers cannot or do not deserve to make so many spam calls as before. Therefore, this method cannot prevent all the call spams by itself, but it is very helpful to mitigate the total number of call spams either as a separate VoIP anti-spam mechanism, or as a module in other VoIP anti-spam mechanisms. In this section, we will describe the system architecture of our computational puzzle based VoIP anti-spam system, including modules and their functions.

Figure 3.3 shows the big picture of our system. When a new call comes, it is first filtered by the white/black list module. If it is an anonymous call, it will enter the challenge-response module. Then a puzzle (challenge) will be sent to the caller. The puzzle is computed based on an appropriate puzzle cost which is obtained by checking the puzzle cost computation module. When the caller's response is received, the challenge-response module will verify it. If the verification fails, the SIP INVITE request will be sent to a user-configured decision module. Based on the rules in decision module, this SIP INVITE request will be redirected to the callee's "trashbox" (user-configured automatical answering system), or be refused automatically, or even be accepted if the callee does not care.

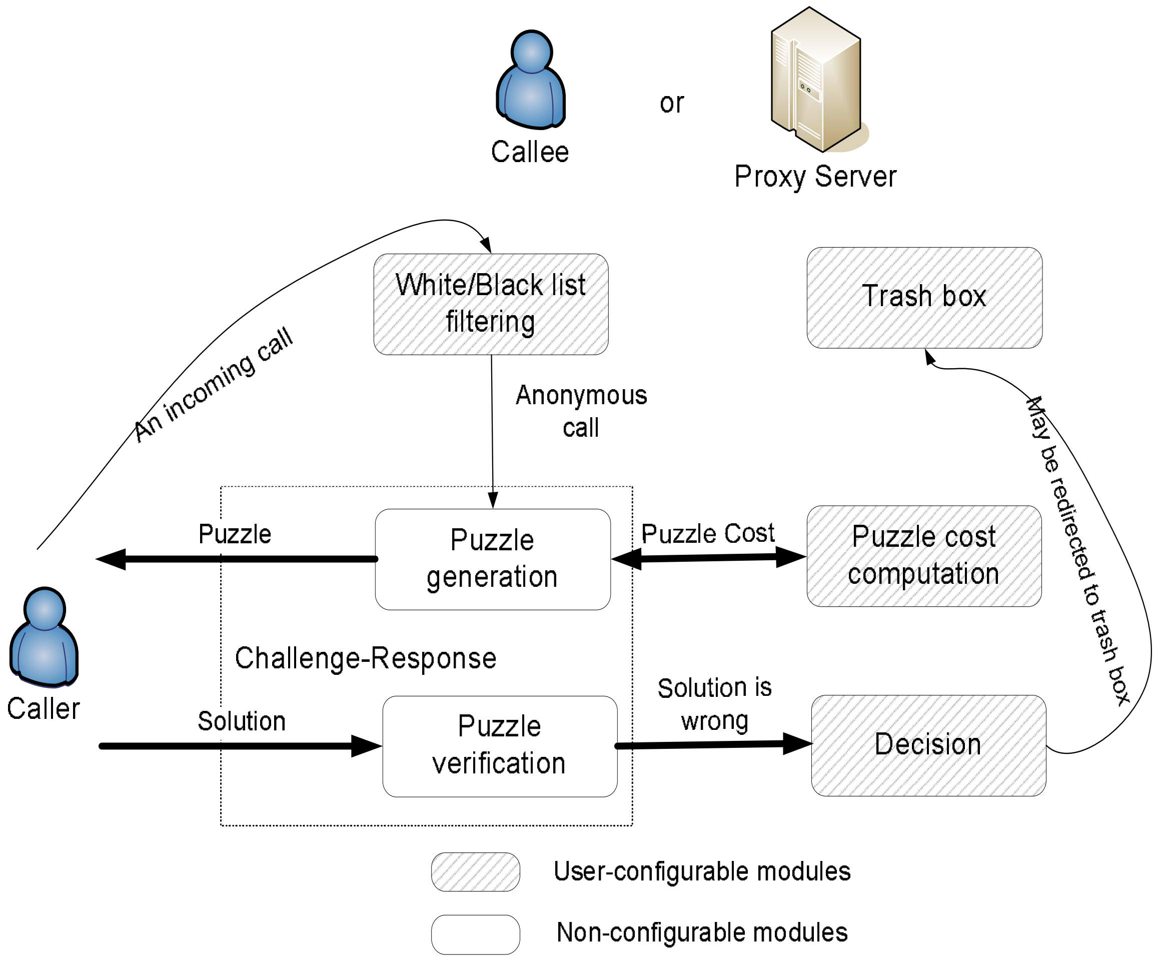

Figure 3.3: Computational puzzle based VoIP anti-spam system

Our computational puzzle based VoIP anti-spam system fulfill almost all the criteria that are defined in section 2.3. Our VoIP anti-spam method works before the phone rings, so that the user is not disturbed in his current activity. Computational puzzle increases the costs for the spammer, making spam less profitable. Our system could be located at the service provider (proxy server) only, so little user maintenance is required. Combining with white/black list module, the delay caused by the challenge-response module will be partly canceled out, because only anonymous callers would be challenged. When the caller has fulfilled his "proof of effort", he can be automatically or manually added to the white list of the callee, which will result in no challenge, no cost and no delay for the next calls between the caller and callee.

Another main advantage of this system architecture is its configuration flexibility and extendability. Firstly, this system is location-unaware. It can be placed at any user-trusted location in the VoIP system. It can be either integrated into the callee (or caller)'s domain proxy server as a VoIP spam firewall for the whole domain, or directly placed at callee side as a client-based VoIP anti-spam solution. And it can also be applied on both proxy server side and the callee side. To avoid duplicated computational puzzles from proxy severs and the callee, after the caller solves the proxy server's puzzle, the proxy server can insert a "puzzle resolved" indicator before it continues forwarding the request to the callee. The security mechanism between the proxy server and the callee can prevent faked "puzzle resolved" indicator. It can be implemented either via realm authentication scheme [13] or via a secure link, such as TLS [6]. Then the callee could choose to continue challenging the caller or stop it based on his own specific configuration. Secondly, most of the modules can be specifically configured by different users. Besides user-oriented black/white list and decision modules, users can also change the parameters in puzzle cost generation and challenge-response modules to meet their different requirements. Thirdly, when a more advanced and accurate filter technique is proposed in the future, we can easily integrate it into this system as a new module between black/whtie list module and the challenge-response module. Fourthly, this system is a self-learning system. Initially, users have to react to validated anonymous calls, either adding them to white list or black list. But soon, as more elements are added to white/black list, the system would begin to filter messages. The pace of this process could be left up to individual users, based upon how quickly their own friends and colleagues adopted this system. If well implemented using a common, open standard agreed upon by all vendors, it would be straightforward for users to adopt. Easy for everyone, except for the spammers.

3.3 Challenge-Response Module

In this section, we present the core module - challenge-response module in our VoIP anti-spam system. It covers the computational puzzle generation and verification procedures. The idea behind this module can be summarized as follows. When a callee, say Bob, receives call invitation from an anonymous caller (not in white/black list), say Alice, Bob will send a computational puzzle back to Alice for solving. There is enough randomization involved that Alice really do have to solve the puzzle on a case-by-case basis. Only when Alice sends Bob the correct result, Bob will accept the call invitation from Alice. If the computation is complex enough, it will take far longer to send large numbers of unsolicited SIP invitation than it does now.

Many existing studies [8] [16] have proposed computational puzzle algorithms to solve different practical problems. Our focus is how to integrate computational puzzle algorithm into VoIP system to mitigate VoIP spams. Therefore, we picks a simple computational puzzle algorithm for our system. The puzzle is the brute-force reversal of a one-way hash function, such as MD5 or SHA-1. This is a practical choice because the hash functions are computable with a wide variety of hardware and the brute-force testing of different inputs is likely to be the most efficient way for computing the inverse of these functions. In this section, we describe this algorithm and the related SIP protocol extension. Our algorithm and message format are similar to those described in [14].

The algorithm works as follows: when a caller sends a SIP invitation message to a callee via a proxy server, the proxy server (or callee) generates a random global unique string for this call request, calculates and saves the hash (such as MD5, SHA-1, SHA-2) output, and sends the hash output back to the caller. The caller is asked to search for a string that has the same hash output and send back the string as the answer. The server controls the puzzle complexity by controlling the hash algorithm and the search space size. The call flow diagram of this algorithm is illustrated in Figure 3.4. It includes 5 primary SIP request and response messages.

1. Caller $\Rightarrow$ Proxy server(or Callee): INVITE
2. Caller $\Leftarrow$ Proxy server(or Callee): 419 Puzzle required
3. Caller $\Rightarrow$ Proxy server(or Callee): re-INVITE (with solution included)
4. Caller $\Leftarrow$ Proxy server(or Callee): 200 OK (if the solution is correct)

5. Caller ⇐ Proxy server(or Callee): 403 Forbidden (if the solution is wrong)

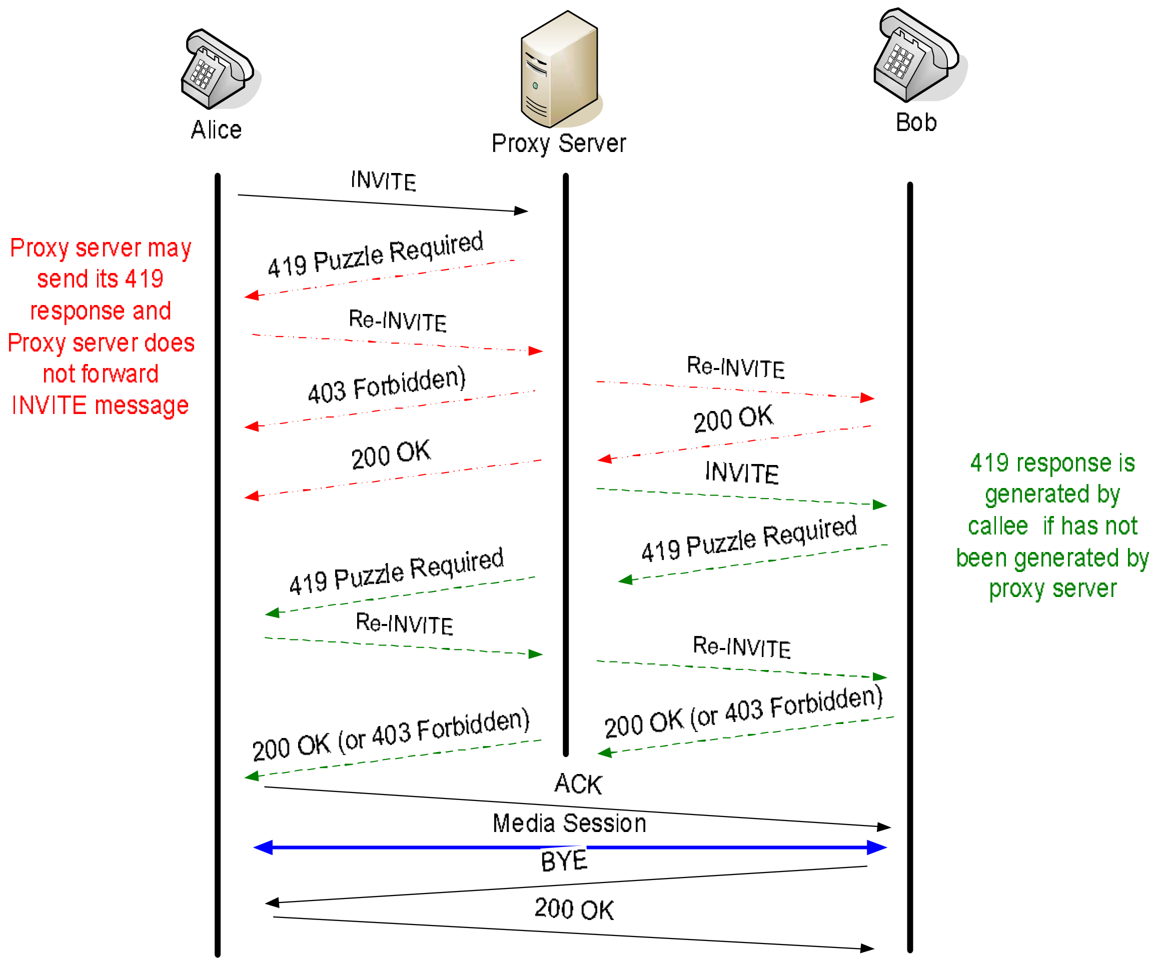
Alice
Proxy Server
Bob
INVITE
Proxy server may send its 419 response and Proxy server does not forward INVITE message
419 Puzzle Required
Re-INVITE
403 Forbidden)
200 OK
Re-INVITE
200 OK
INVITE
419 response is generated by callee if has not been generated by proxy server
419 Puzzle Required
419 Puzzle Required
Re-INVITE
Re-INVITE
200 OK (or 403 Forbidden)
200 OK (or 403 Forbidden)
ACK
Media Session
BYE
200 OK

Figure 3.4: SIP call flow with computational puzzle

Compared with standard call flow diagram in section 3.1.2, two extra messages are added in this call flow diagram, which enabled the puzzle generation and verification involved in the call flow. Firstly, if the caller is anonymous to the proxy server (or callee), the proxy server (or callee) will send a “419 puzzle required” SIP response message to the caller before he decides to accept the call. Secondly, after a brute-force computation, the caller will send a new SIP INVITE (we call it re-INVITE) message which includes the computational solution back to the proxy server (or callee).

Now let us explain puzzle generation procedure, solution generation procedure and the related puzzle format in detail. As illustrated in Figure 3.5. it is a two-step process to generate the puzzle. All the notations used in the explanation are displayed in table 3.1 as below.

Table 3.1: Notation in the puzzle generation procedure

Symbol	Meaning
N_s	Callee nounce
N_c	INVITE dialog identifier extracted from the caller's request
N_t	Callee's system time when the puzzle is generated
k	Difficulty level

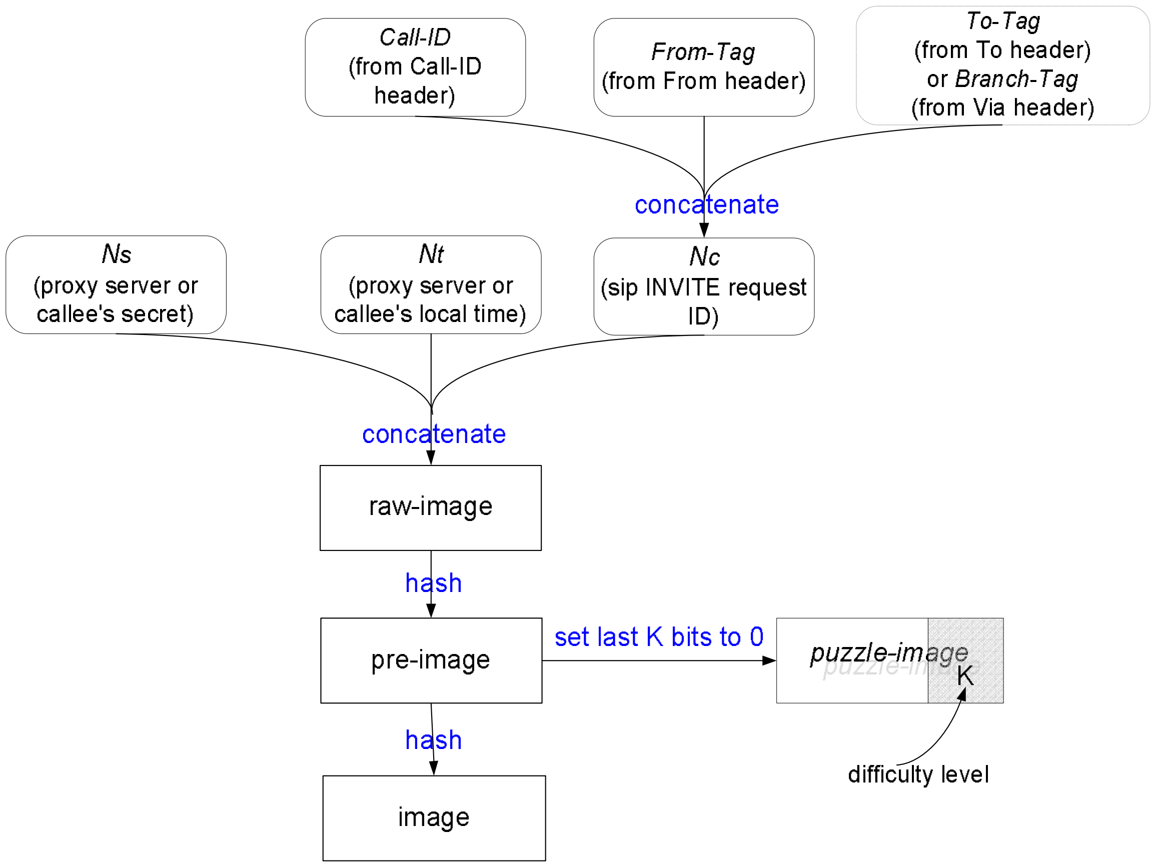
Call-ID (from Call-ID header)
From-Tag (from From header)
To-Tag (from To header) or Branch-Tag (from Via header)
concatenate
Ns (proxy server or callee's secret)
Nt (proxy server or callee's local time)
Nc (sip INVITE request ID)
concatenate
raw-image
hash
pre-image
set last K bits to 0
puzzle-image
K
difficulty level
hash
image

Figure 3.5: Computational puzzle generation

Firstly, the proxy sever (or callee) concatenates N_s, N_c and N_t to generate a new string, called "raw-image". N_s is a proxy server (or callee)'s own nounce string. To create new puzzles, the callee periodically generates a nounce, which is a secret known only by the proxy server (or callee) itself. To prevent the spammers from precomputing solutions, the nounce needs to be random and not predictable and needs to be updated periodically. N_t is a timestamp of the proxy server (or callee)'s local system. And N_c is an identifier string which is used to distinguish different SIP INVITE requests. In SIP, a dialog is a peer-to-peer relationship between two VoIP user agents. As explained in section 3.1, invite dialog is used for making and accepting call requests. In each dialog, there is only one original INVITE message. And there may have one or more re-INVITE messages which are used to update the request information for the original INVITE message. N_c is used to check if an INVITE is a re-INVITE message (belonging to previous INVITE dialog) or an original INVITE message. In SIP, the combination of Caller-ID (in Caller-ID header), From-Tag (in From header) and To-Tag (in To header) is a global unique identifier for a SIP dialog, which can be used as N_c. Please note, the To-Tag is not included in the original SIP INVITE message, and it is generated by the callee. In case that proxy server generates the puzzle, the To-Tag is replaced by the Branch-Tag (in Via Header) to generate N_c, because the To-Tag has not been generated when the original INVITE message arrives at the proxy server.

Secondly, we compute a hash on the "raw-image" using a hash algorithm (such as MD5). We call the result string as "pre-image". Then, we calculate another hash on "pre-image" using the same hash algorithm and call the hashed out string as "image". Finally we set the last k bit in "pre-image" as zero and call this string as "puzzle-image". k is called "difficulty level", and it controls the search space size for caller's brute-force computation.

To deliver the puzzle to the caller, the 419 response message includes a new "puzzle header" besides the standard headers (From, TO, Caller-ID, Via, Cseq), with the following format:

```
Puzzle:  difficulty="k", puzzle="puzzle-image", image="image", method="MD5"
```

The "puzzle-image" and "image" fields are filled by the values discussed above. When the caller receives the 419 response message, it will perform a "brute-force" search for the last k bits of "puzzle-image" string, in a way that the caller computes the hashes of all the possible candidates until the hash output match the "image" string. Therefore, the caller has to try $2^{(k-1)}$ times hash operations in average. After the caller finds out the solution, he will send a re-INVITE message back to the proxy server (or callee). The re-INVITE message is quite same as the original INVITE message except for two headers. First, the

Cseq number in the Cseq Header is increased by 1. It is used to distinguish the re-INVITE message from the original INVITE message. Second, a "puzzle header" is added to the re-INVITE message, in which "pre-image" means the solution to the puzzle used in puzzle generation procedure.

```
Puzzle:  difficulty="k", solution="pre-image", image="image", method="MD5"
```

3.4 Adaptive Cost Assignment in Computational Puzzles

With the knowledge of our computational puzzle algorithm, this section explains how to assign puzzle "difficulty level" for each anonymous SIP INVITE message. This is the puzzle cost computation module in our VoIP anti-spam system.

A simple idea is to set same cost for all the anonymous SIP INVITE messages. However, in this way, all innocent anonymous calls will be treated same as suspicious calls. This inspires us to assign different costs for different anonymous calls based on its probability of being spams. There is a popular email anti-spam method which includes a scheme of evaluating the possibility of being spams, called the Nalve Bayes method [18], often known as Bayesian Spam Filtering. Its idea is based on the study that some words occur more frequently in known spam, and other words occur more frequently in legitimate messages. With statistical and mathematical techniques, it is possible to generate a "spam-indicative probability" for each word and then determine the overall "spam probability" of a novel message. But this kind of spam filtering is completely useless for call spam. This can be accounted to two reasons. Firstly, the content of an anonymous call cannot be analyzed before the user answers it. We want to mitigate VoIP spam before the callee answers the call. Otherwise, the user would have already paid attention and been interrupted. Secondly, if the content is stored before the user accesses it, the content will be in the form of recorded audio or video. Currently, speech and video recognition technology is not mature enough to make accurate decision on whether or not a message is spam. But we can borrow the statistical idea from Bayesian Spam Filtering, and assign the puzzle cost for each anonymous call based on the caller's call history.

We choose four guidelines for the puzzle cost assignment. First, we would like to assign zero or negligible computational costs to all the anonymous calls if currently call spam are very rare overall. Second, when a new spammer is detected, we will increase the puzzle cost for all the other anonymous callers coming from the same domain. Third, when a new call is classified as suspicious call spam, we would like to increase the puzzle cost for all the future calls from this specific caller. Fourth, when a new anonymous call cannot solve the puzzle correctly, we will increase the puzzle cost for this specific caller appropriately. However, if the caller solves the puzzle correctly, we will decrease its puzzle cost appropriately.

From callee's perspective, it is easy to determine if an anonymous call is a call spam and if it needs to be added into the white/black list as soon as callee answers the call. Therefore, we only apply the adaptive cost assignment to SIP proxy servers. The adaptive puzzle cost computation is based on a summation $C(m) = D + Q(m) \times T_q + R(m) \times T_r - S(m) \times T_s + F(m) \times T_f$. All the notations used in this section are displayed in table 3.2. $C(m)$ is the cost level for the anonymous call m. It means how many bits need to be set as 0 in "puzzle-image" string. D is the initial puzzle cost for all the anonymous calls. $Q(m)$ is the probability that an anonymous caller from this caller's domain is a spammer over recent period of time. $R(m)$ represents the number of users who have reported this anonymous caller as a spammer over recent period of time. $S(m)$ means how many times this caller cannot solve the puzzle. $F(m)$ means how many times this caller has solved the puzzle. T_q is a translation function used to map $Q(m)$ to the number of bits. Similarly, T_r, T_s and T_f are used to map $R(m)$, $S(m)$ and $F(m)$ to number of bits respectively.

Table 3.2: Notation in the puzzle cost computation procedure

Symbol	Meaning
$C(m)$	puzzle cost for anonymous call m
D	initial puzzle cost
$Q(m)$	spam call probability of caller's domain
T_q	translation function used to map $Q(m)$ to the number of bits
$R(m)$	how many users have reported this caller as a spammer
T_r	translation function is used to map $R(m)$ to the number of bits
$S(m)$	how many times this caller has solved the puzzle
T_s	translation function is used to map $S(m)$ to the number of bits
$F(m)$	how many times this caller cannot solve the puzzle
T_f	translation function is used to map $F(m)$ to the number of bits

Chapter 4

Implementation

To evaluate the performance of our VoIP anti-spam system, we implement it as an extension to Mjsip [19]. Mjsip is an open source VoIP software. It is a complete java-based implementation of the SIP stack. It includes the complete SIP stack architecture as defined in RFC 3261 [13], and is fully compliant with the standard. In this chapter, we will describe our implementation in details. We will first review the layered SIP implementation in Mjsip, and then explain the implementation of our computational puzzle system, with the help of data flow and state transaction diagram.

4.1 SIP Stack Implementation in Mjsip

According to the SIP architecture defined in RFC 3261, the core of MjSip is structured including three base layers: Transport, Transaction, and Dialog. On top of these layers, MjSip also provides Call Control and application level layers, with the corresponding APIs. Figure 4.1 shows the layered architecture in Mjsip.

From bottom to up, the lowest layer is the transport layer, which provides the transport of SIP messages. The SipProvider is an MjSip object that provides the transport service to all upper layers. It is responsible for sending and receiving SIP messages through different lower layer transport protocols (such as UDP or TCP), and to demultiplex incoming messages toward the appropriate upper layer entities.

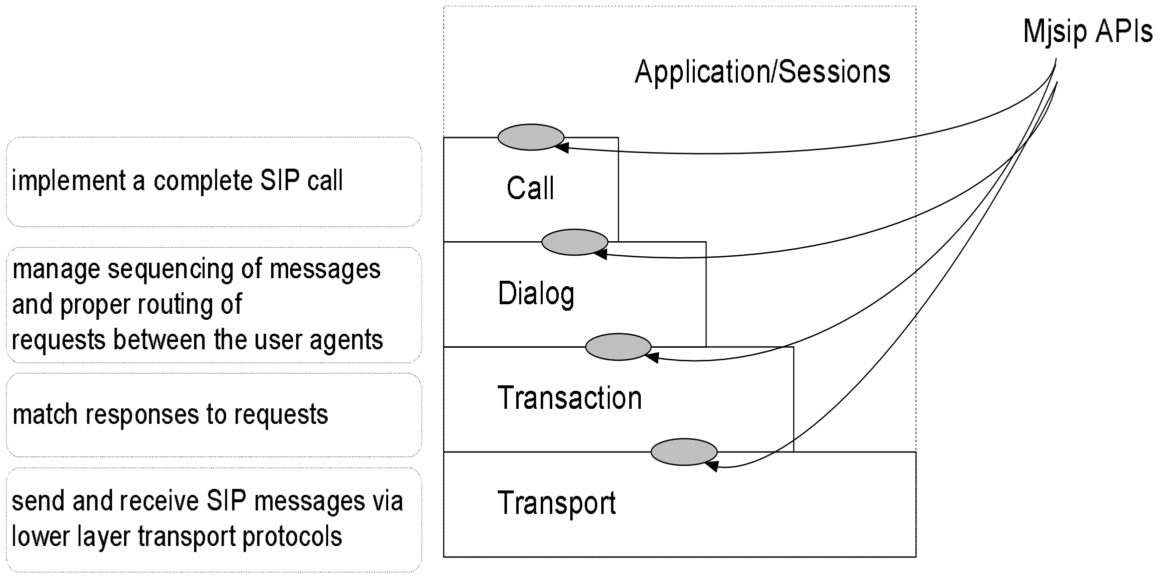
Mjsip APIs
Application/Sessions
implement a complete SIP call
Call
manage sequencing of messages and proper routing of requests between the user agents
Dialog
match responses to requests
Transaction
send and receive SIP messages via lower layer transport protocols
Transport

Figure 4.1: Mjsip layered architecture

The second layer is the transaction layer. A transaction is a fundamental component of SIP, which is a request sent by a transaction client (caller) to a transaction server (callee) along with all responses to that request sent from the transaction server back to the client. The transaction layer handles upper-layer retransmissions, matching of responses to requests, and timeouts.

The third layer is the dialog layer that binds different transactions within the same "session". As we have described in section 3.3, a SIP dialog is a peer-to-peer relationship between two user agents that persists for some time. The dialog facilitates sequencing of messages and proper routing of requests between the user agents. As defined in RFC 2631, the INVITE method establishes an INVITE dialog.

Above the three base layers, there is a fourth layer called Call Control layer, which implements a complete SIP call. It is implemented by the Call API, which is a simple-to-use interface to handle incoming and outgoing SIP calls. Note that a call may consist of more than one dialogs. SIP sessions, standing upon all the four layers, bind application entities (participants) on different systems.

In the MjSip package, APIs are provided to access various layers, from SipProvider to Call Control through the following classes: Call, InviteDialog, InviteTransactionClient, InviteTransactionServer, and SipProvider. The interfaces between adjacent layers are based on a Provider → Listener model, as illustrated in Figure 4.2. When a class wants to interact with an underlying layer, it has to extend the relative LayerListener class for that layer (i.e. the layer provider) and add itself to the list of possible listeners of the events generated by the lower layer/provider. The events are captured by the upper class through specific listener methods inherited by the specific Listener class.

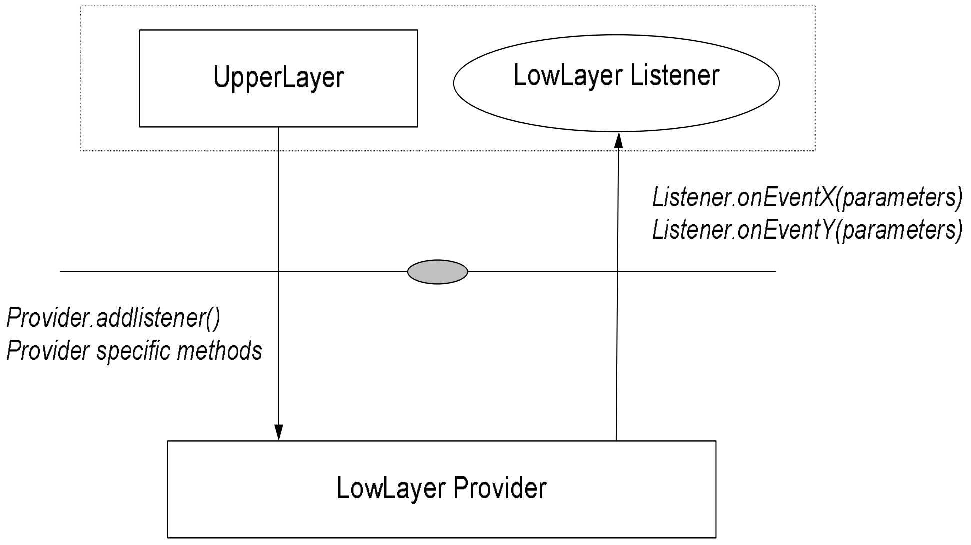
UpperLayer
LowLayer Listener
Listener.onEventX(parameters)
Listener.onEventY(parameters)
Provider.addlistener()
Provider specific methods
LowLayer Provider

Figure 4.2: Provider → Listener model

4.2 Computational Puzzle Implementation in Mjsip

We implemented computational puzzle based VoIP anti-spam system both on the proxy server and on normal user agents (callers and callees). Because proxy servers are only in charge of forwarding callers' INVITE requests to callees, the challenge-response implementation on proxy servers is simpler than that at the callee side. We will take the callee side implementation to explain our extension to Mjsip. The scenario is the same as that in section 3.1.1. None of servers are involved, and both callers and callees are connected to an IP network and know each other's IP addresses.

SipProvider is an Mjsip object that represents Mjsip's transport layer. The main classes we discuss in this chapter are displayed in table 4.1. SipProvider sends and receives

SIP messages, receives SIP messages from the network (through the UDP/TCP layers), and delivers them to the appropriate upper layer entity, which could be a transaction, a dialog, or an application entity. By inserting SipProvider listeners, the upper layer objects in transaction and dialog layers could communicate with SipProvider objects. Specifically, when the SipProvider receives a new message from the lower layers (UDP/TCP layers), it would choose the appropriate upper layer entity based on the matching of transaction-id, dialog-id, or message type with the list of the current registered SipProvider listeners. InviteTransactionServer, InviteTransactionClient are the objects of transaction layer. ExtendedDialog is the object of dialog layer. All of these three are the main targets of our implementation, since the computational puzzle method mainly focuses on the extension for standard SIP invite dialog.

Table 4.1: Main mjsip classes used in the computational puzzle implementation

Mjsip layer	Class
Transport layer	SipProvider
Transaction layer	InviteTransactionClient, InviteTransactionServer
Dialog layer	ExtendedInviteDialog, Dialog
Call Control layer	Call

We will first describe the puzzle generation and verification procedure, which focuses on the communication between SipProvider and the upper layer objects. Then we explain the corresponding state transition diagrams corresponding for the transaction and dialog layers.

4.2.1 Puzzle Generation and Verification Procedure

Figure 4.3 shows the whole picture for computational puzzle generating, solving and verifying procedures.

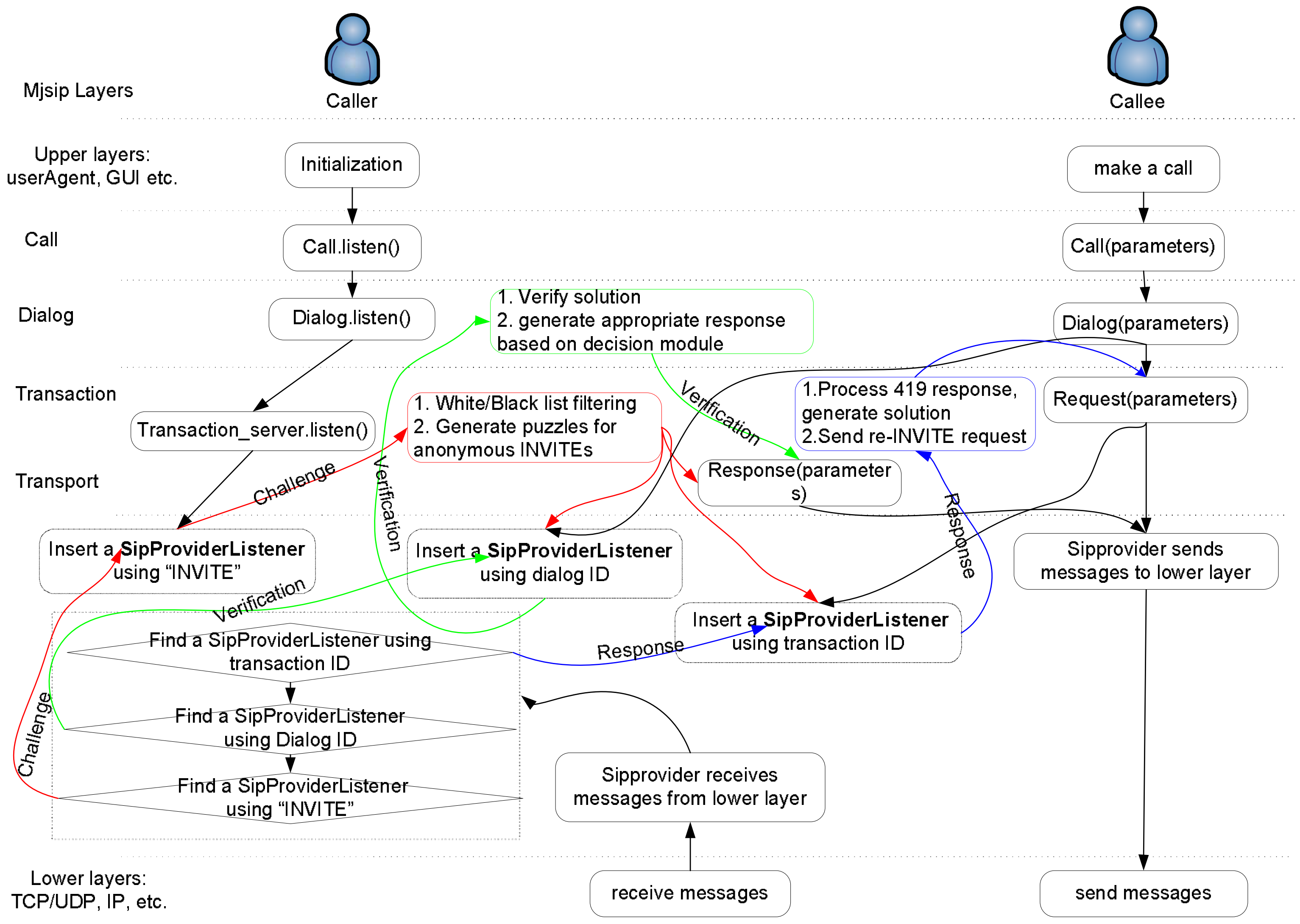
Mjsip Layers
Caller
Callee
Upper layers: userAgent, GUI etc.
Initialization
make a call
Call
Call.listen()
Call(parameters)
Dialog
Dialog.listen()
1. Verify solution
2. generate appropriate response based on decision module
Dialog(parameters)
Transaction
Transaction_server.listen()
1. White/Black list filtering
2. Generate puzzles for anonymous INVITEs
Verification
1.Process 419 response, generate solution
2.Send re-INVITE request
Request(parameters)
Transport
Challenge
Verification
Response(parameters)
Response
Insert a SipProviderListener using "INVITE"
Insert a SipProviderListener using dialog ID
Sipprovider sends messages to lower layer
Verification
Insert a SipProviderListener using transaction ID
Find a SipProviderListener using transaction ID
Response
Find a SipProviderListener using Dialog ID
Challenge
Find a SipProviderListener using "INVITE"
Sipprovider receives messages from lower layer
Lower layers: TCP/UDP, IP, etc.
receive messages
send messages

Figure 4.3: Puzzle generation and verification procedure

At the caller side, an instance of the Call class is triggered when a caller makes a call. This is in fact a SIP INVITE request, and an ExtendedInviteDialog instance will be generated by the Call instance. Then the ExtendedInviteDialog will generate a corresponding SIP INVITE request message and pass it to the InviteTransactionClient generated in this Dialog. Also, the ExtendedInviteDialog instance will insert a SipProvider listener with its Dialog ID. As described previously, a Dialog ID is composed by Call-ID (in Call-ID header), From-Tag (in From header) and To-tag (in To header). If one or more components do not exist, "null" will be used to occupy the corresponding positions. At this time, the To-Tag does not exist and will be updated when the caller receives the response message from the callee. Next, InviteTransactionClient will insert a SipProvider listener with its transaction ID, and send the INVITE message to the callee via the included SipProvider objects. The transaction ID is composed of Call-ID (in Call header), Cseq (in Cseq header), Method (in the first line of the message), Sent By (in Via header) and Branch (in From header if used in transaction client, To header if used in transaction server).

At the callee side, Mjsip is initialized with an empty Call instance. By calling the corresponding listen method, the Call instance creates an empty Dialog instance, which in turn, creates an InviteTransactionServer instance. Finally, this InviteTransactionServer instance inserts a SipProviderListener with a specific transaction ID - "INVITE", and this listener is used to wait for the SIP INVITE request.

Whenever a new message is received and passed from lower layers (UDP/TCP) to the transport layer (SipProvider), the OnReceivedMessage method will be called. Its responsibility is to find a matched SipProvider listener through which it passes the message to the upper layer. The listener matching procedure involves many steps. Among those, there are three important steps: matching SipProvider listener using transaction ID, using dialog ID, and using specific transaction ID ("INVITE").

When a new INVITE message arrives at the callee side, the SipProviderListener with ID "INVITE" will be trigged. This listener first removes itself from the listener list and then inserts a new listener with the actual transaction ID. Next, it will call onTransRequest procedure included in InviteTransactionServer to process the INVITE message. The black/white list filtering module and the puzzle generating part in challenge-response module are implemented here. And the onTransRequest procedure also inserts a SipProviderListener with the actual dialog ID.

When a 419 "puzzle require" message reaching caller side, the onTransFailureResponse procedure included in transaction client will be trigged to process it. The puzzle solving part in challenge-response module is implemented here.

Finally, when the re-INVITE message with puzzle solution reaches the callee side, the SipProviderListener with the actual dialog ID will be trigged. The OnReceivedMessage procedure included in the dialog will process the re-INVITE message. And the puzzle verification part in challenge-response module and decision module take their responsibilities here.

4.2.2 State Transition Diagram

The state transition diagram of our VoIP anti-spam system has two INVITE transactions in the INVITE dialog of an anonymous caller. One is for the original VoIP INVITE request, the other is for the added re-INVITE request. Figure 4.4 and Figure 4.5 illustrate the transaction client side and transaction server side state transition diagrams for these two INVITE transactions. Figure 4.6 shows the dialog-level state transition diagram for INVITE dialogs. The added "verifying" state is used to represent challenge-response procedure.

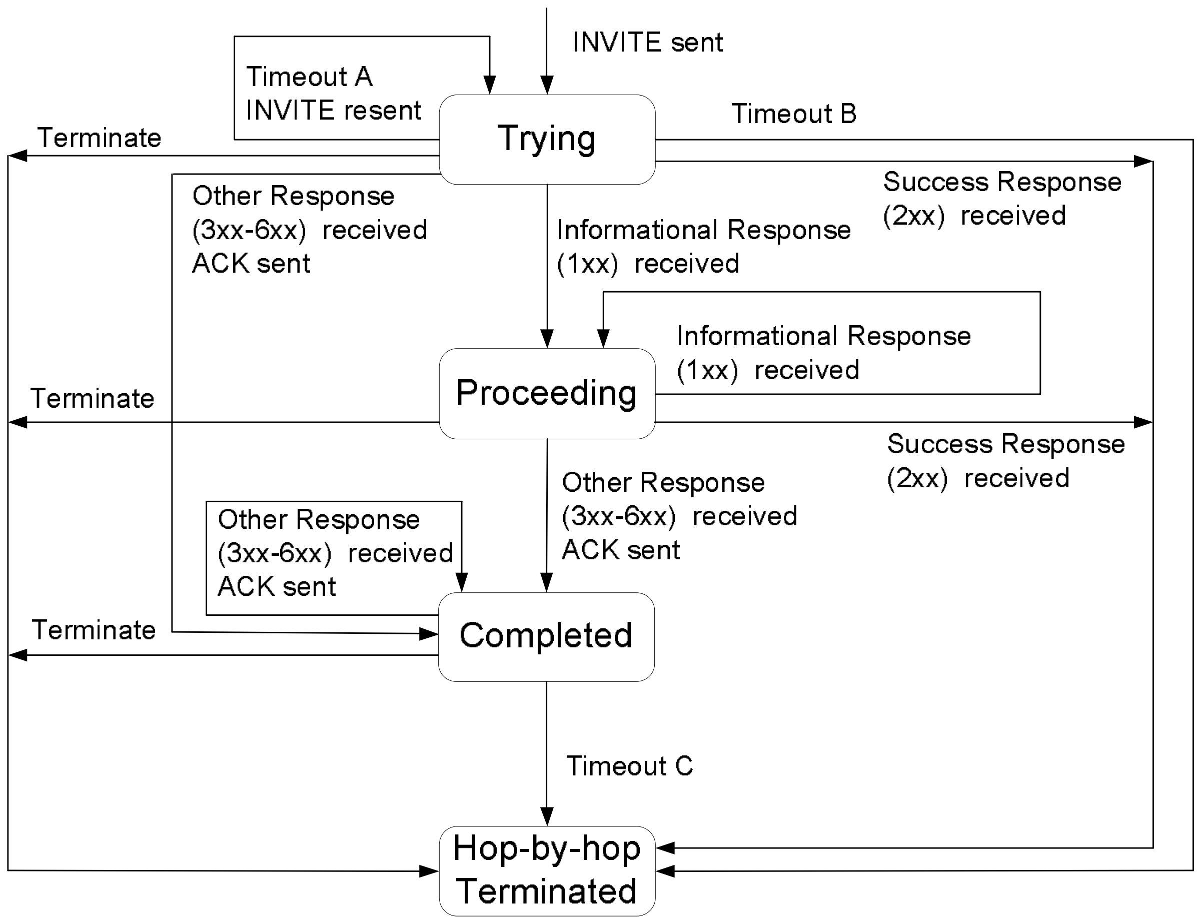

Figure 4.4: Invite transaction client state diagram

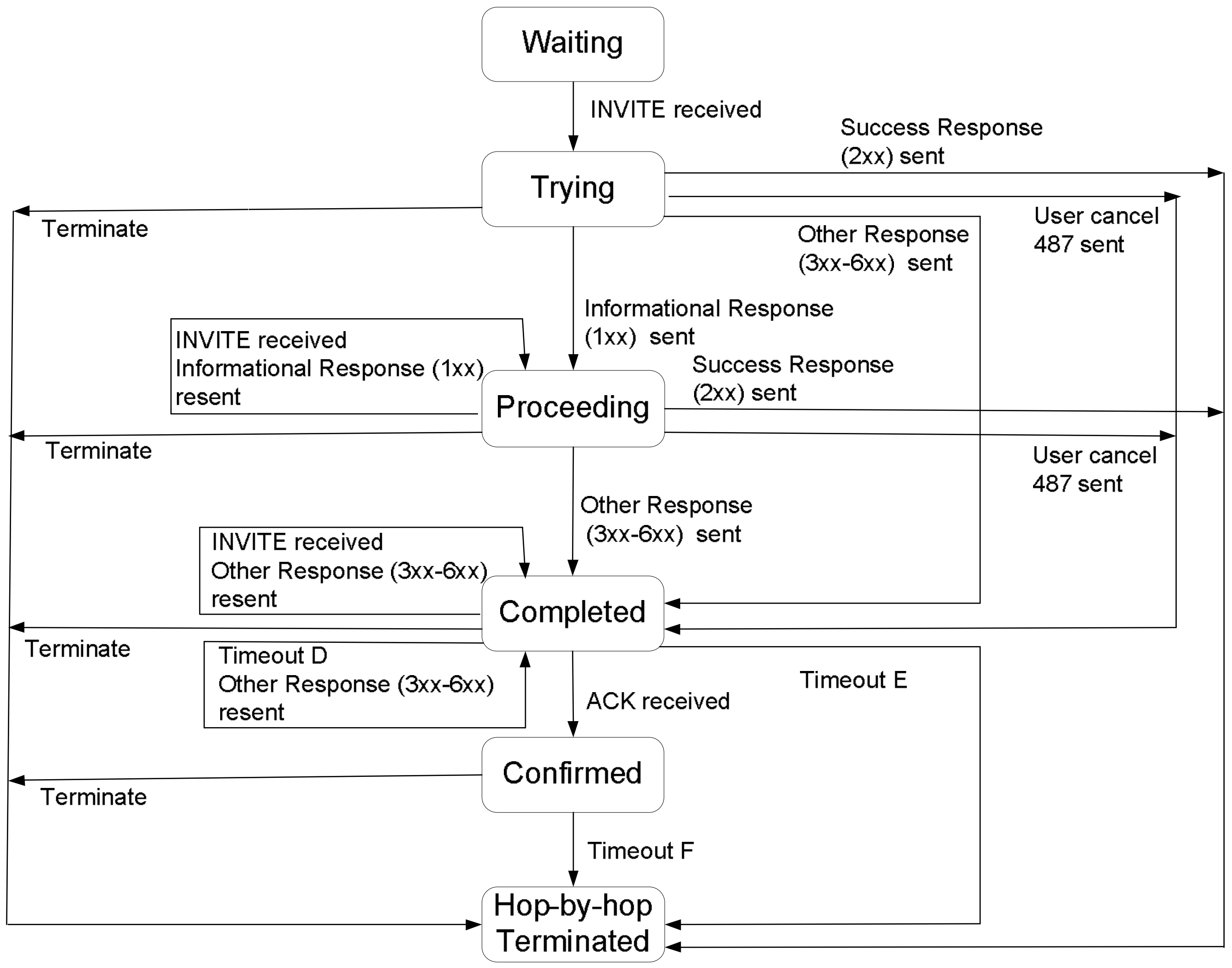
Waiting
INVITE received
Trying
Success Response (2xx) sent
User cancel 487 sent
Other Response (3xx-6xx) sent
Terminate
Informational Response (1xx) sent
INVITE received Informational Response (1xx) resent
Proceeding
Success Response (2xx) sent
Terminate
User cancel 487 sent
Other Response (3xx-6xx) sent
INVITE received Other Response (3xx-6xx) resent
Completed
Terminate
Timeout D Other Response (3xx-6xx) resent
Timeout E
ACK received
Confirmed
Terminate
Timeout F
Hop-by-hop Terminated

Figure 4.5: Invite transaction server state diagram

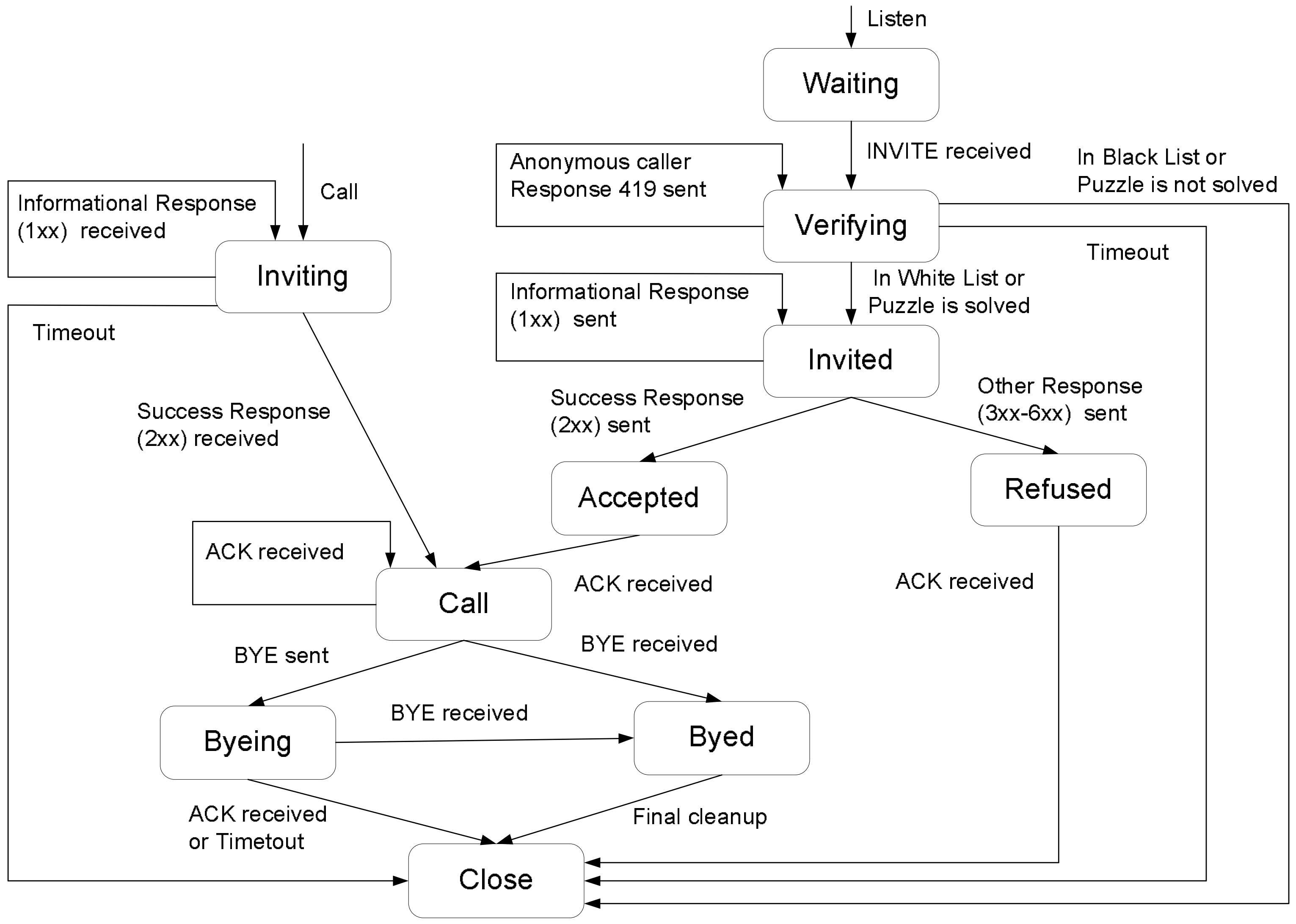

Figure 4.6: Invite dialog state diagram

Chapter 5

System Evaluation

This chapter discusses the evaluation of our VoIP anti-spam system which includes white/black list and computational puzzle based system. Section 5.1 describes the four performance metrics used for evaluation and the experiment setup. From section 5.2 to section 5.5, we present the experimental results corresponding to each of the metrics respectively. Section 5.6 discusses one of the key functions in the puzzle challenging process. And section 5.7 discusses a possible strategy that spammers may adopt to respond to our system, and the success of the anti-spam system in combating it.

5.1 Evaluation Metrics and Experiment Setup

The following metrics are adopted to evaluate the effectiveness of the computational puzzle based VoIP anti-spam system:

- **Block rate**: It is the percentage of blocked VoIP spam calls (or spammers) out of all VoIP spam calls (or spammers). It reflects the system's blocking ability on spam calls.

- **Spam ratio**: It is the percentage of successfully accomplished VoIP spam calls out of all VoIP calls. It indicates the spam severity of the system.

- **Overhead for proxy servers and legitimate end users**: The overhead introduced by the anti-spam method. It should be limited to a tolerable degree.
- **Spam call generating ability**: It depicts VoIP spammers' attacking ability, in terms of call generation rate.

We will show the success of our VoIP anti-spam system in achieving high block rate, low spam ratio, low overhead, as well as the good capability in restraining spammers' spam generating ability. Moreover, we will discuss the benefits of the adaptive puzzle cost assignment method. We will also go further to discuss the possible spammer strategies and show the effectiveness of our system in the case of smarter spammers.

We use emulation to evaluate the VoIP anti-spam system. Figure 5.1 illustrates the topology used in our experiments. The experiment setup consists of three VoIP domains: one receiving domain and two call generating domains. In each domain, there is one SIP proxy server and at most 70 VoIP end users. The end users in the receiving domain act as callees, who receive VoIP calls from the end users in the other two domains. All the proxy servers and end users are implemented using Mjsip VoIP software with our VoIP anti-spam extension. They are strictly in compliant with SIP RFC [13]. Moreover, they are compatible with the real SIP phones and can establish call sessions with existing SIP phones. In our experiments, different VoIP domains are implemented on different machines. All the machines in the systems are 1.6-2.0GHz Dell or Lenovo laptops, running windows XP, and they are connected through a 54Mbps wireless router.

Before the start of the experiments, we randomly choose a subset of users as spammers from the two call generating domains. We also assume that the end users in the call generating domains know the URIs (or phone number) of all the end users in the receiving domains, so that they can make calls to any of the callees. In the experiments, both the spammers and the legitimate callers randomly generate calls to the end users in the receiving domain.

5.2 Block Rate

Block rate directly reflects the system's anti-spam ability. We assume a fixed number of spammers, and observe how effective the system is in blocking spams. In fact,

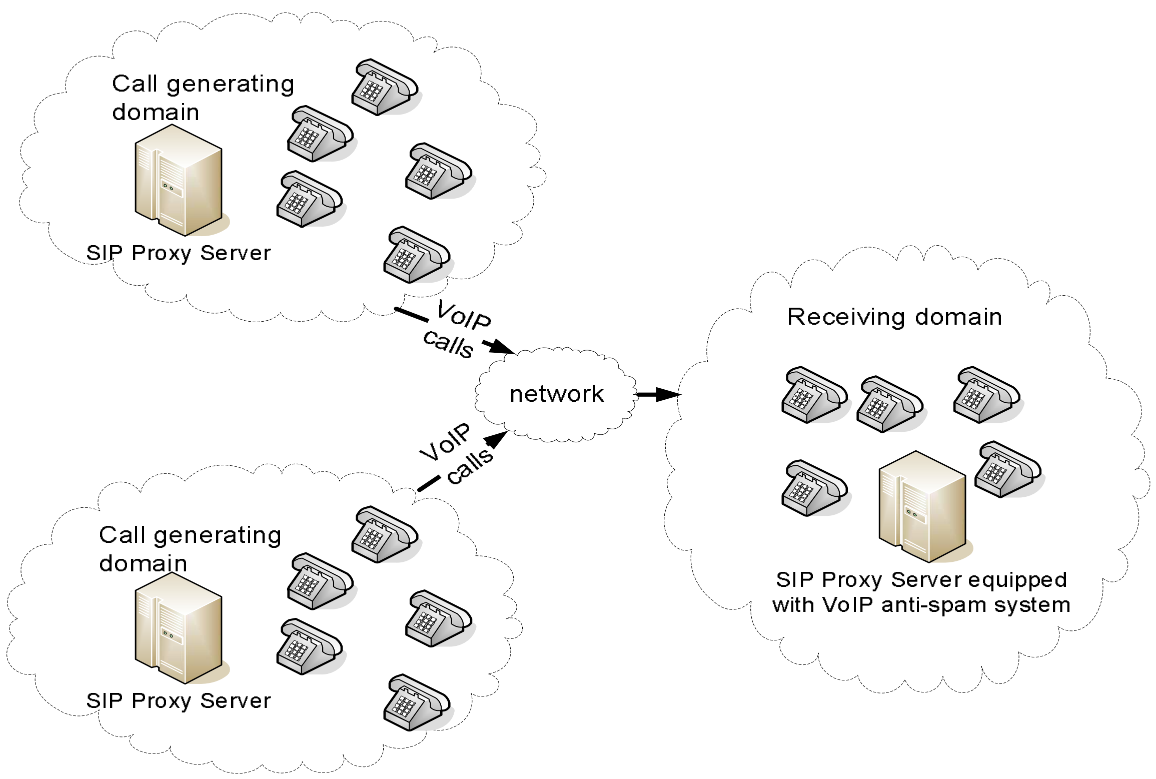

Figure 5.1: Experiment setup

block rate could be viewed either as the percentage of blocked spam calls or as the percentage of blocked spammers.

The experimental scenario is set as follows. We choose 5 spammers and 30 legitimate callers in each call generating domain, and let them randomly make calls to the end users in the receiving domain. At the beginning of the experiments, all the callers are anonymous. The puzzle challenge function is only setup at the proxy servers. When the calls arrive at the proxy server of the receiving domain, puzzles are generated and sent to the callers. If the puzzles are solved correctly, the proxy server would forward the calls to the appropriate callees. All the callees would accept the call automatically in 1 second and hang it up in another 1 second. The inter-arrival time of call generation for each caller follows exponential distribution with parameter $\lambda = 0.00625$. That is, each caller generates calls with an average time interval of 160 seconds. We assume spam calls are generated at each spammer with a rate of 0.0125 calls/second. If a callee received a spam call, it would report it to the proxy server with 70% probability. And when the proxy server receives spam reports for the same caller from 5% different callees, it would put this caller into its black list.

Figure 5.2 shows the block rate increased with the experiment time. The changes of block rate are not triggered by computational puzzle based system but by black list. However, as an evaluation of the integration of white/black list and computational puzzle based system, we still need to evaluate this part. Initially, the block rate is low, because it is the learning period for the proxy server to identify spam calls. Although the anonymous callers are challenged and delay is introduced, the calls are still permitted. With the time passing by, more and more spam reports are received and spammers are gradually put into the proxy server's black list. The percentage of blocked spam calls gradually increase. The percentage of blocked spammers is very high at the end of the experiment.

Figure 5.3 compares total calls, spam calls and blocked spam calls during the experiment. Note that all the spam calls would be blocked at the end of the experiment. In this scenario, we assume VoIP users would report spams with a high probability. Nowadays, a significant number of email users have been contributing to email spam reports, so it would reasonable to believe the same user behavior for VoIP users. The trend of increase in block rate would remain the same.

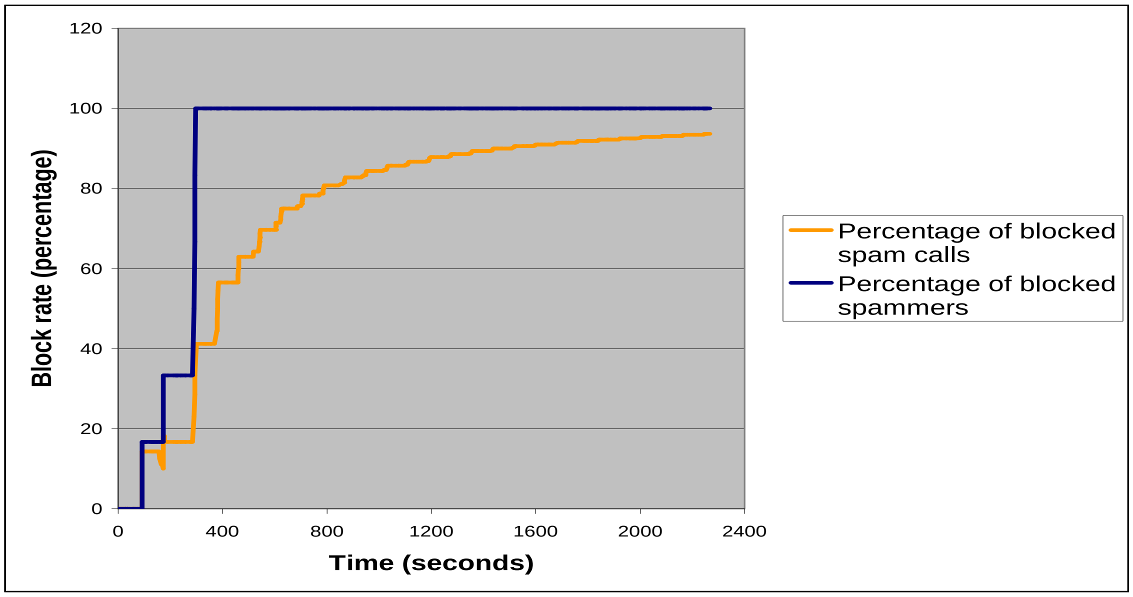

Figure 5.2: Block rate for spam calls

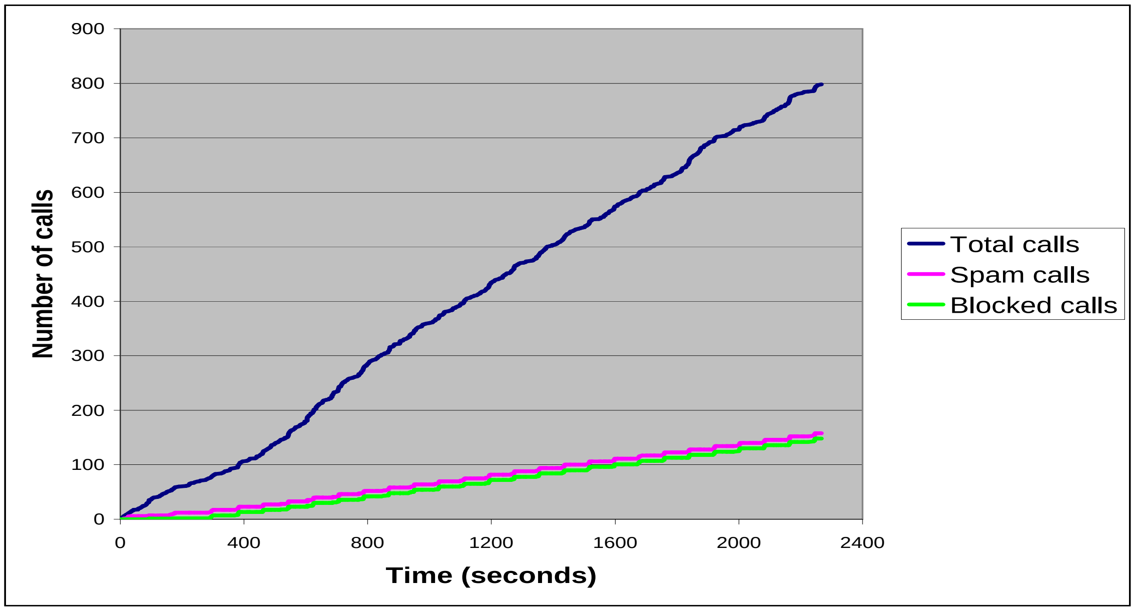

Figure 5.3: Comparison of total calls, spam calls and blocked spam calls

5.3 Spam Ratio

Figure 5.4 reflects the changes in spam ratio as time goes. Our VoIP anti-spam method shows its resistance to spams even at the early stage of the experiment. This is due to the puzzle challenging procedure. In the learning period, black list does not contribute to the anti-spam system, since spammers are not identified yet and are still out of the blacklist. It is the puzzle solving process that consumes spammers' resources and slows down the attack. At later stage, spammers are gradually identified and reported. Black list takes effects and blocks all the spammers. From this, we can see the computational puzzle based VoIP system would be especially effective when the learning period is long.

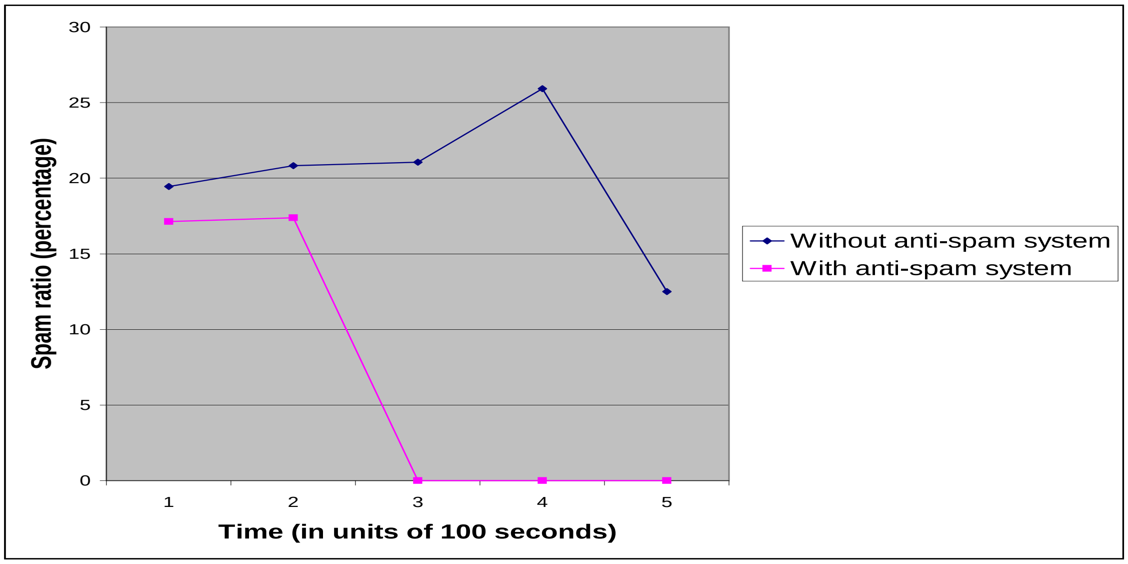

Figure 5.4: Spam ratio (percentage of successfully accomplished VoIP spam calls in all the VoIP calls)

5.4 Overhead for Proxy Server and Legitimate VoIP End Users

There is a tradeoff between anti-spam and overhead. On one hand, with our VoIP anti-spam system, legitimate VoIP end users can be protected from VoIP spams. On the other hand, proxy servers and legitimate VoIP end users have to sacrifice some computing resources in terms of delay for this anti-spam benefit.

Note that legitimate users usually make one call at a time. So the overhead for legitimate users, could be evaluated in terms of the delay in puzzle generation and verification, puzzle solving as well as the added communication delay. For VoIP calls, communication delay could be neglectable. We focus on the delay spent on the puzzle generation and verification procedure for callees or proxy servers, and the delay spent on the puzzle solving procedure for legitimate callers.

Firstly, we do a simple experiment, which includes two scenarios. One scenario is that calls are between VoIP end users in the same domain and no proxy server is involved. Puzzle generation and verification is performed on the callee side. The other scenario is that callers and callees are in different domains. Puzzle generation and verification is performed on the proxy server only. In both scenarios, there is only one call at a time.

We obtain the delays by logging the receiving and sending time of SIP requests/responses, and the results are listed in table 5.1. The results show that the added cost for proxy servers and callees are both very low and could be neglectable. And compared with the puzzle solving delay, the delay introduced by puzzle generation and verification could be neglectable.

Note that proxy servers are different from callees in that a callee can only deal with one call at any time while a proxy server may process a great amount of calls simultaneously. So it is necessary to consider the accumulated overhead when a SIP proxy server are processing multiple puzzle generation and verification simultaneously. Figure 5.5 illustrates this case. With the increasing of simultaneously anonymous calls, the cost per call, in terms of the processing time (including both puzzle generation and verification) is only slightly increased. In view of the small overhead for puzzle generation and verification, as we can see from table 5.1, we believe that both proxy servers and legitimate VoIP end users can tolerate such light overhead.

Table 5.1: Overhead for proxy servers and legitimate VoIP end users

Procedure	Time (seconds)
Without proxy server	
Puzzle generation	
Callee: receive INVITE request → send 419 “puzzle required” response	0.05
Puzzle solving (cost difficulty level is 18)	
Caller: receive 419 “puzzle required” response → send re-INVITE request	5.65
Puzzle verification	
Callee: receive re-INVITE request → send 180 Ringing response	0.01
With proxy server	
Puzzle generation	
Proxy: receive INVITE request → send 419 “puzzle required” response	0.01
Puzzle solving (cost difficulty level is 18)	
Caller: receive 419 “puzzle required” response → send re-INVITE request	5.65
Puzzle verification	
Proxy: receive re-INVITE request → forward re-INVITE to callee	0.03

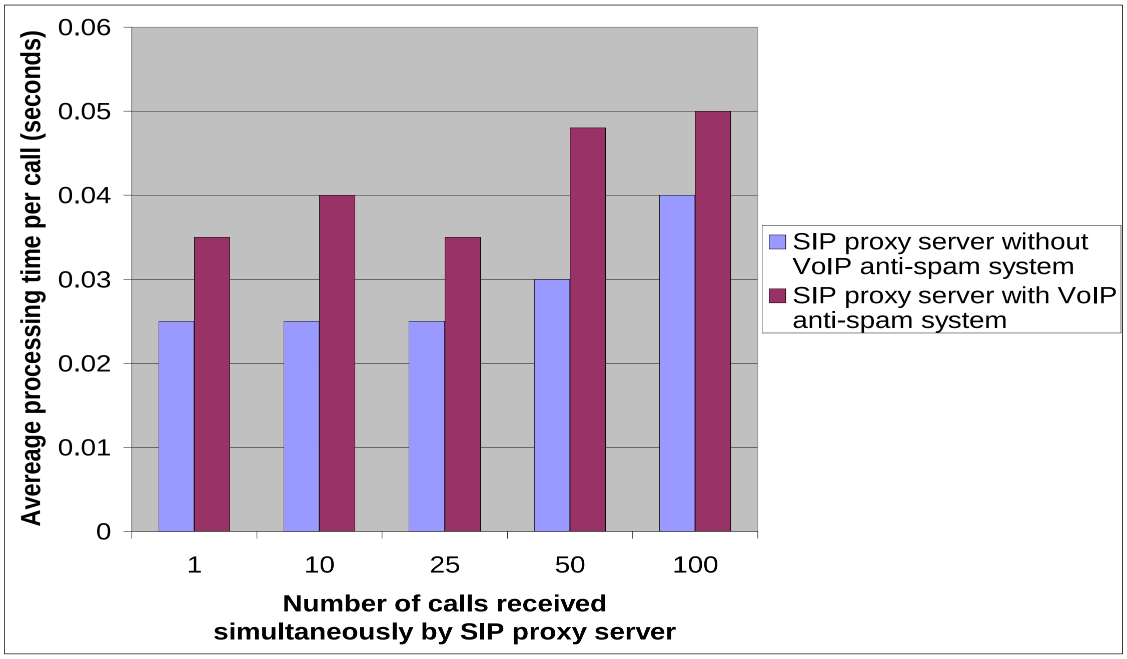

Figure 5.5: Overhead for SIP proxy server

5.5 Spam Call Generating Ability

One main difference between legitimate VoIP users and VoIP spammers is that spammers usually make a large amount of calls simultaneously. Our computational puzzle based method has great advantage in this situation. By consuming computational resources with each call, our method can punish VoIP spammers much severely than legitimate anonymous VoIP users. Figure 5.6 illustrates that the spam call generating ability of a VoIP spammer is effectively limited by the computational puzzles. The More calls made simultaneously, the more severely this spammer is punished.

Figure 5.7 illustrated the impact of different puzzle cost levels on a VoIP spammer. The processing time spent on solving puzzles was increased with the puzzle difficulty levels. When the required computational resources (CPU, memory, etc) outreach spammers' capability, the processing time per call would be increased dramatically. At this time, VoIP spammers either have to decrease the spam call generating rate, or add more computational resources which will make their spam calls less profitable.

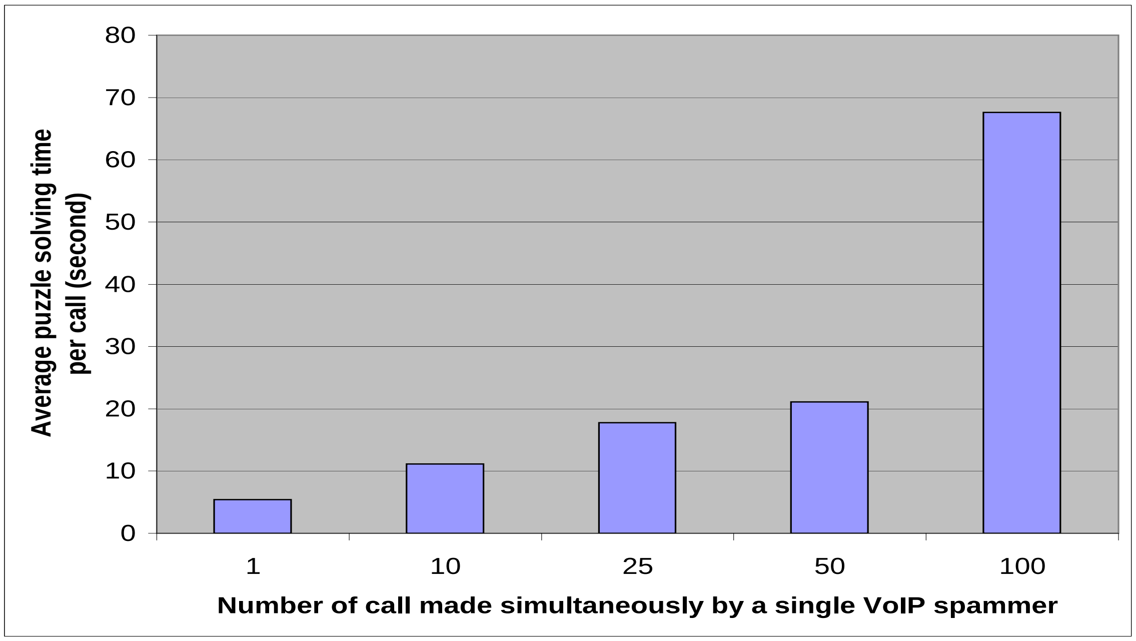

Figure 5.6: Average puzzle solving time per call for a VoIP spammer (with network size)

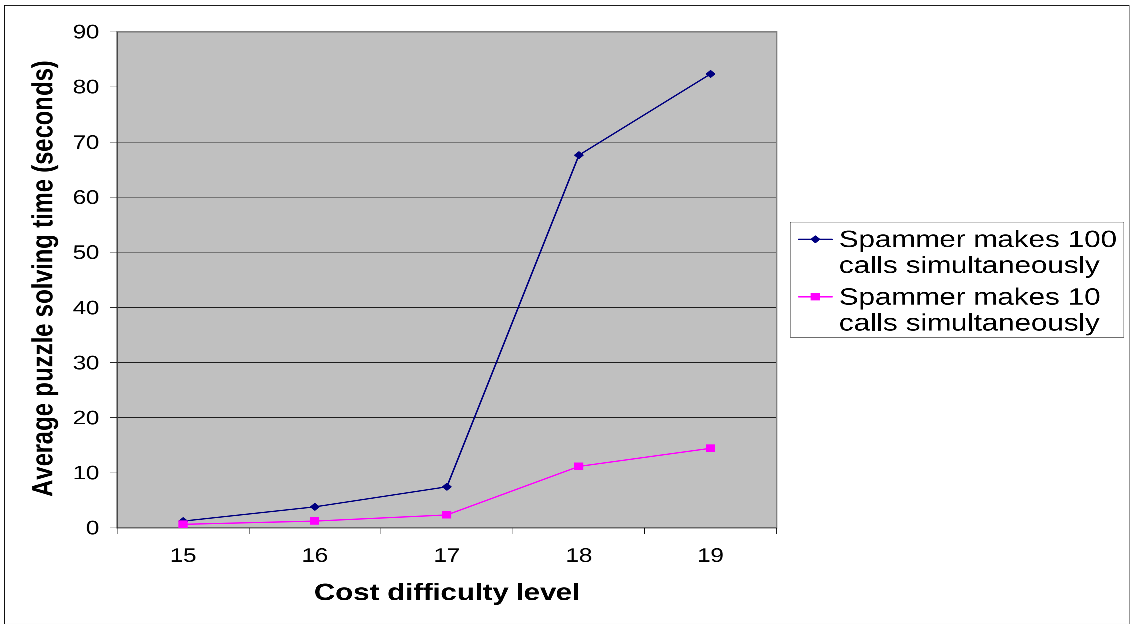

Figure 5.7: Average puzzle solving time per call for a VoIP spammer (with cost difficulty level)

5.6 Adaptive Cost Assignment

In our VoIP anti-spam system, we use an adaptive cost assignment method, as discussed in section 3.4. The benefit of this method is punishing spammers more severely than legitimate end users. With the same experimental scenario in section 5.2, figure 5.8 shows the changes in difficulty levels for legitimate anonymous calls and spam calls respectively. Note that difficulty level 30 is not a real difficulty level, instead, it is used to indicate the call is blocked. We can see from the figure that the difficulty level of legitimate anonymous calls have a slight increase at the beginning. This is because the spammer ratio in the call generating domain increases. When the spammer ratio in a specific domain increases, the overall puzzle cost applied to the calls from this domain increases. However, this would not have significant impact on innocent legitimate callers. With the increasing of puzzles solved by a legitimate caller, the puzzle difficulty level for this specific caller would drop gradually. It is noticeable in the figure that the difficulty level for legitimate users has a trend to decrease. We can also clearly see from the figure that the difficulty level for spammers increases after the early stage of the experiment and kept higher than that of innocent legitimate callers. This is because the adaptive puzzle cost for spammers increases with the increasing of spam reports received at the proxy server. The adaptive puzzle cost method enables the dynamic adjustment of puzzle difficulty levels based on users' behavior. Through the comparison with static cost assignment, we can see with the time going innocent callers would be encouraged while spammers would be punished or blocked.

From figure 5.9, we can see the advantage of adaptive cost assignment in another view. It shows the changes in puzzle solving time for legitimate anonymous calls and spam calls respectively. Note 120 is not a real puzzle solving time, instead, it is used to indicate the call was blocked.

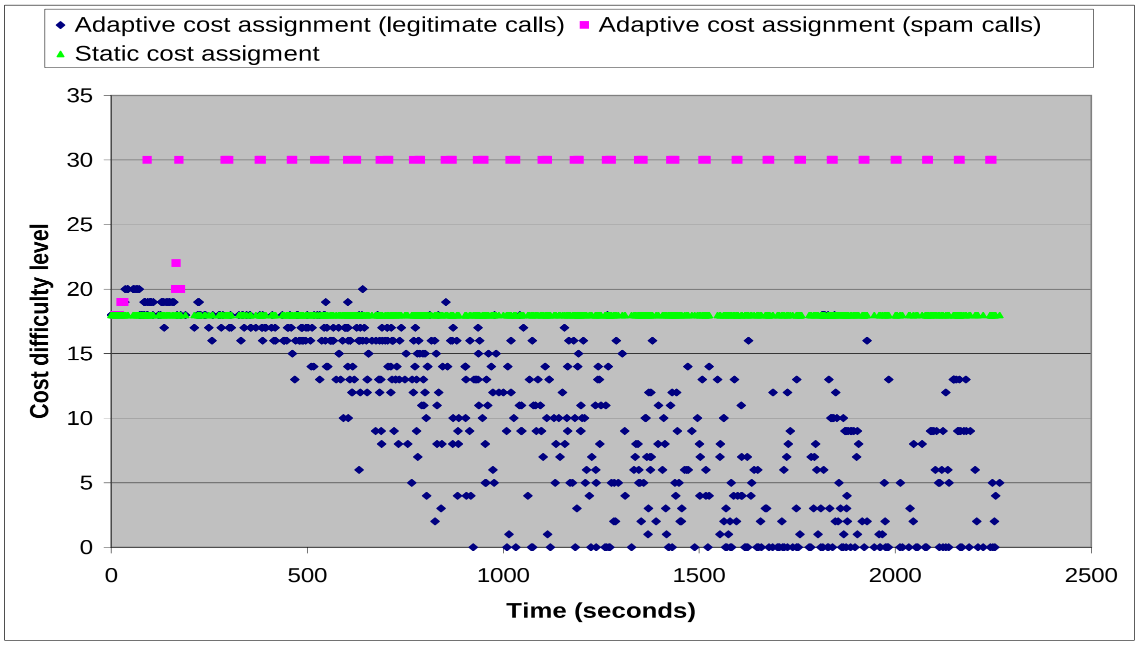

Figure 5.8: Comparison of assigned puzzle cost for spam calls and legitimate calls

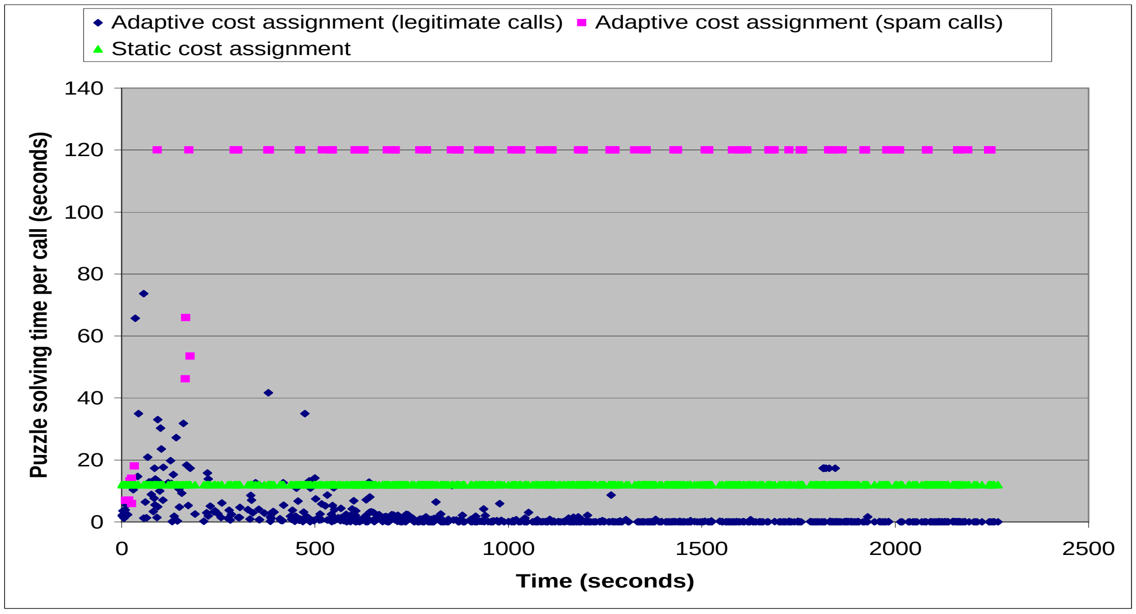

Figure 5.9: Comparison of puzzle solving time for spam calls and legitimate calls

5.7 Smart Spammers

In the previous experiments, we assume the spammers adopt the best effort strategy, in which spammers keep making spam calls as long as they have resources, regardless of the cost of the computational puzzles. However, in the perspective of spammers, it is essential to protect their own computing resources for launching large scale spam attacks. Another possible strategy, is a smarter one. The spammers could adapt their behaviors based on their cost and only make spam calls when the cost is low. If an attempt to make a spam call leads to a high cost, the spammer abandons it immediately without devoting any resources. Figure 5.10 shows the changes in difficulty levels for legitimate anonymous calls and spam calls under the two different spammers' strategies. Same as figure 5.8, the difficulty level 30 is not a real difficulty level, instead, it is used to indicate the call is blocked. We can see from the figure that the only benefit that smart spammers can get is that they can send more SIP INVITE requests to SIP proxy server. But these SIP requests cannot be handled by SIP proxy server until puzzles are solved. So a spammer with smart strategy will solve the same number of puzzles as he does not use smart strategy. To save the computing resource of SIP proxy servers, we can log the number of times that an anonymous caller does not send back puzzle solution. When this number reaches a limit, we can block this caller directly.

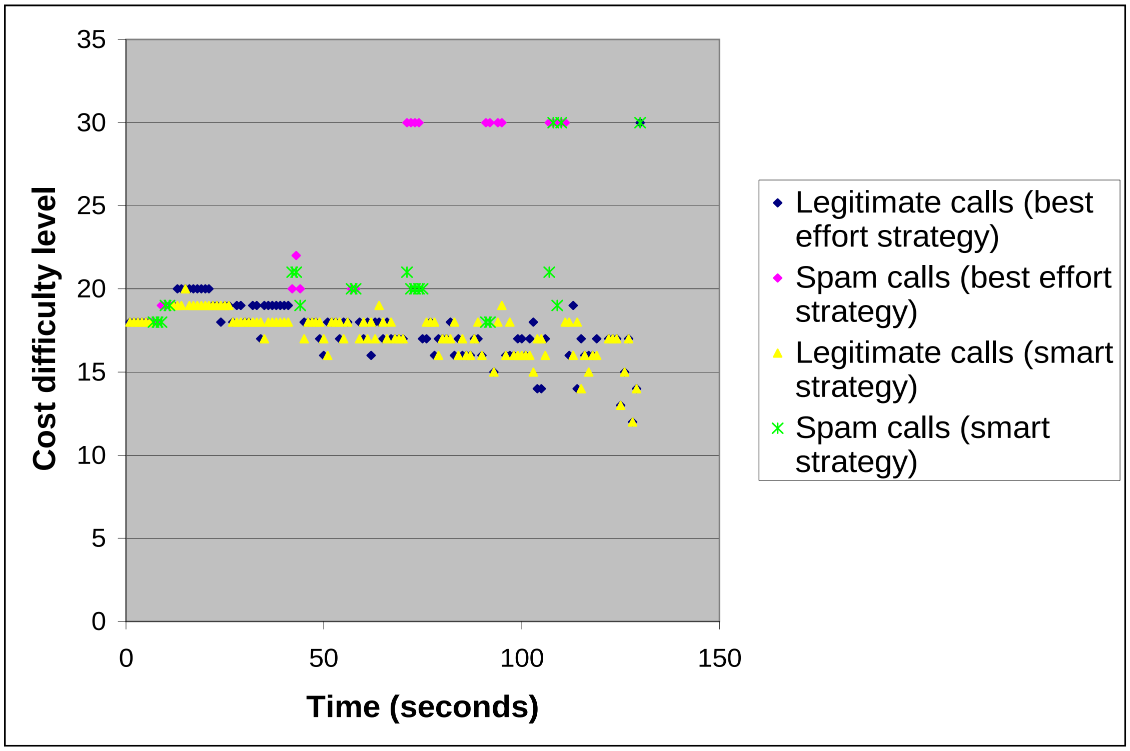

Figure 5.10: Spammers' strategies: smart strategy vs best-effort strategy

Chapter 6

Conclusion and Future Work

VoIP spam is a very real threat for current and future VoIP users. The differences between VoIP spam and email spam indicate that current email anti-spam methods cannot be directly applied to VoIP anti-spam problem. In this thesis, we classify and analyze current VoIP anti-spam methods, and develop and implement a computational puzzle based VoIP anti-spam system. This system allows VoIP users to accept anonymous calls while mitigating VoIP spams. The system challenges anonymous VoIP callers with computational puzzles and only permits calls with correct solution to the given puzzle. The puzzle is easy to be generated and verified, but hard to be solved. Our experimental results show, by consuming spammers' computational resources, the puzzle solving procedure can greatly limit spammers' capability. With the help of white/black list and adaptive cost assignment method, our VoIP anti-spam method slows down spam generation while only adding little cost for legitimate VoIP callers and proxy servers. Moreover, our VoIP anti-spam system can be either used as a separated VoIP anti-spam method, or be integrated as a part of other VoIP anti-spam system.

6.1 Future Work

Similar to most existing VoIP anti-spam techniques, our method imposes requirements on VoIP devices. The computational capability of VoIP devices varies. VoIP softphones, which runs on computers, differs greatly with VoIP hardphones in computational resources. To choose an appropriate puzzle difficulty level for the devices, we need to work out a mechanism to identify the device type. In [12], the author proposed a passive fingerprint method to check the information of remote VoIP devices. In the future, we could integrate such VoIP device identification method to our VoIP anti-spam system and make our system more stronger.

In our VoIP anti-spam system, both proxy servers and callees can use computational puzzles to challenge anonymous callers. Configuring puzzle challenge function at both would introduce cooperation problem. One solution is that we could let proxy servers insert a tag after it verifies the solution. When a callee receives a call with the tag, it would not challenge the caller any more. This method requires a mutual authentication scheme between the proxy server and callees. Since they are in the same domain, we can use a secure link, such as TLS, to complete the mutual authentication. This would be an interesting topic for further research.

Bibliography

[1] Clueless virus filters spam innocent third parties. Available from URL `http://www.joewein.de/sw/spam-joejob-info.htm`.

[2] B. Campbell. The message session relay protocol, draft-ietf-simple-message-sessions-18 (work in progress), 2006.

[3] Eric Y. Chen. Confirmed cases of spit. Available from URL `http://www.voipsa.org/pipermail/voipsec_voipsa.org/2006-March/001326.html`.

[4] Cynthia Dwork, Andrew Goldberg and Moni Naor. On memory-bound functions for fighting spam. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference*, Santa Barbara, California, USA.

[5] Ram Dantu and Prakash Kolan. Detecting spam in voip networks. In *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet Workshop*, Cambridge, MA.

[6] T. Dierks and E. Rescorla. The transport layer security (tls) protocol version 1.1, rfc4346, 2006.

[7] T. Dierks and E. Rescorla. The tls protocol version 1.2, draft-ietf-tls-rfc4346-bis-04 (work in progress), 2007.

[8] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *CRYPTO '92: Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, pages 139–147, 1993.

[9] E. Edelson. Voice over ip: security pitfalls. In *Network Security, vol. 2005*, pages 4–7, 1998.

[10] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson. Rtp: A transport protocol for real-time applications, rfc3550, 2003.

[11] Evan Harris. The next step in the spam control war: Greylisting, 2003. Available from URL `http://projects.puremagic.com/greylisting/whitepaper.html`.

[12] Hong Yan, Kunwadee Sripanidkulchai, Hui Zhang and Zon yin Shae. Incorporating active fingerprinting into spit prevention systems. In *The Third Annual VoIP Security Workshop*, Berlin, Germany, 2006.

[13] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley and E. Schooler. Sip: Session initiation protocol, rfc 3261, 2002.

[14] C. Jennings. Computational puzzles for spam reduction in sip, draft-jennings-sip-hashcash-04 (work in progress). Available from URL `http://tools.ietf.org/id/draft-jennings-sip-hashcash-04.txt`.

[15] A.B. Johnston. *Sip: Understanding the Session Initiation Protocol.* Artech House, 2001.

[16] Ari Juels and John G. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 1999*, San Diego, California, USA.

[17] R. MacIntosh and D. Vinokurov. Detection and mitigation of spam in ip telephony networks using signaling protocol analysis. In *Proceedings of the IEEE Symposium on Advances in Wired and Wireless Communication, IEEE Computer Society Press*, pages 49–52, Los ALamitos, CA, 2005.

[18] Mehran Sahami, Susan Dumais, David Heckerman and Eric Horvitz. A bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.

[19] Dpt. of Information Engineering at University of Parma and by Dpt. of Information Engineering at University of Roma ˙Mjsip.

[20] J. Pessage and J. Seedorf. Voice over ip: Unsafe at any bandwidth ? In *Eurescom Summit*, Heidelberg, 2005.

[21] J. Peterson and C. Jennings. Enhancements for authenticated identity management in the session initiation protocol (sip), rfc 4474, 2006.

[22] A. Roach. Session initiation protocol (sip)-specific notification, rfc 3265, 2002.

[23] J. Rosenberg. A presence event package for the session initiation protocol (sip), rfc 3856, 2004.

[24] J. Rosenberg. A watcher information event template-package for the session initiation protocol (sip), rfc 3857, 2004.

[25] Jonathan Rosenberg and Cullen Jennings. The session initiation protocol (sip) and spam, draft-ietf-sippingspam-04 (work in progress), 2007.

[26] International Telecommunication Union. Packet based multimedia communication systems. recommendation h.323, telecommunication standardization sector of itu, 2000.

[27] Zhenyu Zhong, Kun Huang and Kang Li. Throttling outgoing spam for webmail services. In *CEAS 2005 - Second Conference on Email and Anti-Spam.*