A Holistic Security Approach for Protecting
Government eServices against
Denial-of-Service Attack

by
Mohammed Abdulaziz M Alhabeeb
BSc. Computer Information System (KSU)
Master of Network Computing (Monash University)

Thesis
Submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy

September 2012

Caulfield School of Information Technology
Faculty of Information Technology
Monash University
Australia
This thesis is dedicated to my beloved parents, who inspired me and sparked my interest in pursuing higher education, who are praying for me and who provided me with support, help and encouragement every moment along the long academic road that I have followed.

To my wife Amani whose love makes everything worthwhile.

To my daughter Dana, my daughter Almasah and my son Abdulaziz who brighten my every day.

To Professor Srini who helped me to achieve this thesis.
Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university or other institution and affirms that to the best of my knowledge, the thesis contains no material previously published or written by another person, except where due reference is made in the text of thesis.

Mohammed Alhabeeb

4 September 2012
Acknowledgements

I take this opportunity to thank and acknowledge everyone who has helped me and encouraged me throughout my PhD study. This thesis would not have been possible without their great support.

First of all I would like to give my grateful and thanks for my GOD, Allah, for giving me wisdom and strength and the opportunity to achieve this thesis.

I thank my wonderful parents, my father Abdulaziz and my mother Madawi, for their love, support, encouragement and prayers. I have been away from them for so long, and I wish to rejoin them soon so that I can show my gratitude in person.

I would like to thank my beautiful wife Amani for her patience, prayers and encouragement to complete this work. Words cannot express how much I owe her and how much I appreciate the support she has provided so that I could do this research. In addition, I would like to thank my daughter Dana, my daughter Almasah, my son Abdulaziz and my brother and my sisters for their love.

I would like to express my sincere and boundless gratitude to Professor Bala Srinivasan for accepting me to perform this work under his supervision. He has spent many hours clarifying my ideas, attending my presentations, training me and reading and correcting my this thesis drafts. In addition I want to thank Dr Phu Dung Le for accepting my application into the PhD program at Monash
University. I want also to thank Professor Frada Burstein and Dr Henry Linger for their advice and assistance. In addition I would like to thank all the department and faculty support staff for their help.

I express my thanks to my office mate Abdullah Almuhaideb for spending so much time with me in discussing academic topics and helping me to find solutions for any problem facing us. Both inside and outside the university, he is a true friend.

I would like to thank Dr Yahya Ahmed Alnashri for helping me to correct and review the mathematical parts in this thesis.

I would like to thank Dr Noriaki Sato for helping me to proof the language of my papers and improve the structure of this thesis, and also Alison Caddick, Peter Eade and Sophie Gebhardt for language editing of this thesis. I also thank Dr Megan Seen; I will never forget the help she provided in proofing the final draft of this thesis.

I thank my friends Suliman Alsaloom, Fehaid Algahtani, Waleed Alfehaid, Saleh AL-Rebish, Hamad Alsuhaim, Omer Alluhaidan and the rest of our friend group. They are family to me.

Last but not least, I would like to thank Australia for being our home for the last three years, and Monash University for being such a wonderful environment in which to do a PhD.
List of Publications


- Mohammed Alhabeeb, Phu Dung Le, and Bala Srinivasan; “Preventing Denial of Service Attacks in Government eServices Using A New Efficient Packet Filtering Technique”. *The 9th IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA 2011)*, IEEE, Korea, 2011.

- Mohammed Alhabeeb, Abdullah Almuhaideb, Phu Dung Le, and Bala Srinivasan, “A New Security Approach For Critical Information Systems: Preventing Flooding In The Non-authenticated Client Area


- Alhabeeb, Mohammed; Le, Phu Dung; Srinivasan, Bala; “Cyber Attack Defence System for Government Critical System” *The First Annual Conference for Higher Degree by Research Students in the Faculty of Information Technology (ARCHER '10)*, Melbourne, 2010.
Other Publications during the PhD


Abstract

Security is one of the most important factors in providing eServices. This is particularly the case for organizations, including government departments, that provide critical systems. Denial-of-service (DoS) attacks are a threat to systems, individuals, organizations and society because they cause damage that is far-reaching. Protecting systems against DoS attack is a priority.

To provide an acceptable level of protection for eServices against DoS attack, requirements need to be met in four key areas: functionality, security, decision accuracy and performance. A review of current models revealed that not one of the existing solutions for DoS attack satisfies all requirements.

Our search for a solution to the DoS-attack problem began with an analysis of government eService business flows. We then designed a formal-analysis model to understand and describe the critical government systems. Following this, we built a comprehensive model (DoS-PIF) that integrates the three required protection tasks – packet filtering, attack detection and attack prevention – and that includes communication controls. In order to demonstrate the practical value of the model, a security approach, called the Holistic Approach for Securing and Protecting Critical Government eService Systems (HASP-CGeSS), was derived and realized.
In order to implement the communication control in HASP-CGeSS, we proposed a Token-Filtering Technique (TFT) to provide an authentication and filtration layer for packet filtering. TFT offers fast packet filtering through the use of packet headers. We then designed the Token-Filtering-Technique Protocol (TFTP) to prevent DoS attack.

Our subsequent evaluation indicated that the TFTP enhances packet-filtering rates, prevents the spoofing of communication sources, filters packets without needing to decrypt them, protects systems from modified and malformed packets, can manage communication timing and free the client from unproductive processing, provides stateful features in a stateless connection and provides efficient communication control between comparative communication controls. In addition, the results show that the proposed protocol is efficient and DoS-resistant for legitimate users and under seven types of DoS-attack scenarios. Furthermore, the results show that (unlike other solutions) the size of the proposed repeat-communications list does not limit the number of clients. Finally, the protocol filters DoS attacks faster than existing protocols and consequently creates greater resource availability in a system because of the reduced time required to filter DoS attacks.

This research has proposed a number of innovative security advances to protect critical government systems. The outcomes of the research justify continuing investigation in this domain for the betterment of citizens, organisations, government and society.
# Table of Contents

Declaration ................................................................................................................................. iv

Acknowledgements ................................................................................................................... v

List of Publications ................................................................................................................... vii

Abstract .................................................................................................................................... x

List of Figures .......................................................................................................................... xvii

List of Tables ........................................................................................................................... xx

Chapter 1 .................................................................................................................................. 1

Introduction ............................................................................................................................... 1

1.1 e-Government .................................................................................................................... 1

1.2 e-Government Security ................................................................................................. 4

1.3 Denial-of-Service (DoS) Attack .................................................................................... 6

1.4 Motivation and Objectives of the Thesis ................................................................. 10

1.5 Research Contributions ............................................................................................. 12

1.6 Thesis Organization ....................................................................................................... 14

Chapter 2 .................................................................................................................................. 17

Background ............................................................................................................................... 17

2.1 Introduction ..................................................................................................................... 17

2.2 Denial-of-Service (DoS) Attack .................................................................................... 20

2.2.1 Logic Attack ............................................................................................................. 20

2.2.2 Resource Attack ....................................................................................................... 21
2.3 Existing Models and Solutions ................................................................. 27
  2.3.1 Packet Filtering ..................................................................................... 27
    2.3.1.1 Firewalls .......................................................................................... 28
  2.3.2 DoS-Attack Detection .......................................................................... 36
    2.3.2.1 Intrusion-Detection Systems (IDS) ................................................ 37
  2.3.3 DoS-Attack Detection and Prevention ............................................... 40
    2.3.3.1 Intrusion Detection-and-Prevention Systems (IDPS) .................... 40
    2.3.3.2 Unified Threat Management (UTM) ............................................. 43
  2.3.4 Communication-Queue Management (CQM) ..................................... 44
    2.3.4.1 SYN cache ...................................................................................... 45
    2.3.4.2 Queue-Management Policy (QMP) ............................................... 45
  2.3.5 Re-engineering Internet Routing (RIR) ............................................. 46
    2.3.5.1 DoS Network Attack, Recognition and Defence (D-WARD) ........ 46
    2.3.5.2 Traceback ....................................................................................... 48
  2.3.6 Vulnerability Analysis .......................................................................... 49
    2.3.6.1 Topological Vulnerability Analysis (TVA) .................................... 50
    2.3.6.2 Network Access Control (NAC) Firewalls ................................... 52
  2.3.7 The Communication-Controlling Model (CCM) .................................. 53
    2.3.7.1 SYN Cookies .................................................................................. 53
    2.3.7.2 The Client-Puzzle Technique ....................................................... 55
  2.3.8 Summary of the Existing Models ....................................................... 56

2.4 Key Requirements (Assessment Parameters) ......................................... 60
  2.4.1 Functionality ........................................................................................ 60
  2.4.2 Security ................................................................................................ 63
  2.4.3 Decision Accuracy ................................................................................ 64
  2.4.4 Performance .......................................................................................... 65
  2.4.5 Evaluating Existing Solutions ............................................................. 66

2.5 Conclusion ............................................................................................. 69

Chapter 3 ........................................................................................................ 71

A Security Model and Approach for Protecting Critical Government eService Systems (CGeSS) from DoS Attack ........................................... 71
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>71</td>
</tr>
<tr>
<td>3.2</td>
<td>Background to eServices in Government Systems</td>
<td>75</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Summary</td>
<td>80</td>
</tr>
<tr>
<td>3.3</td>
<td>A Formal Model for Analyzing CGeSS</td>
<td>82</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Summary</td>
<td>95</td>
</tr>
<tr>
<td>3.4</td>
<td>The Proposed Security Model</td>
<td>96</td>
</tr>
<tr>
<td>3.4.1</td>
<td>The Filtering Component</td>
<td>98</td>
</tr>
<tr>
<td>3.4.2</td>
<td>The Analysis Component</td>
<td>103</td>
</tr>
<tr>
<td>3.4.3</td>
<td>The Attack-Recognition Component</td>
<td>103</td>
</tr>
<tr>
<td>3.4.4</td>
<td>The Avoidance-Strategy-Manager Component</td>
<td>104</td>
</tr>
<tr>
<td>3.4.5</td>
<td>Summary</td>
<td>104</td>
</tr>
<tr>
<td>3.5</td>
<td>A Holistic Approach for Securing and Protecting Critical Government eService Systems (HASP-CGeSS)</td>
<td>105</td>
</tr>
<tr>
<td>3.5.1</td>
<td>The Client-Authentication (CA) Component</td>
<td>107</td>
</tr>
<tr>
<td>3.5.2</td>
<td>The Authenticated-Client-Communication (ACC) Component</td>
<td>111</td>
</tr>
<tr>
<td>3.5.3</td>
<td>The Analysis Component</td>
<td>115</td>
</tr>
<tr>
<td>3.6</td>
<td>Conclusion</td>
<td>118</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>120</td>
</tr>
<tr>
<td>4.2</td>
<td>Token Overview</td>
<td>121</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Token Specifications</td>
<td>125</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Token Generation</td>
<td>141</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Token Verification</td>
<td>143</td>
</tr>
<tr>
<td>4.3</td>
<td>TFT Filtering Lists</td>
<td>149</td>
</tr>
<tr>
<td>4.3.1</td>
<td>The Blocked-Communication-List</td>
<td>149</td>
</tr>
<tr>
<td>4.3.2</td>
<td>The Monitoring-Repeated-Communication-List</td>
<td>150</td>
</tr>
<tr>
<td>4.4</td>
<td>The Token-Filtering-Technique Protocol (TFTP)</td>
<td>155</td>
</tr>
</tbody>
</table>
Chapter 4 ........................................................................................................ 155

4.1 Communication-Control ......................................................................... 155

4.2 The DoS-Stage ......................................................................................... 157

4.3 The Initial Stage ........................................................................................ 157

4.4 The Communication Stage ........................................................................ 159

4.5 The Authentication Stage .......................................................................... 165

4.6 The Communicating Stage ......................................................................... 169

4.7 DoS-Attack Filtering, Detection and Prevention Processes ................ 171

4.7.1 The Filtering-Process .............................................................................. 172

4.7.1.1 Filtering-Process Inspections ................................................................. 172

4.7.1.2 Filtering-Process Decision Making ....................................................... 176

4.7.2 The Detection-Process ............................................................................ 178

4.7.2.1 Detecting Whether the System is Under DoS Attack ......................... 178

4.7.2.2 Identifying the Protocol Step Receiving a DoS Attack ....................... 183

4.7.2.3 Detecting the Source of the Attack ..................................................... 187

4.7.2.4 Defining the Attack ........................................................................... 188

4.7.3 The Prevention-Process ......................................................................... 189

4.8 Conclusion ............................................................................................... 190

Chapter 5 ......................................................................................................... 192

Characterization ............................................................................................ 192

5.1 Introduction .............................................................................................. 192

5.2 Comparison and Performance ............................................................... 194

5.2.1 The Meadows's Cost-Based Framework................................................. 196

5.2.2 Protocol Cost-Based Analysis ................................................................. 203

5.2.3 Protocol Assessment ............................................................................... 205

5.2.4 DoS-Resistance .................................................................................... 216

5.2.5 Operational efficiency .......................................................................... 219

5.2.6 Functionality of the Proposed Protocol ................................................ 226

5.3 Communication-Control Analysis ......................................................... 230

5.3.1 Client-Server Formal-Analysis Model ................................................... 230

5.3.2 Filtering DoS Packets Based on their Headers ...................................... 237

5.3.3 Detecting Spoofed Sources ................................................................... 240

5.3.4 Detecting the Use of Incorrect Tokens .................................................. 241
Chapter 6 ......................................................................................................... 259

Conclusion ........................................................................................................ 259

6.1 Research Contributions ............................................................................. 261

6.2 Future Work ................................................................................................ 266

References ........................................................................................................ 268
List of Figures

Figure 1.1 Architecture of an e-Government Platform ........................................ 3
Figure 1.2 Flooding Attack .................................................................................. 7
Figure 2.1 TCP-SYN Flooding Attack ................................................................ 22
Figure 2.2 Distributed Denial-of-Service Attack .............................................. 25
Figure 2.3 Setup of a DDoS Attack [70] ............................................................. 26
Figure 2.4 Packet Filtering ................................................................................ 28
Figure 2.5 Attack Detection ............................................................................. 37
Figure 2.6 DoS-Attack Detection and Prevention .......................................... 41
Figure 2.7 Communication-Queue Management ............................................. 44
Figure 2.8 Re-engineering Internet Routing ..................................................... 46
Figure 2.9 Vulnerability Analysis ................................................................... 50
Figure 2.10 Topological Vulnerability Analysis (TVA) [120] .............................. 51
Figure 2.11 The Communication-Controlling Model ...................................... 53
Figure 2.12 Key Requirements (Assessment Parameters) ............................... 61
Figure 3.1 The Process of Designing a Solution to Protect CGeSS from DoS Attack ........................................................................................................ 74
Figure 3.2 Communication for User Services: (a) Services for Browsing Users (b) Services for Anonymous Users (c) Services for Partially Authenticated Users (d) Services for Fully Authenticated Users ........................................................................ 77
Figure 3.3 The Proposed Security Model DoS-PIF ........................................... 97
Figure 3.4 Using Controls in Communication between the Client and the Server ........................................................................................................... 100
Figure 5.2  Completing the Client-Communication Process ....................... 245
Figure 5.3  Detecting Incorrect Tokens ......................................................... 245
Figure 5.4  Testing Protocol Dependability when Using the TFT .............. 246
Figure 5.5  Testing the System’s Ability to Detect Spoofed Tokens .......... 247
Figure 5.6  Testing Whether the System can Detect Expired Communication .......................................................... 248
Figure 5.7  Delayed Communication Arrives Before the Delay Time ....... 249
Figure 5.8  Testing the Monitoring-Repeated-Communication-List in Detecting Repeat-Resending Attacks when the Number of Clients is Greater than the Size of the List ........................................ 250
List of Tables

Table 2.1 Example of a State Table [73] .......................................................... 31
Table 2.2 Summary of the Advantages and Disadvantages of Reviewed Models ..................................................................................... 58
Table 2.3 Security-Solution Evaluation Using Assessment Parameters................................................................................................ 68
Table 3.1 Comparing Government eServices.................................................. 81
Table 3.2 Formal Model Notations.................................................................. 83
Table 4.1 Communication States Values......................................................... 128
Table 4.2 Example of a Token’s Random-Numbers Table......................... 130
Table 4.3 Example of a Blocked-Communication-List................................. 150
Table 4.4 Example of a Monitoring-Repeated-Communication-List........... 151
Table 4.5 Result of Testing the Monitoring-Protocol-Behaviour Technique for: (a) No Attack Received (b) Attack Received in Step 1 (c) Attack Received in Step 2 (d) Attack Received in Step 3 ................................................................. 186
Table 5.1 TFTP in the Cost-Based Framework Notation ............................. 204
Table 5.2 Characteristics of Attack Scenarios.............................................. 206
Table 5.3 Processing Cost of Selected Cryptographic Operations [145]................................................................................................................. 220
Table 5.4 Comparison of Computational Costs to the System and Client for Completing Protocol Steps for A Legitimate Client and under Seven Attack Scenarios.................................................. 222
Table 5.5 Comparison of HIP and TFTP Lists ............................................. 224
Table 5.6 Comparison of the Performance of the HIP Using the Client-Puzzle Technique and the HIP Using the TFT ........................ 225
Table 5.7 Comparing Protocol Functionality .............................................. 228
Table 5.8 Comparison of the HIP and TFTP Lists ........................................... 230

Table 5.9 Comparing Data Storage of Authentication Protocols ................. 253

Table 5.10 Comparison of the Functionality of the Proposed Solution with Existing Solutions.......................................................... 257
Chapter 1

Introduction

1.1 e-Government

Contemporary technologies provide a vast range of opportunities for organizations to extend their services into the digital arena. With the emphasis on digital delivery, the dependence on information and telecommunication technologies is also increased.

In utilising the Internet, businesses can function internationally with efficiency, rapidly exchanging information and communicating with their partners, suppliers and customers with ease. Governments also use the Internet to provide online information and to replace manual systems when providing government services [1, 2]. This is known as e-Government.

There is no unique definition for electronic government [3]. e-Government can be defined based on processes and relationships. As such, one definition of e-Government is that it is “…the continuous optimization of service delivery, constituency participation, and governance by transforming internal and external relationships through technology, the Internet, and new media.” [4]. In
addition, e-Government can be defined as a method of providing interactions; e-Government is “...a form of organization that integrates the interactions and the interrelations between government and citizens, companies, customers, and public institutions through the application of modern information and communication technologies” [5].

Fundamentally, e-Government refers to the use of information and communication technologies (ICT) in the process of public administration to facilitate access to government information and services for citizens, government agencies and businesses [6]. As such, an e-Government system should provide government services to its users over open networks and from anywhere at any time [7, 8].

Information technology provides possibilities for further developing the concept of e-Government; however, along with the possibilities is the need to define the infrastructure challenges associated with providing this service. These challenges include [9, 10]:

i. making the information acceptable universally;

ii. ensuring information availability with equality of access;

iii. ensuring security and privacy;

iv. ensuring information usability for public and private organizations;

v. archiving and managing electronic information records; and

vi. developing methods for information technology resource management.
In response to these requirements, citizens, government agencies and public authorized employees can use e-Government services through a universal entrance point – the government portal (Figure 1.1) [6, 11]. This portal can be accessed locally through the government’s system networks, or remotely through the Internet, and through either wired or wireless networks. The government system exchanges information with users in order to provide e-Government services using several collaborative client–server information systems [6].

Figure 1.1 Architecture of an e-Government Platform
In Saudi Arabia, it is intended that all government services will be provided through one government portal. These services are important for all citizens and residents of the country. In addition, other public and private organizations, such as banks, hotels and hospitals, will also be able to communicate with the government through the same portal.

While the provision of e-Government offers numerous benefits, designing such a system opens up issues of security and the associated system requirements. These issues are discussed in the following section.

### 1.2 e-Government Security

Security is one of the most important factors in online services, particularly in the case of organizations such as government departments that have highly critical systems. By critical systems we mean systems that support services that have high availability requirements and are vulnerable to DoS attack [6, 9, 12, 13]. These systems should resist interruption by any kind of attack. Yet, as a result of the growing dependence on information infrastructures to support critical operations, cyber attacks are a major threat to an organization’s computer and network systems. These critical systems need to be protected in order to guarantee a number of security goals. These goals are defined as follows [14, 15]:

i. **Confidentiality**: This is the security goal that creates the requirement for protecting a computer system from deliberate or unintentional
attempts to perform unauthorized readings of data in storage, in communication and during processing. In an e-Government system, user information, transactions and requests should be protected from being read by others at all times.

ii. **Integrity**: This is the security goal that creates the requirement for protecting a computer system from both intentional and unintentional attempts to violate data or system integrity.

iii. **Accountability**: This is the security goal that creates the requirement for tracing actions in entities. This goal supports the non-repudiation goal. An e-Government system should be able to record and log any reading or modification action (such as adding, deleting or altering data).

iv. **Availability**: This is the security goal that creates the requirement for protecting a computer system from intentional or unintentional attempts to cause a denial of service or to carry out an unauthorized deletion of data.

These security goals are currently used to protect e-Government services; however, reports of attacks on e-Government services continue to surface. For example, in July 2009 the websites of the South Korean and North American governments were shut down for more than two days as a result of DoS attacks. Users were disconnected from the sites’ services until the sites recovered from the attacks, causing a high degree of disruption. Because the
culprits could not be immediately identified [16], it was not possible to take instantaneous defensive action and thus mitigate the effects of the attack. It is clear, therefore, that online government services need to be permanently guarded against DoS attack.

1.3 Denial-of-Service (DoS) Attack

Denial-of-Service (DoS) attacks are “...characterized by an explicit attempt to prevent the legitimate use of a service” [17]. DoS attacks are committed using the victim’s own resources to slow down or stop one or more of the victim’s key resources (for example, the CPU, bandwidth or buffer). The attack might take advantage of the vulnerabilities of the victim’s own system to commit the DoS attack [18]. The goal of DoS attack is to either retard or totally prevent the provision of services to clients [18]. DoS attacks on online services, such as e-Government services, affect information availability [6, 9, 13, 19].

Flooding attacks are the main type of DoS attack. In this type of attack, the attacker overpowers the victim’s system with DoS packets (Figure 1.2). Because it is nearly impossible to stop packets that are addressed to a host, internet protocol (IP) routers respond to this surfeit by dropping packets arbitrarily when an overload occurs. In a flooding attack, the issue is determining which packets to drop [20, 21].
Flooding attacks can degrade the quality of service (QoS) in a network and lead to the interruption of critical infrastructure services [22]. Like all DoS attacks, they are a serious threat to the security of networks that provide public services, such as government portals [23], and they are even more difficult to combat if the IP address is spoofed [24].

Another issue to consider is encryption. Communication encryption must be included in critical system security design to achieve the goal of confidentiality. Therefore, critical government services usually require a Public-Key Infrastructure (PKI) encryption method. However, using a strong encryption method, such as PKI, renders a system more vulnerable to DoS attack, because the system needs more time to process encrypted packets.
Protocols that use an encryption solution should be designed carefully because the use of data encryption necessitates an overhead communication process. Dropping flooding packets from their headers is generally the easiest way of providing a high packet-filtering rate [18, 25-27].

A Distributed Denial-of-Service (DDoS) attack is a type of attack that is performed from several subverted machines. Large-scale DDoS attacks performed over the Internet can cause serious harm to public services [28, 29]. DDoS attacks are becoming more sophisticated and automated, creating an even greater threat [30]. Older forms of attack were basic and thus easy to detect and prevent. The newer forms of attack, such as ‘zombie’ attack, are complicated and therefore harder to detect and prevent [31-38].

The security issues that arise from the Internet also need to be considered. The Internet was designed chiefly to provide networking. It provides cheap, easy and fast communication mechanisms that are enabled by higher-level protocols. These protocols guarantee reliable and fast delivery of communication and a convenient level of service [18]. However, with the focus on networking, the Internet’s initial design lacked consideration of the security that would be required. This lack of design consideration opens up the following security issues and makes DoS attack possible.

i. **Internet security is highly interdependent**: An attacker can use systems that lack security to target more secure systems. The insecure systems are used to perform the DoS attack on the victim’s system,
even though the victim’s system may itself have a high level of security [39].

ii. **The Internet is comprised of limited and consumable resources:** Many of the entities on the Internet have limited resources (such as networks, hosts or services) that are shared by many users. If a DoS attack is successfully directed at any one of these entities, it will affect the machines that share the entity’s resources [18, 39].

iii. **The power of few is less than the power of many:** A DoS attack can be committed using a high number of machines. If the resources that are used in the attack are greater than the victim’s resources, then the DoS attack will be successful [18].

iv. **The victim’s processing of the DoS attack creates an overhead:** In a DoS attack, the attacker can process a packet just once. This packet can then be repeated many times against the victim without the need for reprocessing. On the other hand, for each repeat packet received, the victim must expend resources processing it [18].

v. **The distributed design of the Internet:** The policies of one host cannot be implemented on other Internet hosts [18].

vi. **Insufficient accountability:** Usually the attacker uses spoofed sources or remote machines to commit DoS attack. This makes it difficult to identify the source initiating the attack[18, 40].
vii. **Insufficient attention paid to security**: Organizations usually pay more attention to the reliability and simplicity of services provided; security is a secondary priority.

viii. **Insufficient focus on the problem of DoS attack**: In addition to the point above, security managers often regard the defence against DoS attack as an additional job. Generally, the priorities are the security goals of confidentiality, accountability and integrity.

ix. **Insufficient balance in achieving the security goals**: Achieving security goals involves trade-offs between the goals. For example, a high level of encryption can ensure confidentiality and security. This, however, can make a system vulnerable to a low-rate DoS attack and consequently affect the security goal of availability.

These security issues render the task of securing critical government systems particularly challenging.

### 1.4 Motivation and Objectives of the Thesis

As already mentioned, in Saudi Arabia, it is intended that all government services will be provided through the one government portal. Other public and private organizations will also communicate with the government through the same portal. Therefore, these government services need to have high availability at all times. However, e-Government services cannot provide an acceptable level of availability if systems are not protected against DoS attack.
DoS attacks against highly visible Internet sites or services have become commonplace [41]. These attacks endanger Internet operations and cause substantial financial damage [13, 42]. DoS attack is one of the major problems faced by Internet hosts [43]. According to the 2009 CSI Computer Crime and Security Survey report, 29.2% of the respondents detected DoS attacks directed to their organizations. Respondents also stated that, of cyber attacks, DoS attacks were the most costly [44].

The sophistication of DoS-attack methods and strategies has overshadowed progress in the development of adequate security systems [45], and there is a very real need for a defence system that can protect critical government systems. However, not one of the existing solutions can efficiently and accurately filter, detect and prevent DoS attacks on government systems in real time.

The aim of this thesis is to design a solution for filtering, detecting and preventing DoS attacks on e-Government services. In order to do this we compare the different services that are provided by government systems in order to determine those that are critical. We conclude that the services that can only be used by authenticated clients are the most critical and we design a model to protect these services. We then use this model to develop an approach for protecting critical government services. The proposed solution has fully integrated attack-filtering, detection and prevention components. These
components collaborate in protecting critical government services against DoS attack during communication with clients by using a DoS-resistant protocol.

### 1.5 Research Contributions

The main contributions in this thesis are as follows.

- **A process to analyze business flows for all government online services:**
  The analysis process enables us to identify the critical eServices; that is, those electronic services that take priority in terms of needing protection from any disconnection. The analysis also helps us to investigate vulnerabilities in communication processes that might be used by attackers to commit DoS attack.

- **The development of a formal model for analyzing critical government eService systems:** This model is important in characterizing the type of eService, and helps us to identify the system’s elements, processes and requirements in order to protect it against DoS attack.

- **The development of a new security model to protect critical government eService systems against DoS attack:** We design this model based on the formal model’s specifications and requirements. The proposed model – called DoS-PIF – provides the communication that is required to secure the critical services while protecting the system against DoS attack and avoiding the limitations of existing models. This model performs three
main tasks: (i) packet filtering; (ii) attack detection; and (iii) attack prevention.

- **The development of a security approach based on the proposed model:**
  This approach is designed to filter, recognize and avoid DoS attacks on critical government systems. This approach divides the communication point between the client and the server into two communication points. This division is used to: (i) identify the nature of the client’s communication activities with the system; (ii) provide an appropriate solution for DoS attacks in every part of the system; and (iii) secure every stage of the client’s communication with the system.

- **The development of a new communication control that can be used in the proposed packet-filtering mechanism:** This communication control is in a plain-text format in order to enhance the packet filtering based on the control. In addition, it provides complete information about the communication between the server and the client. It also shows any expired communication and can be used to distinguish between clients by using flags. To enhance the filtering mechanism, we designed a new list mechanism in order to detect repeat communications. These lists are not limited by the number of clients and have the same level of efficiency as normal communication repeat lists.
• The development of a communication protocol that can be used to provide critical government eServices: This protocol is DoS resistant and has greater scalability and efficiency than existing protocols.

• The development of a technique for monitoring protocol behaviour: This technique can be used to detect the communication step in which a repeat attack is received.

• The development of a formal protocol-analysis model: This model can be employed to analyze and evaluate the quality of protocols and solutions used to filter DoS packets.

1.6 Thesis Organization

The thesis is organized as follows.

Chapter 1 defines e-Government and introduces its security requirements. The motivation of the thesis is outlined, along with its objectives and theoretical contributions. The chapter closes with a description of the thesis organization.

Chapter 2 gives an overview of DoS attacks and the various types of attack, followed by examples of common DoS attacks. The chapter discusses the advantages and disadvantages of the existing DoS security models. It also provides an overview of the solutions and countermeasures that are based on the existing security models and discusses their limitations. The chapter then
outlines the assessment parameters relevant to the DoS problem and uses these to evaluate existing solutions.

**Chapter 3** analyzes government eServices in order to determine those that have the highest need for protection against DoS attack. In addition, a formal model is established to analyze and characterize these services. This is followed by the presentation of a security model – Dos-PIF – whose design is based on the formal model components and requirements. Finally, using the proposed security model and in response to the model’s goals, a security approach - HASP-CGeSS - is proposed for protecting critical government eService systems against DoS attack.

**Chapter 4** further elaborates the proposed approach. A new communication control mechanism is proposed. The mechanism incorporates filtering lists based on the communication control mechanism methodology in order to detect and block repeat communications. In addition, a new DoS-resistant protocol is designed. This protocol uses the proposed communication control as an essential part of all its messages to provide an efficient communication processing. A new protocol monitoring technique to detect the step at which a repeat attack is received is also presented in this chapter.

**Chapter 5** proposes a formal-analysis model to evaluate the solutions and protocols for protecting systems against DoS attack. In addition, the proposed solution and its techniques are evaluated and compared to the existing solutions using the assessment parameters discussed in Chapter 2.
Chapter 6 summarizes the achievements and contributions of this thesis. The chapter concludes with a discussion of the potential for future research in this area.
Chapter 2

Background

2.1 Introduction

With the developments in information and communication technologies, and the increase in the accessibility of the Internet, organizations have become vulnerable to threats from both insiders and outsiders [46]. A threat is described as a potential undesirable incident [47]. Information systems are constantly exposed to various types of threats. Exposure to these threats can have a serious impact on an organization and can lead to substantial financial loss. The size of the damage caused by an attack can range from small errors, which only harm the integrity of a database, to those that destroy whole computer centres and bring business to a standstill [48]. Protecting assets from such threats has thus become a major concern for both individuals and businesses [49].

The term ‘security countermeasure’ refers to methods that detect, prevent and/or minimize losses associated with one or more information security threats [50]. In order to design effective countermeasures against such threats we need to know the specifications of the threats.
DoS attack is one of the most dangerous threats to electronic services [51]. Many solutions, based on security models, have been suggested and implemented to filter, detect and prevent DoS attack. Some solutions are designed to deal with a specific kind of DoS attack while others are designed to deal with either a large volume or a lesser type of DoS attack.

Each of these solutions has its strengths and weaknesses. In order to select a solution that can be used to protect critical government eService systems (CGeSS) against DoS attack we need to establish key requirements to ensure the required level of protection in the solution. These key requirements can be defined based on the DoS-attack problem and the strengths and the limitations of the existing solutions. A number of steps are involved in identifying these key requirements.

The first step is reviewing the DoS-attack problem by:

- defining the types of DoS attack;
- identifying the most dangerous type (that produces the greatest damage to systems);
- discussing common examples of this type; and
- understanding how the attack is performed and the target of each type of attack.

The second step is reviewing the strengths and the limitations of the existing models and solutions by classifying the existing solutions into models based on
their functionality. This will help us to identify the common advantages and disadvantages for each group of solutions. To do this, we need to discuss at least one example of a solution for each model and determine its strengths and limitations.

The third step involves defining the key requirements based on the DoS-attack problem and the strengths and the limitations of the existing solutions. Additionally, these key requirements can be used as assessment parameters in order to evaluate the existing solutions.

To follow these steps, in this chapter we begin with an overview of the types of DoS attack and provide examples of the most common attacks. We also discuss the models and solutions that have been previously proposed to protect systems against DoS attack. This discussion includes the advantages and disadvantages of these models and the strengths and limitations of the countermeasures. This will help us to identify the key requirements that need to be satisfied by the solutions in order to protect systems against DoS attack; it will also enable us to identify gaps in the existing body of security knowledge.

The remainder of the chapter is organized as follows.

In the next section (Section 2.2), a background to the types of DoS attack is provided. Section 2.3 gives an overview of the existing models and solutions that are designed to protect systems against DoS attack. This discussion will assist us in identifying the key requirements of a solution, which are discussed in Section 2.4. These key requirements are then extended so that they can be
used as assessment parameters to evaluate existing solutions. The final section of the chapter presents the results of the analysis and provides our conclusions.

2.2 Denial-of-Service (DoS) Attack

In this section we provide background information relevant to DoS attacks. These can be categorized into two main types: logic attack and resource attack.

2.2.1 Logic Attack

Logic attacks are the exploitation of an existing software vulnerability in order to cause a crash or to slow down system performance. Malicious packet attack is a type of logic attack (which is also called a malformed packet attack) that occurs when an attacker sends incorrectly formatted packets to a system to cause it to crash. An example of a malicious packet attack is the Ping-of-Death attack [52].

Two types of malicious packet attack are frequently encountered: a packet-address attack and a packet-attribute attack [52]. In an address attack, the attacker changes the source of the packet by spoofing (for example) the receiver’s IP address in the packet’s source. In a packet-attribute attack the attacker changes (for example) the packet’s field or size by changing the sequence of the packet.

In both these forms of attack, the aim of the attacker is to send packets that target any vulnerability in the receiver’s protocol. These attacks cause a delay
or halt the protocol. Malicious packet attacks are becoming more prevalent as attackers continue to identify the limitations of protocols and applications [53].

While some solutions have been developed to defend against malicious packet attacks, attackers constantly find ways to commit new malicious packet attacks [54]. Nonetheless, old malicious packet attacks can still be found [53]. This is because some of the new operating systems and devices are vulnerable to these older forms of attack. For example, the beta version of Windows Vista was vulnerable to a number of old attacks [54] (including Land attack [55]).

While a logic attack is a serious and ongoing threat to computer systems, many attacks can be prevented by filtering particular packet sequences or by upgrading faulty software. As such, the malicious packet attack is not a focus of this thesis [18, 26].

### 2.2.2 Resource Attack

A resource attack occurs when an attacker sends a large quantity of packets to a victim’s site. In this scenario, the victim’s key resources (for example, the CPU, memory or other network resources) can be crashed or slowed as a result of handling a quantity of packets. However, because it is not easy to distinguish between ‘good’ and ‘bad’ communications, it can be extremely difficult to protect systems against a resource attack [18, 26]. In order to understand this type further we need to discuss common examples, as follows.
**TCP-SYN Flooding Attack:** In this attack, the attacker takes advantage of a TCP vulnerability in order to commit the attack. Under the TCP protocol, the client requests a connection from the server by sending a TCP-SYN packet. When the packet is received, the server stores the client information in a backlog queue (or buffer) and then replies to the client with a SYN-ACK packet. This means that the request has been received and that the server is waiting to complete the connection steps, because the connection is half open [56] (Figure 2.1).

![TCP-SYN Flooding Attack Diagram](image)

**Figure 2.1 TCP-SYN Flooding Attack**
At this stage, the legitimate client should reply with an ACK packet to the server to complete the connection establishment. However, an attacker performs the first step many times by sending a high volume of request packets, without performing the third step of the communication. As a result, the server’s buffer becomes full of bogus half-open connections after a certain period of time. Consequently, the legitimate client’s requests cannot be served [56] (Figure 2.1). In this kind of attack, the aim of the attacker is to perform a DoS attack on the memory of the server.

**ICMP-Flooding Attack:** In this form of attack the attacker sends a flood of *ICMP_ECHO* packets to the server. The server then replies to each request, fully occupying the server’s CPU. The source of the packet is usually spoofed. The purpose of an ICMP flooding attack is to perform a DoS attack on the server’s CPU [18, 57, 58].

**Smurf Attack:** This attack is a reflector attack [18], or indirect attack, where the attacker spoofs the victim’s source in DoS-attack packets and then sends these packets to other computer sources. These other sources then reply to the packets’ apparent source, the victim [59-61]. To carry out the attack, a high volume of *ICMP_ECHO* requests are sent to random addresses in the Internet, spoofing the server’s IP address in the source of the requests. The different machines reply to the requests, overwhelming the server’s network and fully occupying the server’s CPU as it tries to handle the requests [18].
**Domain-Name Service-Reflector Attack:** In this attack the attacker generates a high volume of DNS requests to different servers by spoofing the victim server’s IP address in the source of the requests. The servers respond to the DNS requests, keeping the victim’s network and its CPU extremely busy [18].

**Key-Exchange Attack (Key Establishment):** The key-exchange protocol is based on the key-agreement protocol proposed by Diffie-Hellman. These types of protocols authenticate the two pairs of the communication by using the public-key infrastructure, which requires computationally expensive operations. This requirement makes these types of protocols vulnerable to low-cost processing attacks, which affect the server’s CPU [62-67]. In addition, this type of authentication is vulnerable to memory attacks that occur when the attacker sends a high volume of requests to the server by spoofing the sources of the requests with multiple, different, random IP addresses. The server experiences a memory DoS attack, as each request needs to be stored in a backlog queue [68].

**Distributed Denial-of-Service (DDoS):** This attack is a type of DoS attack performed by using several subverted machines (agents or zombies). The most frequent scenario under this attack is that the machines are all simultaneously engaged in sending multiple packets to the victim (Figure 2.2). The aim of a DDoS attack is to interrupt the victim’s Internet services by sending a high volume of flooding packets from multiple sources [69].
Figure 2.2 Distributed Denial-of-Service Attack

DDoS attacks have become sophisticated and highly automated. Using attack tools such as the Tribal Flood Network (TFN), TFN2K, Trinoo and Stacheldraht, almost anyone can perform a DDoS attack [31-37]. A single attacker can set up a number of machines to perform the DoS attack (Figure 2.3), thus increasing the impact of the attack. Figure 2.3 shows how an attacker can attack a high number of vulnerable machines and set them up as agents, which are then divided into groups. Each group can be controlled by another hacked machine, which is set up as a master. At this stage the attacker can control each of the machines in order to perform the DDoS attack on a server.
In order to protect systems against DoS attacks, researchers have designed a number of security models and protection solutions. These are discussed in the next section.

Figure 2.3 Setup of a DDoS Attack [70]

Two features of a DDoS attack are a challenge to the design of successful defences [18]: (i) the attacker can use a spoofed IP source; and (ii) a large number of agent machines are used in the attack-packet headers. The DDoS attack is therefore one of the most harmful of the DoS-attack types.
2.3 Existing Models and Solutions

A number of models have been proposed to protect systems from DoS attack. The strategies used by these models include: focusing on the filtering packets that can potentially cause DoS attack; detecting and preventing DoS attack; identifying a system’s vulnerabilities that can be used to commit a DoS attack; maintaining a system’s resources that might be influenced by a DoS attack; updating the Internet routers to stop a DoS attack; and implementing communication controls in the protocols to prevent a DoS attack.

These models can be classified based on their functionality in terms of protecting a system against DoS attack and also in terms of their positions in the communication between the system and the clients. For each model we will discuss some of the solutions.

2.3.1 Packet Filtering

Data is transferred on the Internet by using small pieces of data containers, called packets [71]. Each packet has two main parts, the header and the data. The header consists of fields that are used by network equipment to move the packets. The source IP address and destination IP address are stored in these fields. The data part holds the data that is to be delivered by the packet. Networks should have a checking mechanism for received packets. Packet filtering refers to the process of allowing or denying the received packets that pass through the packet-filtering points [72, 73] (Figure 2.4).
Both the header and the data are used to make a decision, and the decision is made based on the security rules that are built into the network according to security policies [72, 73]. In the following section, we discuss the firewall as a well-known example solution for the packet-filtering model.

### 2.3.1.1 Firewalls

The firewall is a network device that can be used to protect computers and networks from some kinds of attacks [74]. Firewalls have been widely used in enterprise, small-size and home networks and have been used as the front line of defence to secure networks against unauthorized traffic [75]. Firewalls filter traffic by allowing or denying network connections and packets [75, 76]. Firewalls make decisions based on a set of structured filtering rules, which are written based on the security policy requirements of the secured network [74, 75]. Typically, firewalls rely on restrictions in the network topology to perform
this filtering [77]. Therefore, the network topology and its policies must be well defined; the network administrator should know all the expected in and out traffic to design the firewall’s rules [74].

A firewall can be established as either host based or network based. A host-based firewall is usually included as software in the end host, while a network-based firewall, which is also known as a perimeter firewall, is a special device [74]. To make a firewall effective it should be strategically located to filter all traffic between the network and the outside world. Firewalls are therefore traditionally placed at the points that are connected to the outside network (that is, the network’s ingress and egress locations). These are called choke points [77]. Firewalls are designed to filter each packet separately. Depending on the information contained in the packet, the firewall will determine whether to allow or reject the packet [77].

Firewalls use three methodologies, depending on the level (packet, session or application), to perform filtering. The simplest form of firewall performs packet filtering, which is based on IP addresses and port numbers. More complex firewalls are called circuit gateways and run on TCP sessions, where the firewall re-assembles and examines each packet in every TCP circuit. This type of firewall is more expensive to build than those that use packet filtering, due to their added functionality. The final methodology involves application relays and is the most complicated form of firewall. An application relay runs as a proxy for some applications or services (such as for mail, telnet or the
Internet). It provides extra functions compared to the circuit gateway (such as stripping macros from some received files (for example, a Microsoft Word document)) [78].

Firewalling is often united with other technologies, and many of the technologies that are often associated with firewalls are actually part of these other technologies. A number of the filtering features in firewalls are used to enforce an organization’s policies that are not directly related to security. Some firewall filtering is enforced by other security countermeasures to prevent attacks that are detected by these countermeasures [73].

A firewall is usually the main solution to packet filtering. In the following section we provide an overview of some of the most popular types of firewall-filtering technologies and the common usage for these techniques.

**Packet Filtering: Stateless-Inspection Firewalls**

The most basic type of firewall is packet filtering that provides an access-control function for hosts and communication sessions. This kind of firewall is known as a stateless-inspection firewall. While packet filtering is an essential feature of most modern firewalls, almost all firewalls sold today have additional functionalities.

Packet filtering tests the packet’s header within its rule set and access-control lists; as such, it is not concerned with the content of each packet. The packet’s source IP address, destination address, transport protocol, session source and destination ports, and the packet direction (inbound or outbound) are the most
important attributes used in filtering. Stateless-packet filters are vulnerable to exploitation and attacks that take advantage of weaknesses of the TCP/IP specification and protocol stack. For instance, spoofed packets cannot be detected by this kind of filter. However, higher-layer firewalls can prevent some spoofing attacks by authenticating users, or by verifying the established session before passing the traffic [73]. Because the packet-filtering firewall provides a simple method for filtering packets, it has high performance in terms of packet filtering [79].

**Packet Filtering: Stateful Inspection**

Stateful inspection performs packet filtering by examining the state of the connections in the network layer. Stateful inspection only passes packets that are from an expected state. This is accomplished by using an awareness of the transport layer. Stateful inspection filters packets in the network layer by intercepting each packet at that point. The filter then checks whether the packet, according to the firewall’s rules, is allowed to proceed. The stateful inspection tracks and records each connection in a state table for each packet (Table 2.1).

<table>
<thead>
<tr>
<th>Source Address</th>
<th>Source Port</th>
<th>Destination Address</th>
<th>Destination Port</th>
<th>Connection State</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.1.100</td>
<td>1030</td>
<td>192.1.2.71</td>
<td>80</td>
<td>Initiated</td>
</tr>
<tr>
<td>192.168.1.102</td>
<td>1031</td>
<td>10.12.18.74</td>
<td>80</td>
<td>Established</td>
</tr>
<tr>
<td>192.168.1.101</td>
<td>1033</td>
<td>10.66.32.122</td>
<td>25</td>
<td>Established</td>
</tr>
<tr>
<td>192.168.1.106</td>
<td>1035</td>
<td>10.231.32.12</td>
<td>79</td>
<td>Established</td>
</tr>
<tr>
<td>192.168.1.102</td>
<td>1031</td>
<td>10.12.18.74</td>
<td>80</td>
<td>Established</td>
</tr>
</tbody>
</table>
The above table represents all the connection details, including the source IP address, destination IP address, port numbers and the connection-state information. TCP traffic has three major states in stateful inspection: establishment, usage and termination. Stateful inspection checks certain values of the TCP headers to monitor each connection state. The stateful-inspection firewall only passes packets that are identified as being in the expected state. However, this technique will allow any packet that seems to be part of an open connection [73], including spoofed packets that have established connection [73]. This weakness limits the accuracy of the technique’s decision making (whether a decision is correct or incorrect) as it may allow illegitimate packets and block legitimate packets. If a legitimate communication is blocked, then the system has a false-positive incorrect decision. If an attacker’s communication is accepted as legitimate, then the system has a false negative. Decision accuracy is further discussed in Section 2.4.3.

**Application Firewalls**

In application firewalls, the additional capability of stateful-protocol analysis (which is sometimes called deep packet inspection) is added to stateful inspection. Stateful-firewall analysis has basic intrusion-detection technology to improve the standard stateful inspection. This happens because it has an inspection engine that analyzes protocols at a higher network layer (that is, the application layer). This lets the firewall determine network accessibility, based on how applications are running over the network. An application firewall can
check the type of file attached to emails in case they are not permitted to be received by the organization. For example, some organizations do not allow executable files, or instant messaging (IM) over port 80, which is typically reserved for HTTP. In addition, the application firewall can block connections for specific actions (for example, a user performing unexpected commands). However, this kind of firewall is not a complete solution for detecting DoS attacks. Other solutions such as intrusion-detection systems (IDS) and intrusion-detection-prevention systems (IDPS) offer wider attack-detection capabilities. In addition, the application firewall was built as a filtering technique and the limited detection feature was added to support the filtering process. Adding an IDS feature negatively affects its performance. [73]. Application firewalls also have the same limitation in terms of decision accuracy; consequently they are vulnerable to high false-positive and low false-negative alarms [80, 81]. In addition, checking only against a table (such as the state table) can pass checked packets very quickly – a good thing if decisions are correct, but not so good for incorrect decisions. Network-layer firewalls thus have low latency and high throughput. The application-firewall performance is low, because it has the additional function of filtering traffic and communication in the application layer which requires deeper inspection and uses costly operations [82].
**Application-Proxy (AP) Gateways**

The application-proxy (AP) gateway combines upper-layer functionality with lower-layer access control. This feature can be found in advanced firewalls that contain a proxy agent that works as an intermediary between two communicating hosts and denies direct connection between them. For each successful connection, two separate connections are created, one between the true destination and the proxy server, and another between the proxy server and the client. The proxy agent has a direct interface with the firewall rule set so it can make decisions about allowing or denying network traffic. Some proxy agents can perform authentication for each individual network user as a feature of the rule set. AP gateways provide a higher level of security than application firewalls for some applications, because they inspect the content of traffic to identify policy violations and because they avoid direct connections between two hosts. In addition, they can decrypt packets, inspect them and then encrypt them again. However, AP gateways have a major disadvantage. Because they have ‘full packet awareness’ they require more time to read and interpret each packet and consequently require a high bandwidth for communication and processing [73].

**Virtual Private Networking (VPN)**

Firewall devices usually perform at the edge of a network, so they are sometimes required to do additional tasks, such as encrypting and decrypting network traffic. Because of this, VPNs include additional authentication and
filtering mechanisms. For example, by using Internet Protocol Security (IPSec) as part of a VPN solution, spoofing attacks are less likely to occur, because the source IP address and the destination IP address are added to the encryption. Two common VPN architectures are gateway-to-gateway and host-to-gateway. The first architecture is used to connect multiple fixed sites through public lines or networks. The host-to-gateway architecture is used to connect (usually remote) users securely with the network. All remote-access VPNs require the firewall administrator to define the network resources that users can access. This access control is defined for either a user or a group. In a group case, usually another technique (such as RADIUS) is used to perform authentication for users. However, this adds more load to the firewall as a result of the associated encryption and decryption tasks that require more processing time. In addition, many packets may be dropped after they are decrypted, leaving the firewall vulnerable to flooding attacks. To run a VPN mechanism on a firewall requires additional resources, planning and complex controlling [73].

In a firewall situation, end-to-end encryption, an important part of information system security, can be a threat to firewalls, as it prevents firewalls from inspecting packet fields [77]. In addition, firewalls find it difficult to detect spoofed packets if the VPN is not enabled [77, 83]. However, usually attackers hide their identity and location with spoofed IP addresses when they perform DoS attacks to make them harder to detect [77]. Also, because network topologies are complex, it is difficult for administrators to write or modify firewall rules [74]. Locating a firewall at the choke points and then increasing
network connection speeds might make a firewall’s policies more complex. In this case, a firewall may become a bottleneck and limit the quantity of legitimate information which can pass through it [77]. In addition, a firewall does not provide monitoring on networks to detect the possible attacks [84]. Moreover, probing/scanning techniques have advanced to a level that makes it possible for intelligent attackers to obtain a firewall’s policy with a reasonable number of scanner packets [85-87]. Also, firewalls may partially perform some security activities that overlap those of other models and technologies. In these cases, the same tasks may be repeated in other defence lines. All intrusion-defence systems require minimal false positives and false negatives. However, because a firewall blocks a substantial amount of traffic (an open port 80, for example) false-negative DoS attacks can gain access to a network. In addition, because a firewall cannot provide an alert when it has been incorrectly configured, skilled professionals are required to minimize security risks. Nor do firewalls provide notification of attack. To overcome these shortfalls, organizations need other security monitoring tools [88].

2.3.2 DoS-Attack Detection

DoS-attack detection can be defined as the technique of monitoring network and/or system activities in order to detect and log possible DoS attacks, policy violations and malicious activities. DoS-attack detection also includes pre-defined reactions, such as producing reports to a management station (Figure 2.5).
In the following section we discuss the solutions that are based on the attack-detection model.

2.3.2.1 Intrusion-Detection Systems (IDS)

An Intrusion-Detection System (IDS) is used for real-time monitoring on networks to detect a DoS attack before it happens [84]. The IDS performs four important functions: (i) monitoring the target system that requires protection; (ii) collecting data while monitoring; (iii) processing and correlating the gathered information; and (iv) initiating responses if an intrusion is substantiated. [89]. Depending on the source of input for the IDS, it can be classified as a Network-based Intrusion-Detection System (NIDS), Host-based...
Intrusion-Detection System (HIDS) or Hybrid Intrusion-Detection System [89]. A NIDS monitors network traffic to collect input data whereas a HIDS monitors the host to collect input data. A hybrid intrusion-detection system monitors both network traffic and hosts to collect input data. HIDS use two main techniques: anomaly detection and misuse detection.

**Anomaly-Based Intrusion Detection**

The detection of anomalous behaviour relies on the use of a wide range of methods including statistic-analysis methodology, data-mining technology, artificial-neural-network technology and artificial-immune technology. Using complex algorithms, the technique matches current usage patterns (the client-communication behaviour) with a normal (valid) usage pattern to identify any abnormal activities. However, this method is vulnerable when detecting TCP-SYN attacks as it needs to store all information that belongs to half-connections. When an attacker sends a high volume of SYN packets with spoofed sources, this technique stores these data in memory, which is costly and difficult to process [90]. The method is used to detect other types of DoS attacks, such as masquerade and unauthorised-accessing attacks, but again, the system needs to store high amounts of data [91]. The anomaly-intrusion-detection system usually involves a large number of false positives. Moreover, training a system to respond dynamically to new DoS attacks has practical concerns because it is difficult to identify all normal behaviour of legitimate
Typically, detection models are generated off-line due to the enormous amount of archived audit data necessary for the learning process [92, 93].

**Misuse-Intrusion Detection (Signature-Based Detection)**

Misuse-intrusion detection involves TCP/IP analysis, expert-system methods and pattern matching [89]. Whereas the anomaly technique compares client-communication behaviour with recorded normal behaviour, the misuse-detection technique compares current usage with pre-defined (or pre-described or pre-modelled) attacks. The descriptions of the predefined attacks are called signatures. The signatures are compared to a current stream of audit data, so that the modelled attacks (the signatures) can be detected [78, 94-96]. While this process requires large amounts of data storage [91], this type of detection is more efficient than the anomaly technique [97, 98]. It uses predefined signatures and detects attacks without using complex algorithms.

A number of limitations are associated with misuse-intrusion detection. This approach may not be able to detect new forms of DoS attack as it only detects pre-defined attacks. As mentioned, learning typically takes place offline, so an attack may occur before a solution is in place. In addition, signature-based IDS are vulnerable to attack and by injecting the victim’s network with traffic that has been modeled, the network becomes overloaded with a high volume of detection alerts. This form of DoS attack is used to exhaust the network bandwidth and system resources. It also hides the attacking packets within the
high volume of produced alerts [94, 97]. In addition, some kinds of DoS attack are very difficult to detect, such as attacks that use spoofed IP addresses.

2.3.3 DoS-Attack Detection and Prevention

DoS-attack detection and prevention is the technique of detecting and preventing DoS attack by: (i) identifying possible incidents; (ii) logging information about them; (ii) attempting to stop them; and (iv) reporting them to security administrators (Figure 2.6).

In the following sections we discuss the solutions that use this model.

2.3.3.1 Intrusion Detection-and-Prevention Systems (IDPS)

IDPS technologies are differentiated from IDS technologies by one characteristic: IDPS technologies can respond to a detected threat by attempting to prevent it from succeeding. An IDS performs its task by monitoring and gathering data from a target system, hosts and networks. It then analyzes the collected data to evaluate the network status and find evidence for any DoS attack [89, 99]. Rather than being pro-active, it is a passive technique that does not actually counter DoS attack.
Figure 2.6 DoS-Attack Detection and Prevention

While IDS were introduced as a dynamic security solution for encountering attacks [100], IDPS were introduced to activate security systems to block suspicious activities before they take place. Suspicious activities are classified into two main categories: misuse and anomaly [101] (previously discussed). Similarly to the IDS, the IDPS compares actual network traffic with listed normal user behaviour. Applying anomaly detection principle allows the IDPS system to detect new forms of attacks. The IDPS process involves [101]:

i. capturing network-traffic data and normalizing the network flow;
ii. extracting significant features from the network traffic. The collected data is normalized, discredited and managed for errors. The collected data is formalized into a format suitable for the next step;

iii. inducing the rules of the IDPS by using a pattern-recognition method or a learning-process algorithm; and

iv. applying the rules generated by the previous step with the actual network traffic to determine whether the system is experiencing intrusions [101].

The limitations that apply to IDS also apply to IDPS. Standard out-of-the-box configuration may result in normal activities being identified as suspicious and lead to false positives. Because the IDPS may classify some normal behaviours as malicious activities, the problem might reach the point that causes self-denial of service [102]. In addition, it has been reported that 99% of the reported alerts by IDS/IDPS are not related to security aspects [102, 103]. In contrast, malicious activities may be classified as normal, causing a high false-negative rate. Attackers may take advantage of this weakness and trigger an IPDS by sending large volumes of attacks that will be recognised and dismissed by the security operator; however, such attacks may hide an unknown attack that may slip past an unwary security operator.
2.3.3.2 Unified Threat Management (UTM)

UTM refers to the concept of managing all security defence techniques from one point. All firewall-filtering techniques, IDS, IDPS, user authentication, IPSec and gateway antivirus, anti-spam and URL filtering are included in the one security device [104]. UTM also tries to bring DoS filtering, detection and prevention features together in one security defence system. The integration of these features is performed by centralizing the control of these features even though they are provided by different techniques. For example, many firewalls combine multiple features into a single system. It is easier to set and maintain policies on a single system than on many systems deployed at the same location on a network [73].

UTM incorporates different kinds of filtering, detection and prevention techniques. These techniques work as if they are alone, but they integrate with each other to provide a combination of services, such as malware detection and eradication, firewall protection and the sensing and blocking of suspicious network probes [73].

Merging multiple, not-completely-related functions into a single system has both advantages and disadvantages. For instance, deploying a UTM system reduces complexity by producing a single system that can achieve multiple security objectives – if the UTM system has all the features required to meet the security objectives. In addition, this kind of solution needs massive
performance upgrades [73]. A UTM system thus has limitations in performance and may involve tradeoffs in protection decisions [105].

2.3.4 Communication-Queue Management (CQM)

The CQM model is designed to deal with the communication backlog queue, which contains the stages and states of clients (Figure 2.7).

![Communication-Queue Management](image_url)

**Figure 2.7 Communication-Queue Management**

This model checks the queue entries and applies any required maintenance such as deleting, expanding or dividing the queue entries to avoiding flooding attacks.

In the following we discuss solutions built using this model.
2.3.4.1 SYN cache

A SYN cache is a mechanism especially designed to counter TCP-SYN flooding attack. This solution is applied on the server side. In this approach, the server has a global hash table for half-open states instead of saving the half-open states in the backlog queue. This solution thus increases the server’s ability to handle half-open connections (two-way handshake) and consequently the impact of TCP-SYN flooding attacks is reduced. However, this method does not provide a solution for flooding attacks because attackers can increase the volume of unproductive traffic [33]. In addition, when the queue is full, older communication is removed from the queue, thus creating potential for false negatives and false positives [106].

2.3.4.2 Queue-Management Policy (QMP)

QMP is an end-defence approach that manages the backlog queue (the queue that handles half-open connections). This method enables the size of the backlog queue to remain flexible, based on the availability of system resources, and decreases the timeout period. As a result, the number of free slots increases and the system is able to handle further requests. However, due to the expansion limits, this method cannot handle a large-scale DoS attack. A large-scale DoS attack would necessitate the allocation of the whole system’s resources until the system became out of service [34]. In addition, this technique has the potential to create false positives [107].
2.3.5 Re-engineering Internet Routing (RIR)

The Internet routing model involves the installation of detection-and-prevention techniques in all Internet routers. This model aims to facilitate a solution that would counter all DoS attacks around the world for Internet users (Figure 2.8).

In the following we discuss the solutions which are built based on this model.

![Figure 2.8 Re-engineering Internet Routing](image)

2.3.5.1 DoS Network Attack, Recognition and Defence (D-WARD)

D-WARD is designed to detect and prevent incoming and outgoing DDoS attacks while allowing legitimate outgoing traffic to pass through the security gate. Although it is a centralized system, deploying it in a distributed-system
environment could enhance its performance when alert signals are exchanged. D-WARD is installed at the source router that controls the traffic between the source network and the rest of the Internet. D-WARD monitors every packet, sent or received. It has a set of local addresses (all the addresses in the local network) called the police-address set [108].

D-WARD consists of three components: observation, rate-limiting and traffic policing. The observation component observes all incoming and outgoing traffic and keeps statistics on two-way communication between the police-set addresses and the rest of the Internet. Occasionally, statistics are compared with legitimate traffic models to identify potential attacks. The information is then passed to the rate-limiting component. The rate-limiting component is responsible for imposing, modifying or removing the limitation on the sending rate. After generating rate rules, the rate-limiting component sends these rules to the traffic-policing component to apply these rules to network traffic [108].

D-WARD faces a number of challenges. Considering that IPv4 is widely used on the Internet and considering the size of an address (2 to the power of 32), it is nearly impossible to maintain statistics for each communication between the police addresses and foreign hosts. Moreover, the probability of gaining false negatives is high because detecting models use the comparison between incoming and outgoing traffic. On other words, every time D-WARD detects a higher incoming rate compared to the outgoing rate, the system triggers itself
to handle the situation as if there were an attack, which might not be the case [109].

D-WARD only stops DoS attack. Implementing D-WARD universally would thus be impractical, particularly as the use of D-WARD would substantially degrade system performance and place a large overhead on routers. In D-WARD, a router must perform the following activities every second (that is, 86,400 times) of the day: (i) monitoring traffic; (ii) classifying traffic; (iii) measuring statistics; and (iv) comparing statistics. This requires a router with resources that do not exist in many ISP routers. Because there is no benefit for ISPs to implement D-WARD, there is little incentive for them to pay large costs for such a router [17].

2.3.5.2 Traceback

IP traceback means that the ability of identifying the real source of packets that move in the Internet [110]. Probabilistic packet marking (PPM) is a technique designed to detect and filter spoofed IP-address packets in order to defend against DoS attack. If the physical source of the spoofed IP addresses can be identified, combating the DoS attack should be easier [40]. Source identification or IP traceback [111], for spoofed IP addresses would be facilitated if all ISPs implemented certain enhancements to packets. Several solutions, including PPM, have been proposed to solve the IP-traceback difficulty. PPM (first proposed in [111, 112]) allows routers to probabilistically mark packets with partial-path information during packet forwarding. The
victim server can then follow the attack path back to the originating server by analysing a significant number of collected packets [17].

While PPM has advantages over packet marking in terms of effectiveness and ease of implementation, it also has a number of drawbacks. PPM has weaknesses of scalability. It cannot efficiently trace attacks of more than 100 attackers [113, 114]. Also, the attacker may break the traceback by sending packets that have spoofed source IP addresses as well as spoofed marking-field values [115]. In this case, incorrect attack paths will be generated. In addition, these solutions would necessitate changing the nature of ISPs work. This is not really involved of the pairs of communications, which are clients and servers. The PPM technique would also require changes in all routers in the Internet - a costly and difficult exercise. Furthermore, additional processing in each router in the Internet, due to routers needing to probabilistically mark packets, may decrease the performance of the Internet [17]. The attacker can also protect malicious packets by sending packets of maximum size so that routers cannot add any bits to the packet to mark it. Finally, PPM has a high potential for false positives and a lower potential for false negatives [31, 116-119].

2.3.6 Vulnerability Analysis

Vulnerability analysis is the mechanism of understanding network assets, their connections and system policies, and then analysing them to reveal possible vulnerabilities in the network. These vulnerabilities are considered in light of the potential DoS attacks the system might receive (Figure 2.9).
In the following section we discuss the security solutions based on this model.

Figure 2.9 Vulnerability Analysis

2.3.6.1 Topological Vulnerability Analysis (TVA)

Topological Vulnerability Analysis (TVA) or an Attack Graph reveals potential paths of vulnerability that allow attackers to break through a network. It discovers critical vulnerabilities and then provides protection suggestions for critical network assets. This approach allows the protection of networks before DoS attacks occur. TVA detects the network configuration (including software), its vulnerabilities, and the network’s and software’s relationship to vulnerable services. It then simulates multi-step attack penetration by matching the network configuration with a database of modelled attacker exploits. In the simulation, the attack graph can be forced according to user-defined attack
scenarios. The resulting attack graph generates recommendations for computers and devices to achieve optimal network hardening. It also has the ability to provide sophisticated visualization for interactive DoS attack-graph exploration (Figure 2.10) [120-122]. In TVA, data from network scans and known vulnerabilities are combined into a model of the network security environment. A multi-step attack graph for this environment provides a context for overall network security [120].

![Figure 2.10 Topological Vulnerability Analysis (TVA) [120]](image)

Like all security techniques, TVA has advantages and disadvantages. Existing scanners are helpful in scanning vulnerabilities in single or multiple hosts in a target network and in terms of finding system vulnerabilities. On the other hand, these tools check security holes only in order to analyse vulnerabilities in a network. In addition, TVA has a potential for false alarms, because it is difficult to have a 100% accurate definition for system vulnerabilities which
might be used by attackers. The system may accept attack-communication
paths as normal and reject legitimate paths as malicious. Because TVA is
sufficiently flexible to address a full variety of network configuration and
vulnerability types, it can be used to detect spoofed packets and DoS attacks
[123]. However, it is not a feasible defence system for finding and preventing
DoS attack because it is not a true real-time system. In addition, TVA depends
on many different techniques; failure in any one of these will affect the
working of the TVA in protecting the system against DoS attack.

2.3.6.2 Network Access Control (NAC) Firewalls

NAC is the mechanism of checking clients’ or users’ machines before they
connect to the system. This checking focuses on the vulnerabilities of these
machines in order to minimize weaknesses that could be exploited by DoS
attacks while client machines are connected to a network through a firewall.
This health check includes security software updates, configuration settings for
security controls, previous malware scans, and operating system and selected
application patch levels. To perform this health check, additional software that
is controlled by the firewall is required on the user’s system. The ability of the
client to access network or Internet resources might be temporarily restricted as
a result of the check. [73]. This restriction is helpful in protecting the system
from DoS attack that might be caused by client, rather than server,
vulnerability.
2.3.7 The Communication-Controlling Model (CCM)

The CCM focuses on the communication protocol. The protocol is designed in a way that prevents DoS attack. In addition, the system never starts the communication with the client from the first request. Instead, the system gives the client a piece of data, which is known as a communication control. This data should be provided with the next communication (Figure 2.11).

![Figure 2.11 The Communication-Controlling Model](image)

In the following sections we discuss solutions based on this model.

2.3.7.1 SYN Cookies

A cookie is a piece of data that is sent to the client and which is to be provided in the client’s next communication [124]. This technique was originally suggested by Bernstein [125, 126] to protect systems against SYN attacks in the TCP three-way handshake. The cookie has a time limit to detect expired communications and is used to protect the TCP backlog queue [56, 124].
first use of this technique was in the key-exchange protocol proposed by Karn and Simpson [65]. In this protocol, the cookie technique is used to protect a system against flooding attack performed with bogus IP addresses. The protocol involves the parties in the communication passing cookies to each other. The first step of the communication starts when the initiator (client) requests a cookie from the responder. The initiator cannot perform any further steps without providing this cookie. This mechanism, using the cookie technique, has been used in different proposed protocols such as ISAKMP [127], Oakley [128], IKE [66], IKEv2, [129], Just-Fast Keying (JFK) and Just-Fast Keying initiator variant (JFKi) [130].

The JFKi authentication protocol [130] is a key-agreement protocol that is designed to protect systems against DoS attack in a potentially hostile environment such as the Internet. It uses the SYNCookie technique, which involves a hashed cryptographic function being given to the client when the backlog queue is full. Although the SYNCookie approach protects a system against flooding attacks that lead to TCP-SYN attacks, it is basically only an authentication scheme that is able to prevent the system from receiving spoofed source IP-address packets. However, an attacker who controls a large number of zombies can flood the system by sending a large number of SYN packets without spoofing their sources, and then by sending ACK packets based on the received SYN-ACK packets. In this case the system will be subject to TCP-SYN attack [37, 131].
2.3.7.2 The Client-Puzzle Technique

The idea of a client puzzle was proposed by Dwork and Noar [132] as a computational method for combating junk mail. This concept has also been used by July and Brainard [68] as a protection technique against DoS attack caused by connection depletion. In this approach, the system accepts communication normally, but if the system is under attack the client is required to provide a solution to a mathematical puzzle that is posed by the system in order for the communication to continue. The client-puzzle approach places some burden for communication back on the client and also creates a delay before communication continues. The delay allows system resources to be conserved, and if required elsewhere, temporarily redirected.

Many schemes have been suggested for the client puzzle and they can be classified, based on the impact on the client machine’s resources, into two categories [133]: (i) charging the CPU cycles (Aura et al. 2000 [134]; Juels and Brainard [68]; Wang and Reiter [135]; Waters et al. [136]; Feng [137]; Parno et al. [138]; Dean and Stubblefield [139]); and (ii) charging the memory cycles (Abadi et al. [140]). In addition, a number of protocols have been designed based on the client-puzzle concept, such as the client-aided RSA protocol [141], the Lee and Fung protocol [142] and the Host-Identity Protocol (HIP) [143].

The HIP is a protocol that is used to establish security association. It uses the client-puzzle technique to avoid DoS attacks on authentication protocols. The
HIP technique requires substantial unproductive overhead computation processing from clients in order to attain high-level security. This is not desirable for some applications, especially when the client uses a mobile device in the communication. In addition, the HIP technique itself can be a target for DoS attack, because it performs cryptography operations in puzzle creation and verification. The attacker can make use of this technique and send a high volume of packets that have puzzles of their own to the system in order to occupy the system with verifying these puzzles. In addition, accessing a website that uses the client-puzzle technique can decrease the quality of the service, because the client must wait until his or her computer solves the puzzle. A service that poses a difficult puzzle which takes time to solve may cause a frustrated legitimate client to leave the system [37]. The implementation of the client-puzzle approach also requires special client-side software [68]. This problem can be solved by building the puzzle software into the client’s browser [37]. However, because the client’s permission is required to accept the software’s execution, an attacker might hack an inexpert client’s machine and install malicious code.

### 2.3.8 Summary of the Existing Models

In this section, the advantages and disadvantages of the existing models in protecting systems are discussed. Table 2.2 summaries the strengths and limitations of each model described above.
To start with, we can see that the packet-filtering techniques operate in real time. This ability is one of the most important features in terms of protecting systems against DoS attack. However, packet-filtering decisions are built based on a single received packet. While the packet might not describe the attack itself, it might be part of an attack which consists of several packets. This characteristic makes this model vulnerable to spoofed and modified packets, a common cause of DoS attack.

The DoS-attack-detection model solves this problem by storing information about the received packets for a period and then trying to detect DoS attacks. However, this model does not have any technique to filter detected attacks; it only notifies the network administrator about the attack. This is a major limitation because any subsequent filtering of the attack will only be after part or all of the attack is committed.

The attack detection-and-prevention model solves the DoS-attack-detection model limitations by providing an attack-prevention solution. This type of model must usually be integrated with other models to perform packet filtering. This, however, is insufficient, because it is difficult to monitor the results of the prevention, and the response time could be low. It also has the same detection limitations as the DoS-attack-detection model. In addition, because it is not a filtering solution, not all detected suspicious activities are halted.
<table>
<thead>
<tr>
<th>Model Type</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
</table>
| Packet Filtering                   | • Provides packet filtering  
|                                   | • Operates in real-time                                                     | • Bases decision on a single packet  
|                                   |                                                                           | • Cannot detect spoofed packets                                                  |
| Attack Detection                   | • Bases its decision on a group of packets  
|                                   | • Provides attack detection                                                 | • Does not provide packet filtering  
|                                   |                                                                           | • Makes decisions after the attack                                              |
| Attack Detection and Prevention    | • Provides attack detection.  
|                                   | • Attempts to protect the system from detected attacks                       | • Usually needs to integrate with other models to perform prevention  
|                                   |                                                                           | • Makes detection and prevention decisions after the attack                     |
|                                   |                                                                           | • Doesn’t stop all detected suspicious activities                              |
| Topological Vulnerabilities Analysis| • Finds system vulnerabilities                                              | • New threats could use undiscovered vulnerabilities  
|                                   |                                                                           | • The repairs of these vulnerabilities are usually done using techniques not within the model |
| Communication Queue Management     | • Provides a solution for DoS attacks that affect the backlog queue          | • Only solves DoS attacks that affect the backlog queue  
|                                   |                                                                           | • May replace real requests with DoS requests                                 |
| Re-engineering Internet Routing    | • Provides a solution for DoS attacks around the world                       | • Would be difficult to achieve and is still at a theoretical stage             |
| Communication Controlling          | • Enhances system efficiency                                                 | • Does not provide attack detection and prevention                             |
The TVA model scans the system’s resources in order to find system vulnerabilities that might be used by attackers to commit DoS attack. However, new threats could use undiscovered vulnerabilities to commit DoS attack. In addition, an attacker can still perform DoS attacks without taking advantage of a system’s vulnerabilities, such as performing flooding-request DoS attacks. Also vulnerability repairs are usually done using techniques that are not within the model’s control.

The CQM model can protect the system against specific types of DoS attack, such as TCP-SYN attacks. However, it maintains the system log queue without knowing exactly the nature of the entry, so a real request might be replaced with a DoS request.

The CC model tries to avoid this replacement problem by ensuring that only real clients take a place in the queue. It protects the system against a high number of DoS attacks by improving the system’s ability to process DoS-attack packets efficiently. However, it does not provide any kind of attack detection and prevention. Under DDoS attack, therefore, this feature might not be helpful.

The RIR model tries to solve the DoS-attack problem permanently by making changes in all Internet routers. This is helpful as it negates the Internet limitations which enable DoS attacks. However, this solution appears unachievable due to its expense and because it only covers current limitations.
From the above discussion, the existing-solutions overview and the DoS-attack background (Section 2.2) we can define the key requirements for designing a solution that can be used to protect CGeSS against DoS attack. The key requirements of a security solution also serve as assessment parameters to facilitate the evaluation of existing solutions.

2.4 Key Requirements (Assessment Parameters)

In order to design an efficient, comprehensive and well-built solution to protect CGeSS from DoS attack, solution designers require a set of assessment parameters that can be used to build, analyze and evaluate solutions. Based on the DoS-attack problem overview (Section 2.2) and the existing models and solutions background (Section 2.3) the DoS-attack defence systems can be analysed and evaluated from at least four characteristics: functionality, security, accuracy and performance (Figure 3.12). These criteria are discussed in the following sections.

2.4.1 Functionality

Functionality refers to the nature of the solution’s components and the modality of the solution in achieving its goals. Any proposed solution should have high functionality in order to protect CGeSS from DoS attack. The functionality-assessment parameter is used to measure the comprehensiveness of the solution’s activities in protecting systems against DoS attack. It can be classified into six aspects as follows.
Figure 2.12 Key Requirements (Assessment Parameters)
i. **Packet filtering:** The first aspect of the functionality parameter determines whether the solution provides packet filtering. Packet filtering refers to the process of testing packets directed to the system. Using packet filtering, the solution should be able to reject unwanted packets.

ii. **Attack detection:** The second aspect determines whether the solution provides attack detection. Attack detection refers to the process of monitoring system traffic in order to find security breaches. An attack should be detected through the analysis of one or more packets. The system should maintain a log of all communications over a period of time, and use this log to help detect DoS attack.

iii. **Attack prevention:** The third aspect determines whether the solution provides attack prevention. The system should be able to prevent any detected DoS attack by designing prevention strategies. This can be achieved by having full control over the ability to update filtering rules. However, the prevention decision may not be the same for all similar types of attacks, as the attacks may happen at different system states.

iv. **Components integration:** The fourth aspect determines whether the solution integrates the previous three aspects. Attack filtering, detection and prevention should work as one component and have only one source for a communications log. This helps to avoid conflict in the system’s decisions.
v. **Independence from the network topology:** The fifth aspect determines whether the solution is independent of the network topology. The degree of network topology independence depends on the level of prior knowledge of the network topology. When a solution requires a low prior degree of knowledge, greater independence can be achieved. When a solution requires a high prior knowledge, the solution is more difficult to manage, because any change in the network topology would be followed by a change or more in the solution’s configuration. A more complex solution is likely to have weaknesses caused by mis-configurations or wrong configurations, which again render it vulnerable to DoS attack.

vi. **Real-time processing:** The last aspect of the functionality parameter determines whether the solution offers real-time processing. The solution should utilise a real-time technique so it can instantaneously prevent DoS attack.

### 2.4.2 Security

Two parameters can be used to evaluate a solution’s security features.

v. **Spoofed-attack detection:** The first aspect of the security parameter determines whether the system has the ability to detect spoofed communications. This kind of attack happens when the attacker changes identity when committing the DoS attack. Spoofing adds to the
complexity of attacks. A solution should be designed to detect and prevent spoofed DoS attacks.

vi. **Repeat-Resending-communication detection**: The second parameter determines whether the system has the ability to detect repeat-resending communications. This kind of attack arises when the attacker sends a communication to the system which is accepted. The attacker then repeats this communication to keep the system busy with processing these repeated communications.

### 2.4.3 Decision Accuracy

The solution’s decisions should be accurate in determining acceptable and non-acceptable requests. Two parameters can be used to evaluate a solution’s decision accuracy.

vii. **False positives**: The first parameter relates to the production of false-positive decisions. A false-positive decision arises when a system flags a non-malicious activity as malicious. A solution should minimize the production of false-positive decisions.

viii. **False negatives**: The second parameter relates to the production of false-negative decisions. If a system flags a malicious activity as non-malicious, this is a false-negative decision. The solution should minimize the number of false negatives.
2.4.4 Performance

The performance of any solution designed to protect systems against DoS attack is an important criteria. Two parameters can be used to evaluate a solution’s performance.

i. **Operational efficiency:** The first aspect of the performance parameter determines whether the system minimizes overhead costs. Operational efficiency can be measured by computation cost. The computation cost refers to the overhead processing of the additional operations that the proposed system requires. The goal of the proposed solution is to maintain the availability of the CGeSS. If a solution incurs a high level of additional communication costs, the system can become a target for DoS attack [17].

ii. **System scalability:** The second aspect of the performance parameter refers to the storage the solution requires to manage communications. The greater the size of the storage required, the greater the chance for DoS attacks to occur. In addition, if a solution needs to store information relating to a connection, this will limit the number of connections (that is, the number of clients) that the system can handle at any one time.

These four key requirements – functionality, security, decision making and performance – can be used as assessment parameters to evaluate existing
solutions. Using these parameters will help us determine whether any of the previously described solutions can be used to protect CGeSS against DoS attack. The following section assesses existing solutions using the key requirements as assessment parameters.

2.4.5 Evaluating Existing Solutions

In this section we evaluate the existing solutions by using the key requirements as assessment parameters. Table 2.3 summarizes the results of our evaluation.

Table 2.3 shows that the packet-filtering firewall provides high performance when filtering packets because it is a stateless technique. However, it cannot detect spoofed and resending packets, because its filtering technique is based on the received packet header only. While the stateful technique provides a solution for this by holding the received packet information in order to protect the system from repeated attacks, this opens a vulnerability in system scalability. In addition, neither of these techniques is capable of detecting a DoS attack because they do not perform attack detection.

An IDS provides a detection feature but it cannot actually protect a system from a detected attack. An IDPS provides this additional prevention feature. However, the prevention functionality is limited because IDPS use filtering techniques from other models. UTM was proposed to overcome these limitations. While UTM is a promising technique, it has limitations in its performance. This limitation is not acceptable, because the efficiency of packet
processing is an important factor in protecting a system against DoS attack. In addition, UTM inherits the limitations of the integrated techniques, including their low levels of decision accuracy.

The SYN cache and QMP provide solutions for TCP-SYN attack which is helpful, but this is just one attack of a high number of DoS attacks. The attack graph provides a solution for DoS attacks by finding system vulnerabilities that might be used by attackers as a focus for DoS attacks. However, many attacks are committed because of limitations related to the Internet, rather than system design. In addition, an attack graph does not provide DoS-attack filtering, detection and prevention.

While the JFKi protocol protects systems from memory DoS attacks perfectly, the HIP has an additional advantage in being able to delay client messages. This feature can be used to protect a system from CPU attack. However, the client is required, in this solution, to perform unproductive complex mathematical operations in order to connect to the system. In addition, both JFKi protocol and the HIP use cryptographic hashes in their solutions to protect communication controls from modification. This operation does not require a high level of processing from the system, but if an attacker sends a high volume of DDoS attack, the system might be subject to CPU-DoS attack. Nor do these solutions provide any type of attack detection.
<table>
<thead>
<tr>
<th>Techniques</th>
<th>Functionality</th>
<th>Security Detection</th>
<th>Decision Accuracy</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Packet Filtering</td>
<td>Attack Detection</td>
<td>Attack Prevention</td>
<td>Real-time Processing</td>
</tr>
<tr>
<td>Firewalls</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Packet Filtering (Stateless)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Stateful Inspection</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Application Firewalls</td>
<td>Yes/Low</td>
<td>Yes/Low</td>
<td>Yes</td>
<td>High</td>
</tr>
<tr>
<td>Application-Proxy Gateways</td>
<td>Yes/Low</td>
<td>Yes/Low</td>
<td>Yes</td>
<td>High</td>
</tr>
<tr>
<td>Virtual Private Networking</td>
<td>Yes/Low</td>
<td>No</td>
<td>Yes</td>
<td>High</td>
</tr>
<tr>
<td>IDS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Misuse</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Anomaly</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>difficult</td>
</tr>
<tr>
<td>IDPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Misuse</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Anomaly</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>difficult</td>
</tr>
<tr>
<td>Unified Threat Management (UTM)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>D-WARD</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Traceback</td>
<td>Yes/Low</td>
<td>Yes/Low</td>
<td>Yes</td>
<td>Low</td>
</tr>
<tr>
<td>SYN Cache</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Queue Management Policy (QMP)</td>
<td>No</td>
<td>No</td>
<td>Med</td>
<td>Yes</td>
</tr>
<tr>
<td>Network Access Control (NAC)</td>
<td>No</td>
<td>Low</td>
<td>Med</td>
<td>Yes</td>
</tr>
<tr>
<td>Attack Graph</td>
<td>No</td>
<td>Low</td>
<td>Med</td>
<td>High</td>
</tr>
<tr>
<td>JFKi</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>HIP</td>
<td>Yes</td>
<td>Low</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

* In this technique, the filter position affects the decision accuracy. If a packet filter is applied first (that is, before other filtering techniques), then the false positives will be high. If a packet filter is applied last (that is, after other filtering techniques), then the false negatives will be high.
The D-WARD and Traceback solutions propose universal solutions for DoS attack. However, these solutions would require changes in every router in the Internet, which is impracticable. In addition, they have limitations in both performance and decision accuracy.

Based on this evaluation, we conclude that none of the existing solutions satisfies the key requirements. A new solution to satisfy these key requirements is required so that CGeSS can be safeguarded against DoS attack. This solution needs to provide fully integrated attack filtering, detection and prevention within the one security model.

2.5 Conclusion

This chapter provided an overview of the DoS-attack types. We have seen that the flooding attack is the most dangerous type. We reviewed common attacks of this type and described how the attacker can increase the damage caused by this attack by performing a Distributed Denial-of-Service attack. This review helped us to understand the DoS-attack problem.

We also reviewed the existing protection models and discussed their advantages and disadvantages. For each model we presented examples of solutions built on these models and evaluated the strengths and limitations of each solution.

Using this background we developed key requirements for DoS-protection solutions. These key requirements can be used by solution designers to identify
the characteristics and features required for building new and effective protection solutions. However, we used these key requirements as assessment parameters to evaluate the existing reviewed solutions. We found that none of the proposed solutions satisfies all these key requirements. Some of the solutions’ limitations are caused by weaknesses in the models on which the solutions are built. Therefore, a new security model, as well as a comprehensive security solution, is needed to satisfy all the key requirements and so protect CGeSS from DoS attack. We present these security initiatives in Chapter Three.
Chapter 3

A Security Model and Approach for Protecting Critical Government eService Systems (CGeSS) from DoS Attack

3.1 Introduction

In the previous chapter (Chapter 2), we discussed the advantages and disadvantages of the existing security models that are designed to protect systems from DoS attack. Although the attack-filtering model can filter the DoS-attack packets based on information within the packet itself and by using a set of rules, it cannot recognize the type of DoS attack. This limits its decision accuracy, because the DoS-attack packets that form part of the unrecognized attacks might be accepted for processing in the system. On the other hand, while the attack-detection model can detect DoS attacks, it does not have a filtering or prevention component to protect the system from detected attacks. The attack detection-and-prevention model has an additional feature that filters the DoS-attack packets. However, this function is performed by
filtering components that are outside the model’s behaviour. This limitation affects solutions that are built using the attack detection-and-prevention model.

In the previous chapter (Chapter 2), we also reviewed the weaknesses of the existing solutions. These solutions are built on a number of models. The packet filtering of firewalls provides an efficient packet-filtering method, but it cannot detect and prevent spoofed DoS-attack packets. While the stateful technique provides a solution for detecting spoofed packets, it requires an additional process (a communication-list search) and it may be susceptible to SYN attacks. The SynCookie solution offers a solution for SYN attack and key-establishment attack but does not offer a solution for controlling the timing of clients’ messages. While the client-puzzle technique solves this problem by controlling the time period of the communication between the client and the system, it requires superfluous overhead processing at the client’s machine. The SynCookie and the client-puzzle solutions are only designed to protect a system against SYN attacks and key-establishment attacks; they do not guard against the other DoS attacks.

In addition, all security solutions based on the existing models, except the CCM and the RIR model, require an additional configuration that is based on the network topology. Any change in the network topology usually leads to a configuration overhead in these solutions. This increases the possibility of human error and the opportunity for DoS attack. However, solutions that are built based on the CCM do not have a method for identifying the type of DoS
attack. Instead, the CCM is designed to protect systems against memory and key-establishment attacks; it also has the ability to control the timing of client messaging [68].

The background discussion closed by defining key requirements that need to be satisfied before defence solutions can guarantee protection against DoS attack. Not one of the existing solutions satisfies these key requirements.

To respond to the limitations of existing models, the current chapter proposes a security model that can be used to protect CGeSS from DoS attack, and uses this model to design a security solution. The proposed model should be able to perform DoS-attack filtering, detection and prevention. It should also control the communications between the client and the system, and not require a reconfiguration based on the network topology. In order to reach the goal of building this model we need to (Figure 3.1):

i. understand typical process flows in government eService systems, and compare them to identify the critical services that require protection from DoS attack;

ii. design a formal model that analyzes CGeSS to (a) identify the system’s actors; (b) understand the relationship between these actors; (c) determine the required communications and transactions; (d) understand the nature of these communications; and (e) verify the important security properties that must be satisfied in order to protect CGeSS against DoS attack;
iii. design a security model that can be used as a base for building DoS-attack defence systems. This model should comply with the requirements of the formal model and cover the gaps of the existing DoS models; and

iv. develop a security approach based on the proposed security model. This approach will provide the defence system to satisfy the model goals.

---

**Figure 3.1 The Process of Designing a Solution to Protect CGeSS from DoS Attack**

The remainder of this chapter is organized as follows. In the next section (Section 3.2) we provide a background to government systems. We review the nature of the services that the systems provide to clients and the communications that execute these services. These requirements are used to identify the critical services that require protection. With the services identified, we then present a formal model to characterize and analyze these services (Section 3.3). This model gives us descriptions of the system’s elements, the relationship between these elements, the system’s processes and the system’s security requirements. Based on the formal model components and requirements, a security model that we call DoS Prevent, Identify and Filter (DoS-PIF) is then proposed in Section 3.4. Finally, based on the
proposed security model and in response to the model’s goals, a security approach that we called the Holistic Approach for Securing and Protecting Critical Government eService Systems (HASP-CGeSS) is proposed (Section 3.5).

### 3.2 Background to eServices in Government Systems

In order to design a security solution that protects a system from DoS attack, we need to recognize the business flows of services that are provided by the system. With this knowledge we can identify the services that have high priority in terms of ensuring their availability and providing protection from disconnection. This also helps us to investigate the communication vulnerabilities that might be susceptible to a DoS attack.

In order to examine the vulnerability and the required level of availability for services we use four parameters: the security of the exchanged information; the required level of availability; the required authentication; and the required system-processing cost.

i. **The security of exchanged information**: This parameter describes the sensitivity level of the information that will be sent via public networks in order to complete the service. This includes both the information
provided by the client to the system and the information provided by the system to the client.

ii. **The required level of availability**: This parameter reflects the access needs of the clients to the service and the consequent degree of system availability that needs to be ensured. There are three levels of availability: high, medium or low. If a service requires a high level of availability, any disconnection or inaccessibility of the service is unacceptable.

iii. **The required authentication**: This parameter refers to the authentication technique required to access the service. The authentication technique enables the client to prove his or her identity and access the requested service.

iv. **The required system processing cost**: This parameter refers to the processing cost that the system incurs in order to perform the required service.

In addition to the above parameters, we also consider the type of online government service. These can be typified into four categories based on the relationship between the client and the system as follows: (i) services for browsing users; (ii) services for anonymous users; (iii) services for partially authenticated users; and (iv) services for fully authenticated users.

In terms of the four parameters, the four types of government online services can be characterized as follows (Figure 3.2).
Figure 3.2 Communication for User Services: (a) Services for Browsing Users (b) Services for Anonymous Users (c) Services for Partially Authenticated Users (d) Services for Fully Authenticated Users

i. **Services for browsing by users:** In this type of service, the user visits the government web site (for example) to read news, download e-forms, review governments rules and regulations or to find a job (Figure 3.2 (a)). Communication starts when the client sends a request to gain information. Once the system receives this request, it transfers this request to the web server that has the required information. The web server will then send the required information to the client. The
browsing client does not require any kind of authentication. This type of service requires minimal system processing. It also has a low availability requirement.

ii. **Services for anonymous users:** In this type of service, the user needs to retrieve personal information from the system. The user is required to provide selected information (such as date of birth, or other data stored in a smart card) to provide proof of identity to the system (Figure 3.2 (b)). The communication starts when the client sends a request to retrieve personal information from the system. Once the system receives this request, the system transfers this request to the replicated database server that holds the required information. To perform the client’s request, the client’s identity needs to be verified. This verification is performed by the replicated database server. When the verification is successful, the replicated database sends the required information. The security of the required information is low and the availability requirement of this information is also low. This type of service incurs minimal processing costs because the main server transfers requests for this type of service to the replicated databases. The verification of client identity, the processing of the request and the supplying of the required information are all performed in the replicated database.
iii. **Services for partially authenticated users:** This type of service assumes that users need to retrieve secret personal information from the system in order to follow up previous transactions or to reserve appointments. In this type of communication, the client is required to be authenticated to his or her smartcard before retrieving the required information (Figure 3.2 (c)). Communication of this type begins when the client sends a request. The system transfers the request to the replicated database server that holds the required information. The database server then asks the client to authenticate to the client’s smartcard. The client inserts the smartcard into the smartcard reader and enters a secret PIN. By doing so, the client confirms possession and ownership of the smartcard. The result of the authentication is sent to the server, and, if the authentication is positive, the server then sends the requested information. The security of the required information is medium and the availability requirement of this information is low. This type of service incurs minimal processing costs. Authentication is performed using the smartcard and the replicated database only receives the result of the authentication.

iv. **Services for fully authenticated users:** In this type of service, the user performs important mandatory services such as ordering a passport, updating data, adding a dependent child or spouse, registering a marriage, paying fees or uploading signed forms or documents. Delays and interruptions to these important services can cause substantial
inconvenience. When a client requests this type of service, the system will transfer this request to the authentication server (Figure 3.2 (d)). At this stage, the client is required to provide proof of identity to the system by providing a signature. This requires the client to authenticate to his or her smartcard, which stores the signature of the client. The client will encrypt his or her signature using the server public key, and then send it to the system. The system will decrypt and authenticate this signature. Once the client is authenticated, the system will open a secure encrypted channel so that the client can perform the desired transactions with the system.

The security level of the information used in these services is high, and the availability requirement of these services also high. Because authentication to this type of service is performed based on the Public Key Infrastructure PKI, it requires high-level processing within the system’s resources. The traffic between the client and the system must also be encrypted and decrypted. Consequently, this kind of service requires expensive processing operations from the system.

### 3.2.1 Summary

In this section we discussed background information for services that online government systems need to provide. The goal of this discussion was to identify, based on the four parameters, the services that require protection. The results of the discussion are tabulated in Table 3.1.
As can be seen in the table, a service involving fully authenticated users is the most critical type of service, because it is the most costly service (in terms of processing) and requires a high degree of availability. Compared to the other types of services, protection for this type of service against DoS attack takes priority. This thesis will thus focus on CGeSS and protecting these systems against potential DoS attack.

To design a security solution to protect this type of system against DoS attack, we need to analyze this type of service system in order to understand the nature
of its communications, transactions and security requirements. This formal analysis is undertaken in the next section.

3.3 A Formal Model for Analyzing CGeSS

In the previous section (Section 3.2) we found that, among online government systems, eServices that require fully authenticated users have the potential to suffer the greatest damage as a result of service disruption. This type of service must be protected from disconnection caused by DoS attack. To provide this protection we need to characterize this type of service. In this section, we will provide a formal model for analyzing critical government systems. This model will help us to identify the system’s elements, processes and requirements. Table 3.2 summarizes the notations that are used in the description of the formal model.

A CGeSS can be modelled as the following:

**Definition 3.1: CGeSS**

A CGeSS $G$ can be defined by the following sets:

$$G = EP \cup CE \cup ST \cup \{SP\} \cup RSec$$  \hspace{1cm} (3.1)

where

- $EP$ represents the set of communication-engaging parties in $G$ and $EP \neq \phi$;
• $CE \neq \phi$ and $CE$ represents the communication environment, which consists of the government system servers, networks and client devices;

• $ST$, where $ST \neq \phi$, represents the service transaction between $EP$ elements using $CE$;

• $SP$ represents the system performance; and

• $RSec$ represents the required security for the system.

<table>
<thead>
<tr>
<th>Table 3.2 Formal Model Notations</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$G$</td>
<td>CGeSS</td>
</tr>
<tr>
<td>$EP$</td>
<td>Communication-engaging parties</td>
</tr>
<tr>
<td>$CE$</td>
<td>Communication environment</td>
</tr>
<tr>
<td>$ST$</td>
<td>Service transaction between $EP$ using $CE$</td>
</tr>
<tr>
<td>$SP$</td>
<td>System performance</td>
</tr>
<tr>
<td>$RSec$</td>
<td>Required security for the system</td>
</tr>
<tr>
<td>$S$</td>
<td>Government system that provides the services</td>
</tr>
<tr>
<td>$C$</td>
<td>Client who wants to obtain a service from $S$</td>
</tr>
<tr>
<td>$IC$</td>
<td>Identification of the client $C$ that is given by the system $S$</td>
</tr>
<tr>
<td>$IS$</td>
<td>Identification $S$ that is given by the Certificate Authority CA</td>
</tr>
<tr>
<td>$D$</td>
<td>Device that a client $C$ uses to perform the transaction $ST$</td>
</tr>
<tr>
<td>$N$</td>
<td>Network that is used to connect $C$ with $S$</td>
</tr>
<tr>
<td>$TY$</td>
<td>Service type that a client $C$ needs to perform</td>
</tr>
<tr>
<td>$SO$</td>
<td>Service-ordering action</td>
</tr>
<tr>
<td>$RIV$</td>
<td>Request for identity proof - verification action</td>
</tr>
<tr>
<td>$PIV$</td>
<td>Provision of the required identity - verification action</td>
</tr>
<tr>
<td>$RP1$</td>
<td>Request for action to provide required information to the service</td>
</tr>
<tr>
<td>$PI$</td>
<td>Action of providing required information regarding the service</td>
</tr>
<tr>
<td>$SR$</td>
<td>Service-processing result</td>
</tr>
<tr>
<td>$CD$</td>
<td>All clients information and events regarding the service transaction</td>
</tr>
<tr>
<td>$SD$</td>
<td>All system information and events regarding the service transaction</td>
</tr>
<tr>
<td>$IV$</td>
<td>Client’s identity verifications</td>
</tr>
<tr>
<td>$IV_{ty}$</td>
<td>Client’s identity verifications required for a service of type $ty$</td>
</tr>
<tr>
<td>$R_f$</td>
<td>All required functions $R_f$ that the system performs to handle any communication coming from the client</td>
</tr>
<tr>
<td>$R_f(a_{cy})$</td>
<td>Required functions to handle a communication $a_{cy}$ that comes from a client and is of type $ty$</td>
</tr>
<tr>
<td>$c$</td>
<td>Cost of the system for each function $r_f \in R_f$</td>
</tr>
<tr>
<td>$R_{sy}(a_{cy})$</td>
<td>System-required processing for $a$ of service type $ty$</td>
</tr>
<tr>
<td>$t$</td>
<td>Unit of time</td>
</tr>
<tr>
<td>$a \propto b$</td>
<td>$a$ and $b$ has a direct relationship</td>
</tr>
</tbody>
</table>
We can see that $EP, CE$ and $ST$ comprise the main players for a transaction within CGeSS $G$. So any transaction that is to gain a service from the system must consist of the engaged parties set $EP$, which performs the government services transaction $T$, using the communication environment $CE$, with an acceptable level of system performance $SP$, while satisfying $RSec$ to enhance the security of the system.

**Engaging Parties**

The CGeSS $G$ consists of a set of engaging parties $EP$ as follows:

$$EP = \{S, C, IC, IS\}$$ \hspace{1cm} (3.2)

where

- $S$ represents the government system that provides the services;
- $C$ represents the client who wants to obtain a service from $S$;
- $IC$ is the identification of the client $C$ that is given by the system $S$. $IC$ is checked by $S$ to ensure $C$’s identity; and
- $IS$ is the identification of the system $S$ that is given by the Certificate Authority CA. $IS$ is checked by $C$ to ensure the identity of $S$.

**Communication Environment**

In the CGeSS $G$, the communication environment $CE$ is defined as the following:
\[ CE = EP \cup \{ D, N \} \] (3.3)

where

- \( D \) represents the device that a client \( C \) uses to perform the transaction \( ST \).
  However, \( D \) could consist of several elements, such as a PC, smartcard reader, fingerprint reader or scanner; and

- \( N \) represents the networks that are used to connect the client \( C \) with the government system \( S \).

**Service Type**

Service type \( TY \) is the type of service that a client \( C \) needs to perform. \( TY \) has four types as follows:

\[ TY = \{ Browsing, Anonymous, Partial, Fully Authenticated \} \] (3.4)

Because our focus in this analytical model is on critical services, we will discuss services only of the fully authenticated type.

**Service Transaction**

In order to identify the service transaction we need to define the following:

- **Service Ordering** \( SO_{ty} \) is an interaction between \( C \) and \( S \), which describes that \( C \) requests to perform a service of type \( ty \) in \( S \). Note that \( SO_{ty} \) does not provide any information or descriptions about the required service except the type of the service, because our focus is on
the communications and their steps. The service type is important because it affects the authentication process.

- **Request Identity Verification** $RIV_{ty}$ is the interaction between $S$ and $C$, which describes that $S$ requests $C$ to provide the identity verifications that are required to achieve a service of type $ty$. So $RIV_{ty}$ is a request that $S$ uses to request client $C$ to verify his or her identity. This verification should be verifiable by $S$ and provide enough evidence to describe the identity of $C$. In addition, this function tells $C$ the verifications that are required for the requested service.

- **Providing Identity Verification** $PIV_{ty}$ is the interaction between $C$ and $S$, which describes that $C$ provides its identity verification to $S$.

- **Request Providing Information** $RPI_{ty}$ is the interaction between $S$ and $C$, which describes that $S$ asks $C$ to provide information or data that is required to achieve the requested service. However, this information is not used for identity verification.

- **Providing Information** $PI_{ty}$ is the interaction between $C$ and $S$, which describes that $C$ provides information or data that is required to achieve the requested service, such as the uploading of document copies or the addition of a dependent’s name or the number of family members. This information is not used for identity verification.
• **Service Result** $SR_{ty}$ is the interaction between $S$ and $C$. It is used by $S$ to provide information and feedback to $C$ about the result of the execution of the requested service, such as ‘the service is achieved’, the receipt, ‘the service is not achieved’ or the reason for this.

• Let $A = \{SO_{ty}, PIV_{ty}, PI_{ty}\}$ and $B = \{RIV_{ty}, RPI_{ty}, SR_{ty}\}$.

• $CD$ is the set that contains all the information that is known to the client $C$ and the events that are performed by the client $C$ regarding the service transaction. This includes items such as the client’s name, address, date of birth, password or smart card insertions.

• $SD$ is the set that contains all the information that is known to system $S$ and the events that are performed by system $S$ regarding the service transaction. This includes items such as identity checks, source checks, date-of-birth checks or password checks.

The service transaction is defined as a set of messages from $A$ and $B$, which are between the engaging parties $EP$, over a set of communication networks $N$, to perform the service. It can be represented as follows:

$$ST = EP \cup A \cup B \cup \{D, N\} \cup CD \cup SD$$

(3.5)

Based on Equation (3.3), the service transaction $ST$ can be represented by a number of interactions that regard the service transactions in the communication environment $CE$ as follows:
In order to complete the transaction, the system requires identity verification from the client. These identity verifications are defined as:

\[ IV = \{ iv \in ST : iv \text{ is an identification that describes the identity of client } C \text{ and can be verified by } S \} \]  

(3.7)

For every type of service there is a requirement to use a group \( IV \) set as follows:

\[ IV_{ty} = \left\{ iv_{ty} \in IV : \text{is an identification that describes the identity of client } C, \text{can be verified by } S, \text{and is required to execute a service of type } ty \right\} \]  

(3.8)

The \( IV \) set can be redefined as follows:

\[ IV = \bigcup_{ty \in TY} IV_{ty} \]  

(3.9)

Each \( a \in A \), which is a communication that is sent by \( C \) to \( S \), is required to be processed in the system \( S \). This processing consists of multi-sequence steps. All required functions \( R_f \) that the system performs to handle all communications that come from the client can be defined as:

\[ R_f = \{ r_f \in ST : r_f \text{ describe a system function that the system performs in completing } ST \} \]  

(3.10)
Based on the above definition, we can define the required functions $R_f(\cdot)$ that the system performs to handle a communication $a_{ty}$, and which comes from a client and is of type $ty$, as:

$$R_f(a_{ty}) = \{ r_f \in R_f: r_f \text{ describes a system function that is performed to complete } a \in A \text{ of type } ty \} \quad (3.11)$$

Based on the above definition, we can define the required functions to manage each of the three members of the $A$ set. The required functions to manage the $SO_{ty}$ can thus be defined as:

$$R_f(SO_{ty}) = \{ checkSource_{c,ty}, checkOrder_{ty} \} \quad (3.12)$$

where

- $R_f(SO_{ty})$ is the required function that the system needs to perform in managing the service order $SO_{ty}$;
- $checkSource_c$ is the function that checks the eligibility of the source $C$ to perform a service order of type $ty$; and
- $checkOrder_{ty}$ is the function that checks the correctness of the order for a service of type $ty$.

The required functions to manage the $PIV_{ty}$ can be defined as:

$$R_f(PIV_{ty}) = \{ checkSource_{c,ty}, checkIdentity_{ty} \} \quad (3.13)$$
where

- \( R_f(PIV_{ty}) \) is the function the system needs to perform when managing the provision of identity verification \( PIV_{ty} \); and

- \( checkIdentity_{ty} \) is the function of checking the correctness of the identity of \( C \) that is provided in \( PIV_{ty} \). In addition, \( checkIdentity_{ty} \) checks whether this identity is authorized to perform the service or not.

The required functions to manage the \( PI_{ty} \) can be defined as:

\[
R_{\text{func}}(PI_{ty}) = \{ checkSource_{c,ty}, checkInformation_{ty} \} \quad (3.14)
\]

where

- \( R_f(PI_{ty}) \) is the function the system needs to perform when managing the provision of information \( PI_{ty} \); and

- \( checkInformation_{ty} \) is the function of checking the correctness of the information that is provided in \( PI_{ty} \).

The processing cost of the system for each function \( r_f \in R_f \) can be defined as \( c \). The system-required processing for communication \( a \) of type \( ty \) can thus be defined as:

\[
R_p(a_{ty}) = \sum_{i=1}^{n} c_i \quad (3.15)
\]
where

- \( n \) is the number of required functions for managing \( a_{ty} \);
- \( c_i \) is the system-processing cost that is required to perform the function \( i \);
- \( R_p(a_{ty}) \) is the system processing necessary to manage \( a_{ty} \). \( R_p(a_{ty}) \) calculates the required process of the government system \( S \) in managing \( a_{ty} \).

**CGeSS Performance**

The system performance \( SP \) is defined by the number of transactions that can be processed during a specific period of time \( t \), such as transactions per second. In other words, it measures the number of clients the system can serve in a specific time period \( t \). System performance can be enhanced by either minimizing the required processing time for \( ST \) or by increasing the number of clients \( C \) that can be served at a time. Thus the relationship between system performance \( SP \) and the required processing of a transaction \( R_p(ST) \) is an inverse relationship as follows:

\[
SP \propto \frac{1}{R_p(ST)} \tag{3.16}
\]

**Required Security**

In this section we demonstrate the security that is necessary for the CGeSS \( G \).
The CGeSS should satisfy a set of required security properties $RSec$ which is defined as:

$$RSec = TS \cup SysSec \cup \{SAvailability\} \quad (3.17)$$

where

- $TS$ represents the required security of the transaction;
- $SysSec$ represents the required security of the system $S$; and
- $SAvailability$ represents the availability of $S$. The system should have a high level of availability at all times in order to protect $S$ from being unreachable as a result of DoS attack.

**Transaction Security**

The CGeSS should satisfy the following service’s transaction security set:

$$TS = \{Parties'\ Authentication,\ Transaction\ Privacy,\ Transaction\ Integrity,\ Non\ Repudiation\} \quad (3.18)$$

where

- *Parties' authentication:* Both $C$ and $S$ can be authenticated by each other;
- *Transaction privacy:* Any information in a message of the transaction is only readable by the communication parties $C$ and $S$;
• **Transaction integrity**: Both parties of communication $C$ and $S$ can ensure that the message has not been changed during the transmission; and

• **Non – repudiation**: No transaction can be denied by the party that has performed it.

### System Security

The service system $S$ should perform the following set of activities in order to be protected from DoS attack:

$$SysSec = \{\text{AttackFiltering}, \text{AttackDetection}, \text{AttackPrevention}\}$$  \hspace{1cm} (3.19)

where

• **AttackFiltering**: Represents the process of filtering communication that should not enter the system;

• **AttackDetection**: Represents the process of examining the traffic in order to find DoS attacks; and

• **AttackPrevention**: Represents the process of stopping attacks that are detected.

### System Availability

Online services should be available at all times for clients. CGeSS should therefore also have high levels of availability. This can be achieved by rendering systems impervious to DoS attack. The relationship between the
system availability and the system performance is a direct relationship as follows:

\[ S_{\text{Availability}} \propto SP \] (3.20)

By combining the Equations (3.20) and (3.16) we get the following:

\[ S_{\text{Availability}} \propto \frac{1}{R_p(ST)} \] (3.21)

If we minimize the required processing of \( ST \), the system will have higher availability. The system can therefore serve more clients at a time if clients can be served with less processing. Based on Equation (3.15) we can rewrite Equation (3.21) as follows:

\[ S_{\text{Availability}} \propto \frac{1}{R_p(SO_{ty}) + R_p(PIV_{ty}) + R_p(PI_{ty}) + R_p(RIV_{ty}) + R_p(RPI_{ty}) + R_p(SR_{ty})} \] (3.22)

However, \( RIV_{ty} \), \( RPI_{ty} \) and \( SR_{ty} \) are not performed by the system unless other communications are received. If one of the required functions \( R_f(SO_{ty}) \) has an unsuccessful result, then \( RIV \) will not be executed. In addition, if one of the \( R_f(PIV_{ty}) \) is unsuccessful, then \( RPI \) will not be executed. Also, if one of \( R_f(PI_{ty}) \) is unsuccessful, then \( SR \) will not be executed. Therefore \( R_f(SO_{ty}) \), \( R_f(PIV_{ty}) \) and \( R_f(PI_{ty}) \) require improvement in order to filter the attackers’ communications with minimal processing costs. This will enhance the performance of system \( SP \). Based on this, we can rewrite Equation (3.22) as:
By combining Equation (3.23) with Equations (3.12-16) we obtain the following:

\[
S_{\text{Availability}} \propto \frac{1}{R_p(SO_{ty}) + R_p(PIV_{ty}) + R_p(PI_{ty})}
\]

From Equation (3.24) we can see that the \( checkSource_{C,ty} \) event is the filtering event for each step. If this event is efficient, then \( SP \) will be improved and system availability will be enhanced. In addition, if this event is accurate, then system availability will be increased, because the DoS communication will be filtered and rejected, and subsequent transaction functions will be protected from useless and extraneous processing.

### 3.3.1 Summary

This section has provided a formal model for a CGeSS. This model defines communication parties, the communication environment, transactions, system performance and security requirements. In addition, this model describes the relationship between these elements. We found that system availability can be optimized by enhancing the number of clients that are served at any one time. This can also be achieved by minimizing the processing required to perform
transactions. In addition, we found that a single client action can cause multiple actions and have repercussions throughout the system. This vulnerability can be used by the attacker to commit low-cost DoS attacks on a system. However, the system designer can use this design to advantage by implementing multiple layers of packet filtering, starting with a lightweight filter and ending with a more expensive filter (in processing terms). Using this design, if the system receives an attack, it can stop the bogus communication at the first layer of filtering and avoid the expense associated with processing the multiple actions.

This formal analytical model gives us the necessary detail and guidelines to now propose a security model for protecting systems against DoS attack.

### 3.4 The Proposed Security Model

The previous section used a formal model to describe the requirements necessary to protect CGeSS against DoS attack. Due to the limitations of the existing security models designed to protect online systems against DoS attack, we propose a new model called DoS Prevent, Identify and Filter (DoS-PIF), shown in Figure 3.3. This model secures the necessary communications for the critical services and can be used to build a solution that stratifies the DoS solutions key requirements (Section 2.4). This overcomes the limitations of existing models (described in Section 2.3).

DoS-PIF consists of four main components: Filtering, Analysis, Attack Recognition and an Avoidance-Strategy Manager. These components work
together as a single defence unit and use the same log to provide the three model tasks: packet filtering, attack detection and attack prevention.

![Diagram of the Proposed Security Model DoS-PIF](image)

**Figure 3.3 The Proposed Security Model DoS-PIF**

The packet-filtering task is performed by the Filtering component. This component inspects received packets and makes decisions, based on its rules, to either accept or deny packets. The detection task is performed by the Analysis and Attack-Recognition components. This task is responsible for checking both accepted and denied communications in order to garner information about the DoS attacks. Attacks might be detected through either a single packet or multiple communications. In addition, the attack-detection task should be able to identify the type of the attack and its source.
The attack-prevention task is performed by the Avoidance-Strategy-Manager component. This task involves designing defence strategies based on the information about the detected attacks (which comes from the attack-detection task). This strategy is then implemented in the Filtering component.

Within DoS-PIF, the attack filtering, detection and prevention activities all take place within the one defence-system component. The communication between the system and the clients, in valid cases, only goes through the Filtering component. However, when the system receives an attack or is suspicious about a communication, the Analysis, Attack-Recognition and Avoidance-Strategy-Manager components will also be used. These three components will try to (i) identify the nature of the received attack; (ii) recognize the source of the received attack; and (iii) design strategies that are implemented in the Filtering component to protect the system from the attack. In the following sections, each component of the proposed model is described.

3.4.1 The Filtering Component

The Filtering component is the gateway through which the client communicates with the system. It checks and filters all communications from all clients. The filtering rules are held in this component. These rules are changed if new avoidance strategies are received from the Avoidance-Strategy Manager. Received packets are either accepted or denied. In both cases, the results are reported to the Analysis component. The reasons for denying packets are attached to the denied packet details. The denied communications
are reported to the Analysis component to determine whether the system is
under an attack. The accepted communications are also reported to the
Analysis component, because, while these communications may not have been
attacks themselves, they may have been used as parts of an attack.

The main tasks of the Filter component are receiving client requests, filtering
requests and sending system replies to clients. The Filter component may
attach controls to the client-reply packets. There are two types of controls. The
first type is the information that the client is required to attach to his or her next
communication. The system can use this attached information to verify the type
and source of the communication and also to ascertain the length of time
between the current communication and the previous one. The second type of
control can delay a client’s next communication. A delay can be implemented
for a single client, a group of clients or for all clients.

The Filtering component requires an efficient and strong packet-checking
technique, not only because of its need to attach controls, but also because of
the attributes that are included in the received packets. These attributes come
from using the communication controls (discussed in the next section, Section
3.4.1). This gives the Filtering component the ability to drop DoS-attack
packets from their headers and thus enhance the system’s ability to process
more packets at a time. In addition, controlling the time between client
communications gives the Filtering component the ability to organize the
arrival time of client communications. A busy system juggling a high volume
of communications may drop some communications; controlling the arrival time minimizes the likelihood of this eventuality. The ability of a delay function to limit, and hence manage, the load on a system’s resources is also useful if, for example, a system’s resources need to be temporarily redirected to help with other processing tasks (such as dealing with a DoS attack). More details about these controls are discussed in the following section.

**Communication Controls**

A server cannot stop initial communication directed to the system. The server can only filter subsequent communication after a first communication has been received. An additional control to limit subsequent inappropriate communication would ease the filtering task. This can be achieved by giving a control to the client to be included in his or her next communication (Figure 3.4). The system can then use this control as an additional filtering technique.

![Figure 3.4 Using Controls in Communication between the Client and the Server](image)

In Figure 3.4, the client sends a request to the system (Communication X). Once the system receives this request, it designs a control, C, based on
information related to the client and Communication X. This control will then be given to the client with the reply communication. The client is then required to provide this control to the system with his or her next communication. The system can use this control as an additional layer of communication filtering.

The use of controls provides the following features to the filtering process.

- **Communication information:** A control can be used to provide information about a communication (such as the source of the communication, the destination of the communication and the nature of the data). This alleviates the system from having to hold information relating to client communication and thus saves memory space. Information is stored in the control, which is then sent to clients. In addition, the attacker cannot spoof or change any field of the communication attributes because the attributes are integral to the control.

- **Lightweight filtering:** A control, because it includes information about a communication, can be used to provide a first-defence filtering technique for received communications. Should a client use an incorrect control when communicating, the system will deny the communication and make a decision (such as blocking any communication that comes from the client for a certain amount of time). Some communications (such as encrypted communications) require substantial processing. If these communications are used in an attack – even if they are denied – part of the communication can still monopolize system resources by keeping the system busy with
high-level filtering tasks and consequently decrease system performance. By adding lightweight filtering controls, these communications can be dropped before they are processed.

- **Communication-Arrival Timing:** The system can instruct a client to postpone the sending of the next communication. The postponement affects the arrival time of the communication. A control can be used to ensure that the delayed client communication comes after the specified time. This type of control can be used to manage the timing of communications received from a particular client or from a specified group of clients in order to conserve processing power. For example, if part of a system is under attack, then a system can delay clients and conserve processing power in order to manage the attack. In addition, this feature of the control is used to achieve the goal of proof of work [144]. This feature requires the CPU of the client’s machine to perform an additional processing as a solution to preventing DoS attack.

- **Sequence Control:** A control can be used to ensure the appropriate sequence within a client’s communication process. By using this type of control, the system ensures that no client can start communicating with the system from the middle of the communication process, because in any middle step of the communication, the client is required to provide a specific control which would have been given to the client in a previous step.
More details about the technical implementation of the controls are discussed in the next chapter (Section 4.2). In the next section, the second component of DoS-PIF is discussed.

3.4.2 The Analysis Component

The Analysis component is responsible for analyzing traffic information. It receives the results and the details of accepted and rejected communications, (such as the source of the communication, the admission point and the time of arrival). It groups these data every specific period of time based on the source and type of communication. In addition, the Analysis component gives statistics about the sources and types of communications. The Analysis component analyzes and groups these data for use by the Attack-Recognition component.

3.4.3 The Attack-Recognition Component

The Attack-Recognition component identifies DoS attacks by using data from the Analysis component. The Attack-Recognition component must thus recognize DoS-attack patterns. Once the Attack-Recognition component finds a DoS-attack pattern in the analyzed data, it recognizes the received attack and tries to find the source of the attack. It subsequently reports these specifications to the Avoidance-Strategy-Manager component. In addition, the Attack-Recognition component uses additional system-behaviour alerts from the Analysis component. This data may indicate that the system is suspicious of
being under DoS attack, but that this attack has yet to be detected. This warning helps the system to detect a DoS attack in the early stages of the attack. In this case, the Attack-Recognition component raises this concern by defining the required verification and passing it to the Avoidance-Strategy Manager.

3.4.4 The Avoidance-Strategy-Manager Component

The Avoidance-Strategy Manager prevents the DoS attacks. It designs filtering rules based on detected attacks and accepted communications. It then passes these new filtering rules to the Filtering component in order to avoid a DoS attack. The rules are designed both to prevent actual attacks and to verify the existence of an attack should there be an alert. The decisions made in the Avoidance-Strategy Manager are drawn from the information received from the Attack-Recognition component.

3.4.5 Summary

In this section, the proposed security model (DoS-PIF) and its components were discussed. The model is designed to protect CGeSS against DoS attack. It was developed using formal model specifications and requirements, and can be used to design solutions that satisfy the DoS defence system key requirements (Section 2.4). DoS-PIF avoids the disadvantages of previous models but maintains their advantages (discussed in Section 2.3). The model performs three security tasks: packet filtering, attack detection and attack prevention.
These three tasks are self-contained within the model, use the same log, communicate with each other to produce convergent protection decisions and are directly under the model’s control. Consequently the tasks are well integrated, self-reliant and independent of components belonging to other external models. DoS-PIF also includes communication controls in order to enhance the efficiency of the filtering component and to manage communications with clients and improve filtering decisions.

Based on DoS-PIF, in the next section we propose a security approach as a viable solution for protecting CGeSS against DoS attack.

### 3.5 A Holistic Approach for Securing and Protecting Critical Government eService Systems (HASP-CGeSS)

Based on the security model DoS-PIF, in this section we propose a security approach called the Holistic Approach for Securing and Protecting Critical Government eService Systems (HASP-CGeSS) (Figure 3.5). This approach is designed to filter, recognize and avoid DoS attacks in CGeSS. This approach consists of three main components: Client Authentication (CA), Authenticated-Client-Communication (ACC) and Analysis. Both of the CA and the ACC components are responsible for communicating with the client, depending on the stage in which the client communicates with the CGeSS. This division is
important in order to identify the nature of the client’s communications activities with the system. In addition, this division helps to provide an appropriate solution for flooding attacks in every part of the system, and in every stage of the client’s communication with the system. Using this division, the system can securely provide the required communication services.

Figure 3.5  A Holistic Approach for Securing and Protecting Critical Government eService Systems (HASP-CGeSS)

In order to provide communication control a new proposed packet-filtering mechanism, called the Token-Filtering Technique (TFT), has been incorporated into this approach. In this technique, the system generates a token for each client’s communication with the system. It provides the filtering process in the CA and ACC components with the filtering features discussed in Section 3.4.1. It also supports the analysis-component decisions in its attack
detection-and-prevention processes. The details of the TFT are discussed in Section 4.3.

The client can request communication with the system through the CA component. Authentication and services to be provided to the clients are determined in this component. If the CA component is satisfied with a client’s communication, all communications belonging to the authenticated client are moved to the ACC component. In this component, the client can request the needed services. The Analysis component is responsible for detecting and avoiding potential DoS attacks while the client is communicating with the system. The Analysis component follows up the client’s communications with the system and then applies strategies to guard against DoS attacks by changing the filtering rules in the CA and ACC components.

The CA, ACC and Analysis components are discussed further in following three sections.

3.5.1 The Client-Authentication (CA) Component

The CA component handles all requests from unauthenticated clients (Figure 3.6). It filters clients’ requests, authenticates clients who send requests, and issues tickets for each client depending on the services each client is allowed to access. The CA provides a secure channel to the client while it authenticates the client. Each client is required to provide a signature to represent his or her identity.
The Filter-and-Redirect (FR) Engine is the first element of the CA component. It receives clients’ requests, filters them, and accepts only valid requests. The second element in the CA component is the Ticket Engine. This authenticates clients and issues different types of tickets for each client. A ticket’s type is determined based on the services relevant to each client. The FR Engine is the main element in preventing DoS attacks in the CA component and so is discussed further in the following section.

**Figure 3.6 The Client-Authentication (CA) Component**

**The Filter-and-Redirect (FR) Engine**

The FR Engine is the main element of the CA component. It receives all unauthenticated requests from clients. These requests are in a specific format and size and are unencrypted (Figure 3.7). In addition to receiving client requests, the engine also performs a number of other tasks.
i. The FR Engine filters requests and performs all the communications necessary to connect the client to the system. However, in order to avoid TCP-SYN flooding attacks, it does not hold any information belonging to clients.

![Diagram of the Filter-and-Redirect Engine]

**Figure 3.7 The Filter-and-Redirect Engine**

ii. The FR Engine communicates with clients in order to authenticate them.

iii. The FR Engine provides a secure channel for client authentication processes.

iv. The FR Engine sends details of rejected packets to the Analysis component.

v. The FR Engine checks client signatures using a function provided by the Signature Issuing and Verification (SIV) component. This function is used to check the signature’s form and format. This process negates the
need for the system to check incorrect signatures and hence conserves
processing time and resources.

vi. The FR Engine sends correct signatures to the Ticket Engine for
authentication.

vii. The FR Engine adds invalid signatures to a black list and requests the
client to renew his or her signature. Any received signature must be
checked against the black list before being processed. This list consists of
two columns containing the invalid signatures and the number of times
each invalid signature has been spoofed (Figure 3.8).

<table>
<thead>
<tr>
<th>Signature</th>
<th>No of Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign. 5</td>
<td>456</td>
</tr>
<tr>
<td>Sign. 101</td>
<td>430</td>
</tr>
<tr>
<td>Sign. 6</td>
<td>399</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
<tr>
<td>Sign. n</td>
<td>1</td>
</tr>
</tbody>
</table>

**Figure 3.8 Example of a Black-Signatures List**

The list is sorted by the second column according to the frequency.
Although the signatures may be correct in format, they may still be
invalid because they have expired or been stolen; in these cases the SIV
will revoke them. Addition of new signatures to the list takes place after
invalid signatures have been rejected by the SIV. Although the checks
against this list might be done only partially, this step eliminates invalid
signatures that might be used by attackers, and also prevents other parts of the system from being occupied by the workload caused by invalid signatures.

viii. The FR Engine sends the next IP address. When a client is authenticated, the FR Engine sends the next IP address to the client so that the client can continue his or her communication. This IP address is in the ACC component and is determined by the Next-IP Engine.

This final task is the end of client’s communication within the CA component of the system. As the next IP address is in the ACC component, all following communication will be through the ACC component (discussed in the following section).

3.5.2 The Authenticated-Client-Communication (ACC) Component

The ACC is the second component of the proposed approach. It communicates only with authenticated clients via a fully encrypted channel. It consists of two main elements: the Authenticated-Client Engine and the Next-IP Engine. The first element filters client packets while the second element determines the address where the client’s next communication will be sent. The roles of each part of this component of the system are shown in Figure 3.9.
Figure 3.9 The Authenticated-Client-Communication (ACC) Component

The ACC component uses the TFT to filter authenticated client requests. In addition, it implements the proposed dynamic multi-point communication mechanism.

Dynamic Multi-Point Communication

We have proposed the dynamic multi-point communication technique to divide the server area into multiple IP addresses. The technique seeks an appropriate communication for each client by dynamically changing between these multiple addresses. Each packet arriving at the system must come through a specific IP address. In Figure 3.10, $IP_1$ receives a communication packet from Client A and Packet B. The communication is accepted, because it had been expected by $IP_1$. However Packet B is rejected because it had not been anticipated at this IP address. On the other hand, at $IP_2$, the communication from Client C was rejected and Packet B was accepted, because Packet B had
been expected at this IP address, while the communication from Client C had not been anticipated.

This dynamic multi-point communication technique is used to represent an additional layer of packet filtering. This layer filters packets by their headers, before they are decrypted. In addition, the technique helps to identify, at the time of the attack, the exact type of attack. The significance of using this technique to protect the system against DoS attack is discussed further in the next section where we describe the authenticated-client engine that uses this technique.

![Diagram](image)

**Figure 3.10** An Example of Using the Dynamic Multi-Point-Communication Technique in the Authenticated-Client-Communication Engine
The Authenticated-Client Engine

The Authenticated-Client Engine handles communications for all authenticated clients. It uses more than one IP address, because it implements the dynamic multi-point-communication technique. In addition, it performs two more steps to protect this area from flooding attacks.

i. When the packet is received, the packet is initially checked based on the data shown in the packet itself. After this, its existence in the expected-communication list in the receiver IP address is checked. This check occurs because each IP handles only specific packets and communications from specific clients. These specifications are determined by the Next-IP Engine. Every unexpected packet is dropped at this stage. After each client’s communication with the system, the client should receive a new IP address that can be used to continue further communication. This IP address-checking in the dynamic multi-point-communication technique enables a layer of packet filtering using packet headers. The specification of any rejected packet is reported to the Analysis component.

ii. After the packet succeeds in passing through the previous filters, the packet will be decrypted. Its header should be included in the encrypted part in the packet (Figure 3.11). The last layer of filtering then takes place, whereby the encrypted packet’s header is compared to the decrypted packet’s header.
3.5.3 The Analysis Component

While the first two components of the security approach focus on packet filtering, the Analysis component performs attack detection and prevention. The performance of the system should not be affected by this component, because it performs tasks in the background while priority is given to the filtering tasks taking place in the other components. However, in particular cases, such as being under a substantial attack, the Analysis component might need higher processing power to prevent the system from an attack. The Analysis component has four major elements: (a) Analysis; (b) Accepted-Communication-Learning; (c) Attack Recognition; and (d) the Avoidance-Strategy Manager (Figure 3.12).
The Attack-Recognition element works in the background of the system and uses the results of the packet filtering to analyze filtered communications in order to identify attacks. If an attack is identified, the Avoidance-Strategy Manager designs a defence strategy that the system can use to prevent further attack.

The Breach-Analysis element is responsible for analyzing breaches (that is, dropped communications) that are received during a specified period of time. It then groups these breaches based on sets of criteria (such as the filters that caused the dropped packets and the sources of the failed communications). The results of the analysis are moved to the Attack-Recognition element.

**Figure 3.12 The Analysis Component**
In the Attack-Recognition element, the rules of attack definitions are created. These rules are used to identify the nature of the potential attacks through the analyzed results that were received from the Breach-Analysis element. The Attack-Recognition element should be able to define even complicated merged attacks, because this element stores information about the causes of dropped packets. Once an attack is detected by this element, the specification of the attack is reported to the Avoidance-Strategy Manager in order to protect the system against the attack.

The Avoidance-Strategy-Manager element is responsible for designing attack-avoidance strategies. It performs two main tasks: determining an appropriate defence strategy and applying the selected defence strategy. Determining a defence strategy involves receiving the defined attacks from the Attack-Recognition element and then defining the range of the accepted communications (if necessary). Depending on the nature of the attack, the Avoidance-Strategy Manager then makes the required decisions (such as adding an IP address to the IP-address black list or asking the Next-IP Engine to withdraw direction from a questionable client for a time). If the Avoidance-Strategy Manager decides to only accept communication from a particular range of sources, it receives this range of sources from the Accepted-Communication-Learning element.

The Accepted-Communication-Learning component receives the details of the accepted communications and then analyzes them to define the different ranges
of the sources of the accepted communications. This is necessary to prevent certain types of attacks, such as substantial distributed DoS attacks that use external sources. The Accepted-Communication-Learning element performs two main tasks: (i) saving the sources associated with accepted communications over a period of time; and (ii) designing and updating the accepted communication boundaries in the Communication-Source Boundaries. This information will be used in the case of a large-scale attack. In such a case, the system might only accept communication from within specific source boundaries. These sources should have a high frequency of successful communication with the system.

3.6 Conclusion

This chapter identified a certain type of government service system that requires a high level of protection because of its critical nature. A formal model analysis was then conducted to examine the components of CGeSS. This model defines communication parties, the communication environment and transactions, and describes the relationship between these elements. Using this formal model, we concluded that system availability can be enhanced by minimizing the processing required to achieve service transactions. In addition, the system can reduce the influence of low-cost DoS attacks by implementing multiple layers of filtering. These layers of filters should start with a lightweight filter and end with a more expensive filter. This formal model
gives us the necessary detail and guidelines to design a security model for protecting systems against DoS attack.

The chapter elaborated on the results of the formal model by proposing a novel security model (DoS-PIF) to protect critical government systems against DoS attack. This model can be used to design solutions that satisfy the DoS defence system key requirements. This model was further developed into an approach – HASP-CGeSS – in order to make it applicable to a real-world context.

The proposed approach has practical significance in filtering, detecting and preventing DoS attacks because it performs these three protection activities within one integrated defence system. The defence system also controls communications between the system and the clients. In addition, it is a real-time solution and is not affected by changes in the configuration of the network topology.

This chapter has discussed the theoretical part of the proposed solution. Our proposed technical specification for this solution is discussed in Chapter 4.
Chapter 4

The Token-Filtering Technique

4.1 Introduction

In the previous chapter (Chapter 3) we examined the communication process and the business flows in online government systems. In addition, using a formal-analysis model, we described the actors in CGeSS, the relationship between these actors and the requirements of the system. A new security model called DoS Prevent, Identify and Filter (DoS-PIF) was also proposed. This model was designed based on the formal-analysis model and can be used as a base for any new security solution in protecting CGeSS. Based on this model, a holistic security approach called the Holistic Approach for Securing and Protecting Critical Government eService Systems (HASP-CGeSS) was proposed. The specifications of this approach’s components and the relationships between these components were discussed.

This chapter elaborates on the HASP-CGeSS in two aspects: the communications and the processes that the HASP-CGeSS performs in order to guard against DoS attack. In order to implement the communication control and enhance the security of the HASP-CGeSS, a new packet-filtering mechanism, called the Token-Filtering Technique (TFT), is proposed in this
chapter. The TFT is incorporated as a part of the HASP-CGeSS’s communications and is used as an authentication and filtration layer for packet filtering. To detect and block repeated communications, we designed two filtering lists based on the TFT.

The use of the TFT in the HASP-CGeSS’s communications is described further in a proposed new protocol called the Token-Filtering-Technique Protocol (TFTP). This protocol shows the messages between the client and the system in the HASP-CGeSS. In order to protect the system against DoS attacks in the TFTP, the HASP-CGeSS performs three main processes: Filtering-Process, Detection-Process and Prevention-Process. These processes are integrated to protect the system against DoS attack.

The remainder of this chapter is organized as follows. The next section (Section 4.2) describes the token specifications and the token generation-and-verification processes. Section 4.3 defines the two lists used in the TFT. The TFTP is discussed in Section 4.4. Section 4.5 illustrates the system-protection processes of filtering, detection and prevention. The last section, Section 4.6, summarizes the chapter.

### 4.2 Token Overview

The token in the TFT is an application of the communication control. It is in a plain-text format. The token is designed to offer efficient packet filtering using packet headers. It is generated by the system (server), based on information
about the client and the client’s current communication with the system, and is
given to the client to be added to the client’s next packet header. Figure 4.1
illustrates the use of tokens in communication.

![Diagram of token usage in communication]

**Figure 4.1 An Example of Using Tokens in Communications**

Once the system receives a request for communication by a client, the system
generates the token $T_1$. This token is attached to the reply message sent to the
client. As the client is required to include this token in the next packet header,
the system can use this token in the filtering process. If the system accepts this
message, then the system will generate another token, $T_2$, and attach it to the
reply packet. Again the client is required to include the new token in the
The client’s next packet header directed to the system. The client is thus required to include a token in all communication after the first packet. TFT uses packet headers and facilitates deep screening of the packets prior to packet decryption. The token represents information about the packet and the communication. The time and effort taken by the system to generate and check the token should be minimal; in contrast, it should be very difficult for an attacker to decipher the token.

To maximize the goal of using the TFT, the token should have the following characteristics.

- The token should have a plain-text format, so all its parts are unencrypted. This makes the token easier to generate and check.
- The token should be generated by the system.
- The token should give complete information about the communication so that the system does not need to store information about clients and their communication.
- The value of the token should be calculated using a number of parameters based on the specifications of the client’s details and the client’s current communication with the system. These parameters are discussed in Section 4.2.1.
• The token should only be used once for each packet. The client must use a specific token, which is given by the system, for each packet. This is important to avoid the spoofing of tokens.

• The token should give the system all the information required to perform packet filtering. This increases the dependability of the token-filtering process.

• The token should be incomprehensible and impervious to attackers. The system should detect any attacker’s attempt to change the token.

• The token is used in order to drop flooding, expired and spoofed packets based on their headers, as all the required filtering information is contained in the token which is stored in the header of the packet.

• The token should be able to show delayed communication and the time of this delay. This is important for organizing communications to protect the system against flooding attacks. If a system receives a flooding attack, the system should be able to delay new communication so as to offer more resources to recover from the attack. Delaying communication has the added benefit of making a DoS attack more difficult to launch, because the attacker must wait for communications to restart.

• The token should require minimal time for generation and verification in order to enhance the efficiency of the system’s filtering processes.
The implementation of these features will be discussed in the next section, where the token’s specifications are described.

4.2.1 Token Specifications

In this section we discuss the methodology of designing an effective token. A proposed method for designing a token is given as follows.

The token consists of two main parts:

\[
Token = \{ Token_{value} \} \cup Token_{Keys} \tag{4.1}
\]

where

- \( Token_{value} \) represents the token value. The value is shown in the token after the value has been calculated and is used to validate the token; and

- \( Token_{Keys} \) represent the set of key values that are shown in the token. Key values can be used in the filtering process.

The token’s value is a result of calculating the token parameters \( tokenParameters \) using a token’s formula \( TF(x) \) as follows:

\[
Token_{value} = F(tokenParameters) \tag{4.2}
\]

In the following discussion, the parameters used to generate the token, the methodology for generating random-number parameters, the token’s formula that is used to generate the token’s value, the token’s keys, and the final form of the token, are illustrated.
Token Parameters

Four important factors are involved in choosing the token’s parameters $tokenParameters$ that are used to represent the token’s value $TokenValue$.

i. The parameters must give full details about the communication because the system will use this information to filter packets. Thus the system should not hold any information about the communication. All required information should be stored in the token. This protects the system from receiving memory DoS attacks, such as SYN attacks. In addition, it protects the system from spending time searching the memory. This factor makes the filtering process stateless.

ii. Most of the parameters used should not be constants. This reduces the chances of an attacker identifying the token-generation method. It is important to include random-number parameters to support this factor.

iii. One or more parameters should be based on time, in order to make the token’s value difficult to calculate, and to ensure that the value of at least one parameter varies from time to time. The time unit used should be small enough to ensure this factor. If we use a parameter based on a long time unit, such as an hour, then this parameter will be similar to constant values for the period of the time unit. This conflicts with the second factor.
iv. At least one parameter should represent the generation time and the expiration time of the token. This will support the Filtering-Process, by ensuring the process knows when the previous communication took place and enabling it to detect expired tokens.

v. At least one parameter should represent the delay period for the communication. This is an important factor that allows the TFT to delay a client’s next communication.

vi. The chosen parameters should generate a unique token in order to make re-using or spoofing the token impossible.

Based on the above factors, we use the following values in the proposed token-design technique to generate the token:

- Client IP (I): This is the IP address of the client; the source of the packet received by the server. By including this parameter in the token, the re-use of this token by another source is impossible. The system will use the source IP of the client, which is shown in the packet header, to verify the token. If the token has been re-used, then the verification result will be negative. This parameter is a response to the first factor above.

- Server IP (S): This is the IP address of the server and also represents the destination IP address for the client’s next communication. This value ensures that the token received by the client is designed for the packet which is to be sent to the receiving server. This parameter is also a response to the first factor of the above four factors.
• Communication state \((M)\): This represents the client communication’s stage or state with the system (that is, establishing or established, authenticating or authenticated). These states are represented by integer values (Table 4.1). This parameter again responds to the first factor above.

| Table 4.1  Communication States Values |
|-----------------|------------------|
| State           | Value            |
| Establishing    | 1                |
| Established     | 2                |
| Authenticating  | 3                |
| Authenticated   | 4                |

• Token-Generation Time \((T)\): This represents the time of the token’s generation (for example, in 022230412055, 022 refers to the 22\textsuperscript{nd} day of the year, 23 refers to 11 pm, 04 refers to minutes, 12 refers to seconds and 055 refers to milliseconds). This parameter supports the above fourth and sixth factors.

• Accepted duration \((D)\): This describes the length of time that the server can wait for the next packet before the packet expires. This parameter is also included as a response to the fourth factor above.

• Flag value \((F)\): This flag is optional. It can be used to verify activities such as providing notification of a specific kind of attack or to
distinguish a communication in a certain way. For example, a system may need to verify that all communications served in one step have moved to the next step. The system can set a flag for these communications in the first step and count these flagged communications in the next step.

- Delay-time value ($V$): This parameter represents the length of delay prior to a client next packet commencing communication. Once the delay-time value has expired, the client can start communications. The value affects the delivery time of the packet at the server. If a packet arrives at the server before the delay-time value has expired, the packet is likely to be illegitimate and the packet will be dropped. This parameter is selected to respond to the fifth factor detailed above.

- Random numbers ($RN_1$, $RN_2$, $RN_3$): These are random numbers from the random-numbers table that are used to generate the token. They are chosen based on the token’s generation time. The system must have the ability to find these values when the token is received in order to verify the token. This parameter is used as a response to the third factor. In addition, it consists of at least three values as a response to the second factor. This makes the token more difficult for attackers to guess.

The implementation of the random-numbers table is discussed in the following section.
The Random-Numbers Table

The random numbers table is used for token generation and verification (Table 4.2). It consists of four columns. The first column shows the time period that represents the row that is used for generating and verifying tokens. Each cell of this column consists of two points of time values: $TP_{\text{start}}$ and $TP_{\text{end}}$. $TP_{\text{start}}$ is the start value of the time period and $TP_{\text{end}}$ is the end value of the time period. The following three columns represent the random numbers $R_1$, $R_2$ and $R_3$ that are used to generate and verify tokens.

<table>
<thead>
<tr>
<th>Time</th>
<th>RN 1</th>
<th>RN 2</th>
<th>RN3</th>
</tr>
</thead>
<tbody>
<tr>
<td>230400-230414</td>
<td>12345678</td>
<td>58635215</td>
<td>25685495</td>
</tr>
<tr>
<td>230415-230429</td>
<td>256854952</td>
<td>2158698</td>
<td>1266857</td>
</tr>
<tr>
<td>230430-230444</td>
<td>25487369</td>
<td>25985321</td>
<td>3698542</td>
</tr>
<tr>
<td>230445-230459</td>
<td>1259874</td>
<td>21598532</td>
<td>2369854</td>
</tr>
<tr>
<td>230500-230514</td>
<td>1266857</td>
<td>25896314</td>
<td>58635215</td>
</tr>
<tr>
<td>230515-230529</td>
<td>3698542</td>
<td>98745632</td>
<td>2158698</td>
</tr>
<tr>
<td>230530-230544</td>
<td>2369854</td>
<td>52369814</td>
<td>25985321</td>
</tr>
<tr>
<td>230545-230559</td>
<td>1250875</td>
<td>21448577</td>
<td>8934809</td>
</tr>
<tr>
<td>……</td>
<td>……</td>
<td>……</td>
<td>……</td>
</tr>
</tbody>
</table>

The length of this list is based on two factors: (i) the time duration necessary for the system to accept the next communication from a source; and (ii) the number of random numbers that are used in each minute. So in Table 4.2, the first row of random numbers (that is, ‘12345678’, ‘58635215’ and ‘25685495’)

---

130
will be used for any token generated between 11:04:00 PM and 11:04:14 PM. However, tokens that are generated using this row should be different from each other. This requirement is discussed in the next section when the issue of the token’s formula is raised.

In Table 4.2, the system generates four rows of random numbers every minute. In this example, the suggested acceptable time between communications from a source is two minutes. In this case, the length of the list will be eight. However, the length of this time is configurable by the system administrator based on the implementation requirements. The shorter the time available to accept the next communication, the greater the security for the system, because the shorter period reduces the possibility of flooding attacks that re-use the tokens.

This table can be described as a matrix $A_{n \times 3}$ as follows:

$$
A_{n \times 3} = \begin{pmatrix}
12345678 & 58635215 & 25685495 \\
\vdots & \vdots & \vdots \\
\alpha_{n1} & \alpha_{n2} & \alpha_{n3}
\end{pmatrix}
$$

(4.3)

where

$$
RN_{x}(T_y) = \alpha_{yx}
$$

(4.4)

and where the element $\alpha_{yx}$ is located in the matrix $A_{n \times 3}$ in the column $y = 1,2,3$ and the row $x = 1,2, ..., n$ and $T_y \in [TP_{start}, TP_{end}]$. 
The Token’s Formula

The token’s formula is used to represent the token’s parameters, \(tokenParameters\), in one value. One or more different calculation operations can thus be used to generate one value which represents \(I, S, T, D, M, F, V, RN_1, RN_2,\) and \(RN_3\). Four factors need to be considered in designing the token’s formula.

i. The token resulting from the formula should require minimal effort on the server’s part to generate and verify. This is important in order to increase system efficiency when filtering packets.

ii. The formula should be designed in such a way that it is difficult for attackers to decipher the formula.

iii. The largest exponentiation in the token’s formula should be the largest number of tokens that the attacker needs to predict in order to decipher the method of generating the tokens, because the token’s value will be the result of a more complex formula. However, this factor should be used carefully, because it directly affects the performance of the system: the higher the value, the poorer the system performance.

iv. The formula should be changed from time to time.

Responding to the above factors, the following formula for token generation is proposed:
We can see that all the token’s parameters have been used in the above token’s formula. Any change in these parameters will give an incorrect result in token verification, because the token is verified based on the token’s formula. This is discussed further in Section 4.2.3.

In this formula we use multiplication operations and exponentiation. This is due to the requirements of difficulty and complexity, which are mentioned above in the second and third formula-design factors. However, we avoid large values in exponentiation so that we do not affect the efficiency requirement for the generation and verification of the token, as mentioned in the first factor. Also, the system should change this formula frequently to make deciphering it more difficult, as mentioned in the fourth factor.

In the event that the system uses this formula to generate a token, all these parameters are known to the system from the received packet fields and the system time. In the token verification, $T, D, M, F$ and $V$ will be taken from the token itself. They will be shown in the token’s keys (discussed in the next section).

**Token Keys**

The token’s keys are used to filter the received packets. This represents the first level of token checking. In addition, the token’s keys are used to find the
values of the token’s parameters, which are used by the token’s formula to find the token’s value. The token keys are selected by taking into consideration the following three factors.

i. The keys should omit some token parameters in order to avoid the token formula being predicted by attackers.

ii. The token’s keys should not be represented in the same values of the token’s parameters which are used in the token’s formula to generate the token’s value.

iii. The keys should include the generation and expiration time of the token. This enables the system to reject packets associated with flooding attacks by using data included in the keys. Using the expiration time obviates the effort of calculating token values for malicious packets.

Based on the above three factors in the proposed token design we have chosen the following five keys:

\[
\text{Token}_{\text{keys}} = \{\text{key}_T, \text{key}_M, \text{key}_V, \text{key}_F, \text{key}_D\} \quad (4.6)
\]

where

- \(\text{Token}_{\text{keys}}\) represents the keys set that will be included in the token;

- \(\text{key}_T\) represents the token’s generation-time key. This key shows the time at which the token was generated. From this key we can find the
value of the token’s generated-time parameter $T$ when the token is received in a communication;

- $key_M$ represents the stage of communication when the token is included in the packet. The system can use this key to obtain the token parameter $M$;

- $key_F$ represents the time that the next packet should be received after it. This key is used by the system to find the token parameter $V$;

- $key_F$ represents the token’s flag. This key is used to show the token parameter $F$; and

- $key_D$ represents the duration of the token’s validation. The system can use this key to obtain the value of token parameter $D$.

These keys are generated and included in the token in order to be used in the packet-filtering process. These keys are also used to calculate the token’s formula in order to validate the token. The method of generating these keys is discussed in the next section.

**The Time-Generation Key**

The time-generation key $key_T$ represents the time at which the token was generated. It tells the system when the token was generated. Using this key, the system can determine the time period between a communication of a received packet, that has a token, and the previous one. In order to display this value, the
system must generate and include this key. To generate $key_T$, we use the following function:

$$key_T = TokenKey_T(T)$$  \hspace{2cm} (4.7)$$

where the $TokenKey_T(T)$ function converts the current system time $T$ so it can be represented by $key_T$ in order to include this key in the token. This function determines the values of time that should be included in the token. It should be designed to represent the $key_T$ in a form that is different from the original token-generation time format. This avoids malformed attacks in the token itself. However, when the token key is received, it must be understood by the system so that the real value of the token-generation time can be recognized.

To know the real-time value of the token-generation time that is presented in $key_T$, we use the following inverse function:

$$T = TokenKey_T^{-1}(key_T) \iff key_T = TokenKey_T(T)$$  \hspace{2cm} (4.8)$$

where $TokenKey_T^{-1}(key_T)$ tells the system the real value of token-generation time $T$ that is represented by $key_T$ in order to use this time in packet filtering and in token-value verification.

**The Communication-Stage Key**

The communication-stage key $key_M$ represents the stage of communication $M$ of the received packet. When the packet is received, this key enables the system to know the communication stage of the received packet. This alleviates
the need for the system to analyze the packet’s fields in order to know the packet’s stage of communication. To generate the $key_M$, we use the following function:

$$key_M = TokenKey_M(M) \quad (4.9)$$

The $TokenKey_M(M)$ function represents the stage of the communication $M$ as a $key_M$ so it can be included in $Token_{keys}$. However, the system must know the real value of $M$, which is represented in this key, in order to use it in the packet-filtering process and to verify the token value.

To know the real value of $M$ that is represented in $key_M$, the system uses the following inverse function:

$$M = TokenKey^{-1}_M(key_M) \iff key_M = TokenKey_M(M) \quad (4.10)$$

where $KeyValue^{-1}_M(key_M)$ returns the time that is presented in $key_M$.

**The Communication-Delay Key**

The Communication-Delay Key $key_v$ represents the time after which the packet should be received. With this key, the system can know whether the received packet was (intentionally) delayed and whether the received packet thus arrived after the delayed time. To generate $key_v$ we use the following function:

$$key_v = TokenKey_v(V) \quad (4.11)$$
where the $TokenKey_{\nu}(V)$ function converts the state variable $V$ to match the $key_\nu$ form, so it can be used in the $Token_{\text{keys}}$ part. The system uses this function in generating tokens in order to include the communication-delay parameter as communication-delay key $key_\nu$ in the token. This key can be used to know the real value of the communication-delay parameter when the packet is received.

To know the parameter value when the token is received we can use following inverse function:

$$V = TokenKey_{\nu}^{-1}(key_\nu) \iff key_\nu = TokenKey_{\nu}(V) \quad (4.12)$$

where $TokenKey_{\nu}^{-1}(key_\nu)$ returns the value of the communication-delay parameter that is represented in $key_\nu$.

**The Token-Flag Key**

The token-flag key $key_f$ gives the system the ability to mark packets and to know if a received packet is marked. To generate $key_f$ we use the following function:

$$key_f = TokenKey_{\nu}(F) \quad (4.13)$$

where the $TokenKey_{\nu}(F)$ function converts the flag variable $F$ so that it can be represented as $key_f$ in the token. This gives the system the ability to know whether the received token was flagged and, if so, the value of the flag.
To know the value of the flag presented in $key_F$, we use the following inverse function:

$$F = TokenKey_F^{-1}(key_F) \iff key_F = TokenKey_F(F) \quad (4.14)$$

where $TokenKey_F^{-1}(key_F)$ returns the value of $F$ that is presented in $key_F$.

**The Token-Validation Key**

The token-validation key $key_D$ tells the system whether the token of the received packet has expired. This protects the system from repeat flooding attacks. The system can assign different values for each packet or for each group of packets. To generate $key_D$ we use the following function:

$$key_D = TokenKey_D(D) \quad (4.15)$$

where the $TokenKey_D(D)$ function converts the time variable so that it can be represented in the token as $key_D$. When a token is received, the system can check whether this token is expired or not. The system checks the value of the token’s validation parameter that is represented in $key_D$.

To know the value of the token-validation time for a received token we can use the following inverse function:

$$D = TokenKey_D^{-1}(key_D) \iff key_D = TokenKey_D(D) \quad (4.16)$$

where $TokenKey_D^{-1}(key_D)$ returns the time that is presented in $key_D$. 
These five keys are used to calculate the $Token_{value}$ in order to validate the token. We use functions to generate these keys and to obtain the values of the token’s parameters represented in these keys. These functions give the token’s designer the ability to represent the token keys using values that are equivalent to their real parameter values. These measures are designed to thwart attackers by defying their understanding of the keys. In addition, the keys are allocated with the token values in one piece of data to represent the whole token form.

**The Token Form**

The final form of the token is as follows (Figure 4.2).

\[
\begin{align*}
\{ & M \cdot I \cdot V + S \cdot RN_1 \cdot T \\
& + RN_2^2 + RN_3 \cdot D \cdot F \}\{key_T\}, \{key_M\}, \{key_V\}, \{key_F\}, \{key_D\}
\end{align*}
\]

Figure 4.2 Token Form

where

- $\{ M \cdot I + S \cdot RN_1 \cdot T + RN_2^2 \cdot F + RN_3 \cdot D \}$ represents the token’s value $Token_{value}$; and

- $\{key_T\}, \{key_M\}, \{key_V\}, \{key_F\}, \{key_D\}$ represents the token’s keys $Token_{keys}$.
So the token has six main parts, \( M \times l + S \times R N_1 \times T + R N_2^2 \times F + R N_3 \times D \), \( key_r \), \( key_m \), \( key_v \), \( key_f \) and \( key_d \). The first part is the token value \( Token_{value} \), and the following five parts represent the token’s keys \( Token_{keys} \).

The token’s details, which have already been discussed in this section, are used as the primary input for the processes of token generation and verification. These two processes are discussed in the following sections.

### 4.2.2 Token Generation

Before the system starts generating tokens, it must generate the random-numbers table. When the system receives a client’s request, or a packet that has a valid token, it generates a new token. This token is sent with the reply packet to the client to be included in the client’s next packet header. The steps involved in generating the token are illustrated in the flowchart below (Figure 4.3).

As shown in Figure 4.3, token generation starts by reading the system’s time, because the token’s value fully depend on it. This gives us two important features. First, the token is difficult to compromise, because the value of each token is based on the point of time at which the token was generated. Each token will thus be different from other tokens. Second, the system knows the point of time at which that the token was generated. This gives the system the ability to know the expiration time of the token.
Figure 4.3  Token-Generation Flowchart
After the system reads the point of time $T$ at which the token was generated, it obtains the three random numbers $(RN_1, RN_2, RN_3)$ from the random-number matrix $A_{n \times 3}$ based on $T$. These random numbers are used to calculate $Token_{value}$. The system then finds the client’s IP address $I$, the next IP address of the system $S$ that the client can use for communication with the system, the communication stage for the following communication $M$, the flag $F$ which is used if the system decides to assign a flag for the client’s next communication, the time validation duration of the token $D$, and the delay time of the client’s next communication.

These six values, along with the three random numbers and the time $T$, are used to calculate the token’s value $Token_{value}$ by using the token’s formula. The system subsequently finds the values of the token’s keys $key_T$, $key_M$, $key_V$, $key_F$, and $key_D$. The system then collates all of the token’s components into one part, $Token$. $Token$ is then sent to the client with the reply communication.

### 4.2.3 Token Verification

Any packet that comes from the client after the initial request packet must contain a token. Packets that do not contain a token will be dropped. As we mentioned before, the token gives the system the ability to filter packets based on the packets’ tokens and headers. In this way the token prevents spoofing attacks, because it gives all the required information about the packet in an
unencrypted short text. This text is unchangeable by the client. The TFT provides a layer of packet filtering before packets are decrypted and thus becomes the first layer of packet filtering in the system. In addition, the token can be used to detect and verify DoS attacks.

In the following section, we discuss five important checks that can be performed on the received packets using the TFT.

**Checking the Expiration of the Token**

The system employs a process that checks the expiration time of the token to determine whether a received token has expired. This checking is important for detecting and preventing repeat-resending flooding attacks. If the attacker resends a packet with a token that has expired, the check alerts the system. The system reads the token’s generation time key $key_T$ and the token’s validation-duration key $key_D$. The system then finds the real values of these keys $T$ and $D$ using the $TokenKey_T^{-1}(key_T)$ and $TokenKey_D^{-1}(key_D)$ functions. After this, the system compares the duration-validation time with the result of the difference between the current time ($CT$) and the token’s generation time as follows:

\[
CT - T = \begin{cases} 
\leq D, & \text{if packet is not expired} \\
> D, & \text{if packet is expired} 
\end{cases}
\]  

(4.17)

If $CT - T \leq D$ then the packet has been received within the required time-frame. Otherwise, the packet has expired and is dropped.
**Checking the Token’s Delay Time**

In this check, the system determines whether the packet received by the system has abided by the system-specified delay conditions and arrived at or after the designated delay time. This process starts when the system reads the token’s delay-time key $key_v$. The system will then find the real value $V$ of this key using the $TokenKey_v^{-1}(key_v)$ function. If $V > 0$ then the packet is delayed, and if $V = 0$ then the packet is not delayed. In the second case, the system will not check the delay time of the packet. In the first case, if the packet was delayed, then the system compares the delay value with the current time $CT$ as follows:

$$TokenKey_v^{-1}(key_v) - CT = \begin{cases} 
\leq 0, & \text{if packet is received after the delay time} \\
> 0, & \text{if packet is received before the delay time}
\end{cases}$$ (4.18)

If $CT \leq V$ then the packet has been received after the delay time, is legitimate and will be accepted. Otherwise, the packet has been received before the delay time and will be dropped.

**Checking the Flag of the Token**

This check is important for identifying whether the received packet has been flagged (that is, marked). This check starts when the system reads the token’s flag key $key_f$. The system then finds the real value $F$ of this key using the $TokenKey_f^{-1}(key_f)$ function. If $F > 1$ then the packet is flagged and if $F = 1$
then the packet is not flagged. The decision the system makes in the case of flagged tokens depends on system policies and its attack detection-and-prevention strategies.

**Checking the Stage of the Token**

Checking the stage of the token is important as this check tells the filtering component the communication stage of the received packet. In addition, it is important to check whether the type of communication for which the packet is used is the same as the communication stage for which the token was designed. This check is important in order to prevent the re-use of tokens in different communications.

This check starts when the system reads the token’s stage key $key_M$. The system then finds the real value $M$ from this key using the $TokenKey^-1_M(key_M)$ function. After this, the system finds the real stage of communication $M'$ based on the packet’s fields. The system will then check the equalisation of these two values, as described in the following two equations:

\[
M' = TokenKey^-1_M(key_M) 
\]  \hspace{1cm} (4.19)

\[
M' = M, \text{if the stage is correct} 
\]  \hspace{1cm} (4.20)

The decision resulting from any disparity again depends on system policies and the attack detection-and-prevention strategies.
Checking the Validation of the Token’s Value

The previous token checks represent the initial stage of token checking, because the token might have been modified. To ensure the token has not been modified, we need to check the token’s value to ensure that the keys that were used to generate the token are the correct keys. First of all, the system uses the real value of the keys (that is, $T', D', F', V'$ and $M'$). The system then finds the client IP address $I'$ and the server IP address $S'$ based on the received packet header. After this, the system finds the three random-numbers addresses $RN'_1$, $RN'_2$ and $RN'_3$ based on the token’s generation time $T'$ using Equations (4.3) and (4.4). These parameters are used to find the expected token-value ($ETV$) result of the token’s formula as follows:

$$ETV = F(M', I', S', T', D', V', F', R'_1, R'_2, R'_3)$$  \hspace{1cm} (4.21)

The system checks whether the $ETV$ equals the $Token_{value}$ as follows:

$$ETV = Token_{value} \iff \text{token is correct}$$  \hspace{1cm} (4.22)

If the token is correct, this infers that the token has been generated for the received packet and has not been modified. On the other hand, if the token is not correct, the token has been changed or it has been used by an invalid client or packet; the packet will be dropped. Figure 4.4 below shows the token-verification process using a flowchart.
Figure 4.4 Token-Verification Flowchart
The sequence of the token-verification steps are changeable based on the system’s policies and the system’s rules. These can be modified if it is necessary to filter or detect attacks within a shorter time-frame based on the specific attack detection-and-prevention strategies.

Because malformed tokens can be detected, and also because clients use a unique token for each communication, a token can be used to detect repeat-resending DoS attack. The source and the token of communications can be stored in a watch list in order to block the sources of these attacks from communicating with the system. This function is discussed in the following section.

4.3 TFT Filtering Lists

In order to detect DoS repeat-resending attacks and to block the sources of DoS attack, the TFT uses two lists: a Blocked-Communication-List and a Monitoring-Repeated-Communication-List. The functionality of these two lists is discussed in the following sections.

4.3.1 The Blocked-Communication-List

The Blocked-Communication-List is placed at the front of the packet-filtering component. It represents the first layer of the packet-filtering layers. It records all the sources that the system has decided to block for a time because of their involvement in DoS attacks (Table 4.3). This list comprises two columns: IP
Address and Time. The IP-Address column lists the IP addresses of all blocked sources. The time column shows the duration of the block. The length of this time might differ between blocked sources. Once the time expires, the blocked source is removed from the Blocked-Communication-List.

Table 4.3 Example of a Blocked-Communication-List

<table>
<thead>
<tr>
<th>IP Address</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.10.10.10</td>
<td>12.05.01.15:20</td>
</tr>
<tr>
<td>20.20.20.20</td>
<td>12.01.01.01:35</td>
</tr>
<tr>
<td>30.30.30.30</td>
<td>12.04.29.23:09</td>
</tr>
<tr>
<td>........</td>
<td>........</td>
</tr>
<tr>
<td>40.40.40.40</td>
<td>12.03.03.13:59</td>
</tr>
</tbody>
</table>

4.3.2 The Monitoring-Repeated-Communication-List

The Monitoring-Repeated-Communication-List holds the IP addresses and the used tokens of a specified last number of communications (Table 4.4). This data is used when there is a need to determine whether the system is receiving a repeat flooding attack or to identify the source addresses involved in the attack. If the list is enabled, for each communication received, the system will search for a matching IP address in the list. If a match is made, then the system will report the match to the Analysis component. The system may also take additional action such as dropping the communication. If no match is found, the system will remove the oldest communication from the list and then insert the newer communication at the beginning of the list.
Table 4.4  Example of a Monitoring-Repeated-Communication-List

<table>
<thead>
<tr>
<th>IP Address</th>
<th>Token</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.10.10.10</td>
<td>Token1</td>
</tr>
<tr>
<td>20.20.20.20</td>
<td>Token2</td>
</tr>
<tr>
<td>30.30.30.30</td>
<td>Token3</td>
</tr>
<tr>
<td>..........</td>
<td>..........</td>
</tr>
<tr>
<td>40.40.40.40</td>
<td>TokenN</td>
</tr>
</tbody>
</table>

This list has three parameters: the size of the list (\( ListSize \)), the steps of the communication for which this list is enabled (\( EnableGroup \)) and the required decision (\( ReqDecision \)). The size of the list \( ListSize \) describes the number of entries that the list can hold. The \( EnableGroup \) parameters can be described as the following subset:

\[
EnableGroup \subseteq \{step_0, step_{i+1}, ..., step_n\} \quad (4.23)
\]

where

\( \{step_0, step_{i+1}, ..., step_n\} \) are the steps of communication.

From the \( EnableGroup \) definition we can see that the list can be enabled in one or more steps at a time. This helps the system to enable the list in critical steps of the communication. In addition, the system can also enable the list in any specific step in order to detect an attack being performed at that step.

The required decision \( ReqDecision \) is the decision that the system performs when the received communication exists on the list. It can be defined as follows:
\[ \text{ReqDecision} = \begin{cases} \{\text{drop}, \text{block}, \ldots\}, & \text{if enabled in Step}_x \\ \{\text{decision}_1, \ldots, \text{decision}_n\}, & \text{if enabled in Step}_y \end{cases} \] (4.24)

where \(x\) and \(y\) describe the step numbers of the protocol. The protocol steps are defined in Section 4.4.

This \text{ReqDecision} is updated by the Avoidance-Strategy-Manager component. The \text{ReqDecision} can have more than one decision. Moreover, it might have different decisions for different types of communications. The types of decisions are discussed in Section 4.5.1.2.

We recommend that the list be built using the linked-list data structure (Figure 4.5). Newer communications are inserted at the top of the list and older communications are deleted from the tail of the list. This saves time in maintaining the list.

The system follows the flowchart in Figure 4.6 to check whether a received communication is on the list. In the flowchart, checking for the existence of the received packet in the list starts by determining whether the list is enabled in the step to which the received packet belongs. In addition, the size of the list must be checked. If the size of the list is zero then the check should not be performed because no entries are on the list. Otherwise, the system will scan the list in order to find any entry equal to the received packet’s token and source. If this happens, the system moves this entry to the top of the list, reports the entry to the Analysis component and then makes decisions (if these have been defined). If no entry is found, then the received packet’s IP address
and token will be inserted at the top of the list. If the list is full then the oldest element in the linked list will be deleted after any new insertion.

Figure 4.5 (a) Example of Implementing a Monitoring-Repeated-Communication-List (b) Receiving Existed Communication (c) Receiving Unexcited Communication and the List is Not Full (d) Receiving Unexcited Communication and the List is Full

In this section we have discussed the two lists designed to support the TFT in filtering, detecting and preventing DoS attack. The following section provides a demonstration of the system-communication protocol that we have designed based on the TFT.
Figure 4.6 Checking the Existence of Received Communication in the Monitoring-Repeated-Communication-List
4.4 The Token-Filtering-Technique Protocol (TFTP)

The Token-Filtering-Technique Protocol (TFTP) is a new protocol designed to protect critical government services against DoS attack. It is designed for the HASP-CGeSS and based on the TFT. Tokens are therefore a compulsory part of its messages. This inclusion gives the system the following features.

i. The TFTP provides a larger number of attributes about the communication and the client. These attributes can be used to improve the system’s decisions in filtering, detecting and preventing DoS attack.

ii. The TFTP provides a stateless filtering technique by filtering flooding-attack packets based on their headers, which contain the tokens.

iii. The TFTP gives the system the ability to detect expired communication without holding any information about the client or performing any kind of encryption. This feature enhances system performance as it ensures that flooding-attack packets, which come from repeat-resending communications, are rejected, with little processing on the part of the server.

iv. The TFTP gives the system the ability to control the communication between the client and system by delaying the communication from a specific client or group of clients, or by delaying a specific type of
communication. This is achieved by using the delay key of the TFT. When
the system receives a high number of communications that need to be
organized, the system can place the new communications on hold until the
system has time to manage the new arrivals. Controlling the
communication timing enables the system to organize communications
without resorting to rejecting communication due to overloading. This
helps the system to prevent flooding DoS attack.

v. The TFTP enables the system to handle a larger number of clients. Rather
than the system holding information relating to clients, all the required
information is provided in the token, which is given to the client and
contained in the client’s next packet. Without this data-storage requirement
and burden on memory size, the system does not have a memory limitation
that restricts the system to communicating with only a limited number of
clients. This feature guards the system against SYN attack.

vi. The TFTP enables a diagnostic indicator in the system to highlight changes
in the received packet header. Clients can only make limited changes to the
packet header. The diagnostic indicator reduces the likelihood of the
system experiencing malformed packet attack.

vii. The TFTP ensures the sequential order of a client’s communications.
Neither the client nor the attacker can start communicating with the system
from a middle step of the communication process. This reduces the
likelihood of flooding attacks that target certain steps of the communication process.

viii. The TFTP prevents spoofing attacks because the attacker needs to have a token that is given by the system in each step of communication. The token also represents the client’s IP address.

The TFTP divides the communication between the client and system into three stages (Figure 4.7): (i) the connection stage; (ii) the authentication stage; and (iii) the service-communication stage. These stages are sequentially performed online by the server and the client. However, while each stage may use a different protocol standard, the token must pass between the stages. Before these three communications stages, an initial offline stage is required. These four stages and their messages are discussed in the following four sections.

4.4.1 The Initial Stage

The initial stage is a required stage for the client before starting communication with the system in order to gain access to fully authenticated services. This stage has one step of protocol to establish the client’s unique identification: the provision of a smart card. This smart card is issued by the system. It contains the client’s public key as a unique identifier for the client. This key is used in both authentication and encryption activities. This step of the protocol is shown as follows:

$$S \rightarrow C: \{ \text{PrivKey}_c, \text{PubKey}_c, \text{Sig}_{\text{key}_s}(\text{Cert}_c) \} \quad (4.25)$$
In the above step, the system provides a smart card that contains unique pairs of public and private keys and a certificate that is signed by the system’s private key. The smartcard is protected by a PIN that is only known by the client. This step of the protocol is performed offline. Once the client decides to use the fully authenticated eService, possession of a smartcard that contains these three pieces of data is essential.
4.4.2 The Communication Stage

In this stage, the client can start the communication with the system and perform the three-way handshake. This stage is performed through the Filter-and-Redirect (FR) Engine in the CA component of the HASP-CGESS. It starts when the client performs the following first step:

\[ \text{Step 1: } C \rightarrow S: ConnReq_{C-S} \]  

(4.26)

In this step, the client makes a request to communicate with the system by sending a request connection to the server. At this stage, the client does not provide any information to the server other than the connection request. The server does not have any information about this client except the IP address shown in the client’s packet header. The client source IP address might be spoofed, in which case the client is an attacker. In either case, the system will process this message by using the following algorithm:

Start Step1
Read ClientIP I;
If(C in BlockedList) then
    {Drop packet;
     ReportAnalysis(I,"Blocked Source");
     Exit;
    }
If(Request Form is not CORRECT) then
    {Drop packet;
     ReportAnalysis(I,"Incorrect Request Form");
     Exit;
    }
If(RepeatList is ENABLED on Step1 AND C in RepeatList) then
    {ReportAnalysis(I, "Source in Watchlist"); perform
     ReqDecisions(Step1);
    }
End Step1
In the above algorithm, we can see that the system first finds the source of the request. Then it checks whether the source exists in the Blocked-Communication-List. If it is found in the list, this infers that the system has earlier decided not to receive communications from this source. The system will report to the Analysis component that it has received a communication from a blocked source. In this case, the system will stop running the algorithm for this message from the client. On the other hand, if the source was not blocked, the system will move to the next statement of the algorithm. The next statement checks the correctness of the request based on the protocol form. If a request is in an invalid form the system will drop the request and report this to the Analysis component. If the request was in the correct form, the system will check whether the repeat-communication list, RepeatList, is enabled in this step and whether the source exists on this list. If so, the system will report this to the Analysis component and continue running the algorithm to perform the designated action. These actions reflect previous decisions relating to the detection of repeated communication on the Monitoring-Repeated-Communication-List. If the decision tells the system to drop the packet, then the packet is dropped and the system stops running the algorithm. Otherwise, if the communication is not on the list or a stopping decision is not assigned, then the system accepts the request and moves to the next step.

Step 2 of the protocol is responsible for replying to this initial request, as follows:
In Step 2, the system sends a reply to the client-request message. This message includes the type of the message *Replay*, a token *Token*¹, and the delaying time *DelayComm*. *DelayComm* is the period of time that the client must wait before sending a return message. To generate this message, the system uses the following algorithm:

\[
\text{Start Step2}
\]

\[
\text{Read ServerIP } S; \quad \text{// *starting of token generation}
\]

\[
M = \text{StateTable(State="Established")};
\]

\[
D = \text{AcceptedTimeDuration(S,C,Step3)};
\]

\[
V = \text{DelayTime(S,C,Step3)};
\]

\[
F = \text{FindFlagValue(S,C,Step3)};
\]

\[
R1 = \text{RandomNoTable(R1,T)};
\]

\[
R2 = \text{RandomNoTable(R2,T)};
\]

\[
R3 = \text{RandomNoTable(R3,T)};
\]

\[
\text{Token1(Value) = } F(M,I,S,T,D,V,F,R1,R2,R3S)
\]

\[
\text{KeyT= TokenKeyT(T);}
\]

\[
\text{KeyM= TokenKeyM(M);}
\]

\[
\text{KeyD= TokenKeyD(D);}
\]

\[
\text{KeyV= TokenKeyV(V);}
\]

\[
\text{KeyF= TokenKeyF(F);}
\]

\[
\text{Token1(Keys) = \{KeyT,KeyM,KeyV,KeyF,KeyD\};}
\]

\[
\text{//ending of token generation}
\]

\[
\text{Message=GenerateMessage{Reply,S,I ||Token1,Order to Delay for D time}};
\]

\[
\text{Send Message;}
\]

\[
\text{End Step2;}
\]

As we can see from the algorithm above, the system first generates a token, *Token*¹, which is based on *I*, *S*, *T*, *M*, *D*, *F*, *V*, *RN₁*, *RN₂*, *RN₃*. The values of *D*, *F* and *V* are determined based on the Avoidance-Strategy-Manager component’s prevention strategies. The client is not required to read the content of *Token*¹. At this stage the system does not hold or store any data.
belonging to the client and so no communication backlog queue is used. This helps to avoid the TCP-SYN flooding attack as the system can engage with an unlimited number of clients. All the required communication data are stored in the token that is given to the client. However, the client cannot change these data, because the client cannot understand the information in the token, and any change in the token would give incorrect results in the token’s value. 

DelayComm is the message from the system to the client that instructs the client to wait before sending the next message. This delay time is represented in Token\(^1\). In normal cases the delay time is zero, so the client can continue communicating with the system without hesitation.

The client, after sending the Step 1 request message, waits until he or she receives the Step 2 message. Once this message is received, the client waits for the period of time specified in DelayComm. When this period of time is expired, or when it is zero, the client moves to the next step of communication:

\[
\text{Step 3: } C \rightarrow S: \text{Ack,Token}^1
\]  

In this step, the client sends the ACK message, which is the last step of the three-way handshake process. In this message, the client must provide the Token\(^1\) that was received in the Step 2 message. This message must be sent after the DelayComm time ordered in the Step 2 message. The system will use the following algorithm to process this message:
Start Step3
Read ClientIP I;  //*starting of token checking
If( C in BlockedList) then
{Drop packet;
 ReportAnalysis(I,"Blocked Source",Step3);
 Exit;}
If(C in RepeatList) then
 {ReportAnalysis(I," Source in RepeatList",Step3);
  perform ReqDecisions(Step1);}
If(Token is not included) then
{Drop packet;
 ReportAnalysis(I,"excluded Token",Step3);
 Exit;}
Read Token Token1
M' = Token1.Token(keyValueM(KeyM))
If(M'!= 3) then
{Drop packet;
 ReportAnalysis(I,"Incorrect State");
 Exit;}
Read Current Time T;
V' = Token1.Token(keyValueV(KeyV))
If(V!=0 AND V'> T) then
{Drop packet;
 ReportAnalysis(I,"Come Before Delay Time");
 Exit;}
D' = Token1.Token(keyValueD(KeyD))
T' = Token1.Token(keyValueT(KeyT))
If((V!=0 AND V'+D'<T)OR (V==0 AND T'+D<T) then
{Drop packet;
 ReportAnalysis(I,"Expired Token",Step3);
 Exit;}
F' = Token1.Token(keyValueF(KeyF))
If((F'!=0) then
  {ReportAnalysis(I,"Flag F'",Step3);
 Perform Required Decisions}
R1' = RandomNoTable(R1,T');
R2' = RandomNoTable(R2,T');
R3' = RandomNoTable(R3,T');
Token' = F(M',I,S,T',D',V',F',R1',R2',R3')
If((Token'!=Token1.Token(Value)) then
{Drop packet;
 ReportAnalysis(I,"Invalid Token",Step3);
 Exit;}
else Token is valid and accepted;
//end of token checking
End Step3
As can be seen in the algorithm above, the system performs three levels of packet checking. The server first reads the source IP address of the packet. The system then checks whether this source is blocked from communicating with the system. If it is blocked, then the server will drop the packet and report this to the Analysis component. Otherwise, the system will check whether the repeat communication list, RepeatList, is enabled in this step and if the source is on this list. If so, the system will report this to the Analysis component and continue the algorithm and perform the required decisions. Otherwise, the system will move to the next step of the algorithm.

After the source is checked against the Blocked-Communication-List and the Monitoring-Repeated-Communication-List, the server will perform token checking. This starts when the server checks whether the packet has a token. If the token does not exist in the packet, the packet will be dropped and this action will be reported to the Analysis component, because the packet must have a token in this step of the communication. If the packet has a token, then the system will find the state value from the token-state key, and check if this token is designed for this step of the communication. If it is not, then the packet will be dropped and this action will be reported to the Analysis component. If the token is designed for this step, then the system will find the value of the communication delay, using the token’s delay key. If the communication is delayed and the communication is received before this delay time, the packet is dropped and this action is reported to the Analysis component. Otherwise, the system will check whether the packet has expired.
If it has expired then the packet will be dropped and this action will be reported to the Analysis component. However, depending on the system policies, a new token can be sent to the client with a request to resend the packet. If the packet is sent within time, the system checks the validity of the token using the client’s IP address, the server’s IP address, the calculated values of the received token’s keys and the three random numbers that are provided based on the received token’s generation time. The result of this calculation is compared to the received token’s value. If it is not equal then the token is not correct, and the token is dropped and reported to the Analysis component. Otherwise, the token is accepted. This is the last step of token checking.

At this stage, the server has accepted the communication and the three-way handshake is complete. The protocol has finished the first stage of the communication with the client. Communication between the client and the system can now move to the authentication stage.

4.4.3 The Authentication Stage

During the authentication stage the server authenticates clients. This authentication is performed using a client’s signature. This stage consists of three steps. In these steps the server asks the client to provide his or her signature. This stage is the most important stage because it requires a higher processing cost from the server. The token can help to minimize this cost for communication arising as a result of DoS attack.
Step 4: $S \rightarrow C$

\[
: \text{ProvCertReq}_C, \text{Cert}_S, \text{ProvRN1Req}_C, \text{Token}^2, \text{DelayComm}
\]

(4.29)

This message is sent by the server to the client. In this message, the server asks the client to provide his or her certificate so that the server can authenticate the client. In addition, the server asks the client to provide a random number. This random number is used as part of the shared key that is used to encrypt communications. The server also gives its certificate to the client. This is so the client can verify that the communication comes from the server. The client uses the server’s public key to encrypt his or her certificate and the required random number before sending them to the server. This protects the client’s signature from sniffing by attackers. The signature is the real identity of the client. It is encrypted and stored in a smartcard, and it can be retrieved by using a password. In this message also, the server gives the client a new token, $Token^2$, that must be included in the client’s next communication. In addition, the server gives the client the required delay time for the next communication. When the client receives this message, the client will perform the following step:

Step 5: $C \rightarrow S: \{\text{Cert}_C, \text{RN}_C\}_{pkey_S}, \text{Token}^2$

(3.30)

This message is sent by the client to the server. The client sends his or her certificate and the required random number. This information is encrypted by
the server’s public key. In addition, the client includes $Token^2$, which has been received in the previous communication, in the packet header.

This message is the most important for the server because it incurs a high processing cost. The processing cost is due to the encryption task. This step could therefore be used to commit a DoS attack. The server filters the message based on the Blocked-Communication-List, the Monitoring-Repeated-Communication-List and the token, before encrypting the client’s certificate and the random number. Because this step is costly, the Monitoring-Repeated-Communication-List should be enabled in this step as a default setting in order to protect the system from low-cost DoS attack, and to detect the initial stages of any potential attack that occurs in this step.

Once the packet has passed the three levels of packet checking, the encrypted part of the packet will be decrypted and the system will authenticate the client. After this the system will check for the existence of the received signature in the signature black list, which is in the FR Engine. If the signature does not exist in the black list, the FR Engine will send this signature to the Ticket Engine for authentication. If the system finds that this certificate has been revoked, then the client will be asked to renew his or her signature and restart the communication from Step 1 at a later time. In addition, this signature will be added to the signature black list. On the other hand, if the client is authenticated to the system, then the communication will move to the next step as follows:
Step 6: $S \rightarrow C: \{Token^3, \quad RN_s, \quad NextIP\}_{pkey_c} \quad (3.31)$

This message is sent by the server to the client. It is the last step of the authentication stage. This step also incurs a high processing cost by the server. However, Step 5 is more critical to the system than Step 6, because Step 5 incurs a higher processing cost than Step 6 and the execution of Step 6 depends on the result of the execution of Step 5. In the message in Step 6 the server gives the client three items: (i) a random number, which is the second part of the shared key; (ii) the next IP address through which the client can continue communicating with the system; and (iii) a new token, $Token^3$, that must be included in the client’s next packet header. Three items are all encrypted by the client’s public key. The next IP address is located in the Authenticated-Client Engine in the Authenticated-Client-Communication component and is assigned by the Next-IP Engine. Once the current stage is complete, all the client’s ensuing communications are through the Authenticated-Client Component. Both the client and the server use one shared key, $SK_{s-c}$, to encrypt traffic. This key is generated based on a combination of the client’s and the server’s random numbers. This key is the result of the following hash function:

$$SK_{s-c} = h(RN_s, RN_c) \quad (3.32)$$

where $RN_s$ is the random number that is generated by the system and $RN_c$ is the random number generated by the client. Both of these numbers are used to generate the shared key.
4.4.4 The Communicating Stage

By this third stage of the interaction between the client and the system, all clients are authenticated. Clients can therefore access the required eServices. The first message of this stage is as follows:

\[
\text{Step 7: } C \rightarrow S: \{\text{data, PacketHeader }\}_{SK_{S-C}}, \text{Token}^3
\]

This message is sent by the client to the server. It consists of two parts. The first part contains the data. This data describes all the information that the client includes regarding the required service. This data is encrypted by the shared key $SK_{S-C}$. The second part of the message is the token, which has been given by the server in the previous message. The token must be included in the packet header and be unencrypted.

The token is used to perform the first layer of packet filtering (previously described). The second layer involves checking whether the received communication arrived at the system through the expected IP address. This checking arises as a result of the Dynamic Multi-Point-Communication technique implemented in this area of communication. If the communication did arrive as expected, after the communication has passed through these two layers of filtering, the system will then decrypt the encrypted part of the communication. The encrypted part contains the packet header. This encrypted packet header is compared to the received packet header. This comparison protects the system from repeat attacks, in which the attacker uses an encrypted packet that does not belong to the current message. After the message is
accepted, the communication will move to the next step of the protocol as follows:

\[
\text{Step 8: } S \rightarrow C: \{\text{Token}^4, \text{NextIP}, \text{Data}\}_{SK_{S-C}} \quad (4.34)
\]

The message in Step 8 is sent by the server to the client. It contains three parts: (i) the data regarding the required service; (ii) the IP address of the system where the client’s next communication must arrive; and (iii) a new token, which must be included in the client’s next communication. These three parts are encrypted by the shared key.

The communication between the system and the client is continued by using Step 7 and Step 8. Once the client has finished with his or her transaction on the system, the client can use the following final step to end the communication:

\[
\text{Step 9: } C \rightarrow S: \{\text{end} \}_{SK_{S-C}}, \text{Token}^x \quad (3.35)
\]

The ending request must be included in the encrypted part of the communication to avoid an attacker stopping the communication.

In this section, we have described the TFTP, which is designed to protect a system against DoS attack. The system’s processes for filtering, detecting and preventing DoS attacks in this protocol are discussed in the following section.
4.5 DoS-Attack Filtering, Detection and Prevention Processes

In the previous section (Section 4.4), we discussed the communication protocol between the server and the client. In this section, we describe the system’s processes. The HASP-CGESS processes can be divided into three main processes: the filtering process, the detection process and the prevention process (Figure 4.8).

![Diagram of Filtering, Detection, and Prevention Processes]

**Figure 4.8 The Integration of Filtering, Detection and Prevention Processes**

The Prevention-Process performs packet filtering based on the filtering rules. These rules can be updated by the Prevention-Process, if this is required in
order to prevent or verify an attack. The Detection-Process tries to identify the
nature of the received attack and its source. It uses data provided by the
Filtering-Process and moves the detection result to the Prevention-Process. The
Prevention-Process makes decisions that protect the system from DoS attack
and implements these decisions in the Filtering-Process.

In the following sections we describe these three processes.

4.5.1 The Filtering-Process

The Filtering-Process can be described as an eligibility inspection of the
received messages followed by a decision that either allows or denies system
processing of the messages. The most important task in the Filtering-Process is
the message inspection. Both the inspection and system decisions are
performed based on filtering instructions (that is, the rules). The TFT facilitates
decision making and decision accuracy because it adds a token to the packet
header. The token gives the Filtering-Process additional reliability by providing
it with extra parameters to check.

The Filtering-Process in the HASP-CGESS performs two main tasks: (i)
inspecting the received packet based on the parameters; and (ii) assigning
values to the parameters of the packet that will be sent to the client. The
following sections discuss these two tasks.

4.5.1.1 Filtering-Process Inspections

The TFT can include the following inspection parameters in the filtering rules.
Expired Packets

The TFT facilitates the identification of expired packets because an expired packet is clearly defined in the token’s validation key $key_D$. Furthermore, the Filtering-Process can establish the time that has lapsed since the packet’s expiration. This gives the Filtering-Process the ability to distinguish between expired packets. With this inspection the following example rule can be implemented:

#If the packet is expired for less than 1 minute then
Renew and send a new token

As we can see from the above rule, by using the TFT the system can establish exactly when and how long ago the communication expired. In the above rule we have nominated one minute for time expiration. The expiration time period and the required decision are designated based on the system’s security policies for the detection and prevention of DoS attack.

The Packet-Generation Time

The Filtering-Process knows exactly when a packet is generated because of the token’s generation key $key_r$. The system can therefore filter packets based on the packet-generation time as follows:

#If the packet was generated between time $T_1$ and time $T_2$ then
assign a flag $f$ for this communication

In this rule the system needs to distinguish packets that are generated between times $T_1$ and $T_2$. The system may be suspicious of these communications and
want to follow them in order to detect an attack. This is organized by giving the communications a mark that assigns a value in the flag parameter on their tokens.

**The Type of Packet**

The system can identify the packet’s type without reading or analyzing the entire packet. Instead, the identification can be performed using the token’s stage key $key_m$. Thus, we can say the following:

```markdown
#If the packet belongs to authentication stage step 4 then
Delay the communication for a time T
```

In this example, the system decides to delay packets that were going to be processed in Step 4 of the authentication stage. The system may want to save processing resources for a period of time in order to detect or manage an attack elsewhere in the system.

**Marked Packets**

By using flag key $key_f$, we can recognize packets that have been marked by the system. The Filtering-Process can then find these packets in the received packet stream as follows:

```markdown
#If the packet has F marked and has expired for more than 10 minutes then
Block this source for 30 minutes
```

In this example, the system decides to block communication with the system for 30 minutes. These communications were long expired (that is, for more
than 10 minutes) and so could be distinguished from other communications by the F value in the flag parameter on their tokens.

**Malformed Packets**

The token gives diagnostic information about packet fields. With this information, the Filtering-Process knows if an attacker has changed any field or spoofed a packet. Using the TFT we can write the following:

```
#If the packet is malformed or spoofed then
Block the source for 1 day
```

In this example, the system decides to block communication with the system for one day. The invalid token indicated that the packets were either malformed or spoofed.

**Delayed Packets**

Because the token’s delay key $key_d$ indicates whether the packet has been delayed, we can find packets that were sent before the delay time expired, as follows:

```
#If the packet was received before the delay time then
Block the source for 1 day
```

In this example, the system decides to block communication with the system for one day. The token indicated that the communications were sent before the delay time and were thus illegitimate.
Repeated Packets

By including the token in the Monitoring-Repeated-Communication-List, we can learn whether the packet of a source has been repeated, because the list checking is based on the source and the token. A legitimate client might have sent different packets each time (and these packets are held in the list), but each packet represents a different stage of the communication. Even if the client is disconnected in the middle of a communication, the communication can be restarted. The Filtering-Process can distinguish communications coming from a sole genuine client, because the tokens will be different, having been generated at different times. So we can conveniently write the following role:

#If the packet or the message is repeated then
Delay the next communication for 10 minutes

In this example, the system decides to delay the next communication for ten minutes for any client who has sent a repeat packet to the system.

4.5.1.2 Filtering-Process Decision Making

Subsequent to performing inspections, the Filtering-Process can make a number of decisions, as follows.

Dropping Packets

Once the Filtering-Process determines that a packet that should not gain access to a system, it has the ability to drop the packet as in the following rule:

#Drop this packet
**Marking Packets**

The system has the ability to mark packets using the token’s flag key $key_f$ as in the following rule:

#Give this packet an F mark

**Delaying Packets**

The system has the ability to delay a specific packet or a specific group of packets using the token’s delay key $key_d$. The system can thus make the following decision, where the $Tx$ value is the delay time:

#Delay the packet by the time $Tx$

**Minimizing a Packet’s Validation Time**

Using the token’s validation key $key_v$, the system has the ability to minimize the packet’s validation time by generating a token that has a short expiry time. The system can then include the following rule:

#Make the expiry time shorter than 1 minute

We have used one minute in the rule above as an example of a reduced expiration time. This time is determined according to the system’s security policies for the detection and prevention of DoS attack.

**Reporting to the Detection-Process**

In some cases, the Detection-Process needs to know specific data resulting from some aspect of the communication. The Filtering-Process might include the following rule:
#Report the packet’s result and information to the Analysis Component

As we have now demonstrated the Filtering-Process, the following section discusses the second type of system process: the Detection-Process.

### 4.5.2 The Detection-Process

The attack Detection-Process monitors the system’s resources and analyzes the reported communication. It investigates whether the system is receiving an attack and identifies and verifies the source of the attack. The Detection-Process builds its decisions based on reports about the system’s resources and the reported communication results. The reported communication results are grouped in a usable form by the Analysis component as a part of the attack Detection-Process. The attack Detection-Process makes its decision in the Attack-Recognition component. The following sections discuss the four different types of attack detection: (i) detecting whether the system is receiving an attack; (ii) detecting the step in which the attack is committed; (iii) finding the source of the attack; and (iv) defining the attack.

#### 4.5.2.1 Detecting Whether the System is Under DoS Attack

In this type of attack detection, the Detection-Process monitors the system’s resources in order to learn whether the system is receiving a DoS attack, or whether there is any suspicion of a DoS attack. Three parameters can be used
to check whether the system is receiving a DoS attack: system load, the number
of expired packets and the number of clients compared to the system load.

System Load
System load (SysLoad) represents the percentage of system-processing usage.
It has three states: Normal, Busy, and Overloaded (Figure 4.9).

Figure 4.9 System Load States
The state of the system’s load can be calculated using the following function:
/&6"8, 2D 1¯/"; ¥ 50%
" 1¯/"; ¶ G .71, 2D 50% ¥ 1¯/"; ¥ 75% §
B &8/"; ; , 2D 1¯/"; ¦ 75%

4.36

A 50% system load should be enough to serve the normal number of clients.
On the other hand, if the system load has a value greater than 50% and less
than 76%, then the system might be receiving a DoS attack. However, this load
could also be caused by an unexpected number of genuine clients
communicating with the system at the same time. In this case the system will
investigate whether this load due to a DoS attack or not. Because 25% of the
system load is allocated to detection and prevention tasks, when the system

179


load is more than 75%, the system will apply more restrictive filtering-and-detection policies.

**Number of Expired Packets (NEP)**

DoS attack can also be detected by considering the amount of dropped packets. The Number of Expired Packets, *NEP*, represents the number of packets that are dropped as a result of token expiry. A high NEP may indicate the occurrence of a flooding DoS attack. For example, when an attacker repetitively sends DoS packets to the system in order to commit a flooding attack, these packets will be accepted by the system as they have a valid token. Once the packets’ tokens expire, however, the remainder of these packets will be dropped and reported to the Detection-Process. The Detection-Process will identify the number of expired packets and determine whether the system might be receiving a DoS attack, taking into account the normal percentage rate of expired packets, or *NPEPR*. This rate is defined by a system administrator based on the speed of the communication and system-statistics reports, as follows:

\[
NPEPR = 5\% 
\]  

(4.37)

The *NEP* has three states: normal, suspicious and attack. These states are defined based on the percentage of the *NEP* from the number of all communications at a time, *NoC*. The following formula is an example of defining the state of the *NEP*:
DoS attack can also be detected by considering the number of clients compared to the system load \((SLvsNC)\), at a particular time. This comparison is used to determine whether the system load is caused by an expected number of clients or by some other reason. The system load is compared to the maximum number of clients that this load can serve at a time. The genuine client should perform a single step of the protocol at a time. However, each message of the protocol requires a specific processing cost from the system. The required system processing at Step 5 is the most costly because of the associated encryption and decryption tasks; consequently, the system load should peak at this stage of the protocol. As this should be the heaviest load under normal circumstances, we use the processing cost of Step 5 to calculate the system load based on the number of clients, \(SLvsNC\), using threshold-based detection methodology.

Let us say we have a server with a 3.0 GHz processor. The system speed is 3000000000 cycle/Sec. Based on the Wei Dai [145] cryptographic protocol benchmarks, Step 5 of the protocol requires processing of 2950 Kcycle from the server. The maximum system load is thus \(3000000000/2950000=1016\).
client/Sec. So at 50% of system load, the $SLvsNC=508$. Let us assume that the attacker has a maximum communication speed of 32 kb/Sec. The maximum throughput of the attacker in Step 5 is thus 26 packets/Sec. Figure 4.10 compares the results of system loads.

![Figure 4.10 Comparing the System Load with the Number of Clients](image)

In Figure 4.10 we can see that $SLvsNC=508$ when the system load is 50%. This implies that the system is not receiving a DoS attack, because this load is caused by a normal number of clients. On the other hand, we can see that the $SLvsNC≈379$ for the same system. This means that fewer clients are causing a more-than-normal system load. It is logical to conclude that some of these clients are attackers sending repeat DoS packets.
These three parameters can be used by the system to detect whether the system is under DoS attack, and is the first type of attack-detection process. The second type of process pinpoints the protocol step receiving a DoS attack.

4.5.2.2 Identifying the Protocol Step Receiving a DoS Attack

In this type of Detection-Process, the system will try to determine which protocol step is receiving a DoS attack. This detection is important because it focuses detection policies on the step in order to detect the attack and identify its source. The following sections provide two parameters that can be used to discover the step in which an attack is received.

Number of Expired Packets in a Step (NEPS)

The Number of Expired Packets in a Step, NEPS, is the same as NEP but it is based on a specific step of the protocol. It represents the number of packets that are dropped in a specific step of the protocol due to expiration. NEPS can determine whether a step might be receiving a DoS attack, taking into account the normal percentage rate of expired packets in a step, NPEPSR. This rate is defined by a system administrator based on the speed of the communication and system statistic reports, as follows:

\[ \text{NPEPSR} = 5\% \]  

\[ (4.39) \]

NEPS has three states: normal, suspicious and attack. These states are defined based on the percentage of the NEPS from the number of all communication at
a time, $NoCS$. The following formula is an example of finding the state of the number of expired packets in the step $x$:

$$StateOfNEPS(Step_x) :$$

$$= \begin{cases} 
\text{Normal}, & \text{if } \left( \frac{NEPS(Step_x)}{NoCS(Step_x)} \* 100 \right) - NPEPSR \leq 0\% \\
\text{Suspicious}, & \text{if } 0\% \leq \left( \frac{NEPS(Step_x)}{NoCS(Step_x)} \* 100 \right) - NPEPSR \leq 10\% \ (4.40) \\
\text{Attack}, & \text{if } \left( \frac{NEPS(Step_x)}{NoCS(Step_x)} \* 100 \right) - NPEPSR > 10\% 
\end{cases}$$

where $NoCS(Step_x)$ represents the number of all communications at a time in step $x$ of the protocol.

If there is an alert that the system is receiving a DoS attack, the communication step that has the highest percentage of expired packets is most likely the target of the attack.

**The Monitoring-Protocol-Behaviour Technique**

The purpose of the Monitoring-Protocol-Behaviour Technique is to identify the protocol step that is the recipient of a flooding DoS attack. The technique monitors protocol behaviour during clients’ communication with the system in order to detect repeat-resending flooding attacks. It is designed based on the following formula:

$$TIC(T_i, T_{i+1}) \geq PerDC + \sum_{x=1}^{3} step_x(T_i, T_{i+1}) - exit_x(T_i, T_{i+1}) \quad (4.41)$$
where

- $TIC(T_i, T_{i+1})$ is the Token-Issue Counter that describes how many tokens the system has issued in the period of time between $T_i$ and $T_{i+1}$;
- $PerDC$ represents the percentage of normal communication disconnections;
- $step_x(T_i, T_{i+1})$ represents the number of communications that have been processed in step $x$ in the period of time between $T_i$ and $T_{i+1}$; and
- $exit_x(T_i, T_{i+1})$ represents the number of communications that have moved from step $x$ to the next step in the period of time between $T_i$ and $T_{i+1}$.

Once this technique is implemented, a negative value in any one of the steps’ counters indicates that the following step is receiving a repeat flooding DoS attack. Table 4.5 shows the test results of the Monitoring-Protocol-Behaviour Technique.

The row T describes the time period and the row TIC describes the total number of clients that are communicating with the system during time T. So in time 3-4 of Table 4.5 (a), the total number of clients communicating with the system is 1536 and the number of clients communicating with the system in Step 2 is 512.
Table 4.5 Result of Testing the Monitoring-Protocol-Behaviour Technique for: (a) No Attack Received (b) Attack Received in Step 1 (c) Attack Received in Step 2 (d) Attack Received in Step 3

(a)

<table>
<thead>
<tr>
<th>T</th>
<th>0-1</th>
<th>1-2</th>
<th>2-3</th>
<th>3-4</th>
<th>4-5</th>
<th>5-6</th>
<th>6-7</th>
<th>7-8</th>
<th>8-9</th>
<th>9-10</th>
<th>10-11</th>
<th>11-12</th>
<th>12-13</th>
<th>13-14</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIC</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
</tr>
<tr>
<td>Step 1</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
</tr>
<tr>
<td>Step 2</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
</tr>
<tr>
<td>Step 3</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
</tr>
</tbody>
</table>

(b)

<table>
<thead>
<tr>
<th>T</th>
<th>0-1</th>
<th>1-2</th>
<th>2-3</th>
<th>3-4</th>
<th>4-5</th>
<th>5-6</th>
<th>6-7</th>
<th>7-8</th>
<th>8-9</th>
<th>9-10</th>
<th>10-11</th>
<th>11-12</th>
<th>12-13</th>
<th>13-14</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIC</td>
<td>1664</td>
<td>1792</td>
<td>1920</td>
<td>2048</td>
<td>2176</td>
<td>2304</td>
<td>2432</td>
<td>2560</td>
<td>2688</td>
<td>2816</td>
<td>2944</td>
<td>3072</td>
<td>3200</td>
<td>3328</td>
</tr>
<tr>
<td>Step 1</td>
<td>640</td>
<td>768</td>
<td>896</td>
<td>1024</td>
<td>1152</td>
<td>1280</td>
<td>1408</td>
<td>1536</td>
<td>1664</td>
<td>1792</td>
<td>1920</td>
<td>2048</td>
<td>2176</td>
<td>2304</td>
</tr>
<tr>
<td>Step 2</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
</tr>
<tr>
<td>Step 3</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
</tr>
</tbody>
</table>

(c)

<table>
<thead>
<tr>
<th>T</th>
<th>0-1</th>
<th>1-2</th>
<th>2-3</th>
<th>3-4</th>
<th>4-5</th>
<th>5-6</th>
<th>6-7</th>
<th>7-8</th>
<th>8-9</th>
<th>9-10</th>
<th>10-11</th>
<th>11-12</th>
<th>12-13</th>
<th>13-14</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIC</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
</tr>
<tr>
<td>Step 1</td>
<td>512</td>
<td>384</td>
<td>256</td>
<td>128</td>
<td>0</td>
<td>-128</td>
<td>-256</td>
<td>-384</td>
<td>-512</td>
<td>-640</td>
<td>-768</td>
<td>-896</td>
<td>-1024</td>
<td>-1152</td>
</tr>
<tr>
<td>Step 2</td>
<td>512</td>
<td>640</td>
<td>768</td>
<td>896</td>
<td>1024</td>
<td>1152</td>
<td>1280</td>
<td>1408</td>
<td>1536</td>
<td>1664</td>
<td>1792</td>
<td>1920</td>
<td>2048</td>
<td>2176</td>
</tr>
<tr>
<td>Step 3</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
</tr>
</tbody>
</table>

(d)

<table>
<thead>
<tr>
<th>T</th>
<th>0-1</th>
<th>1-2</th>
<th>2-3</th>
<th>3-4</th>
<th>4-5</th>
<th>5-6</th>
<th>6-7</th>
<th>7-8</th>
<th>8-9</th>
<th>9-10</th>
<th>10-11</th>
<th>11-12</th>
<th>12-13</th>
<th>13-14</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIC</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
</tr>
<tr>
<td>Step 1</td>
<td>512</td>
<td>512</td>
<td>384</td>
<td>256</td>
<td>128</td>
<td>0</td>
<td>-128</td>
<td>-256</td>
<td>-384</td>
<td>-512</td>
<td>-640</td>
<td>-768</td>
<td>-896</td>
<td>-1024</td>
</tr>
<tr>
<td>Step 2</td>
<td>512</td>
<td>512</td>
<td>640</td>
<td>768</td>
<td>896</td>
<td>1024</td>
<td>1152</td>
<td>1280</td>
<td>1408</td>
<td>1536</td>
<td>1664</td>
<td>1792</td>
<td>1920</td>
<td>2048</td>
</tr>
<tr>
<td>Step 3</td>
<td>512</td>
<td>512</td>
<td>640</td>
<td>768</td>
<td>896</td>
<td>1024</td>
<td>1152</td>
<td>1280</td>
<td>1408</td>
<td>1536</td>
<td>1664</td>
<td>1792</td>
<td>1920</td>
<td>2048</td>
</tr>
</tbody>
</table>
In this test we implemented a simulation for the Monitoring-Protocol-Behaviour Technique to check whether this technique can detect the location of the received repeat attacks. We designed four scenarios. The first scenario describes normal communication; the system does not receive any attack. In the remaining three scenarios the system receives a flooding repeat attack in a particular step of the protocol. Table 4.5 shows the results of these four scenarios.

The results in Table 5.8 (a) show no negative values. Nor do the results in (b) show any negative values, even though the system received an attack. However, while this technique does not show negative results in the first step of the protocol, we can see that a high number of communications were received in this step, and became progressively higher in each subsequent step, whereas the communication numbers in other steps were more constant. On the other hand, results from (c) and (d) for the first and second steps show negative values that indicate that the second and the third steps respectively received attacks.

The next section discusses the system’s detection technique for recognising the source of a DoS attack.

**4.5.2.3 Detecting the Source of the Attack**

A system can detect the source of an attack by using the parameter that counts the number of repeated communications performed by a client. If the
Monitoring-Repeated-Communication-List is enabled in a step of the protocol, and a received communication is in the list, the system will record the number of times the communication has been received and report this to the Analysis component. The Analysis component groups these repeated communications based on their source. The Detection-Process will then make a decision to block (for example) any communication from this source.

4.5.2.4 Defining the Attack

Based on the three types of detection, the system can design the attacks’ signatures in order to define them. These definitions are then moved to the Prevention-Process where a decision is made about a detected attack. In addition, the Detection-Process can define suspicious communication signatures. So if the system detects any one of these suspicious communications, this communication is moved to the Prevention-Process in order to take the required decisions that verify whether the communication is an attack. The following is an example of a Detection-Process code:

If ((SysLoad ≥ 50%) AND (stateOfNOEP(step x) = "suspicion") then {suspicion-attack(flooding, Step x, medium)}
If ((SysLoad ≥ 75%) AND (Monitoring_protocol_Behaviour has<0 values in step x ) AND (Client C has>1 repeated communication)) then { attack(flooding, source C, high)}

This section demonstrated how a system detects DoS attack and recognizes the source of the attack. The following section describes how the system uses this information to prevent the detected attack.
4.5.3 The Prevention-Process

The Prevention-Process designs new prevention strategies that are implemented in the Filtering-Process. These strategies protect the system from detected DoS attacks, discover the source or nature of detected attacks and investigate suspicious communications to confirm DoS attacks. New strategies result from updating the rules of inspections and the decision tasks of the Filtering-Process (such as updating the size or the enabling group of the Monitoring-Repeated-Communication-List, updating the Blocked-Communication-List, changing the time of token validation, adjusting the time of communications or marking communications). The following gives an example of part of the Prevention-Process code:

If (suspicion-attack(flooding, Step x, medium)) then
{add step x to the monitoring list}
If (attack(flooding, source C, high)) then
{ (add source C to the Blocked-Communication-List) &&
  (monitoring list.enableGroup="step x") &&
  (monitoring list.ReqDecision=add to the Blocked-Communication-List) }

In the first statement of this code, the prevention process enables step $x$ of the protocol in the Monitoring-Repeated-Communication-List if there is a medium-suspicion alert about a flooding attack at this step of the protocol. This helps the system to detect the source of the attack.
4.6 Conclusion

In this chapter we proposed the Token-Filtering Technique (TFT). This technique adds additional plain-text data to the packet header. The TFT provides complete information about the communication. The token consists of two parts: the token’s value and the keys. The system can use these keys to filter packets. These keys cannot be modified by the client. The system can use the token to detect expired, spoofed or malformed packets. In addition, this technique gives the system the ability to delay a client’s next communication. However, it does not require the client to perform unproductive processing tasks. This technique can be used to enhance the efficiency of the packet Filtering-Process and the accuracy of the DoS-attack detection and prevention processes.

In this chapter, we also proposed the Token-Filtering Technique Protocol (TFTP). We designed this protocol based on the TFT, which means a token is included in every message of the protocol. The protocol is thus a stateless connection that provides stateful features, because the token provides all the required information about the communication including the state of the communication. The protocol describes all the required communication messages between the client and the system.

This chapter also discussed filtering, detection and prevention processes. We have seen that by using the TFT, system-security administrators can write
specific rules to filter, detect and prevent a DoS attack. In order to enhance the system’s ability to detect repeat DoS attacks, the proposed solution includes the Monitoring-Protocol-Behaviour technique. This technique is designed to detect the communication step that is the target of the attack. With this information, the system can assign system resources to more readily manage the attack.

This chapter described the TFT, the TFTP and their associated processes. The techniques described in this chapter are evaluated in the next chapter (Chapter 5). This evaluation examines system security, performance and functionality and compares the proposed solution with current solutions.
Chapter 5

Characterization

5.1 Introduction

In Chapter 2, we presented a background to DoS attack and the existing models and solutions that have been proposed to protect systems against this type of attack. The results of this discussion showed that none of the existing solutions satisfies the defined assessment parameters (Table 2.5). In Chapter 3, therefore, we proposed a new security model, DoS-PIF, that protects CGeSS from DoS attack. This model was further developed into an approach called HASP-CGeSS. The specifications of this approach’s techniques were described in the previous chapter (Chapter 4). This description included the proposed communication control TFT, the proposed protocol TFTP, the system lists Blocked-Communication-List and Monitoring-Repeated-Communication-List and the technique Monitoring-Protocol-Behaviour. These descriptions were followed by an explanation of how these components work together to filter, detect and prevent DoS attack.

In this chapter this proposed solution is analyzed in terms of its functionality, security, correctness and performance – the parameters developed in Chapter 2.
The results of these evaluations are used to determine the ability of the proposed solution to meet the key requirements. In order to reach this goal we perform the following.

First of all, the DoS-resistance of the TFTP is examined. Using the recommendations and suggestions found in the literature, the design of the protocol and its messages are analyzed to determine whether the protocol is DoS-resistant. Part of this analysis, a cost-based formal framework, is used to analyze the operational cost of communications for legitimate users and under DoS-attack scenarios. The performance of the protocol is also analyzed by comparing it to other DoS-resistant protocols in terms of operational efficiency and scalability.

Second, we analyze the dependability of the proposed communication control in filtering packets based on their headers in order to provide higher availability for the system. This includes an analysis of whether this control can protect the system from spoofed attacks, and from any modification in the packet’s header or in the control itself. This analysis is performed using a new proposed formal-analysis model that is designed to evaluate protocols and their communication controls in detecting and preventing DoS attack.

The correctness of the proposed token method in achieving the TFT and TFTP goals as a communication control is evaluated using a simulation. The ability of the Monitoring-Repeated-Communication-List in detecting repeat attacks is also evaluated using a simulation.
The rest of this chapter is organized as follows. In the next section (Section 5.2) the proposed protocol’s DoS-resistance and operational efficiency are analyzed. This is followed by an analysis of the proposed communication control (Section 5.3). Section 5.4 discusses the simulation’s results relating to the proposed token method and the Monitoring-Repeated-Communication-List. An analysis of system scalability is presented in Section 5.5. The last section summarizes and concludes this chapter.

5.2 Comparison and Performance

CGeSS require a strong level of authentication to authenticate clients. This kind of authentication might be used by attackers to perform DoS attacks to minimize the availability of a system. Systems that require fully authenticated users provide critical services that are used by a large number of clients. These services must have high availability at all times; any disconnection of these services is unacceptable. Hence, the communication protocol should be designed in a way that resists DoS attack so as to offer greater availability of the system.

Responding to this requirement, protocol engineers provide different groups of strategies for designing and evaluating protocols to ensure a certain level of DoS resistance. Suratose [67, 146] classifies these strategies as preventing memory exhaustion, preventing CPU exhaustion and gradual authentication.
Leiwo et al. [147] provide a number of recommendations to improve the DoS resistance of protocols, including the following.

i. The client’s information should not be stored in the system before the authentication is completed.

ii. The protocol should include attack-detection methods to stop malicious communication before the authentication task starts.

iii. The process load of the system should not be more than the process load of the client.

iv. The system should be able to assign different levels of client process loads, so in the case of an attack the system can increase the client’s processing load as prevention against DoS attacks.

Meadows [148] suggests that additional DoS-resistant strategies to implement and evaluate PKI authentication protocols should include:

i. minimizing the processing load of the system’s engaging operations in the protocol, and increasing the system’s resources;

ii. using authentication methods to identify the source of any attack; and

iii. using weak authentication at the beginning of the authentication stage and strong authentication at the end of the authentication stage.

Ensuring that a protocol is secure against DoS attack is complex because we must analyze the prevention of DoS attack in all protocol steps [149].
Meadows in [149] and [148] proposes a formal framework to analyze and evaluate the DoS-resistance of cryptographic protocols by comparing the cost to the attacker and the cost to the responder. In the next section we use this framework to analyze the DoS-resistance of TFTP. To do so, we first provide a background to the Meadows’s framework.

5.2.1 The Meadows’s Cost-Based Framework

The construction of the Meadows’s framework starts by observing all message points during the execution of the protocol. This observation is used to check whether the responder might accept bogus messages, which are used to launch DoS attacks, as genuine. If there is a risk of the responder accepting a bogus message as genuine, a DoS attack could possibly be committed in the responder’s system. These illegal messages are launched in order to occupy the receiver’s resources with processing the messages. The protocol must therefore be designed so that it offers a certain level of guaranteed protection against these attacks. The Gong and Syverson's fail-stop-protocols concept in [150] provides such a guarantee.

The fail-stop protocols provide advantageous features for a DoS-resistant protocol. However, they must use strong authentication from the beginning as a fail-stop property. This makes these protocols potentially vulnerable to DoS attack that is caused through the receiver verifying bogus messages. The Meadows’s framework has modified the use of the fail-stop concept by
requiring strong authentication at the end of the authentication process to enhance the protocols for DoS-resistance.

The Meadows’s framework uses the Alice-and-Bob notation to present message sequences between an initiator and responder. The definitions of this framework are as follows.

(i) The communication between two communication parties $A$ and $B$ is a sequence of messages in the form $A \rightarrow B : M$.

Because our goal is to analyze the protocol for DoS-resistance, we need to include message-processing steps as follows.

(ii) The communication between $A$ and $B$ is a sequence of messages in the form $A \rightarrow B : T_1, \ldots, T_k \parallel M \parallel O_1, \ldots, O_n$. $T_i$ refers to the operations that are performed by $A$, and $O_i$ refers to the operations that are performed by $B$.

The sequence $T_1, \ldots, T_k$ represents the operations that are performed by $A$ in order to produce $M$, while the sequence $O_1, \ldots, O_n$ represents the operations that are performed by $B$ in order to process and verify $M$. The difference between these events is defined in the following way.

(iii) Let $L = A \rightarrow B : T_1, \ldots, T_k \parallel M \parallel O_1, \ldots, O_n$ be a notation line in the Alice-and-Bob specification. $X$ can then be defined as an event that occurs in $L$ if:

a. $X$ is one of $T_i$ or $O_i$; or
b. \( X \) is \( \text{'A sends } M \text{ to } B' \) or \( \text{'B receives } M \text{ from } A' \).

The events \( \text{'A sends } M \text{ to } B' \) and \( T_1, \ldots, T_k \) occur at \( A \), and the \( \text{'B receives } M \text{ from } A' \) and \( O_1, \ldots, O_n \) events occur at \( B \). These events can be divided into three types: (i) a normal event, which contains sending and receiving events; (ii) verification events; and (iii) an accept event. The first type of event can occur at either the sender or receiver. The second and third types only occur at the receiver. \( O_n \) represents the accept event.

In order to be specific about the actual sequence in which events occur, the notion of a desirably-precedes relation is included in this notation. It is similar to Gong and Syverson's fail-stop model's notation and based on the causally-precedes relation of Lamport [151]. Desirably-precedes can be defined as follows.

(iv) Desirably-precedes has the following characteristics:

a. if \( A \rightarrow B : R_1, \ldots, R_m \parallel M \parallel O_1, \ldots, O_n \) appears in a specification then the event \( O_i \), which is performed by \( B \) in order to process \( M \), is a desirably-precedes for any event \( O_j \) where \( i < j \).

b. if \( A \rightarrow B : R_1, \ldots, R_m \parallel M \parallel O_1, \ldots, O_n \) appears in a specification then the event \( R_i \), which is performed by \( A \) in order to process \( M \), is a desirably-precedes for any event \( R_j \) where \( i < j \).
c. if $A \rightarrow B : R_1, \ldots, R_m \parallel M \parallel O_1, \ldots, O_n$ precedes $B \rightarrow Y : S_1, \ldots, S_m \parallel M \parallel T_1, \ldots, T_n$, then $O_n$ desirably-precedes $S_1$.

d. if $A \rightarrow B : R_1, \ldots, R_m \parallel M \parallel O_1, \ldots, O_n$ then the event ‘$A$ sends $M$ to $B$’ desirably-precedes the event ‘$B$ receives $M$ from $A$’.

e. if $E_1$ desirably-precedes $E_2$ and $E_2$ desirably-precedes $E_3$ then $E_1$ desirably-precedes $E_3$.

Note: in the definition of desirably-precedes the message received by $B$ is the same as the message $M$ which was sent by $A$. This is needed to find the cost sets and protocol-cost function.

(v) A cost set $C$ consists of a set with + operations with partial order $<$ such that $x + y \geq y$ and $x + y \geq x$, where $x$ and $y$ in $C$.

(vi) A function $\delta$ is the event-cost function that is used to represent the cost set $C$. It consists of four members: $0 <$ cheap $<$ medium $<$ expensive. Medium cost members include the following cryptographic operations (where encrypting is ‘encrypt’, decrypting is ‘decrypt’ and performing pre-mathematical expressions is ‘preexp’), and expensive cost members involve the following cryptographic operations (where difficult expressions is ‘exp’, checking signature is ‘checksig’ and signing a data using a signature is ‘sign’). All other events are cheap.
(vii) The function $\delta'$ is the message-processing-cost function that is associated with $\delta$ on verification events. So if the line $A \rightarrow B : O_1, \ldots, O_k \parallel M \parallel V_1, \ldots, V_n$ appears, then for each verification event $V_j: \delta'(V_j) = \delta(V_j) + \cdots + \delta(V_n)$.

The message-processing cost represents the cost of message processing up to the failed verification event.

(viii) The function $\Delta$ is the protocol-engagement function. It is defined as an accept event as follows:

If the line $A \rightarrow B : O_1, \ldots, O_k \parallel M \parallel V_1, \ldots, V_n$ appears, where $V_n$ is the accept event, then:

a. If the line $B \rightarrow X : O'_1, \ldots, O'_k \parallel M' \parallel V'_1, \ldots, V'_{n-1}$, where $V_n$ immediately desirably-precedes $O'_1$, does not exist then $\Delta(V_n)$ describes the sum of the cost for all operations which are at $B$ and desirably-precedes $V_n$ desirably-precedes $V_n$.

b. If the line $B \rightarrow X : O'_1, \ldots, O'_k \parallel M' \parallel V'_1, \ldots, V'_{n-1}$, where $V_n$ immediately desirably-precedes $O'_1$, does exist then $\Delta(V_n)$ describes the sum of the cost for all operations which are at $B$ and desirably-precedes $V_n$ plus the cost of $O'_1$. 
(ix) If $C$ is a cost set, then the function $\phi$ is a function from the attacker’s actions set. The attacker cost function $\Phi$ describes the capability of the attacker. It can be defined as $\Phi = (\{x_1 \ldots x_n\}) = \phi(x_1) + \cdots + \phi(x_n)$.

We assume that the attacker costs and capabilities are as follows:

a. sending a legitimate message (Cost = the cost of computing the message);

b. forging a return address (Cost = cheap);

c. reading messages (Cost = medium);

d. creating a new message out of old messages (Cost = cost of deconstructing the old messages + cost of creating the new messages);

e. disabling of a legitimate principal (Cost = medium; this requires another denial-of-service attack, but since a disabled principal can be used more than once, the cost can be amortized over a number of attacks);

f. substituting bogus messages for genuine ones in real time, as would be done in a man-in-the-middle attack (Cost = very expensive);

g. breaking cryptosystems (Cost = maximal); and

h. inducing a principal to initiate communication with a bogus, disabled, or dishonest principal (Cost = expensive to very expensive; it is not
hard to induce a principal to do this a few times, but it is probably difficult to induce this the number of times required to launch a DoS attack).

Now we will provide the definition of the fail-stop from the Gong and Syverson's fail-stop-protocols concept [150], slightly modified as follows.

(x) A cryptography protocol is fail-stop if no event or verification on the message is executed desirably-after the failed verification.

(xi) Let $\Theta$ be defined as the attack function from the events set.

(xii) Let $G$ be an attacker cost set and $C$ be a defender cost set. A tolerance relation $C \times G$, which consists of all pairs $(c, g)$, is defined so the protocol designed to protect the system resources from $c$ cost which is greater than $g$ cost. The protocol is therefore protected against DoS attack if $(c', g')$ exists in the tolerance relation $\tau = \Delta(E, \Theta(E))$ where $(c, g) \in \tau$ if $g' \geq g$ and $c' \leq c$ [146]. The tolerance relation $\tau$ can thus be represented as follows [146]:

$$\tau = \left\{ (cheap, cheap), (cheap, medium), (cheap, expensive), (medium, cheap), (medium, medium), (medium, expensive), (expensive, expensive) \right\}$$

When other pairs of tolerance relation $\tau$ (such as $(expensive, cheap)$ and $(expensive, medium)$) exist in the analysis of an examined protocol then the protocol is vulnerable to low-cost DoS attack.
This framework is used in the next two sections as a cost-based analysis and protocol assessment to examine the DoS-resistance of the TFTP.

5.2.2 Protocol Cost-Based Analysis

In this section we examine the vulnerability of the TFTP to DoS attack and identify the protocol’s most costly steps. These steps are the most critical and are targeted by the attackers because these steps perform costly processing operations that tie up system resources. Table 5.1 shows TFTP cost-based specifications that are translated from the TFTP specifications (that is, Section 4.4) using the Meadows’s framework notation.

From the first step in Table 5.1 we can see that the TFTP does not store any data belonging to the client in the system. The protocol is thus a stateless connection and prevents memory DoS attacks. To test the protocol’s ability in preventing CPU DoS attacks, we need to identify the most costly steps of the protocol and then examine the DoS resistance of these steps under well-known flooding attacks by using the cost-based Meadows’s framework.

In the cost-based Meadows’s framework (Section 5.2.1 xii), a protocol is vulnerable to DoS attack if it has a potential for \((\text{expensive, cheap})\) or \((\text{expensive, medium})\) pairs in a step of the protocol.
### Table 5.1 TFTP in the Cost-Based Framework Notation

<table>
<thead>
<tr>
<th>Stage</th>
<th>Step</th>
<th>MSG</th>
<th>Cost-based function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection stage</td>
<td>1</td>
<td>$C \rightarrow S$</td>
<td>$\text{GetServerIP}</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>$S \rightarrow C$</td>
<td>$\text{generateToken}(\text{Token}^1)</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>$C \rightarrow S$</td>
<td>$\text{waitForDelay}, \text{IncludeToken}^1</td>
</tr>
<tr>
<td>Authentication stage</td>
<td>1</td>
<td>$S \rightarrow C$</td>
<td>$\text{generateToken}(\text{Token}^2)</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>$C \rightarrow S$</td>
<td>$\text{IncludeToken}^2, \text{generateRandomNo}(\text{RN}_C), \text{encrypt(Pubkey}_S, {\text{Cert}_C, \text{RN}_C})</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>$S \rightarrow C$</td>
<td>$\text{generateToken}(\text{Token}^3), \text{generateRandomNo}(\text{RN}_S), \text{encryptPKI(Pubkey}_C, {\text{RN}_S, \text{NextIP}, \text{Token}^3})</td>
</tr>
<tr>
<td>Service communication stage</td>
<td>1</td>
<td>$C \rightarrow S$</td>
<td>$\text{IncludeToken}^3, \text{computeSharedKey(\text{RN}<em>S, \text{RN}<em>C) = ShKey}</em>{S-C}, \text{encryptSymmetric(ShKey}</em>{S-C}, {\text{data, PacketHeader}, })</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>$S \rightarrow C$</td>
<td>$\text{generateToken}(\text{Token}^4), \text{encryptSymmetric(ShKey}_{S-C}, {\text{data, Token}^4, \text{NextIP} })</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>$C \rightarrow S$</td>
<td>$\text{IncludeToken}^x, \text{encryptSymmetric(ShKey}_{S-C}, {\text{end, PacketHeader} })</td>
</tr>
</tbody>
</table>
The *expensive* operations in the TFTP only exist in the authentication stage of the protocol. This stage is the key-establishment phase of the protocol. The DoS resistance of this stage is tested with several defined attack scenarios as follows.

### 5.2.3 Protocol Assessment

In this section we perform a cost-based analysis of the protocol by comparing the cost to the attacker and the system when the protocol is under possible DoS-attack scenarios (Table 5.2).

**Attack Scenario 1 (Sending a high rate of flooding requests):** An attacker needs to consider two factors in order to make this attack succeed in the authentication stage: i) the use of spoofed IP addresses; and ii) repetitively sending the communication over a long period. In the TFTP, the system only accepts packets that have a token in the authentication stage. The attacker thus requires a valid token in order to communicate with the system in the first step of the authentication stage. The system provides this token in the last step of the connection stage. The token represents the IP address of the attacker. So, if the attacker spoofed his or her IP address, then the system drops the packet because an incorrect token is used in its header. On the other hand, if the attacker uses his or her real IP address, then the attacker cannot continuously repeat the communication, because the token is only valid for a specific period of time. Once the token expires, the system drops the packets as the token in the header is no longer valid.
<table>
<thead>
<tr>
<th>No</th>
<th>The attack scenario</th>
<th>Targeted resources</th>
<th>Processing cost</th>
<th>Targeted step</th>
<th>Attacker-used activities</th>
<th>Difficulty of the attack if successful</th>
<th>Time duration of the attack if successful</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The attacker sends a high rate of flooding requests</td>
<td>High Low</td>
<td>Low Low</td>
<td>1ˢᵗ N/A</td>
<td>Yes Yes No</td>
<td>Med</td>
<td>Unlimited</td>
</tr>
<tr>
<td></td>
<td>The attacker stops the communication after Step 3 of the authentication stage</td>
<td>Low High</td>
<td>High Low</td>
<td>N/A All</td>
<td>No No No</td>
<td>Low</td>
<td>one time</td>
</tr>
<tr>
<td>2</td>
<td>The attacker performs Step 3 with incorrect encrypted parts</td>
<td>No Yes</td>
<td>Med High</td>
<td>N/A 3ʳᵈ</td>
<td>No Yes Yes</td>
<td>High</td>
<td>Med</td>
</tr>
<tr>
<td>3</td>
<td>The attacker performs Step 3 with an incorrect token and an incorrect encrypted part</td>
<td>No Yes</td>
<td>Low Low</td>
<td>N/A 3ʳᵈ</td>
<td>No Yes Yes</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>4</td>
<td>The delayed attacker sends a correct packet before the specified time in Step 3 of the authentication stage</td>
<td>No Yes</td>
<td>Low Low</td>
<td>N/A 3ʳᵈ</td>
<td>No Yes Yes</td>
<td>High</td>
<td>Med</td>
</tr>
<tr>
<td>5</td>
<td>The delayed attacker sends an incorrect packet before the specified time in Step 3</td>
<td>No Yes</td>
<td>Low Low</td>
<td>N/A 3ʳᵈ</td>
<td>No Yes Yes</td>
<td>High</td>
<td>Very High</td>
</tr>
<tr>
<td>6</td>
<td>The attacker uses repeat encrypted signatures</td>
<td>No Yes</td>
<td>High Very High</td>
<td>N/A 3ʳᵈ</td>
<td>No Yes No</td>
<td>Med</td>
<td>Low</td>
</tr>
</tbody>
</table>

Table 5.2 Characteristics of Attack Scenarios
Use of a token thus prevents the attacker from performing this attack in the authentication stage. However, if the attacker repeats the last step of the connection stage the result of this attack is as follows:

\[ \delta'(verify_{request}) = cheap \]

\[ \theta (verify_{i}) = cheap \]

\[ (\Delta (accept_3), \theta (accept_3)) \in \mathcal{T} \]

\[ (Cheap, cheap) \in \mathcal{T} \]

From the cost-based analysis above we can see that the system is not vulnerable under this attack scenario, because the system only requires cheap operations to manage these communications; they do not consume too much of its resources. However, there is a potential for this attack to succeed if it is performed in the first step of the protocol in the connection stage. In this attack, the attacker sends a high rate of bogus requests without continuing the communication. The aims of the attacker are: (i) to enact a SYN attack on the system, which is done by filling the system’s backlog queue with useless connection information, and/or (ii) to enact a flooding attack on the system by burdening it with processing these bogus requests and thereby rendering it incapable of serving any genuine requests.

When the system receives each request, the system generates a token. This token contains all the required information about the communication. The
system does not hold any information belonging to the client and consequently the system cannot receive any kind of SYN attack. The DoS resistance of the protocol for this kind of attack can be calculated as follows:

$$\delta'(verify_{request}) = cheap$$

$$\theta(verify_{i}) = cheap$$

$$(\Delta(accept_{1}), \theta(accept_{1})) \in \mathcal{T}$$

$$(Cheap, cheap) \in \mathcal{T}$$

The cost-based analysis above demonstrates that the protocol is not vulnerable to a flooding attack in the first step of the protocol. Minimal processing is required to manage these communications and so few resources are consumed.

**Attack Scenario 2 (The attacker stops the communication after Step 3 of the authentication stage):** In this attack the attacker normally communicates with the server until the fifth message of the protocol, which is the second message of the authentication stage, and then the attacker exits the communication. In this type of attack, the attacker tries to occupy the system with the processing required for the authentication stage, and also tries to make it difficult for the system to detect him or her as an attacker. The result of this attack is as follows:

$$\delta'(verifySig) = expensive$$

$$\theta(decryptPKI) = expensive$$
From the cost-based analysis above, we can see that the system is not vulnerable to this type of attack, because for both the system and the attacker it involves expensive operations. The attacker needs expensive operations to launch the attack; consequently the protocol is not vulnerable to low-cost DoS attack under this scenario.

Attack Scenario 3 (The attacker performs Step 3 of the authentication-stage with an incorrect encrypted part): In this attack, the attacker commences the communication with the system normally until the second step of the authentication stage. At this point, the attacker waits the required time, and then sends incorrect data in the encrypted part of the message. The attacker’s goal is to keep the system busy with decrypting a high number of encrypted signatures. The attacker under this attack scenario has a correct token and waits the required delay time, but the attacker does not decrypt the encrypted message received from the system in Step 2 of the authentication stage. The result of this attack scenario is as follows:

\[
\delta'(\text{verifyPrivateKey}) = \text{medium}
\]

\[
\theta(\text{decryptPKI}) = \text{expensive}
\]

\[
(\Delta(\text{decryptPKI}), \theta(\text{decryptPKI})) \in \mathcal{T}
\]
(medium, expensive) ∈ T

From the cost-based analysis above, we can see that the system is not vulnerable to this type of attack, because the attacker needs to perform expensive operations to commit this attack. On the other hand, for each expensive operation performed by the attacker, the system needs to perform a medium-cost operation. The system is thus resistant to low-cost DoS attacks under this scenario.

If the attacker does not verify the signature in Step 2 (in order to obtain the maximum time for repeating the message), yet uses a correct token, the analysis shows that the protocol is still resistant to this type of attack:

δ'(verifyPrivateKey) = cheap

θ (decryptPKI) = medium

(Δ (decryptPKI), θ (decryptPKI)) ∈ T

(medium, cheap) ∈ T

From the cost-based analysis above, we can see that the protocol is not vulnerable to this type of attack, because the system only requires medium-cost operations to handle the attacker’s communications. The system remains steadfast under this type of attack scenario.

**Attack Scenario 4 (The attacker performs Step 3 with an incorrect token and incorrect encrypted part):** In this scenario, the attacker sends a packet
with an incorrect token and incorrect data in the encrypted part. In addition, in order to gain maximum time for repeating the message, the attacker does not verify the signature in Step 2. The difference between this scenario and Attack Scenario 3 is that the attack uses an incorrect token in this scenario in order to use the time that comes from waiting for a new token to send incorrect encrypted signatures. The incorrect token is presumably spoofed. Under this attack scenario the attack will be detected after the token is verified (using \(verify_{Token}(Token^2)\)) and the \(decryptPKI\) operation will not be executed. Consequently the accept event for the step will not be reached. The result of this attack scenario is as follows:

\[
\delta'(verify_{Token}(Token^2)) = cheap
\]

\[
\theta(verify_1) = cheap
\]

\[
(\Delta(completion of verify_{Token}), \theta(completion of verify_{Token})) \in T
\]

\[
(cheap, cheap) \in T
\]

From the cost-based analysis above, it can be seen that the protocol is not vulnerable to this attack scenario. Minimal processing is required to manage these communications and so few resources are consumed.

**Attack Scenario 5 (The delayed attacker sends a correct packet before the specified time in Step 3 of the authentication stage):** In this case, the attacker proceeds as in Attack Scenario 3, but sends the packet before the delay time specified by the server. Under this scenario the packet is dropped. The packet’s
token indicates that the packet should arrive at or after the specified delay time, but the packet arrives before this time. The result of this attack scenario is as follows:

\[ \delta'(verify_{Token}) = \text{cheap} \]

\[ \theta (verify_t) = \text{cheap} \]

\[ (\Delta(\text{completion of verify}_{Token}), \theta(\text{completion of verify}_{Token})) \in T \]

\[ (\text{cheap,cheap}) \in T \]

The cost-based analysis above demonstrates that the protocol is not vulnerable under this attack scenario. The system only requires cheap operations to manage these communications and few system resources are consumed.

**Attack scenario 6 (The delayed attacker sends an incorrect packet before the specified time in the 3rd step):** In this case the attacker proceeds as in Attack Scenario three, but uses incorrect data and sends the packet before the delay time specified by the server. Under this scenario the packet is dropped before it is processed, because the token shows the delay time after which the packet should arrive. The result of this attack scenario is as follows:

\[ \delta'(verify_{Token}) = \text{cheap} \]

\[ \theta (verify_t) = \text{cheap} \]

\[ (\Delta(\text{completion of verify}_{Token}), \theta(\text{completion of verify}_{Token})) \in T \]
The cost-based analysis above indicates that the protocol is not vulnerable to this attack scenario. The system only requires cheap operations to manage these communications and few system resources are consumed.

**Attack Scenario 7 (Repeating encrypted signatures):** Under this attack scenario the attacker continues to communicate with the server until the third message. The attacker then starts repeating Step 3 of the authentication stage. The attacker starts the communication with the system as a genuine client, and then keeps the system busy with decrypting signatures. The attacker targets this step because it is the most costly step for the system. Under this attack scenario, all operations in Step 3 will be executed until the token expires. The cost of the first message, until the repeat packets start, will be same as the cost of Attack Scenario 2. The result of the repeat packets after their tokens expire is as follows:

\[ \delta'(verify_{Token}) = cheap \]

\[ \theta(verify_i) = cheap \]

\[ (\Delta(completion \ of \ verify_{Token}), \theta(completion \ of \ verify_{Token})) \in \mathcal{T} \]

\[ (cheap, cheap) \in \mathcal{T} \]

From the cost-based analysis above, we can see that the system is not vulnerable to this part of the attack, because only cheap operations are required
to manage these communications and these consume minimal system resources. On the other hand, the result of the repeat packets after the first message and before their tokens expire is as follows:

\[ \delta'(\text{verifySig}) = \text{cheap} \]

\[ \theta(\text{decryptPKI}) = \text{expensive} \]

\[ (\Delta(\text{accept}_5), \theta(\text{accept}_5)) \notin \mathcal{T} \]

\[ (\text{expensive,cheap}) \notin \mathcal{T} \]

The cost-based analysis above shows that the TFT protocol is vulnerable under this part of the scenario for low-cost Dos attack, because the system requires expensive operations to manage these communications. In contrast, the attacker only requires cheap operations to launch the attack. However, this result is expected in cryptography protocols [68]. To avoid the expense of these operations, the system requires a check list in this stage to avoid re-authenticating already authenticated signatures [68]. This is performed in the HASP-CGESS by implementing the Monitoring-Repeated-Communication-List as discussed in Section 4.5.2.2. Once the system detects a repeat communication, the source of the communication is included in the Blocked-Communication-List (Section 4.3.1). When this happens, then:

\[ \delta'(\text{verifyToken}) = \text{cheap} \]

\[ \theta(\text{verify}_t) = \text{cheap} \]
\[(\Delta (\text{completion of } verify_{\text{Token}}), \Theta (\text{completion of } verify_{\text{Token}})) \in T\]

\[(\text{cheap,cheap}) \in T\]

As can be seen in the above result, the protocol is not vulnerable under this scenario if the attacking source is detected on the repeat list. The system only requires cheap operations to manage these communications and so minimal resources are consumed. However, if the repeat communication is not detected on the list, then the second part of the repeat communication, when the token expires, costs as follows:

\[\delta'(verify_{\text{Token}}) = \text{cheap}\]

\[\Theta (verify_{\text{Token}}) = \text{cheap}\]

\[(\Delta(\text{completion of } verify_{\text{Token}}), \Theta(\text{completion of } verify_{\text{Token}})) \in T\]

\[(\text{cheap,cheap}) \in T\]

Based on the above result, under Attack Scenario 7 the protocol is not vulnerable to this part of the attack because the system only requires cheap operations to manage these communications. The consumption of system resources is thus minimized.

The above analysis results are now used to evaluate the TFTP’s DoS-resistance.
5.2.4 DoS-Resistance

The strategies reviewed in Section 5.2 indicate that a number of challenges need to be met in order to strengthen the DoS resistance of protocols. These challenges relate to: (i) the storage of client information; (ii) the inclusion of attack-detection methods; (iii) the comparative size of process loads; and (iv) the control of communication timing and the assignment of delay periods.

The above analysis has shown that the TFTP possesses the features to meet these challenges, as follows.

i. **The storage of client information**: With the TFTP, it is unnecessary for the system to store any information belonging to clients before the authentication is completed. All required communication information is stored in the token, which is given to the client on the promise that the client will produce the token in the next communication. This feature responds to the first requirement of Leiwo et. al.’s suggested strategies concerning the storage of client information.

ii. **The attack-detection process**: The proposed protocol is monitored by an attack-detection process. This process attempts to stop malicious communication in the connection stage before the authentication task begins. In addition, by using the TFT, the system can detect both spoofed sources and expired communications. The proposed protocol thus includes attack-detection methods to stop malicious communication before the authentication task begins. This feature
responds to the second requirement of Leiwo et. al.’s suggested strategies: that protocols should include attack-detection methods to stop malicious communication before the authentication task starts.

iii. The comparative size of process loads: For all steps of the TFTP, the required processing of the system is less than or equal to the processing required from the client. The design of the protocol thus satisfies the third requirement of Leiwo et. al.’s suggested strategies: that the process load of the system should not be more than the process load of the client.

iv. Control of the communication timing and the assignment of delay periods: Using the token in the TFTP, a client can be delayed for a period of time without requiring the client to perform unproductive processing. This enables the client to be placed on hold and gives the system the ability to temporarily redirect resources. This achieves the goal of the proof of work [144], which makes the DoS attack more difficult for an attacker to perform. Using the TFTP, the system can assign different delay periods to different clients. The last recommendation of Leiwo et. al.’s suggested strategies requires a system to be able to assign different levels of processing load to different clients. The goal of managing this processing load is to be able to delay different clients for different time periods. The proposed TFTP achieves this goal without involving the client in unproductive
processing. Consequently, the protocol satisfies the fourth requirement of Leiwo et. al.’s suggested strategies.

From these four features we can see that the proposed protocol is DoS-resistant based on Leiwo et. al.’s [147] suggested strategies.

In addition to these features, the protocol’s steps start with tasks (such as giving the client a token) that involve minimal processing costs and the more expensive processing tasks (such as signature authentication) are performed later. In addition, for each step of the protocol where the system receives a communication from the client, the system starts with a low-process authentication operation (such as authenticating the token), and then proceeds to the more expensive operations. This aspect of the TFTP’s design responds to the first and the last requirements of Meadows’s [148] DoS-resistant strategies. In addition, the protocol can authenticate the source of the client because it uses a token to store the client’s IP address. This feature responds to the second requirement of Meadows’s [148] DoS-resistant strategies. Because of these three features, the proposed protocol is DoS-resistant according to Meadows’s DoS-resistant strategies [148].

However, there are other existing protocols which are also DoS-resistant based on Meadows’s logic. We need to compare the efficiency and the functionality of the TFTP with these existing protocols. This will help us determine whether the TFT protocol offers any advantages over these other DOS-resistant protocols, and whether it avoids their limitations.
5.2.5 Operational efficiency

In this section we evaluate the operation efficiency of the proposed protocol in preventing DoS attack using a computation of cost analysis. This evaluation focuses on the authentication stage of the proposed protocol because the authentication stage is the most expensive stage. The performance of the TFT protocol in resisting DoS attack under key-establishment attack is also tested. This evaluation includes a comparison with other protocols that provide solutions for these types of attacks.

Comparing protocols is challenging. Different protocols employ different algorithms. Comparing algorithms such as hashing, exponentiations and encryption is difficult because of the different types of implementation. To overcome this problem, in the performance evaluation we use the Wei Dai [145] cryptographic-protocol benchmarks. These cryptographic-protocol benchmarks show the speed for the most commonly used cryptographic algorithms. They were coded in C++, compiled with Microsoft Visual C++ 2005 SP1, and run on an Intel Core processor 2 1.83 GHz using Windows Vista in 32-bit mode. Table 5.3 shows the results for selected cryptographic algorithms that are usually employed in key-establishment protocols.

The costs (Table 5.3) are used to compare the performance of the proposed protocol with the Just-Fast-Keying (JFKi) authentication protocol (Section 2.3.7.1) and the Host-Identity-authentication Protocol (HIP) (Section 2.3.7.2). The JFKi protocol is an example of a protocol that implements the cookies
technique and the HIP is an example of a protocol that uses the client-puzzle technique. Both of these protocols are DoS-resistant. We have chosen these protocols because they are well-known protocols that protect systems against DoS attack using a communication control. These two solutions and the proposed TFTP solution are stateless and include different types of communication controls. However, the additional costs that arise from using these types of controls vary. Figure 5.1 provides a comparison of the additional processing costs for each of these communication controls.

Table 5.3 Processing Cost of Selected Cryptographic Operations [145]

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>CPU Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hash</td>
<td>Cycle Per Byte</td>
</tr>
<tr>
<td>MD5</td>
<td>6.8</td>
</tr>
<tr>
<td>SHA-512</td>
<td>17.7</td>
</tr>
<tr>
<td>HMAC(SHA-1)</td>
<td>11.9</td>
</tr>
<tr>
<td>PKI cryptography operations</td>
<td>Keycycle/operation</td>
</tr>
<tr>
<td>RSA 1024 Encryp.</td>
<td>140</td>
</tr>
<tr>
<td>RSA 1024 Decryp.</td>
<td>2680</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>CPU Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symmetric crypto. operations</td>
<td>Cycle Per Byte</td>
</tr>
<tr>
<td>AES/ECB (128-bit key)</td>
<td>16</td>
</tr>
<tr>
<td>DES/CTR</td>
<td>54.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PKI cryptography operations</th>
<th>Keycycle/operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSA 1024 Signature</td>
<td>2710</td>
</tr>
<tr>
<td>RSA 1024 Verification</td>
<td>130</td>
</tr>
</tbody>
</table>

Figure 5.1 indicates that the TFTP is the most efficient technique for filtering packets based on the communication controls. This is because it uses only two calculation operations. The JFKi protocol uses two hashing operations to handle a cookie. The HIP uses two calculation operations and two hash operations. These costs are very small but they are important in DDoS attacks at the first step of the protocol where the flooding request attacks are performed.
The efficiency of a protocol in preventing DoS attacks for a system can be measured by comparing the estimated computational cost that is required to complete the protocol steps for the system and the client. Table 5.4 provides the cost comparison for the three cryptographic protocols. The comparison includes the legitimate client and the seven types of attack scenarios that were defined in Section 5.2.3. However, the cost of processing the controls is ignored because (i) it is small compared to other costs, and (ii) we need to compare the performance of the protocol under low-cost DoS attack.
<table>
<thead>
<tr>
<th>Communication Scenario</th>
<th>Required Processing (CPU Cycles)</th>
<th>JFKi</th>
<th>HIP</th>
<th>TFTP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Client</td>
<td>Server</td>
<td>Client</td>
</tr>
<tr>
<td>legitimate user request</td>
<td></td>
<td>5790</td>
<td>5660</td>
<td>3901</td>
</tr>
<tr>
<td>the attacker sends a high rate of invalid requests</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>the attacker stops communication after step 3</td>
<td></td>
<td>2980</td>
<td>5660</td>
<td>3769</td>
</tr>
<tr>
<td>the attacker performs step 3 with an incorrect encrypted part</td>
<td></td>
<td>0</td>
<td>2810</td>
<td>2</td>
</tr>
<tr>
<td>the attacker performs step 3 with an incorrect communication control and an incorrect encrypted part</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>the delayed attacker sends a correct packet before the specified time in step 3</td>
<td></td>
<td>N/A</td>
<td>N/A</td>
<td>3769</td>
</tr>
<tr>
<td>the delayed attacker sends an incorrect packet before the specified time in step 3</td>
<td></td>
<td>N/A</td>
<td>N/A</td>
<td>0</td>
</tr>
<tr>
<td>the attacker uses repeat encrypted signatures</td>
<td></td>
<td>0</td>
<td>5660</td>
<td>0</td>
</tr>
<tr>
<td>unexpired control</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>expired control</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>blocked communication</td>
<td></td>
<td>N/A</td>
<td>N/A</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 5.4 shows that the TFTP is the most efficient when handling a legitimate client. This efficiency provides greater availability of the system because more clients can be served in a unit of time. On the other hand, all three protocols incur the same cost when preventing flooding attacks because all of them provide communication controls. The JKFi protocol is vulnerable to the type of low-cost DoS attack performed in Attack Scenario 2 (defined in Section 5.2.3). However, the proposed TFT protocol is not vulnerable to this type of attack, because it involves similar processing costs for both the system and the attacker. Because all three protocols provide communication controls, they all prevent the type of DoS attack performed in Attack Scenario 4. However, the JKFi communication control, which uses SynCookies, does not provide a control in time for the next communication and consequently does not have the ability to delay communications. On the other hand, the TFTP and HIP provide this feature. This indicates that these two protocols can prevent attacks of the type faced in Attack Scenarios 5 and 6 (defined in Section 5.2.3).

All three protocols can prevent flooding attacks when the attacker uses repeat encrypted signatures. However, the TFTP provides the most efficient protocol of the three in this attack and before the control is expired. As this case is the most critical of the low-cost DoS attacks, both the TFTP and the HIP provide lists to protect the system from repeat encrypted signatures in Step 3 of the authentication stage prior to the control expiring. The HIP protocol holds all authenticated signatures in a list, while the TFTP holds only the last
communications in its list and uses a blocking list for detecting repeated communications.

In order to compare the efficiency of these two protocols’ lists we coded both these list types and tested them with three loads of attacker sizes (Table 5.5).

Table 5.5 Comparison of HIP and TFTP Lists

<table>
<thead>
<tr>
<th>Time required to detect repeat attacks in Step 3 of the authentication stage</th>
<th>Required Processing (nano sec.)</th>
<th>HIP</th>
<th>TFTP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Attacker</td>
<td>Legitimate</td>
<td>Attacker</td>
</tr>
<tr>
<td>1 attacker</td>
<td>8314</td>
<td>15960</td>
<td>364</td>
</tr>
<tr>
<td>50% of clients</td>
<td>8187</td>
<td>15996</td>
<td>4211</td>
</tr>
<tr>
<td>90% of clients</td>
<td>8202</td>
<td>15965</td>
<td>7122</td>
</tr>
</tbody>
</table>

From Table 5.5, we can see that the HIP protocol’s list is more efficient at filtering legitimate users than the TFTP lists if the system receives an attack consisting of only one attacker. However, the TFTP list is more efficient at filtering legitimate users if the system receives attacks of 50% and 90% of the number of users that communicate with the system at a time. This is because the TFTP uses a blocking list. The TFTP list is more efficient at filtering attackers’ repeated packets irrespective of the load on the system.

In order to compare the required client processing of the TFT and client-puzzle technique for the different levels of client delay, we need to test them in the same protocol. Table 5.6 shows the efficiency of the HIP at providing different levels of client delay using its client-puzzle control [146], and also shows the HIP at the same level of client delay using the TFT control.
Table 5.6 Comparison of the Performance of the HIP Using the Client-Puzzle Technique and the HIP Using the TFT

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Client</th>
<th>Server-required processing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Puzzle difficulty</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( F=1 )</td>
<td>( F=2 )</td>
</tr>
<tr>
<td>Legitimate client</td>
<td>3901</td>
<td>4923</td>
</tr>
<tr>
<td>Attack Scenario 1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Attack Scenario 2</td>
<td>3769</td>
<td>4791</td>
</tr>
<tr>
<td>Attack Scenario 3</td>
<td>2</td>
<td>1024</td>
</tr>
<tr>
<td>Attack Scenario 4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Attack Scenario 5</td>
<td>3769</td>
<td>4791</td>
</tr>
<tr>
<td>Attack Scenario 6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Attack Scenario 7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>The delay time for the client before continuing the next step of the communication</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( D=1 )</td>
<td>( D=2 )</td>
</tr>
<tr>
<td>Legitimate client</td>
<td>3899</td>
<td>3899</td>
</tr>
<tr>
<td>Attack Scenario 1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Attack Scenario 2</td>
<td>3767</td>
<td>3767</td>
</tr>
<tr>
<td>Attack Scenario 3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Attack Scenario 4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Attack Scenario 5</td>
<td>3767</td>
<td>3767</td>
</tr>
<tr>
<td>Attack Scenario 6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Attack Scenario 7</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\( F \)=The difficulty level of the puzzle, with \( F=3 \) the most difficult level. \( D \)=Delay time, with \( D=3 \) the longest delay.

We can see that the TFT control provides a similar level of efficiency as the client-puzzle technique, but without the client being required to perform unproductive processing. In contrast, the HIP uses unproductive processing as a delay technique for clients. For example, when the HIP uses the client-puzzle technique, the system applies a delay of level 3 and requires an additional
1048574 cycles in order to delay the legitimate client for a time. On the other hand, when the HIP uses the TFT the client can be delayed without any additional unproductive processing. This leaves the client machine free to execute other tasks during this time.

The ability to control the timing of client communication is just one feature of the TFTP’s functionality. A full discussion of the proposed functionality of the TFT protocol is presented in the following section.

### 5.2.6 Functionality of the Proposed Protocol

Based on the description of the protocol (Section 4.4) and the above evaluation of the protocol’s efficiency, Table 5.7 compares the functionality of the TFTP with two other cryptographic protocols, HIP and JFKi.

Table 5.7 shows that all of the tested protocols offer fast filtering techniques. This is because they have implemented communication controls. However, the functionalities of these controls are different. The TFTP token carries all the communication’s required information. This information can be used for packet filtering based on the token, which takes place in the packet’s header. In addition, this technique enables the system to detect malformed, spoofed and repeat-resending packet attack and ensures the correct sequence of communication messages. In contrast, the other two protocols only carry a portion of the information that is required to prevent a DoS attack. For example, the HIP only carries part of the packet information with the puzzle
and its solution. The JFKi protocol only carries the data that is usually stored in the backlog queue.

All three protocols can detect expired controls (and consequently, expired communications), but only the TFTP and the HIP can delay the client’s next communication. The HIP requires the client’s machine to perform additional processing to find the solution of the puzzle in order to delay the client’s next communication. The difficulty of the puzzle changes depending on the required time delay. However, this processing is unproductive for clients and it is not necessary for communication completion. It is only used to keep the client busy for a time. The client-puzzle technique is, therefore, unsuitable for mobile devices that need to minimize processing in order to conserve battery life. In addition, the required delay time is not accurate and differs between devices, because faster devices will find the puzzle’s solution sooner. The time-delay technique also makes many of the proposed puzzles vulnerable to being solved in parallel because the attacker can distribute the puzzle between different machines in order to find the puzzle’s solution in a shorter time [152]. In contrast, the TFTP orders the client to stop the communication for a period of time. This time is shown in the token and cannot be modified by the client. This leaves the client’s machine free to perform other processing jobs while waiting for the delay time to lapse. The delay time is thus accurate for all client machines types. The TFTP also can assign different delay times to different clients.
Table 5.7 Comparing Protocol Functionality

<table>
<thead>
<tr>
<th>Feature</th>
<th>JFKi</th>
<th>HIP</th>
<th>TFTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offers fast filtering</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Carries connection information</td>
<td>Yes</td>
<td>Partial</td>
<td>Yes</td>
</tr>
<tr>
<td>Applies connection control to the packet expiration</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Applies control in the client time delay</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Provides accuracy in client time delay</td>
<td>N/A</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Can apply delay in a particular part of the communication</td>
<td>N/A</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Does not require overhead processing for a client</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Includes a solution for attack detection</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Does not require special software on the client side</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Does not require pre-agreed data</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Can distinguish communication</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Can detect malformed communication</td>
<td>Yes</td>
<td>Partial</td>
<td>Yes</td>
</tr>
<tr>
<td>Provides the sequence of the communication messages</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Detects spoofed attacks</td>
<td>Yes</td>
<td>Partial</td>
<td>Yes</td>
</tr>
<tr>
<td>Has a solution for repeat-resending attack</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Detects repeat-resending flooding attacks in all steps of the protocol</td>
<td>N/A</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Detects the step that receives the DoS attack</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Because the token provides full information about the client and the communication, it can be used for detecting and preventing DoS attacks. The system provides attack detection by taking a filtered token’s log and using it to find the received attack. Token generation can then be used to protect the system receiving the DoS attack. For example, by using TFTP the system can distinguish between communications by assigning different flags in the tokens’ flag parameters.
The TFTP provides a solution for repeat-resending attacks using the Monitoring-Repeated-Communication-List. While the HIP provides a list of the authenticated signatures (so any signature that is repeated can be detected and dropped), this list is limited. If the system is under DDoS attack and the list is full, the system will then be restricted by the list size (Table 5.8). The Monitoring-Repeated-Communication-List solves this problem. It only holds the last previous communications. While the repeated attack is not guaranteed to be detected on the first repetition if the number of clients and attackers is greater than the list size, it nonetheless provides acceptable performance, as shown in the simulation (see Section 5.4.1). The HIP list can only detect repeated attacks in Step 3 of the authentication protocol. On the other hand, the Monitoring-Repeated-Communication-List can be enabled in any step in the protocol; furthermore, it can be enabled in some or all of the communication steps. This helps the system detect repeat attacks in other steps, when the cost of these steps is low but the influence of the attack is high because of the number of attackers or communication messages. The proposed solution uses the Monitoring-Protocol-Behaviour technique to detect the protocol step in which the attack is received (described in Section 4.5.2.2). This technique helps the system determine in which step or steps the Monitoring-Repeated-Communication-List should be enabled.
Table 5.8 Comparison of the HIP and TFTP Lists

<table>
<thead>
<tr>
<th></th>
<th>HIP</th>
<th>TFTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detects repeat-resending attack of all protocol steps</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Detects the repeat-resending attack in Step 3</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Detects repeat-resending attack from the first attack</td>
<td>Yes</td>
<td>‘Not guaranteed’*</td>
</tr>
<tr>
<td>Detects the repeat-resending attack of Step 3 when: No. of attackers is higher than the list size</td>
<td>Yes: for the part of attack until the list is full; No: For the rest</td>
<td>‘Not guaranteed’*</td>
</tr>
<tr>
<td>Detects the repeat-resending attacks of the 4th scenario</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

* Detection of the repeat-resending attack is not guaranteed at the first repetition.

In this section we have discussed the functionality and the efficiency of TFTP. In addition, we have shown the enhancement that the TFT brings to a protocol’s functionality and efficiency. We next discuss the role of the TFT as a communication control.

5.3 Communication-Control Analysis

In this section we analyze the ability of the proposed communication control (TFT) to: (i) filter DoS-attack packets based on their headers and provide more resource availability to a system; (ii) detect spoofed sources; and (iii) detect the use of incorrect tokens. To do this we propose using a formal model. We begin by presenting the new proposed model definitions.

5.3.1 Client-Server Formal-Analysis Model

In this section we develop the new client-server formal-analysis model for use in analyzing and evaluating protocols and solutions in terms of filtering DoS
Definition 5.1. We consider $\Phi$ as the system’s clock. This is the universal time for all the communication parties. Each event thus occurs on $\varphi_i \in \Phi$.

Definition 5.2. We define $\Delta$ as the bound transmission time delay. Any packet should arrive within $\Delta$ or be considered lost.

Definition 5.3. The Client-and–Server specification is a sequence of packets of the form $C \rightarrow S : P$, where $P$ is the packet sent from client $C$ to server $S$.

Definition 5.4. The communication uses $IP_x$, which describes the IP address belonging to $x$, and is the source or destination of a packet.

Definition 5.5. The packet $P$ has the following attributes:

(i) the source of the packet. This attribute contains the value $IP_i$, which is the IP address of the source $i$ where $i$ is a valid IP address;

(ii) the destination of the packet. This attribute contains the value $IP_j$, which is the IP address of the source $j$ where $j$ is a valid IP address; and

(iii) the type of the packet $Y = \{y_{REC}, y_{ACK}, y_{SYN-ACK}, y_{SigReq}, y_{Sig}, \ldots, y_f\}$, where $y_{REC}$ is the type of the requested communication packet, $y_{ACK}$ is the type of $ACK$ packet, $y_{SYN-ACK}$ is the type of the $SYN - ACK$ (which is the final message of the three-way handshake protocol),
$y_{\text{SigReq}}$ is the packet requesting a signature and $y_{\text{Sig}}$ is the message type providing the signature.

**Definition 5.6.** The duration $\Phi$, annotated as $\varphi_{i \rightarrow j} = \{ \varphi_n \in \Phi : i \leq n \leq j \} \subseteq \Phi$.

**Definition 5.7.** The Client-and-Server specification is a sequence of packets of the form $C \rightarrow S : G_1, \ldots, G_k \parallel P \parallel O_1, \ldots, O_n$. The sequence $G_1, \ldots, G_k$ represents the sequence of operations that are performed by $C$ to produce $P$, while the sequence $O_1, \ldots, O_n$ represents the sequence of operations that are performed by $S$ to process and verify $P$.

**Definition 5.8.** Let $L = C \rightarrow S : G_1, \ldots, G_k \parallel P \parallel O_1, \ldots, O_n$ be a line in an annotated Client-Server specification. We say that event $l$ occurs in $L$ if:

- $l$ is one of $G_i$ or $O_j : 0 < i \leq k, 0 < j \leq n$;

- $l$ is ‘$C$ sends $P$ to $S$’ or ‘$S$ receives $P$ from $C$’. In this case $= \{G_1, \ldots, G_k\}$ or $\{O_1, \ldots, O_n\}$.

The events "$G_1, \ldots, G_k$" and "$C$ sends $P$ to $S$" occur at $C$, while "$O_1, \ldots, O_n$" and ‘$S$ receives $P$ from $C$’ occur at $S$. These events can be categorized into two event types: normal and verification. The normal events can occur at either the sender or receiver and have only a success result. On the other hand, the verification events have two potential results: success or failure, and they occur only at the receiver. If the operation succeeds then $S$ continues in the next event.
in $L$. However, if the operation fails, then $S$ does not continue to perform the next event in $L$.

**Definition 5.9.** $C_i$ uses the $cl_{send_{i,j,f,x}}$ event to send the $f$ type packet to server $S_j$ at time $z$ where $z \in \Phi$, and where $i,j$ are valid IP addresses. $S_j$, which applies technique $Tec$, uses the $ser_{recv_{j,i,f,x}}^{Tec}$ event to receive this packet; therefore

$$cl_{send_{i,j,f,x}} = \{G_1, G_2, \ldots, G_k; k \in \mathbb{N}\}$$

and

$$ser_{recv_{j,i,f,x}}^{Tec} = \{O_1, O_2, \ldots, O_n\}.$$ 

In addition, $S_j$ uses the $ser_{send_{j,i,f,x}}^{Tec}$ event to send a packet to $C_i$, and $C_i$ uses the $cl_{recv_{i,j,f,x}}$ event to receive this packet. The set of each event can be defined as the following four sets:

$$A_1 = \{cl_{send_{i,j,f,x}} : i \neq j\} \quad (5.1)$$

$$A_2 = \{ser_{recv_{j,i,f,x}}^{Tec} : i \neq j\} \quad (5.2)$$

$$A_3 = \{ser_{send_{j,i,f,x}}^{Tec} : i \neq j\} \quad (5.3)$$

$$A_4 = \{cl_{recv_{i,j,f,x}} : i \neq j\} \quad (5.4)$$

**Definition 5.10.** Two relationships, $R_1$ and $R_2$, exist between the above four sets. These relationships describe the influences of events in each of the other events. These relationships thus link every two events in which the occurrence of one of them causes an occurrence of the other event as follows:

$$R_1: A_1 \rightarrow A_2 \quad (5.5)$$
This relationship is between the $cl_{send_{i,j,f,z}}$ and $ser_{recy_{j,i,f,z}}^{Tec}$ events. So for each $cl_{send_{i,j,f,z}}$ event there is a $ser_{recy_{j,i,f,z}}^{Tec}$ event, and the $ser_{recy_{j,i,f,z}}^{Tec}$ event does not happen without the occurrence of the $cl_{send_{i,j,f,z}}$ event.

The second relationship can be defined as:

$$R_2: A_3 \rightarrow A_4$$

This relationship is between the $ser_{send_{j,i,f,z}}^{Tec}$ and $cl_{recv_{i,j,f,z}}$ events. So for each $ser_{send_{j,i,f,z}}^{Tec}$ event there is a $cl_{recv_{i,j,f,z}}$ event that occurs.

**Definition 5.11.** $S_j$ uses the $ser_{handle_{j,i,f,z}}^{Tec}$ event to handle a request as follows:

$$ser_{handle_{j,i,REQ,z}}^{Tec} = ser_{recv_{j,i,REQ,z}}^{Tec} + ser_{send_{j,i,ACK,z}}^{Tec}$$

**Definition 5.12.** Let us define $N(l)$ as a function that is used to find the number of required operations of the CPU to process $l$, such as in the following example:

$$N\left( ser_{recv_{j,i,f,z}}^{Tec} \right) = N(O_1) + \cdots + N(O_n)$$

where $n \in \mathbb{N}$.

**Definition 5.13.** Let $\delta(O)$ be the function that calculates the time cost that is required to perform one or multiple operations. $\delta\left( ser_{recv_{j,i,f,z}}^{Tec} \right)$ can then be calculated as:
\[ \delta \left( \text{ser}_{\text{recy}_{j,i.f,x}}^{\text{Tec}} \right) = \delta(O_i) + \cdots + \delta(O_n) \geq 0 \quad (5.9) \]

**Definition 5.14.** We define \( t \) as the minimum time required to process a single operation in the CPU. Therefore:

\[ \delta(O_i) = N(O_i) \cdot t \geq 0 \quad (5.10) \]

From Equation (5.10), we can rewrite Equation (5.9) as follows:

\[ \delta \left( \text{ser}_{\text{recy}_{j,i.f,x}}^{\text{Tec}} \right) = (N(O_i) + \cdots + N(O_n)) \cdot t \geq 0 \quad (5.11) \]

**Definition 5.15.** Let \( r(l, \varphi_{n-m}) \) give the rate for the number of times that \( l \) occurs in \( \varphi_{n-m} \). This function can be calculated as follows:

\[ r(l, \varphi_{n-m}) = \frac{\varphi_{n-m}}{\delta(l)}; \varphi_{n-m} > 0 \quad (5.12) \]

**Lemma 5.1:** \( r(l_i, \varphi_{n-m}) > r(l_j, \varphi_{n-m}) \iff \delta(l_i) < \delta(l_j) \).

**Proof:** Let us say the following is correct:

\[ \delta(l_i) < \delta(l_j) \quad (5.13) \]

Because \( \varphi_{n-m} > 0 \), we can then divide the above formula (5.13) by \( \varphi_{n-m} \) as follows:

\[ \frac{\delta(l_i)}{\varphi_{n-m}} < \frac{\delta(l_j)}{\varphi_{n-m}} \quad (5.14) \]

Now we will take the inverse of the above formula (5.14) as:
\[
\frac{\varphi_{n-m}}{\delta(l_j)} > \frac{\varphi_{n-m}}{\delta(l_i)} = r(l_y \varphi_{n-m}) > r(l_y, \varphi_{n-m})
\]  
(5.15)

and consequently:

\[
\delta(l_i) < \delta(l_j) \Rightarrow r(l_y \varphi_{n-m}) > r(l_y, \varphi_{n-m})
\]  
(5.16)

Let say the following is correct:

\[
r(l_y \varphi_{n-m}) > r(l_y, \varphi_{n-m})
\]  
(5.17)

Based on Equation (5.12) we find the following:

\[
\frac{\varphi_{n-m}}{\delta(l_i)} > \frac{\varphi_{n-m}}{\delta(l_j)} = \varphi_{n-m} \delta(l_i) > \varphi_{n-m} \delta(l_j)
\]  
(5.18)

If we divide the above Equation (5.13) by \( \varphi_{n-m} \), we have the following:

\[
\delta(l_j) > \delta(l_i)
\]  
(5.19)

\[
r(l_y \varphi_{n-m}) > r(l_y, \varphi_{n-m}) \Rightarrow \delta(l_i) < \delta(l_j)
\]  
(5.20)

From Equations (5.16) and (5.20) the following is correct.

\[
r(l_y \varphi_{n-m}) > r(l_y, \varphi_{n-m}) \iff \delta(l_i) < \delta(l_j)
\]  
(5.21)

**Definition 5.16.** Let \( \uparrow l \) describe the maximum continuous-sequence occurrence of \( l \) at a time. For example, \( \uparrow \text{ser}^{\text{recv}_{j,i,f,z}} \) means the maximum repeating request of \( \text{ser}^{\text{recv}_{j,i,f,z}} \) without stopping over a duration of time such as \( \varphi_{n-m}; n \leq z \leq m \).

**Definition 5.17.** The attacker \( k \) uses similar client activities and events to commit a DoS attack on the server \( j \). These activities can be denoted in the
same manner as the client-event specifications but instead they start with a ‘#’ (hash) sign. Thus $\#c_{send_{k,j,REQ}}$ means that the attacker $k$ sends a request packet to the system.

**Definition 5.18.** When $k$ uses a spoofed $i$, this event is annotated in the Client- and-Server specification as $\#c_{send_{\delta i,j,f,z}}$.

These definitions are now used to test the communication efficiency of the TFT in terms of (i) filtering DoS-attack packets based on their headers and thus enabling greater resource availability for a system; (ii) detecting spoofed sources; and (iii) detecting the use of incorrect tokens.

### 5.3.2 Filtering DoS Packets Based on their Headers

In order to check whether a system has the ability to filter packets based on their headers and save processing time, we need to analyze the TFT when the system receives a repeat-resending signature attack.

**Theorem 5.1:** TFT saves time by dropping expired encrypted signature packets in the repeat-resending attacks based on their headers.

**Proof:** Let us say that an attacker $K$ is successful in performing the connection stage with the server $S_0$ (by completing the three-way-handshake protocol), and then performs the $\#c_{send_{k,S_0,sig,z}} \forall \varphi_z \in \varphi_{n-m}$ event where $S_0$ implements the TFT. Based on Definition 5.6 the following is correct:

$$R_1\left(\#c_{send_{k,S_0,sig,z}}\right) = \text{ser}^{TFT}_{recv_{S_0,k,sig,z+\Delta}} \tag{5.22}$$
where \( \text{sig} \in Y \), which is the type of packet the client uses to provide his or her encrypted signature.

From the above equation (Equation 5.22), if the event \(#cl_{send,k,S_0,sig,x}\) occurs then the \( se_{TFT}^{\text{recv}_S,k,sig,x+\Delta} \) event will occur. This means that once the attacker sends a packet to the server using the \(#cl_{send,k,S_0,sig,x}\) event, and this packet arrives at the system after a time \( \Delta \), then the system will receive this packet by using the \( se_{TFT}^{\text{recv}_S,k,sig,x+\Delta} \) event.

Let us say that the token, which is in \(#cl_{send,k,S_0,sig,x}\), is correct for the duration \( \varphi_{n-m} \) where \( n < z < m \). There are two possible cases when processing \( se_{TFT}^{\text{recv}_S,k,sig,x+\Delta} \).

i. The token in \( p_{i,S_0,sig}, \varphi_x \) is not expired until \( \varphi_b \) and \( \varphi_b \geq \varphi_m \). In this case the \( \delta \left( se_{TFT}^{\text{recv}_S,k,sig,x+\Delta} \right) \) can be calculated as follows:

\[
\delta \left( se_{TFT}^{\text{recv}_S,k,sig,x+\Delta} \right) = \delta(\text{TokenValidation}) + \delta(\text{dec(sig)}) \quad (5.23)
\]

where

- \( \text{dec(sig)} \) is the processing required to decrypt the signature. This task requires \( T_{d,sig} \) time for execution; and
- \( \text{TokenValidation} \) is the processing required to validate the received token; it requires \( T_{vToken} \) time for execution.
The number of packets that can be processed at the time $\varphi_{n-m}$ is $N_{p,n-m}$ which can be computed as follows:

$$N'_{p,n-m} = r\left( \delta \left( \text{ser}_{\text{recv}_{S_0,k}^{\Delta}}^{TFT}, \varphi_{n-m} \right) \right) = \frac{\varphi_{n-m}}{T_{vToken}+T_{d.sig}} \tag{5.24}$$

ii. The token in $p_{i,S_0,sig}, \varphi_x$ is valid until $\varphi_b$ and $\varphi_b \leq \varphi_m$. In this case

$\delta \left( \text{ser}_{\text{recv}_{S_0,k}^{\Delta}}^{TFT} \right)$ can be calculated as follows:

For the duration $n \rightarrow b$, $N''_{p,n-b}$ can be calculated as:

$$N''_{p,n-b} = \frac{\varphi_{n-b}}{T_{vToken} + T_{d.sig}} \tag{5.25}$$

For the reminder of time $\varphi_b \rightarrow \varphi_m$ in the $N''_{p,n-b}$ the token will be expired and the system will reject the communication without executing the following operations. $N''_{p,b-m}$ can thus be calculated as:

$$N''_{p,b-m} = \frac{\varphi_{b-m}}{T_{vToken}} \tag{5.26}$$

From Equation (5.25) and Equation (5.26), $N''_{p,n-m}$ can be calculated as:

$$N''_{p,n-m} = \frac{\varphi_{n-b}}{T_{vToken} + T_{d.sig}} + \frac{\varphi_{b-m}}{T_{vToken}} \tag{5.27}$$

If we compare the required time to process $\text{ser}_{\text{recv}_{S_0,k}^{\Delta}}^{TFT}$ for the two cases in Equation (5.24) and Equation (5.27) the result will be as follows:

$$\frac{\varphi_{n-b}}{T_{vToken} + T_{d.sig}} + \frac{\varphi_{b-m}}{T_{vToken} + T_{d.sig}} < \frac{\varphi_{n-b}}{T_{vToken} + T_{d.sig}} + \frac{\varphi_{b-m}}{T_{vToken}} \tag{5.28}$$

$$\Rightarrow N'_{p,n-m} > N''_{p,n-m}. \tag{5.29}$$
This demonstrates that using the TFT in the protocol gives the system the ability to more easily filter encrypted packets through the use of packet headers. Being able to filter packets more easily enhances the system processing rate and contributes to a higher system availability.

### 5.3.3 Detecting Spoofed Sources

To check whether an attacker can communicate with the system using a spoofed source, we analyze this type of communication as follows.

If the attacker $w$ uses a spoofed IP $i$ and tries to communicate with the system, he or she will perform the $\text{#cl}_{sl,ij,REQ,z}$ event. Based on the relationship $R_1$, the system will perform $\text{ser}_{REQ,ij,REQ,z+\Delta}$. At this stage the system does not know that this client is an attacker. The system will next generate a token based on the received packet’s data, $j, i, REQ$ and $z + \Delta$, and will then perform the $\text{ser}_{sendj,LACK,z+\Delta}$ event, which contains the generated token. However this message will not be received by the attacker. Instead, it will go to client $i$, because the attacker does not own this source and spoofs this source to appear in the attacker’s request event. The attacker uses $\$i$, which is different to the attacker’s source. Because of this difference, the attacker cannot continue communicating with the system using a spoofed IP.
5.3.4 Detecting the Use of Incorrect Tokens

Incorrect tokens include spoofed, re-used and modified tokens. A token is generated based on the token’s parameters as discussed in Section 4.2. The system can therefore verify the received token based on these parameters. The system takes these parameters from the token’s keys (which are in the token itself), the received packet, the system’s time and the random-numbers list. When the attacker performs \( \#c_{send_{i,j,f,s}}; f \neq REQ \), the system will check the received token based on the token’s parameters. Changes in any one of the parameters will lead to an incorrect token result in the token verification. The following are the possible scenarios where a token will be different.

i. The source of the packet is not correct. This happens if the token is spoofed.

ii. The stage of the communication is not correct. This happens when the attacker uses a token from a previous stage.

iii. The IP of the server is not correct. This happens when the token is designed to be received by another server’s IP.

iv. The token keys, which are used to calculate the token value, are modified.

Any of these situations will lead to an incorrect token result in the token verification. Thus, under any of these situations, the attacker cannot re-use, spoof or change the token. However, the attacker can re-use a token if the
attacker exactly and completely copies and spoofs the entire packet of a client, including the token. This activity is expensive and takes a longer time to perform [148].

Despite the malicious intent of the attacker, copying and sending the entire packet does not affect the communication of the legitimate client with the system, because under the TFT protocol, the system does not hold the client’s communication information in the system. All the required communication information is stored in the token. So once the system receives the attacker’s packet, the system will reply to the communication normally by generating a new token. In this case, the client will receive two correct replies from the system. The client can use one of the correct responses to continue communication with the system.

In contrast, if a fully copied packet were to be sent to a system that used a stateful backlog queue (as required by some of the existing techniques), then this attack would affect communication between a legitimate client and the server. Because the system would receive an unexpected communication from the client (according to the backlog communication list), the system would then block communication from the client’s source as the communication would appear suspicious.
5.3.5 Summary

In this section we have proposed a formal-analysis model to analyze the TFT. This model has been used to analyze the ability of the TFT in detecting and filtering expired, incorrect and spoofed tokens. The results show that the proposed communication control, TFT, has the ability to filter packets based on their header, detect and prevent spoofed sources, detect and prevent re-used tokens and prevent any change in the token’s value or keys.

In the next section we analyze the correctness of this token implementation method (initially discussed in Section 4.2.1).

5.4 Correctness of the Proposed Token Method

In this section we test the proposed implementation method of the token (Section 4.2.1) in achieving TFT functionality and security goals using a simulation. In addition, the capability of the Monitoring-Repeated-Communication-List in detecting repeated communication is tested. We assign the following testing goals and subgoals for this simulation in order to:

i. Test the proposed implementation method of TFT. To achieve this goal we need to:
   a. Test the correctness and the viability of the proposed method for token generation and verification.
b. Test the protocol’s dependability when using the TFT proposed method to replace the stateful-communication-backlog queue and provide all the stateful features in a full stateless technique.

c. Test the system’s ability to detect spoofed tokens and test whether spoofed packets affect the legitimate communication’s source.

d. Test whether the system can detect expired communication.

e. Test the viability of the method to determine whether it can delay communications and whether it can detect any communications that arrive before the delay time has lapsed.

ii. Test the efficiency of the Monitoring-Repeated-Communication-List in detecting repeat-resending attacks when the number of clients is greater than the size of the list.

Based on the above goals we have designed tests scenarios that can be used to evaluate the proposed solution. The results of these scenarios are discussed in the following section.

5.4.1 Simulation Results

In this section we discuss the results of the simulation. Each result is based on a test scenario, which is designed using a simulation goal. We illustrate and review the results for each simulation goal.
Simulation Goal (i) a: Test the correctness and viability of the proposed method for token generation and verification.

In order to test this goal, we implemented two tests. The first involved monitoring the communication of a client to determine whether the client could complete his or her communication with the system through the three steps based on the token technique. The second involved testing whether the system accepted a client’s communication with a modified token. The results of these two tests are as follows (Figure 5.2).

<table>
<thead>
<tr>
<th>Client IP:11111111113</th>
<th>Client’s Stage:1</th>
<th>Client’s Token’s keys: 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client IP:11111111113</td>
<td>Client’s Stage:2</td>
<td>Client’s Token’s keys: 111111337008</td>
</tr>
<tr>
<td>Client IP:11111111113</td>
<td>Client’s Stage:3</td>
<td>Client’s Token’s keys: 111111337011</td>
</tr>
<tr>
<td>Client IP:11111111113</td>
<td>Client’s Stage:4</td>
<td>Client’s Token’s keys: 111111337014</td>
</tr>
</tbody>
</table>

**Figure 5.2 Completing the Client-Communication Process**

Figure 5.2 shows the output result of the test. The client starts the communication with the system in Step 1. At this stage this client does not have a token, so the token key’s value is zero. The client then moves through the three steps until the client exits the system.

To test whether the system accepts a modified token, we modified the client’s token and obtained the following result from the communication (Figure 5.3).

**Figure 5.3 Detecting Incorrect Tokens**

In Figure 5.3 we can see that the system does not accept an incorrect token.
From the two tests above, we can verify that the TFT method is viable for generating and verifying tokens.

**Simulation Goal (i) b: Test the protocol dependability when using the TFT proposed method to replace the stateful-communication-backlog queue and provide all the stateful features in a full stateless technique.**

From the previous test (Figure 5.3) we can see that the system can communicate with a client, depending on the token, without using any backlog queue. Therefore, by using the TFT, the protocol is a full stateless protocol. To determine whether the protocol provides the stateful features, we need to determine whether the system can detect changes in the state of the communication. We designed a scenario where a client packet state was changed during the communication. We obtained the following result (Figure 5.4).

![Communication dropped: Client IP: lllllllllllf71 Has incorrect Token](image)

**Figure 5.4 Testing Protocol Dependability when Using the TFT**

From this result we can see that the system did not accept the change in the state of the communication, because the state of the communication was included in the token. When the system received the communication and the communication state did not match the token, the result indicated that the system did not accept the incorrect token.
Simulation Goal (i) c: Test the system’s ability to detect spoofed tokens and test whether spoofed packets affect the legitimate communication’s source

To verify the system’s ability to detect spoofed tokens, we designed a scenario that required a client to use a token that was designed for another client. After we ran this scenario, we received the following outcome (Figure 5.5).

![Communication dropped: Client ID: 11111111704 Has incorrect Token](image)

**Figure 5.5 Testing the System’s Ability to Detect Spoofed Tokens**

We can see from the result above that the system did not accept the communication because the token was incorrect. This is because the IP address that was used in the token verification was different from that used for token generation. We also tested whether the system accepted the communication if the attacker spoofed the token and all the packet fields. The system responded by sending the new communication to the legitimate client who owned the IP address. This did not affect or interrupt the communication of the client. On the other hand, if this test were to be implemented in a system that used a stateful-backlog queue, then this attack would affect the client’s communication with the system. In this case, the system would change the state of the client in the backlog queue and the client’s communication would not be accepted, because it would not have a correct state.
Simulation Goal (i) d: Test whether the system can detect expired communication.

To undertake this test, we designed a scenario for a repeating attack without enabling the blocking list. Under this scenario the system receives all the attacker’s packets. However, once the token expired, the system could detect this packet and showed the following result (Figure 5.6).

![Communication dropped: Client IP: 11111111120 It has expired token, new token was sent](image)

**Figure 5.6 Testing Whether the System can Detect Expired Communication**

From the above result, we can see that the system detects expired tokens. The setup of the system, when this scenario was implemented, was such that if the system’s load were equal to or more than 50%, then it would block the expired packet. If the load were less than 50%, it would send a new token to the client. As the system load was less than 50%, a new token was sent to the attacker, but the attacker did not use the new token.

Simulation Goal (i) e: Test the viability of the method to determine whether it can delay communications and detect whether any communication arrives before the delay time has lapsed.

To run this test, we designed a scenario in which the system delayed all communications for one minute. We then made one client send the next communication of the protocol without waiting for the delay time. We obtained the following result (Figure 5.7).
Figure 5.7 Delayed Communication Arrives Before the Delay Time

The above result shows that the system detected a delayed communication that was received before the lapsed delay time. The system then blocked the communication. This demonstrates that by using the proposed communication control, the system can delay communications and detect any that do not wait for this delay time.

Simulation Goal (ii): Test the efficiency of the Monitoring-Repeated-Communication-List in detecting repeat-resending attacks when the number of clients is greater than the size of the list.

To run this test, we designed a new scenario. In this scenario the system received a repeated communication attack in the third step of the authentication stage. The test was performed multiple times by one attacker. The attacker could send four packets at a time. The number of clients varied from being the same as the list size to being 19 times more than the list size. The results of this scenario were drawn from the average of these tests, and are given in Figure 5.8. Based on the results, we can see that the system can detect repeat attacks from the first instance when the number of clients was three times more than the list size. But we can also see that the system cannot guarantee that repeated communication will continue to be detected after a certain number of repetitions.
This section provided an evaluation of the proposed method for implementing the TFT, and it also evaluated the associated Monitoring-Repeated-Communication-List for its ability to detect repeat flooding communication. This method can achieve the security and efficiency goals of the TFT. However, we also need to know the scalability of the system when these techniques are used.

### 5.5 Scalability Analysis

Finally, in order for the proposed protocol to be useful to a system, we need to evaluate whether the protocol is scalable under system parameters. We perform
the scalability analysis of the proposed solution using a formal-based and calculation-based analysis. This includes comparisons with existing techniques.

### 5.5.1 Formal-based analysis

In this section, the scalability of the TFT will be evaluated using the client-server formal-analysis model (Section 5.3.1). This evaluation will include a scalability comparison between the TFT and the stateful-filtering techniques.

**Theorem 5.2:** $S_0$ is a stateless technique if it uses TFT, and the SYN attack will not occur.

**Proof:** The stateful-filtering technique stores the details of each request in the backlog queue, $Q$. This queue has a limited size, $Q_{\text{max.size}}$. In addition, the stateful technique uses the $\text{fetch}^{\text{stateful}}(Q, i)$ function to check the existence of the source $i$ in $Q$. When $i$ is not stored in $Q$, then the $\text{add}^{\text{stateful}}(Q, i)$ function will be used to add $i$ to $Q$. The length of $Q$ at $\varphi_z$ is $Q_{\text{length}}$, which gives the number of entries in $Q$. The SYN flooding attack occurs when $\exists_{\text{ser}_{\text{rec}}^{\text{stateful}}(j, i, \text{REQ}, z)} Q_{\text{length}} = Q_{\text{max.size}}$. If the attacker $k$ wants to commit a SYN attack in $S_0$, he or she performs $\uparrow \# c_{\text{send}_{\text{sl}}, S_0, \text{REQ}, z}$ and $i$ varies from all previous values of $i$. In this case the server will perform $\text{ser}_{\text{rec}}^{\text{stateful}}(j, i, \text{REQ}, z)$. Based on this, the existence of the received communication’s source $i$ will be checked within $Q$ using $\text{fetch}^{\text{stateful}}(Q, i)$. There are three probable outputs for $\text{fetch}^{\text{stateful}}(Q, i)$ as follows:
a) \( i \text{ is in } Q \): In this case, if the communication is repeated the system will drop the communication, and if the communication is not repeated then the system will perform \( ser_{send_{Q_i},ACK_z}^{stateful} \) and then change the state of the communication in the list.

b) \( i \text{ is not in } Q \wedge Q_{\text{length}} < Q_{\text{max.size}} \): In this case the server will perform \( add_{stateful}^{Q_i} \) (Q, i) which adds the communication’s information to Q and then performs \( ser_{send_{Q_i},i}^{stateful} \).

c) \( i \text{ is not in } Q \wedge Q_{\text{length}} = Q_{\text{max.size}} \): In this case the system will perform \( fetch_{stateful}^{Q_i} \) (Q, i) and the result will be that i does not exist in Q; consequently the system will drop the communication or will perform \( add_{stateful}^{Q_i} \) (Q, i). However, this event will fail because the list Q is full.

This case describes the SYN attack.

In the TFT \( \forall \#cl_{send_{S_i,REQ_z}} \) the system performs \( ser_{send_{S_i,REQ_z}}^{TFTP} \), generates a token and then performs \( ser_{send_{S_i,ACK_z}}^{Tec}(p_{S_i,ACK_z}, \varphi) \).

From the above analysis we can see that the stateful technique has the potential to receive a SYN attack, as shown in (c). On the other hand, the TFT obviates the possibility of a SYN attack.

### 5.5.2 Calculation-Based Analysis

In this section we test the scalability of the proposed protocol TFTP with two authentication protocols, JFKi and the HIP. Table 5.9 provides a comparison
between the three protocols for data storage during the communication of the
system with the client.

Table 5.9 Comparing Data Storage of Authentication Protocols

<table>
<thead>
<tr>
<th></th>
<th>JFKi</th>
<th>HIP</th>
<th>TFTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stores data before the communication</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Stores data during the authentication</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Holds data after user is authenticated to detect repeat attacks</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>reason</td>
<td>N/A</td>
<td>Authenticated clients’ signatures list</td>
<td>Last Communication List + Block List</td>
</tr>
<tr>
<td>size</td>
<td>N/A</td>
<td>Not limited</td>
<td>Limited</td>
</tr>
</tbody>
</table>

Table 5.9 shows that none of the three protocols stores any data during the
authentication stage. The TFTP stores limited data before the running of the
protocol. This data is used to generate and verify the tokens. However, this data
is small, limited and has a fixed size. It does not increase based on the number
of clients or communications. After the authentication stage, the HIP stores all
authenticated signatures in a list in order to detect repeat-sending encrypted-
signatures flooding attacks. This list is not limited in size and increases
depending on the number of clients. Once the list is full, the system drops any
new communication. On the other hand, the TFTP provides a last-
communication list. The size of this list is limited and the list is structured as a
queue. If the list is full, the oldest communication will be removed from the list
and a new communication will take its place. Consequently the TFTP is more
scalable. This scalability allows the TFTP to communicate with a larger number of clients at any given time compared to the HIP.

5.6 Conclusion

This chapter analyzed and evaluated the proposed solution and its techniques. The DoS-resistance of the proposed protocol was analyzed using the Meadows’s cost-based framework. The result of this analysis shows that the proposed protocol is DoS-resistant for legitimate users and under seven types of DoS-attack scenarios. In addition, the protocol satisfies the DoS-resistance requirements for both the Meadows and Leiwo et. al. strategies.

The efficiency of the proposed protocol was compared with two DoS-resistant protocols. The results show that the proposed communication control, the TFT, is more efficient than the other protocols. In addition, the proposed protocol is the most efficient protocol for communicating with legitimate users. It is also the most efficient under all attack scenarios but the third. The HIP protocol is more efficient here, because it requires pre-agreed values that can be used to avoid the PKI operation in this scenario. However, the TFTP does not use these kinds of pre-agreed values, because the TFTP is designed for critical government services that use the PKI standard authentication to minimize authentication attacks. Nonetheless, the proposed solution provides a blocking list. If the system detects an incorrect signature under this attack scenario, the
system will block the source of the attack. Therefore the system is vulnerable
to this attack only once per source.

Because the TFTP is designed to meet the security requirements of CGeSS, it
was tested in the HIP protocol where it replaced the client-puzzle technique in
order to compare the functionality, security and performance of the TFT and
the client-puzzle technique. The result shows that the performances of the two
techniques are the same. However, the TFT can delay a client without requiring
any unproductive processing on the part of the client.

The proposed communication-control technique, the Token-Filtering
Technique (TFT), was analyzed using a proposed formal-analysis model. This
model was designed to evaluate the protocols and communication-control
techniques in securing the system against DoS attack. The results showed that
the TFT can be used to filter packets based on their headers, and can enhance
the packet-filtering rate by preventing the system from decrypting packets
belonging to DoS attacks. In addition, the TFT prevents attackers that are using
spoofed IP addresses from communicating with the system after the first step
of the protocol. The TFT also protects the communications, after the first step
of the protocol, from any modified or malformed packets. This includes
spoofed tokens, re-used tokens or any change in the packet’s header or the
token itself. The proposed communication control therefore provides the
security features necessary to enhance the protocol’s DoS-resistance.
The discussion of the TFT was followed by a discussion of the proposed implementation method of the token using a simulation. The results showed that the system can be implemented and the token can replace the stateful-backlog queue and provide all the stateful benefits. The results also showed that the system can detect changes in the token or the packet header’s data. In addition, the system can detect spoofed and expired packets. The simulation also showed that the system can delay communications and detect any communication that does not wait for the specified delay time. The simulation also showed that the Monitoring-Repeat-Communication-List can detect attacks efficiently even if the number of clients is greater than the list size.

The scalability of the proposed solution was tested using the client-server formal-analysis model. The results showed that the system is a stateless technique and does not restrict communication to a limited quantity of clients due to memory size.

As we discussed in Chapter 3, the proposed solution is a real-time solution that provides attack filtering, detection and prevention. These three features are implemented in one defence system (Table 5.10). They use one shared log and are all involved in the decisions; conflict does not arise. This accord comes from integrating these three features. This integration is not provided in the previous, existing solutions. In all our experiments using the proposed solution, we have not found any false positives or false negatives in the TFT generation, verification or filtering processes.
<table>
<thead>
<tr>
<th>Techniques</th>
<th>Functionality</th>
<th>Security Detection</th>
<th>Decision Accuracy</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Packet</td>
<td>Attack Detection</td>
<td>Attack Prevention</td>
<td>Real-time</td>
</tr>
<tr>
<td></td>
<td>Filtering</td>
<td></td>
<td></td>
<td>Processing</td>
</tr>
<tr>
<td>Firewall</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Packet Filtering (Stateless)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Stateful Inspection</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Application Firewalls</td>
<td>Yes</td>
<td>Yes/Low</td>
<td>Yes/Low</td>
<td>Yes</td>
</tr>
<tr>
<td>Application-Proxy Gateways</td>
<td>Yes</td>
<td>Yes/Low</td>
<td>Yes/Low</td>
<td>Yes</td>
</tr>
<tr>
<td>Virtual Private Networking</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>IDS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Misuse</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Anomaly</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Difficult</td>
</tr>
<tr>
<td>IDPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Misuse</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Anomaly</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Difficult</td>
</tr>
<tr>
<td>Unified Threat Management (UTM)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>D-WARD</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Traceback</td>
<td>Yes</td>
<td>Yes/Low</td>
<td>Yes/Low</td>
<td>Yes</td>
</tr>
<tr>
<td>SYN cache</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Queue Management Policy</td>
<td>No</td>
<td>Low</td>
<td>Med</td>
<td>Yes</td>
</tr>
<tr>
<td>Network Access Control</td>
<td>No</td>
<td>No</td>
<td>Low</td>
<td>No</td>
</tr>
<tr>
<td>Attack Graph</td>
<td>No</td>
<td>Low</td>
<td>Med</td>
<td>Low</td>
</tr>
<tr>
<td>JFKi</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>HIP</td>
<td>Yes</td>
<td>Low</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Proposed Solution: HASP-CGeSS</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

* In this technique, the filter position affects the decision accuracy. If a packet filter is applied first (that is, before the other filtering techniques), then the false positives will be high. If a packet filter is applied last (that is, after the other filtering techniques), then the false negatives will be high.
In addition, our test of the Monitoring-Repeated-Communication List (Section 5.5.4) showed that the detection of repeated communication gives zero false-negative decisions when the number of clients is equal to the list size. It also provides almost zero false-negative decisions when the number of clients is three times the doubled size of the list. In addition, using the TFT reduces incorrect decisions in the detection process because it provides more parameters that can be used in detection decisions (see Sections 4.4 and 4.5.3). Further evaluation of the system’s decisions will be done in future work when the detection algorithms are designed.

This chapter proposed a formal-analysis model to evaluate the solutions and protocols for protecting systems against DoS attack. The proposed solution and its techniques were evaluated and compared to the existing solutions using the assessment parameters originally discussed in Chapter 2. The next chapter – the final chapter – summarizes the achievements and contributions of this thesis and concludes with a discussion of the potential for future research in this area.
Chapter 6

Conclusion

The dependency on computer systems to deliver government services is constantly increasing. Government services are important to the well-being of individuals, organizations and society. Loss of these services can cause major disruption, inconvenience and financial loss. Critical eServices must be available at all times for citizens and public and private organizations; protection of these services from any interruption or disconnection caused by Denial-of-Service attack is therefore paramount.

To achieve this protection, security solutions need to satisfy at least four key requirements to guard against DoS attack, as follows.

i. **Functionality**: The solution must operate in real time and not require additional configuration because of the network topology. In addition, it must provide DoS attack filtering, detection and prevention, and these three components must be well integrated.

ii. **Security**: The solution must detect and filter spoofing and repeat-resending communications.
iii. **Decision accuracy**: The solution should have minimal incorrect decisions.

iv. **Performance**: The solution should provide high efficiency and scalability.

None of the existing approaches satisfied all these key requirements. To overcome the limitations of the existing solutions, this thesis proposed a number of security initiatives to satisfy the key requirements.

First, a new security model - DoS-PIF - was proposed. This model achieves the advantages and avoids the disadvantages of the existing models. It is designed to be implemented as a real-time solution and does not require any additional configuration based on the network topology. DoS-PIF comprises three components: DoS-attack filtering, detection and prevention. These components, which are built together in a single defence model, are integrated and involved in all protection decisions. In addition, DoS-PIF incorporates communication controls.

In order to demonstrate the practical value of the proposed model, a security approach, called the Holistic Approach for Securing and Protecting Critical Government eService Systems (HASP-CGeSS), was derived and realized. In order to implement the communication controls in HASP-CGeSS, we proposed a Token-Filtering Technique (TFT). This technique is used as an authentication and filtration layer for packet filtering and is designed to offer fast packet filtering using packet headers. This technique is an essential part of system
communications. The Token-Filtering-Technique Protocol (TFTP) is designed to show, in HASP-CGeSS, the messages between the client and the system.

Based on our evaluation, the proposed solution satisfies the key requirements of functionality, security, decision accuracy and performance necessary for solutions protecting critical government eService systems against DoS attack.

6.1 Research Contributions

The research in this thesis adds to the body of knowledge concerning the protection of government eServices against DoS attack. Its contributions include the following.

- **An analysis of business flows for all services that are provided by online government services**: This analysis maximizes the quality of the solution for protecting government systems by identifying critical services that have high priority in terms of needing protection from disconnection. The analysis also investigates communication vulnerabilities that might be used by attackers to commit DoS attack. We found that services requiring fully authenticated users are the most critical, because they are the most costly service and need to be constantly available.

- **A formal model for analysing critical government eService systems to further specify security solutions**: This model characterizes the critical services by defining the entities, the relationship between these entities, the transactions between these entities, the security requirements of these
services and the relationship between the availability of these services and the system performance. This analytical model gives us the necessary specifications and recommendations to design a security model for protecting critical government eService systems against DoS attack.

- **A new security model based on the formal model**: The new model – DoS-PIF – was designed using the specifications and requirements of the formal model. DoS-PIF achieves the advantages and avoids the disadvantages of the existing models. It provides secure communications for critical services while protecting these systems against DoS attack. DoS-PIF is designed to be implemented as a real-time solution and does not require configuration based on the network topology. The model performs three integrated main tasks: (i) packet filtering; (ii) attack detection; and (iii) attack prevention. In addition, the model enables the use of communication controls.

- **A security approach, called the Holistic Approach for Securing and Protecting Critical Government eService Systems (HASP-CGeSS)**: This approach is accompanied by a realization to demonstrate its practical value. This approach is designed to filter, recognize and avoid DoS attacks in critical government systems. The approach divides the communication point between the client and the server into two communication points. This division is used to: (i) identify the nature of the client’s communication activities with the system; (ii) provide an appropriate
solution for DoS attack in every part of the system; and (iii) to secure every stage of the client’s communication with the system. In addition, the approach provides an analysis component that is responsible for detecting and avoiding potential DoS attack while the client is communicating with the system.

• **A new communication control, called the Token-Filtering Technique (TFT):** This control enables the implementation of the communication controls in HASP-CGeSS. It can be used in the proposed packet-filtering mechanism and so is used as an authentication and filtration layer for packet filtering. This communication control is in a plain-text format. This form supports the packet-filtering process by filtering packets efficiently, based on the controls. In addition, the TFT provides full information about the communication between the server and the client. It also shows any expired communication in order to limit the influence of repeat-resending attacks, and can be used to distinguish between clients by using flags. The TFT can be used to filter packets, even if they are encrypted, based on their headers. It prevents the system from spoofed communications and detects any malicious changes in packet fields. The employment of the TFT in the detection and filtering of DoS attack enhances the accuracy of the system’s decisions. In addition, the TFT enables the system administrator to define filtering, detection and prevention rules that are more detailed and specific. Because of these features, the TFT is a compulsory part of all system
protocol messages in order to enhance system efficiency, scalability and security.

• **A new proposed protocol, called the Token-Filtering-Technique Protocol (TFTP):** The TFTP is designed to provide secure communication between the client and the system in CGeSS when resisting DoS attacks. The use of the TFT as a communication control gives the protocol the following features.

  i. The TFTP provides an efficient stateless-filtering technique by filtering flooding-attack packets based on their headers, which contain tokens. The proposed protocol minimises computation costs, because the used communication controls do not include any cryptographic operations. This exclusion makes the filtering process more efficient. In addition, the TFTP minimises the required processing cost of its messages. Because TFTP is efficient, it conserves resources and creates more availability for the system.

  ii. The TFTP is characterized by the features of stateful communication. All required communication information that the stateful technique holds in its lists is provided by the TFTP.

  iii. The TFTP has high scalability. Because the system does not hold any information belonging to clients, the system does not face any limitations caused by memory size. This eliminates the likelihood of the system receiving any kind of SYN attack.
iv. The TFTP gives the system the ability to delay, and hence control, the communication between the client and the system. The delay is achieved without requiring the client to perform unproductive overhead processing.

v. The TFTP offers the ability to adjust client’s communication timing (such as delaying the next client’s communication) and client attributes. These attributes can be used to improve the system’s decisions when filtering, detecting and preventing DoS attack. They also reduce the number of incorrect decisions in the system.

vi. The TFTP can detect expired communication.

vii. The TFTP can prevent spoofed attacks.

viii. The TFTP can ensure the sequentiality of communications to the client.

ix. The TFTP can detect malicious changes in packet fields.

- Two new list mechanisms designed to detect and block repeat communications: This feature enhances the filtering mechanism of the solution. These lists are not limited by the number of clients communicating with the system, and they are as efficient as normal lists to check repeat communications. These lists can be enabled for one or more steps of the protocol depending on the selected detection-and-prevention strategies.
• **A new protocol-behaviour monitoring technique**: This technique is used to detect the communication step in which a repeat DoS attack is received. This information enables the system to allocate system resources to support DoS-attack detection.

• **A formal-analysis model called the Client-Server Formal-Analysis Model**: This model was developed to evaluate the dependability of the proposed communication control in filtering packets based on their headers and in providing a greater availability of resources for the system. This model can be employed to analyze and evaluate the protocols and solutions used to filter DoS packets.

### 6.2 Future Work

In terms of future research, we plan to implement the anomaly and the misuse detection techniques in the analysis component of the HAPS-CGeSS. Because the TFT offers attributes that can be used in detection decisions, using the TFT in the filtering process can enhance the efficiency and the decision accuracy of the anomaly and misuse detection techniques.

We plan to extend the use of the TFT to mobile devices to further protect service providers from DoS attack. The TFT does not require cryptographic operations and is able to delay communication without involving a client’s device in unproductive overhead processing. These features should enable the use of the TFT for mobile devices and provide protection against DoS attack.
for service providers without compromising the quality of service for mobile users.

In conclusion, this research has proposed a number of innovative security advances to protect critical government systems. The outcomes justify continuing investigation in this domain for the betterment of citizens, organisations, governments and countries.
References


Distributed System Security Symposium (NDSS), San Diego, CA, USA, pp. 8-11, 2009.


[102] M. Sourour, B. Adel, and A. Tarek, "Environmental awareness intrusion detection and prevention system toward reducing false


[132] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Advances in Cryptology, CRYPTO'92*, Santa Barbara, California, USA, pp. 139-147, 1993.


Information Networks: Communications and Multimedia Security


