Enhancing Online Banking Transaction Authentication by Using Tamper Proof & Cloud Computing

Hatim Mohamed Elhag

Submitted for the Degree of Doctor of Philosophy
From the University of Surrey

Institute for Communication Systems (ICS)
Department of Electronic Engineering
Faculty of Engineering and Physical Sciences
University of Surrey
Guildford, Surrey GU2 7XH, UK
2015
© H.M. Elhag 2015
Abstract

The recent information technology development has vastly helped in accelerating and facilitating the banking services and operations in general. In spite of this accelerated development in the banking sector, the risk of invading electronic banking systems is evident. This is manifested in many harmful functions such as unauthorised money transfer, disclosure of client information, denial of online banking services as well as various threats linked with online banking at different lineages especially through authentication of the client online.

This thesis utilizes cloud computing in the banking system from technological and economic perspectives, and the possible benefits that a cloud computing provider gives. The definitions and functions of enterprise architecture both for cloud computing and the financial sector are discussed, then the new architecture model that I developed by merging the cloud and e-banking architectures is thoroughly explained.

This study presents a novel, unique tamper proof USB, sustained with an operating system dedicated to serve the bank’s clients. This device is realised by embedding the bank application in this tamper proof USB while creating an isolation layer in the client’s PC when the client plugs in this USB. The modified operating system platform is based on the puppy Linux operating system. It has the capability to multiplex physical resources at the granularity of an entire operating system while being able to provide isolation between different operating systems.

This tamper proof device is supported by four authentication measures which are; unique tamper proof ID, User account, password and fingerprint with a client secure socket layer. Moreover, I designed two different channels, one with cloud for authentication and transferring an encrypted session key while the other channel is used for communication between the client and the bank after re-authentication accompanied by a one-time password and finger printer image authentication parameter plus session key.

The simulation testbed is used to solve the fundamental flow of the mechanism in sufficient detail, using Network Miner to parse libpcap files to do a live packet capture of the network traffic between cloud provider and the client; using Foglight monitoring tools to utilise the simulated server. Netwalk tools are used to represent the percentage of IP usage and Kali Linux, wireshark for penetration testing.

Key words: online transaction, Security, tamper proof devices, cloud computing, architecture model

Email: H.elhag@surrey.ac.uk
Dedication

This thesis is dedicated to the Almighty God who is the source of all wisdom, knowledge and understanding after that to spirit of my twine Hisham.
Acknowledgments

First and foremost I would like to thank the Almighty God Who by signs and wonders has brought me this far and sustained me throughout the period of my research. It is the guidance and mercy of God that has made this work possible. Special thanks to my supervisors Dr. Haitham Cruickshank and co-supervisor Prof. Zhili Sun for their insights, diligence and direction throughout the duration of my research. I want to specially thank Dr. Haitham Cruickshank for his kindness and support during the course of my research. I would like to thank the members of staff at the Institute for Communication Systems, University of Surrey for the cordial and conducive working environment during the period of my research.

My sincere gratitude also goes to Eng. Khalid Hamad, Haifa Sati, Rayan Shabo and Omeco Company with whom I engaged in fruitful lab and discussions which contributed to further my research. I thank every member of the security club group for the wonderful times we shared doing our meetings and presentations.
Chapter 1 Introduction

1.1 Motivation ................................................................. 1
1.2 Objective .................................................................. 2
1.3 Problem Statement .................................................. 2
1.4 Major Contribution .................................................. 3
1.5 Organization of Thesis .............................................. 6

Chapter 2 Online banking transaction overview and architecture ................. 9

2.1 Short Banking History ................................................... 9
2.2 Traditional Methods of Delivering Financial Services ....................... 10
   2.2.1 Overview of Financial Institutions .......................... 10
   2.2.2 Services Offered by Financial Institutions ............... 11
   2.2.3 Financial Institutions’ Client Interaction ................. 12
2.3 Electronic Banking Methods ......................................... 12
   2.3.1 Types of Electronic Currency ............................... 13
   2.3.2 Patterns of Electronic Banking Services ................. 14
   2.3.3 Electronic Banking Procedure and Sequence .......... 15
2.4 Other Types of Methods Using E-Banking ................................ 17
   2.4.1 Advantages of Electronic Banking Services ............ 19
   2.4.2 Disadvantages of Online Banking Services ............. 19
2.5 Operating System Environment ...................................... 20
   2.5.1 O.S Introduction .................................................. 20
   2.5.2 Problem with Current Operation System ............... 20
4.2.1.2 Homomorphic Properties ................................................................. 70

4.3 Security Semantics ................................................................................. 71
  4.3.1 Statement of Fine-Grained Access Control ........................................ 72
  4.3.2 Private Cloud (bank trusted zone): ....................................................... 73
  4.3.3 Authentication Schemes ................................................................. 73

4.4 Authentication Connectivity Link Between Cloud Provider and Bank Data Base ... 77

4.5 Testbed Performance and Evaluation ..................................................... 79

4.6 Summary ............................................................................................... 87

Chapter 5 Tamper Proof Authentication Solution ........................................ 88

5.1 Client Authentication Traditionally ...................................................... 89
  5.1.1 Client Authentication ................................................................. 89
  5.1.2 Password Authentication .......................................................... 89

5.2. Online Banking and Security Integration ........................................... 90

5.3 Framework .......................................................................................... 92
  5.3.1 Tamper Proof Overview .............................................................. 92
  5.3.2 Tamper Proof Architecture ......................................................... 93
  5.3.3 Tamper Proof USB Flash Partition ................................................ 94
  5.3.4 SCSI RBC Set and BIOS Interrupt Services .................................... 95
  5.3.5 Java Syslinux and the BIOS Interrupt Services ............................... 95
  5.3.6 Advanced Tamper Proof Management ......................................... 95

5.4 Tamper Proof Enforces a High Level of Security ................................... 96
  5.4.1 Technical Descriptions ................................................................. 96
    5.4.1.1 Tamper Proof Data Structure ................................................ 96
    5.4.1.2 Tamper Proof Data Import ................................................... 97

5.5 Online Banking Security Tamper Proof Features ................................ 98
  5.5.1 Transporting the Tamper Proof Information .................................... 98
    5.5.1.1 Tamper proof data encryption ................................................. 98
    5.5.1.2 Tamper proof key ............................................................... 99
  5.5.2 Password Verification ..................................................................... 100
  5.5.3 Digital Signature Verification ....................................................... 102
  5.5.4 Automatic Time Drift Management .............................................. 102

5.6 Operating System Architecture Overview ........................................... 104
  5.6.1 Puppy Security Model ................................................................. 104
  5.6.2 The Process of Dealing with Command ......................................... 104
  5.6.3 Module Loading at Boot up ........................................................... 106
5.6.4 Pup_event_backend ......................................................................................................... 106
5.6.5 Pup_event ....................................................................................................................... 108
5.6.6 Slacko .......................................................................................................................... 110
5.6.7 Puppy security .......................................................................................................... 111

5.7 The Storage Domain ....................................................................................................... 112

5.8 Advanced Configuration and Power Interface (ACPI) Support ...................................... 114

5.9 Network Policy Enforcement ......................................................................................... 115

5.10 Analysis of Potential Attack Vectors ............................................................................. 115

5.10.1 Potential Bugs in the Kernel ...................................................................................... 116
5.10.2 Additional Potential Vectors from Driver Domains (2-stage attacks) ...................... 116
5.10.3 Potential TXT Bypass .................................................................................................. 116
5.10.4 Driver Domain to MBR Attacks ................................................................................ 117
5.10.5 Potential Delayed DMA Attack on Dom (attack mostly from storage domain) .... 117
5.10.6 Potential TXT Bypasses (attack mostly from storage domain) .................................. 117
5.10.7 Resistance to Physical Attacks .................................................................................. 119

5.11 Summary ....................................................................................................................... 120

Chapter 6 Cloud model and tamper Proof experiment result ................................................. 121

6.1 Cloud Scheme to Communicate with the Bank ............................................................... 122

6.2 General Security Test and Result .................................................................................. 123

6.2.1 Simulations and Performance Evaluation ................................................................... 123
6.2.2 Simulation Setup ........................................................................................................ 124
6.2.3 Simulation Result: ...................................................................................................... 126

6.3 Proposed Schemes and Data Security Issues .................................................................. 128

6.3.1 Access Control to Encrypted Data in the Cloud. ......................................................... 128
6.3.2 Data Outsourcing – .................................................................................................... 129

6.4 Cloud Computing as Third Parties (enforcement phase): ............................................ 130

6.4.1 Test Key Generation and Distribution ...................................................................... 131
6.4.2 Cashing Re-Encryption ............................................................................................. 132

6.5 Testbed Connectivity Link Between Cloud Provider and the Bank .............................. 133

6.6 Relation Between Bank and Cloud Provider ................................................................. 136

6.6.1 Enterprise Architectures: ......................................................................................... 136
6.6.2 The New Financial Cloud Reference Model ............................................................... 139

6.7 Tamper Proof Security Overview and Experiment Results ........................................... 143

6.8 Scalability Issues ........................................................................................................... 145

6.9 Overview of Attacks ...................................................................................................... 146
6.9.1 Malware or Malicious (and Other Malicious Code-Based Redirection Schemes) ........146
6.9.2 Recording Software Keyboard Buttons .................................................................147
6.9.3 Scams. Phishing Websites/Emails ............................................................................149
6.9.4 Test for Web Storage SQL Injection ........................................................................149
6.9.5 Man In The Middle (MITM) Hacking Online Banking and Credit Card Transactions ....150
6.9.6 Test for Brute Force Protection .................................................................................151
6.9.7 Test for Authentication .............................................................................................151

6.10 Summary ....................................................................................................................152

Chapter 7 Conclusion and Future Work .............................................................................153
7.1 Conclusion ....................................................................................................................153
7.2 Overview of Key Contributions .....................................................................................153

Bibliography .....................................................................................................................158
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Contribution Diagram</td>
<td>05</td>
</tr>
<tr>
<td>5.2</td>
<td>Four Main Stages for Online Banking</td>
<td>16</td>
</tr>
<tr>
<td>5.3</td>
<td>The Main Reasons for Deciding Not to use mobile Payment</td>
<td>18</td>
</tr>
<tr>
<td>5.4</td>
<td>Provisioning For Peak Load and Underproviding</td>
<td>26</td>
</tr>
<tr>
<td>5.5</td>
<td>Security Certificate Alert</td>
<td>36</td>
</tr>
<tr>
<td>5.6</td>
<td>Attack Graph S.phish</td>
<td>37</td>
</tr>
<tr>
<td>5.7</td>
<td>Banking Attack Method</td>
<td>39</td>
</tr>
<tr>
<td>5.8</td>
<td>Counterfeiting Operates</td>
<td>43</td>
</tr>
<tr>
<td>5.9</td>
<td>HSBC Bank Post Security Adverts</td>
<td>45</td>
</tr>
<tr>
<td>5.10</td>
<td>Pack Secure SSL Session</td>
<td>46</td>
</tr>
<tr>
<td>5.11</td>
<td>Finger Print Characteristics</td>
<td>51</td>
</tr>
<tr>
<td>5.12</td>
<td>Security PKI Certificate Flow</td>
<td>54</td>
</tr>
<tr>
<td>5.13</td>
<td>Vasco Announcement</td>
<td>61</td>
</tr>
<tr>
<td>5.14</td>
<td>Vasco Products</td>
<td>61</td>
</tr>
<tr>
<td>5.15</td>
<td>Gemalto Products</td>
<td>62</td>
</tr>
<tr>
<td>5.16</td>
<td>Gemalto Ezio</td>
<td>63</td>
</tr>
<tr>
<td>5.17</td>
<td>Bank Application Interface</td>
<td>66</td>
</tr>
<tr>
<td>5.18</td>
<td>Client Authentication Factor Flow</td>
<td>67</td>
</tr>
<tr>
<td>5.19</td>
<td>Three Participant Entities</td>
<td>72</td>
</tr>
<tr>
<td>5.20</td>
<td>The Real Time and Delay Time</td>
<td>79</td>
</tr>
<tr>
<td>5.21</td>
<td>Offset and All The Text Encrypted</td>
<td>80</td>
</tr>
<tr>
<td>5.22</td>
<td>Packet Size and the Distribution Between Puppy, Cloud and the Bank</td>
<td>80</td>
</tr>
<tr>
<td>5.23</td>
<td>Time Comparison</td>
<td>84</td>
</tr>
<tr>
<td>5.24</td>
<td>Authenticated Process in Tamper Proof Device</td>
<td>91</td>
</tr>
<tr>
<td>5.25</td>
<td>Tamper Proof Hardware Used For Authentication</td>
<td>93</td>
</tr>
<tr>
<td>5.26</td>
<td>Tamper Proof Architecture</td>
<td>94</td>
</tr>
<tr>
<td>5.27</td>
<td>Import Tamper Proof Data Flow Chart</td>
<td>95</td>
</tr>
<tr>
<td>5.28</td>
<td>Snapshot of the Code to Encrypt the SDK File</td>
<td>96</td>
</tr>
<tr>
<td>5.29</td>
<td>Represents 3 DES Storage keys to Encrypt the Tamper Proof Data</td>
<td>98</td>
</tr>
<tr>
<td>5.30</td>
<td>Password Verification Routine</td>
<td>100</td>
</tr>
<tr>
<td>5.31</td>
<td>Message Appears to Client if the Tamper Proof ID is Wrong</td>
<td>101</td>
</tr>
<tr>
<td>5.32</td>
<td>Boot Time in Milliseconds</td>
<td>105</td>
</tr>
<tr>
<td>5.33</td>
<td>Puppy Linux Boot Up</td>
<td>106</td>
</tr>
<tr>
<td>5.34</td>
<td>_Event Backend Functionary</td>
<td>107</td>
</tr>
<tr>
<td>5.35</td>
<td>Link Between Pup_Event_Backend and Pup_Event_Fontend</td>
<td>108</td>
</tr>
</tbody>
</table>
Figure 5.1: Puppy Display.................................................................109
Figure 5.14: Loading Time in Milliseconds......................................110
Figure 5.15: Puppy Slack View........................................................110
Figure 5.16: Puppy Linux Installs New Process Security Attributes ....111
Figure 5.17: USB Drive that Tries to Exploit a Potential Bug in the USB Driver.........................................................113
Figure 6.1: Cloud Bank Infrastructure..............................................122
Figure 6.2: Packet Capture of the Network Traffic Between Cloud Provider and the Client.....................................................123
Figure 6.3: Time Comparison............................................................127
Figure 6.4: Bytes Comparison...........................................................128
Figure 6.5: Top IP Hosts.................................................................129
Figure 6.6: Network Traffic Between Cloud Provider and Client........130
Figure 6.7: The Packet Size Distribution............................................131
Figure 6.8: Snap Shot from Client PC...............................................135
Figure 6.9: Financial Institution Architecture....................................137
Figure 6.10: Cloud Architecture Domain........................................138
Figure 6-11: Financial Cloud Reference Model..................................140
Figure 6-12: Proposal for Financial Cloud Reference Model Area......141
Figure 6.13: Kernel and Interface Link.............................................144
Figure 6.14: Concurrent Processes/ O.S ........................................145
Figure 6.15: Server Response / Error...............................................147
Figure 6.16: Fingerprint Mechanise................................................148
Figure 6.17: Scapshout Source Code ...............................................149
Figure 6.18 Traceroute Process ......................................................150
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Comparison Between Linux and Windows</td>
<td>23</td>
</tr>
<tr>
<td>2.2</td>
<td>Network configuration setup for our modify puppy Linux O.S</td>
<td>26</td>
</tr>
<tr>
<td>2.3</td>
<td>Competitive matrix</td>
<td>27</td>
</tr>
<tr>
<td>3.1</td>
<td>Applicability of Attacks in Different Authentication Mechanisms</td>
<td>47</td>
</tr>
<tr>
<td>3.2</td>
<td>Current Solutions Functionality and Ranking to Solve the Banking Problem</td>
<td>53</td>
</tr>
<tr>
<td>4.1</td>
<td>Summary Result Table For Non Spoofing Scenario</td>
<td>85</td>
</tr>
<tr>
<td>5.1</td>
<td>Comparison of O.S. Performance Test</td>
<td>105</td>
</tr>
<tr>
<td>5.2</td>
<td>Web Site Loading Time in Milliseconds</td>
<td>108</td>
</tr>
<tr>
<td>6.1</td>
<td>Simulation Result</td>
<td>125</td>
</tr>
<tr>
<td>6.2</td>
<td>Overall Packet Size Distribute</td>
<td>132</td>
</tr>
</tbody>
</table>
**Glossary of Terms**

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACPI</td>
<td>Advanced Configuration and Power Interface</td>
</tr>
<tr>
<td>ACS</td>
<td>Assertion Consumer Service</td>
</tr>
<tr>
<td>AHPE</td>
<td>Additive Homomorphic Probabilistic Public Key Encryption</td>
</tr>
<tr>
<td>AML</td>
<td>ACPI machine language</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ARQ</td>
<td>Automatic Repeat Request</td>
</tr>
<tr>
<td>CA</td>
<td>Certificate authorization</td>
</tr>
<tr>
<td>CFSDK</td>
<td>characteristic fingerprint software Developer kits</td>
</tr>
<tr>
<td>COS</td>
<td>Class of services</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CR</td>
<td>Challenge Response</td>
</tr>
<tr>
<td>CRE</td>
<td>Cashing re-encryption</td>
</tr>
<tr>
<td>CSP</td>
<td>Cloud Service Provider</td>
</tr>
<tr>
<td>CT</td>
<td>Custody Transfer</td>
</tr>
<tr>
<td>DDoS</td>
<td>Distributed Denial of Service</td>
</tr>
<tr>
<td>DMA</td>
<td>Direct memory access</td>
</tr>
<tr>
<td>ECC</td>
<td>Elliptic Curve Cryptography</td>
</tr>
<tr>
<td>FSDK</td>
<td>fingerprint software Developer kits</td>
</tr>
<tr>
<td>FAT</td>
<td>File Allocation Table</td>
</tr>
<tr>
<td>FAR</td>
<td>false acceptance rate</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>FRR</td>
<td>false rejection rate</td>
</tr>
<tr>
<td>GSM</td>
<td>Global System for Mobile Communications</td>
</tr>
<tr>
<td>GRUB</td>
<td>default boot loader and manager for Ubuntu</td>
</tr>
<tr>
<td>HMAC</td>
<td>Hashed-Message Authentication Code</td>
</tr>
<tr>
<td>ICMP</td>
<td>Internet Control Message Protocol</td>
</tr>
<tr>
<td>IND- CCA2</td>
<td>Indistinguishability- adaptive chosen-ciphertext attacks</td>
</tr>
<tr>
<td>IR</td>
<td>Information Retrieval</td>
</tr>
<tr>
<td>KDC</td>
<td>key-distribution center</td>
</tr>
<tr>
<td>LSM</td>
<td>Linux security management</td>
</tr>
<tr>
<td>MBR</td>
<td>master boot record</td>
</tr>
<tr>
<td>MFN</td>
<td>machine frame number</td>
</tr>
<tr>
<td>MITM</td>
<td>Man in the middle</td>
</tr>
<tr>
<td>NFAT</td>
<td>Network Forensic Analysis Tool</td>
</tr>
<tr>
<td>OATH</td>
<td>Open authentication</td>
</tr>
<tr>
<td>OLTP</td>
<td>On-Line Transaction Processing</td>
</tr>
<tr>
<td>OSDB</td>
<td>Open Source Database Benchmark suite</td>
</tr>
<tr>
<td>OTG</td>
<td>USB On-The-Go</td>
</tr>
<tr>
<td>OTP</td>
<td>One time password</td>
</tr>
<tr>
<td>PFN</td>
<td>physical frame number</td>
</tr>
<tr>
<td>RBC</td>
<td>Reduced Block Command</td>
</tr>
<tr>
<td>SAML</td>
<td>Security Assertion Markup Language</td>
</tr>
<tr>
<td>SCSI</td>
<td>Small Computer System Interface</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kits</td>
</tr>
<tr>
<td>SLA</td>
<td>Services level agreement</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>SOC</td>
<td>system on chip</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Sockets Layer</td>
</tr>
<tr>
<td>STS</td>
<td>Secure Transaction Sharing</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>TPSA</td>
<td>Tamper Proof Security Area</td>
</tr>
<tr>
<td>TPS</td>
<td>Tamper proof security area</td>
</tr>
<tr>
<td>TXT</td>
<td>Trusted Execution Technology</td>
</tr>
<tr>
<td>TXZ</td>
<td>Standard packet format</td>
</tr>
<tr>
<td>Wi-Fi</td>
<td>Wireless Fidelity</td>
</tr>
</tbody>
</table>
List of Publications

Conference Publications


Chapter 1

Introduction

The Internet has become an important banking sector feature. It is the bank tool for various services provided by many sectors including the financial sector. Banks started to use the internet not only as an innovative payment method for increasing client convenience, but also as a way to reduce costs and enhance profits.

The term Electronic Banking (e-Banking) or Internet Banking is an advanced and comprehensive concept that emerged in the early nineties as a concept of financial services by remote control [32, 38], remote electronic banking or home banking. Clients require the bank to create services with restricted access. The bank provides the client with a package of software—either free or for a fees and this enables the client to use bank services at a certain distance. Alternatively the client purchases a package of software known as Personal Financial Management software (PFM). Hence the concept of financial services is based on distance PC banking between the Bank and the client’s personal computer.

Other Electronic banking, also known as Electronic Funds Transfer (EFT), is simply the use of electronic means to transfer funds directly from one account to another, rather than by cheque or cash. Electronic funds transfer can be used to:

- Pay cheques deposited directly into your bank or credit union checking account.
- Withdraw money from your current account or an ATM machine with a personal identification number (PIN), for convenience, at any time.
- Instruct bank or credit union to automatically pay certain monthly bills from a specific account, such as auto loan or mortgage payment.
- To transfer funds each month from your checking account to mutual fund account.
- To deposit benefits cheque or tax refund deposited directly into cheque account.
- Buy groceries, gasoline and other purchases at the point-of-sale, using a check card rather than cash, credit or a personal check.
- Use a smart card with a prepaid amount of money for use instead of cash at a pay phone, expressway road toll, or on college campuses at the library's photocopy machine or bookstores.
• Use computer and personal finance software to coordinate the total personal financial management process, integrating data and activities related to income, spending, saving, investing, recordkeeping, bill-paying and taxes, along with basic financial analysis and decision making.

1.1 Motivation

The main motivation for this research is to propose and develop lightweight (processing power, memories, performance’s) mechanisms and protocol that can protect the online banking account, control access and react quickly, accurately against online banking authentication attacks. The proposed mechanisms should make the online banking transaction services resilient against any intruders by protecting all the resources such as client identity, bandwidth, communication channel and the banking application. The common security goals are ensuring end-to-end data integrity and confidentiality.

1.2 Objective

The main objectives of the thesis can be described as follows:

• Study the financial sector and understand characteristics, development and the relation between business and information technology applications.

• Detail analyses of the existing electronic banking mechanisms such ATM, online banking, telephone/IVR which provide new challenges such as intruder, hijacking and other cybercrime methods.

• Analyse the security and networking requirements for online banking and evaluate the security, resilience authentication, disruption and account attacks.

• Propose a new method for mitigating authentication attacks on electronic banking by using the virtualisation technique and public/private key cryptography.

• Create a new architecture model to clarify the heterogeneity between the bank’s existing models and cloud existing models to invent a new homogenous architecture based on the similarity between the two models to facilitate the banking sector when using cloud computing technology.

1.3 Problem Statement

Studying the strengths of the security in the financial sector and cybercrime in general, obviously cybercrime still exists and the risk to the financial sector is becoming worse [28, 32], especially on the client side [7]. Security concerns still exist although the topography of banking has changed dramatically with the initiation of online banking as a new approach to running the banking business. However banks’ clients are seeking more comfort in accessing banking
facilities and expect their banks to deliver value added services through channels like online banking.

Most authentication methods used at the moment like static password, Soft-token Certificate/SSL-TLS, hard-token Certificate/SSL-TLS, one-time Password/Time-based Code Generator and Challenge-response [32] can be bypassed using a man-in-the-middle attack [41,53]. In order to combat security breaches associated with online banking, a number of techniques mitigate the intrusion attacks on various online banking activities and many solutions are proposed and implemented but unfortunately online banking still suffers from unauthorized access to clients’ accounts and the loss of large amounts of money.

This situation of the banking authentication method makes it a requirement for banks to adopt and leverage another new approach for success in the performance of internet banking.

Online banking is not merely a technical IT aspect but rather a whole business approach. This encompasses internet banking applications such as funds’ transfer, bill payment and other services together with online accounting and financial management services, which are beyond the scope of the technical application, adding value to the clients. Subsequently, banks need a confirmed identity for the claimant to guarantee the right person gets access to that account.

1.4 Major Contribution

The growing demand for a consistent and secure online banking system with the need to have security solutions that incorporate the concept of online banking necessitated the contribution in this thesis. The novel work undertaken in this thesis was tailored towards addressing the problems discussed in the previous section. In this work, the following contributions are made:

1. **Propose and Implement the New Secure Transaction Sharing (STS) Protocol**

   Secure Transaction Sharing (STS) protocol using additive homomorphic encryption as an operation performed on a set of ciphertexts such that decrypting the result of the operation is the same as the result of some operation performed on the plaintext access. Use homomorphic cryptographic for re-encrypting the message over a cloud provider without needing to know the plain text transfer from the client to the bank.

2. **Study and Develop a Trusted Operating System**

   Study and develop an extension of the existing puppy Linux operating system by rebuilding the puppy Linux boot loader framework utilized to transfer measurement values from Application to Kernel. Several mechanisms were implemented for communicating between application and kernel, such as a grant table, ring buffers, an event channel, and Store.
3. **Realisation of a Tamper Proof Authentication Device**
Design and implement an authentication Tamper proof device for extremely high security demands and confidentiality of the online banking application using the biometric security technology system. Therefore, we add a java code to our lightweight application domain to support web browser application request for authentication parameters, which is part of the online banking system.

4. **Development of a Reference Architecture of Framework**
Propose and implement a reference architecture framework. The reference architecture model for cloud computing might change to satisfy what the financial sector is looking for. Defining building blocks and outlining foundations to support coordination and comprehension between cloud provider and financial sector.

5. **Simulations and Performance Evaluation**
The function of simulation is to solve the fundamental flow of the mechanism and equation in sufficient detail. Once this is achieved, we can predict the consequences of change in operating practice. This will assure the model accuracy, and achieve better results in identifying bottlenecks. Therefore Simulation is an excellent presentation and auditing tool.

Using Network Miner to parse libpcap files to do a live packet capture of the network traffic between cloud provider and the client; using Foglight monitoring tools to utilise the research server. Use Netwalk tools to represent the percentage of IP usage and Kali Linux, wireshark for penetration testing.

In figure 1.1 the Contribution diagram shows the major contribution combined with the highlight point for each element of that contribution flowing sequentially.
Chapter 1. Introduction

Figure 1.1: contribution diagram

Chapter 4
- Describe and discuss the three entities combined to create connectivity for online banking application
- Study Paillier additive Homomorphic cryptosystem
- STS protocol and Algorithm
- Authentication scheme is the process of validating and authentication

Chapter 5
- Present the operating system model,
- design the security frames for the online banking application scenario
- process of dealing with command
- the module loading at boot up

Chapter 5
- Tamper proof hardware component
- Combination of hardware and software
- Divide tamper proof into three stages
- discuss the simulation results of tamper proof

Chapter 6
- Study the technological components of enterprise architecture
- Represent the technologies that support end-user interactions
- Define building blocks and outlining foundations
- Design reference architecture according to the target operating model requirements.
- Consequently, design reference architecture with flexibility to allow plug-and-play approach towards new distribution channel products through Cloud integrated solutions.

Chapter 6
- Investigate efficient methods for handling user access rights
- Using Network Miner to parse libpcap files
- Using Foglight monitoring tools to utilise the research server
- Netwalk tools in our system function to represent the percentage of IP usage
1.5 Organization of Thesis

The main focus of this thesis is on providing an access control system that addresses authentication and authorization issues while taking the identified limitations in the online banking transaction environment into consideration. The remaining part of the thesis is organized as follows:

**Chapter 2,** state of the art and related work: Overview of Financial Institutions Banking history: This chapter defines financial institutions and gives a historical background including the development of this sector. Illustrating the major type of financial institution and the services offered by financial institutions and customer interaction through new modern technology, this chapter also mentions the electronic banking operating system method and the type of electronic currency in addition to patterns of electronic banking services; it explains the advantage of this technology and disadvantage and provides a cloud definition.

**Chapter 3,** Security and Access Control

Reviews security from the general network and online banking perspectives as well as accesses control and its implementation in disruptive electronic banking concepts and attack tree, Trojan methods: this chapter analyses the potential risk and explains the most common way used by hackers to penetrate the financial sector and explores modern piracy tools. Also it gives a detailed description of the threat to personal data like phishing, farming and how to frog the address through DNS. After that it clarifies the efficiency of the current solution to mitigate these risks, with a slight definition of Homomorphic cryptography and Pillar cryptosystem.

**Chapter 4,** new financial model and definition

This chapter defines and describes the three entities combined to create a connectivity model of our design proposal for online banking application then a prototype of the appliance and the topology. This chapter reviews the Secure Transaction Sharing (STS) proposal and problem statement (Fine-grained access control). Also this chapter introduces and describes authentication schemes and the authentication code.

**Chapter 5,** Authentication concept model

Tamper proof security is the topic of this chapter; with the adoption of dynamic cloud services boundary leading the network, the system and application of the financial sector is to be controlled by the service provider domain. To compensate for the loss of control of the security boundaries the financial sector should rely on other high-level control. This chapter proposes control manifesting as strong authentication and authorization based upon different identity factors and identity access.
Chapter 6.

This chapter defines the enterprise architecture concept and the necessity of enterprise architecture to any global organization, with the essential background of enterprise architecture and related work. After that it explains IT architecture components and financial architecture plus cloud computing architecture. Then a new model merges these architecture models to bring a new perspective model to be implemented in the financial sector when using cloud computing technology. For the tamper proof model experiment, this chapter provides a testbed of the work done and illustrates our work and result.

Chapter 7

Contribution and future work.
Chapter 2 Online banking transaction overview and architecture

Background Study of online banking and technical environment

Banking has witnessed a large number of transformations in the past and up to the high tech digital world of today where systems play the role of stewards to personal valuables. The focus of this chapter is: (1) overview of financial institutions banking history, disruption and development as well as electronic banking, (2) traditional methods of delivering financial services, (3) technology used to offer banking services and advantages and disadvantages of electronic banking. The chapter is organized as follows: section 2.1 discusses banking history, development as well as examples of traditional methods of delivering financial services. Section 2.2, 2.3 and section 2.4 present an overview of traditional methods of delivering financial services and electronic banking method. Section 2.5 is an overview of operating systems and section 2.6 cloud computing definition with 2.7 as a summary.

2.1 Short Banking History

In 1156, in Genoa, occurred the earliest known foreign exchange contract [34, 38]. In 1440, Gutenberg invented the modern printing press although Europe already knew of the use of paper money in China. In 1455 the Turks overran the Serbian silver mines, and in 1460 captured the last Bosnian mine. The last Venetian silver gross was mined in 1462. Even the smallest of small change became scarce [35]. This is the point in time where the beginning of modern banking can be traced. In 1472 the Monte deiPaschi di Siena bank was founded, the oldest current bank in the world. At that time moneychangers were already called bankers, though the term "bank" generally referred to their offices, and did not transmit the meaning it does today. In 1565 the London Royal Exchange was established, marking the beginning of a new concept [35].

In the 20th century the banking system all over the world saw a rise and fall, including the 1929 Wall Street Crash, when 9,000 banks closed, wiping out a third of the money supply in the United States [56], then being heavily influenced by the political situation in the First and Second World Wars. In the last quarter of the 20th century, computer technology was embedded in the financial sector to aid banks. As a result, this generated profitable times and banks did lucratively well during the late 1990s. When the tech bubble burst, it precipitated a string of new legislation to prevent conflicts of interest within banks. Investment banking research analysts had been actively promoting stocks to investors while privately acknowledging they were not attractive investments. These kinds of scandals spread across America in the late 1990s [9].
Chapter 2: Background Study of online banking and technical environment

2.2. Traditional Methods of Delivering Financial Services
The 21st century was marked by three major events in the banking world:

- Great adoption of computer networks, that provided new ways of communication, data integration and a larger resource pool.
- Rise to power of developing countries like India, China and some Middle East countries due to an increase in international capital flow.
- Washington Mutual collapses (2008), the largest bank failure in history that propagated through the market and started the chain reaction of a global recession.

The latest was based on the speculative bubble in housing prices within the American market. In March, 2008, the Federal government began using a variety of bailout measures to prop up other firms from collapsing, but the measurements came too late therefore the crisis manifested at a global level.

2.2.1 Overview of Financial Institutions
Finance is defined as “the science of funds management” [1]; therefore the main role of financial institutions is to act as financial intermediaries. There are three major types of financial institutions: [37]

- Deposit-taking institutions that accept and manage deposits and make loans, including banks, building societies, credit unions, trust companies, and mortgage loan companies
- Insurance companies and pension funds
- Brokers, underwriters and investment funds.

Most financial institutions are generically known as banks, but this term accumulates many types of institutions and services, some of them obviously far from what a bank was initially created to be. Banking activities can be classified as follows: [33]

- Retail banking: dealing directly with individuals and small businesses. The types of banks under this category are: Commercial banks, Community banks, Community development banks, Credit unions, Postal savings banks, Savings bank, Ethical banks, Direct or Internet-Only banks.
- Business banking: providing services to mid-market business
- Corporate banking: directed at large business entities
Chapter 2: Background Study of online banking and technical environment

- Private banking: Have high net worth individuals and families as main clientele, providing them with wealth management.
- Investment banking: related to activities on the financial markets, banks which provide capital to firms in the form of shares rather than loans
- Islamic banks are differentiated by the fact that all banking activities must avoid interest, a concept that is forbidden in Islam. [38]
- Conglomerates are financial service firms that are active in more than one sector of the financial services market.
- Financial services’ companies are corporate entities that provide both banking and insurance
- Central banks: government-owned banks.

All financial institutions share similar core values and services, but set themselves apart using additional domain-specific products.

2.2.2 Services Offered by Financial Institutions

The notion of "financial services" has been in use largely due to the Gramm-Leach-Bliley Act [1] of the late 1990s. Two separate service types emerged:

- Primary operations of keeping money safe and allowing withdrawals; issuance of checkbooks, credit cards and other payment instruments; providing personal loans, commercial loans, mortgage loans and overdraft agreements for the temporary advancement of the bank’s own money to meet monthly spending commitments of a customer in their current account; notary service for financial and other documents [15].
- Other services like foreign exchange services; wealth and asset management; tax planning; capital market services; hedge fund management; custody services; insurance and annuities, life insurance, retirement insurance, health insurance, property and casualty insurance; reinsurance; intermediation or advisory services and angel investment [10].

All these types of services have been influenced by countless evolutions in human society and therefore are highly dependent on customer interaction. For example, foreign exchange services were invented during the colonization times when the need for local currency was more frequent and the loan was invented in the medieval trade fairs of Europe. In more modern times, the health insurance phenomenon as it is understood today began during the Civil war (1861-1865) in the United States, offering coverage against accidents related to train or steamboat travel [34].
2.2.3 Financial Institutions’ Client Interaction

A bank has no purpose unless the market demands its existence. Also a service provided by a financial institution needs to find its customer niche market so, over time, different channels have been put in place to facilitate customer interaction [4, 10.11]. The main channels today are:

- Branches, ATMs,
- Call centers, Mail services,
- Telephone banking, Mobile banking, On-line banking, Video banking
- Relationship managers.

These channels demand not only financial, human and technological investment but also the know-how. This is why financial institutions need specialized assistance.

Bankers are good at handling money and predicting capital investments, but there is a gap that needed to be filled: the technical knowledge to make this complex entity run smooth and flawlessly.

Banking emerged with concepts such as electronic commerce (E-commerce) and online banking services, which in turn led to facilitating flexible and quick and easy financial transactions leading to economic benefits in reducing the time and money spent on obtaining services.

2.3 Electronic Banking Methods

- Automated Teller Machines (ATMs), also called 24-hour tellers, are electronic terminals which give consumers the opportunity to bank at almost any time. To withdraw cash, make deposits or transfer funds between accounts, a consumer needs an ATM card and a personal identification number. Some ATMs charge a usage fee for this service, with a higher fee for consumers who do not have an account at their institution. If a fee is charged, it must be revealed on the terminal screen or on a sign next to the screen.

- Direct Deposit and Withdrawal Services allow consumers to authorize specific deposits, such as paychecks or social security checks, to their accounts on a regular basis. It is also possible to authorize the bank, for a fee, to withdraw funds from your account to pay your recurring bills, such as mortgage payment, installment loan payments, insurance premiums and utility bills.

- Phone Systems let consumers phone their financial institutions with instructions to pay certain bills or to transfer funds between accounts.
Chapter 2: Background Study of online banking and technical environment

- Point-of-Sale Transfer Terminals allow consumers to pay for retail purchases with a check card, a new name for a debit card. This card looks like a credit card but with a significant difference — the money for the purchase is transferred immediately from your account to the store's account. You no longer have the benefit of the credit card "float", that is the time between the purchase transactions and when you pay the credit card bill. With immediate transfer of funds at the point-of-sale, it is easy to go overdrawn on your checking account and incur additional charges unless you keep careful watch on spending.

- Personal Computer Banking Services offer consumers the convenience of conducting many banking transactions electronically using a personal computer. Consumers can view their account balances, request transfers between accounts and pay bills electronically from home.

2.3.1 Types of Electronic Currency

Check cards, the new name for debit cards, can be used instead of cash, personal cheques or credit cards. As stated, when you use a check card you transfer funds immediately from your account to the store's account. A growing number of consumers use check cards because they eliminate the hassle and risks of writing cheques or carrying large amounts of cash; important facts are: [15]

- You have less bargaining power with a check card than with a credit card. With a credit card you have the right to refuse to pay for the purchase if you are not satisfied. With a debit card you have already paid for the product, so you have less bargaining power with the merchant.

- A thief with your check card and PIN number can take all the money in your account. The thief can even make point-of-sale purchases without your PIN.

- Your liability is limited to a fixed amount if you report the check card lost within two days, any longer and your liability can increase. After a certain time you can be responsible for the entire amount.

- In an era of increasing bank fees, consumers can expect to pay for the service of using a check card.

- It is the consumer's responsibility to keep check card receipts and deduct the currency amounts of the purchase from your bank balance immediately, in order to avoid overdraft changes.
Chapter 2: Background Study of online banking and technical environment

Smart Cards, sometimes called stored-value cards, have a specific amount of credit embedded electronically in the card. Smart cards can be used to cover bills or prepaid amounts such as pay phone charges, bridge, parking fees, gas and electricity or Internet purchases. These cards make the transaction fast, easy and convenient.

Smart card technology is in a period of rapid change. Ultimately consumers should be able to customize their smart cards to suit their financial needs with access from their personal computer or cellular phone. Some important consumer issues are:

- Smart cards are the equivalent of cash so must be guarded.
- Procedures for recovering the value of a malfunctioning smart card are unclear.
- The computer chip within the card will contain both financial and personal information. Privacy and security issues could be a problem.
- Smart cards may not be covered by the Electronic Funds Transfer Act in case of loss or misuse of the card.

Digital Cash is designed to allow the consumer to pay cash rather than use a credit card to purchase products on the Internet. One type of digital cash allows consumers to transfer money from a financial institution or a credit card into an "electronic purse". The cash is held in a special bank account that is linked to your computer. Another type of digital cash converts money into digital coins that can be placed on your computer's hard drive.

Digital cheques allow consumers to use their personal computers to pay recurring bills. Consumers can use computer software provided by a bank, or they can use personal finance software packages such as Quicken or Microsoft Money and subscribe to an electronic bill-paying service.

The technology of paying bills electronically by home computers is advancing rapidly, but relatively few businesses currently can accept payments made directly by computers. Digital checking is expensive. Fees generally run from $5 to $10 a month for 20 transactions. Privacy and security issues are major consumer concerns. Encryption technology may lessen privacy concerns in the future [69].

2.3.2 Patterns of Electronic Banking Services:

Not every Bank site on the Internet means an electronic bank. According to global and specific studies of points of supervision and control of the U.S. and Europe; there are three basic images for an e-banking website [34]:

---
Chapter 2: Background Study of online banking and technical environment

I: - Location Information, Informational, a basic level of e-banking or a minimum of electronic banking activity, the bank provides information about its programs and its products and services.

II: - Location Communicative interactive or communication to allow the site some kind of communication exchange between the bank and its clients such as your e-mail and filler requests for samples on-line or modifying the information restrictions and accounts.

III: - Location Cross-transactional and this is the level that the bank exercises its services and activities in an electronic environment, and includes this image to allow the client access to his/her account, to manage and make payments with cash and pay the value of invoices and perform all services and make transfers between accounts within the bank or with third parties.

Internet banking has been able to provide tremendous opportunities for banks; it allows banks to expand and create significant competitive opportunities in their markets through continuity in the activities of attracting deposits and granting credit; it also allows them to provide new banking products and services or to support and strengthen the competitive position in the market.

No doubt the continued development of banking services available through the Internet contributes in one way or another to improving the efficiency of payment systems. Banking systems also gain by reducing the cost of operations for retail customers in the banks both at the regional and international level; consequently, customers and banks have the ability to raise the efficiency of the work.

2.3.3 Electronic Banking Procedure and Sequence

Online banking is a highly profitable channel for financial institutions. It provides customers convenience and flexibility and can be provided at a lower cost than traditional branch banking. Financial institutions have spent a great deal of time and money developing online banking functionality to allow customers an easy and convenient way to manage their money. These changes have created the opportunity for customers to move away from using bank branches that are costly to operate. In fact, some banks are driving customers to the online channel by charging fees for tasks commonly available to a customer in a branch, such as charging for teller visits [8]. Online customers have the ability to log into their accounts to pay bills, transfer money from one account to another, and perform account maintenance, such as address, phone, and e-mail changes. In many cases, accessing online accounts also gives the user the ability to see account numbers, routing numbers, balance, and transaction information. At one time, most of these services required customers to physically enter a physical building; these services can now be performed in an online environment. Financial institutions continue to add more services to the online channel, without increasing the amount of authentication needed to perform the services.
Chapter 2: Background Study of online banking and technical environment

Generally, Internet banking is a series of operations carried out by the bank's client: by logging on to the website of the bank through a Web browser installed on your computer you can perform various banking transactions such as:

Transfer funds between accounts.

Transfer funds to accounts of a third party within or outside the bank.

Enquire about account balances.

View last transactions.

Display an image from the instruments.

Use the applications for loans.

Personal online banking, which lets you watch and make your transactions, keeps you in touch with your account and your bank from anywhere around the clock.

Online banking can take place in four main stages as shown in figure 2.1 below: front end application, user communication stage, user authentication stage and back end bank data base stage.

Figure 2.1 four main stages for online banking
Chapter 2: Background Study of online banking and technical environment

1. The user is running an OS system.
2. After opening the browser they access the Internet banking site and identify themselves and enter PIN and password by using the keyboard.
3. Encrypt the data entered and transmitted or passed to the Bank site.
4. The Bank is to decrypt data sent, and then performs user verification.

There are three essential components of this structure [1]:

1. User: there are two users who deal with the system, they are either a normal user (client) called the bank customer and administrator, an individual system or another program to oversee the application of the bank.
2. Application server: a bank that is responding to requests from users and will oversee the database and pass all requests to the database through this server.
3. Database: usually a stand-alone server responding to requests sent by the server application.

The application shown above is based on Protocol reply/request, where the user creates a request to the server, the server replies by implementing logical operations required or the application is installed; the request to connect to the database may or may not require this.

2.4 Other Types of Methods Using E- Banking

There are other types of methods of personal computers and Internet banking systems:

I. Banking transactions through mobile:
Express banking transactions using mobile or Personal Digital Assistant (PDA) to perform banking operations of checking stock or operations on accounts and transactions. The first accounting services through short message service (SMS) appeared in 1990; the smart phones using WAP technology enabled mobile Web usage and European banks were the first to provide mobile banking services with WAP technology for customers.

Up to 2011 mobile banking transactions were completed using SMS via mobile Web. The success of Apple with the IPhone operating system based on the technique of the Android operating system, and the appearance of Google and other web browsers on the scene have been influential factors in increasing the use of special software the customers call apps, and had an impact on the development of banking applications via mobile phones.

The use of mobile banking has increased by more than a third in the past year (2013), and it
appears likely to continue to increase as more and more clients use smartphones. While still small, the use of mobile phones to make payments at the point of sale has increased even more rapidly. Over a quarter of mobile phone users express some interest in using their phones to make payments at the point of sale, giving mobile payments substantial growth potential as the ability to make these payments becomes more widespread.

Figure 2.2 shows that the reason limiting client adoption of mobile banking and payments is concerns about the security of the technology and a sense that they don’t offer any real benefits to the user over existing methods for banking or making payments. With regards to security, clients have actually become more likely in the past year to report that they simply don’t know how safe it is to use mobile banking, suggesting that clients need to be provided with reliable and accurate information on the level of security associated with the various means of accessing mobile banking.

![Figure 2.2 mobile usage statistics][83]

From Figure 2.2 we can know the main reasons for deciding not to use mobile payment

II. Telephone Banking:

The service provided by the financial institutions allows customers to conduct transactions via land line phone, and most telephone banking services using an IVR system (Interactive voice response) that responds to the client according to the figures of the services requested by using the keyboard on the phone or through technical distinction sounds.

To provide protection and confidentiality, clients must be confirmed first by oral password (voice) or by using the numeric keypad on the telephone or answering a question from the bank.
2.4.1 Advantages of Electronic Banking Services

- Provide quick service and avoid the problem of waiting at banks to conduct a transaction or even to access the bank, whenever there is distance between the user and bank [4].

- Low-cost:
  
  Firstly, from the client’s side and the proportion of money lost in roaming and searching for the service.
  
  Secondly, from the bank, as follows
  
  o Minimize human errors as they are very likely if the bank officer performs the banking process manually for many account numbers with long and many fields to be filled for each transaction.
  
  o The world these days is marked by the emergence of many advanced technologies and the Internet.
  
  o Objects moving towards the Internet have increased; therefore, convergence and overlapping activities were logical linking various financial trading systems.
  
  o The ways the bank deals with markets by buying certain goods from the market, followed by the withdrawal of the corresponding value from the customer and converting to a direct market account. [9]

All these factors resulted in a great many people banking online knowing that manual transactions still exist, but there is a very large acceleration in the search for the service online if possible.

Online banking does not mean changing habits and financial transactions known to the client, but using techniques from computer and Internet networks to give you an option to overcome obstacles that could constitute a burden of time and cost and so therefore be effective and represent added value in a world of financial transactions.

2.4.2 Disadvantages of Online Banking Services

- Must provide Internet service.

- Must acquire basic skills in dealing with computer and Internet.

- Of the biggest problems facing these services is a problem of trust: users unsure of correct procedure; this request can be transformed into non-authorized access, and could delay the response for long periods as a result of being attacked. [33]
Chapter 2: Background Study of online banking and technical environment

- The problem of learning a new technique: some people feel uncomfortable with trying to learn new techniques, particularly techniques for the financial side.

- Abandon paper-based transactions: some people prefer to deal directly with people in cases involving money [38].

- This flexibility encourages many threats of fraudulent websites, theft of personal documents, whether passwords, account numbers, confidential card numbers or other confidential documents those are supposed to be safe by pirates online.

Standardizing the security of Internet banking was identified quickly, causing the creation of a number of guidance documents and best practices, such as those published by the Basel Committee. Within the Basel framework, a number of Internet banking security controls should be implemented, including

1- Authentication of Internet banking customers.
2- Non repudiation and accountability for Internet-banking transactions.
3- Appropriate measures to ensure segregation of duties.
4- Proper authorization controls within Internet banking systems.
5- Databases and applications.
6- Data integrity of Internet banking transactions.
7- Records and information.
8- Establishment of clear audit trails for Internet banking transactions.
9- Confidentiality of key bank information.

2.5 Operating System Environment

2.5.1 O.S Introduction
The extension of the existing operating system aims to build a secure operating system for desktop and laptop computers. The stress is on security, which is achieved by exploiting the isolation capabilities of the bar-metal machine, with integrated capability to modern hardware technologies, such as Intel VT-d and Trusted Execution Technology [106].

2.5.2 Problem with Current Operation System
Current mainstream operating systems that are used for desktop computing, e.g. Windows, Mac OS x based systems prove unsatisfactory when it comes to security.
# Linux VS Windows [29]

<table>
<thead>
<tr>
<th>Price</th>
<th>Although the majority of Linux variants have improved dramatically in ease of use, Windows is still much easier to use for most computer users because of the familiarity of Windows and because it's more likely they are using a Windows computer at home, in school, or at the office.</th>
<th>Microsoft has made several advancements and changes that have made it much easier to use its operating system, and although arguably it may not be the easiest operating system, it is still easier than Linux.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
<td>The majority of Linux variants and versions are notoriously reliable and can often run for months and years without needing to be rebooted.</td>
<td>Windows, still cannot match the reliability of Linux.</td>
</tr>
<tr>
<td>Software</td>
<td>Linux has a large variety of available software programs, utilities, and games. However, Windows has a much larger selection of available software.</td>
<td>Because of the large amount of Microsoft Windows users, there is a much larger selection of available software programs, utilities, and games for Windows.</td>
</tr>
<tr>
<td>Software Cost</td>
<td>Many of the available software programs, utilities, and games available on Linux are freeware or open source. Even such complex programs such as Gimp, Open Office, Star Office, and wine are available for free or at a low cost.</td>
<td>Although Windows does have software programs, utilities, and games for free, the majority of the programs will cost anywhere between $20.00 - $200.00+ US dollars per</td>
</tr>
<tr>
<td><strong>Hardware</strong></td>
<td>Although hardware manufacturers have made great advancements in supporting Linux it still will not support most hardware devices. However, for the hardware devices that have driver support they usually work in all versions of Linux.</td>
<td>Because of the number of Microsoft Windows users and the broader driver support, Windows has much larger support for hardware devices and almost all hardware manufacturers will support their products in Microsoft Windows.</td>
</tr>
<tr>
<td><strong>Security</strong></td>
<td>Linux is and has always been a very secure operating system. Although it still can be attacked when compared to Windows, it is much more secure.</td>
<td>Although Microsoft has made great improvements over the years with security on their operating system, their operating system continues to be the most vulnerable to viruses and other attacks.</td>
</tr>
<tr>
<td><strong>Open Source</strong></td>
<td>Many of the Linux variants and many Linux programs are open source and enable users to customize or modify the code however they want.</td>
<td>Microsoft Windows is not open source and the majority of Windows programs are not open source.</td>
</tr>
</tbody>
</table>
Chapter 2: Background Study of online banking and technical environment

| Support | Although it may be more difficult to find users familiar with all Linux variants, there are vast amounts of available online documentation and help, available books, and support available for Linux. | Microsoft Windows includes its own help section, has vast amounts of available online documentation and help, as well as books on each of the versions of Windows. |

Table 2.1: Comparison between Linux and Windows

When we compare Linux against Mac OS we found that Mac OS is based on a Berkeley Software Distribution (BSD) code, while Linux is an independent development of a Unix-like system, this means that these systems are similar, but not binary compatible. Furthermore, Mac OS has a lot of applications not open source and built on unknown libraries, the other thing Mac OS only available on Apple computer.

The major problem with current systems is their inability to provide effective isolation between various programs running on one machine. If the user’s web browser gets compromised (due to a bug exploited by a malicious website), the OS is usually unable to protect other users’ applications and data from also being compromised. Similarly, if certain system core components get compromised, e.g. WI-FI driver or stack, none of the above mentioned systems can defend themselves from a complete compromise of all the applications and data [101].

The above situation is a direct result of certain architectural design decisions, which include over complexity of OS API, insecure GUI design, and the monolithic kernel architecture. Systems that are based on such insecure architecture simply cannot be given strong security.

The reactive approach, as many vendors use today, tries to patch every single known security bug. But such an approach does not scale well especially for users who require more than average security. Patching can only address known and popular attacks, but offers no protection against new, or less popular, more targeted threats. Consequently, we need a different approach to build a secure system.

Obviously building a new system can be a very time consuming task that is why reusing as many read to use building blocks as possible is important. This in turn implies the use of virtualization technology, which is used for two primary reasons: offering excellent security isolation.
properties, as well as the ability to reuse a large amount of software, including most of the applications and drivers written for mainstream OSes, like Linux or Windows.

2.5.3 Common Desktop and Security

2.5.3.1 GUI

One of the main goals of puppy OS in our scheme is to provide seamless integration of the banking application hosted in USB onto a common user desktop, so that it is easy for the non-technical user to work with the application. Another important goal is to provide a near-native performance, especially for displaying banking applications that use rich graphics and multimedia, like web browsers.

2.5.3.2 The Network Domain

In a typical operating system like Windows or Linux, the core networking code, that includes network card drivers and various protocol stacks (802.11, tcp/ip, etc), runs in the kernel. This means that a potential bug in the network code, when exploited by an attacker, always results in full system compromise.

In our architecture we try to limit potential attack vectors in the system as much as possible. That is why the whole networking code, including drivers and protocol stacks, has been built on an unprivileged zone called the network domain.

The network domain has direct access to the networking hardware, i.e. the Wi-Fi and Ethernet cards. The access is granted by new Intel VT-d technology “if used”, which allows for safe device assignment to unprivileged code. When a potential bug in the networking code, e.g. in the Wi-Fi driver, is exploited by an attacker, all that attacker can gain is the control over the network domain, but not the application. This does not give the attacker any advantage compared to what he or she can already do when controlling e.g. the hotel LAN or Wi-Fi network (which assumes all the usual networking attacks like sniffing of unencrypted traffic or performing man-in-the-middle attacks). Particularly, our applications use a form of cryptography-protected network protocols. The table below represents the network configuration setup for our puppy O. S.

<table>
<thead>
<tr>
<th></th>
<th>activate network by right clicking on network icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>selects setup networking and chooses which option to connect to the internet</td>
</tr>
<tr>
<td>3</td>
<td>selects wired or wireless LAN to connect internet via wireless LAN</td>
</tr>
<tr>
<td>4</td>
<td>when clicking on above option a new screen appears then select simple network setup</td>
</tr>
<tr>
<td>5</td>
<td>in interface tab select wlan0 which appears as an available network then enter key</td>
</tr>
<tr>
<td>6</td>
<td>Download jre1.8.0_11 in order to run file.jar</td>
</tr>
</tbody>
</table>
Chapter 2: Background Study of online banking and technical environment

| 7 | Encoding=UTF-8       |
|  | Name [en_US]= name of a file  
  | Generic Name=GUI Port Scanner  
  | Exec=java -jar /path of file/name of file.jar  
  | Terminal=false  
  | Icon [en_US] =/path of picture/name of picture .png  
  | Type=Application  
  | Categories=Application; Network; Security;  
  | Comment [en_US]= name of file Graph Editor  |

| 8 | Drag and drop file to desktop |

| 9 | change the name of file by right clicking on file and selecting edit item then enter the name of a file to appear in the desktop |

| 10 | Optionally can change the icon of a file by right clicking and selecting file'name of a file' and choose set icon and simply drag and drop the specified icon |

| 11 | The final step is click on file and it will open in browser |

Table 2.2: Network configuration setup for our modify puppy Linux O.S.

2.6 Cloud Computing Definition

Cloud computing [58] is a means by which highly scalable and technology enabled services can be easily consumed over the Internet on as-needed-basis. This innovative paradigm has generated a significant interest in both the market place and the academic world, resulting in a number of notable commercial and individual cloud computing services, e.g., from Amazon, Google, Microsoft, Yahoo, and Salesforce.

In general, cloud computing architecture is divided into two layers: the bottom resource layers and the upper service layer. The bottom is the foundation, based on virtualized resources in the form of storage and computing, and the upper service layer provides specific services. Cloud Computing is clearly one of the enticing technologies, at least in part because of cost-efficiency and flexibility.
2.6.1. Cost Minimizing

Assume our service has a predictable daily demand where the peak requires 100 servers at noon but the trough requires only 20 servers at midnight; as long as the average utilization over a whole day is 50 servers, the actual utilization over the whole day (shaded area under the curve) is 50 x 24 = 1200 server-hours; but since we must provision to the peak of 100 servers, we pay for 100 x 24 = 2400 server-hours, a factor of 1.5 more than what is needed. Therefore, as long as the pay-as-you-go cost per server-hour over 3 years is less than 15 times the cost of buying the server, we can save money using utility computing.

Figure 2.3: Provisioning for peak load and underprovisioning

From Figure 2.3 can show that in: (a) even if peak load can be correctly anticipated, without elasticity we waste resources (shaded area) during nonpeak times. (b) Under provisioning case 1: potential revenue from users not served (shaded area) is sacrificed. (c) Under provisioning case 2: some users desert the site permanently after experiencing poor service; this attrition and possible negative press results in a permanent loss of a portion of the revenue stream.

However, with all these cloud computing capabilities and potential to save cost, large enterprises and others should step back, move cautiously and analyze the security risks and concerns associated with cloud computing before adapting the technology.

Table 2.3 shows the categories of the cloud services and which provider can support that kind of service.

<table>
<thead>
<tr>
<th>Provider</th>
<th>IaaS</th>
<th>PaaS</th>
<th>Compute Billing</th>
<th>Storage Billing</th>
<th>Relational Database</th>
<th>Hybrid capabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows Azure</td>
<td>No</td>
<td>Yes (Note, Java, Pay-per-use)</td>
<td>Pay-per-use</td>
<td>Yes (SQL Server)</td>
<td>Yes (on-premise to cloud)</td>
<td></td>
</tr>
<tr>
<td>Amazon Web</td>
<td>Yes</td>
<td>No</td>
<td>Pay-per-use</td>
<td>Pay-per-use</td>
<td>Yes (MySQL-)</td>
<td>Yes (via third-party)</td>
</tr>
</tbody>
</table>
Table 2.3: Competitive matrix.

<table>
<thead>
<tr>
<th>Rack space</th>
<th>Pay-per-use (IaaS); Monthly</th>
<th>Pay-per-use (IaaS); Included in</th>
<th>Yes (Fathom DB)</th>
<th>No (but dedicated resources to cloud)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joyent</td>
<td>Yes</td>
<td>Monthly</td>
<td>No</td>
<td>Yes (on-premise to cloud)</td>
</tr>
<tr>
<td>Google</td>
<td>No</td>
<td>Pay-per-use</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GoGrid</td>
<td>Yes</td>
<td>Pay-per-use or pre-paid</td>
<td>No</td>
<td>Yes (dedicated resources to cloud)</td>
</tr>
</tbody>
</table>

Nevertheless, several security issues in the cloud are impeding the vision of cloud computing as a new IT procurement model. This security observation can be put into three categories [5].

- **Traditional security** - Involves concerns related to computer and network intrusions. Some of these attacks include VM-level attacks [25], cloud provider vulnerabilities [57], phishing cloud providers, authentication and authorization, and forensics in the cloud. Cloud providers respond to these concerns by arguing that their security measures and processes are more mature and tested than those of the average company.

- **Availability** - Involves concerns on critical applications and data availability. Server uptime issues, single points of failure and attack, and the inability of an enterprise to ensure that the cloud provider is faithfully running a hosted application and giving the valid results make companies nervous.

- **Third-party data control** - For the data owner who out-sources data to the cloud, the cloud acts as a semi-trusted third-party. Data placed in the cloud can reside anywhere. Outsourcing data to the Cloud should not lead to de facto outsourcing of data control [71].

In general, encryption is a useful tool for protecting the confidentiality of sensitive information transactions so that even if a transaction is compromised by an intruder, the information remains protected even after the transaction has been successfully attacked or stolen. Provided that the encryption is done properly and the decryption keys are not accessible to the attacker, encryption
can provide protection for the sensitive data stored in the database, reduce the legal ability of the data owners, and reduce the cost to society of fraud and identity theft.

2.6.2 Background Works on Cloud Computing.
Cloud computing is an emerging fielding with many research and implementation challenges. While it is tempting to view cloud computing in terms of existing computing models, the cloud presents many unique characteristics outlined in [72]. Data security in the cloud is one of them. As mentioned before, the cloud presents unique challenges; techniques designed for computing environments may move to the cloud or be used in a hybrid environment. Our fundamental theorem is based upon additive homomorphic encryption (such as Paillier’s cryptosystem [109] and cashing re-encryption [64, 74] to secure data in the cloud and in transit from the cloud to a client.

2.7 Summary
This chapter discussed the banking system and tried to give a quick glance of the banking sector background. Also this chapter demonstrated the electronic method used to facilitate the banking services. After that we presented the advantages and disadvantages of electronic and online banking services. Finally, we gave an overview of operating system technology and made a comparison between different operating systems.

The guidance for banks issued by the Federal Financial Institutions Examination Council (FFIEC) describes enhanced authentication methods underlining the ineffectiveness of single-factor authentication [13]. More standards, such as the International Organization for Standardization’s ISO21188:2006, describe a framework of requirements to enable certificate-based solutions for secure Internet banking applications [19]. ISO21188:2006 defines security targets, as well as procedures that facilitate the risk management process. However the electronic/online banking problems still grow and become more sophisticated.
Chapter 3: Security and Authentication Access Control

Chapter 3 Overview Security, Authentication and Access Control

Security is simply the state of feeling safe and protected from threat and danger. The increasing preference for online communication and transactions underscores the need for security of information in addition to security of communication infrastructures and devices. This is because of the vital nature of information to communication as well as being a critical resource in carrying out organizations’ activities. The security concept varies from one networking environment to another and also depends on the operating assumptions of the network in question. Understanding the basic concepts security was designed to address is very important. Wireless networks provide the simplest means of connectivity and are the most vulnerable to network threats hence the need for access control. Security issues in wireless networks have been discussed in [31-60] and a detailed discussion is out of the scope of this work.

The focus of this chapter is to review security from the general network and online banking perspectives as well as access control and its implementation in disruptive electronic banking concepts discussed in the previous chapter. The chapter is structured as follows: section 3.1 reviews the network security with focus on basic security terminologies, network security threats, security services, cryptography and cryptographic algorithms. Section 3.2 discusses security issues in online banking with focus on goals, threats, requirements, design consideration, transaction security, factors affecting electronic banking security and key research challenges. Section 3.3 presents an overview of cryptography and access control with focus on requirements, access control elements, existing research works and access control implementation in the online banking environment. Section 3.4 is intruder methods used to hack internet users and section 3.5 covers the general defence used in the financial sector. Section 3.6 gives some information about homomorphic cryptography and section 3.7 is a summary

3.1 General Network Security

Network security discussed in [45-74] is the process of preventing and detecting unauthorized use of resources as well as reacting against an established threat or danger. How well the network is secured against threats will depend on the ability of a node to: verify the authenticity of every received packet (preventive); identify when malicious activities have taken place and the entity involved (detective); and reject every future packet from the malicious entity and notify the administrator for expulsion (reactive). This section discusses basic security terminologies, network security threats, security services, cryptography and cryptographic algorithms.
3.1.1 Basic Security Terminology

This section presents and describes some basic security terminologies used during the course of this research work.

**Plaintext**: The original readable message in plain form before being encrypted into ciphertext or after being decrypted.

**Ciphertext**: The output result obtained when an encryption is performed on a plaintext using a cipher to make it unreadable before decryption.

**Cipher**: This is an algorithm for performing an encryption or decryption and its operation is dependent on a key.

**Key**: This is a piece of information or parameter used to determine the output of a cryptographic algorithm. It is often of various sizes depending on the algorithm used.

**Encryption**: The process of distorting data (plaintext) with a key and an algorithm (cipher) to make it unreadable to an unauthorized party.

**Decryption**: The process of applying a relevant key to an encrypted data and an algorithm to make the resultant output readable to an authorized party.

**Signature**: A signature is a bit string attached to the document when signed (e.g one-way hash of the document encrypted with the private key). Digital signature algorithms are public-key based algorithms with secret information to sign documents and public information to verify signatures.

**Nonce**: This is a number or bit string used only once in a secured communication.

**Hash-based Message Authentication Code (HMAC)**: This is a specific construction for calculating a message authentication code (MAC) involving a cryptographic hash function (MD5 or SHA) in combination with a secret key.

**(Secure Sockets Layer) SSL**: Standard security technologies for establishing an encrypted link between a server and a client—typically a web server (website) and a browser; or a mail server and a mail client.

**Transport Layer Security (TLS)**: TLS [45] is a transport-layer security protocol that uses a key-agreement protocol such as Diffie-Hellman [45] to establish a Confidentiality channel with integrity guarantees between two parties. Authentication of the server is provided by an X509 certificate chain rooted in one of several trusted third parties whose certificates are provided to client’s out-of-band.
3.1.2 Cryptographic Hash Function:

This is a deterministic procedure that takes an arbitrary block of data and returns a fixed-size bit string (hash value) that changes with a change to the data.

**Timestamp:** This is a sequence of characters denoting the date and/or time of occurrence of an event.

**Certificate:** A certificate is a tool used to bind a public key to a principal or a public key to an attribute. Certificates (with X.509 as example) are always generated by a trusted authority to ensure the correctness of conveyed information.

**Attribute:** This is a specification that defines an object’s, element’s or file’s property. It comprises a name and a value in the case of an object; a type or class name in the case of an element; a name and extension in the case of a file.

3.2 Security Goals

Security goals are measures aimed at preventing and detecting the activities of malicious entities through the provision of information security. The general goals in network security are:

- Only intended audience should know the content of transmitted or stored data
- Every data must have a source to facilitate detection of any possible modification
- Every communication event must be traced to an entity
- Network services should be available and function correctly
- Certain services or resources should be restricted to only authorized entities

3.2.1 Security Threats

A threat as defined in [41] is “any possible event or sequence of actions that might lead to a violation of one or more security goals.” The actual realization is known as an attack. System threats are always targeted at the protocols and can be directed against: the cryptographic algorithms used in protocols; the cryptographic techniques used to implement the algorithms and protocols as well as against the protocols themselves. These threats are either passive (where a non-participant in a protocol eavesdrops on some or all of the protocol without affecting the protocol) or active (an attack where the attacker under the disguise of being someone else tries to alter the protocol to his/her own advantage). This is much more serious in protocols where there is lack of trust among the communicating parties. The form of active attacks depends on the network and requires active intervention. The classified attacks are eavesdropping, masquerading, modification, replay, repudiation, authorization violation and Denial of Service.
3.2.2 Security Services

Security service according to [51] is “an abstract service that seeks to ensure a specific security property.” It can be realized through the use of cryptographic algorithms and protocols or conventional method. A combination of both cryptographic approach and conventional method is considered more effective. The five known security services of confidentiality, data integrity, authentication, non-repudiation and access control are discussed below.

**Confidentiality** is the process where the content or meaning of transmitted information is concealed in the form of a ciphertext to restrict access to only authorized entities during transmission over an insecure channel. Both symmetric and asymmetric key algorithms are suitable in carrying out this process. This service can mitigate masquerading, eavesdropping and authorization violation attacks.

**Data Integrity** is the process which ascertains the originality of a transmitted message and assures the communicating entities that the transmitted message was not tampered with. This is realized by the use of symmetric-based HMAC or an asymmetric-based digital signature to provide protection against malicious alteration. This service can mitigate masquerading, authorization violation, modification and man-in-the-middle attacks.

**Non-repudiation (Accountability)** is the process which prevents a communicating party from refuting the validity of an already taken action like sending of a message. This service is only possible with asymmetric cryptosystems that create digital signatures since a symmetric cryptosystem does not have a mechanism to prove the creator of the original message. A third party involvement has been identified as a way of enforcing this service. This service can mitigate masquerading, authorization violation, repudiation and modification attacks.

**Authentication** is the process of verifying either the true identity of another communicating party or the source of data origin when the verifying party does not have personal information of the other’s identity. This service can be realized by both symmetric and asymmetric based techniques. This service can mitigate masquerading, modification and man-in-the-middle attacks.

**Access Control** is the process of protecting the network from unauthenticated entities and usage of system/network resources from unauthorized entities. An access control system that incorporates the AAA (Authentication, Authorization and Accounting) operations is usually considered very effective. It can be realized by both symmetric and asymmetric based techniques. The service mitigates masquerading, authorization violation, modification and denial of service attacks.
3.3 Cryptography

Cryptography discussed in [52] is the study and practice of providing information security. It employs mathematical techniques with cryptographic keys to provide protection against network attacks. Kirchhoff’s principle of having a publicly-known algorithm and a secret key is the base on which modern cryptographic systems are built. The presence of redundancy (to make transmitted messages unintelligible) and message freshness (to prevent replay of old messages) are the two fundamental cryptographic principles underlying all the cryptographic systems. A cryptographic algorithm used in cryptographic protocols is a mathematical transformation of input data to output data; while cryptographic protocol is a series of steps and message exchanges between multiple entities with the sole aim of achieving a specific security objective.

**Cryptographic Algorithm**

Cryptographic algorithm is a sequence of finite instructions employed for calculation and data processing to achieve cryptographic (security) goals. The two known types are symmetric-key and asymmetric-key cryptography.

**Symmetric-key Cryptography** is a cryptographic method that uses the same key for message encryption and decryption. Examples of symmetric-key algorithms include Data Encryption Standard (DES), triple-DES (3DES) and Advance Encryption Standard (AES) which is the most secured and widely use. These algorithms can be used in electronic code book mode, cipher block chaining mode, cipher feedback mode, stream cipher mode and counter mode. AES supports key lengths and block sizes from 128 bits to 256 bits in steps of 32 bits, and specifies that the block size must be 128 bits and the key length must be 128, 192 or 256 bits. Its characteristics are:

- It is not easily scalable
- Online exchange of keys over insecure media is not safe
- Has no provision for non-repudiation
- Authentication is provided only for communicating entities with the secret key
- Provides integrity and prevents outsiders from carrying out modification

**Asymmetric-key Cryptography** also known as public-key cryptography does not require an initial exchange of secret keys for secure communication between two communicating parties. Different keys are used for both encryption and decryption and derivation of decryption key from encryption is impossible thus making publishing of the public key possible. The two known types are RSA (Rivest, Shamir and Adleman) cryptography and Elliptic Curve Cryptography (ECC). Though ECC with some interesting features is an emerging alternative to RSA, RSA is still the most widely used. These two cryptosystems have been discussed in [68-69] with their advantages and disadvantages highlighted. RSA is the preferred choice for implementation because it has a
very low verification time compared to ECC. It involves three steps of key generation, encryption and decryption. It forms the base of most practical security because of its strong nature and is based on some principles from number theory. The characteristics of asymmetric cryptography are:

- Scalable with increasing number of users.
- Online exchange of public keys over insecure media is safe.
- Has provision for non-repudiation.
- Authentication is provided for any communicating entity.
- Provides integrity and prevents outsiders/insiders except the sender from carrying out modification.

3.4 Intruder Methods used to Hack Internet User

The normal operating environment in the user's computer is vulnerable to a number of risks and threats because of the bad use of the system or installing the system incorrectly as well as using unreliable software; and if a user tries to conduct transactions via such an environment, there is no guarantee of the safety of the transactions [28, 29, 32, 53, 102].

Generally, we find that any information sent between any two entities may be vulnerable to intruders in four ways: Interception, that there is someone in the middle between sender and receiver who will copy the contents of the message and send. Interruption, that there are people in the middle between sender and receiver that can grasp the message and not send it to the other party. Amendment (Modification), to have someone in the middle between sender and receiver to modify the contents of the message then send it. Fabrication, an identity theft of one of the parties whether the sender or receiver.

On the other side online banking has grown and flourished over the years, but is now facing challenges due to the risks of phishing, data compromises, and other attacks. The rise of these attacks could cause a drop in the use of online banking and negatively affect client confidence in the ability of a financial institution to protect them. Clients are questioning the safety of their money and information and are expecting banks to fix the problem. Generally speaking there are four main types of attacks on e-banking. All of the attacks seen at the moment fall into the first two categories:

1- Getting authentication credentials from the client victim
2- Modifying the client victim’s legitimate transactions.
3- Denying them access to their banking
4- Merely observing the transactions of the client victim
A very well-known online fraud is phishing, which is an attack designed to convince the victim to give away their online banking credentials to a third party. This and similar scams or attacks which reveal credentials to the attacker fall into the class of credential harvesting.

The researcher defines phishing as: a technique used to deceptively obtain information, including personal data, banking and credit information, passwords and other account numbers from Internet users by using emails and websites designed to look like a legitimate business, financial institution, or government agency. The purpose of obtaining the information is usually to commit identity theft or identity fraud. Phishing is an evolving scam; with every attack comes a new method to help the phisher lure more victims. It is difficult to detect and prevent attacks because they seldom have the same characteristics. There are several different types of phishing attacks including misleading e-mails, man-in-the-middle, URL obfuscation, page content overriding, malware phishing, key loggers and screen grabbers, session hijackers, web Trojans, IP address manipulation, and system reconfiguration attacks.

There is a Web browser threat (Man-in-the-Browser); this type of attack redirects the user to fake sites. Most banks provide a password (often a One-Time Password (OTP)) to protect the user from getting their password stolen and reused again after the authorizing client logs off. This security method can be hacked to disrupt the functionality of changing the password every time, using a proven tool to modify the file (in windows SAM file), which converts domain names to addresses (host file) or intercepts a user during receipt of banking services by bank page coverage of HTML Injection. Another threat that can affect the SSL or Secure Socket Layer connection (Man-in-the-Middle) is an encryption protocol that provides a guarantee to connect safely online and almost all bank sites use these protocols to protect communications between them and their customers, but SSL protocol cannot ensure a 100% secure connection.

The attacker can succeed in penetrating the client side SSL session in the following sequence:

1- The attacker manipulates and converts address names in the domain name server (through DNS spoofing) and changes protocol conversion to logical addresses from physical addresses (ARP spoofing).

2- Then they prepare a fake page with a false certificate. The browser then opens the meeting and asks if the certificate is valid and not false as shown in figure 3.1
3- Validating the warning message shown above will appear to the user, but few users are experienced with the types of attacks and will press the button "yes" without thinking.

4- Despite the number of binaries that encrypts the communication (128 bits) the attacker can penetrate this connection.

5- Then the attacker uses fraudulent certificates and uses a tool called (SSL Dump) to decrypt the packets and gain user information.

3.4.1 Malware or Malicious (and Other Malicious Code-Based Redirection Schemes):

adopt the method that the user is not defined by malware (Malicious Code) on his computer desktop or to redirect the user without his knowledge to a location similar to the location where they want to access and collect information that the user enters, and this process is called forwarding (Redirection).

And from other types of malware:

**Recording software keyboard buttons (key logger):** where to record what the user enters in terms of characters and numbers to enter the financial account, and send it to the hunter (Phisher) to withdraw funds from that account.

**Backdoor:** this program gives the hunter access to device user data and financial accounts, to transfer funds, so it appears that the same user has done this process and therefore in many cases when the user denies the money transfers from the account it is difficult to substantiate the claim.

3.4.2 Scams.Phish Phishing Websites/Emails

Phishing websites are the most commonly seen form of credential harvesting attack on the web, as seen from figure 3.2, usually combined with an email which tricks the user into accessing the website. They are purely a social engineering attack which relies on user misunderstanding of security features and problems with how security indicators are presented to users.
Extended character sets: In order to have a successful phishing website the URL must look plausible. The first technique for doing this is using web browser support for extended character sets in domain names. Homograph attacks were first described in 2002 [82]. The current standard for encoding Unicode characters in a legal URL is the puny code standard [105] which uses two hyphens as an escape. Such URLs are displayed as the Unicode characters in the address bar and in links. Because Unicode contains several distinct characters which appear very similar (for example ‘g’ (Unicode 0x0581) and ‘h’ (Unicode 0x10b9), from the Armenian and Georgian alphabets respectively) it is possible to create a URL which is distinct, but appears the same to the human eye. This was used to attack PayPal [51].

Typo-squatting: Typo-squatting is a technique where attackers register similar domains to the target website, either through simple spelling mistakes or common typing errors. These are used to host pornographic websites, ads, malware and sometimes phishing attacks. McAfee recently released a report on typo-squatting [12].

Sub-domains: A common phishing technique is to rely on the fact that the user cannot tell the difference between similar URLs such as www.XXX.com and www.XXX.secure-banking.com. The average member of the public will see the word XXX and assume they are owned by the same company, whereas in fact there is no guarantee that this is the case at all. This technique is helped by the fact that many companies are using URLs outside their normal domain for legitimate sites [18]. This makes it almost impossible for a user to tell them apart. Because phishing is such a wide-spread threat, many people have been deploying or proposing solutions. I will try to reflect on the available measures used to combat phishing attacks and evaluate their effectiveness.
3.4.3 Electronic Banking Attack Trees

Schneier was the first to associate the term “Attack Tree” [32] with the use of tree style models for expressing different ways in which a system can be attacked. He proposes an attack tree as a formal and methodical way for describing the security of the system based on varying attacks.

Attack trees provide a way for modeling goals of an attack and alternative ways to achieve that goal. Attack trees provide formal methodology for analyzing the security of systems and subsystems. They provide a way to think about security, capture and reuse expertise about security, and respond to changes in security. An attack tree has a root node and leaf nodes. The root node represents the target of the attack, while the leaf nodes represent the means for reaching the target, which are the events that comprise the attack. The attack tree is depicted in figure3.3 and the area which this thesis focuses on has a bold Red colour.
The attack tree is used to gain a general view on the different types of attacks, the analysis of which should facilitate the process of studying the adequacy of existing countermeasure used by the bank.
The attack tree has one root node, representing the final target to the attacker, which is the compromise of the user’s bank account. An intruder may use one of the leaf nodes as a means for reaching the target. To categorize Internet banking attacks, each component to the process should be examined: the user terminal/user (UT/U), the communication channel (CC) and the Internet banking server (IBS). The following types of attacks are identified:

**UT/U attacks**—these attacks target the user equipment, including the tokens that may be involved, such as smartcards or other password generators, as well as the actions of the user himself/herself. UT/U attacks include:

– **UT/U1**—Procedural attacks:
  - UT/U1a: User surveillance (piggybacking)—similar to the personal identification number (PIN) thefts facilitated by the installation of cameras in automatic teller machines (ATMs); the user’s actions may be monitored to capture credentials.
  - UT/U1b: Theft of tokens and handwritten notes giving Internet banking user names are usually long and have to be written down. Users may also keep their passwords written, despite the security guidance provided by their banks. Notes may be stolen, providing access to the user’s credentials. Tokens may also be stolen, providing the attacker with one authentication factor that, when combined with other types of attacks (such as PIN calculators), can lead to identity theft.

– **UT/U2**—Malicious software installation. The embedding of malicious content for compromising the user’s login information and password (e.g., keyboard logger or screen capture in image or video files) may take place via a number of different methods, including:
  - UT/U2a: Hiddencode—this is the use of hidden code within a webpage that exploits a known vulnerability of the customer’s web browser and installs malicious software in the user terminal. The exploit may target permissions on Java runtime support, Active X support, multimedia extensions, and automated download and running of software through the browser.
  - UT/U2b: Worms and bots—Worms search for vulnerabilities and exploit them automatically. This includes the exploitation of instant messaging and chatting communication software (that allows the embedding of dynamic content), which may automatically be deployed using bots.
  - UT/U2c: E-mails with malicious code—this is the submission of e-mails with malicious content, such as executable files or HTML code with embedded applets.

– **UT/U3**—Token attack tools:
• UT/U3a: Smart card analyzers—Attacks against smart cards, such as power consumption analysis or time analysis, may expose the security of the smart card by revealing cryptographic keys and passwords. Such attacks are sophisticated and not easy to implement, but are very effective, especially if the necessary counter measures (noise generators, time-neutral code design) against the types of attacks are not implemented by the smart card manufacturer.

• UT/U3b: Smart card reader manipulator—this is applicable to non-certified smart card readers within secure interfaces, which may expose the contents of the smart card by conducting unauthorized operations.

• UT/U3c: Brute-force attacks with PIN calculators—these attacks focus on breaking the security of tokens that generate random PINs. The attack exploits the fact that a time window is necessary, for synchronization reasons. In some implementations, except for the present PIN, the subsequent and preceding codes are active for the same purpose. It is reported that it is possible to break such mechanisms with a minimum window of three PINs.

–UT/U4—Phishing[7] These attacks use social engineering techniques—masquerading as a trustworthy person or business in electronic communication—in an attempt to fraudulently acquire sensitive information, such as passwords and credit card details. The term was initially used in the mid-1990s by hackers who were stealing America Online (AOL) accounts by scamming passwords from unsuspecting AOL users. The attacks include:

• UT/U4a: Social engineering—the attacks focus on the compromise of the user’s credentials by non-technical means, such as phone calls or the submission of e-mails masquerading as an official bank, asking the user for user name and password.

• UT/U4b: Web page obfuscation—these attacks are based on links that do not correspond to the destination they describe, or the use of Internet Protocol (IP) addresses instead of universal resource locators (URL) for confusing the user. Other techniques deploy hidden frames. These are used for covering the real activity of a web page by using several frames with malicious content, while the user sees only the URL of the master frame set. Other methods use graphics that spoof the interface of a web browser, such as the address bar.

• CC attacks—this type of attack focuses on communication links. Examples include:

• CC1: Pharming—These involve compromising domain name servers (DNSs), altering DNS tables and connecting the user to fraudulent sites, instead of the official
bank’s site, where information regarding the user’s account may be derived.

- CC2: Sniffing—Active sniffing attacks masquerade the two communicating entities to each other (user client and the Internet banking server) to capture information, such as user name and password. Passive sniffing captures information from the communication medium, without interception.

- CC3: Active man-in-the-middle attacks—this type of attack regards a schema where the attacker receives and forwards information between the UT and the IBS. The attacker sends malformed user packets or injects new traffic, such as transfer commands, from one account to another.

- CC4: Session hijacking—Attacks that force the user to connect to the IBS with a preset session ID. Once the user authenticates to the server, the attacker may utilize the known session ID to send packets to the IBS, spoofing the user’s identity.

- IBS attacks—these types of attacks are offline attacks against the servers that host the Internet banking application. Examples include:
  - IBS1: Brute-force attacks—Brute-force attacks in certain password-based mechanisms are reported to be feasible by sending random user names and passwords. The attacked mechanisms implement a scheme based on guessable user names and four-digit passwords. The attack mechanism is based on distributed zombie personal computers, hosting automated programs for user name-or password-based calculation. This attack may be combined with user name filtering methods for determining the identity of the user. These methods filter the different responses of the server, in the case of valid or invalid user names.
  - IBS2: Bank security policy violation—violating the bank’s security policy in combination with weak access control and logging mechanisms, an employee may cause an internal security incident and expose a customer’s account.
  - IBS3: Website manipulation —exploiting the vulnerabilities of the Internet banking web server may permit the alteration of its contents, such as the links to the Internet banking login page. This may redirect the user to a fraudulent website where his/her credentials may be captured.

From attack tree topology can be found a major root node comprising user terminal and user. To implement security, authentication mechanisms that complement each other’s resistance to attacks identify the appropriate patterns. For example, hard tokens may be combined with biometrics and supported by a number of countermeasures toward a mechanism that is resistant to all of the attacks described in the attack tree.
Security assessment conducted for the Internet banking authentication mechanism was presented by way of applicable attack and not applicable attack manner for each authentication factor.

3.4.4 Pharming

Domain name system server: any site on the internet uses a domain name, usually as a title (e.g. www.examplebank.com), yet the real title is to be determined through the Internet (IP) protocol; the address usually consists of four digits in the fourth edition of (IPv4). When the user types a domain name in the title field with their browser and presses a button and enters, the domain name is translated to an address IP domain name system server (DNS) and then the browser will connect to the server via this address and then load the webpage. After a user visits a site often, it stores this address for this site to a user's computer in the cache for DNS; this way it does not require the computer to connect to the domain name system server whenever the user wants to visit the same site [42].

How to forge the addresses through DNS:

This process does not use any new ways: it can be done using known methods such as poisoning the cache of the domain name system, or simulation of domain names or domain hijacking, but these methods are used for different purposes. While in the past these methods were used to disable the old services and cause problems for users, now the problems relate to money. These routes still exist because the site managers and owners do not give protection for monitoring the domain name system server as it would cost millions in firewalls.

The illustration in figure 3, 4 shows how counterfeiting operates:

Figure 3.4: counterfeiting operates
1. Intentionally broken to DNS in cache on the user's computer or on a network server or local Internet service provider's server. The intruder uses many methods to change the Internet Protocol address of a known location’s — perhaps a bank — site to another site address that is usually similar in name.

2. The user types the correct address for the location that was forged by their browser's address field and then presses the Enter button.

3. The user's computer searches in temporary memory the Internet Protocol address for this site or domain name system server or asks for this address if it does not exist in the cache.

4. If the forged site exists in the cache, the Internet Protocol addresses the browser as false, and if a browser question server was poisoned, it gets a fake address.

5. Browser connects to this title without knowing this is a false site.

After the intruder redirects the user to a fake site several methods can then be used to simulate the original site in order to obtain confidential information from the user such as their password or bank account number. In this method counterfeiting can deceive the broken server of a multitude of users without their knowledge.

**How to carry out a forgery:** the user can perform an operation using multiple methods of forgery:

- **Poison the cache of the local domain name system:** several methods can be used including sending an email virus or exploiting any vulnerability in a system to change the title to a specific location in the cache.

- **Poison the cache of DNS server:** domain name system servers save all previous requests of users in the cache for a specified period in order to speed up the response to users of sites with most applications. This cache can reportedly be poisoned by exploiting a vulnerability in the server operating system or application that is used to provide the same service.

- **DNS Hijack:** All domain names enrolled are reregistered with store information about the scale and location of the DNS servers for this domain and if there is a need to change this information the owner's permission must be taken first. The owner is able to change the registrant to another registrar, but this requires the approval of the owner and the old registrar and the new registrar together. If there is a gap in the system of registration of domain names, the hacker can exploit this vulnerability to change the information or to transfer it to another registrar.
Man in the middle (MITM) Hacking Online Banking and Credit Card Transactions

The Scenario

You get internet through Wi-Fi Hot Spot to surf the web. Through the net you check your bank balance and perform some online banking and maybe purchase something online. As an end-user, you feel this is a nice quick service and quite secure, as you see the lock in the bottom corner of your Internet browser, symbolizing that the online banking or online credit card transaction is safe from snooping eyes. What lets you be more confident about your data, including username, password, credit card info, etc. will be that it is encrypted with 128-bit encryption.

A common thought is that doing that is secure, as this is done via SSL. For the most part, this is true and the sessions are secure. The HSBC Bank in 2012 (for example), posts the following security adverts on their website as shown in figure 3.5:

Figure 3.5: HSBC bank post security adverts

The question here is: do the software and tools fix the problem and secure the client completely? Anti-virus, firewalls and other security software are important but unfortunately not enough. Various studies and recent incidents show that these tools are not always effective in preventing criminals from taking money from your account. As criminals become more sophisticated, your bank strongly recommends additional layers of protection on your computer to enable safe online banking.

The problem is that it is not “virtually impossible” for someone else to see your data, such as login information or credit card numbers. It can actually be relatively easy if you as an end-user are not knowledgeable about how you can be exploited and know the signs that this is occurring.
Figure 3.6: fake Secure SSL Session

Continuing with the scenario, an intruder has intercepted the online banking login credentials and credit card information and can log into the online banking website or purchase items with the victim’s credit card via an SSL Man-in-the-Middle (MITM), as can see from figure 3.6 there are fake lock to give fouls indicate that web is secure.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>UT/U1a: Usersurveillance</td>
<td>A</td>
<td>X</td>
<td>X</td>
<td>A</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>UT/U1b: Token/notes theft</td>
<td>A</td>
<td>X</td>
<td>A</td>
<td>A</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>UT/U2a: Hidden code</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>X</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>UT/U2b: Worms</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>X</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>UT/U2c: E-mails with malicious code</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>X</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>UT/U3a: Smartcard</td>
<td>X</td>
<td>X</td>
<td>A</td>
<td>A</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>UT/U3b: Smartcard reader manipulator</td>
<td>X</td>
<td>X</td>
<td>A</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>UT/U3c: Brute-force attacks with PIN calculators</td>
<td>X</td>
<td>X</td>
<td>A</td>
<td>A</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
### Table 3.1: Authentication Mechanisms Vulnerability (32)

From Table 3.1 can show applicability of Attacks in Different Authentication Mechanisms
3.5 General defences
3.5.1 Static Password

The most common authentication mechanism, the static password, is based on proof by knowledge. Password-based mechanisms are widely utilized in Internet banking applications. Even in the case of custom non-web-based Internet banking software, the authentication process is usually password-oriented. This mechanism is prone to all types of attacks, excluding those customized to smart cards, which are not applicable, and IBS2, which concerns an internal attack. Capture, replay, guessing or phishing are common defective attacks. It is taken for granted that password authentication takes place through a Secure Sockets Layer (SSL) channel after authenticating IBS, which is a standard approach that banks follow. Static passwords, however, may be captured when attacking the secure channel establishment process (e.g., by deploying sniffing attacks, where the SSL channel is split into one SSL channel between the UT and the attacker and another between the attacker and IBS). The splitting may be performed in one-way SSL authentication (using only the certificate of the IBS) by sending fraudulent certificates to the user, who usually does not check the true origin and validity [32, 41, and 53].

Soft-token Certificate/SSL-TLS

This mechanism conducts mutual authentication between the UT and IBS, based on certificates in the user’s browser. The mechanism is prone to UT/U2 attacks, which compromise the UT, where the user certificate is stored. The latter may permit access to the user certificate, resulting in identity theft.

Hard-token Certificate/SSL-TLS

This mechanism is based on the “proof by possession” principle, since the user possesses an object as a token toward authentication [94]. The use of hardware tokens addresses the vulnerabilities of storing the certificate in the user’s browser. This mechanism, however, is prone to a number of hardware (UT/U3) and software attacks, which analyze the operation of the hard token. Other attacks (UT/U2) exploit vulnerable interfaces of the hard token, which may permit unauthorized hard-token commands, such as digital signing, leading to identity theft. Token theft (UT/U1b) is also an issue, but it has to be combined with the PIN compromise.

3.5.2 One-Time Password/Time-Based Code Generator

This mechanism may also fall into the category of proof by possession authentication mechanisms [96]. One-time passwords are generated by a random calculator, using a seed that is pre shared between the user’s device (protected by a PIN) and the IBS. In mobile banking,
Chapter 3: Security and Authentication Access Control

the random codes may be submitted to the UT by the mobile operator through SMS, or generated by an application downloaded to the user’s mobile device. The user reads the SMS or software/hardware-generated code and provides the one-time code to the banking application for authentication. These mechanisms are prone to a number of attacks, including the regular device theft attacks (UT/U1b), combined with user surveillance (UT/U1a) or notes for obtaining the PIN. More sophisticated attacks are deployed exploiting the PIN time window (UT/U3c). This fact makes brute-force attacks possible (IBS1), since the likelihood of guessing possibilities is increased. Other attacks (UT/U2, CC, IBS3, UT/U1a) are also possible since the codes are submitted through the browser, but for a very limited time frame (as long as the code is active plus the time window). Hardware attacks (UT/U3) are also possible by deploying the hardware analysis methods presented in the attack tree description.

3.5.3 Challenge-Response

This mechanism is similar to the previous one, adding uniqueness to the authentication process. IBS generates a challenge that is processed by the UT/U for producing a response. Challenge-response is prone to man-in-the-middle and session hijacking attacks (CC), since an entity may intercept the communication between the UT/U and IBS, and capture and replay messages from the predefined session method described in the attack tree description. This mechanism covers all root node attacks facing the user terminal and user; unfortunately it fails against the communication channel attack.

3.5.4 Biometrics

Biometrics [76], provide strong authentication by adding the “proof by property” principle, completing or substituting the existing “proof by knowledge” and “proof by possession” mechanisms. Biometrics-based mechanisms, however, should be created by following best practices and standards to reach an acceptable security level. Internet banking applications incorporating biometrics have been introduced by realizing the common scenario of identity verification by the remote comparison of a biometric template (comparison of the enrolled template at IBS, with a template generated by a biometric measurement at the time of authentication). Other state-of-the-art scenarios describe local biometrics authentication for gaining access to a device or token. Biometrics are prone to malicious code attacks (UT/U2), phishing attacks (UT/U4) and communication links (CC) that may expose the user’s personal data and threaten the whole security chain, since biometric templates may be compromised and replayed. For example, phishing techniques in architectures that require remote biometric template comparison may request the submission of the user template (even in encrypted form) and resend them to the IBS for compromising
the user’s bank account.

3.5.4.1 Fingerprint

3.5.4.1 Fingerprint Encryption Technology
Fingerprint encryption technology is a biotechnology based tamper proof authentication technology used in our application. It translates the fingerprint into a fixed length encryption key through the fingerprint encryption algorithm, and then generates the fingerprint features value encrypted by the encryption key. Every value is unique for every enrollment. Thus, the fingerprint encryption technology is considered to be a high-level security method in authentication systems.

3.5.4.2 Fingerprint Principles
The tamper proof fingerprint authentication factor is a popular biometric trait used for recognizing a person. Properties which make fingerprints popular are their wide acceptability in public and ease in collecting the data.
Genrally speaking every fingerprint consists of a number of ridges and valleys. Ridges are the upper skin layer segments of the finger and valleys are the lower segments. The ridges form so-called minutia points: ridge endings – where a ridge ends – and ridge bifurcations – where a ridge splits.
At registration-enrollment the minutia points are located at the device and the relative positions to each other and their directions are recorded. This data forms the template to authenticate a person. At the matching stage, the incoming fingerprint image is pre-processed and the minutia points are extracted. The minutia points are compared with the registered template, trying to locate as many similar points as possible within a certain boundary. The result is usually the number of matching minutiae. A threshold is then applied, determining how large this number needs to be for the fingerprint and the template to match. Fingerprint have general characteristic ridge patterns that allow them to be systematically identified.

Fingerprints follow three fundamental principles:

1- Fingerprint is an individual characteristic
2- No two people have been found with the exact same fingerprint pattern
3- Fingerprint pattern will remain unchanged for the life of an individual; however, the print itself may change due to permanent scars and diseases.

3.5.4.3 Ridge Characteristics
Fingerprint characteristics can include sub-areas of certain interest including ridge thickness, curvature, or density. Due to this increased depth of data a pattern-based algorithm is less dependent on the size of the fingerprint sensor and is independent of the number of minutiae
points in a fingerprint. Pattern-based algorithms do not, to the same extent as minutia-based methods, suffer from difficulties of recognizing a finger with varying fingerprint quality.

Fingerprints also have minutiae points, which are points where the ridge structure changes. These are useful in matching a fingerprint to a specific person and figure 3.7 define all characteristic

- Core
- Ending Ridge
- Short Ridge
- Dot or Island
- Fork or Bifurcation
- Enclosure
- Bridge
- Hook
- Delta
- specialty
- Crossover
- Eye

**Figure 3.7:** Fingerprint characteristics

### 3.5.4.4 Fingerprint Classes:

The graphical image is obtained from a capture device to distinguish it from a template stored in a database. Processing software examines the fingerprint image and locates the image center, which may be off-center from the fingerprint core. The image is then cropped a fixed distance around this graphical center. The cropped region is then compressed and stored for subsequent matching.

Fingerprints can be classified into three different groups based on the pattern of the ridges.

#### Arches

- **Ridges enter on one side & exit on the other side**

  - Plain Arch
  - Tented Arch

#### Loops

- **Ridges enter on one side & exit on the same side**
Chapter 3: Security and Authentication Access Control

Whorls

Consist of circles, more than one loop, or a mixture of pattern types

Plain Whorl

Central Pocket Whorl

Double Loop Whorl

Accidental Whorl

The verification procedure in figure 3.7 begins with the pre-processing of the incoming fingerprint image. The registered small images from the template are then compared with the fingerprint image to determine to what degree the template matches the image. A threshold describing the smallest allowable deviation is then used to decide if the finger matches the stored template.

3.5.5. Knowledge-Based

This mechanism consists of a number of questions that the user has to answer to gain access to the account. This could be also considered as behavioral biometric authentication, depending on the content of the questions or, instead, a more sophisticated combined password mechanism. Although this mechanism is resistant to UT/U1 attacks, it is prone to most of the other attack types, since malicious code may copy answers and create a knowledge database for the user (UT/U2). The user may also answer questions from an attacker, due to being tricked by phishing (UT/U4), man – in – the – middle attacks (generally CC) or malformed IBS websites (IBS3). Brute- force attacks are difficult to deploy due to the multitude of possible answers.

a. An analysis of e banking vulnerability
   i. Attack tree
iii. Active vector
iv. Vulnerability analysis
v. Credential harvesting

b. Banking provided measurement
c. Define effectiveness
d. Efficient cryptographic protocol preventing man in the middle attack

3.5.6 Current Solutions to Solve the Problem

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Bank A</th>
<th>Bank B</th>
<th>Bank C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server authentication</td>
<td>SSL/TLS</td>
<td>SSL/TLS</td>
<td>SSL/TLS</td>
</tr>
<tr>
<td>User authentication</td>
<td>ID/ password OTP</td>
<td>ID/ password / third parties</td>
<td>ID/ password</td>
</tr>
<tr>
<td>Data integrity</td>
<td>SSL / TLS</td>
<td>SSL / TLS</td>
<td>SSL / TLS</td>
</tr>
<tr>
<td>Non – reputation</td>
<td>-</td>
<td>Digital signature</td>
<td>-</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>SSL / TLS</td>
<td>SSL / TLS</td>
<td>SSL / TLS</td>
</tr>
<tr>
<td>Malware detection</td>
<td>Anti-virus</td>
<td>Anti-virus</td>
<td>Anti-virus</td>
</tr>
<tr>
<td>Network access control</td>
<td>Firewall</td>
<td>Firewall</td>
<td>Firewall</td>
</tr>
<tr>
<td>Anti – key logger</td>
<td>Keystroke enc</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 3.2: Current Solutions functionality and ranking to solve the banking problem [32]

There is some technology in general use outside of the Internet banking field which can be used to secure online transactions as can see in table 3.2.

**Using digital certificates:** a digital certificate is an electronic document submitted digitally to bind the public key with the identity you want to publicize and therefore prevent plagiarism or forgery, which might happen to those who hold public and private transactions relating to financial aspects. But there are a number of problems related to digital certificates:

**The process of renewing certificates:** when a certificate expires, to renew it the customer must go personally for approval of a new certificate and to verify their identity. But this renewal
process is creating a nuisance, so this method has been replaced by dealing with the granting of certificates, but the threats are many; the attacker can receive the certificate renewal request, and then respond that it is responsible.

**Display private key disclosure:** the private key may not change for a whole year and this gives more opportunity for hackers to detect that key.

![Diagram of security PKI certificate flow](image.png)

Figure 3.8 security PKI certificate flow

From figure 3.8 shows the sequence flow of certificate from client A until reach bank server cross CA which is cloud provider in our case

**Hardware protection unit (Hardware Security Module):**

This method is based on using the user information task such as the private key and certificates and keys that are exchanged in the SSL. It also has many different functions, which include the establishment of an electronic signature private key, as well as creating the electronic signature and the generation of random numbers and all these functions are programmed directly on physical hardware. Problems related to this method are:

i) **High cost.**
ii) **Loss of flexibility**: If you find a weakness in the encryption algorithm used, it is difficult to amend on this machine because the ratio has been programmed in physical assets.

iii) **Failure and damage**: difficulty maintaining a secret key if the device has been lost and used by someone else and if a damaged device must be replaced by another device.

iv) **Verification via mobile phone (Mobile Phones)**: The idea of using mobile phones in the verification process integrated with a third party far from cyberspace. When the username and password are sent to your bank, the banks send a special code to your mobile phone number already registered in the bank database. Through that message it confirms that the bank’s site is correct and performs normal banking operations but the problems with this solution are as follows:

- Dependence on third party: you may get problems with reliance on telecom companies to provide services related to sensitive matters such as money transfer.
- With the entry of third generation mobile phones they have become a part of the internet so mobile devices have become equally threatened by viruses, e.g. Trojan.
- Solution assumes that anyone who wants to use this technique should acquire a mobile phone.

The mobile phone solutions are as follows:

1. The client starts to connect to the desired location using the computer and by the Protocol (Hypertext Transfer Protocol Secure) HTTPS it logs on to the bank page. The client’s computer sends a message (Hello) to the bank, where the agreement is available when the client is on the algorithms that will be used for encryption.
2. The bank’s letter (Hello) is sent to the client, the bank approves algorithms that have been chosen by the client and the certificate is requested.
3. Building on the success of the bank certificate verification, a message appears in the client mobile phone to request the client to enter a personal identification number (PIN) in order to ascertain that the bearer of this phone is a real client.
4. The client shall enter the PIN.
5. Subsequently, the client certificate is merged with its first key and the signed public key and is encrypted using secure socket layer for the bank, after verifying the PIN.
6. The bank verifies the client certificate signed by decryption via public key, and then the bank
generates a session key, according to the process of generating the information distributed between them.

7. The client computer generates the same session key, which will be used in encryption and decryption. At this point, the verification ends between the bank and the client.

**One-time password OTP** [68]: another idea relying on providing a device that generates a password one time only is known as PAL (OTP One Time Password). This idea has been popular with proven efficiency and reduces damage to recipients across the banking services. The scenario is as follows: the client opens a device for generating a code after entering their user name and password on the site; they then enter the bank’s code required by the bank and generated from this device, often a process code can be connected to the network to generate HTML code or there is a static algorithm in that device to let the bank verify the client. This allows the client to be sure that the page is actually its own bank’s page, not a fake page. On the other hand, we need to note the high cost of the machine and reduced ease of use for the client.

**TLS**

There is a mechanism for similar certificates to provide client authentication, but very few sites use it and very few users have appropriate certificates.

Because of the extensibility of browser platforms, it is often possible to provide fake TLS indicators that will pass even close scrutiny. In addition, due to the costs associated with acquiring TLS certificates many sites have certificates that are not rooted in the trusted certificates shipped with the browser. When presented with such certificates, browsers will typically prompt the user to accept them. Such prompts do not provide information that will allow the majority of users to make an appropriate security decision since they do not understand the terms used. In such cases the user, who just wants to access the content, has the habit of accepting any question they are asked if it will allow them to do what they want to do.

**Spam filtering**

There are many forms of unsolicited bulk email (spam) and it is a large problem in itself. This has led to a number of ‘classifiers’ [51, 55] which try to analyse features of emails to assign a score corresponding to the likelihood of the email being spam. This score is used to discard mail whose score is above a given threshold. Since for a large amount of phishing attacks initial contact is performed via unsolicited email claiming to be from a bank these are potential targets for being discarded by a spam classifier.

Spam classifiers work by trying to identify features in which legitimate email and spam differ. This is known to be a hard task for a number of reasons:
Software cannot generally know what email a user does not want to receive. Many emails which the user wants to read they do not know in advance that they are going to receive.

**Password wizards**

All major web browsers today give the option to remember passwords for websites and to auto-complete the log-in forms so that the user does not have to remember the password. Traditionally, this has been regarded as security vulnerability, since the password can be recovered from where it has been stored in the web browser. However, there may be cases where it improves security. The web browser is much better than the user at distinguishing whether a site is the same site that it has visited previously. While the user might be tricked by URLs that are similar but not the same, the web browser will not. If the user only ever logs in using the saved password then it will be obvious when they visit a phishing website, since the browser will not offer to complete the form.

**Take down**

One of the main defenses today is the completely reactive one of take down. There are a number of groups monitoring phishing emails and websites. Suspicious ones are then reported to the ISPs in question and removed. By definition, there will be some lag between a new phishing website being released and the site being removed in which victims can fall foul of the scam. However, this is not the only problem. A number of ISPs have been lax at responding to take-down requests [16], which increases the window for successful scams.

Some phishing gangs have developed phishing attacks using a large number of websites hosted on compromised machines. Thus, each email can be sent out with a different link to the host, which makes the observers’ work much harder.

**Web browsers**

Since the main part of a phishing attack happens through web browsers, the leading browser manufacturers have all come up with defenses in the latest versions of their products. All of the software solutions are ineffective against compromise of the user’s computer, but some of them combat other threats.

**Trust Bar**

TrusstBar [55] is a combination of a black list and some enhancements to TLS. The black list is similar to all of the others; there is an option to report sites that are vetted by the TrustBar authors to be added to the black list. Since this is a black list specific to TrustBar, as with eBay Toolbar, it will evolve more slowly than other lists.
DSS

Dhamija and Tygar have proposed a system called Dynamic Security Skins [32] (DSS). This combines the personalization present in YURL pet names and the synchronization from SRD with a trusted input window to create quite a convincing anti-phishing solution.

The idea behind DSS is that the user only enters their credentials into an input window provided by DSS, not directly into the web page. When installing DSS, the user selects a ‘skin’ for the input window. This is a custom image that is always displayed behind the user name and password fields in the input window which makes it hard for an attacker to spoof the input window, since they cannot replicate the image. The input window also displays a dynamically generated pattern (called a visual hash) which has been created in cooperation with the server and is mirrored on the input form to which the input dialog is currently attached. In DSS, authentication is done not by sending a password to the server, but by using the SRP [75] protocol. SRP is a form of Zero-Knowledge proof where the server can authenticate the user without a password being sent over the network. The result of the SRP authentication is a hash which is only known to the server holding the verifier and to the client with the password. This hash is used to generate a visual hash which is displayed both on the page and in the trusted input box.

Because the password is not sent to the server, a middle-person attack cannot learn the password or know what images to display on a spoofed webpage such that they synchronize with the trusted input window. DSS is not designed to protect against a compromised end station or one with any sort of keystroke logger.

TANs

Transaction Authorization Numbers (TANs) [32] are one-time passwords used by some banks. The customer is issued with a sheet containing multiple passwords; typically this is done via some out-of-band mechanism, usually the postal service or in person at a bank branch. For each transaction, the next TAN in order is requested in order to complete the transaction.

TANs are either essentially the same as Lamport hash chains [32] used by authentication systems like One-time Passwords-[68] or more simply just a list of random numbers which are stored on the server. As with other one-time password authentication systems, TANs protect against snooping of the credentials on a legitimate transaction in order to use them later. However, TANs are no defense against a middle-person attack in which the victim is convinced to connect to the attacker instead of the real bank and the transaction challenges and responses are forwarded between the real bank and the user, but the transaction details are altered to transfer funds to the
attacker instead. There have also been cases of phishing websites harvesting several codes for later use, without the complexity of a middle-person attack [32].

**SecurID**

The RSA SecurID range raises the bar slightly for attacks on online transactions. However, none of the solutions prevent middle-person attacks, which are in current use. The USB variants fail particularly badly in the presence of a compromised terminal.

**CAP**

One form of multi-factor authentication which banks are starting to introduce derives from the EMV (Chip and PIN) specification [100] and is called the chip authentication protocol [100]. This is currently being rolled out in the UK by Barclays [32].

The CAP reader is a handheld device very similar to a calculator. It has a gap in the top for inserting a credit or debit card, a PIN pad and a small display on the front. It uses the feature of the EMV protocol whereby the card will produce the message authentication code (MAC) of data with which it is supplied using the secret key it shares with the bank. There are two primary modes of operation with CAP: generating random nonces and challenge/response.

**Random nonce generation**

The first mode is used for most types of transactions. The card is used to generate a random nonce. When performing a transaction, the website will ask the user to generate a code with the reader which they do by inserting their debit card and pressing the ‘new code’ button. The CAP reader displays the code on the screen and the user transcribes this into the website and the server checks the MAC is valid. This scheme has many flaws. In particular, it is essentially a weaker form of SecurID. It is susceptible to real-time and off-line middle-person attacks because the user cannot know what they are authorizing. In addition, the code is not time-dependent, so can be cached by the attacker for later use.

**Challenge-Response**

The second mode is a challenge-response mode. Current proposals use this mode for transactions such as setting up new mandates. In this case, the user is asked to enter some details from the transaction as part of the challenge. These are then sent to the card which produces the MAC under the key shared with the bank. The user copies the number into the website as before. This is an improvement over many of the schemes discussed, but unless the user has to enter all the details of the transaction there is still scope for a middle-person attack. In addition, since it is only used for setting up mandates, there are many types of transactions that are still vulnerable.
It has been shown that the common, well-known attacks on Internet banking are not the only attacks that criminals are capable of performing. It was also shown that the majority of the protection currently afforded to Internet banking and similar targets is aimed at stopping the common attacks but not the more sophisticated attacks. Even those that claim to be more thorough do not successfully prevent the exploits. The particular attack to which there is no robust defense is that of compromise of the computer used to access the online banking service. The Trojan is the most powerful attack vector available to attackers and the most difficult to defend against. It is no wonder, therefore, that many systems for protecting online banking have chosen to conveniently ignore this in their attacker model. For a while, this was an acceptable amount of risk because that class of attack was relatively uncommon. Even now it is not wide-spread, but the instances of it which have been seen, suggest that as less complicated attacks become less successful there will be no problem with the attacker switching to using Trojans.

What is required is a defense which is robust even in the presence of a Trojan on the computer being used to access the service. The attacker places himself between the victim and the target of the attack. The classic middle-person attack involves the attacker somehow spoofing the target so that the victim connects to the attacker by mistake. The chosen protocol attack is, however, much more general than that. Solutions for this involve ensuring that messages are unambiguous as to their purpose and destination and not allowing any one to use the same authentication system. The problem with this is that there is a growing desire from consumers and merchants for a single system they can all use. In addition, the unambiguity of the message must be clear to the user, not just to the computers involved. This poses a problem when the assumption is that the computer in use is compromised. This lack of transparency is at the root of all phishing and pharming attacks. If the user could be sure they work in a healthy environment and completely isolated, fraud would not be possible.

3.5.7 Related Solution

Vasco - DIGIPASS Nano®

Vasco Company announces their product as:

Mobile banking applications face multiple challenges. In order to gain widespread user acceptance, they must be secure without compromising user-friendliness. VASCO addresses these challenges with its innovative solution DIGIPASS Nano. DIGIPASS Nano enhances the security of online service channels through the use of e-signatures and end-user authentication.

VASCO fulfills a growing need by enhancing the security of online service channels through the use of e-signatures and the authentication of end-users. DIGIPASS Nano® is a reflection of VASCO’s innovation and continuously evolving efforts to bring high-level security to all
environments. The solution allows mobile phone users to embed the best-of-breed authentication technology into a familiar, daily-used device.

With its unique form factor, DIGIPASS Nano offers banks and corporations new perspectives in using mobile devices as authentication means, providing them with a new channel to interact with their customers. Mobile users will be able to perform secure transactions, access business-critical data or transfer money anywhere at any given time.

Figure 3.9 Vasco announcement

Figure 3.9 show Vasco SIM card product and from my view I can say this solution is quite good but depends upon company support and lock-on technology

**VASCO DIGIPASS Token Authentication & Token Management Module**

Access Matrix™ Universal Authentication Server (UAS) is a versatile authentication server which enables organizations to unify their different authentication mechanisms and simplify integration complexities by providing off-the-shelf strong authentication modules for common applications. The out-of-the-box end-to-end token life cycle management module greatly streamlines the administration and management of token logistics.

Figure 3.10 Vasco product

From figure 3.10 above shows that Vasco declarer Supports Full Range of Tokens with Access Matrix™ hierarchy based administration model is also designed for Managed Security Service Providers and SAAS vendors with multi-tenancy support. UAS has been tightly integrated with Vasco’s VACMAN Controller Library to provide all key token features for authentication (both synchronous and challenge response), authorization, signature, verification and host return code. It also provides advanced token management features to address
various aspects of token logistics such as token issuance, pin-mailers, lost tokens, out-of-sync tokens which are critical to the day to day operations.

**End to End Token Management for VASCO Tokens**

UAS-VTM provides a convenient way to manage the entire life-cycle of Vasco Tokens from token assignment, token status tracking, application integration, OTP comparison, and auto re-sync, lost tokens and token replacement.

**Comprehensive Support for VASCO Security Features**

UAS-VTM is seamlessly integrated with Vasco’s VACMAN Controller and provides a set of flexible integration APIs to support the Vasco advanced security features.

**Support Full Range of VASCO DIGIPASS Tokens**

UAS-VTM has been certified to work with the entire product line of Vasco DIGIPASS Tokens: hard, soft, web and mobile tokens.

Most of these features can be used to secure bank transactions but need multi service provided by the company.

**Gemalto eBanking**

Gemalto eBanking are experts in strong authentication. They declare that they provide technology that allows people everywhere to prove their identity and intentions online; safe from harm, with their privacy respected and secure from fraud.

“In doing so we enable cost savings, new revenue models, increased efficiency for retail as well as corporate banks.

However, if you dig deeper than that – what makes everything else possible is trust. The trust between the bank and its online customer, trust in the technology itself and the possibilities it brings, trust as the foundation for a safer and more convenient digital world”.

**Gemalto Ezio Plug&Sign Device**

Figure 3.11 Gemalto products
Chapter 3: Security and Authentication Access Control

From Figure 3.11 shows the variety of Gemalto USB types, Ezio Plug &Sign is a self-contained device that embeds a portable infrastructure, including all applications to securely view, accept and sign transactions.

Figure 3.12: Gemalto Ezio

Figure 3.12 is Gemalto's Ezio Suite brings together a unique authentication server, plug-in modules and a range of authentication devices. The result is a completely flexible, future-proof system that lets you mix and match authentication devices, standards and algorithms. It also supports advanced technology such as Sign-What-You-See.

3.6 Homomorphic Cryptography

A homomorphic encryption scheme is a crypto system that allows computations to be performed on data without decrypting it [52, 73, 78]. In other words it is a form of encryption that allows computations to be passed out on ciphertext —an operation performed on a set of ciphertexts such that decrypting the result of the operation is the same as the result of some operation performed on the plaintexts. A given cryptosystem is considered additively homomorphic iff

\[ \exists \Delta: \varepsilon(x_1) \Delta \varepsilon(x_2) = \varepsilon(x_1 + x_2) \]

Paillier's Homomorphic Cryptosystem:

The Paillier scheme was first published by Pascal Paillier in 1999 [107, 108,109]. The Paillier probabilistic scheme has created a good amount of interest and further study since it was originated. The homomorphic property allows this scheme to do normal addition operations on several encrypted values and achieving the encrypted sum, the encrypted sum can be decrypted later without even knowing the values that made up the sum. This thesis used Paillier homomorphic because it is light and easy to use without many complicated mathematic tools required like other homographic cryptosystems.
3.7 Summary

In this chapter, I have reviewed security from the general network perspective describing the basic security concepts required for network security implementation. The possible network threats were then highlighted while the security services used to counter the threats and the cryptographic algorithms implemented by these security services were also discussed. I then gave an insight into security issues in online banking. In the course of the discussion it was discovered that the notion of security in a general network is not different from that in the online banking network. The implementation approach however differs because of the conflicting design assumptions of online banking and Internet which calls for a new concept in security implementation. Identified attacks, factors affecting online banking security and key research challenges have also been discussed. A closer look at the identified tree attacks in the banking sector reveals that having access operations is desirable. I then presented an overview of control with focus on implementation requirements, the two elements of authentication and authorization, existing works with consideration for these elements and finally establishing the similarity between the two established security companies making new security products as well as the applicability concepts in authentication for access implementation in the banking environment.

The next chapter presents an overview of security validation, reviews of some existing network simulation models and discusses the simulation framework development process.
Chapter 4 New Secure Financial Model using public and private clouds

This chapter describes and discusses the three entities combined to create my proposed connectivity for online banking application then the prototype of appliance and the topology. The chapter is structured as follows: Section 4.1 is the prototype of appliance sequence and topology. Section 4.2 reviews Secure Transaction Sharing (STS) proposal and problem statement for fine-grained access control. Section 4.3 introduces and describes an authentication scheme and the authentication code. Section 4.4 presents the bank involved URLs in this process.

Cloud computing was used to transfer data from client to the bank as out-of-band channel; we used cloud because Cloud computing technology can provide banks competitive advantage in the market cost reduction, higher margins, simplified maintenance and management of application, greatly extended scalability, agility, high availability, automation, large data storages and reliable backup mechanism are the clear benefit bank can gain from the cloud provider.

The communication between the bank client and the cloud go through a public cloud and is secured by many security parameters like SSL client, fingerprint authentication factor and session key.

The cloud is used to transfer data from client to the bank and from the bank to client; well-established holomorphic encryption algorithms applied to allow cloud storage handle data encrypt prior send to cloud, and decrypt it after moving it back to the bank or client. Used Private Cloud to communicate between the bank and cloud provider, hence the cloud infrastructure is operated solely for the bank and managed by the bank. It will be secure by the bank security policy and equipment.

The three entities combining to create connectivity for online banking application are:

- Public cloud (cloud provider)
- Private cloud (connectivity relation interface between bank and cloud provider)
- Internet Banking (the bank interface link between client and the bank application)

At the first step the client inserts his/her tamper proof device; the device contains the bank application under the (Puppy Linux) OS platform with auto run application interface; then it asks the client to enter a username, password/pin, fingerprint as can be seen from figure 4.1 and the system compares automatically the biometric fingerprint (tamper proof companies with fingerprint screen) and tamper proof ID.
There are six steps:

1. Unique tamper proof ID automatically acquired to check public provider and at the same time create a hash value for other authentication factors combined with unique Tamper proof ID. Username, pin number or ID number, and finger print all encrypted by constant key given to client when asked to open account.

2. Cloud provider matched the authentication factor; cloud providers send these values to the bank system through a private connection after success matching.

3. The bank checks these values by decrypting the message from the client using the same key; if matched, then a one time password (OPT)/ challenge response and random key is generated (to make a second channel between the bank and client directly encrypt all other values by this key except username).

4. Send this random key to the bank interface and to public cloud to send again to the client.

5. The client then responds to this parameter to create a new session using the key sent from the bank but the username is encrypted by a constant key and band with other values encrypted by a random key generated by the bank’s private key.

6. Check these values by decrypting the username using the same constant key and decrypting other values using the random key; if there is a match then the client can access the account.

### 4.1 Prototype of Appliance

1. Customer plugs in tamper proof to access their account
2. Puppy Linux operating system pops up and opens pre-bank Interface
3. Client enters four part authentication factor
   a. User name
   b. Account number / pin number
c. Password (combined with tamper proof ID feed automatically)
d. Fingerprint

4. Client submits request
5. Cloud provider checks tamper proof ID and fingerprint
6. Cloud provider sends request to the bank
7. Bank returns Order ID (OTP/challenge response) and Session ID
8. Cloud provider saves Order ID and Session ID
9. Cloud provider redirects to the client received URL
10. Cloud opens First Authentication for the client
11. Client uses session ID and other fingerprint factor
12. Client submits authentication
13. Bank checks authentication
14. Bank opens Second Authentication challenge/response and OTP for the client
15. Client responds to new authentication
16. Bank approves client authentication
17. Client Success

From the diagram in figure 4.2 the client tries to access the account by providing username, password, T.P ID and fingerprint and sends the request to the public cloud. The public cloud checks just T.P. ID and fingerprint. If it is right it passes the request to the bank through a private cloud connection. The bank checks the parameters then passes the session key through the cloud to access the internet bank application to start a direct connection to the client.

![Diagram](image-url)

Figure 4.2: client authentication factor flow
4.2 The Proposed Security Scheme

A new protocol, Secure Transaction Sharing (STS), provides the bank with fine-grained access control over data outsourced to the cloud environment. STS allows the bank to authorize Clients to securely access their data. We will then extend this solution to address the problems of collusion between the CSP and a client.

4.2.1 Proposed STS Framework

STS protocol is constructed using additive homomorphic encryption as an operation performed on a set of ciphertexts such that decrypting the result of the operation is the same as the result of some operation performed on the plaintext. Homomorphic Addition given the cryptosystem is considered additively homomorphic iff
\[ \exists \Delta : \varepsilon(x_1) \Delta \varepsilon(x_2) = \varepsilon(x_1 + x_2) \]

4.2.1.1 Paillier Additive Homomorphic Cryptosystem

The Paillier cryptosystem, named after and invented by Pascal Paillier in 1999, is a probabilistic asymmetric algorithm for public key cryptography [107, 108, 109]. The scheme is an additive homomorphic cryptosystem: given only the public-key and the encryption of \( m_1 \) and \( m_2 \), one can compute the encryption of \( m_1 + m_2 \).

**Definition 1.** Let \( k \in \mathbb{Z} \) be some number. The sets \( \mathbb{Z}_k \) and \( \mathbb{Z}_k^* \) are defined as
\[
\mathbb{Z}_k = \{ z | z \in \mathbb{Z}, 0 \leq z < n \} \\
\mathbb{Z}_k^* = \{ z | z \in \mathbb{Z}, 0 \leq z < n, \gcd(z, k) = 1 \}.
\]

The corresponding multiplicative group will be denoted \( (\mathbb{Z}_k^*, \cdot) \), with modulo \( k \) multiplication.

**Definition 2.** \( n = pq \) will be the product of two primes \( p \) and \( q \).

**Definition 3.** A number \( z \equiv n \notin Z^* \) is said to be an \( n \)-th residues modulo \( n^2 \) if there exists a number \( y \equiv n \notin Z^* \) such that \( z \equiv y^n \mod n^2 \).

Let \( NR \) be the set of \( n \)-th residues modulo \( n^2 \).

**Definition 4** (Euler’s \( \Phi \)-Function). Let \( x = \prod_{i=1}^{\ell} (p_i - 1) \) with \( p_i \) pairwise different prime numbers.
Euler’s Φ–Function is defined as
\[ \Phi(x) = \prod_{i} p_i^{j_i-1}(p_i - 1). \]

**Lemma 5.**
\[ \Phi(x^2) = x\Phi(x) \]

**Lemma 6.** For any \( x \in \mathbb{Z}_n \), \( (1 + n)^x = 1 + xn \mod n^2 \)

These are roots of unity in \( \mathbb{Z}_{n^2}^* \)

Proof by induction over \( x \). Basis \( x = 0 \) is obviously true.

Inductive step \( x \to x + 1 \):

\[ (1 + n)^x = 1 + xn \mod n^2 \]

\[ \Rightarrow (1 + n)^{x+1} = (1 + xn)(1 + n) \mod n^2 \]
\[ = 1 + xn + n + xn^2 = 1 + (x + 1)n \mod n^2 \]

Algorithm

The scheme works as follows: \( g \in \mathbb{Z}_{n^2}^* \)

**Key generation**

1. Choose two large prime numbers (often 512 bits or above) \( p \) and \( q \) randomly and independently of each other such that \( \gcd(pq, (p - 1)(q - 1)) = 1 \). This property is assured if both primes are of equal length.
2. Compute \( n = pq \) and \( \lambda = \text{lcm}(p - 1, q - 1) \)
3. Select random integer \( g \) where \( g \in \mathbb{Z}_{n^2}^* \)
4. Ensure \( n \) divides the order of \( g \) by checking the existence of the following modular multiplicative inverse: \( \mu = (L(g^\mu \mod n^2))^\lambda \mod n^2 \),

Where function \( L \) is defined as: \( L(\mu) = \frac{\mu - 1}{n} \)

Note that the notation \( a \div b \) does not denote the modular multiplication of \( a \) times the modular multiplicative inverse of \( b \) but rather the quotient of \( a \) divided by \( b \), i.e., the largest integer value \( \mu \geq 0 \) satisfies the relation \( \mu \mid b \).
The public (encryption) key is \((n, g)\).

The private (decryption) key is \((\lambda, \mu)\).

If using \(p, q\) of equivalent length, a simpler variant of the above key generation steps would be to set \(g = n + 1, \lambda = \alpha^{(n)}\) and \(\mu = \alpha^{(n) - 1} \mod n\), where \(\alpha^{(n)} = (p-1)(q-1)\).

**Encryption**

1. Let \(m\) be a message to be encrypted where \(m \in \mathbb{Z}_n\)
2. Select random \(r\) where \(r \in \mathbb{Z}_n^*\)
3. Compute ciphertext as: \(c = g^m \cdot r^n \mod n^2\)

**Decryption**

1. Let \(c\) be the ciphertext to decrypt, where \(c \in \mathbb{Z}_n^* n^2\)
2. Compute the plaintext message as: \(m = L (c^\lambda \mod n^2), \mu \mod n\)

Decryption is "essentially one exponentiation modulo \(n^2\)."

### 4.2.1.2 Homomorphic Properties

A notable feature of the Paillier cryptosystem is its homomorphic properties. As the encryption function is additively homomorphic, the following identities can be described:

- **Homomorphic addition of plaintexts**

  The product of two ciphertexts will decrypt to the sum of their corresponding plaintexts,
  \[
  D (E (m_1, r_1) \cdot E (m_2, r_2) \mod n^2) = m_1 + m_2 \mod n.
  \]

  The product of a ciphertext with a plaintext raising \(g\) will decrypt to the sum of the corresponding plaintexts,
  \[
  D (E (m_1, r_1) \cdot g^{m_2} \mod n^2) = m_1 + m_2 \mod n.
  \]

- **Homomorphic multiplication of plaintexts**

  An encrypted plaintext raised to the power of another plaintext will decrypt to the product of the two plaintexts,
  \[
  D (E (m_1, r_1)^{m_2} \mod n^2) = m_1 \cdot m_2 \mod n,
  \]
\[ D \left( E (m_1, r_1)^m \mod n^2 \right) = m_1 \mod n, \]
More generally, an encrypted plaintext raised to a constant \( k \) will decrypt to the product of the plaintext and the constant,
\[ D \left( E (m_1, r_1)^k \right) \mod n^2 = km_1 \mod n. \]

However, given the Paillier encryptions of two messages there is no known way to compute an encryption of the product of these messages without knowing the private key.

**Background**
Paillier cryptosystem exploits the fact that certain discrete logarithms can be computed easily.

For example, by binomial theorem,
\[
(1 + n)^x = \sum_{k=0}^{x} \binom{x}{k} n^k = 1 + nx + \binom{x}{2} n^2 + \text{higher powers of } n
\]

This indicates that:
\[
(1 + n)^x \equiv 1 + nx \pmod{n^2}
\]

Therefore, if:
\[
y = (1 + n)^x \mod n^2
\]

Then
\[
x = \frac{y - 1}{n} \mod n
\]

Thus:
\[
L \left( (1 + n)^x \mod n^2 \right) = x \pmod{n},
\]
Where function \( L \) is defined as \( L (u) = \frac{u - 1}{n} \) (quotient of integer division) and \( x \in \mathbb{Z}_n \).

**4.3 Security Semantics**
Irrespective of the above-mentioned homomorphic properties however, the system is malleable, and therefore does not adopt the highest echelon of semantic security that protects against adaptive chosen-ciphertext attacks (IND-CCA2) \([73, 92]\). Usually in cryptography the notion of malleability is not seen as an "advantage," but under definite applications such as secure electronic transaction and threshold cryptosystems, this process may not acceptable.

In my proposal I used a homomorphic cryptographic for re-encrypting the message over cloud provider without the need to know the plain text transfer from client to the bank or vice versa. To
avoid the malleability of homomorphic cryptography I obtain cashing re-encryption algorithm combined with this protocol to secure the communication transaction between client and the bank over cloud services.

4.3.1 Statement of Fine-Grained Access Control
Our Fine-grained access control scheme was used to achieve fine-grained data sharing and access control over cloud. The Secure Transaction Sharing (STS) framework uses homomorphic encryption and cashing re-encryption as the underlying subroutines.

Figure 4.3: three participant entities
As can see from figure 4.3 the Diagram showing the three entities participating in my proposal
In our statement setting, the bank encrypts data locally to ensure privacy. Moreover the bank shares the authentication access to cloud provider as out of band and other authentication channel access. To facilitate fine-grained access control, asset of attributes is associated with each client’s account transaction, which helps to control client access to a specific set of account fields for each authorized user. The bank issues a decryption key tamper proof for each authorized client account according to access rights.
Subjects include re-authorizing a revoked client account who later rejoins the system with potentially different access privileges, potential collusion between a revoked client and an authorized other client account, and collusion between a client and the Cloud Service Provider as discussed later in this chapter.
4.3.2 Private Cloud (bank trusted zone):

Bank trusted zone is an area service that receives four variables and checks all these variables; if matched then it returns other parameters. Also this function generates a random key for encryption and sends this key to the bank and to the client upon a one-time password (OTP) algorithm with a challenge response feature.

4.3.3 Authentication Schemes

Authentication is the process of validating something as authentic. Whenever the user tries to access a resource, the server will check if the user has appropriate permissions to access the resource or not. Network eavesdropping attacks consist of monitoring network traffic between two target nodes. The attack is achieved by joining the network of one of the two target nodes, such as a hub LAN, switched LAN, or WLAN. Once the attacker is in the network, they use different techniques, depending on the type of network, to start capturing network traffic. For instance, in networks composed of hubs, the attacker just needs to run a network sniffer in promiscuous mode; in switched networks an extra condition, such as an ARP poisoning attack, is required. Wireless networks require the attacker to join the network before executing the attack on the hub or switch behind the access point. If encryption is used on the wireless access point, it would need to be circumvented prior to further testing. All of these variations have different protection methods, depending on the attack type. Testing for them is done with the aid of network attack and monitoring tools. My proposal used a one-time password and challenge response to mitigate this risk plus generating another encryption key.

Three types of one-time password are used in my experiment:

- A mathematical algorithm to generate a new password based on previous password
- Time – synchronization between the authentication server and user
- Mathematical algorithm for new password based on challenge and control

The authentication code

U: user
S: server (cloud)
A: an attacker
UC: user account
PW: password
Chapter 4. New financial model and definition

Dong.ID: tamper proof identity
FP: fingerprint
H: one way hash function
N: random number
⊕ or number
U __ S : transmitting U to S authenticated channel

Scheme consists of two phases:
1- Registration phase
2- Authentication phase

**Phase one**
Registration phase

R1: A user (U) input <UC, PW, FP>, tamper proof identity (generate automatically)

Generate three random numbers

\[ N_0, N_1, N_2 \]

U Tamper proof N₁, N₂

Calculate \( G_0, G_1, G_2, D_1, D_2 \) by

\[ G_0 = h(UC, PW, FP, N_0) \]
\[ G_1 = h(UC, PW, FP, N_1) \]
\[ G_2 = h(UC, PW, FP, N_2) \]
\[ D_1 = h(G_0, G_1) \]
\[ D_2 = h(G_1, G_2) \]

R2: user sends < UC, G₀, G₁, D₁, D₂ > to cloud

R3: cloud stores the received message < ID, G₀, G₁, D₁, D₂ >

**Phase two** (authentication)

In this phase the user requests the \( i \)th service.

U: execute \( i \)th authentication session

U finishes the \((i - 1)\)th authentication session of the scheme
U cloud \( <N_i, N_{i+1}> \)

- The bank sends to cloud provider the public (encryption) key which is \((n, g)\).
- The private (decryption) key is \((\lambda, \mu)\).

S cloud \( <UC, Gi_0, Gi, Di, Di_{i+1}> \)

The secure authentication as

A1.

U input \((UC, PW, FP)\)

U sends a login request to S cloud

Verify STS Homomorphic algorithm as follows:

Select two large prime numbers \( p \) and \( q \) randomly

Algorithm

The scheme works as follows:

Key generation

Compute \( n = pq \) and \( \lambda = \text{lcm}(p-1, q-1) \).

Select random integer \( g \) where \( g \in Z^*_n \)

Ensure \( n \) divides the order of \( g \) by checking the existence of the following modular multiplicative inverse: \( \mu = L(g^\lambda \mod n^2)^{-1} \mod n \),

Where function \( L \) is defined as \( L(u) = \frac{u-1}{n} \).

Note that the notation \( \frac{a}{b} \) does not denote the modular multiplication of \( a \) times the modular multiplicative inverse of \( b \) but rather the quotient of \( a \) divided by \( b \), i.e., the largest integer value \( v \geq 0 \) to satisfy the relation \( a \geq vb \).

A2

S ------ U Ek (UC \( \oplus \) Ts \( \oplus i \))

S computes \( <UC \oplus Ts \oplus i> \), where \( Ts \) is S’s current timestamp and \( I \) is current session number.
S encrypts \(<\text{UC} \oplus \text{Ts} \oplus \text{i}>\), using the share key K send the form to the bank through private channel.

S send \(E_k<\text{UC} \oplus \text{Ts} \oplus \text{i}>\) to U

A3

\[
\text{U} \quad \text{-----} \quad \text{S} \quad E_k (\text{UC} \oplus \text{Ts} \oplus \text{i}), \quad G_{i-1} \oplus G_i \oplus G_{i+1}, \quad D_{i+1} \oplus D_{i+2} \quad h(D_{i+2})
\]

U decrypt \(E_k (\text{UC} \oplus \text{Ts} \oplus \text{i})\)

\[
\text{U} \quad \text{check} \quad (T_u - T_s) \geq \Delta T \quad \text{if} \quad (T_u - T_s) \geq \Delta T, \quad \text{U} \quad \text{quits the login request}, \quad \text{where} \quad \Delta T \quad \text{is expected valid time interval.}
\]

Otherwise, U generates a new random number \(N_{i+2}\)

\[
\text{U} \quad \text{computes} \quad < G_i, \quad G_{i+1}, \quad G_{i+2}>\quad \text{where}
\]

\[
G_i = h(\text{UC}, \text{PW}, \text{FP}, \text{N}_i)
\]

\[
G_{i+1} = h(\text{UC}, \text{PW}, \text{FP}, \text{N}_{i+1})
\]

\[
G_{i+2} = h(\text{UC}, \text{PW}, \text{FP}, \text{N}_{i+2})
\]

\[
D_{i+2} = h(G_{i+1}, \quad G_{i+2})
\]

U stores \(< N_{i+1}, \quad N_{i+2}>\) instead of \(< N_i, \quad N_{i+1}>\)

U computes \(< \text{UC} \oplus \text{T}_u \oplus \text{i}>\) where \(T_u\) is U’s current timestamp and \(i\) is the current session number

\[
\text{U} \quad \text{encrypts} \quad < \text{UC} \oplus \text{T}_u \oplus \text{i}> \quad \text{using} \quad \text{the share key} \quad K \quad \text{received from the bank via cloud provider}
\]

\[
\text{U} \quad \text{sends} \quad (E_k(\text{ID} \oplus \text{T}_u \oplus \text{i}), \quad G_{i-1} \oplus G_i \oplus G_{i+1}, \quad D_{i+1} \oplus D_{i+2}, \quad h(D_{i+2})) > \quad \text{to} \quad S \quad \text{of} \quad \text{cloud}
\]

A4

Encryption at bank servers

Let \(m\) be a message to be encrypted where \(m \in Z_n\)

Select random \(r\) where \(r \in Z^*_n\)

Compute ciphertext as: \(c = g^m \cdot r^n \mod n^2\)

Decryption

Let \(c\) be the ciphertext to decrypt, where \(c \in Z^*_n\)
Compute the plaintext message as:  \( m = L (c^\lambda \mod n^2). \mu \mod n \)

Decryption is "essentially one exponentiation modulo \( n^2.\)"

S receives from the bank < \( E_k(ID \oplus T_u \oplus I), G_{i-1} \oplus G \oplus G_{i+1}, D_{i+1} \oplus D_{i+2}, h(D_{i+2}) > \)

S bank decrypts \( E_k(ID \oplus T_u \oplus I). \)

S bank checks \( (T_u - T_s \geq \Delta T) \). if \( (T_u - T_s \geq \Delta T) \), S rejects the login request, where \( \Delta T \) is the expected valid time interval.

Otherwise, S computes \( D'_{i+2} = (D_{i+1} \oplus D_{i+2}) \oplus D_{i+1}. \)

S checks if \( h(D'_{i+2}) \) is equal to \( h(D_{i+2}). \)

If \( h(D'_{i+2}) = h(D_{i+2}) \), then S obtains \( G_{i+1} = (G_{i-1} \oplus G \oplus G_{i+1}) \oplus G_{i-1} \oplus G_i \)

S bank computes \( D'_{i+1} = h(G_i, G_{i+1}) \), using the stored \( G_i \) and the obtained \( G_{i+1}. \)

If they are equal,

U is authenticated

S stores < \( ID, G_i, G_{i+1}, D_{i+1}, D_{i+2} > \) in place of < \( ID, G_{i-1}, G_i, D_i, D_{i+1} > \)

Otherwise

S rejects U’s login.

4.4 Authentication Connectivity Link Between Cloud Provider and Bank Data Base

However without loss of generality, let us assume that the Bank’s account is composed of records and each data record \( d \) has \( n \) attributes. Each record \( d \) can be represented as \( d_1, d_n, \) where \( d_i \) is the value of \( i \) the attribute of \( n. \) In order to mask the number of attributes as well as actual attribute values of \( d, \) the Bank selects \( n + m \) random integers (denoted by \( r_1, ..., r_n+m \)) from the domain \( Z_N \) for each record \( d. \) Note that the value of \( m \) may vary for each data record \( d \) and is chosen randomly from \( [1, R_{\text{max}}]. \) where \( R_{\text{max}} \) is an additional security parameter decided by the Bank depending on the security requirements. For example, \( R_{\text{max}} \) can be set to 100 for sensitive data such as client’s lending amount. On the other hand, for normal amount transaction \( R_{\text{max}} \) can be set to 10. Let \( d' = d_1 + r_1, ..., d_n + r_n, r_n+1, ..., r_n+m \) be the new data record generated from \( d \) consisting of \( n + m \) attributes. For convenience, we denoted the \( i^{\text{th}} \) value in \( d. \) For each data record \( d \) that the Bank wishes to outsource, it computes \( d' \) and encrypts it attribute-wise using master public key \( pk_a. \) That is, the Bank computes \( E_{pk_a}(d_i + r_1). \)
Chapter 4. New financial model and definition

$E_{pk_a}(d_n + r_n), E_{pk_a}(r_{n+1}), \ldots, E_{pk_a}(r_{n+m})$. Assume that $E(d')$ denotes the new encrypted data record, that is:

$E_{pk_a}(d') = E_{pk_a}(d')_1, \ldots, E_{pk_a}(d')_{n+m}$

During the implementation, the Bank needs to compute $E_{pk_a}(d')$ only once for each data record $d$. After this, the Bank generates the authorization session key for $E_{pk_a}(d')$ verified by the client tamper proof device. We consider the following two cases for each data record $d$ depending on whether the client is granted access to $d$ or not:

**Case 1:** If the Bank wishes to grant access to the client for asset of attributes $S(\subseteq d)$, then they generate the authorization tamper proof for the client corresponding to $d$ as below:

$$td_b = \{ \text{Client}, rk_{pk_a \rightarrow pk_b}, E_{pk_b}(\alpha_1), \ldots, E_{pk_b}(\alpha_{n+m}) \}$$

Here $rk_{pk_a \rightarrow pk_b}$ is the cashing re-encryption key generated using the cashing re-encryption method in Stage 1. For $1 \leq i \leq n + m$, we define $\alpha_i$ as follows:

$$\alpha_i = \begin{cases} -r_i & \text{if } 1 \leq i \leq n \land d_i \in S \\ -d'_{i} & \text{otherwise} \end{cases}$$

Note that, $d'_{i} = d_i + r_i$, for $1 \leq i \leq n$, and $d_i = r_i$, for $n \leq i \leq n + m$. In general, $N - x$ is equivalent to $-x$ under $\mathbb{Z}_N$.

**Case 2:** If the Bank does not wish to grant access to the client for a data record $d$, then no authorization tamper proof is generated or sent; i.e., $b$ = null. We denote $T^d$ as the list of authorization tamper proofs, generated as explained above, of all authorized users who can access either all or some attributes of $d$. The Bank exports the data $(T^d, E_{pk_a}(d))$ to the cloud. Note that if $T^d$ is null, then it is not included in $T^d$. The overall steps performed by the Bank are to generate $T^d$ using $S$ and $d$ for the Client. The Bank first computes $(T^d, E_{pk_a}(d'))$ for all data records that wish to be outsourced to the cloud. Then the Bank uploads the authorization tamper proofs along with the corresponding encrypted data records to the cloud as a single large chunk, thereby reducing the communication cost to a large extent. In addition, our approach can be easily extended to the scenario where a group of users have the same set of access attributes. Under this situation, the bank generates a single public/private key pair for the group and sends it to all users of the group. Instead of generating tamper proofs for each user in the group for $d$, the bank generates a single tamper proof for the group by simply including the group ID in the tamper proof. For example, consider a group $G$ with two accounts who are authorized to access the same set of attributes in $d$, then the corresponding tamper proof can be given as:
\{G-ID, \text{rkpk}_n \rightarrow \text{pk}_g, E_{\text{pk}_g}(\alpha_1), \ldots, E_{\text{pk}_g}(\alpha_n + m) \}

Where pkg and prg are the public and private key so if group G (note that private key prg is known only to bank, account1, and account2) and G-ID denotes the group ID of G, by this control only the group can access the specific account.

4.5 Testedbed Performance and Evaluation

Used Network Walk toools to analyse the bath, the network utilities and bandwidth

**Bank Involved URLs in the process result**

- https://212.0.141.164:4444/SUDAPAN
- https://212.0.141.164:7777/exec
- https://212.0.141.164:7778/index.jsp
- https://212.0.141.164:7778/runtran.jsp
- https://212.0.141.164:7778/MPI/TransactionProcessing.jsp
- https://212.0.141.164:7778/MPI/ResponseProcessing.jsp
- https://212.0.141.164:7778/mpirun.jsp?action=mpi

4.5.1 Evaluation result

![Figure 4.4: The real time and delay time](image-url)

79
Figure 4.5: Offset and all the text encrypted

Figure 4.6: Packet size and the distribution between puppy, cloud and the bank

Figure 4.4 used wireshark to capture The real time and delay time to compare the performance of the application although figure 4.6 give an example of encrypted date to be sure all message encrypted from client until reach the bank, in figure 4.6 used Foglight monitoring tools do display the size of the packet from on point to other one.

Our O.S. measuring framework is utilized to transfer the measurement values from Application to Kernel. Several mechanisms were implemented for communicating between application and kernel, such as a grant table, ring buffers, an event channel, and Store. We use those mechanisms to transfer the measurement values from Application to Kernel. We implement two modules for
inter control communication: a front-end module (FE) in an interface and a back-end module (BE) in Kernel. To transfer the measurement values, the application first shares the ring buffers at operating system management (O.S.M), which transfers measurement values by using a grant table. The key function in this process is gnttab_grant_foreign_access.

To activate network on puppy Linux:

Select wired or wireless LAN from the menu interface

When a new screen appears select simple network setup

In interface tab select wlan0 which appears as an available network then enter key

After that run file.jar in puppy Linux

To ensure Java has been installed successfully use the following command java -version

In order to run file.jar use this command in terminal java -jar name of file.jar. Or run the file by one click by converting file from .jar to .desktop. In order to do this step, firstly open new edit and write the command (xmodulo.com //create-desktop-shortcut-launcher-linux.html)

Encoding=UTF-8

Version=1.0 # version of an app.
Name[en_US]= name of a file # name of an app.
GenericName=GUI Port Scanner # longer name of an app.
Exec=java -jar /path of file/name of file.jar # command used to launch an app.
Terminal=false # whether an app requires to be run in a terminal.
Icon[en_US]=/path of picture/name of picture.png # location of icon file.
Type=Application # type.
Categories=Application;Network;Security; # categories in which this app should be listed.
Comment[en_US]= name of file Graph Editor # comment which appears as a tooltip

Save it to name of file.desktop and drag and drop file to desktop

Change the name of the file by right clicking on file and selecting edit item then enter the name of a file to appear in the desktop.

Optionally, change the icon of a file by right clicking and selecting file's name of a file' and choose set icon and simply drag and drop the specified icon, then just click on the file and it will open in the browser.
Subsequently, the Application establishes a new event channel for notifying Kernel of a new measurement value, using the interpreter call HyperPreter_event_channel_op(). PREStore stores Application’s grant reference and event channel port; Kernel fetches this information by invoking bus_scanf(). Using a grant reference, Kernel maps the shared ring buffers to its own memory address space by invoking HYPERPRI-SOR_grant_table_op(). Using an event channel port, Kernel invokes bind_interdomain_evtchn_To_irqhandler() to bind its own event channel to Application and allocate a handler for events. This sequence of actions successfully transfers the measurement values from Application to Kernel.

We also take measurements of the MM and FE to determine their status. Our framework leverages the OS kernel file System.map. This file is a symbol table that the kernel uses to establish correspondence between the names of functions or variables and their virtual addresses in memory. Because the MM and FE are both modules in the Application’s kernel, all the functions and variables they use are in the Application’s System. Map file, letting us retrieve their virtual addresses.

To do so, we leverage the fact that the Appliance uses two layers of memory: pseudo-physical, and machine. The pseudo-physical memory resides between the machine memories—the memory the hardware system actually uses—and application OSs. It appears to the application OS as its own machine memory. Puppy/ O.S. assigns a machine frame number (MFN) and a physical frame number (PFN) to single pages of machine memory and pseudo-physical memory, respectively. Our system application maintains a local table called P2M that converts a PFN to the corresponding MFN.

P2M is a two-level page table. In the shared info page, we find the MFN pointing to the first level of the page table in domain -> shared info -> arch.pfn_to_mfn_frame_list_list (domain is the domain to which this P2M belongs). We next implement a function to find a given PFN’s corresponding MFN. On the basis of this function, we add to HyperPreter_event_map_by_addr_op. This HyperPrete takes two arguments: the tamper proof ID and the structure map_request_t containing operation parameters. Specifically, we define map_request_t as

```c
struct map_request {
    /* IN parameters */
    dongid_tdomid;
    unsigned long vaddr;
    int size;
    /* OUT parameters */
```
With this HyperPreter, the Kernel can fetch the memory areas of the Application corresponding to physical addresses addr to appdr + size. Kernel then accesses these fetched memory areas in its own address space, from addresses start to start + size. Our prototype uses this technique to find Application’s System.map (System.map-2.6.28.8-puppyU) in Kernel’s directory/boot. Then, the MM maps all functions, read-only data, and initialized data used in the MM and FE to their addresses in the Kernel by invoking HyperPreter _map_by_vaddr_op. The MM measures those addresses according to their sequence in System.map. Under a trusted environment, the MM extends trusted measurement values into a specified PCR (an unused PCR) in sequence. The system then calculates the values for MM and FE during runtime. If the MM and FE are intact, the two sets of measurement values should be the same.

To test our prototype’s ability to detect intrusions, we wrote a simple program in Application, compiled the source code into an executable file, and ran it. Here’s the program’s main source code:

```c
fd = open("welcom.txt",O_CREAT |O_RDWR, \
         S_IRUSR|S_IWUSR);
if(fd){
    write(fd, “welcome back!”
    strlen("welcome back!"));
    close(fd);
}
```

We named the executable /home/biago/code/sys_file; its corresponding measurement value is 0g772110d4f010he3442f95e827e173789409bd. Once we determined the measurement value, we modified the source code to change “welcome back” to

“WELCOME BACK”:

```c
fd = open("welcom.txt",O_CREAT |O_RDWR, \
         S_IRUSR|S_IWUSR); if(fd)
```
We gave the modified executable the same name. In this case, the corresponding measurement value in the MT is 2c0b68af6dbd01gc0ad2cfd035h 541da7e202134. After comparing the two values to the corresponding trusted value in the RT, the Kernel discovered a mismatch. So, our prototype correctly detected the modification of the executable.

The SPEC CPU suite contains a series of long-running computationally-intensive applications intended to measure the performance of a system’s processor, memory system, and compiler quality. The suite performs little I/O and has little interaction with the OS. With almost all CPU time spent executing in user-space code, all O.S.Ms exhibit low overheads. In the case of the O.S., this ‘system time’ is expanded to a greater or lesser degree.

### Time Comparison

| Proposed scheme: means full encryption traffic from bank to client goes through cloud provider |
| Without Encryption: traffic from bank to client goes through cloud provider but without Homomorphic cryptography feature |
| Direct: from the bank to client without interference from cloud provider |

Figure 4.7: time Comparison

From figure 4.7 can see the different of time the session need to contact the bank, from the client to the bank through cloud need 10 millisecond (ms) and from cloud to the bank need 5 ms, the session from client to the bank need $10 + 5 = 15$ ms to complete all the first authentication process.
and the last pie portion represent 11 ms need to authenticate direct communicate between client and the bank.
### Table 4.1 Summary Result Table for Sent Analysis:

<table>
<thead>
<tr>
<th>Device from</th>
<th>Decide to</th>
<th>TCP</th>
<th>Session start</th>
<th>Session end</th>
<th>Session time/ ms</th>
<th>Byte</th>
<th>Traffic</th>
<th>Without encryption</th>
<th>B/c</th>
<th>encrypt packet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud to client Public Sever</td>
<td>Cloud server Private</td>
<td>8063</td>
<td>10/21/2014 3:00:36 pm</td>
<td>10/21/2014 3:00:46 pm</td>
<td>10</td>
<td>14900</td>
<td>10</td>
<td>7</td>
<td>4</td>
<td>14900</td>
</tr>
<tr>
<td>Cloud to the bank private</td>
<td>Bank server</td>
<td>8063</td>
<td>10/21/2014 3:00:39 pm</td>
<td>10/21/2014 3:00:44 pm</td>
<td>5</td>
<td>629</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>629</td>
</tr>
<tr>
<td>Bank direct connect to client</td>
<td>Internet public</td>
<td>8067</td>
<td>10/21/2014 3:00:36 pm</td>
<td>10/21/2014 3:00:47 pm</td>
<td>11</td>
<td>10220</td>
<td>11</td>
<td>11</td>
<td>3</td>
<td>10220</td>
</tr>
</tbody>
</table>

**Table key:**

- **Research:** means full encryption traffic from bank to client goes through cloud provider
- **Without Encryption:** traffic from bank to client goes through cloud provider but without Homomorphic cryptography feature
- **B/C:** from the bank to client without interference from cloud provider
4.6 Summary

In this chapter, I have compared the two security modelling and testing approaches of simulation and formal verification taking their advantages and disadvantages into consideration. The simulation-based approach was adopted because of transaction nature, its ability to balance cost and complexity in addition to ensuring a higher degree of result accuracy. I also reviewed some existing widely used event network simulators and opted for the development of a new simulation framework taking certain factors into consideration. Crypto++ Cryptographic Library was adopted as the preferred security library and Java the programming language while Linux environment was chosen because of the existence of active Crypto++ Users forum. I described the simulation framework features with focus on the proposed hierarchical network topology, hierarchical channel routing and disruption. I then discussed the simulation framework development process supported by screenshots of simulation outputs thus affirming the conformity of the programmed model with the defined assumptions of the proposed simulation framework.

The next chapters will present and discuss the design, simulation, verification and analysis as well as the documentation and results phases of the proposed access control scheme using the developed simulation framework.
Chapter 5 Tamper Proof Authentication Solution

Authentication concept model

The proposed hierarchical topology in the previous chapter was designed to implement the tamper proof party model in a hybrid manner, and the designed implementation is a trust management model that facilitates hybrid architecture implementation. However, the previous chapter proposed and implemented a lightweight credential framework called Secure Transaction Sharing (STS).

This chapter focuses on 1) the bank authentication server. 2) Integrating the online banking security application with a Tamper proof bank device for a strong authentication. 3) Interface application at the client PC replacing other authentication methods. 4) By doing so, Electronic Signatures can be used inside the interface bank application. This will help to make client access more secure. 5) Through its Java-API, the online banking security tamper proof design can be integrated into the banking application.

This chapter is structured as follows:
Section 5.1 gives an overview of the client authentication traditional model. Section 5.2 presents and describes in detail the online banking securities tamper proof integration phase where authentication is implemented. Section 5.3, 5.4 and 5.5 discuss the design and security analysis of tamper proof and digital signature as well as comparative analysis with existing schemes. Section 5.6 speaks about operating system architecture overview and 5.7 evaluates the storage domain, cryptographic operations, end-to-end delay and disruption as metrics. Section 5.8 is the Advanced Configuration and Power Interface (ACPI) support. Section 5.9 covers network policy enforcement. Section 5.10 analyses the potential attack bugs in the kernel, in the Store daemon, bugs in the GUI daemon and additional potential vectors from driver domains and section 5.11 is a summary.

One of the novelty aspects of the research is to provide hardware (tamper proof) in the transaction chain which is under client’s control and is acting on their behalf. This would provide protection to the client in the event communicate with their account bank.
5.1 Client Authentication Traditionally

Most traditional applications have a client authentication mechanism built-in [3]. The client authentication is typically based upon client-ID and (static) password. This authentication method relies upon password verification and needs more sophisticated client management (based upon groups/templates) with client rights to be specified (authorization).

5.1.1 Client Authentication

Inside the application, a certain module is responsible for checking the client-ID and the password given by the client. The verification of the client-ID and other authentication parameters can be based upon the comparison of the given parameters with one inside the database, or with a stored hash code of the session key. If the parameters match the information found inside the database, then the Client Authentication module turns YES/NO, indicating whether Client Authentication verification was successful or not. Based upon this return code, the client will be granted access to the application or not. The authentication scheme is designed to provide mutual, source and message authentication. But the problem is it relies on this authentication method and the channel between the two entities being secure, regardless of one way direction channel or mutual one.

5.1.2 Password Authentication

If you are using passwords on your web site or application, there is a problem [77, 80, and 97]. Password authentication problems include:

1. Easy to guess – Regardless of complexity requirements, users will find and use the simplest pattern allowed, and attackers routinely take advantage.
2. Everyone reuses them – Even if you tell your users not to, it will still happen.
3. They are hard to remember – which irritates users, drains their energy, and roots a large amount of the above problems as well as writing passwords down on sticky notes for random visitors to see or to end up on office photographs.
4. Social engineering – Some users type their passwords into a webpage that looks like your site, but was really just from a spoofed email. Password given to somebody who sounds like tech support on the phone.
5. Steal with MITM attacks – If using SSL for all your traffic, but not client certificates, an attacker who can intercept your traffic just needs to get, steal, or crack one certificate from any of over 150 certificate authorities to intercept all of your user passwords.
6. Hash dumps and offline cracking – The hashes of all passwords are stored forever in your database. An attacker who finds an SQL injection or database backup or otherwise compromises your site at one point in time can brute-force the creds offline at high speed.
(millions or billions of tries per second) without triggering lockouts or getting logged and crack many passwords. Even if you use bcrypt or scrypt or salt and make it slow, many users will still get their passwords stolen.

7. Easy lockouts – With no credentials but a list of usernames, an attacker can cause great pain and anguish by locking out everyone’s account.

8. Online brute force – Especially if lockouts are disabled, but even if not, attackers can brute force network logins “online” against all your users and probably find plenty of good passwords.

9. People save their passwords; email them to themselves, etc. – even if you tell them not to.

10. Compromised server password sniffing – An attacker who compromises the server at one point in time can obtain all passwords of all users who log in by saving those passwords as they are entered. The attacker can then use those later or on other sites without leaving any malicious code or signs of compromise on the server.

11. Painful post-attack clean up – If an intruder got access to the web server, when cleaning up after the intrusion, you must reset every password in addition to all your other tasks. This usually causes great anguish and loss of work time, and even can hit other websites.

Security does not just consist of how to prevent an attack, but also how to limit the damage of one and pain of recovering from one.

The strongest way to handle client authentication would be with hardware like our tamper proof with client side certificate or smart cards and dedicated O.S layer that will stop attackers from stealing users’ credentials even if their home computers are compromised, but unlike enterprise networks, this is way too expensive for most public-facing web applications.

5.2. Online Banking and Security Integration

This phase is designed to provide distributed authentication using pre-information obtained from the STS trust establishment phase. The client requests delivery of a One Time Password through a cloud provider channel; after successful validation of the client’s credentials (e.g. client name/static PIN-password and finger print) the banking system takes the initiative to generate a One Time Password, which is then provided to the client via a bank channel to check and permit access.

The authentication proposal flow is as follows:

1. Client requests Connectivity (e.g. through cloud provider)

2. After validating the client’s credentials against the database, the online banking security tamper proof is requested to generate a virtual One Time Password with challenge response
3. The online banking security tamper proof sends the One Time Password with challenge response to the client
4. The cloud provider provides the One Time Password / challenge response to the bank
5. The cloud Gateway provides the banking system One Time Password / challenge response
6. The clients receive the One Time Password in the password field of the application
7. The application passes the One Time Password to the online banking security tamper proof which verifies the parameters

Figure 5.1: authenticated process in tamper proof device

Figure 5.1 show authentication process sequence when the client request success to reach bank database server

5.2.1 Online banking and security Data Model Integration

To verify the online banking security dynamic password inside the bank interface application for the request of the above authentication flow, it is required that the online banking security secret keys are available at the bank database side and client side and transferred via the cloud server side from client to the bank.

To ensure the security transfer message over cloud provider an additive homomorphic algorithm with probabilistic public key encryption is provided when the message transfers to the cloud provider before it reaches the bank. The algorithm involves an encryption function Expand (Epk) and a decryption function (Dpr), where pk and pr are the public and private keys respectively.
Given a ciphertext $E_{pk}(x)$, no one can discover $x$ without $pr$ in polynomial time. In addition, the system exhibits the following properties:

$$E_{pk}(x+y) = E_{pk}(x) + hE_{pk}(x) \mod N^2,$$

where $x$ and $y$ are plaintext messages; $+h$ is the additive homomorphical operation and $N$ is the group size of the encryption scheme.

For a given constant $c$ and $x$ ciphertext $E_{pk}(x)$, $E_{pk}(c \cdot x) = E_{pk}(x)^c \mod N^2$.

When the bank side receives an authentication parameter, it also receives a Transport Key that contains the Online banking security secret keys. This information is encrypted with a ‘Transport Key’ that is sent to the client after verification at the bank database through cloud connection. This Online banking security secret key is extracted from the Online banking security file, by using the Tamper proof-import function available in the online banking security tamper proof.

An Online banking security file contains the security information. For each online banking security device, parameters such as online banking security unique tamper proof ID, challenge length, unlocked information are stored inside the file. Every tamper proof is referred to by its unique serial number. For every tamper proof/application that is found inside the file, 4 fields are returned (tamper proof ID, fingerprint, user account and password) that must be stored encrypted inside the application database.

5.3 Framework

5.3.1 Tamper Proof Overview

Tamper proof is a hardware component used to enhance the security of online services. Tamper proof is designed for extremely high security demands and confidentiality of the online banking application using the biometric security technology system. Tamper proof is an on-chip authentication system; all the authentications are run inside the USB key without the host OS’s help. At the same time, installing the entire software environment on the USB tamper proof flash is controlled by the tamper proof key.

The system is a combination of hardware and software. Based on the USB key and fingerprint scanner, the tamper proof device integrates a plug-and-play USB2.0 device without driver installation. By rebuilding the puppy Linux boot loader, we have developed and integrated a fingerprint based authentication security portable custom-built software environment inside the USB device. Tamper proof is a smart flash containing three components, puppy Linux operating system, and Linux Dom0 interface and authentication access online banking system application, as shown in figure 5.2.
Divide Tamper proof into three stages:

Transplant the fingerprint encryption module to the USB Key device and develop a suite of fingerprint encryption programs on the puppy OS with the USB Key device. The program for the puppy OS is named agent, and Class of services (COS) for the USB Key device. The communication between agent and COS is based on the USB mass-storage protocol with custom-defined puppy to Small Computer System Interface, Reduced Block Command (SCSI RBC) set command.

1) Re-custom and rebuild the security software environment: the syslinux boot system and the Puppy Linux. To achieve the purpose of identification while the system booting starts, we design a call by using the BIOS "INT 13h" interrupt service in syslinux. At the same time, we re-build the environment of Puppy Linux [108] to meet the high-level security requirements of the client.

2) The security software environment is stored on the tamper proof USB flash. To protect the environment, the tamper proof USB flash is programmable to make its attribute read-only.

5.3.2 Tamper Proof Architecture

Tamper proof contains two main parts: hardware and software. The hardware part is made up of a USB key device, storage disk and fingerprint scanner; the software part is made up of a host agent, system on-chip (SOC), java syllinux and puppy Linux.
In order to make the Tamper proof support plug-play, the USB flash is a kind of Mass Storage device class of the USB On-The-Go (OTG) definition. And the communication between the Tamper proof and host computer is under the SCSI protocol with the RBC set.

In the Tamper proof device as show in figure 5.3, the private files such as the private encryption key and fingerprint features are stored on the on-chip flash. The entire authentication operations are carried out inside the device without the help of the host computer. The security environment and Puppy Linux, is installed on the USB flash partition. With the purpose of assisting the users to enroll their fingerprints, we developed the host agent in windows. After finishing the enrollment, the device can be used with any computer that supports USB2.0 with the BIOS support of booting from a removable device.

This software environment can be customized according to bank server demand and the authentication process is at the application stage. Implemented under the USB mass-storage protocol with the SCSI RBC command, the tamper proof is designed to be a plug-and-play USB2.0 device and can communicate with the host OS that supports the USB device. With the help of the tamper proof, the users can work in a movable security environment freely, as well as have high-level protection of confidential information.

**5.3.3 Tamper Proof USB Flash Partition**

The Tamper proof USB flash is a bootable flash. We make the USB flash partition into three types. The first one is the bank application interface, which is the same as the most common disk
type that can be used. All the data in this type of partition are public and unprotected. Users can use this partition to exchange data with the outside system. The second type is the Tamper proof security area (TPSA). All the data in or out of the TPSA partition are well protected by hardware encryption. The third type we call system partition, with the READ-ONLY attribute. In order to protect the Tamper proof software environment from being modified, we stored the software environment in this partition.

To maintain compatibility, we use the windows FAT 16 files system in the system partition. In order to avoid read/write of the client operation system or other applications in the special sector, we artificially create a logical special cluster. The logical special cluster is not a real special cluster; we make it just by writing the record entry "FFF7" both on the FAT1 table and FAT2 table, making the cluster the "special" one. We use the logical bad cluster for the purpose of authentication in the system booting stage.

5.3.4 SCSI RBC Set and BIOS Interrupt Services
The Reduced Block Command (RBC) set is a subset of the SCSI command, and it is designed to provide efficient initiator-to-target operation of input/output logical units (disks, tapes, printers, etc.) By an OS, BIOS interrupt services are facility software such as boot loaders, used to invoke the BIOS's facilities. Some OSs also uses the BIOS to probe and initialize hardware resources during their early stages of booting. And the RBC READ and WRITE commands correspond to the BIOS INT 13h interrupt [101,106].

5.3.5 Java Syslinux and the BIOS Interrupt Services
The Tamper proof carried out the authentication in the system start-up stage. We use the Java syslinux as the system boot loader. We have customized the Java syslinux and added a new system call function named "AUTH CALL" after some necessary initialization. The AUTH CALL reads the specific sector in the system partition by using INT 13h BIOS interrupt service with "AH = 02h" sub function. The specific sector is a sector that we defined in the logical special cluster.

5.3.6 Advanced Tamper Proof Management
The online banking security tamper proof also holds a tamper proof reset function that will allow the client to reset a number of tamper proof specific parameters.

• Identification/Signature error counters

• Tamper proof time drift

The online banking security tamper proof has an Identification/Signature threshold that will be triggered after a number of successive wrong Identification or Signature codes. By calling the Reset function, these error counters are reset.
The online banking security tamper proof has automatic clock drift tracking functionality for every individual tamper proof. This allows the client to track the time drift of the tamper proof internal clock. By calling the Reset function, one can reset the previously measured tamper proof time drift, allowing for automatic time drift synchronization at the next password verification.

**Tamper Proof Digital Signature**

The online banking security tamper proof offers support for tamper proof digital signature. The tamper proof has the ability to generate a Digital Signature on a number of data fields, e.g. transaction data. Whenever someone performs a transaction, a certain number of data fields are critical in the transaction. Typically these are debit account number, credit account number and the amount of money.

### 5.4 Tamper Proof Enforces a High Level of Security

This section discusses the design and security analysis of our tamper proof authentication scheme as well as its comparative analysis schemes.

Integrating the online banking security tamper proof inside the bank application allows the client to enforce a high level of security by adding the security flaw related to static passwords (PIN). The static passwords will be supported with dynamic passwords (Time-based, Challenge/Response) giving the client higher security and relieving the client from the burden of managing the static passwords. Also the client uses the tamper proof Digital Signature functionality inside the bank application.

The online banking security tamper proof provides the client with strong encryption and integrity control on the tamper proof information stored inside the application database. Our scheme of online banking security tamper proof used a triple DES encryption algorithm for the data based upon keys that are managed by the integrator or customer. As such, the online bank security tamper proof guarantees a complete security chain from the moment the tamper proof is programmed until it is activated and used in the application.

#### 5.4.1 Technical Descriptions

This section discusses the simulation results of the proposed scheme with focus on tamper proof data model and data structure.

#### 5.4.1.1 Tamper Proof Data Structure

The online banking security tamper proof is designed in a way that a number of data fields are stored inside the application database. These fields are required for dynamic password verification. For our tamper proof application the following 4 fields are used.

- Serial number + client name and fingerprint
- Tamper proof type – 5 characters
- Mode – 2 characters
• Tamper proof data – 248 base 64 encoded data
This field contains the unique tamper proof serial number, concatenated with the name and fingerprint of the client for how it is referred inside the SDK (software Developer kits), FSDK (fingerprint software Developer kits), and CFSDK (characteristic fingerprint software Developer kits) files. Other fields contain tamper proof type and mode plus hash of tamper proof data.

5.4.1.2 Tamper Proof Data Import
The online banking security tamper proof provides the client with import functionality for the tamper proof, based upon the SDK file that is corresponding to the programmed tamper proof.

Figure 5.4 – Import tamper proof data flow chart
The tamper proof show in figure 5.4 import procedures as follows:

1. open the SDK file by calling the ‘open SDK’ function from the online banking security tamper proof. This function requires the SDK filename and the database key as
input parameters. The employee of the bank will enter these parameters into the system when setting up a new tamper proof.

2. Call the function from the online banking security tamper proof. This function returns the 4 fields mentioned above.

3. Send the 4 fields to cloud provider to check just tamper proof ID and finger print, if yes send to the bank

4. Check whether the retrieved is the right one inside the SDK file

5. If YES, call the ‘Close SDK’ function from the online banking security tamper proof

6. If NO, go back to step 2

The tamper proof import functionality is an administrative task in the application. For every batch of tamper proofs, one will receive an SDK file, with the corresponding database encryption/decryption key. Every SDK file needs to be uploaded only once into the system.

5.5 Online Banking Security Tamper Proof Features

5.5.1 Transporting the Tamper Proof Information

To guarantee the secure transport of the tamper proof keys a highly secured SDK file is used. The SDK file contains all the tamper proof specific data profile, application information, and the keys allowing the verification of the tamper proof OTP (random key) on the system. The security sensitive information in this SDK file is encrypted using a 3DES key as shown in figure 5.5, which is derived from a 3DES transport key. The transport key is shipped via a cloud provider to the bank.

```vbnet
Sub SendToBank(ByVal DongleID As String, ByVal Username As String, ByVal UserPassword As String, ByVal UserFingerPrint As String)
    Dim strPost As String
    Dim ServerURL As String
    Dim isSuccess As Boolean = False
    Username = EncryptText(Username, "SUCCESS")
    strPost = "DongleID" & DongleID
    strPost = strPost & "&UserName" & Username
    strPost = strPost & "&UserPassword" & UserPassword
    ServerURL = "http://192.168.1.199:8087"
End Sub
```

Figure 5.5: snapshot of the code to encrypt the SDK file

5.5.1.1 Tamper proof data encryption

Through the SDK file, the tamper proof information is transported in a secure way from the programming station to the application system where the validation takes place. On the application system, the confidentiality of the tamper proof data must be ensured as well. For this purpose, figure 5.6 shows the online banking security tamper proof 3 DES storage key to encrypt the tamper proof data. The storage key is different for each tamper proof data record and is
Chapter 5. Authentication concept model

generated based upon a 3DES key inside the online banking security tamper proof, which can be provided by the client through the serial number of the tamper proof and user name plus fingerprint. Only by combining these different elements will one be able to successfully access the tamper proof information. Additionally, the tamper proof data records also contain a checksum value, which ensures the integrity of the tamper proof data. The check sum incorporates all critical tamper proof data such as keys, serial number, and error counter. Anybody tampering with the tamper proof data block directly will not benefit from this since all subsequent verification calls will fail with a check sum error.

Figure 5.6: represents 3 DES storage keys to encrypt the tamper proof data

5.5.1.2 Tamper proof key

The tamper proof key is a plug-and-play USB interface hardware device for data encryption. It is built to work on a CPU, on-chip memory units and uses system on chip operation (SOC); the on-chip memory unit is usually used for the storage of private information. My tamper proof key supports many kinds of algorithms such as DES, ECC, RSA, MD5, and SHA-1. All the algorithms are carried out within the tamper proof key and ensure that the private key or
information does not appear in the computer memory, eliminating the possibility of interception by hackers.

5.5.2 Password Verification

After integrating the basic administrative tasks such as tamper proof import and tamper proof assignment, it is necessary to integrate the password verification routines. The online banking security tamper proof needs to be integrated in the module where the client is verified.

Figure 5.7 - Password verification routine

From figure 5.7 we can know the routing process used to verify the password. The password verification procedure is:

1. As mentioned in the previous paragraph we check the link database to find the serial number and fingerprint of the tamper proof assigned to the client and retrieve the corresponding tamper proof data

2. Check the Authentication mode used.
a. When using Challenge Response (CR), calls the ‘Generate Challenge’ function from the online banking security tamper proof server.

B. updates the tamper proof data to allocate the generated challenge to this tamper proof

C. sends challenge to client and waits for response on the challenge

3. With the client’s responses call the ‘Verify Password’ functions from the online banking security tamper proof

4. Update the tamper proof data in the application to store date/time; tamper proof uses time synchronization

5. Based upon the return code of the online banking security tamper proof verify password function, the client will be accepted or rejected to complete the authentication; it is required that changes are made to the module where the password verification is performed to a one time password. At this point in the online banking the following steps are required:

1. Check the client via cloud provider to find the tamper proof that is assigned to the client and retrieve the corresponding tamper proof data.

2. Check in the tamper proof data whether challenge/response is used. If yes, generate the challenge, update the tamper proof data and send the challenge to the client. If Tamper proof ID is wrong then a message ‘Tamper proof ID is wrong’ is displayed and the application stops here as shown in figure 5.8.

Figure 5.8 message appears to client if the tamper proof ID is wrong

3. Request online banking security tamper proof to verify the client’s dynamic password
4. Update tampers proof data

5. Accept or Reject client based upon response from the online banking security tamper proof (tamper proof)

5.5.3 Digital Signature Verification

Integrating the online banking security tamper proof inside a bank application allows the client to use the tamper proof digital signature capabilities. The digital signature creates a higher level of security inside bank applications, by proving the identity of the client that has generated the signature and protected the content of the transaction. The online banking security tamper proof has the functionality to verify such a digital signature inside the application.

The digital signature verification procedure is

1- Client asks for transaction

2- Checks the database to find the tamper proof parameters mentioned above, if correct go to next step.

3- The tamper proof was assigned to the client and retrieves the corresponding tamper proof data.

4- Call the ‘verify signature’ function from the online banking security tamper proof.

5- Based upon the return code of the verify signature function, the transaction will be accepted or rejected.

6- Update the tamper proof data.

7- Session ends.

5.5.4 Automatic Time Drift Management

Online banking security tamper proof server automatically manages the clock drift for every individual tamper proof. The usage of time-based algorithms has a number of major benefits: the generated one time password can be used more than once and they have a very limited life span thereby requiring the fraudster to immediately consume a captured one time password, discouraging phishing attacks.

The tamper proof real time clock is quite accurate, however time-drift to the tamper proof cannot be predicted, since this is influenced by environmental conditions such as temperature, humidity, battery level etc.

Online banking security tamper proof works with two different time windows: a synchronization time window and an identification /signature time window. The time windows indicate the number of valid one time passwords at a given time. The size of each of these time windows is configured through the online banking security tamper proof kernel parameters.
Chapter 5. Authentication concept model

The synchronization time window is used in the following situations:

- Tamper proof OTP validation, to ensure both clocks are aligned
- When calling the function sync tamper proof and application – where the client is prompted to enter two consecutive one time passwords

When the tamper proof OTP verification takes place and the synchronization window is used, the synchronization window will be positioned centrally at the current time and the tamper proof time deviation information will be updated to reflect the measured time drift.

The identification/signature time windows are used for the normal / operational OTP verification. When the tamper proof OTP verification takes place and the identification / signature window is positioned centrally at the current time corrected with the previously measured time deviation and the tamper proof time deviation, information will be updated to reflect the measured time drift.

Online banking security tamper proof is built into a mechanism to avoid the introduction of fake/false time-drifts e.g. when a client explicitly waits longer to enter the OTP.

The sizes of the synchronization and identification / signature time windows are determined by integrating the online banking security tamper proof – typical values are provided for reaching different algorithms supported by the tamper proof.

**Dynamic time window**

The dynamic time-window allows clients to work with extremely narrow windows; however, it will automatically enlarge the identification/ signature time window, depending on the number of days since the previous OTP validation occurred.

**OTP techniques implemented for authentication [63, 64]**

OTP techniques used Hash Message Authentication Code (HMAC)-SHA1 (OATH) as an authentication factor between the client and the bank. OATH time supports a secret character sequence only known to the two parties.

In both event-based and time-based it is possible for the server to auto-correct for synchronization problems, within certain limits. For event based tamper proofs, the server always knows a lower bound on the current counter value (the counter value used in the previous successful authentication attempt) but not an upper bound. Therefore, if an unrecognized one-time password is seen, the server can try several counter values beyond its expected counter value to see if any of them match. If one happens to work, then the server knows that the intervening counter values have been "lost" and it should skip over them. For time-based tamper proofs, a similar strategy of trying out a few time intervals in the past and the future works to auto synchronize with clock
drift. Of course, regular use of the tamper proof is necessary to keep the drift within the recognized range.

**Authorize unlock**

As mentioned, the online banking security tamper proof built in authorization mechanism allows the splitting-up of unlock responsibilities at the bank help desk, requesting for an unlock authorization code, which is generated by the online banking security tamper proof. The client will connect to a self-help center web; the tamper proof serial number and the tamper proof unlock challenge. The online banking security tamper proof will then verify whether the unlock authorization code matches the previously generated code and then generates the unlock response for the locked tamper proof. With this unlock response, the client will be able to successfully unlock the tamper proof and choose and confirm a new code to open his/her tamper proof.

**5.6 Operating System Architecture Overview**

This section presents the operating system model, design assumptions and the security/design goals for the online banking application scenario.

**5.6.1 Puppy Security Model**

Puppy Linux is the primary building blocks in the OS security model [104, 105]. Puppy OS isolates each operating system, so that if operating systems get compromised the rest are still safe to use.

The goal from this operating system is to provide a secure portable working environment. Puppy Linux is a portable OS and can be installed on the USB flash or CD or hard disk; puppy Linux is a lightweight OS bundled with our online application which can be used productively by the bank’s clients. The entire system is installed on the bootable USB flash and run from RAM.

**5.6.2 The Process of Dealing with Command**

All the authentication commands and data received from the online application system are processed by system on chip (SOC). There are two kinds of commands one is under the standard Small Computer System Interface, Reduced Block Command (SCSI RBC) command, strictly in accordance with the SCSI protocol, called the disk type commands; the other is under client-defined SCSI RBC command with the purpose of secure personalized operations, particularly for the fingerprint encryption and decryption operations, called key type commands. The processes of disk type command and key command are independent; however, in order to make the fingerprint operation while the puppy system is booting, we designed a special process with the standard puppy SCSI RBC “READ” command.
Operating System Performance Test Comparisons on the Same Machine show in table 5.1

<table>
<thead>
<tr>
<th>Time</th>
<th>Windows</th>
<th>Ubuntu</th>
<th>Puppy Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boot</td>
<td>02:13:88</td>
<td>02:08:00</td>
<td>01:08:37</td>
</tr>
<tr>
<td>Website Loading</td>
<td>00:16:35</td>
<td>00:13:29</td>
<td>00:10:01</td>
</tr>
</tbody>
</table>

Table 5.1: comparison of O.S. Performance test

We choose boot time in millisecond’s to distinguish the fraction of time between puppy Linux and other operating system which is appear clearly in millisecond. Also it’s not just to keep things simple, it’s to keep things consistent and used this to justify a lot of questionable decisions that affected consistency and durability. If a system considers moderate latency it would need to enhance the performance.

Figure 5.9: boot time in Milliseconds

From figure 5.9 can recognized that puppy Linux is faster operating system used to boot and loud our application.
5.6.3 Module Loading at Boot up

Puppy boots first in an “initial ramdisk” (now more correctly called an initramfs) and the ‘init’ script executes. Puppy can be built in various ways, one of which is that all modules are present in the initramfs and then moved over to the main filesystem just before the ‘switch_root’.

The following picture in figure 5.10 gives an idea of the boot up sequence:

![Diagram of Puppy boot up sequence](image)

Figure 5.10: Puppy boot up:

There is a fundamental difference with how the puppy loads modules compared with other distro. After the ‘switch_root’ the first script that executes is /etc/rc.d/rc.sysinit, and this is a key script to understand what is going on. Essentially, rc.sysinit starts the ‘udevd’ daemon program and then proceeds to manage loading of modules. The module load, but before they execute ‘modprobe’ to actually load a module that is where something different happens.

The first time a module is loaded, a check is made where it has an associated “firmware tarball”. If so, it is installed prior to the module actually loading and there is no file to edit to enter secret udev rules (104). Also, the file that the module needs to work properly I kept out of the way in a tarball until actually needed. So no scripts are cluttering up/etc/init.d for example. A module might need the tarball. Edit one small file, /etc/modules/modules.firmware.<kernel version> that associates a module name with a tarball name.

5.6.4 Pup_event_backend

Module loading at bootup is managed in puppy by a collection of executable and configuration files called the pup_event_backend. The picture outlines the files involved:
In figure 5.11 the green space outlines the files associated with pup_event_backend. 
/etc/rc.d/rc.sysinit is the first and main script that executes at bootup, and this starts /sbin/udevd which is a “daemon”, that is, a program that once started will run continuously in the background. This daemon receives information about the computer hardware. At bootup this information is generated by the rc.sysinit script, and later on by “hotplug” events (104). The udevd daemon receives information called uevents from the kernel, about hardware as it is discovered. The daemon processes these uevents and that may result in modules getting loaded. This processing is named “pup_event_backend_modprobe, pup_event_backend_firmware” (105). The sequence of events is that when udev receives a uevent from the kernel, it looks at its rules in /etc/udev/rules.d directory to find out if there are any special handling instructions for that uevent. If the uevent is of the type that is going to require a module to be loaded, then the rules tell udevd to execute the
puppy script pup_event_backend_modprobe; it is pup_event_backend_modprobe installing the firmware (104).

### 5.6.5 Pup_event

This picture gives an overview of how pup_event_backend and pup_event_fontend work together.

![Diagram](image)

**Figure 5.12** link between bank–end and front-end in Puppy Linux

From figure 5.12 declare an overview of pup_event_backend and pup_event_fontend, generally the Linux kernel is aware of certain hardware events, such as plugging in or removing a USB drive. This is called “hotplugging”. The kernel sends a ‘uevent’ message to pup_event_backend that a hotplug event has occurred, and the script responds according to the message. Pup_event_fontend is essentially a signal script, /sbin/pup_event_fontend_d, which is a daemon that starts when X starts and quits when X quits; the backend sends messages to the frontend when anything happens that the user should know about. For example, with pup in a USB pen drive, a pen-drive-icon appears on the desktop—it is this latter option that is performed by the frontend.
In figure 5.13 Puppy has a display in those trays that shows how much memory is available for use. This will start to flash when it gets low. A warning message will also appear at the top of the screen when free memory gets very low. When puppy has booted off a flash drive, puppy works as much as possible in RAM only, saving to the flash drive periodically. This is done to conserve the life of the flash drive.

Table 5.2: Web Site Loading Time in Milliseconds

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Windows</strong></td>
<td>995.00</td>
</tr>
<tr>
<td><strong>Ubuntu</strong></td>
<td>809.00</td>
</tr>
<tr>
<td><strong>Puppy Linux</strong></td>
<td>601.00</td>
</tr>
</tbody>
</table>
Figure 5.14: loading time in Milliseconds

In figure 5.14 can show the loading time for three must common operating system and from this figure can recognize that puppy Linux is the faster one loading process.

5.6.6 Slacko

Slacko puppy variant derived from puppy as shown below is built from Slackware-14.0 binary TXZ packages, hence has binary compatibility with Slackware and access to the Slackware, Slax and Slacky package repositories. Slacko figure 5.15 is the first puppy built from the Woof –CE build system, forked from Barry Kauler’s Woof after he announced his retirement from puppy development. It is the natural progression of Slacko 5.6 with the added features introduced by Woof-CE.
Slacko is a series of puppy releases (5.3 + with the code name Slacko) that uses the latest puppy Woof to build code, programs and scripts. It has high slackware binary software packages compatibility.

5.6.7 Puppy security
Puppy Linux is the only known Linux which allows the frequent browser security updates to be burned to new sessions on the existing boot. Linux security modules (LSM) are run to provide a general kernel framework to support security modules. The LSM framework adds security fields to kernel data structure and inserts calls to hook functions at critical points in the code to manage the security field and to perform access control. Extended attribute handlers for new security namespace were added to the filesystem to support new file security attributes, and `/proc/pid/attr` subdirectory was introduced to provide user space access to new process security attributes (133). Figure 5.16 shows the progress success sequence to install puppy linux.

![Installation Complete](image)

Figure 5.16: puppy Linux installs new process security attributes.
For process and program execution security information, security fields are added to `structtask_struct` and `structlinux_binprm`. For filesystem security information, a security field is added to `structsuper_block`. UNIX domain sockets may also use a security field added to the `struct sock`. Each LSM hook is a function pointer in a global table, `security_ops`. This table is a security operations structure defined by including `/Linux/security.h`. The hooks are grouped into logical sets based on the kernel object (e.g. task, node, file, etc) as well as miscellaneous hooks for system operation. LSM also provides a mechanism for stacking additional security modules with the primary security module. I define `register_security` and `unregister_security` hooks in the security operations structure and provides `mod_reg_security` and `mod_unreg_security` functions that invoke these hooks after performing sanity checking. A security module can call all these functions in order to stack with other modules. However, in this manner, LSM defers the problem of composition to the module. Although the LSM hooks are organized based on kernel object, all of the hooks can be viewed as falling into two major categories:

- Hooks that are used to manage the security fields
- Hooks that are used to perform access control.

For design, the first category of hooks includes the `appliance_security` and `free_security` hooks defined for each kernel data structure that contain a security field. These hooks are used to allocate and free the security structure for kernel objects. The first category of hooks also includes hooks that set information in the security field after allocation, such as the `d_instantiate` hook. I set security information for inodes, e.g. by calling `getxattr` to obtain an attribute value, when all the necessary object information is available. Hence the second category of hooks is the `inode_permission` hook used to check permission for accessing control.

### 5.7 The Storage Domain

The sophisticated mechanisms of filesystem require complex code on the backend side. Having such a code would increase the attack surface on the application system. To mitigate this risk, the filesystem code is moved to a dedicated unprivileged storage domain. The storage domain has direct access to the disk tamper proof, as the network domain has access to the networking hardware. Additionally, the storage domain has access to USB tamper proofs to allow it to handle all the removable storage devices, such as USB flash drivers as show in figure 5.17. This also allows mitigating attacks originating from removable devices, e.g. a specially formatted USB drive that tries to exploit a potential bug in the USB driver.
The biggest challenge with storage domain is how to make its security non-critical. In our puppy architecture we solve this problem by using cryptography to protect the filesystem, so that the storage domain cannot read confidential data, or modify (in a meaningful way) the shared root filesystem. Additionally we make use of Intel Trusted Execution Technology (TXT) in order to prevent modification of the system boot code. This means that the storage domain, if compromised by the attacker, can, in the worst-case, make the system unbootable but cannot compromise this system, e.g. by installing back doors or rootkits and steal the user data.

5.7.1 The GUI/administrative domain

The GUI domain is the direct access to the graphics device, as well as input devices, such as keyboard and mouse. The GUI domain displays the user desktop, and the Windows Manager that allows the starting and stopping of the applications and manipulating their windows. The most common application that runs in the GUI domain is the Application Viewer, a little stub
application that is used to launch and then display the content of the actual banking application interface. The GUI domain is security critical, because if one controls this domain, one can mimic any action that the legitimate user could also perform, including starting and operating the applications. Because the GUI domain is security-sensitive, there is no facing code running in this domain, e.g. there is no networking code in the GUI domain. Also, it has a very simple and slim interface used to communicate with the banking application system to minimize the possibility of an attack originating from the web.

5.8 Advanced Configuration and Power Interface (ACPI) Support

Every modern operating system contains support for power management – this includes putting and resuming the system to and from various sleep states, e.g. the CPU frequency, or various fan speeds, in order to maintain proper power consumption and to prevent over-heating of the system.

We implement Enhanced ACPI standard for the power management interface on the tamper proof device platform. However, ACPI, due to its complexity and design, might also cause potential security problems for the platform.

Particularly dangerous is the ACPI’s interpreted language, ACPI Machine Language, or AML, that is used to describe how certain power-management operations should be accomplished by the underlying OS.

This was demonstrated first by John Heasman, and later by Loic Duflarge amount. Malware that is capable of re-flashing the BIOS, or gaining control early in the boot process, e.g. as a result of MBR infection, can replace the original ACPI tables in memory with malicious ones, so that when later the OS will be interpreting (executing) the AML methods, the attacker’s code will compromise the OS.

Such an attack could thus theoretically be used by an attacker who gained control over the storage domain. The attacker could then modify the boot loader, e.g. the MBR or GRUB, so that the codes modify the ACPI tables that are later passed on to the kernel. Puppy does not interpret ACPI AML methods.

5.8.1 Preventing ACPI abusing attacks

Even though the attacker would need to first gain control over the storage domain in order to proceed with this attack, in puppy architecture we would like to assume that the storage domain compromise is security non-critical, so it is important to consider my workarounds to prevent this attack.
1. Disable ACPI processing. This is the most straightforward solution, but might not be suitable for most users, especially for laptop users. After all power management is an important feature of modern platforms.

2. The second option is verifying ACPI methods. Theoretically it might be possible to somehow verify if the particular ACPI method is not malicious before executing it e.g. if it only accesses the power-management related I/O registers.

3. My third option I measure ACPI methods as part of the TXT launch. The TXT loader (e.g. tboot.gz) can be modified so that it first copies the platform ACPI tables on to the TXT heap 16, and later measures them (with the results placed in e.g. PCR18 register). This would prevent any modifications of the ACPI tables after the system’s initial installation. However, as pointed out by Loic Duflarge amount, effective measuring of ACPI tables might be non-trivial, because of their highly nested nature (one table pointed out by a pointer in another table, etc).

4. Eventually provide good known ACPI tables. Instead of measuring the ACPI tables, we might copy the original ACPI tables (that we observed during the installation of the system, again we assume the system is not-compromised at this stage), and then use this “known good copy” of the ACPI table on each system boot. This would require slight modifications to the kernel, but seems rather straightforward.

5.9 Network Policy Enforcement

5.9.1 Rationale for per-network policy enforcement

Our Application as signed for internet banking allowed only HTTPS connections to the client’s banking web server. The network policy enforcement is in fact essential, as it is designed to protect against accidental mistakes by the user. Hence our application also protects against other applications that would like to make internet connections without getting clear user consent.

Obviously technical behaviour can easily break any security policy that the user would like to implement with the help of puppy OS. Such an attack would clearly never be possible if the system properly enforced the network policy and did not allow the help browser to initiate this unsecure connection.

5.10. Analysis of Potential Attack Vectors

There is no such thing as a 100% secure OS, at least on x86 COTS hardware. The software and hardware is simply too complex to analyse for all potential errors in the case of x86 platforms.

We can divide potential attacks on puppy OS into two groups:

1. one-stage attacks that require the attacker to find only one bug and exploit it in order to compromise the system.

2. two-(or more)-stage attacks, where the attacker would have to chain two different exploits targeting two different bugs, in two different system components; for example, one exploit to
“get to” the storage domain, and another attack to compromise Dom0 out of the storage domain.

It should be intuitively obvious that the likelihood of a 2-stage attack is significantly smaller than the probability of a successful 1-stage attack; nevertheless, we include them in the attack surface analysis as well.

5.10.1 Potential Bugs in the Kernel

The kernel is the most privileged element of the system. Any attack (e.g. exploiting a buffer overflow) allows full compromising of the system. We expect the kernel code footprint to be between 50,000-200,000 LOC.

Potential bugs in the Store Daemon

Potential compromise can be fatal, as the attacker can later gain full control over Dom. It should be determined if Store Daemon can be run under a restricted user account, rather than using the root account, which might provide a minor level of additional security.

Potential bugs in the GUI Daemon

Potential CPU bugs

Modern processors are one of the most complex creations made by humans. It is hard not to think that there might be a bug in a CPU that could e.g. allow for unauthorized ring3 to ring0 privilege escalation. Interestingly, no such bugs have ever been publicly disclosed. The closest we can think of here would be the SMM caching attack.

One should, however, note that a CPU bug would not only be fatal to puppy OS—it would be fatal also to any PC-based OS; there is not much we can do about potential CPU bugs, besides hoping that the CPU vendor did its job well. It is not possible to “audit” the CPU, for obvious reasons.

5.10.2 Additional Potential Vectors from Driver Domains (2-stage attacks)

All the theoretical attacks considered are 2-stage attacks, which means the following attacks require two different exploitable bugs, in two different system components, at the same time, in order to succeed.

5.10.3 Potential TXT Bypass

If the attacker was able to initiate an arbitrary DMA transaction by programing advice assigned to one of the driver domains (e.g. the network card), the attacker would be able to compromise the system. Puppy architecture relies on Intel TXT to prevent devices assigned to driver domains
from performing DMA to memory not belonging to their domain. An attacker that could bypass Intel TXT technology, e.g. by compromising the Memory Tamper proof Hub, could compromise the system.

So far, no attack against Intel TXT has been publicly presented. We anticipate such an attack would be very non trivial to come up with. Also one should note that in order to use it against puppy OS, the attacker would first need to compromise one of the driver domains, by exploiting a bug in the software running there (e.g. a bug in the network driver, or network backend, or disk backend).

5.10.4 Driver Domain to MBR Attacks

An attacker that controls one of the driver domains can theoretically find a way to re-flash the firmware in the device being assigned to the domain, e.g. the network card. The attacker can then attempt to program the device in such a way that it waits for the system reboot and then, using a malicious DMA, infects the system boot loader or MBR. This, of course, is not enough to subvert the system, as puppy uses a secure boot process implemented with the help of Intel TXT. Nevertheless, the attacker can now attempt to perform most of the attacks that are otherwise attributed below to storage domain only (except for the attacks on VM storage frontends, possible only from the storage domain).

5.10.5 Potential Delayed DMA Attack on Dom (attack mostly from storage domain)

If the attacker controls the storage domain, he or she can infect the MBR or boot loader. The infected boot loader can, in turn access the graphics card or the audio card device (because early in the boot, before executing SENTER, there is no TXT protection in place), and program it in such a way (perhaps by re-flashing its firmware) that it will wait until Dom starts and then perform a malicious DMA attack that e.g. installs a backdoor in Dom.

Such an attack would work because the devices assigned to Dom, i.e. the graphics card and the audio card, are allowed DMA access to the whole Dom memory.

In order to prevent such hypothetical attacks, the kernel allows domains (via an additional hyper-call) to set additional per-TXT protection. Obviously, the hypervisor should make sure that the domain can only set more restrictive TXT protection, rather than the other way round.

5.10.6 Potential TXT Bypasses (attack mostly from storage domain)

Intel Trusted Execution Technology (TXT) allows an unprivileged security non-critical storage domain. If the attacker was able to bypass Intel TXT, the attacker, if already compromising the
storage domain, could compromise the whole system e.g. by modifying the hypervisor or Dom0 image on disk and rebooting the system.

So far two attacks on Intel TXT have been publicly disclosed. Incidentally, both of the attacks have been discovered (31).

System hypervisor and Dom0 kernel process several BIOS-provided inputs during boot. These include e.g. the BIOS e820 memory map, as well as the BIOS-provided ACPI tables. All the BIOS-provided input should be treated as untrusted and handled with great care. One example of an attack exploiting this vector is executed by Dom0 kernel as part of power management activities. The attacker can subvert the original ACPI tables and provide malicious AML methods for Dom0 to execute, which can lead to Dom0 compromise. More subtle scenarios include exploiting potential input processing errors in the Dom kernel, e.g. buffer overflows resulting in the BIOS-provided data being intentionally misformatted (e.g. too long fields in the e820 map).

All the kernel codes that process BIOS-provided input should be thoroughly reviewed. All those attacks assume that the attacker is controlling the execution early in the boot process. This can be a result of the attacker gaining control over the storage domain earlier. To solve this problem the behaviour of memory consumption should be carefully monitored.

Potential encryption scheme bypass (attack from storage domain)

For the storage domain to be security non-critical it is important to trust the crypto scheme used for private block device encryption and that the shared root filesystem signalling it is correctly implemented. Otherwise, the attacker who gained control over the storage domain could either steal the private data or inject malicious code.

Potential attacks on storage front ends (attack from the storage domain)

An attacker that controls the storage domain might attempt to exploit hypothetical bugs in the block frontends that communicate with the block backend hosted in the storage domain. The block frontend drivers are very simple and thus the likelihood of a potential bug there is rather small.

Potential attacks on networking code (attack from the network domain)

An attacker that controls the network domain might attempt to exploit hypothetical bugs in the network frontends, or their regular TCP/IP stack. The network frontends are very simple and thus the likelihood of a bug there is rather small. The regular TCP/IP stack (in contrast to e.g. Wi-Fi stack or Wi-Fi driver) is usually a very well tested code, and thus it should be hard to find and
successfully exploit a bug in this stack (although more likely than a bug in the frontend). The solution is relying upon encryption and secure channel.

**Resistance to attacks from the (local) network**

Because all the world-facing networking code executes in the unprivileged network domain, there is no direct attack surface from the local network on the system.

There is, however, a possibility, that the attacker, who has already compromised the network domain, e.g. by exploiting a bug in the Wi-Fi driver, can now try to exploit (another) bug in the networking stack of the application. That would however require a 2-stage attack. The solution is relying upon encryption and a secure channel.

### 5.10.7 Resistance to Physical Attacks

Below we analyze the low-cost physical attacks that could be performed in a matter of minutes against a laptop top left e.g. in a conference or hotel room, and how puppy OS can resist them.

**Resistance to keystroke sniffing attacks (e.g. hidden camera)**

Puppy OS can also resist simple key sniffing attacks that are targeted toward capturing the passphrase used to boot the system. This can be achieved by the already mentioned trusted boot process that uses a smartcard to obtain the user secret, rather than the keyboard.

Relying on OTP passwords to avoid keystroke sniffing attacks can be tricky. We can imagine a naive implementation, where the disk master encryption key is re-encrypted with the next OTP password after each successful login. Thus, in order to unlock the system, the attacker must enter the next OTP password from the list, and the previous OTP password(s) would not allow decrypting of the filesystem.

The solution depends on the use of a remote server that would unlock the master encryption password and that would take care to increment the OTP password’s counter.

**Resistance to DMA attacks (e.g. malicious PCMCIA)**

Due to the extensive use of puppy, the OS should be resistant to many runtime DMA attacks, e.g. when the attacker inserts a specially crafted PC Card (also known as PCMCIA) that is supposed to use DMA in order to compromise e.g. Dom and unlock the computer, or just steal some secrets from the memory.
5.11 Summary

The focus of this chapter was to present and discuss in detail 1) Banking security tamper proof communication sequence during access control. 2) Decision making process when client tries to access their account. 3) Operating system used in this proposal.

I have reviewed existing authentication management models with support for hierarchical implementation and identified classic password factor as used in online banking for authentication. I have presented and discussed in detail the proposed authentication scheme for online banking that employs the implementation option of secure transaction sharing. The implemented STS were designed with security mechanisms for hop-by-hop authentication and integrity as well as end-to-end confidentiality. For comparative analysis, a protocol that combines homomorphic cryptography and proxy re-authenticate that is fully asymmetric cryptography based were considered. These protocols were modified, modelled and implemented for transaction over a cloud environment. The performance of the proposed scheme and the reference schemes were evaluated in a simulation environment using the Network Miner and Foglight software framework. Considered performance metrics were contact, communication exchanges, cryptographic operations, end-to-end delay and disruption with the proposed scheme having a better performance.

In addition to the proposed scheme having better communication efficiency and less computational on individual network entities, I was able to establish the independence security tamper proof designed in a way that uses a number of data fields to store inside the application database.

I have also illustrated using puppy OS in our scheme to provide seamless integration of the banking application hosted in USB, so that it is easy for the non-technical user to work with the application. I simulate a near- native performance, especially for displaying banking applications that use rich graphics and multimedia, like web browsers.

The chapter investigated the designs of the operating system that depend upon puppy Linux with added security fields structtask_struct and structlinux_binprm. Filesystem security information was discussed in detail in this chapter, and a security field added to structsuper_block. UNIX domain sockets.
Chapter 6 Cloud model and tamper Proof experiment result

This chapter will try to describe and discuss the cloud computing architecture and how the authentication is being used by a Cloud Service Provider (CSP) and different entities. Conversely the legal implications of the data and application authentication being verified by a third party are complex and not well understood [88]. Assume there is a potential lack of control and transparency when a CSP holds the data. However my resulting security concerns due diligence, audit ability, contractual obligations, cloud provider espionage, data lock-in, and the transitive nature of the data control. My scheme in general depends on trusted computing and applied cryptographic techniques [91].

The previous chapter discussed security from the general network perspective, security issues of the online bank as well as access control and its implementation in different environments. Security modelling and testing is defined as the process of ensuring that the proposed security solution meets defined specification in addition to realizing its intended purpose with minimal effect. Security modelling and testing is associated with verification (compliance of the proposed solution with the initial specification), validation (correctness of the overall output of the proposed solution) and evaluation (ascertaining the effect of the proposed solution on the system).

The focus of this chapter is to: (1) Review the cloud modelling and testing approaches of our servers and formal verification; (2) Justify the preference of this work for the real server based approach; (3) Establish connection between cloud providers and client and (4) Discuss the models framework development process. The chapter is structured as follows: Section 6.1 carries out a cloud computing definition 6.2 general Security issues related to the cloud and in this section our data security issues are reviewed and analysis incoming and out coming network session established. Section 6.3 3 proposes schemes and data security issues and introduces my testbed simulation programming environment/language and access control to path encrypted data over the cloud provider. Section 6.4 presents and describes cloud computing as third parties. Section 6.5 represents the Test bed connectivity link between cloud provider and the bank. Section 6.6 discusses the relation between bank and cloud provider. While 6.7 discusses tamper proof security overview and the experiment results. Section 6.8 scalability issues Section 6.9 makes an overview test analysis and result against the attacks mentioned in chapter three and the summary is section 6.10.
6.1 Cloud Scheme to Communicate with the Bank

Generally, this experiment used Private Cloud to communicate between the bank and cloud provider, hence the cloud infrastructure is operated solely for the bank and managed by the bank. It will be secure by the bank security policy and equipment.

From figure 6.1 there are two portions, the first one a link between the cloud provider and the bank, the second one is a link bank client with cloud provider.

The communication between the bank client and the cloud go through a public cloud and is secured by many security parameters used like SSL client, fingerprint authentication factor and session key.

The cloud is used to transfer data from client to the bank [2, 8 and 26]; well-established cryptographic algorithms allow a cloud storage consumer to encrypt the data prior to insertion in the cloud, and decrypt it after moving it back to the bank or client. It should be noted, however, that such general purpose cryptographic algorithms are not effective if the data is intended for use within the cloud. Of course, merely encrypting the data does not mean that information cannot be inferred from careful observation of the data stream, or of the storage devices under the cloud.
provider’s control. In some cases it may be possible for a computational engine (such as a virtual machine in a cloud environment) to get information, for that reason operations can be performed entirely on encrypted data, producing results without exposing the homomorphic encryption. Input data or the result in plaintext form using schemes known as

6.2 General Security Test and Result

This section investigates efficient methods for handling user access rights, revoking, hence rights efficiently, and issuing either the same or different access rights to a returning user through cloud provider. It addresses the issues of security from a “curious” cloud, collusion among users, and collusion between a user and the Cloud Service Provider. Given the nature of cloud computing, there are a number of new security risks as well as the same traditional risks [36, 54, 56, 59, 79, and 81].

6.2.1 Simulations and Performance Evaluation

Using the Network Miner to parse libpcap files to do a live packet capture of the network traffic between cloud provider and the client is shown in the capture photo below.

Figure 6.2: packet capture of the network traffic between cloud provider and the client

From the figure 6.2 we can see that 42 packets sent (15,278 Bytes), 0.00 % cleartext (0 of 0 Bytes)
Chapter 6. Cloud model & Tamper proof security overview and experiments result

192.168.1.199 (Windows) -> 192.168.1.197 (Windows) : 18 packets (3,190 Bytes), 0.00 % cleartext (0 of 0 Bytes)

TCP: 2639 - 8063 : 12 packets (2,309 Bytes), 0.00 % cleartext (0 of 0 Bytes)
TCP: 2647 - 8063 : 4 packets (172 Bytes), 0.00 % cleartext (0 of 0 Bytes)
TCP: 8055 - 7003 : 2 packets (709 Bytes), 0.00 % cleartext (0 of 0 Bytes)

192.168.1.199 (Windows) -> 192.168.1.199 (Windows) : 14 packets (11,420 Bytes), 0.00 % cleartext (0 of 0 Bytes)

TCP: 2641 - 8067 : 4 packets (776 Bytes), 0.00 % cleartext (0 of 0 Bytes)
TCP: 8067 - 2641 : 10 packets (10,644 Bytes), 0.00 % cleartext (0 of 0 Bytes)

From the screenshot above we can recognize that there are 42 packets of data sent and all these packets are completely encrypted as the clear text is 0%.

The analyzing of Public Server link between cloud provider and client can be explained as follows:

IP: 192.168.1.197
MAC: Unknown
Hostname: N/A

6.2.2 Simulation Setup
OS: Windows
TTL: 128 (distance: 31)
Open TCP Ports: 8063
TCP 7003 - Entropy (in \ out): 69.20 \ 69.01 Typical data (in \ out): HTTP/1.1 000 Cootaeceee/
control: \ tose /tesao to/seeoeotioe/ se

Sent: 20 packets (16,529 Bytes), 0.00 % cleartext (0 of 0 Bytes)

192.168.1.197 (Windows) -> 192.168.1.199 (Windows) : 19 packets (16,419 Bytes), 0.00 % cleartext (0 of 0 Bytes)

TCP: 8063 - 2639 : 13 packets (15,432 Bytes), 0.00 % cleartext (0 of 0 Bytes)
TCP: 8063 - 2647 : 3 packets (132 Bytes), 0.00 % cleartext (0 of 0 Bytes)
As we can see from the analyses of port: 8063, the port opens with TCP protocol and the port sent 20 packets = 16,529 with 0.00% clear text. Also they received 18 packets = 3,190 bytes with 0.00% clear text.

As can show from capture above there are 10 packets send from IP: 192.168.1.197 which carries 16,529 Bytes but the actual size packet received to 192.168.1.199 (bank side) is 3,190 Bytes which is mean that the cloud provider operate with of 13,339 bytes and sends the rest to the bank.

**Incoming sessions: 2**

**Server: 192.168.1.197 (Windows) TCP 7003**

Server: 192.168.1.197 (Windows) TCP 7003 (735 data bytes sent), Client: 192.168.1.199 (Windows) TCP 8055 (629 data bytes sent), Session start: 10/21/2014 3:00:46 PM, Session end: 10/21/2014 3:00:46 PM

**Server: 192.168.1.197 (Windows) TCP 7003**

Server: 192.168.1.197 (Windows) TCP 8063 (0 data bytes sent), Client: 192.168.1.199 (Windows) TCP 2647 (0 data bytes sent), Session start: 10/21/2014 3:00:39 PM, Session end: 10/21/2014 3:00:44 PM

**Outgoing sessions: 1**

**Server: 192.168.1.199 (Windows) TCP 2639**
Server: 192.168.1.199 (Windows) TCP 2639 (1829 data bytes sent), Client: 192.168.1.197 (Windows) TCP 8063 (14900 data bytes sent), Session start: 10/21/2014 3:00:36 PM, Session end: 10/21/2014 3:00:46 PM

### 6.2.3 Simulation Result:

<table>
<thead>
<tr>
<th>Device from</th>
<th>TCP Decide to</th>
<th>TCP</th>
<th>Session start</th>
<th>Session end</th>
<th>Session time</th>
<th>Byte encrypt Without encryption</th>
<th>B/C</th>
<th>Proposal scheme packet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public Server</td>
<td>8063</td>
<td>Private</td>
<td>2639</td>
<td>10/21/2014 3:00:36 pm</td>
<td>10/21/2014 3:00:46 pm</td>
<td>10</td>
<td>1490 0</td>
<td>10</td>
</tr>
<tr>
<td>private</td>
<td>8063</td>
<td>bank</td>
<td>1647</td>
<td>10/21/2014 3:00:39 pm</td>
<td>10/21/2014 3:00:44 pm</td>
<td>5</td>
<td>629</td>
<td>5</td>
</tr>
<tr>
<td>bank</td>
<td>8067</td>
<td>public</td>
<td>1241</td>
<td>10/21/2014 3:00:36 pm</td>
<td>10/21/2014 3:00:47 pm</td>
<td>11</td>
<td>1022 0</td>
<td>11</td>
</tr>
</tbody>
</table>

**encrypt:** Network encrypted traffic from the current analysis  
**Without Encryption** Network traffic without encryption  
**B/C** Direct Network traffic from client to Internet Banking Server

<table>
<thead>
<tr>
<th>encrypt:</th>
<th>Network encrypted traffic from the current analysis</th>
<th>Without Encryption</th>
<th>Network traffic without encryption</th>
<th>B/C</th>
<th>Direct Network traffic from client to Internet Banking Server</th>
</tr>
</thead>
</table>

Table 6.1: simulation result

Table 6.1 show the three server communication: public server which placed at cloud provider datacentre and link the client to the bank through cloud server, private one which place in cloud provider datacentre and link cloud provider with the bank datacentre through lease line dedicate for bank services only, bank server link between client and the bank directly without any interfering from cloud provider. The table show the port in used and the session started/ end time by millisecond and the byte size of each packet this packet encrypted by homographic encryption algorithm and in other case not used homomorphic algorithm. And the result of this table can be represent in figure 6.3 and 3.4.
**Chapter 6. Cloud model & Tamper proof security overview and experiments result**

**Table key:**

<table>
<thead>
<tr>
<th>Proposed scheme: means full encryption traffic from bank to client goes through cloud provider</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without Encryption: traffic from bank to client goes through cloud provider but without Homomorphic cryptography feature</td>
</tr>
<tr>
<td>Direct: from the bank to client without interference from cloud provider</td>
</tr>
</tbody>
</table>

From figure 6.3 above, can display the time comparison between three different entities, public to private means network link between bank client and cloud provider, private to bank means network link between cloud provider and the bank, bank to public means network link between client bank and the bank. Time consumption to link bank to client through proposed scheme is less than 10 MS while time need to link cloud to the bank when using proposed scheme is 4 MS and the communication between bank client and the bank via proposed scheme is less than 12 MS; the chart declares that the highest time is above 10 MSc at the link between bank and client when using the proposed scheme while the same link if not used proposed scheme take 7 MSc and that is indicate there is slight different in time although there is huge benefit when used proposed scheme because the data is encrypt.

**Figure 6.3: time comparison/MS**

**Proposed scheme**

- From figure 6.3 above, can display the time comparison between three different entities, public to private means network link between bank client and cloud provider, private to bank means network link between cloud provider and the bank, bank to public means network link between client bank and the bank. Time consumption to link bank to client through proposed scheme is less than 10 MS while time need to link cloud to the bank when using proposed scheme is 4 MS and the communication between bank client and the bank via proposed scheme is less than 12 MS; the chart declares that the highest time is above 10 MSc at the link between bank and client when using the proposed scheme while the same link if not used proposed scheme take 7 MSc and that is indicate there is slight different in time although there is huge benefit when used proposed scheme because the data is encrypt.
From figure 6.4 the bytes comparison can be observed and when tracing the proposal scheme packet can observed the number of byt in a public to private is 14900 bytes while 629 /b between private and the bank and in the link between bank and the public 1022 bytes.

6.3 Proposed Schemes and Data Security Issues.

Data Security presents a number of general issues as well as issues specific to data in the cloud. Some proposed schemes rely upon the published SLA of the provider [46, 48]. My scheme test bed of data security depends on trust enforcement phase (discussed later in section 6.6), facilitated using trust information (shared key(s) is in the next data outsource section, to obtain the secure session.

6.3.1 Access Control to Encrypted Data in the Cloud.

Our approach relies heavily upon encrypted data in the cloud. This approach has been studied in a number of different models [20, 73, and 87]. My models use encrypted data in the cloud to protecting the data from the cloud itself. The problem with encrypted data in the cloud is the
probability of limitations placed upon data searches [70, 72]. Our solution is to fix this problem and to access control to make couple access control with re-encryption and guard data from the cloud for unauthorized access without influencing the utilisation or performance of the application. Figure 6.5 represents IP access to cloud provider server simultaneously. Bank interface represents 16.44% of the server capacity in the time diagram capture, cloud provider interface linking bank client to the cloud provider represents 13.27% and the bank DB link to the cloud server to the bank server represents 22.73%. From this diagram there is a limit to the server queue to hand each IP packet without impacting on the performance.

Figure 6.5: top IP host

Netwalk tools in our system function to represent the percentage of IP usage for communicating the client to the bank using puppy Linux through cloud provider.

From the diagram in figure 6.6, although access control and re-encryption are applied still the cloud provider to client interface (public) percentage is 13.27% with carry of 2.725 packets representing 543.63 KB and this volume is less than the performance of the transaction.

6.3.2 Data Outsourcing –

This section presents the proposed authentication scheme as well as compares schemes in a real server environment using contact, communication exchanges, cryptographic operations, end-to-end authentication and disruption as metrics. The primary aim of this study is to: (1) Investigate if the proposed scheme meets the set objectives, depending on server availability as well as placing no load on the server during post trust establishment network communication; (2) Establish the
comparative advantage of the proposed scheme over the reference existing schemes. Common to the proposed and the reference schemes is the sender’s contact with the STS framework to obtain authentication share key used to encrypt/decrypt the message which is assumed to take place prior to the process of forwarding the message to the destination [40,62,65].

6.4 Cloud Computing as Third Parties (enforcement phase):

This section discusses Entities that can create an established trusted party among themselves. A trusted party can solve many problems concerning security services and key exchange.

A solution is to use a trusted cloud provider, and conceder as a key-distribution center (KDC) and transit point. A secret private link communicates the cloud provider with the bank.

A secret key is established between the KDC and each client with an SSL client authentication session. Any client has a secret key with the KDC, referred to as $K_{\text{client account + tamper proof id}}$. The bank has a secret key with the KDC, referred to as $K_{\text{Bank}}$. The process for the client to establish a connection is as follows:

1. Client sends a request to the cloud (KDC) stating the need for a session secret key between client and the bank
2. The KDC informs the bank about the client request after checking tamper proof ID and session key between them.
3. If bank agrees a session key is created between the two.

The secret key between client and the bank established with KDS is used to authenticate the client to the bank and to prevent any eavesdropping.

Cloud provider creates a session key $K_{\text{CB}}$ between the client and bank by the following steps:

1. Client sends a message to the cloud provider (KDC) to obtain a symmetric session key. The message contains client account, tamper proof ID, user identity and client fingerprint.
2. The KDC receives the message and creates what is called a ticket (ssl session key).
3. The cloud provider (KDC) verifies the tamper proof ID is correct first then sends a client request to the bank
4. The bank sends a challenge to the client ($R_B$) and one time password, encrypted with the session key to KDC
5. KDC sends an encrypted message to the client that includes client and bank information and session key and encrypted ticket for the bank. The whole message is encrypted with a client tamper proof ID key.
6. Client sends the bank’s ticket (session key) to the bank
7. Client responds to the bank’s challenge, Note that the response carries $R_B-1$ instead of $R_B$
Figure 6.6: network traffic between cloud provider and client

Figure 6.8 used wireshark to gives an overview of all traffic inbound or outbound from the bank to cloud provider and from this figure we can understand the huge number of IP passes to the cloud provider. In this figure used wireshark to let bank server analysis all the traffic inbound and outbound the bank web application to utilize the performance of rel. time and date time.

6.4.1 Test Key Generation and Distribution

This stage acts as an initialization step when the Bank generates two kinds of key pairs based on the additive homomorphic encryption as follows:

First, the Bank creates a master key pair, (pka, pra). The Bank keeps the private key (pra) as their secret.

Second, for each authorized account, the bank creates a public/private key pair (pkb, prb) and gives it to the client when initializing their account with a tamper proof device. In addition, the Bank’s public key pka and client’s public key pkb are treated as public and sent to the cloud provider.

Furthermore, the Bank generates $r_{pka \rightarrow pkb}$ (cashing re-encryption key) for each authorized client tamper proof (which will be used during the Data Outsourcing Stage) using pra and pkb.

In this proposal we deal with three entities which are bank side, client and cloud provider used as a middle body for out of band authentication processes and to update the banking system application at the client side.
We have three different components; public cloud (cloud provider), private cloud (relation sbtween bank and cloud provider) and Internet Banking (the Bank).

For security transactions, at first the client inserts a USB device (tamper proof); this device contains a bank application run on a different OS and completely run in an isolated environment on the client P.C. The USB is capable of running automatically and requires the client to enter his/her fingerprint, user name and password, pin number.

When the client presses the button to confirm (after entering user name, password, pin and his/her fingerprint) these values pulse the unique tamper proof ID register, sent to the public cloud, and we check only tamper proof ID and password plus session key created between the cloud provider and client; if it matched then these four values are sent through the private cloud connection to the bank. In this step we checked all values and if they matched then the client can access their account. The bandwidth and the packet size can be light as shown from the snapshot taken from my real test bed scenario.

6.4.2 Cashing Re-Encryption

Cashing re-encryption (CRE) [76, 79] allows a semi-trusted intermediary to re-encrypt data for delivery to a specific user without requiring the data to be decrypted and re-encrypted with a trust certificate embedded on a cashing server to avoid any untrusted message. Furthermore, a key pair can be generated to allow the encrypted data to be delivered in a re-encrypted form such that the Client can decrypt the data while the cloud cannot.

Definition1. Let x be the account owned by the Bank with public key \( pk_a \). Without loss of generality, assume that T (the cloud provider) knows the cashing re-encryption key certificate denoted as \( rk_{pk_a \rightarrow pk_b} \), where \( pk_b \) is the client’s public key. If the bank computes \( E_{pk_a}(x) \) and hands it to T, then T re-encrypts it for the client:

\[
\text{PRE.ReEnc} (E_{pk_a}(x), \text{rk}_{pk_a \rightarrow pk_b}) \rightarrow E_{pk_b}(x) \quad (1)
\]

Where PRE. ReEnc is the cashing re-encryption function. After this, T sends the output \( E_{pk_b}(x) \) to the client who can decrypt it to retrieve \( x \) using his private key \( pr_b \).

The observation is that T can first operate on the encrypted data sent by the Bank, i.e., \( E_{pk_a}(x) \), using additive homomorphic properties and apply the PRE certificate to the updated data. Therefore, the combination of additive homomorphic properties with the PRE certificate will benefit the data owners to shift the work load to T (i.e., the cloud that is computationally unbounded) and will provide a medium for T to operate on the encrypted data.
Chapter 6. Cloud model & Tamper proof security overview and experiments result

<table>
<thead>
<tr>
<th>Device from (Cloud – server)</th>
<th>%</th>
<th>Session start</th>
<th>Session end</th>
<th>Session colour</th>
<th>Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public Sever (Cloud – client)</td>
<td>25.20%</td>
<td>10/21/2014 3:00:36 pm</td>
<td>10/21/2014 3:00:46 pm</td>
<td>Red</td>
<td>14900</td>
</tr>
<tr>
<td>Private (Cloud – Bank)</td>
<td>20.55</td>
<td>10/21/2014 3:00:39 pm</td>
<td></td>
<td>Green</td>
<td>629</td>
</tr>
<tr>
<td>Bank (Bank – Client)</td>
<td>36.67%</td>
<td>10/21/2014 3:00:36 pm</td>
<td>10/21/2014 3:00:47 pm</td>
<td>Blue</td>
<td>10220</td>
</tr>
<tr>
<td>Puppy (Client – cloud)</td>
<td>12.42%</td>
<td></td>
<td></td>
<td>Pink</td>
<td>154</td>
</tr>
</tbody>
</table>

Table 6.2: overall packet size distribute

Figure 6.7: Packet size distribution after applying the caching re-encryption algorithm.

From table 6.3 and figure 6.10 we can show the package percentage usage for each server and the capacity size consumed for the request coming from the client represented by puppy through public server (cloud –client) then to bank from cloud via private (cloud -bank), and finally the connection between bank and client.

6.5 Testbed Connectivity Link Between Cloud Provider and the Bank

The result of my test bed is as follows:

Server: 192.168.1.199 (Windows) TCP 2641
Server: 192.168.1.199 (Windows) TCP 2641 (615 data bytes sent), Client: 192.168.1.199 (Windows) TCP 8067 (10220 data bytes sent), Session start: 10/21/2014 3:00:36 PM, Session end: 10/21/2014 3:00:47 PM

Figure 6.8: snap shot from client PC

From this result in figure 6.8 we can say data outsourcing is proof of this formula and the functionality is accurate.

Data Access- Consider account1, who sends a data request to the cloud. Upon receiving the data request from account1, the cloud first checks whether account1 is authorized to access any of the data records. For simplicity, we consider the case of a single data record.

Algorithm1 Data sharing key (d,S)

Require: The Bank wishes to share data record d=d1,..,dn (Note: The master private key prA is known only to the Bank; prB is known to both bank and client; pkA and pkB are public)

{Steps1-13 performed by the Bank}

1- Pick sn+m random numbers r1,..,rn+m(ri∈ZN,for 1≤i≤n+m)

2- Generate d=d1+r1,..,dn+rn, r_{n+1},...,r_{n+m}
Algorithm 2 Data_Access(d)

Require: the client sends a data request to the cloud (Note: The private key $pk_b$ is only known to bank and client.)

{Steps 1-10 performed by the cloud}

1. Receive data request from client
2. If No entry for client in $T^d$ then
3. Abort
4. Else
5. For $i := 1$ to $n + m$ do
6. $Epk_b(d_i) ← PRE.ReEnc(Epk_a(d_i), rk_{pk_a} → pk_b)$
7. $Epk_b(d_i + \alpha_i) ← Epk_b(d_i) + hEpk_b(\alpha_i) \mod N^2$
8. End for
9. Sends $E(d + \alpha)$ to Client, for $1 \leq i \leq n + m$
10. End if

{Steps 11-14 performed by Client}

11. If client is authorized to access $d$ then
12. Receive data from the cloud
13. $D'_i + \alpha_i \leftarrow D (E_{pk_b}(d'_i + \alpha_i))$, for $1 \leq i \leq n + m$
14. End if
However, similar steps can be used for all data records in the cloud. For a data record d, the cloud checks whether the client is authorized to access d by using the authorization tamper proof list Td associated with d (as discussed in Stage2). If there is no entry related to the client in Td, then the cloud simply aborts the client's request. On the other hand, if there exists an entry for the client, i.e., Td, then the cloud proceeds as follows:

Uses the cashing re-encryption key rpk → pk in T d and converts Epka (d ) to Epkb (d’) by computing Epkb (d’1), Epka (d’) by computing (Epkb (d’1), …, Epkb (d’n+m)).

Computes Epkb(d1+α1)←Epkb(d1)+h Epkb(αi)modN 2, for 1 ≤ i ≤ n + m. Where +h is the additive homomorphic property and N is the group size which is also part of pkb i.e., the public key of the client.

Sends Epkb(d1 +α1), …,Epkb(dn+m+αn+m) to the client.

6.6 Relation Between Bank and Cloud Provider

6.6.1 Enterprise Architectures:

Focusing on the technological components of enterprise architecture [27, 47, 49, and 50], we have categorized them into three logical tiers:

- Top Tier (Front-End) – Represents the technologies that support end-user interactions (data access, analysis, visualization, collaboration, input, personalization, etc) with information/data and other stakeholders.
- Middle Tier – Represents the middleware utilities, services, and support components that optimize system interactions amongst all people and information.
- Bottom Tier (Back-End) – Represents the core information architecture, system security, and access / identity management components to support a secure, efficient, and effective operation.

Defining building blocks and outlining foundations is a critical first step to support coordination and comprehension. Particularly in the world of enterprise architectures, this process is critical to align stakeholders up front and to put development efforts into perspective. Whether it is boxes, lines, definitions, or discussions, sometimes a little language goes a long way.

Financial institutions have developed different levels of maturity – functional, technology, operational as well as service delivery – across these components. Some of these components are already being delivered as a service from external service providers as shown in the figure below.
Figure 6.9: financial institution architecture [4]

In the figure 6.9 the generic financial architecture is shown and my focusing area in a red color.

Financial institutions demonstrate significant maturity in terms of business architecture, process deployment, standard operating procedures to drive operations, technology adoption as well as service delivery. Financial institutions are forced to drive efficiency and agility results within the
organization; this has become imperative today with the after-effects of the economic impact as well as the mindset changes within the ecosystem.

The reference architecture model for cloud computing as shown in figure 6.10 might be changed to obtain what the financial sector is looking for.

From my view, the Bank sector looking at Cloud based solutions with distinct expectations is like a Dynamic and flexible Technology model, highly optimized with virtualized Infrastructure leveraging the right hardware components and managing computing needs as a function of real time usage, redundancy and resiliency upfront, fully automated Service provision, shared Services delivered across trusted domains delivering Security of data, transaction & operations.

Enabling integrated treatment of security across all the operating platforms enables businesses to deliver business applications to their customers and is associated with multiple devices around-the-clock.

![Cloud Architecture Domain](image-url)

Figure 6.10: cloud architecture domain [49]
There is a clear relationship between cloud computing and federated Identity management (FIDM). This can be considered as a superset of all the corresponding issues from these paradigms and many more [88].

6.6.2 The New Financial Cloud Reference Model

All these architectures need a new model design. The design is formulated as a consequence of constrains that a financial institution must respect and the attributes that would give it an advantage over competitors in the market today. Some of these design principles are:

1. The solutions must be based on novelty and flexibility.
2. Cloud computing is a master component of the future system.
3. Sensitive data must not be outsourced to third party providers.
4. Access to the secure part of the website requires a combination of knowledge and control.
5. The primary channels of Internet banking are the Internet and Call center, Smart phone and any new communication device cloud to appear later.

The development base for a new IT strategy includes areas that could benefit from Cloud computing. These have been identified based on their use of sensitive information that could not be outsourced and are depicted in the Strategy Scope.

The reference architecture is designed according to the target operating model requirements. Consequently, they are represented within different structures of the architecture. Also while designing this reference architecture non-core domains have been outsourced and flexibility has been added to allow for a plug-and-play approach towards new distribution channels and products through Cloud integrated solutions.

The Bank in the Cloud can focus on primary banking activities and worry less about non-core banking software and hardware provisioning. This way the Operations and IT can handle larger customer demand and focus on delivering quality to the clients. The key activities of the bank mainly do not change, but from a business point of view it is back to basics, without great concerns regarding IT infrastructure as can show in figure 6.11. Cloud provides flexibility and scalability in order to put into practice various needs and because of its model, it is pay per use. This represents a large investment difference in comparison to dedicated on-premises software and hardware.
In order to represent this proposed reference architecture, Financial IT components are mapped to the immediate model according to the requirements. Figure 6.15 shows the integration model.
between cloud and financial architecture mode, and the red represents the area which this thesis focuses on.

Figure 6.12: proposal for financial cloud reference model area

The figure 6.12 shows the new cloud finance model with red lines on the focused areas.
To be more explicit, the Reference Architecture is composed of:

- Customer Interaction Cloud, an Open Public SaaS solution package that handles all customer interaction integrating mobile accessibility, Internet portal and Call Center solutions. This solution package should also provide self-service via the Internet, single point of contact for all customers, multi-device interaction and communication via social media. Also Marketing and Financial Advice software solutions should be part of this Cloud, thus improving client-oriented services and increasing communication capabilities amongst software components.

- Financial Cloud, a Community Public SaaS solution package that connects messaging, ATM and market feed from the whole banking community (Central Banks and other financial institutions) in order to provide precise and steady communication.

- HR Enterprise Public Cloud integrating HR data, HR performance management and software solutions.

- Banking Product Development and Management Cloud which is an Enterprise Public SaaS and requires Data storage in order to perform risk calculations and predictions.

- Bank Regulatory Enterprise Public SaaS that should allow automated reports to regulators and integrate seamlessly with the general ledger application.

- IT management, development and test Cloud provided via an Enterprise Public IaaS in order to allow scalability and instant provisioning of resources.

- An Enterprise Private PaaS for security management solution that allows data protection and integrity.

- Financial institution. Also it is the layer that facilitates Cloud brokerage.

- An on-premises Data Warehouse for all the sensitive information.

- Multiple packages of on-premises banking software solutions specialized for different business units.

The proposal of financial organization architectures and their business operations models can indicate that there is no single cloud solution model that can meet all the requirements. I think financial institutions will create and manage a federated service ecosystem of cloud-based solutions along with non-cloud based cases, leveraging the abstraction control provided by public, private and community cloud deployments. A federated ecosystem can leverage multiple archetype implementations, deliver elastic capacity to match business requirements and provide an incremental adoption path building for the success of each increment. This will also enable
external business service providers, who themselves would start offering their product solutions on the cloud, to seamlessly integrate with this ecosystem.

The key resources are converted from technology to knowledge. Technology exists, but putting it to good use, in accordance with regulations, and managing it for best performance is the main focus for any institution adopting Cloud computing. Also valuable resources are the lessons learned in the past, from the process of maturity within the on-premises software. This is the starting point of any SLA and therefore is the main guideline regarding what expectations a financial institution can have from a Cloud provider. Regarding physical resources, the accent shifts towards networking and connectivity. Cloud computing is purposeless in an environment where the networking infrastructure cannot allow access to the applications, processing capacity and storage space. Because a financial institution is taken into consideration, it is worth mentioning that the existing resources are not lost, but just reused in a different scope, for instance to increase the number of customers that can be stored or processed at one time.

6.7 Tamper Proof Security Overview and Experiment Results

We have performed an experiment in order to evaluate the overhead of the techniques relative to running the puppy Linux operating system on the ‘bare metal’. Complex application-level benchmarks that exercise the whole system have been employed to characterize performance under a range of server-type workloads. The test machine was configured with one CPU for these experiments.

Online banking application is implemented based on the puppy Linux platform, which needs a management purview called Kernel. Considering the diversity of the user’s OS, such as different architecture and different version, and the modification to an OS to fit for our application, the user’s OS is not the privileged Kernel. Instead, we build a new-compiled OS kernel as the privileged controllers and make the banking application run as user privileged side by side with the OS as figure 6.16 demonstrates.
Figure 6.13: kernel and interface link

Figure 6.13 show the main hierarchical concept link between hardware layer and other layers

**Test Analysis Report**

Experiments were performed using the PostgreSQL 7.1.3 database, exercised by the Open Source Database Benchmark suite (OSDB) in its default configuration. We present results for the multi-user Information Retrieval (IR) and On-Line Transaction Processing (OLTP) workloads, both measured in tuples per second. The benchmark drives the database via Foglight which is a quest software monitoring application used over a Linux storage socket. Foglight places considerable load on the operating system, and this is reflected in the substantial Linux overheads experienced by O.S. In particular, the OLTP benchmark requires many synchronous disk operations, resulting in many protection domain transitions.

A number of client machines are used to generate load for the server under test, with each machine simulating a collection of users concurrently accessing the web site. In order to demonstrate the performance isolation provided by puppy Linux, we perform a “bake-off” between puppy and other OS-based implementations of performance isolation techniques such as resource containers. I present results showing that Puppy performance isolation works as expected, even in the presence of a malicious workload. We ran two operating system companies with Linux operating system with domains configured with equal resource allocations, two O.S. each running a pair of extremely antisocial processes. The third domain concurrently ran a disk bandwidth hog (sustained dd) together with a file system intensive workload targeting huge
numbers of small file creations within large directories and intensive application, which attempted to allocate and touch 3MB of memory, on failure, free every page and then restart.

Figure 6.14: Concurrent Processes/ O.S, Domains
Figure 6.14 gave complete monitoring view to network, memory, and CPU and storage utilization and from that snapshot we can measure the capacity and performance of the system. As seen from figure 619 CPU utilization is less than 3.4% of the general CPU and the memory in use is 58% from total memory 12GB; also the network is 1.5 Mb/s with no indication of send rate while receive rate is less than 10 Mb/s.

I use Burpsuite (Burp Suite is a Java application that can be used to secure or penetrate web applications) and NetworkMiner which is a Network Forensic Analysis Tool (NFAT) for Windows that can detect the OS, hostname and open ports of network hosts through packet sniffing or by parsing a PCAP file. NetworkMiner can also extract transmitted files from network traffic. For analyzing the requests sent through. Burpsuite is available by default in Backtrack and Loki. In order to intercept the requests and manipulate them, I must configure our browser to use Burp’s proxy, which is 127.0.0.1:8080 by default. We will also be using Wireshark a bit.

6.8 Scalability Issues
Scalability can help increase performance by providing an application server with horizontal scaling. To eliminate all scalability bottlenecks, explore the model on an approach that replicates all three tiers used in the proposal (web server, application server, back-end database). Web application code is executed application server shown by the code embedded in a tamper proof device which provides the functionality of a web application server. The database tier is a bank
Chapter 6. Cloud model & Tamper proof security overview and experiments result

Database server. The web tier is made of the Apache Web Server along with the Apache Tomcat SP/servlet containers provided by the cloud provider. The application tier is a J2EE compliant application server. The Apache Web Server communicates with the Apache Tomcat servlet container using the Apache mod_jk connector. The Tomcat servlet container communicates with the application server using JNDI and RMI calls. The application server communicates with the bank Database using JDBC. The Apache Tomcat servlet container and the J2EE compliant application server both require a JVM to run.

6.9 Overview of Attacks

Although Appendix B can provide details, here is an overview of some attacks and how new models prevent vulnerability as mentioned in chapter 3

6.9.1 Malware or Malicious (and Other Malicious Code-Based Direction Schemes)

Referring to an attacker in chapter 3 section 3.4, the new model can prevent such an attack like this. Obviously any intruder needs to gather some information before attacking their victim; to mitigate the information obtain by reconnaissance method, application server rejects the request information applied by default at the setup page of the browser option service and this means from the beginning the request will not go further.

This page is not available, sensitive metafile information to be exposed, Google cache check removes unnecessary information
From the figure 6.15 above they prevent any chance to create or inject any malware because from the first step they can’t go farther, or capture the information needed to get or post the code.

6.9.2 Recording Software Keyboard Buttons

Referring to the attacker mentioned in chapter 3 section 3.7.1 smart card reader and fingerprint method controls this type of attack, and the control is done over two commands.

- **Perform Web Application Fingerprinting:**
  Use secure channel with pattern encrypted.
Figure 6.16: figure print mechanise

Figure 6.16 show mechanise used to analyses the fingerprint is pattern mechanism.

- **Identify user roles over linux mode:**
  User right for any clients and admin from local network inside the bank only.

By this setup be sure no one can record software keyboard buttons or manipulate the hardware.
6.9.3 Scams. Phishing Websites/Emails

Referring to chapter 3 sections 3.4.2, there is no valuable information that can be obtained when using a spider/crawl script in programs. As shown from the figure 6.17 below there is no way to inject a phish command.

![Figure 6.17 snapshot source code](image)

6.9.4 Test for Web Storage SQL Injection

Referring to chapter 3 section 3.4.3 malicious software installation can control this vulnerability by Check Cross Origin Resource Sharing (CORS) implementation:

Validate URLs passed to XMLHttpRequest.open. Current browsers allow these URLs to be cross domain; this behavior can lead to code injection by a remote attacker. Extra attention to absolute URLs will be paid. Use the Access-Control-Allow function to ensure that URLs are responding with Access-Control-Allow-Origin:

No sensitive content t includes information that might aid the attacker in further attacks. Use the Access-Control-Allow-Origin header only on chosen URLs that need to be accessed cross-domain. Don't use the header for the whole domain.
Allow only selected, trusted domains in the Access-Control-Allow-Origin header.

Current implementations might not perform this request, so it’s important that "ordinary" (GET and POST) requests perform any access control necessary.

The online bank system proposed discards requests received over plain HTTP with HTTPS origins to prevent mixed content bugs.

6.9.5 Man In The Middle (MITM) Hacking Online Banking and Credit Card Transactions

This is the process of verification that an individual or an entity is who they claim to be. Authentication is performed by submitting a user name or ID and items of private information (finger print) that only a given user should know. Wireshark provides support for Lua. Lua is a lightweight yet powerful programming language. It is used for dissectors and taps. It also allows saving the captured packets into various formats which can be utilized for later analysis by Wireshark packet analysis applications. Similarly, for reading packets from various different format packets.

![Figure 6.18: traceroup process](image)

The figure 6.18 above represents tracerout command used to communicate and trace the IP traffic.
6.9.6 Test for Brute Force Protection:

If an attacker is able to guess passwords without the account becoming disabled due to failed authentication attempts, the attacker has an opportunity to continue with a brute force attack until the account is compromised. Automating brute-force/password for guessing attacks on the web applications is a trivial challenge. To mitigate this risk use Password lockout mechanisms, lock out an account if more than three unsuccessful login attempts are made. An attacker that undertakes a large number of authentication attempts on known account names can produce a result that locks out entire blocks of user accounts. Given that the intent of a password lockout system is to protect from brute-force attacks, a sensible strategy is to lockout accounts for a period of time (20 minutes). This significantly slows down attackers, while allowing the accounts to reopen automatically for legitimate users. The solution in this proposal is multi-factor authentication, a very powerful deterrent when trying to prevent brute force attacks since the credentials is a moving target.

6.9.7 Test for Authentication

Use security authentication and security layer (SASL) for application and SMTP AUTH.

First edit /etc/default/saslauthd in order to activate saslauthd. Remove # in front of START=yes, add the PWDIR, PARAMS, and PIDFILE lines and edit the OPTIONS line at the end:

```bash
# This needs to be uncommented before saslauthd will be run automatically
START=yes

PWDIR="/var/spool/postfix/var/run/saslauthd"
PARAMS="-m ${PWDIR}" 
PIDFILE="${PWDIR}/saslauthd.pid"

# You must specify the authentication mechanisms you wish to use.
# This defaults to "pam" for PAM support, but may also include
# "shadow" or "sasldb", like this:
# MECHANISMS="pam shadow"

MECHANISMS="pam"

# Other options (default: -c)
# See the saslauthd man page for information about these options.
#
# Example for postfix users: "-c -m /var/spool/postfix/var/run/saslauthd"
# Note: See /usr/share/doc/sasl2-bin/README.Debian
# OPTIONS="-c"

#make sure you set the options here otherwise it ignores params above and will not work
OPTIONS="-c -m /var/spool/postfix/var/run/saslauthd"
```
**sudo dpkg-statoverride --force --update --add root sasl 755 /var/spool/postfix/var/run/saslauthd**

This may report an error of "--update given" and the "/var/spool/postfix/var/run/saslauthd" directory does not exist. You can ignore this because when you start saslauthd next it will be created.

### 6.10 Summary

The focus of this chapter was: (1) additive homomorphic encryption; and (2) cashing (proxy) re-encryption to secure data sent to the cloud and transited from the cloud to a user. This chapter tested the proposed authentication scheme as well as compared schemes in a real server environment using contact, communication exchanges, cryptographic operations, end-to-end authentication and disruption as metrics. For test Key generation and distribution initialization step the Bank generates two kinds of key pairs based on the additive homomorphic encryption.

For Simulations and Performance Evaluation, in this chapter, I use Network Miner to parse libcap files to do a live packet capture of the network traffic between cloud provider and the client. From the screenshot on this chapter we can recognize that there are 42 packets of data sent and all these packets are completely encrypted as the clear text is 0%. The analysing of Public Server link between cloud provider and client can be explained as port: 8063 open with TCP protocol and the port sent 20 packets = 16.529 with 0.00% clear text. Also they received 18 packets = 3.190 byte with 0.00% clear text. Also it can be observed that the time comparison between bank to client through cloud (public) is above 25 msc at the stage of the direct link between bank and client while the best one is less than 5 msc in the direct stage between public and private. Also it can be observed that the bytes comparison for time between public to private is 25 msc while 15 msc between private and the bank and in the link between bank and the public it is 10 msc.
Chapter 7 Conclusion and Future Work

7.1 Conclusion
The security of a system can be seen as a chain of protection methods, and the global level is just as strong as its weakest link. In the financial world there exist tools, like user behaviour profiling, which are starting to be considered, and that in fact, have been used by the banks for many years in the “real world” in the form of back-end controls. From the dot com boom, banks have tried to push the entire business logic, authentication included, to the front end (the user’s computer), as it is easier for them. The possibility of ubiquitous transactions and distinction between ‘safe’ and ‘dangerous’ ones vanished too. All of these get worse in the cases of user’s identity or credential theft, which need more advanced measures and, curiously, a return to traditional back-end controls should be expected. In the last term, exclusively technical protection methods seem not to be able to completely solve the situation. The focus is towards trying to increase the resistance of the system, instead of its invulnerability that, in any case, cannot be completely achieved in general purpose operating systems. The objectives of this research work were: (1) to propose and implement a novel safety operating system that is isolated against O.S. threats like masquerading, modification, Trojan and viruses; (2) to investigate how traditional authentication and cryptographic techniques can be used judiciously to provide an authentication scheme that does not depend on direct server availability and recipient network connectivity for security processing of received transaction; (3) to propose and implement an alternative access control credential to digital certificate to banking client; (4) establish the comparative advantage of combining symmetric and asymmetric based schemes over using completely asymmetric based schemes.

7.2 Overview of Key Contributions
The main focus of this thesis has been revision of a functional tamper proof-based access control scheme for untrusted PC when accessing online banking. The study develops a trust Operating System, proposed and implemented a new protocol called Secure Transaction Sharing (STS) to grant the security communication between client and the bank via cloud provider and finally develops a new financial, cloud reference architecture framework appropriate to financial sector needs. The proposed scheme’s objectives were realized through simulation. Presented below is an overview of key contributions.
Chapter 7 Conclusion and Future Work

(1) **Revision and functional tamper proof authentication appliance**

An authentication Tamper proof designed for extremely high security demands and confidentiality of the online banking application used the biometric security technology system. Tamper proof is also an on-chop authentication system; all the authentications are run inside the USB key without the host OS’s help. At the same time, installing the entire software environment on the USB tamper proof flash is controlled by the tamper proof key.

The system is a combination of hardware and software as shown in figure 5.2. Based on the USB key and fingerprint scanner, the tamper proof device integrates a plug-and-play USB2.0 device without driver; Tamper proof can be divided into three stages as described in chapter 5 section 5.3.1. We have developed and integrated a fingerprint based authentication security portable custom-built software environment inside the USB device. With the help of the system the clients can work in a movable security environment freely, as well as have high-level protection of confidential information. From the testbed in chapter 6 in figure 6.6 we notice that client interface (public) percentage is 13.27% with carry of 2.725 packets representing 543.63 KB. And from Table 5.3: boot time in Milliseconds is 6517.00 while windows bootable time is 12868.00, from Table 5.2: comparison O.S. Performance test is 01:08:37 and website loading 00:10:01 while windows booting is 02:13:88 and website is 00:16:35 and from this figure performance is 50% better than the current status.

(2) **Applied Automatic time drift management scheme**

I proposed and implemented an amendment Online banking security appliance automatically managing the clock drift for every individual tamper proof. The usage of time-based algorithms has a number of major benefits: the generated one time password can be used more than once and they have a very limited life span thereby requiring the fraudster to immediately consume a captured one time password, discouraging phishing attacks.

(3) **Used fingerprint authentic factor**

I established a fingerprint encryption module to the USB Key device and developed a suite of fingerprint encryption programs both on the host OS (Windows XP) and the tamper proof USB Key device. The communication is based on the USB mass-storage protocol with custom-defined SCSI RBC set command.

(4) **Propose and implement a new protocol, Secure Transaction Sharing (STS))**

I proposed and implemented a new protocol, Secure Transaction Sharing (STS). In our statement setting, the Bank encrypts data locally to ensure privacy. Moreover the Bank shares the
authentication access to cloud provider as out of band and other authentication channel access. To facilitate fine-grained access control, the asset of attributes is associated with each client’s account transaction, which helps to control client access to a specific set of account fields for each authorized user. The Bank issues a decryption key tamper proof for each authorized client account to protect the authentication service against attack; the Secure Transaction Sharing (STS) Authentication Mechanism has been proposed. Two different scenarios were considered: in the first scenario, we assumed that the attacker cannot spoof source addresses. A threshold is set for the number of failed authentication attempts which we assume is distributed as part of the security policy. In the first scenario (non-encrypted scenario), when STS is used as a mitigation mechanism delivery ratio improves by 81.1% or 9.5% compared to when no security mechanism is used respectively. The average latency of transfer when STS is used to combat attacks is 1272 seconds which is 7.1% of TTL compared to 2214 seconds or 12.3% of TLL when pillarer-1024 is used. In terms of overhead ratio, STS reduces the overhead ratio by 48.49% compared to when no security mechanism is adopted.

(5) Study and develop a trust Operating System

I proposed and implemented our O.S. by rebuilding the puppy Linux boot loader framework utilized to transfer measurement values from Application to Kernel. Several mechanisms were implemented for communicating between application and kernel, such as a grant table, ring buffers, an event channel, and Store. We use those mechanisms to transfer the measurement values from Application to Kernel. We implement two modules for inter control communication: a front-end module (FE) in an interface and a back-end module (BE) in Kernel. To transfer the measurement values, the Application first shares the ring buffers at operating system management (O.S.M), which transfer measurement values by using a grant table. The key function in this process is gnttab_grant_foreign_access.

(6) Development of a reference architecture Framework

A comprehensive survey of reference architecture framework and building model techniques in cloud computing and banking sector was provided. Modelling use in cloud architecture can be found in figure 6.13 chapter 6. Three architecture domains were used (cloud model, security control model and compliancy). The other side of the banking architecture mode as found in figure 6.12 chapter 6 covers four layers (client, common services, functional system and back end system. The design according to the target operating model requirements is consequently
Chapter 7 Conclusion and Future Work

represented within different structures of the architecture leading to delay in performance and unclear security approach.

While newly designed reference architecture was addressed in figures 6.14- 6.15 non-core domains have been outsourced and flexibility has been added to allow for a plug-and-play approach towards new distribution channels and products through Cloud integrated solutions. The performance percentage should go up because of worrying less about non-core banking software and hardware provisioning.

The proposed architecture model provides clear security, dedicated area against unwanted layers as shown in figure 6.15 in chapter 6, leading to increasing the efficiency and performance of the system. This means that fewer building blocks are required to apply architecture resulting in lower communication cost.

7.3 Future Work

Though the defined objectives set in section 1.1 have been met by the work presented in this thesis, some possibilities exist for future extension. The future direction regarding the extension of this research work is described below:

- There are still some avenues of attack left open. Obviously if an attacker can threaten the client directly, or deceive them sufficiently, the client may deliberately authorize a transaction to the attacker. There is only so far that technical solution can go to prevent such abuses and such attacks are difficult to control.

- Tamper proof device do nothing to keep the transaction log secret, the primary interface for transaction is still the computer, protecting against reading of the transaction log requires all interaction to be done through the trusted bath and this would significantly increase the cost.

- A complete access control system discussed in chapter 3 incorporating issues with limiting user access to unauthorized accounts, efficiently revoking users’ privileges without re-encrypting massive amounts of account data and re-distributing the new keys to the authorized users, collusion between users, and issuing changes to a user’s access privileges is still undergoing thorough research.

- Network control can be categorized into access control and admission control, and their individual implementation should not degrade the Quality of Service (QoS). It will be
interesting in future to see the possibility of having a common scheme that implements both access and admission controls.

- Cryptography is no more than another link in the chain. Most of the times, it is probably the strongest link, but there are many other weak links which decrease the security level until an unacceptable level for carrying out electronic commerce.

- The implementation of this work was based on authentication threading with focus on only overlay network entities. It will be interesting to exploit the multi-threading concept in future to see how the data mule can handle multiple requests from different sources to different destinations at the same time.
Bibliography


[5] Zakaria Karim1, Karim Mohammed Rezaul2, Aliar Hossain1. (2009) “Towards Secure Information Systems in Online Banking”, Applied Research Centre for Business and Information Technology (ARCBIT) 1, Guildhall College Copyright © by the Institute of Electrical and Electronics Engineers, Inc. All rights reserved.


[12] New McAfee Study Highlights Dangerous Internet Sites That Prey on Mistyped Web
Addresses


[25] A. Hisamatsu, D. Pishva, and G.G.D. Nishantha Ritsumeikan (2011). “Online Banking and Modern Approaches Toward its Enhanced Security Asia Pacific University”, Beppu City, Japan aimahi06@apu.ac.jp, dpishva@apu.ac.jp, HYPERLINK "mailto:gamagelk@apu.ac.jp” gamagelk@apu.ac.jp


[44] IBM Tivoli security solution (October 2005) Federated identity Management & web services security


France.


[53] John Chirillo, “Hack Attacks Revealed, “A Complete Reference with Custom Security Hacking Toolkit”, 8400, fax (978) 750-4744. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 605 Third Avenue, New York, NY 10158-0012, (212) 850-6011, fax (212) 850-6008, E-mail: permreq @ wiley.com.


[57] Tim Mather, and ShahedLatif, (2009) “the Age of Cloud Security and Privacy” Copyright and Shahed Latif. All rights reserved.

[58] Charles M. Firestone Executive Director Patricia K. Kelly Assistant Director (2011)
“Identity in the Age of Cloud Computing”: The next-generation Internet’s impact on business, governance and social interaction Copyright by the Aspen Institute.


[73] Craig Gentry , (September 2009): “A fully homomorphic encryption scheme”, a dissertation submitted to the department of computer science and the committee on graduate studies of Stanford university in partial fulfilment of the requirements for the degree of doctor of philosophy


R. Wang, W. Du and P. Ning, “Contain“Improving Additive and Multiplicative Homomorphic Encryption Schemes Based on Worst-Case Hardness Assumptions” Carlos Aguilar Melchor , Slim Bettaieb , Philippe Gaborit Universit_e de Limoges, 123, av. Albert Thomas 87060 Limoges Cedex, France

Laboratories of America {rchow,p golle,mjakobss,eshi,staddon - CCSW’09, November 13, 2009, Chicago, Illinois, USA.


[102] Anonymous hackers jailed over PayPal attack 24 January 2013, phy.org

[103] Guodong Li, Hu Chen, (2010). “A New High-Level Security Portable System Based on USB Key with Fingerprint”, International Conference On Computer Design And Applications (ICCDA 2010), School of Computer Science and Engineering, South China University of Technology , Guangzhou, China lyguodong@gmail.com


[105] Install "Puppy Linux” as the only operating system in either a real computer or a virtual machine.


Appendix A

Public Cloud

In this system we check only tamper proof ID and fingerprint; if it is right then it sends all variables to private cloud.

In public cloud the tamper proof ID and fingerprint is checked then these four values are sent to the private cloud (bank):

\[
\text{strKey} = \text{SendToPrivate(Tamper proofID, UserName, UserPassword, UserFingerPrint)}
\]

And the username is encrypted:

\[
\text{UserName} = \text{EncryptText(UserName, "SUCCESS")}
\]

The function that is used for encryption:

\[
\text{Function EncryptText(ByVal plaintext AsString, ByVal password AsString)}
\]

\[
\text{Dim wrapper AsNewSimple3Des(password)}
\]

\[
\text{Return wrapper.EncryptData(plaintext)}
\]

\[
\text{EndFunction}
\]

The function that checks the tamper proof ID:

\[
\text{Function isClient(ByVal Tamper proofID AsString) AsInteger}
\]

\[
\text{Dim tblName AsString = "tblConfirmTamper proofID"}
\]

\[
\text{Dim ClientID AsInteger = 0}
\]

\[
\text{Dim OleCon AsNewSqlConnection}
\]

\[
\text{Dim OleTableE AsSqlDataAdapter}
\]

\[
\text{Dim OleDbSE AsNewDataSet}
\]

\[
\text{Dim strSQL AsString = "SELECT Tamper proofID FROM tblConfirmTamper proofID WHERE}
\]

\[
\text{Tamper proofID="" & Tamper proofID & ""}
\]

Try

\[
\text{OleCon.ConnectionString = "Data}
\]

\[
\text{Source=\:\:SQL\:\:EXPRESSION;AttachDbFilename=|DataDirectory|PublicCloud.mdf;Integrated}
\]

\[
\text{Security=True;User Instance=True"}
\]

OleCon.Open()

\[
\text{OleTableE = NewSqlDataAdapter(strSQL, OleCon)}
\]

OleTableE.Fill(OleDbSE, tblName)

ClientID = OleDbSE.Tables(tblName).Rows(0).Item(0)

OleDbSE.Clear()

OleDbSE.Dispose()

OleTableE.Dispose()

OleCon.Close()

\text{Catch ex AsException}

EndTry

\text{Return ClientID}
And the function that sends these values to private:

```vbscript
Function SendToPrivate(ByVal Tamper proofID AsString, ByVal UserName AsString, ByVal UserPassword AsString, ByVal UserFingerPrint AsString) AsString
    Dim wData AsString
    Dim strPost AsString
    Dim ServerURL AsString
    Dim IsSuccess AsBoolean = False
    UserName = EncryptText(UserName, "SUCCESS")
    strPost = "Tamper proofID=" & Tamper proofID
    strPost = strPost & "&UserName=" & UserName
    strPost = strPost & "&UserPassword=" & UserPassword
    strPost = strPost & "&UserFingerPrint=" & UserFingerPrint
    ServerURL = "http://192.168.1.199:8064/TempConvert.asmx/ClientKey"
    wData = WRequest(ServerURL, "POST", strPost)
    Return wData
EndFunction
```

If Tamper proof ID is wrong then a message Tamper proof ID is wrong is displayed and the application stops here.

If private return value does not fail then it sends these values to the bank and also the username is encrypted:

```vbscript
UserName = EncryptText(UserName, "SUCCESS")
```

The other three values are encrypted by a random key generated in the private provider:
Dim encTamper proofID = EncryptText(Tamper proofID, strKey)
Dim encUserPassword = EncryptText(UserPassword, strKey)
Dim encUserFingerPrint = EncryptText(UserFingerPrint, strKey)
SendToBank(encTamper proofID, UserName, encUserPassword, encUserFingerPrint)

The function that sends these values to the bank is:

Sub SendToBank(ByVal Tamper proofID AsString, ByVal UserName AsString, ByVal UserPassword AsString, ByVal UserFingerPrint AsString)
Dim strPost AsString
Dim ServerURL AsString
Dim IsSuccess AsBoolean = False
UserName = EncryptText(UserName, "SUCCESS")
strPost = "Tamper proofID=" & Tamper proofID
strPost = strPost & "&UserName=" & UserName
strPost = strPost & "&UserPassword=" & UserPassword
strPost = strPost & "&UserFingerPrint=" & UserFingerPrint
ServerURL = "http://192.168.1.199:8067"
EndSub

In private provider it checks if these four variables match or not, by decrypting the username to match these values

decUserName = DecryptText(UserName, "SUCCESS")

The function that is used for decryption:

Function DecryptText(ByVal encryptedtext AsString, ByVal password AsString)
Dim wrapper AsNewSimple3Des(password)
Return wrapper.DecryptData(encryptedtext)
EndFunction

If there is no matching found then the application stops here and the client receives a message that the username or password is wrong
Also in private there was a random key generated for encryption and this key was sent to the bank
and to the client if these values are matching.

strKey = GetRandomKey(lengthKey)

This key takes random length (each time it has a different length between 10 and 30):

```
Dim r As New Random
Dim lengthKey As Integer = r.Next(10, 30)
```

The function to generate a random key:

```
Function GetRandomKey(ByVal KeyLength As Integer)
    Dim seconds = DatePart("s", Now)
    Dim strKeys AsString = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
    Dim r As New Random
    Dim sb As New StringBuilder
    Dim idx As Integer = r.Next(0, strKeys.Length)
    For i As Integer = 1 To KeyLength
        idx = r.Next(0, strKeys.Length)
        idx = Math.Truncate(1 + Math.Sqrt(idx * seconds / 2))
        sb.Append(strKeys.Substring(idx, 1))
    Next
    Return sb.ToString()
EndFunction
```
The function that checked these values:

```vbnet
<WebMethod()> Public Function ClientKey(ByVal Tamper proofID As String, ByVal UserName As String, ByVal UserPassword As String, ByVal UserFingerPrint As String) As String
Dim tblName As String = "tblConfirmClient"
Dim tblNameKey As String = "tblClients"
Dim blnSuccess As Boolean = False
Dim ClientID As Integer = 0
Dim strKey As String = "fail"
Dim OleCon As New SqlConnection
Dim OleConKey As New SqlConnection
Dim OleTableE As SqlDataAdapter
Dim OleTableEKey As SqlDataAdapter
Dim OleDSE As New DataSet
Dim OleDSEKey As New DataSet
Dim decUserName As String

    decUserName = DecrpytText(UserName, "SUCCESS")

Dim strSQL As String = "SELECT Tamper proofID,UserName,UserPassword,UserFingerPrint FROM tblConfirmClient WHERE Tamper proofID=" & Tamper proofID & "; AND UserName=" & decUserName & "; AND UserPassword=" & UserPassword & "; AND UserFingerPrint=" & UserFingerPrint & ";"

Try
    OleCon.ConnectionString = "server=OMECO-ERP1-PC\BSSERVER;Initial Catalog=BankDB;Integrated Security=True;"
    OleCon.Open()
    OleTableE = New SqlDataAdapter(strSQL, OleCon)
    OleTableE.Fill(OleDSE, tblName)
    ClientID = OleDSE.Tables(tblName).Rows(0).Item(0)
    If ClientID > 0 Then
        Dim r As New Random
        Dim lengthKey As Integer = r.Next(10, 30)
        strKey = GetRandomKey(lengthKey)
        Dim strKeySQL As String = "UPDATE tblClients SET UserKey=" & strKey & "; WHERE ClientID = 1"
        OleConKey.ConnectionString = "server=OMECO-ERP1-PC\BSSERVER;Initial Catalog=BankDB;Integrated Security=True;"
        OleConKey.Open()
        OleTableEKey = New SqlDataAdapter(strKeySQL, OleConKey)
        OleTableEKey.Fill(OleDbSEKey, tblNameKey)
        OleDbSEKey.Clear()
        OleDbSEKey.Dispose()
        OleDbTableEKey.Dispose()
        OleDbConKey.Close()
        OleDbConKey.Dispose()
    End If
End Try
Catch ex As Exception
    'MsgBox(ex.Message)
End Try
Else
```

173
strKey = "fail"
EndIf
OleDSE.Clear()
OleDSE.Dispose()
OleTableE.Dispose()
OleCon.Close()
OleCon.Dispose()
Catch ex As Exception
' MsgBox(ex.Message)
EndTry
Return strKey
EndFunction

The bank checked these values by comparing them to database values when decrypting these
values using a key generated in private.

decKeyPassword = DecryptText(UserPassword, strUserKey)
decKeyTamper proofID = DecryptText(Tamper proofID, strUserKey)
decKeyUserFingerPrint = DecryptText(UserFingerPrint, strUserKey)
If these values are matching then the client can access his/her account.

The function that is used to check matching:

PublicFunction ClientLogin(ByVal Tamper proofID As String, ByVal decUserName As String,
ByVal UserPassword As String, ByVal UserFingerPrint As String) As String
Dim tblName As String = "tblClients"
Dim ClientID As Integer = 0
Dim strKey As String = "fail"
Dim OleCon As New SqlConnection
Dim OleTableE As SqlDataAdapter
Dim OleDSE As New DataSet
Dim OleConKey As New SqlConnection
Dim OleTableEKey As SqlDataAdapter
Dim OleDSEKey As New DataSet
Dim strUserKey As String
Dim decKeyPassword As String
Dim decKeyTamper proofID As String, decKeyUserFingerPrint As String
Dim strKeySQL As String = "SELECT UserKey FROM tblClients WHERE ClientUserName="" &
decUserName & ""
Try
OleConKey.ConnectionString = "server=OMECO-ERP1-PC\BSSERVER;InitialCatalog=BankDB;Integrated Security=True;"
OleConKey.Open()
OleDbTableEKey = NewSqlDataAdapter(strKeySQL, OleConKey)
OleDbTableEKey.Fill(OleDbDSEKey, tblName)
strUserKey = OleDbDSEKey.Tables(tblName).Rows(0).Item(0)
decKeyPassword = DecryptText(UserPassword, strUserKey)
decKeyTamper proofID = DecryptText(Tamper proofID, strUserKey)
decKeyUserFingerPrint = DecryptText(UserFingerPrint, strUserKey)
OleDbDSEKey.Clear()
OleDbDSEKey.Dispose()
OleDbTableEKey.Dispose()
OleConKey.Close()
OleConKey.Dispose()

Catch ex As Exception
EndTry
Try
Dim strSQL As String
strSQL = "SELECT ClientID,ClientFirstName,ClientSecondName,ClientThirdName,ClientFourthName FROM tblClients WHERE ClientPassword="" & decKeyPassword & ""
Dim oleCon As New OleDbConnection
oleCon.ConnectionString = "server=OMECO-ERP1-PC\BSSERVER;InitialCatalog=BankDB;Integrated Security=True;"
oleCon.Open()
OleDbTableE = NewSqlDataAdapter(strSQL, oleCon)
OleDbTableE.Fill(OleDbDSE, tblName)
ClientID = OleDbDSE.Tables(tblName).Rows(0).Item(0)
ClientName = OleDbDSE.Tables(tblName).Rows(0).Item(1) & " " & OleDbDSE.Tables(tblName).Rows(0).Item(2) & " " & OleDbDSE.Tables(tblName).Rows(0).Item(3) & " " & OleDbDSE.Tables(tblName).Rows(0).Item(4)
If ClientID > 0 Then
    Session("ClientID") = ClientID
    Session("ClientName") = ClientName
    Session("uname") = decUserName
    Session("pass") = UserPassword
    Session.Add("acno", 1)
    Session("AccountID") = 0
Else
    strKey = "fail"
EndIf
OleDbDSE.Clear()
OleDbDSE.Dispose()
OleDbTableE.Dispose()
OleCon.Close()
OleCon.Dispose()
Catch ex As Exception
EndTry
Return strKey
EndFunction

Public cloud functions:

Client Function:
This function takes two parameters (Tamper proof ID, fingerprint); it checks if it is the client or not by comparing this value and the value from the database (tblConfirmTamper proofID, fingerprint).

```vbnet
Function IsClient(ByVal Tamper proofID As String) As Integer
    Dim tblName As String = "tblConfirmTamper proofID"
    Dim ClientID As Integer = 0
    Dim OleCon As New SqlConnection
    Dim OleTableE As SqlDataAdapter
    Dim OleDSE As New DataSet
    Dim strSQL As String = "SELECT Tamper proofID FROM tblConfirmTamper proofID WHERE Tamper proofID =" & Tamper proofID & ""
    Try
        OleCon.ConnectionString = "DataSource=.;SQLEXPRESS;AttachDbFilename=|DataDirectory|\PublicCloud.mdf;Integrated Security=True;User Instance=True"
        OleCon.Open()
        OleTableE = New SqlDataAdapter(strSQL, OleCon)
        OleTableE.Fill(OleDb, tblName)
        ClientID = OleDb.Tables(tblName).Rows(0).Item(0)
        ' Check if the value matches
    Catch ex As Exception
        OleDSE.Dispose()
        OleTableE.Dispose()
        OleCon.Close()
        OleCon.Dispose()
    End Try
    Return ClientID
End Function
```
OleTableE.Fill(OleDbSE, tblName)

ClientID = OleDbSE.Tables(tblName).Rows(0).Item(0)

ClientName = OleDbSE.Tables(tblName).Rows(0).Item(1)
&" "&
OleDbSE.Tables(tblName).Rows(0).Item(2)
&" "&
OleDbSE.Tables(tblName).Rows(0).Item(3)
&" "&
OleDbSE.Tables(tblName).Rows(0).Item(4)

OleDbSE.Clear()
OleDbSE.Dispose()
OleDbTableE.Dispose()
OleDbCon.Close()
OleDbCon.Dispose()

Catch ex As Exception
EndTry

Return ClientID

EndFunction

Send To Private – bank interface - Function:
This function takes four parameters (Tamper proofID, UserName, UserPassword and UserFingerPrint) and sends these values to private cloud.

Function SendToPrivate (ByVal DongleID As String, ByVal UserName As String,
ByVal UserPassword As String, ByVal UserFingerPrint As String) As String

Dim wData As String
Dim strPost As String
Dim ServerURL As String
Dim IsSuccess As Boolean = False

UserName = EncryptText (UserName, "hatim")
UserPassword = EncryptText (UserPassword, "hatim")
strPost = "DongleID=" & DongleID
strPost = strPost & "&UserName=" & UserName
strPost = strPost & "&UserPassword=" & UserPassword
strPost = strPost & "&UserFingerPrint=" & UserFingerPrint
ServerURL = "http://192.168.1.199:8064/TempConvert.asmx/ClientKey"
wData = WRequest(ServerURL, "POST", strPost)

Return wData

End Function
Function SendToPrivate (ByVal Tamper proofID As String, ByVal UserName As String, ByVal UserPassword As String, ByVal UserFingerPrint As String) As String

Dim wData As String
Dim strPost As String
Dim ServerURL As String
Dim IsSuccess As Boolean = False
UserName = EncryptText (UserName, "hatim")
UserPassword = EncryptText (UserPassword, "hatim")
strPost = "Tamper proofID=" & Tamper proofID
strPost = strPost & "&UserName=" & UserName
strPost = strPost & "&UserPassword=" & UserPassword
strPost = strPost & "&UserFingerPrint=" & UserFingerPrint
ServerURL = "http://192.168.1.199:8064/TempConvert.asmx/ClientKey"
wData = WRequest(ServerURL, "POST", strPost)
Return wData
End Function

Send To Bank – database- Function:
This function takes four parameters (Tamper proofID, UserName, UserPassword and UserFingerPrint) and sends these values to the bank.

Sub SendToBank (ByVal DongleID As String, ByVal UserName As String, ByVal UserPassword As String, ByVal UserFingerPrint As String)

Dim strPost As String
Dim ServerURL As String
Dim IsSuccess As Boolean = False
UserName = EncryptText (UserName, "hatim")
UserPassword = EncryptText (UserPassword, "hatim")
strPost = "DongleID=" & DongleID
strPost = strPost & "&UserName=" & UserName
strPost = strPost & "&UserPassword=" & UserPassword
strPost = strPost & "&UserFingerPrint=" & UserFingerPrint
ServerURL = "http://192.168.1.199:8067"
End Sub

Sub SendToBank (ByVal Tamper proofID As String, ByVal UserName As String, ByVal UserPassword As String, ByVal UserFingerPrint As String)
As String)
Dim strPost AsString
Dim ServerURL AsString
Dim IsSuccess AsBoolean = False
UserName = EncryptText(UserName, "hatim")
UserPassword = EncryptText(UserPassword, "hatim")
strPost = "Tamper proofID=" & Tamper proofID
strPost = strPost &"&UserName=" & UserName
strPost = strPost &"&UserPassword=" & UserPassword
strPost = strPost &"&UserFingerPrint=" & UserFingerPrint
ServerURL = http://192.168.1.199:8067
EndSub

Request function:
Send data via http protocol to the web service taking three parameters (URL, method and POSTdata)

Function WRequest(ByVal URL AsString, ByVal method AsString, ByVal POSTdata AsString) AsString
Dim responseData AsString = String.Empty
Try
Dim hwrequest AsHttpWebRequest = WebRequest.Create(URL)
    hwrequest.Accept = "*/*"
    hwrequest.AllowAutoRedirect = True
    hwrequest.UserAgent = "http_requester/0.1"
    hwrequest.Timeout = 600000
    hwrequest.Method = method
If hwrequest.Method = "POST" Then
    hwrequest.ContentType = "application/x-www-form-urlencoded"
Dim encoding AsNew ASCIIEncoding()
Dim postByteArray() AsByte = encoding.GetBytes(POSTdata)
    hwrequest.ContentLength = postByteArray.Length
Dim postStream As IO.Stream = hwrequest.GetRequestStream()
    postStream.Write(postByteArray, 0, postByteArray.Length)
    postStream.Close()
EndIf

Dim hwresponse As Net.HttpWebResponse = hwrequest.GetResponse()
If hwresponse.StatusCode = Net.HttpStatusCode.OK Then
    Dim responseStream As IO.StreamReader = New IO.StreamReader(hwresponse.GetResponseStream())
    responseData = responseStream.ReadToEnd()
EndIf

hwresponse.Close()

Catch e As Exception
    responseData = e.Message
EndTry

Return responseData

EndFunction

Button_Click (Button Confirm):
When pressing this button and if all data is correct, then the internet banking home page for this client will be displayed.

Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles btnConfirm.Click
    Dim ClientID As Integer = 0
    Dim DongleID As String
    Dim UserName As String
    Dim UserPassword As String
    Dim UserFingerPrint As String
    Dim strKey As String
    DongleID = txtDongleID.Text
    UserName = txtUserName.Text
    UserPassword = txtUserPassword.Text
    UserFingerPrint = txtUserFingerPrint.Text
    ClientID = isClient(DongleID)
    If ClientID > 0 Then
        Session("DongleID") = DongleID
        strKey = SendToPrivate(DongleID, UserName, UserPassword, UserFingerPrint)
        If strKey.Contains("success") Then
            SendToBank(DongleID, UserName, UserPassword, UserFingerPrint)
        Else
            labMessage.Text = "UserName or Password is Wrong!!!"
        End If
    Else
        labMessage.Text = "Dongle ID is Wrong!!!"
    End If
End Sub
Dim UserName AsString
Dim UserPassword AsString
Dim UserFingerPrint AsString
Dim strKey AsString
Tamper proofID = txtTamper proofID.Text
UserName = txtUserName.Text
UserPassword = txtUserPassword.Text
UserFingerPrint = txtUserFingerPrint.Text
ClientID = isClient(Tamper proofID)
If ClientID > 0 Then
    Session("Tamper proofID") = Tamper proofID
    strKey = SendToPrivate(Tamper proofID, UserName, UserPassword,
        UserFingerPrint)
    If strKey.Contains("success") Then
        SendToBank(Tamper proofID, UserName, UserPassword,
            UserFingerPrint)
    Else
        labMessage.Text = "UserName or Password is Wrong!!!"
    EndIf
Else
    labMessage.Text = "Tamper proof ID is Wrong!!!"
EndIf
EndSub

<WebMethod()> PublicFunction ClientKey(ByVal Tamper proofID AsString, ByVal
UserName AsString, ByVal
UserPassword As String, ByVal UserFingerPrint AsString) AsString
Dim tblName AsString = "tblConfirmClient"
Dim ClientID AsInteger = 0
Dim strKey AsString = "fail"
Dim OleCon AsNewSqlConnection
Dim OleTableE AsSqlDataAdapter
Dim OleDSE AsNewDataSet
Dim decUserName AsString, decPassword AsString
    decUserName = DecryptText(UserName, "hatim")
decPassword = DecryptText(UserPassword, "hatim")

Dim strSQL As String = "SELECT Tamper proofID,UserName,UserPassword,UserFingerPrint
FROM tblConfirmClient WHERE Tamper proofID="" & Tamper proofID & "" AND
UserName="" & decUserName & "" AND UserPassword="" & decPassword & "" AND
UserFingerPrint="" & UserFingerPrint & ""

Try
    OleCon.ConnectionString = "Data Source=\SQLEXPRESS;AttachDbFilename=|DataDirectory|\basic.mdf;Integrated Security=True;User Instance=True"
    OleCon.Open()
    OleTableE = NewSqlDataAdapter(strSQL, OleCon)
    OleTableE.Fill(OleDSE, tblName)
    ClientID = OleDSE.Tables(tblName).Rows(0).Item(0)
If ClientID > 0 Then
    strKey = "success"
Else
    strKey = "fail"
EndIf
    OleDSE.Clear()
    OleDSE.Dispose()
    OleTableE.Dispose()
    OleCon.Close()
    OleCon.Dispose()
Catch ex AsException
    'MsgBox(ex.Message)
EndTry
Return strKey
EndFunction

Internet Banking

Client Login Function:

This function receives four variables and checks these values by comparing them to database values if matched return is a success.

PublicFunction ClientLogin(ByVal Tamper proofID AsString, ByVal decUserName AsString,
ByVal decPassword AsString, ByVal UserFingerPrint AsString) AsString
Appendix A.

Dim tblName AsString = "tblClients"
Dim ClientID AsInteger = 0
Dim strKey AsString = "fail"
Dim OleCon AsNewSqlConnection
Dim OleTableE AsSqlDataAdapter
Dim OleDSE AsNewDataSet
Dim strSQL AsString = "SELECT ClientID,ClientFirstName,ClientSecondName,ClientThirdName,ClientFourthName FROM tblClients WHERE ClientUserName='" & decUserName & "' AND ClientPassword='" & decPassword & "'"

Try
    OleCon.ConnectionString = "Data Source=\SQL\SQLEXPRESS;AttachDbFilename=|DataDirectory|\Database.mdf;Integrated Security=True;User Instance=True"
    OleCon.Open()
    OleTableE = NewSqlDataAdapter(strSQL, OleCon)
    OleTableE.Fill(OleDSE, tblName)
    ClientID = OleDSE.Tables(tblName).Rows(0).Item(0)
    ClientName = OleDSE.Tables(tblName).Rows(0).Item(1) & " " & OleDSE.Tables(tblName).Rows(0).Item(2) & " " & OleDSE.Tables(tblName).Rows(0).Item(3) & " & OleDSE.Tables(tblName).Rows(0).Item(4)
    If ClientID > 0 Then
        Session("ClientID") = ClientID
        Session("ClientName") = ClientName
        Session("uname") = decUserName
        Session("pass") = decPassword
        Session.Add("acno", 1)
        Session("AccountID") = 0
    Else
        strKey = "fail"
    EndIf
EndTry

OleDSE.Clear()
OleDSE.Dispose()
OleTableE.Dispose()
OleCon.Close()
OleCon.Dispose()

Catch ex As Exception
'ClientName = ex.Message
'OleCon.Close()
EndTry

Return strKey

EndFunction
Appendix B

Result Testing Server Security

Information Gathering

- Spider/crawl for missed or hidden content:
  
  There is no any information valuable for any hacker when used a spider/crawl script in our programmers.

- Check the Webserver Metafiles for information leakage files that expose content, such as robots.txt, sitemap.xml, .DS_Store:
  
  This page is not available, sensitive metafile information to be exposed.
Check the caches of major search engines for publicly accessible sites:
Google cash check to remove unnecessary information
Perform Web Application Fingerprinting:
Used secure channel with pattern encrypted.

- Identify user roles:
  User right for any clients and admin from local network inside the bank only
Appendix B.

- **Identify application entry points:**
  Logging page is highly secure with multi authentication factor and more than two channel way with mutual direction

- **Identify client-side code:**
  Token care for source code and SSL

- **Identify multiple versions/channels (e.g. web, mobile web, mobile app, web services):**
  Web services limited used to the system only, there is no direct access to bank server, all services complete after authentication.

- **Identify co-hosted and related applications:**
  There no any co-hosted code or application

- **Identify third-party hosted content:**
  Our code is isolated completely

**Configuration Management**

- **Test for security HTTP headers (e.g. CSP, X-Frame-Options, HSTS):**
  X-Frame-Options
  This header can be used to prevent Click Jacking in modern browsers.
  Use the same-origin attribute to allow being framed from urls of the same origin or deny blocking all. Example: X-Frame-Options: DENY

**Authentication**

**Authentication**

Is the process of verification that an individual or an entity is who it claims to be. Authentication is commonly performed by submitting a user name or ID and one or more items of private information that only a given user should know.

**Wireshark**

Is a network sniffer or protocol analyzer as a software application or hardware device which is capable of intercepting traffic and logging it for further analysis. A packet sniffer is very crucial for network analysis as well as troubleshooting. This log can be analyzed to:

- Study the interaction of different machines.
- Analyze the packets that are passing by.
- Learn more about the underlying protocols.
- Detect the flaws in the network.
- Examine security issues.

Let’s dive deep into this fantastic tool and understand some of its features:
After installing the application and starting it, the first thing to do is to choose the Interface(s) to start with. Interface list displays all the interfaces present on the machine so we can choose the one(s) we want to listen on. Before starting the capture on the network, we should also specify whether we want to capture packets in promiscuous mode or not. Promiscuous mode if enabled (enabled by default) allows Wireshark to capture all the packets it can over the network, else only packets to and from the machine running Wireshark will be captured. We can decide on this function from the options button in the Capture Interfaces list and start the process of capturing the packets. Now based on the amount of network traffic, the packets will be captured and listed on the interface in real time for analysis.

![Wireshark Interface List and Capture Options](image.png)

Figure b.1: shows the interface list and the options to start the capture.

First thing you’ll notice from this packet capture is that Wireshark uses different color coding for different packets. This color coding helps us to distinguish between different packet types and hence is useful for quick overview. For example, black foreground on green background is used to display HTTP packets while the reverse is used to indicate ICMP errors. These color coding combinations can be customized according to personal preference.
We have seen how Wireshark captures packets in real time and displays them on the interface; now let’s see how to filter these packets.

Packet filtering is a very essential feature. We can see that during the capture, there are various kinds of packets (protocols) that are captured and we need to focus on some specific packets. Wireshark allows traffic filtering based on different filters, which can be specified before as well as after the capture. We can simply input the protocol name in the filter bar and press Enter to see the packets of that specific protocol on the interface with the rest all removed. For example, if we need only HTTP traffic on the interface, we can simply input ‘http’ (without quotes) into the filter box and get the result. Wireshark also supports advanced filters which include expressions, IP address, MAC address, port number etc.

Some of the example filters are as following:

Ethernet broadcast: eth.addr==ff:ff:ff:ff:ff

IP address 212.0.146.122: ip.addr==192.168.0.1

UDP only: udp

No ARP and no DNS: not arp and!(udp.port==53)
Figure b.3 shows the packets being listed according to the applied filter.

All these filters are built-in to the application and can be accessed by clicking on the filter button. But this is not all: Wireshark also allows users to create custom filters and add them to this list and use them in future. This task can simply be accomplished by clicking on the ‘New’ button in the filter list and specify the filter name and filter string. For example, we can create a filter capturing packet from a specific IP range:

IP Range 192.168.0.0/24: ip.addr==192.168.0.0/24

The top block of the interface shows all the packets captured based on the filter applied, the middle block consists of all the detailed information regarding the packet selected in the top block, and the lowest block displays the hexdump of the selected packet. To analyze a packet, we simply need to select it in the top block and browse through the information in the middle block like the IP address, MAC address and other details related to the protocol structure. The information we selected in the middle block is displayed with the corresponding presence in the hexdump.

Let’s analyze a simple packet now. ARP filter has been applied so that the packets displayed are ARP packets only. We have selected an ARP request packet in the top block; an ARP request packet is broadcasted with the IP address that needs to be resolved with the MAC address. Now we can browse through all the information in the middle block like Sender MAC address, Sender IP address, Target IP address, protocol size etc. Figure 4 shows the ARP packet details.
Wireshark also allows seeing the communication between two machines as a stream. We can simply select a packet and right click on it, and then based on the protocol, we can select the option to follow the stream and view the exact communication. This feature is quite useful if we need to understand the communication between two machines or if we are looking for some specific information (e.g. password) that is being passed between them. The text below shows a sample TCP stream.

```
GET / HTTP/1.1
Host: 212.0.146.122
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.11 (KHTML, like gecko) Chrome/23.0.1271.95 Safari/537.11
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
DNT: 1

HTTP/1.1 200 OK
Date: Wed, 08 Oct 2014 08:10:29 GMT
```

Figure b.4. ARP packet analysis
Appendix B.

Server: Apache/2.2.14 (Ubuntu)
Last-Modified: Tue, 14 Oct 2014 07:45:00 GMT
ETag: “2c58a-b1-d2a22183700”
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 146
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html

HTTP/1.1 404 Not Found
Date: Wed, 08 Oct 2014 08:10:29 GMT
Server: Apache/2.2.14 (Ubuntu)
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 240
Keep-Alive: timeout=15, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1
Using Wireshark, we can create ACL (Access Control List) rules for different firewalls including Cisco IOS, IP Filter, NetFilter, Windows Firewall, etc. These rules can be applied to the appropriate firewall interface. To utilize this feature, we simply need to navigate to the Tools menu and select the Firewall ACL Rules options.

Figure shows the firewall ACL rules option present and Lua console in Wireshark.

Wireshark provides support for Lua. Lua is a lightweight yet powerful programming language. It can be used to write dissectors and taps.

Wireshark also allows saving the captured packets into various formats which can be utilized for later analysis by Wireshark or any other packet analysis applications. Similarly, Wireshark is also capable of reading packets from various different format packet captures.

Apart from the GUI interface, we can also utilize the power of this tool through the command line version ‘tshark’. Below is the listing of the options provided by tshark:
Appendix B.
Session Management

Is a process by which a server maintains the state of an entity interacting with it. This is required for a server to remember how to react to subsequent requests throughout a transaction. Sessions
are maintained on the server by a session identifier which can be passed back and forward between the client and server when transmitting and receiving requests. Sessions should be unique per user and computationally very difficult to predict.

**Test for user enumeration:**

Make sure your usernames/user ids are case insensitive. Many sites use email addresses for usernames and email addresses are already case insensitive. Regardless, it would be very strange for user 'smith' and user 'Smith' to be different users.

- **Test for brute force protection:**

  If an attacker is able to guess passwords without the account becoming disabled due to failed authentication attempts, the attacker has an opportunity to continue with a brute force attack until the account is compromised. Automating brute-force/password guessing attacks on web applications is a trivial challenge. Password lockout mechanisms should be employed that lock out an account if more than a preset number of unsuccessful login attempts are made. Password lockout mechanisms have a logical weakness. An attacker that undertakes a large number of authentication attempts on known account names can produce a result that locks out entire blocks of user accounts. Given that the intent of a password lockout system is to protect from brute-force attacks, a sensible strategy is to lockout accounts for a period of time (e.g., 20 minutes). This significantly slows down attackers, while allowing the accounts to reopen automatically for legitimate users.
Also, multi-factor authentication is a very powerful deterrent when trying to prevent brute force attacks since the credentials is a moving target. When multi-factor is implemented and active, account lockout may no longer be necessary.

- **Test for Credentials Transported over an Encrypted Channel**

- **Test password quality rules:**

  A key concern when using passwords for authentication is password strength. A "strong" password policy makes it difficult or even improbable for one to guess the password through either manual or automated means. The following characteristics define a strong password:

  - **Password Length**
    - Longer passwords provide a greater combination of characters and consequently make it more difficult for an attacker to guess.
    - Minimum length of the passwords should be enforced by the application.
      - Passwords shorter than 10 characters are considered to be weak ([1]).
    - While minimum length enforcement may cause problems with memorizing passwords among some users, applications should encourage them to set passphrases (sentences or combination of words) that can be much longer than typical passwords and yet much easier to remember.
    - Maximum password length should not be set too low, as it will prevent users from creating passphrases. Typical maximum length is 128 characters.
      - Passphrases shorter than 20 characters are usually considered weak if they only consist of lower case Latin characters.
    - Every character counts!!
      - Make sure that every character the user types in is actually included in the password. We’ve seen systems that truncate the password at a length shorter than what the user provided (e.g., truncated at 15 characters when they entered 20).
      - This is usually handled by setting the length of ALL password input fields to be exactly the same length as the maximum length password. This is particularly important if your max password length is short, like 20-30 characters.
• Test remember me functionality

Test for autocomplete on password forms/input:

Password mechanisms should allow virtually any character the user can type to be part of their password, including the space character. Passwords should, obviously, be case sensitive in order to increase their complexity. Occasionally, we find systems where passwords aren’t case sensitive, frequently due to legacy system issues like old mainframes that didn’t have case sensitive passwords.

Test password reset and/or recovery

Test password change process:

The password change mechanism should require a minimum level of complexity that makes sense for the application and its user population. For example:

Password must meet at least 3 out of the following 4 complexity rules

at least 1 uppercase character (A-Z)

at least 1 lowercase character (a-z)

at least 1 digit (0-9)

at least 1 special character (punctuation) — do not forget to treat space as special characters too

at least 10 characters
at most 128 characters
not more than 2 identical characters in a row (e.g., 111 not allowed)

- **Test CAPTCHA**
  No CAPTCHA information (except the image itself) stored on the client side. The client has no "control" over the CAPTCHA content and CAPTCHA images always randomly generated without possibility to perform image preprocessing, segmentation and classification, finally CAPTCHA images not be reused.

- **Test multi factor authentication:**
  Used Multi-factor authentication (MFA) to logon or process a transaction:
  Implement One Time Passwords (OTP) as Authentication schemes, using a hardware token, also be key in fighting attacks such as CSRF and client-side malware.

- **Test for logout functionality presence**
• Test for user-accessible authentication history:

The login page and all subsequent authenticated pages exclusively accessed over TLS. The initial login page, referred to as the "login page", served over SSL and TLS.

Session Management
Establish how session management is handled in the application (eg, tokens in cookies, token in URL):

Many implementations of this control include the challenge token in GET (URL) requests as well as POST requests. This is often implemented as a result of sensitive server-side operations being invoked as a result of embedded links in the page or other general design patterns. These patterns are often implemented without knowledge of CSRF and an understanding of CSRF prevention design strategies. While this control does help mitigate the risk of CSRF attacks, the unique per-session token is being exposed for GET requests. CSRF tokens in GET requests are potentially leaked at several locations: browser history, HTTP log files, network appliances that make a point to log the first line of an HTTP request, and Referer headers if the protected site links to an external site.

In the latter case (leaked CSRF token due to the Referer header being parsed by a linked site), it is trivially easy for the linked site to launch a CSRF attack on the protected site, and they will be able to target this attack very effectively, since the Referer header tells them the site as well as the CSRF token. The attack could be run entirely from javascript, so that a simple addition of a script tag to the HTML of a site can launch an attack (whether on an originally malicious site or on a
hacked site). This attack scenario is easy to prevent, the referer will be omitted if the origin of the request is HTTPS. Therefore this attack does not affect web applications that are HTTPS only.

The ideal solution implemented in our solution is to only include the CSRF token in POST requests and modify server-side actions that have state changing affect to only respond to POST requests. This is in fact what the RFC 2616 requires for GET requests. If server-side actions are guaranteed to only ever respond to POST requests, then there is no need to include the token in GET requests.

In most JavaEE web applications, however, HTTP method scoping is rarely ever utilized when retrieving HTTP parameters from a request. Calls to "HttpServletRequest.getParameter" will return a parameter value regardless if it was a GET or POST. This is not to say HTTP method scoping cannot be enforced. It can be achieved if a developer explicitly overrides doPost() in the HttpServlet class or leverages framework specific capabilities such as the AbstractFormController class in Spring.

For these cases, attempting to retrofit this pattern in existing applications requires significant development time and cost, and as a temporary measure it may be better to pass CSRF tokens in the URL. Once the application has been fixed to respond to HTTP GET and POST verbs correctly, CSRF tokens for GET requests should be turned off.

ZK token used for authentication

- **Check session tokens for cookie flags (httpOnly and secure):**
  
  All cookies, even the secret ones, will be submitted with every request. All authentication tokens will be submitted regardless of whether or not the end-user was tricked into submitting the request. Furthermore, session identifiers are simply used by the application container to associate the request with a specific session object. The session identifier does not verify that the end-user intended to submit the request.

Data Validation

- **Test for authentication**
  
  Use security authentication and security layer (SASL) for application and SMTP AUTH.
First you will need to install the `libsasl2-2`, `sasl2-bin` and `libsasl2-modules` from the Main repository [i.e. `sudo apt-get install them all`].

**Note:** when using Ubuntu 6.06 (Dapper Drake) the package name is `libsasl2`. We have to change a few things to make it work properly. Because Postfix runs chrooted in `/var/spool/postfix` we have change a couple paths to live in the false root. (ie. `/var/run/saslauthd` becomes `/var/spool/postfix/var/run/saslauthd`):

**Note:** by changing the `saslauthd` path other applications that use saslauthd may be affected.

First we edit `/etc/default/saslauthd` in order to activate saslauthd. Remove # in front of `START=yes`, add the `PWDIR`, `PARAMS`, and `PIDFILE` lines and edit the `OPTIONS` line at the end:

```bash
START=yes
PWDIR="/var/spool/postfix/var/run/saslauthd"
PARAMS="-m ${PWDIR}"
PIDFILE="${PWDIR}/saslauthd.pid"

# You must specify the authentication mechanisms you wish to use.
# This defaults to "pam" for PAM support, but may also include
# "shadow" or "sasldb", like this:
# MECHANISMS="pam shadow"

MECHANISMS="pam"

# Other options (default: -c)
# See the saslauthd man page for information about these options.
#
# Example for postfix users: ":-c -m /var/spool/postfix/var/run/saslauthd"
# Note: See /usr/share/doc/sasl2-bin/README.Debian
#OPTIONS="-c"

#make sure you set the options here otherwise it ignores params above and will not work
OPTIONS="-c -m /var/spool/postfix/var/run/saslauthd"
```

**Note:** If you prefer, you can use "shadow" instead of "pam". This will use MD5 hashed password transfer and is perfectly secure. The username and password needed to authenticate will be those of the users on the system you are using on the server.

Next, we update the dpkg "state" of `/var/spool/postfix/var/run/saslauthd`. The saslauthd init script uses this setting to create the missing directory with the appropriate permissions and ownership:

```
sudo dpkg-statoverride --force --update --add root sasl 755 /var/spool/postfix/var/run/saslauthd
```

This may report an error that "--update given" and the "/var/spool/postfix/var/run/saslauthd" directory does not exist. You can ignore this because when you start saslauthd next it will be created.

Finally, start saslauthd:

```
sudo /etc/init.d/saslauthd start
```

Testing
To see if SMTP-AUTH and TLS work properly now run the following command:

telnet localhost 25

After you have established the connection to your postfix mail server type

ehlo localhost

If you see the lines

```
250-STARTTLS
250-AUTH
```

among others, everything is working.

Type `quit` returning to the system's shell.

**Configuring saslauthd to Default**

If you don't want to run Postfix in a chroot, or you'd like to not use chroot for troubleshooting purposes you will probably also want to return `saslauthd` back to its default configuration.

The first step in accomplishing this is to edit `/etc/default/saslauthd` comment the following lines we added above:

```
#PWDIR="/var/spool/postfix/var/run/saslauthd"
#PARAMS="-m ${PWDIR}"
#PIDFILE="${PWDIR}/saslauthd.pid"
```

Then return the saslauthd dpkg "state" to its default location:

```
dpkg-statoverride --force --update --add root sasl 755 /var/run/saslauthd
```

And restart saslauthd:

```
sudo /etc/init.d/saslauthd restart
```

**Using Port 587 for Secure Submission**

If you want to use port 587 as the submission port for SMTP mail rather than 25 (many ISPs block port 25), you will need to edit `/etc/postfix/master.cf` and uncomment the line

```
Submission inet n   -n
```
Test for SSI Injection:

SSL Client Authentication, also known as two-way SSL authentication, consists of both, browser and server, sending their respective SSL certificates during the TLS handshake process. Just as you can validate the authenticity of a server by using the certificate and asking a well-known Certificate Authority (CA) if the certificate is valid, the server can authenticate the user by receiving a certificate from the client and validating against a third party CA or its own CA. To do this, the server must provide the user with a certificate generated specifically for him, assigning values to the subject so that these can be used to determine what user the certificate should validate. The user installs the certificate on a browser and now uses it for the website.

It is a good idea to do this when:

- It is acceptable (or even preferred) that the user only has access to the website from only a single computer/browser.
- The user is not easily scared by the process of installing SSL certificates on his browser or there will be someone, probably from IT support that will do this for the user.
- The website requires an extra step of security.

It is also a good thing to use when the website is for an intranet of a company or organization.

It is generally not a good idea to use this method for widely and publicly available websites that will have an average user. For example, it wouldn't be a good idea to implement this for a website
like Facebook. While this technique can prevent the user from having to type a password (thus protecting against an average keylogger from stealing it), it is still considered a good idea to consider using both a password and SSL client authentication combined

**Denial of Service**

- **Test for anti-automation**
  
  In our server we used mavis-new is a wrapper that can call any number of content filtering programs for spam detection, antivirus, etc.

  
  **Installation**

  To begin,

  ```
sudo apt-get install amavisd-new spamassassin clamav-daemon
  
  Install the optional packages for better spam detection (who does not want better spam detection?):
  ```

  ```
sudo apt-get install libnet-dns-perl libmail-spf-query-perl pyzor razor
  
  Note: Replace libmail-spf-query-perl with libmail-spf-perl
  
  Install these optional packages to enable better scanning of attached archive files:
  ```

  ```
sudo apt-get install arj bzip2 cabextract cpio file gzip lha nomarch pax rar unrar unzip unzoo zip zoo
  ```

  **Configuration**

  Clamav

  The default behaviour of Clamav will fit our needs. A daemon is launched (clamd) and signatures are fetched every day. For more Clamav configuration options, check the configuration files in `/etc/clamav`.

  Add clamav user to the amavis group and vice versa in order for Clamav to have access to scan files:

  ```
sudo adduser clamav amavis
sudo adduser amavis clamav
  ```

  **Risky Functionality - File Uploads**

  - **Test that acceptable file types are whitelisted**

    The basic concept behind application whitelisting is to permit only good known files to execute, rather than attempting to block malicious code and activity. Application whitelisting accomplishes its objectives by creating a list of approved hashes and allowing only files with approved hashes to execute. The general concept is quite simple, and in our application is applying.
• Test that file contents match the defined file type
  A ROOT file is a suite of consecutive data records (TKey's). If the key is located past the 32
  bit file limit (> 2 GB) then some fields will be 8 instead of 4 bytes:
  
  1->4 Nbytes = Length of compressed object (in bytes)
  5->6 Version = TKey version identifier
  7->10 ObjLen = Length of uncompressed object
  11->14 Datime = Date and time when object was written to file
  15->16 KeyLen = Length of the key structure (in bytes)
  17->18 Cycle = Cycle of key
  19->22 [19->26] SeekKey = Pointer to record itself (consistency check)
  23->26 [27->34] SeekPdir = Pointer to directory header
  27->27 [35->35] Iname = Number of bytes in the class name
  28->.. [36->..] ClassName = Object Class Name
  ..->.. Iname = Number of bytes in the object name
  ..->.. Name = lName bytes with the name of the object
  ..->.. lTitle = Number of bytes in the object title
  ..->.. Title = Title of the object
  -----> DATA = Data bytes associated to the object

  The first data record starts at byte fBEGIN (currently set to kBEGIN).
  Bytes 1->kBEGIN contain the file description, when fVersion >= 1000000
  it is a large file (> 2 GB) and the offsets will be 8 bytes long and fUnits will be set to 8:
  
  1->4 "root" = Root file identifier
  5->8 fVersion = File format version
  9->12 fBEGIN = Pointer to first data record
  13->16 [13->20] fEND = Pointer to first free word at the EOF
  17->20 [21->28] fSeekFree = Pointer to FREE data record
  21->24 [29->32] fNbytesFree = Number of bytes in FREE data record
  25->28 [33->36] nfree = Number of free data records
  29->32 [37->40] fNbytesName = Number of bytes in TNamed at creation time
  33->33 [41->41] fUnits = Number of bytes for file pointers
  34->37 [42->45] fCompress = Compression level and algorithm
  38->41 [46->53] fSeekInfo = Pointer to TStreamerInfo record
42-45 [54-57] fNbytesInfo = Number of bytes in TStreamerInfo record
46-63 [58-75] fUUID = Universal Unique ID
Appendix B.

Risky Functionality -

- Test for Authentication and Authorization issues

Test for CSRF:

Cross-Site Request Forgery (CSRF) is a type of attack that occurs when a malicious Web site, email, blog, instant message, or program causes a user's Web browser to perform an unwanted action on a trusted site for which the user is currently authenticated. The impact of a successful cross-site request forgery attack is limited to the capabilities exposed by the vulnerable application. For example, this attack could result in a transfer of funds, changing a password, or purchasing an item in the user's context. In effect, CSRF attacks are used by an attacker to make a target system perform a function (funds transfer, form submission etc.) via the target's browser without knowledge of the target user, at least until the unauthorized function has been committed.

Impacts of successful CSRF exploits vary greatly based on the role of the victim. When targeting a normal user, a successful CSRF attack can compromise end-user data and their associated functions. If the targeted end user is an administrator account, a CSRF attack can compromise the entire Web application. The sites that are more likely to be attacked are community Websites (social networking, email) or sites that have high dollar value accounts associated with them (banks, stock brokerages, bill pay services). This attack can happen even if the user is logged into a Web site using strong encryption (HTTPS). Utilizing social engineering, an attacker will embed malicious HTML or JavaScript code into an email or Website to request a specific 'task url'. The task then executes with or without the user's knowledge, either directly or by utilizing a Cross-site Scripting flaw (ex: Samy MySpace Worm).

HTML 5

- Test for Web Storage SQL injection

Check CORS implementation:

Cross Origin Resource Sharing

Validate URLs passed to XMLHttpRequest.open. Current browsers allow these URLs to be cross domain; this behavior can lead to code injection by a remote attacker. Pay extra attention to absolute URLs.

Ensure that URLs responding with Access-Control-Allow-Origin:

we do not include any sensitive content or information that might aid attacker in further attacks. We Use the Access-Control-Allow-Origin header only on chosen URLs that need to be accessed cross-domain. Don't use the header for the whole domain.

Allow only selected, trusted domains in the Access-Control-Allow-Origin header. Prefer whitelisting domains over blacklisting or allowing any domain (we do not use * wildcard nor blindly return the Origin header content without any checks).

Keep in mind that CORS does not prevent the requested data from going to an unauthenticated location. It's still important for the server to perform usual CSRF prevention.
While the RFC recommends a pre-flight request with the OPTIONS verb, current implementations might not perform this request, so it's important that "ordinary" (GET and POST) requests perform any access control necessary.

Our system discard requests received over plain HTTP with HTTPS origins to prevent mixed content bugs.

Don't rely only on the Origin header for Access Control checks. Browser always sends this header in CORS requests, but may be spoofed outside the browser. Application-level protocols should be used to protect sensitive data.

- Check Offline Web Application

Error Handling

- Check for Error Codes:
  
The application may return a different HTTP Error code depending on the authentication attempt response. It may respond with a 200 for a positive result and a 403 for a negative result. Even though a generic error page is shown to a user, the HTTP response code may differ which can leak information about whether the account is valid or not.
Hacking Packets Analysis

Figure b.7: NetworkMiner to parse libcap

Sent: 42 packets (15,278 Bytes), 0.00 % cleartext (0 of 0 Bytes)

192.168.1.199 (Windows) -> 192.168.1.197 (Windows) : 18 packets (3,190 Bytes), 0.00 % cleartext (0 of 0 Bytes)

TCP: 2639 -> 8063 : 12 packets (2,309 Bytes), 0.00 % cleartext (0 of 0 Bytes)
TCP: 2647 -> 8063 : 4 packets (172 Bytes), 0.00 % cleartext (0 of 0 Bytes)
TCP: 8055 -> 7003 : 2 packets (709 Bytes), 0.00 % cleartext (0 of 0 Bytes)

192.168.1.199 (Windows) -> 192.168.1.199 (Windows) : 14 packets (11,420 Bytes), 0.00 % cleartext (0 of 0 Bytes)

TCP: 2641 -> 8067 : 4 packets (776 Bytes), 0.00 % cleartext (0 of 0 Bytes)
TCP: 8067 -> 2641 : 10 packets (10,644 Bytes), 0.00 % cleartext (0 of 0 Bytes)
Figure b.8: Sent Packets
Figure b.9: sent packet traffic result

Received: 33 packets (27,839 Bytes), 0.00 % cleartext (0 of 0 Bytes)

192.168.1.197 (Windows) -> 192.168.1.199 (Windows) : 19 packets (16,419 Bytes), 0.00 % cleartext (0 of 0 Bytes)

TCP: 8063 -> 2639 : 13 packets (15,432 Bytes), 0.00 % cleartext (0 of 0 Bytes)

TCP: 8063 -> 2647 : 3 packets (132 Bytes), 0.00 % cleartext (0 of 0 Bytes)

TCP: 7003 -> 8055 : 3 packets (855 Bytes), 0.00 % cleartext (0 of 0 Bytes)

192.168.1.199 (Windows) -> 192.168.1.197 (Windows) : 14 packets (11,420 Bytes), 0.00 % cleartext (0 of 0 Bytes)

TCP: 2641 -> 8067 : 4 packets (776 Bytes), 0.00 % cleartext (0 of 0 Bytes)

TCP: 8067 -> 2641 : 10 packets (10,644 Bytes), 0.00 % cleartext (0 of 0 Bytes)
### Figure b.10: Received Packets

<table>
<thead>
<tr>
<th>Source IP</th>
<th>Destination IP</th>
<th>Protocol</th>
<th>Packets Sent</th>
<th>Packets Received</th>
<th>Size</th>
<th>Dropped</th>
<th>Missed</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.1.198</td>
<td>192.168.1.186</td>
<td>TCP</td>
<td>522</td>
<td>102</td>
<td>353</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>192.168.1.198</td>
<td>192.168.1.186</td>
<td>UDP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

### Figure b.11: received packet result
Incoming sessions: 2

Server: 192.168.1.199 (Windows) TCP 2639
Server: 192.168.1.199 (Windows) TCP 2639 (1829 data bytes sent), Client: 192.168.1.197 (Windows) TCP 8063 (14900 data bytes sent), Session start: 10/21/2014 3:00:36 PM, Session end: 10/21/2014 3:00:46 PM

Server: 192.168.1.199 (Windows) TCP 2641
Server: 192.168.1.199 (Windows) TCP 2641 (615 data bytes sent), Client: 192.168.1.199 (Windows) TCP 8067 (10220 data bytes sent), Session start: 10/21/2014 3:00:36 PM, Session end: 10/21/2014 3:00:47 PM

Figure b.12: Incoming sessions

Outgoing sessions: 3

Server: 192.168.1.197 (Windows) TCP 7003
Appendix B.

Server: 192.168.1.197 (Windows) TCP 7003 (735 data bytes sent), Client: 192.168.1.199 (Windows) TCP 8055 (629 data bytes sent), Session start: 10/21/2014 3:00:46 PM, Session end: 10/21/2014 3:00:46 PM

Server: 192.168.1.197 (Windows) TCP 8063

Server: 192.168.1.197 (Windows) TCP 8063 (0 data bytes sent), Client: 192.168.1.199 (Windows) TCP 2647 (0 data bytes sent), Session start: 10/21/2014 3:00:39 PM, Session end: 10/21/2014 3:00:44 PM

Server: 192.168.1.199 (Windows) TCP 2641

Server: 192.168.1.199 (Windows) TCP 2641 (615 data bytes sent), Client: 192.168.1.199 (Windows) TCP 8067 (10220 data bytes sent), Session start: 10/21/2014 3:00:36 PM, Session end: 10/21/2014 3:00:47 PM

Figure b.13: Outgoing sessions

Open TCP Ports:

TCP 2639 - Entropy (in \ out): 62.95 \ 72.29 Typical data (in \ out):

TCP 2641 - Entropy (in \ out): 75.50 \ 72.42 Typical data (in \ out): ereoten ees eentGeetteeceitnente
Figure b.14: Open TCP Ports

**Analysing Public Server:**

- **IP:** 192.168.1.197
- **MAC:** Unknown
- **Hostname:**
- **OS:** Windows
- **TTL:** 128 (distance: 31)

**Open TCP Ports: 8063**

- **TCP 7003 - Entropy (in \ out):** 69.20 \ 69.01 Typical data (in \ out): HTTP/1.1 000 Cootaaeee/ 0ntrol: \tose /testaoot te/seeoeotioe/ se
- **Sent:** 20 packets (16,529 Bytes), 0.00 % cleartext (0 of 0 Bytes)
Appendix B.

192.168.1.197 (Windows) -> 192.168.1.199 (Windows) : 19 packets (16,419 Bytes), 0.00 % cleartext (0 of 0 Bytes)

TCP: 8063 -> 2639 : 13 packets (15,432 Bytes), 0.00 % cleartext (0 of 0 Bytes)
TCP: 8063 -> 2647 : 3 packets (132 Bytes), 0.00 % cleartext (0 of 0 Bytes)
TCP: 7003 -> 8055 : 3 packets (855 Bytes), 0.00 % cleartext (0 of 0 Bytes)

192.168.1.197 (Windows) -> 255.255.255.255 : 1 packets (110 Bytes), 0.00 % cleartext (0 of 0 Bytes)
UDP: 50414 -> 1211 : 1 packets (110 Bytes), 0.00 % cleartext (0 of 0 Bytes)

Received: 18 packets (3,190 Bytes), 0.00 % cleartext (0 of 0 Bytes)

192.168.1.199 (Windows) -> 192.168.1.197 (Windows) : 18 packets (3,190 Bytes), 0.00 % cleartext (0 of 0 Bytes)

TCP: 2639 -> 8063 : 12 packets (2,309 Bytes), 0.00 % cleartext (0 of 0 Bytes)
TCP: 2647 -> 8063 : 4 packets (172 Bytes), 0.00 % cleartext (0 of 0 Bytes)
TCP: 8055 -> 7003 : 2 packets (709 Bytes), 0.00 % cleartext (0 of 0 Bytes)

Incoming sessions: 2

Server: 192.168.1.197 (Windows) TCP 7003
Server: 192.168.1.197 (Windows) TCP 7003 (735 data bytes sent), Client: 192.168.1.199 (Windows) TCP 8055 (629 data bytes sent), Session start: 10/21/2014 3:00:46 PM, Session end: 10/21/2014 3:00:46 PM

Server: 192.168.1.197 (Windows) TCP 7003
Server: 192.168.1.197 (Windows) TCP 8063 (0 data bytes sent), Client: 192.168.1.199 (Windows) TCP 2647 (0 data bytes sent), Session start: 10/21/2014 3:00:39 PM, Session end: 10/21/2014 3:00:44 PM

Outgoing sessions: 1

Server: 192.168.1.199 (Windows) TCP 2639
Server: 192.168.1.199 (Windows) TCP 2639 (1829 data bytes sent), Client: 192.168.1.197 (Windows) TCP 8063 (14900 data bytes sent), Session start: 10/21/2014 3:00:36 PM, Session end: 10/21/2014 3:00:46 PM

### Sent Analysis:

<table>
<thead>
<tr>
<th>Device from</th>
<th>Decid e to</th>
<th>TCP</th>
<th>Session start</th>
<th>Session end</th>
<th>Sessio n time</th>
<th>Byte encrypte d</th>
<th>Without encryptio n</th>
<th>B/C</th>
<th>Researc h packet</th>
<th>Without encryptio n</th>
<th>Direc t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public Seve r 8063</td>
<td>Private 263 9</td>
<td>10/21/2014 3:00:36 PM</td>
<td>10/21/2014 3:00:46 PM</td>
<td>10</td>
<td>1490 0</td>
<td>10</td>
<td>7</td>
<td>4</td>
<td>14900</td>
<td>9148</td>
<td>5435</td>
</tr>
<tr>
<td>private 8063</td>
<td>bank 164 7</td>
<td>10/21/2014 3:00:39 PM</td>
<td>10/21/2014 3:00:44 PM</td>
<td>5</td>
<td>629</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>629</td>
<td>429</td>
<td>324</td>
</tr>
<tr>
<td>bank 8067</td>
<td>public 124 1</td>
<td>10/21/2014 3:00:36 PM</td>
<td>10/21/2014 3:00:47 PM</td>
<td>11</td>
<td>1022 0</td>
<td>11</td>
<td>11</td>
<td>3</td>
<td>10220</td>
<td>7417</td>
<td>6189</td>
</tr>
</tbody>
</table>

Table b.1 sent analysis

- **encrypted:** Network traffic from the current research analysis
- **Without Encryption:** Network traffic without encryption
- **B/C:** Direct Network traffic from client to Internet Banking Server

### Time Comparison

---

219
Appendix B.

Figure b.15: time comparison

Bytes Comparison

Figure b.16: byte comparison

Sent: 20 packets (16,529 Bytes), 0.00 % cleartext (0 of 0 Bytes)
Sent Packets : 20 packets

Packets size : 16,529 Bytes

192.168.1.197 (Windows) -> 192.168.1.199 (Windows) : 19 packets (16,419 Bytes), 0.00 % cleartext (0 of 0 Bytes)

192.168.1.197 : Server IP “Public Server”

Windows : Client OS

192.168.1.199: Server IP “ Private Server”

Windows : Server OS

19 packets : Number of packets

16,419 Bytes : packets size

0.00 % cleartext : no clear data

0 of 0 Bytes : Cleartext size

TCP: 8063 -> 2639 : 13 packets (15,432 Bytes), 0.00 % cleartext (0 of 0 Bytes)

8063 : server port (public server)

2639 : client port

13 packets : number of packets

15,432 Bytes : packets size

0.00 % cleartext : no clear data

0 of 0 Bytes : Cleartext size

TCP: 8063 -> 2647 : 3 packets (132 Bytes), 0.00 % cleartext (0 of 0 Bytes)

8063 : server port (Public server)

2647 : client port

3 packets : number of packets

132 Bytes : packets size

0.00 % cleartext : no clear data
0 of 0 Bytes : Cleartext size

TCP: 7003 -> 8055 : 3 packets (855 Bytes), 0.00 % cleartext (0 of 0 Bytes)

7003 : server port (Public server)

8055 : server port (Private server)

3 packets : number of packets

855 Bytes : packets size

0.00 % cleartext : no clear data

0 of 0 Bytes : Cleartext size

Function  SendToPrivate(ByVal DongleID As String, ByVal UserName As String, ByVal UserPassword As String, ByVal UserFingerPrint As String) As String
    Dim ws As New WebReference.WebService
    Dim Success As Boolean = False
    UserName = EncryptText(UserName, "SUCCESS")
    Dim str As String = ws.ClientKey(DongleID, UserName, UserPassword, UserFingerPrint)
    Return str
End Function

Figure b.17: Send To Private Function

192.168.1.197 (Windows) -> 255.255.255.255 : 1 packets (110 Bytes), 0.00 % cleartext (0 of 0 Bytes)

UDP: 50414 -> 1211 : 1 packets (110 Bytes), 0.00 % cleartext (0 of 0 Bytes)

Received Analysis:

Received: 18 packets (3,190 Bytes), 0.00 % cleartext (0 of 0 Bytes)

Received packets : 18 packets

packets size : 3,190 Bytes

192.168.1.199 (Windows) -> 192.168.1.197 (Windows) : 18 packets (3,190 Bytes), 0.00 % cleartext (0 of 0 Bytes)

192.168.1.199: client IP

Windows : Server OS

192.168.1.197 : Server IP “Public Server”

Windows : client OS
18 packets: Number of packets
3,190 Bytes: packets size
0.00% cleartext: no clear data
0 of 0 Bytes: Cleartext size

TCP: 2639 -> 8063: 12 packets (2,309 Bytes), 0.00% cleartext (0 of 0 Bytes)

2639: client port
8063: Server port “Public Server”
12 packets: number of packets
2,309 Bytes: packets size
0.00% cleartext: no clear data
0 of 0 Bytes: Cleartext size

TCP: 2647 -> 8063: 4 packets (172 Bytes), 0.00% cleartext (0 of 0 Bytes)

2647: client port
8063: Server port “Public Server”
4 packets: number of packets
172 Bytes: packets size
0.00% cleartext: no clear data
0 of 0 Bytes: Cleartext size

TCP: 8055 -> 7003: 2 packets (709 Bytes), 0.00% cleartext (0 of 0 Bytes)

8055: Server port “Private Server”
8063: Server port “Public Server”
12 packets: number of packets
2,309 Bytes: packets size
0.00% cleartext: no clear data
Appendix B.

0 of 0 Bytes : Cleartext size

```csharp
Dim tblName As String = "tblConfirmClient"
Dim tblNameKey As String = "ClientID"
Dim clientID As Integer = 0
Dim strKey As String = "\""
Dim OldCon As New SqlConnection
Dim OleDbCon As New SqlConnection
Dim OleDbTable As OleDbConnection
Dim OleDbKey As OleDbAdapter
Dim OleDbJet As New OleDbConnection
Dim OleDbDs As New DataSet
Dim oleDbDs As New OleDbTable
Dim decryptedName As String
Dim sql As String = "SELECT * FROM tblConfirmClient WHERE ClientID = " & clientID & " AND UserName = " & decryptedName & " AND User"
Try
    OleCon = New SqlConnection(sql)
    OleCon.Open()
    OleDbTable = New OleDbDataAdapter(sql, OleCon)
    OleDbKey.Fill(OleDbTable, tblName)
    ClientID = OleDbTable.Rows(0).Item(0)
    If ClientID > 0 Then
        Dim r As New Random()
        Dim lengthKey As Integer = r.Next(18, 36)
        strKey = GetRandomKey(lengthKey) "SUCCESS"
        Dim sqlUpdateKey As String = "UPDATE tblClients SET UserKey = " & strKey & " WHERE ClientID = 1"
        Try
            OleCon = New SqlConnection(sqlUpdateKey)
            OleCon.Open()
            OleDbKey.Fill(OleDbTable, tblNameKey)
            OleDbDs = New OleDbTable(OleDbTable, OleCon)
            OleDbDs.Dispose()
            OleDbKey.Dispose()
            OleCon.Dispose()
            Catch ex As Exception
                throw(ex.Message)
        End Try
    End If
End Try
```

Figure b.18: Private Server Function

Analysing Client, Private Server and Bank Server

192.168.1.199 (Windows)

IP: 192.168.1.199

MAC: Unknown

Hostname:
OS: Windows

TTL: 128 (distance: 31)

Open TCP Ports:

TCP 2639 - Entropy (in \ out): 62.95 \ 72.29 Typical data (in \ out): CET /iaeas/ta leoeotn 1..tg tt

TCP 2641 - Entropy (in \ out): 75.50 \ 72.42 Typical data (in \ out): eetGeeteeceitmente \n
Sent: 42 packets (15,278 Bytes), 0.00 % cleartext (0 of 0 Bytes)

192.168.1.199 (Windows) -> 192.168.1.197 (Windows) : 18 packets (3,190 Bytes), 0.00 % cleartext (0 of 0 Bytes)

TCP: 2639 -> 8063 : 12 packets (2,309 Bytes), 0.00 % cleartext (0 of 0 Bytes)
TCP: 2647 -> 8063 : 4 packets (172 Bytes), 0.00 % cleartext (0 of 0 Bytes)
TCP: 8055 -> 7003 : 2 packets (709 Bytes), 0.00 % cleartext (0 of 0 Bytes)

192.168.1.199 (Windows) -> 192.168.1.199 (Windows) : 14 packets (11,420 Bytes), 0.00 % cleartext (0 of 0 Bytes)

TCP: 2641 -> 8067 : 4 packets (776 Bytes), 0.00 % cleartext (0 of 0 Bytes)
TCP: 8067 -> 2641 : 10 packets (10,644 Bytes), 0.00 % cleartext (0 of 0 Bytes)

Received: 33 packets (27,839 Bytes), 0.00 % cleartext (0 of 0 Bytes)

192.168.1.197 (Windows) -> 192.168.1.199 (Windows) : 19 packets (16,419 Bytes), 0.00 % cleartext (0 of 0 Bytes)

TCP: 8063 -> 2639 : 13 packets (15,432 Bytes), 0.00 % cleartext (0 of 0 Bytes)
TCP: 8063 -> 2647 : 3 packets (132 Bytes), 0.00 % cleartext (0 of 0 Bytes)
TCP: 7003 -> 8055 : 3 packets (855 Bytes), 0.00 % cleartext (0 of 0 Bytes)

192.168.1.199 (Windows) -> 192.168.1.199 (Windows) : 14 packets (11,420 Bytes), 0.00 % cleartext (0 of 0 Bytes)

TCP: 2641 -> 8067 : 4 packets (776 Bytes), 0.00 % cleartext (0 of 0 Bytes)
TCP: 8067 -> 2641 : 10 packets (10,644 Bytes), 0.00 % cleartext (0 of 0 Bytes)
Incoming sessions: 2

Server: 192.168.1.199 (Windows) TCP 2639
Server: 192.168.1.199 (Windows) TCP 2639 (1829 data bytes sent), Client: 192.168.1.197 (Windows) TCP 8063 (14900 data bytes sent), Session start: 10/21/2014 3:00:36 PM, Session end: 10/21/2014 3:00:46 PM

Server: 192.168.1.199 (Windows) TCP 2641
Server: 192.168.1.199 (Windows) TCP 2641 (615 data bytes sent), Client: 192.168.1.199 (Windows) TCP 8067 (10220 data bytes sent), Session start: 10/21/2014 3:00:36 PM, Session end: 10/21/2014 3:00:47 PM

Outgoing sessions: 3

Server: 192.168.1.197 (Windows) TCP 7003
Server: 192.168.1.197 (Windows) TCP 7003 (735 data bytes sent), Client: 192.168.1.199 (Windows) TCP 8055 (629 data bytes sent), Session start: 10/21/2014 3:00:46 PM, Session end: 10/21/2014 3:00:46 PM

Server: 192.168.1.197 (Windows) TCP 8063
Server: 192.168.1.197 (Windows) TCP 8063 (0 data bytes sent), Client: 192.168.1.199 (Windows) TCP 2647 (0 data bytes sent), Session start: 10/21/2014 3:00:39 PM, Session end: 10/21/2014 3:00:44 PM

Server: 192.168.1.199 (Windows) TCP 2641
Server: 192.168.1.199 (Windows) TCP 2641 (615 data bytes sent), Client: 192.168.1.199 (Windows) TCP 8067 (10220 data bytes sent), Session start: 10/21/2014 3:00:36 PM, Session end: 10/21/2014 3:00:47 PM

Sent Analysis:

Sent: 42 packets (15,278 Bytes), 0.00 % cleartext (0 of 0 Bytes)

Sent Packets : 42 packets

Packets size : 15,278 Bytes
192.168.1.199 : Client IP
Windows: Client OS

192.168.1.197: Server IP “Public Server”
Windows: Server OS

18 packets: Number of packets
3,190 Bytes: packets size
0.00 % cleartext: no clear data
0 of 0 Bytes: Cleartext size

TCP: 2639 -> 8063: 12 packets (2,309 Bytes), 0.00 % cleartext (0 of 0 Bytes)

2639: client port
8063: server port (public server)
12 packets: number of packets
3,190 Bytes: packets size
0.00 % cleartext: no clear data
0 of 0 Bytes: Cleartext size

TCP: 2647 -> 8063: 4 packets (172 Bytes), 0.00 % cleartext (0 of 0 Bytes)

2647: client port
8063: server port (public server)
4 packets: number of packets
172 Bytes: packets size
0.00 % cleartext: no clear data
0 of 0 Bytes: Cleartext size

TCP: 8055 -> 7003: 2 packets (709 Bytes), 0.00 % cleartext (0 of 0 Bytes)

8055: server port (private server)
7003 : server port (public server)

2 packets : number of packets

709 Bytes : packets size

0.00 % cleartext : no clear data

0 of 0 Bytes : Cleartext size

<table>
<thead>
<tr>
<th>Local IP Address</th>
<th>Destination IP Address</th>
<th>Packets</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.1.199 (Windows)</td>
<td>192.168.1.199 (Windows)</td>
<td>14 packets (11,420 Bytes)</td>
<td>0.00 % cleartext (0 of 0 Bytes)</td>
</tr>
</tbody>
</table>

192.168.1.199 : Client IP

Windows : Client OS

Windows: Server OS

14 packets: Number of packets

11,420 Bytes: packets size

0.00 % cleartext: no clear data

0 of 0 Bytes: Cleartext size

TCP: 2641 -> 8067: 4 packets (776 Bytes), 0.00 % cleartext (0 of 0 Bytes)

2641: client port

8067: server port (Bnak Server)

4 packets: number of packets

776 Bytes: packets size

0.00 % cleartext: no clear data

0 of 0 Bytes: Cleartext size

Sub SendToBank(ByVal DongleID As String, ByVal UserName As String, ByVal UserPassword As String, ByVal UserFingerPrint As String)
    Dim strPost As String
    Dim ServerURL As String
    Dim IsSuccess As Boolean = False
    UserName = EncryptText(UserName, "SUCCESS")
    strPost = "DongleID=" & DongleID
    strPost = strPost & "&UserName=" & UserName
    strPost = strPost & "&UserPassword=" & UserPassword
    strPost = strPost & "&UserFingerPrint=" & UserFingerPrint
    ServerURL = "http://192.168.1.199:8067"
End Sub

Figure b.20: Send To Bank Function

TCP: 8067 -> 2641: 10 packets (10,644 Bytes), 0.00 % cleartext (0 of 0 Bytes)

8067: server port (Bnak Server)

2641: client port

4 packets: number of packets
776 Bytes : packets size

0.00 % cleartext : no clear data

0 of 0 Bytes : Cleartext size

Received Analysis:

Received : 33 packets (27,839 Bytes), 0.00 % cleartext (0 of 0 Bytes)

Received packets : 33 packets

packets size : 27,839 Bytes

192.168.1.197 (Windows) -> 192.168.1.199 (Windows) : 19 packets (16,419 Bytes),

0.00 % cleartext (0 of 0 Bytes)

192.168.1.197 : Server IP “Public Server”

Windows : Server OS

192.168.1.199: client IP

Windows : client OS

19 packets : Number of packets

16,419 Bytes : packets size

0.00 % cleartext : no clear data

0 of 0 Bytes : Cleartext size

TCP: 8063 -> 2639 : 13 packets (15,432 Bytes), 0.00 % cleartext (0 of 0 Bytes)

8063 Server port “Public Server”

2639 : client port

13 packets : number of packets

13,432 Bytes : packets size

0.00 % cleartext : no clear data

0 of 0 Bytes : Cleartext size

TCP: 8063 -> 2647 : 3 packets (132 Bytes), 0.00 % cleartext (0 of 0 Bytes)
8063 : Server port “Public Server”

2647 : client port

3 packets : number of packets

132 Bytes : packets size

0.00 % cleartext : no clear data

0 of 0 Bytes : Cleartext size

TCP: 7003 -> 8055 : 3 packets (855 Bytes), 0.00 % cleartext (0 of 0 Bytes)

7003 :

8055 : Server port “Private Server”

3 packets : number of packets

132 Bytes : packets size

0.00 % cleartext : no clear data

0 of 0 Bytes : Cleartext size

192.168.1.199 (Windows) -> 192.168.1.199 (Windows) : 14 packets (11,420 Bytes), 0.00 % cleartext (0 of 0 Bytes)


Windows : Server OS

192.168.1.199: client IP

Windows : client OS

14packets : Number of packets

11,420 Bytes : packets size

0.00 % cleartext : no clear data

0 of 0 Bytes : Cleartext size

TCP: 2641 -> 8067 : 4 packets (776 Bytes), 0.00 % cleartext (0 of 0 Bytes)

2641 : Client port
8067 : Server port “Bank Server”

4 packets : number of packets

776 Bytes : packets size

0.00 % cleartext : no clear data

0 of 0 Bytes : Cleartext size

TCP: 8067 -> 2641 : 10 packets (10,644 Bytes), 0.00 % cleartext (0 of 0 Bytes)

8067 : Server port “Bank Server”

2461 : Client port

10 packets : number of packets

10,644 Bytes : packets size

0.00 % cleartext : no clear data

0 of 0 Bytes : Cleartext size
Figure b.21: Bank Server Function

Analysing Incoming sessions

Incoming sessions : 2

Number of Incoming sessions : 2 sessions

Server: 192.168.1.199 (Windows) TCP 2639

Server: 192.168.1.199 (Windows) TCP 2639 (1829 data bytes sent), Client: 192.168.1.197 (Windows) TCP 8063 (14900 data bytes sent), Session start: 10/21/2014 3:00:36 PM, Session end: 10/21/2014 3:00:46 PM

Server: 192.168.1.199 : Server IP

Windows : Client OS
2639 : Server Port

Client: **192.168.1.197** : Public Server IP

**Windows** : Client OS

8063 : Public Server Port

### Server: 192.168.1.199 (Windows) TCP 2641

Server: **192.168.1.199** (Windows) TCP 2641 (615 data bytes sent), Client: **192.168.1.199** (Windows) TCP 8067 (10220 data bytes sent), Session start: **10/21/2014 3:00:36 PM**, Session end: **10/21/2014 3:00:47 PM**

Server: **192.168.1.199** : Server IP

**Windows** : Client OS

2641 : Server Port

Client: **192.168.1.199** : Bank Server IP

**Windows** : Client OS

8067 : Bank Server Port

### Analysing Outgoing sessions

**Outgoing sessions : 3**

**Number of Outgoing sessions : 3 sessions**

### Server: 192.168.1.197 (Windows) TCP 7003

Server: **192.168.1.197** (Windows) TCP 7003 (735 data bytes sent), Client: **192.168.1.199** (Windows) TCP 8055 (629 data bytes sent), Session start: **10/21/2014 3:00:46 PM**, Session end: **10/21/2014 3:00:46 PM**

Server: **192.168.1.197** : Server IP

**Windows** : Server OS
7003 : Server Port

Client: 192.168.1.199 : Private Server IP

Windows : Client OS

8055 : Private Server Port

Sessions

<table>
<thead>
<tr>
<th>Frame nr.</th>
<th>Client host</th>
<th>C. port</th>
<th>Server host</th>
<th>S. port</th>
<th>Protocol</th>
<th>Start Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>192.168.1.199(Windows)</td>
<td>2947</td>
<td>192.168.1.197(Windows)</td>
<td>8063</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table b.2: session table

Clearest:

.pcap file:

" C:\Users\Administrator.OMECO-ERP1-PC\Desktop\NetworkMiner_1-6-1\NetworkMiner_1-6-1\Captures\ NM_2014-10-21T15-00-29.pcap"

version encoding soap Envelope soap soap Header 2005 discovery Action Action Header soap Body Resolve Address Address Resolve Body Envelope version encoding soap Envelope soap soap Header 2005 discovery Action Action Header soap Body Resolve Address Address Resolve Body Envelope GET Host Windows Accept Cookie ASP Connection version encoding soap Envelope soap soap Header 2005 discovery Action Action Header soap Body Resolve Address Address Resolve Body Envelope private Server ASP NET Date Tue Oct GMT PUBLIC head title body form method post action div class input type hidden name value div class input type hidden name value div table border style width height align center class table border style width style width 100 align center class src JPG style height align center top class table border style width style width 100 height class links style width class span style right input name type text class style height style height top class span style input name type text class style height span style input name type text class style height style height top input type submit name value Confirm class style table border style width height style class span form
Appendix B.

version encoding soap Envelope soap soap soap Header 2005 discovery Action Action Header soap Body Resolve Address Address Address Resolve Body Envelope GET Host Windows Accept Cookie ASP Connection Not Found private Server ASP NET Date Tue Oct GMT PUBLIC head title Detailed Error Not Found style type body Helvetica code code pre first first padding legend padding legend color solid solid solid solid link visited color width color background background width width width solid solid table 100 width width table alt table alt color clear clear preferred padding body div header Server Error Application div Internet Information Services div content div class legend Error Summary Error Not Found The resource you are looking for has been removed had its name changed temporarily unavailable div class legend Detailed Error Information div table border class alt Module Web Core class alt Handler Error Code div table border class alt Requested Physical Path public images class alt Method User div class clear div class legend Most likely causes The directory file specified does not exist the Web server The contains error custom module such access the file div class legend Things you can try Create the content the Web server Review the Create tracing rule track failed requests for this status code and see which module calling For more information about creating tracing rule for failed requests here div class legend Links and More Information This error means that the file directory does not exist the server Create the file directory and try the request again View more information GET Host Windows Accept Cookie ASP Connection Not Found private Server ASP NET Date Tue Oct GMT PUBLIC head title Detailed Error Not Found style type body Helvetica code code pre first first padding legend padding legend color solid solid solid solid link visited color width color background background width width width solid solid table 100 width width table alt table alt color clear clear preferred padding body div header Server Error Application version port version port version port version port POST Web Services Protocol Host 444 Expect 100 Continue version encoding soap Envelope soap soap Body Body Envelope private Server ASP NET Date Tue Oct GMT version encoding soap Envelope soap soap Body Body Envelope private Location Server ASP NET Date Tue Oct GMT head title Object moved body Object moved here GET Host Windows Accept Cookie ASP Connection private Server ASP NET Date Tue Oct GMT PUBLIC head title Home Page style title title Aqua style type color padding color padding color color body form method post action div class input type hidden name value input type hidden name value input type hidden name value input type hidden name value input type hidden name value input type hidden name value input type hidden name value false value value submit script src type script src type script type function context context data context false var script src type script type Sys undefined throw new Error ASP NET Ajax failed load script src type div class input type hidden
Appendix B.

name value input type hidden name value div class title strong Internet script type Sys table width 100 height top width style alt Skip Navigation Links src width height style div table style document document src alt Account Summary style class this this style class title Account Summary and Recent Account Summary style height div style display table style style height div style width src alt class this this style class Funds Transfer style height table style style height div style width src alt class this this style class Cheque Book Request style height version encoding soap Envelope soap soap Header 2005 discovery Action Action Header soap Body Resolve Address Address Resolve Body Envelope version encoding soap Envelope soap soap Header 2005 discovery Action Action Header soap Body Resolve Address Address Resolve Body Envelope.