Master Thesis

“Software to provide security for Web Browser Cookies and Passwords using Trusted Computing Technology”

Rajasekhar Kannamanani
Department of Computer Science
Fraunhofer Institute for Security IT
Summer Semester 2008

Supervisors:
Dr. Andres U. Schmidt
Mr. Nicolai Kuntze
Abstract

For further development and enhancement of browser security, there is need to develop something new which provides the better security for the user credential information which will be stored in user PC through the web browser.

Although we have many softwares which provide the security functionalities for system, there are many vulnerabilities, so for this there is need to use some functions and specifications mentioned by Trusted Computing group. Therefore, the goal of this semester thesis was to develop software which provides a better security to the browser cookies and passwords by using the functionality of TPM.

In the course of this master thesis, the software was developed which have to be run before user opens his browser to browse and it works with more accuracy by checking the browser in user system process for every second, so that it will be active until user shut downs his system. Currently this software was developed only to work under Linux operating system to make it work with TPM and TPM emulator which will be used instead of TPM if not exists in user system.
Abstract........................................................................................................................................... 3

1. Introduction .................................................................................................................................... 14
   1.1. Motivation and Goals.................................................................................................................. 14
   1.2. Tasks......................................................................................................................................... 14
   1.3. Outline..................................................................................................................................... 15
   1.4. Acknowledgments...................................................................................................................... 15

2. Web Browser and their basic Security Settings ............................................................................ 16
   2.1. Web browser ............................................................................................................................... 16
   2.2. Web Browser Architecture......................................................................................................... 16
   2.3. Choosing a Browser.................................................................................................................... 18
   2.4. Protocols and Standards............................................................................................................ 18
   2.5. Authentication Credentials......................................................................................................... 20
      2.5.1. Authentication Options......................................................................................................... 20
      2.5.2. Password Cracking Techniques............................................................................................ 21
   2.6. Active Content and Cookies ...................................................................................................... 22
      2.6.1. Cookies................................................................................................................................ 23
      2.6.2. Website Certificates.............................................................................................................. 24
      2.6.2.1. Trusting a Certificate....................................................................................................... 25
      2.6.2.2. Checking a certificate......................................................................................................... 25
   2.7. Weakness of Current Browsers ................................................................................................. 26
      2.7.1. Comparison of Web Browsers and its Vulnerabilities............................................................ 26
      2.7.1.1. Security in IE...................................................................................................................... 27
      2.7.2. Web browser security Summary............................................................................................. 28
      2.7.3. Vulnerabilities........................................................................................................................ 28
         2.7.3.1. Publicly disclosed without a patch.................................................................................... 31
         2.7.3.2. Fully-Disclosed ................................................................................................................ 32
         2.7.3.3. Patch Delay ....................................................................................................................... 34
         2.7.3.4. Total Vulnerabilities......................................................................................................... 35
         2.7.3.5. Fully-disclosed Vulnerabilities ....................................................................................... 36
   2.8. Web Browser’s basic Security Settings ...................................................................................... 37
      2.8.1. Why Secure Your Web Browser? ........................................................................................... 37
      2.8.2. Understanding Web Browser Features.................................................................................... 38
      2.8.3. Vulnerabilities and Attack Vectors....................................................................................... 40
         2.8.3.1. ActiveX Controls............................................................................................................... 41
         2.8.3.2. Java.................................................................................................................................. 41
         2.8.3.3. Cross-Site Scripting......................................................................................................... 41
         2.8.3.4. Cross-Zone and Cross-Domain Vulnerabilities............................................................... 41
         2.8.3.5. Malicious Scripting, Active Content, and HTML............................................................ 42
         2.8.3.6. Spoofing............................................................................................................................. 42
   2.9. How to Secure Your Web Browser ............................................................................................ 42
      2.9.1. Microsoft Internet Explorer................................................................................................... 43
      2.9.2. Mozilla Firefox....................................................................................................................... 50
      2.9.3. Keeping Your Computer Secure ............................................................................................ 57
         2.9.3.1. Install and use antivirus software ..................................................................................... 57
         2.9.3.2. Enable automatic software updates if available ............................................................... 58
         2.9.3.3. Avoid unsafe behavior ...................................................................................................... 58
         2.9.3.4. Follow the principle of least privilege — don't enable it if you don't need it ................. 58
3. Idea and Concepts .............................................................. 59
3.1. Making Browser more secure ........................................... 59
3.2. Concepts used by TPM for Encryption ............................ 60
   3.2.1. Data Security ......................................................... 60
   3.2.2. Cryptography ......................................................... 60
   3.2.2.1. Symmetric Encryption (Private-key Cryptography) ... 63
   3.2.2.2. Asymmetric Encryption (Public-key Cryptography) ... 64
   3.2.2.2.1. Algorithms used in Asymmetric Encryption ........ 65
   3.2.2.2.1.1. Modular Arithmetic ....................................... 66
   3.2.2.2.1.1.1. Congruent relation ..................................... 66
   3.2.2.2.1.1.2. Properties of modular arithmetic .................. 67
   3.2.2.2.1.2. DH - Diffie-Hellman Key Agreement Algorithm ... 67
   3.2.2.2.1.2.1. Key agreement Algorithm ........................... 67
   3.2.2.2.1.2.2. DH - Mathematical Explanation .................... 67
   3.2.2.2.1.2.3. One-Way function in DH .............................. 68
   3.2.2.2.1.3. RSA ........................................................ 68
   3.2.2.2.1.3.1. RSA Encryption ........................................ 68
   3.2.2.2.1.3.2. RSA Key Agreement .................................... 69
   3.2.2.2.1.3.3. RSA Signature .......................................... 70
   3.2.2.2.1.3.4. One-Way function in RSA ......................... 70
   3.2.2.2.1.3.5. RSA – Mathematical Explanation .................. 71
   3.2.2.2.1.4. DSA – Digital Signature Algorithm ................. 72
   3.2.2.2.1.4.1. DSA - Mathematical Explanation ................... 73
   3.2.2.2.1.4.2. One-Way function in DSA ......................... 74
   3.2.2.3. Hash Functions ................................................. 74
   3.2.2.3.1. Hash Function features .................................... 75
   3.2.2.3.2. Hash Function Types .............................. 77
   3.2.2.3.2.1. Modification detection codes (MDCs) ............ 77
   3.2.2.3.2.2. Message authentication codes (MACs) ............. 78
   3.2.2.3.3. Basic properties of MDCs and MACs .................. 79
   3.2.2.3.4. General model for iterated hash functions .......... 80
   3.2.2.3.5. Hash Function Algorithms .............................. 82
   3.2.2.3.5.1. MD5 ......................................................... 82
   3.2.2.3.5.1.1. Algorithms ............................................. 82
   3.2.2.3.5.1.1.1. MD4 hash function Algorithm ................. 82
   3.2.2.3.5.1.1.2. MD5 hash function Algorithm ................. 84
   3.2.2.3.5.2. SHA-1 Algorithm ........................................ 86
3.3. Symmetric vs. asymmetric algorithms ......................... 89
3.4. Structure of the Project ................................................. 90
   3.4.1. Project Initialisation .......................................... 90
   3.4.1.1. Requirements to run the project ......................... 91
   3.4.2. Authentication Process ....................................... 92
   3.4.3. Encryption Process ............................................. 92
   3.4.4. Sending commands to TPM for Encryption ............... 92
   3.4.5. Browser Checking Process ................................... 92
   3.4.6. Decryption Process ............................................. 92
   3.4.7. Sending commands to TPM for Decryption ............... 92
4. Trusted Computing ............................................................ 93
   4.1. Key concepts ........................................................ 93
   4.1.1. Endorsement key ................................................. 93
   4.1.2. Secure I/O ....................................................... 94
4.1.3. Memory curtaining.................................................................94
4.1.4. Sealed storage.......................................................................94
4.1.5. Remote attestation...............................................................95
4.2. Applications of Trusted Computing...............................................95
4.2.1. Digital rights management....................................................95
4.2.2. Identity theft protection.........................................................96
4.2.3. Preventing cheating in online games.......................................96
4.2.4. Protection from viruses and spyware.....................................96
4.2.5. Protection of biometric authentication data............................97
4.2.6. Verification of remote computation for grid computing...........97
5. Trusted Platform Module (TPM)......................................................98
5.1. Introduction.............................................................................98
5.2. Trusted Platform Capabilities...................................................99
5.3. Attestation mechanisms.........................................................99
5.4. TPM Characteristics.............................................................100
5.5. TPM component architecture................................................101
5.6. Purpose of Design................................................................101
5.7. TPM basic Functions............................................................104
5.8. Key types and classes.............................................................105
5.9. Root of Trust........................................................................105
5.9.1. Key protection by the RTS...................................................106
5.10. TPM operational states........................................................108
5.10.1. Combinations of operational states.................................108
5.11. Ownership on TPM.............................................................109
5.11.1. Taking TPM ownership on Windows.................................110
5.11.1.1. Initializing a TPM for First Use..................................110
5.11.1.2. Turning off and Clearing the TPM..............................111
5.11.1.3. Changing the TPM Owner Password.........................113
5.11.2. Taking TPM ownership on Linux.................................113
5.11.3. Clearing the TPM ownership.................................113
5.12. TPM Emulator..................................................................114
5.13. Uses..................................................................................114
5.13.1. Disk encryption.............................................................114
5.14. Miscellaneous.................................................................114
5.14.1. Trusted Platform Module microcontrollers manufacturers........114
6. Implementation of the Project...................................................116
6.1. Concept............................................................................116
6.2. Installation and Usage.........................................................116
6.3. Conditions to make Project work.........................................117
6.3.1. Initialisation of project....................................................118
6.3.2. Login to Project............................................................118
6.3.3. Authentication process..................................................122
6.3.4. Creating User Process......................................................123
6.4. Project implementation......................................................123
6.4.1. Encryption.................................................................123
6.4.2. Browser Checking.........................................................123
6.4.3. Decryption.................................................................124
7. Conclusion...............................................................................125
7.1. Final Thoughts: Better Browsing...........................................125
7.2. Summary............................................................................126
7.3. Final Word........................................................................126
8. Contents of the CD ........................................................................................................ 127
List of Acronyms ........................................................................................................... 128
9. Acronyms .................................................................................................................... 128
10. Bibliography ............................................................................................................. 131
List of Figures

Figure 1 Reference architecture for Web browsers ................................................................. 17
Figure 2 Number of open advisories versus time .................................................................. 35
Figure 3 Number of open high security advisories versus time ........................................... 36
Figure 4 Number of known vulnerability issues present versus time .................................. 36
Figure 5 Relative Cumulative danger versus time ................................................................. 36
Figure 6 Number of open fully-disclosed advisories versus time ......................................... 36
Figure 7 Number of open fully-disclosed high severity advisories versus time ................... 37
Figure 8 Number of fully-disclosed vulnerability issues present versus time ...................... 37
Figure 9 Relative cumulative danger of fully-disclosed issues versus time ......................... 37
Figure 10 Selecting Internet Options to manage Security Settings in IE ............................ 43
Figure 11 Selecting Security to manage Security Settings in IE .......................................... 44
Figure 12 Managing Security Level in IE .............................................................................. 45
Figure 13 Managing Security Level and Trusted Sites in IE ............................................... 46
Figure 14 Adding Trusted websites in IE .............................................................................. 46
Figure 15 Pop-up blocking and privacy in IE ......................................................................... 47
Figure 16 Cookie Handling in IE .......................................................................................... 48
Figure 17 Adding in-secure websites to tell IE not to use cookies ...................................... 48
Figure 18 Managing Advanced Settings in IE ....................................................................... 49
Figure 19 Managing programs to used by windows for each internet service in IE .......... 50
Figure 20 Selecting Options to manage security settings in Mozilla Firefox ..................... 51
Figure 21 Checking Firefox as a default browser ................................................................. 51
Figure 22 Managing Cookies in Mozilla Firefox ................................................................. 52
Figure 23 Setting Master Password in Mozilla Firefox ....................................................... 53
Figure 24 Managing Advanced Content Settings is Mozilla Firefox ................................. 54
Figure 25 Managing download settings in Mozilla Firefox .................................................. 55
Figure 26 Managing download settings in Mozilla Firefox .................................................. 56
Figure 27 Clearing Private Data in Mozilla Firefox .............................................................. 57
Figure 28 Symmetric Key File Encryption and Decryption ................................................... 64
Figure 29 Asymmetric Key File Encryption and Decryption .................................................. 65
Figure 30 Simplified classification of cryptographic hash functions and applications. [22] 78
Figure 31 General model for an iterated hash function. [22] ............................................... 81
Figure 32 Project Structure ................................................................................................. 90
Figure 33 TPM Architecture ............................................................................................... 101
Figure 34 TPM/RTS ........................................................................................................... 107
Figure 35 Key Hierarchy – upper level encrypt lower levels ............................................... 107
Figure 36 Project Implementation ....................................................................................... 118
Figure 37 Login screen to run the Project ........................................................................... 120
Figure 38 Changing Password for Project .......................................................................... 121
Figure 39 Changing SRK Password for TPM ................................................................. 122
List of Tables

Table 1 Various Browser Features ................................................................. 27
Table 2 Various Browsers Security Vulnerabilities ........................................ 30
Table 3 Various Browsers Security Vulnerabilities with out patch .................. 31
Table 4 Various Browsers Security Vulnerabilities fully disclosed .................... 33
Table 5 Various Browsers patch delay in days .............................................. 35
Table 6 Basic comparison between private and Public-Key Cryptosystems .......... 63
Table 7 Truth table for TPM Operational States ............................................ 108
Chapter

1. Introduction

1.1. Motivation and Goals

Till now the web browsers developed are not up to the mark with their vulnerabilities and so many disadvantages including security which is becoming more problematic for users these days if there is any security information exists in their cookies which was stored while browsing. To reduce this aspect I have designed software which reduces the threats to the user after running this project named as LINBROWS, emerged from the words LINUX and BROWSER by including the functionalities of TPM.

Linux was chosen as the target platform for two main reasons: First, almost all currently available TPM-based applications and projects run under Linux. Second, TPM is probably going to be part of more and more state-of-the-art personal computers, there are and will always be situations where a TPM is unavailable or inaccessible so the users may run TPM Emulator [1] instead of TPM and emulator can be used only for development purpose but not recommended for production and system security which was still under development and it can be ran only under LINUX environment rather than any Operating System.

1.2. Tasks

- Analysing the current browsers and their features, drawbacks.

- Developing a structure of the total project to provide security for browser cookies and passwords by using TPM features.
Implementation of the structure of the project into a real-time application to provide better security for browsers than existing softwares using Java language by using open source standard APIs such as jTSS or tpm4java to access TPM facilities.

1.3. Outline

This report is structured as follows: Chapter two gives a brief introduction into Web browser and its vulnerabilities and how to cross these vulnerabilities and the basic security settings of the browser. In chapter three, the Idea and Concepts to make web browser more secure than current existing browsers by using the features of Trusted Computing technologies. Chapter four, describes about Trusted Computing and its key concepts and applications. Chapter five, describes about TPM (Trusted Platform Module) and its characteristics, features, architecture, functions, how to take the ownership of TPM and TPM Emulator and its basic uses. Chapter six, explains about the implementation of the project and its functionality. Chapter seven, describes final word to choose a better browser and make it secure. Chapter eight gives the information about Bibliography used to complete whole project and some useful external website links.

1.4. Acknowledgments

I would like to extend my gratitude to all people who made this semester thesis possible. First of all I would like to thank my university course coordinator Prof. Dr. Klaus Harbusch and my thesis supervisor Prof. Zakrzewicz for their support and encouragement to complete the thesis in mean time and my internal supervisors Dr. Andreas U. Schmidt and Mr. Nicolai Kuntze for giving their support by providing necessary information about TPM, providing necessary hardware and software required for project, supportive guidance and encouragement in Fraunhofer Institute to complete my thesis successfully.
Chapter 2

2. Web Browser and their basic Security Settings

2.1. Web browser

A web browser is a software application that enables a user to display and interact with text, images, videos, music and other information typically located on a Web page at a website on the World Wide Web or a local area network. Text and images on a Web page can contain hyperlinks to other Web pages at the same or different website. Web browsers allow a user to quickly and easily access information provided on many Web pages at many websites by traversing these links. Web browsers format HTML information for display, so the appearance of a Web page may differ between browsers.

Some of the Web browsers available for personal computers include Internet Explorer, Mozilla Firefox, Safari, Opera, and Netscape in order of descending popularity (in November 2007). Web browsers are the most commonly used type of HTTP user agent. Although browsers are typically used to access the World Wide Web, they can also be used to access information provided by Web servers in private networks or content in file systems.

2.2. Web Browser Architecture

The reference architecture we derived is shown in following figure. It comprises eight major subsystems plus the dependencies between them [2].
The **User Interface** subsystem is the layer between the user and the Browser Engine. It provides features such as toolbars, visual page-load progress, smart download handling, preferences, and printing. It may be integrated with the desktop environment to provide browser session management or communication with other desktop applications.

The **Browser Engine** subsystem is an embeddable component that provides a high-level interface to the Rendering Engine. It loads a given URL and supports primitive browsing actions such as forward, back, and reload. It provides hooks for viewing various aspects of the browsing session such as current page load progress and JavaScript alerts. It also allows the querying and manipulation of Rendering Engine settings.

The **Rendering Engine** subsystem produces a visual representation for a given URL. It is capable of displaying HTML and Extensible Markup Language (XML) (Bray et al., 2004) documents, optionally styled with CSS, as well as embedded content such as images. It calculates the exact page layout and may use “reflow” algorithms to incrementally adjust the position of elements on the page. This subsystem also includes the HTML parser.

The **Networking** subsystem implements file transfer protocols such as HTTP and FTP. It translates between different character sets, and resolves MIME media types for files. It may implement a cache of recently retrieved resources.
The JavaScript Interpreter evaluates JavaScript (also known as ECMAScript) code, which may be embedded in web pages. JavaScript is an object-oriented scripting language developed by Netscape. Certain Java-Script functionality, such as the opening of pop-up windows, may be disabled by the Browser Engine or Rendering Engine for security purposes.

The XML Parser subsystem parses XML documents into a Document Object Model (DOM) tree. This is one of the most reusable subsystems in the architecture. In fact, almost all browser implementations leverage an existing XML Parser rather than creating their own from scratch.

The Display Backend subsystem provides drawing and windowing primitives, a set of user interface widgets, and a set of fonts. It may be tied closely with the operating system.

The Data Persistence subsystem stores various data associated with the browsing session on disk. This may be high-level data such as bookmarks or toolbar settings, or it may be low-level data such as cookies, security certificates, or cache.

2.3. Choosing a Browser

A browser is usually included with the installation of your operating system, but you are not restricted to that choice. Some of the factors to consider when deciding which browser best suits your needs include [3]

- **Compatibility** - Does the browser work with your operating system?
- **Security** - Do you feel that your browser offers you the level of security you want?
- **Ease of use** - Are the menus and options easy to understand and use?
- **Functionality** - Does the browser interpret web content correctly? If you need to install other plug-ins or devices to translate certain types of content, do they work?
- **Appeal** - Do you find the interface and way the browser interprets web content visually appealing?

2.4. Protocols and Standards

Web browsers communicate with Web Servers primarily using HTTP (hypertext transfer protocol) to fetch webpages. HTTP allows Web browsers to submit information to Web servers as well as fetch Web pages from them. The most commonly used HTTP is HTTP/1.1,
which is fully defined in RFC 2616. HTTP/1.1 has its own required standards that Internet Explorer does not fully support, but most other current-generation Web browsers do.

Pages are located by means of a URL (uniform resource locator, RFC 1738), which is treated as an address, beginning with \textit{http:} for HTTP access. Many browsers also support a variety of other URL types and their corresponding protocols, such as \textit{gopher:} for Gopher (a hierarchical hyperlinking protocol), \textit{ftp:} for FTP (file transfer protocol), \textit{rtsp:} for RTSP (real-time streaming protocol), and \textit{https:} for HTTPS (an SSL encrypted version of HTTP).

The file format for a Web page is usually HTML (hyper-text markup language) and is identified in the HTTP protocol using a MIME \textit{content type}. Most browsers natively support a variety of formats in addition to HTML, such as the JPEG, PNG and GIF image formats, and can be extended to support more through the use of plugins. The combination of HTTP \textit{content type} and URL protocol specification allows Web page designers to embed images, animations, video, sound, and streaming media into a Web page, or to make them accessible through the Web page.

Early Web browsers supported only a very simple version of HTML. The rapid development of proprietary Web browsers led to the development of non-standard dialects of HTML, leading to problems with Web interoperability. Modern Web browsers support a combination of standards- and defacto-based HTML and XHTML, which should display in the same way across all browsers. No browser fully supports HTML 4.01, XHTML 1.x or CSS 2.1 yet. Currently many sites are designed using WYSIWYG HTML generation programs such as Macromedia Dreamweaver or Microsoft FrontPage. These often generate non-standard HTML by default, hindering the work of the W3C in developing standards, specifically with XHTML and CSS (cascading style sheets, used for page layout).

Some of the more popular browsers include additional components to support Usenet news, IRC (Internet relay chat), and e-mail. Protocols supported may include NNTP (network news transfer protocol), SMTP (simple mail transfer protocol), IMAP (Internet message access protocol), and POP (post office protocol). These browsers are often referred to as Internet suites or \textit{application suites} rather than merely Web browsers [5].
2.5. **Authentication Credentials**

2.5.1. **Authentication Options**

**Passwords:** which are inexpensive to deploy but expensive to manage and support. They offer the weakest security because they can be easily hacked, guessed or stolen.

**Hardware Tokens:** which are smack enough to attach to a key chain and generate a constantly changing one-time password to enable two-factor authentication.

**Software Tokens on PCs:** which enable two-factor authentication but are less portable.

**Software Tokens on Mobile Devices:** Which allow authorized users to gain two-factor authentication from smart phones and PDAs for mobile access to enterprise information.

**Digital Certificates:** Which contain use identities and enable the centralized management of cryptographic keys.

**Smart Cards with Digital Certificates:** Which enable the integration of physical access and logical access via card based authenticators that contain digital credentials.

**USB Tokens with Digital Certificates:** That can be plugged into a standard USB port to enable cryptographic authentication.

**Hybrid One-Time Password USB Tokens:** that can be plugged into a standard USB port to enable two-factor authentication without need to key in a token code. They can also contain digital certificates.

**Biometric Devices:** Which enable authentication according to the physical characteristics of a user. Examples include fingerprint identification and retina scans [4].
2.5.2. Password Cracking Techniques

**Password Cracking Tools:** A variety of software tools, such as LOPtcrack and NT Crack, automate the guessing of passwords through brute force and with extensive dictionaries of frequently used passwords.

**Network Monitoring:** This technique, also known as “sniffing”, allows monitoring (without detection) the contents for any message that streams by and flagging messages based on keywords, such as “login” or “password”.

**Brute Force Dialling:** Programs like ToneLoc automate the process of locating modem telephone lines, then the hacker attempts sign–on with various password alternatives.

**Abuse of Administrative Tools:** Many tools that have been designed to control and improve networks can be misused for destructive purposes.

**Social Engineering:** In contrast to the high-tech tools available to uncover passwords, some intruders use non-technical approaches to steal passwords [7].

**Strong Authentication:** The user must provide significantly stronger proof of identity before being granted access to protected resources. Typically, this proof is established by presenting multiple forms of identity or “factors”. The more factors a user must present, the more secure an application is considered to be, (Password solutions only require one identifier and are therefore considered the least secure.) Identifiers fall into three broad categories [6].

- Something only the user knows. This includes passwords and confidential PINs.
- Something only the user has. This is usually a physical device (e.g. a token or smart card) that contains a unique and hard-to-defeat identifier (for example, a one-time authentication code or encrypted digital certificate).
- Something only the user is. This category includes biometric identifiers that are unique to an individual, such as retinal or fingerprint scans.

Historically, two-factor authentication—which is similar to the model established for ATM cards and machines—has been the most common form of strong authentication for users. To prove identity and gain access, an individual must present two factors, a token or smart card and a confidential PIN. As with an ATM Card, a criminal must steal the physical device and
have access to the user’s PIN in order to impersonate that user. This “raises the bar” sufficiently to discourage many identity thieves, who typically will move on, looking for an easier target.

2.6. **Active Content and Cookies**

Many people browse the Internet without much thought to what is happening behind the scenes. Active content and cookies are common elements that may pose hidden risks when viewed in a browser or email client. What is active content?

To increase functionality or add design embellishments, web sites often rely on scripts that execute programs within the web browser. This active content can be used to create "splash pages" or options like drop-down menus. Unfortunately, these scripts are often a way for attackers to download or execute malicious code on a user's computer.

- **JavaScript** - JavaScript is just one of many web scripts (other examples are VBScript, ECMAScript, and JScript) and is probably the most recognized. Used on almost every web site now, JavaScript and other scripts are popular because users expect the functionality and "look" that it provides, and it's easy to incorporate (many common software programs for building web sites have the capability to add JavaScript features with little effort or knowledge required of the user). However, because of these reasons, attackers can manipulate it to their own purposes. A popular type of attack that relies on JavaScript involves redirecting users from a legitimate web site to a malicious one that may download viruses or collect personal information.

- **Java and ActiveX controls** - Different from JavaScript, Java and ActiveX controls are actual programs that reside on your computer or can be downloaded over the network into your browser. If executed by attackers, untrustworthy ActiveX controls may be able to do anything on your computer that you can do (such as running spyware and collecting personal information, connecting to other computers, and potentially doing other damage). Java applets usually run in a more restricted environment, but if that environment isn't secure, then malicious Java applets may create opportunities for attack as well.

JavaScript and other forms of active content are not always dangerous, but they are common tools for attackers. You can prevent active content from running in most browsers, but realize that the added security may limit functionality and break features of some sites you visit.
Before clicking on a link to a web site that you are not familiar with or do not trust, take the precaution of disabling active content.

These same risks may also apply to the email program you use. Many email clients use the same programs as web browsers to display HTML, so vulnerabilities that affect active content like JavaScript and ActiveX often apply to email. Viewing messages as plain text may resolve this problem [8].

2.6.1. Cookies

When you browse the Internet, information about your computer may be collected and stored. This information might be general information about your computer (such as IP address, the domain you used to connect (e.g., .edu, .com, .net), and the type of browser you used). It might also be more specific information about your browsing habits (such as the last time you visited a particular web site or your personal preferences for viewing that site).

Cookies can be saved for varying lengths of time:

- *Session cookies* - Session cookies store information only as long as you're using the browser; once you close the browser, the information is erased. The primary purpose of session cookies is to help with navigation, such as by indicating whether or not you've already visited a particular page and retaining information about your preferences once you've visited a page.

- *Persistent cookies* - Persistent cookies are stored on your computer so that your personal preferences can be retained. In most browsers, you can adjust the length of time that persistent cookies are stored. It is because of these cookies that your email address appears by default when you open your Yahoo! or Hotmail email account, or your personalized home page appears when you visit your favorite online merchant. If an attacker gains access to your computer, he or she may be able to gather personal information about you through these files.

To increase your level of security, consider adjusting your privacy and security settings to block or limit cookies in your web browser (see Evaluating Your Web Browser's Security Settings for more information). To make sure that other sites are not collecting personal information about you without your knowledge, choose to only allow cookies for the web site
you are visiting; block or limit cookies from a third-party. If you are using a public computer, you should make sure that cookies are disabled to prevent other people from accessing or using your personal information [8].

2.6.2. Website Certificates

You may have been exposed to web site, or host, certificates if you have ever clicked on the padlock in your browser or, when visiting a web site, have been presented with a dialog box claiming that there is an error with the name or date on the certificate. Understanding what these certificates are may help you protect your privacy.

What are web site certificates?

If an organization wants to have a secure web site that uses encryption, it needs to obtain a site, or host, certificate. Some steps you can take to help determine if a site uses encryption are to look for a closed padlock in the status bar at the bottom of your browser window and to look for "https:" rather than "http:" in the URL (see Protecting Your Privacy for more information). By making sure a web site encrypts your information and has a valid certificate, you can help protect yourself against attackers who create malicious sites to gather your information. You want to make sure you know where your information is going before you submit anything (see Avoiding Social Engineering and Phishing Attacks for more information).

If a web site has a valid certificate, it means that a certificate authority has taken steps to verify that the web address actually belongs to that organization. When you type a URL or follow a link to a secure web site, your browser will check the certificate for the following characteristics:

1. the web site address matches the address on the certificate
2. the certificate is signed by a certificate authority that the browser recognizes as a "trusted" authority
2.6.2.1. Trusting a Certificate

The level of trust you put in a certificate is connected to how much you trust the organization and the certificate authority. If the web address matches the address on the certificate, the certificate is signed by a trusted certificate authority, and the date is valid, you can be more confident that the site you want to visit is actually the site that you are visiting. However, unless you personally verify that certificate's unique fingerprint by calling the organization directly, there is no way to be absolutely sure.

By trusting a certificate, you have trusted the certificate authority to perform this verification for you. However, it is important to realize that certificate authorities vary in how strict they are about validating all of the information in the requests and about making sure that their data is secure. By default, your browser contains a list of more than 100 trusted certificate authorities. That means that, by extension, you are trusting all of those certificate authorities to properly verify and validate the information. Before submitting any personal information, you may want to look at the certificate.

2.6.2.2. Checking a certificate

There are two ways to verify a web site's certificate in Internet Explorer or Mozilla. One option is to click on the padlock in the status bar of your browser window. However, your browser may not display the status bar by default. Also, attackers may be able to create malicious web sites that fake a padlock icon and display a false dialog window if you click that icon. A more secure way to find information about the certificate is to look for the certificate feature in the menu options. This information may be under the file properties or the security option within the page information. You will get a dialog box with information about the certificate, including the following:

- **who issued the certificate** - You should make sure that the issuer is a legitimate, trusted certificate authority (you may see names like VeriSign, thawte, or Entrust). Some organizations also have their own certificate authorities that they use to issue certificates to internal sites such as intranets.
- **who the certificate is issued to** - The certificate should be issued to the organization who owns the web site. Do not trust the certificate if the name on the certificate does not match the name of the organization or person you expect.
expiration date - Most certificates are issued for one or two years. One exception is the certificate for the certificate authority itself, which, because of the amount of involvement necessary to distribute the information to all of the organizations who hold its certificates, may be ten years. Be wary of organizations with certificates that are valid for longer than two years or with certificates that have expired.

When visiting a web site, you may have been presented with a dialog box that claims that there is an error with the site certificate. This may happen if the name the certificate is registered to does not match the site name, you have chosen not to trust the company who issued the certificate, or the certificate has expired. You will usually be presented with the option to examine the certificate, after which you can accept the certificate forever, accept it only for that particular visit, or choose not to accept it. The confusion is sometimes easy to resolve (perhaps the certificate was issued to a particular department within the organization rather than the name on file). If you are unsure whether the certificate is valid or question the security of the site, do not submit personal information. Even if the information is encrypted, make sure to read the organization's privacy policy first so that you know what is being done with that information [9].

2.7. Weakness of Current Browsers

2.7.1. Comparison of Web Browsers and its Vulnerabilities

Here's a summary of each of the browser's features [10]:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Firefox Beta 1</th>
<th>2Internet 7 Beta 3</th>
<th>ExplorerOpera 9.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tabbed browsing</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Add-ins</td>
<td>Yes—Extensions</td>
<td>Yes—Add-Ons</td>
<td>Widgets</td>
</tr>
<tr>
<td>Themes</td>
<td>Yes</td>
<td>No</td>
<td>Yes—Skins</td>
</tr>
<tr>
<td>Built-in search with multiple engine choice</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Pop-up blocker</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Feature</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>Anti-Phishing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>One-Click Button to add No</td>
<td>Yes</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>RSS reader</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Download manager</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Can remember open tabs for next session</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Save group of tabs as bookmark</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Thumbnails for tabs</td>
<td>No</td>
<td>Yes—Quick Tabs</td>
<td>Yes—Mouse over tab; also can tile tabs</td>
</tr>
<tr>
<td>Macintosh/Linux version</td>
<td>Yes/Yes</td>
<td>No/No</td>
<td>Yes</td>
</tr>
<tr>
<td>BitTorrent client</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Spell-checker for text boxes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Download pause</td>
<td>Yes</td>
<td>No</td>
<td>Yes (stop and resume transfer)</td>
</tr>
</tbody>
</table>

Table 1 Various Browser Features

2.7.1.1. Security in IE

IE has been possibly the biggest target of security attacks over the last decade, with updates to cover holes a regular occurrence. Since Bill Gates's edict demanding "trustworthy computing" in 2002, Microsoft has been making big efforts towards eliminating these security holes, and IE 7 is intended to be a model child of this initiative. Defender, now in beta 2, is the major salvo in this direction; it's anti-spyware software that both finds spyware on your system and monitors for it in real time while you browse with IE .

According to Microsoft, the following security features are to be found in IE 7:

- ActiveX Opt-in
- Security Status Bar
- Phishing Filter
- Cross-Domain Barriers
- Delete Browsing History
- Address Bar Protection
- International Domain Name Anti-spoofing
- URL Handling Security
And when Windows Vista rolls around, they also plan to add a Protected Mode (where IE runs in its own sandbox) and Parental Controls. When went to some questionable sites, we could see the little animated icon in the lower right doing its job; hovering the cursor over it yielded the tooltip: "Phishing Filter is checking this website." It also caught a website that was trying to install an add-in.

In his column of a couple months ago, Jason Cross has said that IE7 Isn't Good Enough. His main complaints are that the new browser has taken too long to launch, and that it doesn't support the middle mouse button. This reviewer agrees that Microsoft should not have sat on its market-share laurels for five years without producing a new version of IE, but that doesn't mean the new browser doesn't include major advances. If the Firefox 2 beta is any indication, that browser won't be wowing anyone with incredible new features that would send it way ahead of the new IE version any time soon. As to the middle mouse button, it is a nice feature, but we're not sure what percentage of users even know of its existence.

Microsoft doesn't expect to convert the Firefox religious, but rather to bring over those who might have abandoned IE because of its security problems and lack of a tabbed interface. IE7 is a fine effort in this direction [11].

2.7.2. Web browser security Summary

The information was collected from Secunia, a leading computer software security monitoring company. These statistics cover all reported vulnerabilities in Windows versions of Internet Explorer, Firefox, and Opera.

2.7.3. Vulnerabilities

The following table details the number of vulnerabilities and relative danger.

Historical cumulative values are provided in three forms: for all vulnerabilities in the entire of life of these products, for all vulnerabilities that were present within the first 365 days of the first vulnerability reported in the product, and for all vulnerabilities that were present within the last 365 days.
“High severity” values include vulnerability reports that were marked as “highly critical” and above. Relative danger levels are calculated by adding up the square of the criticality levels for each vulnerability report (not critical=1², extremely critical=5²).

Vulnerability is considered unfixed if the vulnerability report does not have a complete vendor patch.

**Notice:** Since Internet Explorer 7 was released, Secunia has not indicated which previously known unfixed Internet Explorer 6 vulnerabilities have been re-tested in IE 7, aside from the most recent few (which, by the way, were confirmed to still affect the new version). Secunia has a history of not listing very old vulnerabilities under new versions even if they still apply, and this is as true with Firefox and Opera as is assumed with Internet Explorer. Until Secunia updates the old advisories with an indication of the status in IE 7, this page will assume they still exist [12].

<table>
<thead>
<tr>
<th>Security vulnerabilities</th>
<th>Internet Explorer</th>
<th>Firefox</th>
<th>Opera</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Historical cumulative values (Product life)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vulnerability reports</td>
<td>126</td>
<td>64</td>
<td>60</td>
</tr>
<tr>
<td>High severity vulnerability reports</td>
<td>59</td>
<td>21</td>
<td>15</td>
</tr>
<tr>
<td>Vulnerability issues</td>
<td>250</td>
<td>202</td>
<td>73</td>
</tr>
<tr>
<td>Relative danger</td>
<td>1403</td>
<td>573</td>
<td>500</td>
</tr>
<tr>
<td><strong>Historical cumulative values (from first 365 days)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vulnerability reports</td>
<td>31</td>
<td>20</td>
<td>18</td>
</tr>
<tr>
<td>High severity vulnerability reports</td>
<td>13</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Vulnerability issues</td>
<td>69</td>
<td>39</td>
<td>23</td>
</tr>
<tr>
<td>Relative danger</td>
<td>331</td>
<td>156</td>
<td>138</td>
</tr>
<tr>
<td><strong>Historical cumulative values (from last 365 days)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vulnerability reports</td>
<td>48</td>
<td>19</td>
<td>7</td>
</tr>
<tr>
<td>High severity vulnerability reports</td>
<td>9</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Vulnerability issues</td>
<td>66</td>
<td>44</td>
<td>11</td>
</tr>
<tr>
<td>Relative danger</td>
<td>318</td>
<td>144</td>
<td>88</td>
</tr>
</tbody>
</table>
## Security vulnerabilities

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Internet Explorer</th>
<th>Firefox</th>
<th>Opera</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Highest values at one time</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vulnerability reports</td>
<td>37</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>High severity vulnerability reports</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Vulnerability issues</td>
<td>39</td>
<td>13</td>
<td>8</td>
</tr>
<tr>
<td>Relative danger</td>
<td>204</td>
<td>44</td>
<td>27</td>
</tr>
<tr>
<td><strong>Mean average per day (from last 365 days)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vulnerability reports</td>
<td>35.43</td>
<td>6.46</td>
<td>0.01</td>
</tr>
<tr>
<td>High severity vulnerability reports</td>
<td>1</td>
<td>0.03</td>
<td>0</td>
</tr>
<tr>
<td>Vulnerability issues</td>
<td>37.22</td>
<td>7.46</td>
<td>0.01</td>
</tr>
<tr>
<td>Relative danger</td>
<td>155.83</td>
<td>24.21</td>
<td>0.05</td>
</tr>
<tr>
<td><strong>Median average per day (from last 365 days)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vulnerability reports</td>
<td>35</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>High severity vulnerability reports</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Vulnerability issues</td>
<td>37</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Relative danger</td>
<td>155</td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td><strong>Present values</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vulnerability reports</td>
<td>36</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>High severity vulnerability reports</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Vulnerability issues</td>
<td>38</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Relative danger</td>
<td>156</td>
<td>24</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2 Various Browsers Security Vulnerabilities

Internet Explorer has had 126 vulnerability reports. 22 were marked as moderately critical, 44 were marked as highly critical, and 15 were marked as extremely critical. There are still 36 remaining, including 9 that were marked as moderately critical and 1 that was marked as highly critical.

Firefox has had 64 vulnerability reports. 19 were marked as moderately critical, 21 were marked as highly critical, and 0 were marked as extremely critical. There are still 7 remaining, including 1 that was marked as moderately critical.
Opera has had 60 vulnerability reports. 19 were marked as moderately critical, 14 were marked as highly critical, and 1 was marked as extremely critical. All of the reported vulnerabilities have since been fixed.

**2.7.3.1. Publicly disclosed without a patch**

Many vulnerabilities are discovered by the browser vendors and patched before they are ever publicly known. Vulnerabilities are most dangerous when they are found elsewhere with no patch available. The following are historical cumulative vulnerability values that only include those vulnerabilities that were publicly known before a patch was available.

<table>
<thead>
<tr>
<th>Security vulnerabilities (in public)</th>
<th>Internet Explorer</th>
<th>Firefox</th>
<th>Opera</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Historical cumulative values (Product life)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vulnerability reports</td>
<td>92</td>
<td>38</td>
<td>30</td>
</tr>
<tr>
<td>High severity vulnerability reports</td>
<td>28</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Vulnerability issues</td>
<td>122</td>
<td>50</td>
<td>35</td>
</tr>
<tr>
<td>Relative danger</td>
<td>849</td>
<td>249</td>
<td>212</td>
</tr>
<tr>
<td><strong>Historical cumulative values (from first 365 days)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vulnerability reports</td>
<td>22</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>High severity vulnerability reports</td>
<td>5</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Vulnerability issues</td>
<td>31</td>
<td>17</td>
<td>19</td>
</tr>
<tr>
<td>Relative danger</td>
<td>185</td>
<td>100</td>
<td>105</td>
</tr>
<tr>
<td><strong>Historical cumulative values (from last 365 days)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vulnerability reports</td>
<td>39</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>High severity vulnerability reports</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Vulnerability issues</td>
<td>41</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>Relative danger</td>
<td>168</td>
<td>58</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 3 Various Browsers Security Vulnerabilities without patch
Internet Explorer has had 92 reports of vulnerabilities discovered in the public without a patch. 20 were marked as moderately critical, 17 were marked as highly critical, and 11 were marked as extremely critical.

Firefox has had 38 reports of vulnerabilities discovered in the public without a patch. 11 were marked as moderately critical, 6 were marked as highly critical, and 0 were marked as extremely critical.

Opera has had 30 reports of vulnerabilities discovered in the public without a patch. 10 were marked as moderately critical, 3 were marked as highly critical, and 1 was marked as extremely critical.

### 2.7.3.2. Fully-Disclosed

The following values only include vulnerabilities that had publicly known exploits or proof-of-concept exploit code before a patch was available, according to Secunia's advisories.

It should be noted that not all theoretical exploits hold the same likelihood of attack. Some vulnerabilities may have publicly available proof-of-concept code that is very difficult to exploit in practice. Criticality levels often provide some indication of the ease of exploitation, but they also represent the sheer potential impact of the flaw whether easily exploitable or not.

<table>
<thead>
<tr>
<th>Security vulnerabilities (fully-disclosed)</th>
<th>Internet Explorer</th>
<th>Firefox</th>
<th>Opera</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Historical cumulative values (Product life)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vulnerability reports</td>
<td>42</td>
<td>12</td>
<td>9</td>
</tr>
<tr>
<td>High severity vulnerability reports</td>
<td>17</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Vulnerability issues</td>
<td>62</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>Relative danger</td>
<td>469</td>
<td>70</td>
<td>66</td>
</tr>
<tr>
<td><strong>Historical cumulative values (from first 365 days)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vulnerability reports</td>
<td>6</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>High severity vulnerability reports</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Aspect</td>
<td>Internet Explorer</td>
<td>Firefox</td>
<td>Opera</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------------------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>Vulnerability issues</td>
<td>6</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>Relative danger</td>
<td>41</td>
<td>13</td>
<td>30</td>
</tr>
</tbody>
</table>

**Historical cumulative values (from last 365 days)**

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Internet Explorer</th>
<th>Firefox</th>
<th>Opera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vulnerability reports</td>
<td>16</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>High severity vulnerability reports</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Vulnerability issues</td>
<td>17</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Relative danger</td>
<td>58</td>
<td>40</td>
<td>4</td>
</tr>
</tbody>
</table>

**Highest values at one time**

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Internet Explorer</th>
<th>Firefox</th>
<th>Opera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vulnerability reports</td>
<td>15</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>High severity vulnerability reports</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Vulnerability issues</td>
<td>16</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>Relative danger</td>
<td>107</td>
<td>31</td>
<td>25</td>
</tr>
</tbody>
</table>

**Mean average per day (from last 365 days)**

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Internet Explorer</th>
<th>Firefox</th>
<th>Opera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vulnerability reports</td>
<td>13.78</td>
<td>4.28</td>
<td>0.01</td>
</tr>
<tr>
<td>High severity vulnerability reports</td>
<td>0</td>
<td>0.03</td>
<td>0</td>
</tr>
<tr>
<td>Vulnerability issues</td>
<td>14.58</td>
<td>4.28</td>
<td>0.01</td>
</tr>
<tr>
<td>Relative danger</td>
<td>49.85</td>
<td>10.83</td>
<td>0.05</td>
</tr>
</tbody>
</table>

**Median average per day (from last 365 days)**

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Internet Explorer</th>
<th>Firefox</th>
<th>Opera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vulnerability reports</td>
<td>13</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>High severity vulnerability reports</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Vulnerability issues</td>
<td>14</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Relative danger</td>
<td>49</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

**Present values**

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Internet Explorer</th>
<th>Firefox</th>
<th>Opera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vulnerability reports</td>
<td>13</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>High severity vulnerability reports</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Vulnerability issues</td>
<td>14</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Relative danger</td>
<td>46</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4 Various Browsers Security Vulnerabilities fully disclosed
Internet Explorer has had 42 fully-disclosed vulnerability reports. 5 were marked as moderately critical, 6 were marked as highly critical, and 11 were marked as extremely critical. There are still 13 remaining, including 3 that were marked as moderately critical.

Firefox has had 12 fully-disclosed vulnerability reports. 2 were marked as moderately critical, 2 were marked as highly critical, and 0 were marked as extremely critical. There are still 4 remaining, all of which were marked as less critical or not critical.

Opera has had 9 fully-disclosed vulnerability reports. 3 were marked as moderately critical, 0 were marked as highly critical, and 1 was marked as extremely critical. All of the reported vulnerabilities have since been fixed.

### 2.7.3.3. Patch Delay

It is also important to consider how quickly each web browser fixes its vulnerabilities. The following table lists the average time taken between Secunia's vulnerability reports and the release dates of their respective patches, if all aging unfixed vulnerabilities (vulnerabilities at least as old as the mean of all fixed vulnerabilities for that browser) were to be fixed today. Data does not include unfixed vulnerabilities less than that age, vulnerabilities with unknown fix dates, or vulnerabilities that were only publicly known after the patch release. Values listed are in days.

<table>
<thead>
<tr>
<th>Patch delay (in days)</th>
<th>Average</th>
<th>Internet Explorer</th>
<th>Firefox</th>
<th>Opera</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Per vulnerability report</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall mean</td>
<td></td>
<td>580</td>
<td>155</td>
<td>76</td>
</tr>
<tr>
<td>Overall median</td>
<td></td>
<td>371</td>
<td>44</td>
<td>29</td>
</tr>
<tr>
<td>High severity mean</td>
<td></td>
<td>135</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>High severity median</td>
<td></td>
<td>53</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td><strong>Per vulnerability issue</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall mean</td>
<td></td>
<td>488</td>
<td>142</td>
<td>118</td>
</tr>
<tr>
<td>Overall median</td>
<td></td>
<td>209</td>
<td>23</td>
<td>44</td>
</tr>
<tr>
<td>Patch delay (in days)</td>
<td>Internet Explorer</td>
<td>Firefox</td>
<td>Opera</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------</td>
<td>---------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High severity mean</td>
<td>124</td>
<td>17</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>High severity median</td>
<td>65</td>
<td>23</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Weighted by relative danger</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall mean</td>
<td>437</td>
<td>140</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>Overall median</td>
<td>121</td>
<td>23</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>High severity mean</td>
<td>129</td>
<td>11</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>High severity median</td>
<td>53</td>
<td>8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Per fully-disclosed vulnerability report</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall mean</td>
<td>431</td>
<td>148</td>
<td>77</td>
<td></td>
</tr>
<tr>
<td>Overall median</td>
<td>121</td>
<td>63</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>High severity mean</td>
<td>60</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>High severity median</td>
<td>50</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Table 5 Various Browsers patch delay in days

2.7.3.4. Total Vulnerabilities

The following graphs include all vulnerabilities listed by Secunia.

Number of open advisories versus time (2004-02-09 to 2008-01-01). Peak: 37

Figure 2 Number of open advisories versus time
2.7.3.5. Fully-disclosed Vulnerabilities

The following graphs only include vulnerabilities that had publicly known exploits or proof of concept exploit code before a patch was available, according to Secunia's advisories.
2.8. **Web Browser’s basic Security Settings**

2.8.1. **Why Secure Your Web Browser?**

Today, web browsers such as Internet Explorer, Mozilla Firefox, and Safari (to name a few), are installed on almost all computers. Because web browsers are used so frequently, it is vital to configure them securely. Often, the web browser that comes with an operating system is not set up in a secure default configuration. Not securing your web browser can lead quickly to a variety of computer problems caused by anything from spyware being installed without your knowledge to intruders taking control of your computer.
Ideally, computer users should evaluate the risks from the software they use. Many computers are sold with software already loaded. Whether installed by a computer manufacturer, operating system maker, internet service provider, or by a retail store, the first step in assessing the vulnerability of your computer is to find out what software is installed and how one program will interact with another. Unfortunately, it is not practical for most people to perform this level of analysis.

There is an increasing threat from software attacks that take advantage of vulnerable web browsers. In recent months, the CERT/CC has observed a trend whereby new software vulnerabilities are exploited and directed at web browsers through the use of compromised or malicious web sites. This problem is made worse by a number of factors, including the following:

- Many web browsers are configured to provide increased functionality at the cost of decreased security.
- New security vulnerabilities may have been discovered since the software was configured and packaged by the manufacturer.
- Many web sites require that users enable certain features or install more software, putting the computer at additional risk.
- Many users do not know how to configure their web browsers securely.
- Many users are unwilling to enable or disable functionality as required to secure their web browser.
- Many users are unaware whether or not their computer has been compromised.
- Many users fail to properly “clean” a compromised computer.

As a result, exploiting vulnerabilities in web browsers has become a popular way for attackers to compromise computer systems [13].

### 2.8.2. Understanding Web Browser Features

It is important to understand the functionality and features of the web browser you use. Enabling some web browser features may lower security. For example, the ActiveX software
feature has a history of vulnerabilities that have lead to severe security impacts when enabled.

Multiple web browsers may be installed on your computer. Other software applications on your computer, such as email clients or document viewers, may use a different browser than the one you normally use to access the web. Also, certain file types may be configured to open with a different web browser. Using one web browser to access web sites does not mean other applications will automatically use the same browser. For this reason, it is important to securely configure each web browser installed on your computer.

Web sites may require the use of a browser that supports scripting or active content, such as JavaScript or ActiveX controls, or the sites themselves may contain vulnerabilities. Web sites can be considered products, and as a user of the product, you can contact the web site administrators and request that the sites be designed so that they do not require the use of features that may pose a computer security risk.

Some specific web browser features and attributes are described in this document. Understanding what different features do will help you understand how they affect your web browser's functionality and the security of your computer.

**ActiveX** is a technology used by Microsoft Internet Explorer on Microsoft Windows. ActiveX allows applications or parts of applications to be utilized by the web browser. A web page can use ActiveX components that may already reside on a Windows system, or may download the component from a web site. This gives extra functionality to traditional web browsing, but may also introduce more severe vulnerabilities if not properly implemented.

**Java** is an object-oriented programming language that can be used to develop active content for web sites. A Java Virtual Machine, or JVM, is used to execute the Java code, or “applet,” provided by the web site. The JVM is designed to separate, or “sandbox,” running code so that it does not affect the rest of the system. Some operating systems come with a JVM, while others require a JVM to be installed before Java can be used. Java applets run independently from the operating systems.
**Active Content, or plug-ins**, are intended for use in the web browser. They are similar to ActiveX controls but cannot be executed outside of a web browser. Macromedia Flash is an example of Active Content that can be provided as a plug-in.

**JavaScript** is a dynamic scripting language that is used to develop active content for web sites. Unlike Java, JavaScript is a language that is interpreted by the web browser directly. There are specifications in the JavaScript standard that restrict certain features such as accessing local files.

**VBScript** is a programming language that is unique to Microsoft Windows. VBScript is similar to JavaScript, but it is not as widely used in web sites because of its limited compatibility with browsers other than Internet Explorer.

**Cookies** are text files placed on your computer to store data that is used by a web site. A cookie can contain any information that a web site is designed to place in it. Cookies may contain information about the sites you visited, or may even contain credentials for accessing the site. Cookies are designed to be readable only by the web site that created them.

**Security Zones and the Domain Model** are methods Microsoft Windows uses designed to provide multiple levels of security settings for a single system. While primarily used by Internet Explorer, it can be invoked by other applications on the system that use components of Internet Explorer. You can learn more about Microsoft’s Security Zones, the Domain Model, and how to secure them at this web site: http://www.microsoft.com/windows/ie/using/howto/security/setup.asp.

### 2.8.3. Vulnerabilities and Attack Vectors

Increasingly, attackers are exploiting client-side systems (your computer) through various vulnerabilities. They use these vulnerabilities to take control of your computer, steal your information, destroy your files, and attack other computers. A low-cost way for attackers to gain control of your computer is by exploiting vulnerabilities in web browsers. An attacker can simply create a malicious web page that will install Trojan software or spyware that will steal information from your computer. Rather than actively targeting and attacking vulnerable
systems, a malicious web site can passively compromise systems as the site is visited. A malicious HTML document can also be emailed to victims. In these cases, the act of opening the email or attachment can compromise the system.

In this section, we will point out some common vulnerabilities in web sites and web browsers that tend to be exploited. We will not go into great detail in this document, but will provide links to other documentation that will help explain the vulnerabilities.

2.8.3.1. ActiveX Controls

ActiveX is a technology that has been plagued with various vulnerabilities and implementation issues. One problem with using ActiveX in a web browser is that it greatly increases the attack surface, or “attackability,” of a system. Vulnerabilities in ActiveX objects may be exploited via Internet Explorer, even if the object was never designed to be used in a web browser. Many vulnerabilities associated with ActiveX controls lead to severe impacts. Attackers exploiting ActiveX vulnerabilities can frequently gain control of computers.

2.8.3.2. Java

Java is an object-oriented programming language developed by Sun Microsystems. A Java applet is machine-independent and requires a Java Virtual Machine (JVM) on the client computer so that it can execute. Java applets traditionally execute within a “sandbox” where the interaction with the rest of the system is limited. However, various implementations of the JVM contain vulnerabilities that allow an applet to bypass these restrictions. Signed Java applets can also bypass sandbox restrictions, but they generally prompt the user before they can execute.

2.8.3.3. Cross-Site Scripting

Cross-site scripting, often referred to as CSS or XSS, is a vulnerability in a web site that permits an attacker to leverage the trust relationship that you have with that site. Note that cross-site scripting is not usually caused by a failure in the web browser.

2.8.3.4. Cross-Zone and Cross-Domain Vulnerabilities

Most web browsers employ security models to prevent a web site from accessing data in a different domain. These security models are primarily based on the Netscape Same Origin

Vulnerabilities in these security models can be used to perform actions that a site could not normally perform. The impact can be similar to a cross-site scripting vulnerability. However, if vulnerability allows for an attacker to cross into the local machine zone or other protected areas; the attacker may be able to execute arbitrary commands on the vulnerable system.

### 2.8.3.5. Malicious Scripting, Active Content, and HTML

Some sites may contain malicious scripts, active content, or HTML that will attempt to trick the visitor into providing information, or performing an action that will enable the attacker to gain some privilege. In the absence of vulnerabilities, the attackers rely on social engineering to gain access to the victim’s information. However, vulnerabilities in web browsers may be exploited to gain privileges as well.

### 2.8.3.6. Spoofing

As it relates to web browsers, spoofing is a term used to describe methods of faking various parts of the browser user interface. This may include the address or location bar, the status bar, the padlock, or other user interface elements. Phishing attacks often utilize some form of spoofing to help convince the user to provide personal information. If a user's browser is vulnerable to spoofing, they are more likely to fall victim to a phishing attack.

### 2.9. How to Secure Your Web Browser

Some software features that provide functionality to a web browser, such as ActiveX, Java, Scripting (JavaScript, VBScript, etc), may also introduce vulnerabilities to the computer system. These may stem from poor implementation of the protocol, poor design, poorly written software, or an insecure configuration. For these reasons, you should understand which browsers support which features and the subsequent risks they could introduce. Some web browsers permit you to fully disable the use of these technologies, while others may only permit you to reduce functionality.

This section shows you how to securely configure a few of the most popular web browsers
and how to disable features that can cause vulnerabilities. We encourage you to visit the web site for the browser you use to learn more.

Web browsers are frequently updated. Depending on the version of your software, the features and options may move or change.

2.9.1. Microsoft Internet Explorer

Microsoft Internet Explorer (IE) is a web browser integrated into the Microsoft Windows operating system. Removal of this application is not practical.

In addition to supporting Java, scripting and other forms of active content, Internet Explorer implements ActiveX technology. While any application is potentially vulnerable to attack, it is possible to mitigate a number of serious vulnerabilities by using a web browser that does not support ActiveX controls. However, using an alternate browser may affect the functionality of some sites that require the use of ActiveX controls. Note that using a different web browser will not remove IE or other Windows components from the system. Other software, such as email clients, may invoke IE, the WebBrowser ActiveX control, or the IE HTML rendering engine (MSHTML).

Here are steps to disable various features in Internet Explorer. Note that menu options may vary between versions of IE, so you should adapt the steps below as appropriate.

In order to change settings for Internet Explorer, select **Tools** then **Internet Options**…

![Image of Internet Options menu]

Figure 10 Selecting Internet Options to manage Security Settings in IE
Select the **Security** tab. On this tab you will find a section at the top, which lists the various security zones that Internet Explorer uses. More information about Internet Explorer security zones is available in the Microsoft document Setting Up Security Zones. For each of these zones, you can select a Custom Level of protection. By clicking the **Custom Level** button, you will see a second window open that permits you to select various security settings for that zone. The **Internet** zone is where all sites initially start out. The security settings for this zone apply to all the web sites that are not listed in the other security zones. We recommend the **High** security setting be applied for this zone. By selecting the High security setting, several features including ActiveX, Active scripting, and Java will be disabled. With these features disabled, the browser will be more secure. Click the **Default Level** button and then drag the slider control up to **High**.

![Figure 11 Selecting Security to manage Security Settings in IE](image.jpg)

For a more fine-grained control over what features are allowed in the zone, click the **Custom Level** button. Here you can control the specific security options that apply to the current zone.
Default values for the High security setting can be selected by choosing **High** and clicking the **Reset** button to apply the changes.

![Security Settings](image)

**Figure 12 Managing Security Level in IE**

**Trusted sites** is a security zone for web sites that you believe are securely designed and contain trustworthy content. To add or remove sites from this zone, you can click the **Sites** button. This will open a new window that will list the sites that you trust and permit you to add or remove sites. You may also require that only sites with Secure Sockets Layer (SSL) implemented can be active in this zone. This permits you to verify that the site you are visiting is the site that it claims to be.
We recommend setting the security level for the **Trusted sites** zone to **Medium**. When the Internet Zone is set to **High**, you may encounter web sites that do not function properly due to one or more of the associated security settings. This is where the **Trusted sites** zone can help.
If you trust that the site will not contain malicious code, you can add it to the list of sites in the Trusted sites zone. Once a site is added to this zone, features such as ActiveX and active scripting will be enabled. The benefit of this type of configuration is that IE will be more secure by default, and sites can be “whitelisted” in the Trusted sites zone to gain extra functionality.

The Privacy tab contains settings for cookies. Cookies are text files placed on your computer by various sites that you visit either directly (first-party) or indirectly (third-party) through ad banners, for example. A cookie can contain any data that a site wishes to store. It is often used to track your computer as you move through a web site and store information such as preferences or credentials. We recommend that you select the Advanced button and select Override automatic cookie handling. Then select Prompt for both first and third-party cookies. This will prompt you each time a site tries to place a cookie on your computer. You can then evaluate the originating site, whether you wish to accept or deny the cookie, and what action to take in the future (always accept, always block, or continue to ask).

![Internet Options window showing Privacy tab settings](image)

Figure 15 Pop-up blocking and privacy in IE
By selecting the **Sites...** button, you can manage the cookie settings for specific sites. You can add or remove sites, and you can change the current settings for existing sites. The bottom section of this window will specify the domain of the site and the action to take when that site wants to place a cookie on your computer. You can use the upper section of this window to change these settings.

The **Advanced** tab contains settings used by all zones. The settings contained in the **Multimedia** section have features that you can adjust to protect against some potential...
vulnerabilities. For instance, attackers may be able to track your usage or exploit the software you use to play multimedia data. We recommend disabling the options to play sounds and videos.

![Internet Options Image]

Figure 18 Managing Advanced Settings in IE

Under the Programs tab, you can specify your default applications for viewing web sites, email messages, and other network related tasks. You can also prevent Internet Explorer from showing you a message asking to be your default web browser.
2.9.2. Mozilla Firefox

Mozilla Firefox supports many of the same features as Internet Explorer, with the exception of ActiveX and the Security Zone model. We recommend looking in the Help, For Internet Explorer Users menu to understand the different terminology used by the two browsers.

Following are steps to disable various features in Mozilla Firefox. Note that some menu options may change between versions or may appear in different locations depending on the host operating system. You should adapt the steps below as appropriate.

To edit the settings for Mozilla Firefox, select Tools, then Options.
You will then see an Options window that has a row of categories along the top. The first category of interest is the **General** category. Under this section, for instance, you can set Firefox as your default browser.
Under the Privacy category, you can select the Cookies subcategory. Here you can disable cookies or change your preferences for how the browser handles them. In general, we recommend enabling cookies for the original site only. Additionally, by enabling the option unless I have removed cookies set by the site, a web site can be “blacklisted” from setting cookies when its cookies are removed manually.

![Figure 22 Managing Cookies in Mozilla Firefox](image)

Many web browsers will allow you to store login information. In general, we recommend against using such features. Should you decide to use the feature, ensure that you use the measures available to protect the password data on your computer. Under the Privacy category, the Passwords subcategory contains various options to manage stored passwords, and a Master Password feature to encrypt the data on your system. We encourage you to use this option if you decide to let Mozilla Firefox manage your passwords.
The **Content** category has an option to **Enable Java**. Java is a programming language that permits web site designers to run applications on your computer. We recommend disabling this feature unless required by the site you wish to visit. Again, you should determine if this site is trustworthy and whether you want to enable Java to view the site’s content. After you are finished visiting the site, we recommend disabling Java until you need it again.

The **Warn me when web sites try to install extensions or themes** option will display a warning bar at the top of the browser when a web site attempts to take such an action.

Press the **Advanced** button to disable specific JavaScript features. We recommend disabling all of the options displayed in this dialog.
Figure 24 Managing Advanced Content Settings is Mozilla Firefox.

The Downloads section has an option to modify actions taken when files are downloading. Any time a file type is configured to open automatically with an associated application, this can make the browser more dangerous to use. Vulnerabilities in these associated applications can be exploited more easily when they are configured to open automatically. Click the View & Edit Actions button to view the current download settings and modify them if necessary.
Figure 25 Managing download settings in Mozilla Firefox

The Download Actions dialog shows the file types and the actions the browser will perform when it encounters a given file type. For any file type listed, click on either Remove Action or Change Action... If you click on Change Action..., select Save them on my computer to save files of that type to the computer. This helps prevent automated exploitation of vulnerabilities that may exist in these applications.
Firefox 1.5 includes a feature to **Clear Private Data**. This option will remove potentially sensitive information from the web browser. Select **Clear Private Data...** from the **Tools** menu to use this privacy feature.
2.9.3. Keeping Your Computer Secure

In addition to selecting and securing your web browser, you can take other steps to protect your computer.

2.9.3.1. Install and use antivirus software

While an up-to-date antivirus software package cannot protect against all malicious code, for most users it remains the best first-line of defence against malicious code attacks. Many antivirus packages support automatic updates of virus definitions. We recommend using these
automatic updates when available. A partial list of antivirus vendors is available on the CERT/CC web site [15].

2.9.3.2. Enable automatic software updates if available

Vendors will usually release patches for their software when a vulnerability has been discovered. Most product documentation tells you how to get updates and patches. You should be able to obtain updates from the vendor's web site. Read the manuals or browse the vendor's web site for more information.

Some applications will automatically check for available updates, and many vendors offer automatic notification of updates via a mailing list. Look on your vendor's web site for information about automatic notification. If no mailing list or other automated notification mechanism is offered, you may need to check the vendor's web site periodically for updates.

2.9.3.3. Avoid unsafe behavior

Additional information on this topic can be found in the document Home Network Security.

- Use caution when opening email attachments or when using peer-to-peer file sharing, instant messaging, or chat rooms.
- Don't enable file sharing on network interfaces exposed directly to the internet.

2.9.3.4. Follow the principle of least privilege — don't enable it if you don't need it

Consider creating and using an account with limited privileges instead of an “administrator” or “root” level account for everyday tasks. Depending on the operating system, you only need to use administrator-level access when installing new software, changing system configurations, and other important tasks. Many vulnerability exploits (e.g., viruses, Trojan horses) are executed with the privileges of the user that runs them — making it far more risky to be logged in as an administrator all the time.
Chapter 3

3. Idea and Concepts

For further implementation and developing security for web browsers instead of basic security settings, there is need to analyze the main problem with security for cookies.

3.1. Making Browser more secure

The main defect in current browsers are the problem with stored cookies and passwords without security and the user can face threat by another user who copies used for his personal purpose when the browser was in active and the system user is also in active means when he is out of system, so there is more threat for user if there is any secure data in cookies like bank information, credit card information, login and id passwords for various accounts, so that it can be copied and used by any enemies or any unknown third person, so there is much need to protect these cookies and passwords. So what the next idea is to protect cookie files by using encryption and decryption technologies, is there any possibility by developing any software using any programming language to do encryption and decryption, but is it safe if we do normal encryption and decryption, but the problem with these encryption and decryption is user have to give some key to do all these whether it is public-key or private-key encryption, but the key distribution is also a problem, and there is no secure way to store key which take care of this whole process and to generate a key using random generators which is also a problem if there is innumerable of users, so for this we need a software or hardware which takes care of storing secure keys and to do encryption and decryption and if possible it have to protect the encrypt data, so if we see all the possibilities what we have up to now, there is no software which protects the data entirely with out any attack from viruses or from hackers, so for this the software also must need a help from hardware to take care of the entire process like storing keys and if possible to store the encrypted data so to take care of the software and browser cookies, so for this I am going to implement Trusted Computing (TC) concepts developed by Trusted Computing Group in software developed by me to protect cookies and passwords of a browser by taking the help of a microcontroller chip called TPM (Trusted
What is Trusted Computing, what are its concepts, what TPM is and how TPM implementing the TC concepts are explained in following sections.

Before knowing TC concepts, we have to know the specifications mentioned by TCG (Trusted Computing Group) which are implemented by TPM (Trusted Platform Module) to do encryption process.

The following sections cover the types of encryption process and hashing functions which are implemented by TPM for encryption which are indirectly used by our software to protect browser cookies.

3.2. Concepts used by TPM for Encryption

3.2.1. Data Security

Data security is the means of ensuring that data is kept safe from corruption and that access to it is suitably controlled. Thus data security helps to ensure privacy. It also helps in protecting personal data.

3.2.2. Cryptography

Cryptography is a field of science and research in which cryptographers engage in the design and development of cryptographic systems, systems that can protect sensitive data from hackers, eavesdroppers, and industrial spies. Cryptographic methods are also used for authentication between users and between computer systems. Cryptographers actively attempt to break the very systems they create in order to understand their limitations. The concept of breaking something that you have created is common in manufacturing. It proves the reliability and safety of a product such as an automobile. Today, a common practice is to enlist public help in breaking cryptographic schemes by offering prizes in the form of money and "prestige" for having broken a scheme.

The data transferred from one system to another over public network can be protected by the method of encryption. On encryption the data is encrypted/ scrambled by any Encryption
algorithm using the ‘key’. Encryption transforms some input into an output that is impossible to read without the proper key. It is performed by running an algorithm that transforms some input called the plaintext into an encrypted form called the ciphertext. While the algorithm always operates the same way, the use of a key ensures that the output will always be different (given the same input). A different key used on the same plaintext will produce different ciphertext. The key is also used to unlock the encrypted data by using the same algorithm in reverse. Because algorithms are usually public and well known, good encryption relies on a solid algorithm and avoiding the use of weak keys.

There are three primary cryptographic techniques. Two are used to encrypt text, graphics, and other information in a form that can be recovered by someone who has an appropriate key. The third, used in authentication and integrity schemes, scrambles input without any intention to recover it [14].

**Secret-key cryptography:** A single key is used to encrypt and decrypt information. This technique is called symmetric key encryption. Encrypted information may be stored on disk or transmitted over non-secure channels. Since there is only one key, some form of secure key exchange is necessary (in-person, courier, and so on).

**Public-key cryptography:** Two keys are used in this scheme—one to encrypt and one to decrypt. Thus, the scheme is asymmetric. Every person has a set of keys and one is held private while the other is made publicly available. To send a private message to someone, you encrypt it with the recipient's public key. The recipient then decrypts it with his or her private key. This eliminates the problems of exchanging keys in advance of using the encryption.

**Hash functions:** A hash function is an algorithm that produces a unique "fingerprint" of a message that can prove that it has not been altered since its creation. The output of the algorithm is called a message digest. A recipient that runs the same algorithm on the message should arrive at the same digest; otherwise, the message is suspect. The technique is used to digitally sign messages and documents.

The cryptology focuses its attention on the design and evaluation of a wide range of methods and techniques for information protection. Information protection covers not only secrecy but also authentication integrity, verifiability, non-repudiation and other specific security issues.
The part of cryptology that deals with the design of algorithms, protocols and systems which are used to protect information against specific threats is called cryptography.

To incorporate information protection into a system, protocol, or service, the designer needs to know [16]:

- A detailed specification of the environment in which the system (protocol or service) is going to work, including a collection of security goals,
- A list of threats together with the description of places in the system where adverse tampering with the information flow can occur,
- The level of protection required or amount of power (in term of accessible computing resources) that is expected from an attacker (or adversary), and
- The projected life span of the system.

Cryptography provides the tools to implement the information protection by including tools like encryption algorithms, authentication codes, one-way functions, hashing functions, secret sharing schemes, signature schemes, pseudorandom bit generators, zero-knowledge proof systems, etc.

Each cryptographic tool is characterized by its security specification which usually indicated the recommended configuration and its strength against specific strengths such as eavesdropping and illegal modification of information.

The basic functions provided by cryptography to achieve security are [17]:

- Confidentiality – Users want to be assured that their data are not intercepted and read by unauthorized individuals.
- Integrity – Users want to be confident that their data are not altered by a third party.
- Authentication – Users want to know that the data they receive were actually sent by the person listed in the "from" field and not some other individual.
- Non-repudiation – Users want to have irrefutable evidence that a data was sent, preventing the sender from later claiming that they never sent the message.
The basic function comparison between Public and Private-key Crypto systems:

<table>
<thead>
<tr>
<th>Goal</th>
<th>Private key Cryptosystem</th>
<th>Public key Cryptosystem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confidentiality</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Integrity</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Authentication</td>
<td>Sometimes</td>
<td>Yes</td>
</tr>
<tr>
<td>Non-Reputation</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 6 Basic comparison between private and Public-Key Cryptosystems

The project implements the tools encryption algorithms and hashing functions based upon the type of encryption implemented by the TPM by following the specifications mentioned by Trusted Computing Group (TCG) are RSA SHA-1 and HMAC for cryptographic systems. So I explained more about Asymmetric encryption and Hash functions.

3.2.2.1. Symmetric Encryption (Private-key Cryptography)

Secret-key encryption uses one key, the secret key, to both encrypt and decrypt messages. This is also called symmetric encryption. The term "private key" is often used inappropriately to refer to the secret key. A private key is one of the keys in the public/private key pair for asymmetric cryptography. In this scheme, a user has two keys, one that is made public and one that is held privately [18].

In secret-key cryptography schemes, a single key is used to encrypt data. A secret key may be held by one person or exchanged between the sender and the receiver of a message. For example, if you encrypt data for storage on a hard drive, you remember the key and usually don't give it to someone else. But if you want to send secure messages to a business partner using symmetric cryptography, you need to make sure your partner knows the key that will decrypt the messages.

If secret-key cryptography is used to send secret messages between two parties, both the sender and receiver must have a copy of the secret key. However, the key may be compromised during transit. If you know the party you are exchanging messages with, you can give them the key in advance. However, if you need to send an encrypted message to someone you have never met, you'll need to figure out a way to exchange keys in a secure
way. One method is to send it via another secure channel or even via overnight express, but this may be risky in some cases.

As mentioned, secret-key cryptography is often used to encrypt data on hard drives. The person encrypting the data holds the key privately and there is no problem with key distribution. Secret-key cryptography is also used for communication devices like bridges that encrypt all data that cross the link. A network administrator programs two devices with the same key, and then personally transports them to their physical locations.

![Figure 28 Symmetric Key File Encryption and Decryption](image)

### 3.2.2.2 Asymmetric Encryption (Public-key Cryptography)

Public-key cryptography provides a way for users to securely exchange information. It also enables a host with other useful security techniques, including authentication (remote connections without the need to exchange sensitive information), digital signing (to provide document integrity), and non-repudiation (someone cannot deny having sent a message) [19].

Assume Bob and Alice wants to exchange private encrypted messages over an unsecure system (like the Internet). They choose an encryption method that will make the messages unreadable to any person who happens to capture the transmissions. Bob encrypts the message using an encryption key. Alice must have this key to decrypt the message. Now, the basic problem: how does Bob get the key to Alice so she can decrypt the message? Bob could call Alice on the phone, but what if the phone line is tapped? Bob could send it via courier, but the key could be compromised. While this seems paranoid, consider that military communications are under constant scrutiny by attackers or foreign defence agencies. The same threat extends into the competitive corporate world and the financial world.

Traditionally, both the sender and receiver have already agreed on a key. Before leaving port, a submarine captain is handed a decoder book that will unscramble encrypted radio messages.
from home port. This is symmetric cryptography (both parties know the same secret key), and it is often referred to as secret-key cryptography. DES (Data Encryption Standard) is a common secret-key encryption. These single-key methods are described under “Symmetric Encryption”.

But it is not always the case that parties who need to exchange messages know and/or trusts each other or has previously exchanged keys. In 1976, Whitfield Diffie and Martin Hellman developed the concept of asymmetric public-key cryptography. In this scheme, a person uses a program to generate two keys. The keys are mathematically related through a special function that is very difficult to reverse. One key is kept private and the other is put in a public place, much like phone numbers are listed in a phone book. To send someone a private message, you look up their public key, encrypt the message, and send it to the owner of the key. The owner decrypts the message with their private key. Only the private key can decrypt messages encrypted with the public key. Therefore, the private key must be kept safe and secure.

![Asymmetric Key File Encryption and Decryption](image)

**Figure 29 Asymmetric Key File Encryption and Decryption**

### 3.2.2.2.1. Algorithms used in Asymmetric Encryption

This section explains few public key algorithms and their functionality. The algorithms covered in this section are [20]

- Key Agreement Algorithms – RSA, DH.
- Encryption Algorithms – RSA.
- Signature Algorithms – RSA, DSA.

This section also gives brief introduction to modular arithmetic.
3.2.2.2.1.1. Modular Arithmetic

Modular arithmetic is the commonly used arithmetic in public key cryptography. Modular arithmetic deals only with integers. Since it involves no floating-point operations, the mathematical calculations are more accurate and efficient than the real number arithmetic. Modular arithmetic over a number \( n \) involves arithmetic operations on integers between 0 and \( n - 1 \), where \( n \) is called the modulus. If the number happens to be out of this range in any of the operation the result, \( r \), is wrapped around in to the range 0 and \( n - 1 \) by repeated subtraction of the modulus \( n \) from the result \( r \). This is equivalent in taking the remainder of division operation \( r / n \).

*For e.g. for modulo 23 arithmetic [20]*

\( n=23, \text{ Let } a=15, b=20 \)

\((a + b) \mod n = (15+20) \mod 23 = 35 \mod 23 = 12\)

Since the result of \( a + b=35 \) which is out of the range \([0,22]\), the result is wrapped around in to the range \([0,22]\) by subtracting 35 with 23 till the result is in range \([0,22]\).

\( a \mod b \) is thus explained as remainder of division \( a/b \).

Subtraction and multiplication can also be explained similarly.

A negative number is added repeatedly with \( n \) till it can be represented in the range \([0, n-1]\).

The modular division \( a/b \mod p \) is defined as \( a \cdot b^{-1} \mod p \). \( b^{-1} \) is the multiplicative inverse of \( b \).

Multiplicative inverse of number \( b \) with respect to \( \mod p \) is defined as a number \( b^{-1} \) such that \( b \cdot b^{-1} \mod p = 1 \).

3.2.2.2.1.1.1. Congruent relation

Modular arithmetic is a congruent relation. Congruence is shown by the symbol ‘\( \equiv \)’. For a modulus \( n \) two numbers \( a \) and \( b \) are said to be congruent if \( a \mod n = b \mod n \). i.e. \( a \equiv b \pmod{n} \) if, \( a \mod n = b \mod n \)

For example consider the modulus 7 i.e. \( n = 7 \)

Then the numbers 2, 9, 16, 23 etc are congruent to each other since

\((2 \mod 7) = (9 \mod 7) = (16 \mod 7) = (23 \mod 7) \) etc
3.2.2.1.1.2. Properties of modular arithmetic

P1. a ≡ b mod n implies a–b=k*n, where k is an integer
P2. a mod n + b mod n ≡ a + b (mod n), also true for other operators ‘-‘, ‘*’and ‘/’
P3. a + b ≡ b + a (mod n), also true for other operators ‘-‘, ‘*’and ‘/’
P4. a ≡ a mod n
P5. a ≡ b mod n implies b ≡ a mod n
P6. Fermat’s little theorem, if M and p are co-prime then $M^{p-1} \equiv 1 (mod \ p)$
P7. if p and q are co-prime and also if $a \equiv b(mod \ p)$ and $a \equiv b(mod \ q)$ then $a \equiv b(mod \ pq)$ [20]

3.2.2.1.2. DH - Diffie-Hellman Key Agreement Algorithm

Diffie-Hellman is a key agreement algorithm that helps two devices to agree on a shared secret between them without the need to exchange any secret/private information. The DH standard is specified in RFC 2631. An overview of the algorithm is given below.

3.2.2.1.2.1. Key agreement Algorithm

For establishing shared secret between two devices A and B, both devices agree on public constants p and g. Where p is a prime number and g is the generator less than p.

D1. Let a and b be the private keys of the devices A and B respectively, Private keys are random number less than p.
D2. Let $g^a mod \ p$ and $g^b mod \ p$ be the public keys of devices A and B respectively
D3. A and B exchanged their public keys.
D4. The end A computes $(g^b mod \ p)^a \ mod \ p$ that is equal to $g^{ba} mod \ p$.
D5. The end B computes $(g^a mod \ p)^b \ mod \ p$ that is equal to $g^{ab} mod \ p$.
D6. Since $K = g^{ba} mod \ p = g^{ab} mod \ p$, shared secret = K. [20]

3.2.2.1.2.2. DH - Mathematical Explanation

From the properties of modular arithmetic P2

$a \ mod \ n * b \ mod \ n \equiv a * b \ (mod \ n)$

Which can be written as
\[(a_1 \mod n) \times (a_2 \mod n) \times \ldots \times (a_k \mod n) \equiv a_1 \times a_2 \times \ldots \times a_k \mod n,\]
if \(a_i = a\), where \(i = 1, 2, 3 \ldots k\)

\[(a \mod n)^k \equiv a^k \mod n, \text{ therefore } \text{ [HX1]}\]

\[(g^a \mod p)^b \mod p = g^{ab} \mod p \text{ and } \]
\[(g^b \mod p)^a \mod p = g^{ba} \mod p\]

For all integers \(g^{ab} = g^{ba}\),

Therefore shared secret \(K = g^{ab} \mod p = g^{ba} \mod p\)

Since it is practically impossible to find the private key \(a\) or \(b\) from the public key \(g^a \mod p\) or \(g^b \mod p\), it is not possible to obtain the shared secret \(K\) for a third party. [20]

### 3.2.2.1.2.3. One-Way function in DH

For device A, Let \(a\) be the private key and \(x = g^a \mod p\) is the public key, Here \(x = g^a \mod p\) is one-way function. The public key \(x\) is obtained easily in the forward operation, but finding ‘\(a\)’ given \(x\), \(g\) and \(p\) is the reverse operation and takes exponentially longer time and is practically impossible. This is known as discrete logarithm problem.

### 3.2.2.1.3. RSA

RSA is a public key algorithm that is used for Encryption, Signature and Key Agreement. RSA typically uses keys of size 1024 to 2048. The RSA standard is specified RFC 3447, RSA Cryptography Specifications Version 2.1.

Overviews of RSA algorithms are given below.

#### 3.2.2.1.3.1. RSA Encryption

*Parameter generation*

R1. Select two prime numbers \(p\) and \(q\).
R2. Find \(n = p \times q\), Where \(n\) is the modulus that is made public. The length of \(n\) is considered as the RSA key length.
R3. Choose a random number ‘e’ as a public key in the range 0<e<(p-1)(q-1) such that gcd(e,(p-1)(q-1))=1.

R4. Find private key d such that ed≡1(mod (p-1)(q-1)). [20]

**Encryption**

Consider the device A that needs to send a message to B securely.

R5. Let e be B’s public key. Since e is public, A has access to e.

R6. To encrypt the message M, represent the message as an integer in the range 0<M<n.

R7. Cipher text C = Me mod n, where n is the modulus. [20]

**Decryption**

R8. Let C be the cipher text received from A.

R9. Calculate Message M = Cd mod n, where d is B’s private key and n is the modulus. [20]

### 3.2.2.2.1.3.2. RSA Key Agreement

Since public key cryptography involves mathematical operation on large numbers, these algorithms are considerably slow compared to the symmetric key algorithm. They are so slow that it is infeasible to encrypt large amount of data. Public key encryption algorithm such as RSA can be used to encrypt small data such as ‘keys’ used in private key algorithm. RSA is thus used as key agreement algorithm.

**Key agreement algorithm**

For establishing shared secret between two device A and B

R10. Generate a random number, key, at device A.

R11. Encrypt key by RSA encryption algorithm using B’s public key and pass the cipher text to B

R12. At B decrypt the cipher text using B’s private key to obtain the key. [20]
3.2.2.1.3.3. RSA Signature

RSA Signature is similar to RSA encryption except that the private key is used for signing and public key is used for verification.

Parameter generation

The parameter generation process is same as that in RSA Encryption.

Signing

Consider the device A that needs to sign the data that it sends to B.
R13. Let d be A’s private key
R14. To sign a data M, represent the data as an integer in the range 0<M<n
R15. Signature C = Md mod n

Verification

R16. Let M be the message and C be the signature received from A
R17. Calculate M' = C^e mod n, where e is A’s public key. Since e is public, B has access to e
R18. If M' = M, the signature is verified, else failed.

3.2.2.1.3.4. One-Way function in RSA

Consider the key generation equation R4, ed = 1 (mod (p-1)(q-1)) and n = p * q Where e is the public key d is the private key, p and q are kept private but n is made public. Since e is public, anybody who has access to p and q could easily generate the private key d using the above equation R4. The security of RSA depends on the difficulty to factorize n to obtain the prime numbers p and q. n is easily obtained by multiplying p and q but the reverse operation of factorizing n to obtain prime numbers p and q is practically impossible if p and q are sufficiently large numbers.
**3.2.2.1.3.5. RSA – Mathematical Explanation**

From parameter generation equation R4
\[ ed \equiv 1 \pmod{(p-1)(q-1)}. \]
From the encryption equation R7
Cipher text \( C = M^e \mod n \)
From the decryption equation R9
Message \( M = C^d \mod n \)
Combining above two equations \( M = (M^e \mod n)^d \mod n \), using equation HX1
\( M = M^{ed} \mod n \)
Similarly by combining signature and verification equation R15 and R17 we get
\( M = M^{ed} \mod n \)
So to prove the correctness of RSA, it has to prove that
\( M = M^{ed} \mod n \), if
\[ ed \equiv 1 \pmod{(p-1)(q-1)} \]
From the above equation \( ed \equiv 1 \pmod{(p-1)(q-1)} \) and property P1 it follows that
\( ed - 1 = K (p-1)(q-1) \), which can also be written as
\( ed - 1 = k (p-1), \) and ---- [RX1]
\( ed - 1 = k'(q-1) ---- [RX2] \)
Where \( K, k \) and \( k' \) are positive integers
Since any integer is congruent to itself it can be written as
\( M^{ed} = M^{ed} \pmod{p} \). i.e.
\[ M^{ed} \equiv M^{ed - 1}\star M \pmod{p}, \]
Using equation RX1 the above equation can be written as
\( M^{ed} \equiv M^{k(p-1)} \star M \pmod{p}, ---- [RX3] \)
Since \( p \) is prime, any integer \( M \) can either be a co-prime with \( p \) or a multiple of \( p \).

**Case 1:** If \( M \) and \( p \) are coprime, then from Fermat’s little theorem
\( M^{p-1} \equiv 1 \pmod{p}, \) or
\( M^{k(p-1)} \equiv 1^k \pmod{p} \), i.e.
\( M^{k(p-1)} \equiv 1 \pmod{p} ---- [RX4] \)
From equations RX3 and RX4
\( M^{ed} \equiv M \pmod{p} \)
Case 2: If \( M \) is a multiple of \( p \), then \( M^{ed} \) will also be a multiple of \( p \), i.e. \( M \mod p = 0 \), also \( \text{Med mod } p = 0 \), thus from congruence relation,
\[
M^{ed} \equiv M \pmod{p}
\]
Similarly using RX2 it can be proved that \( M^{ed} \equiv M \pmod{p} \) for above two cases.

Since \( p \) and \( q \) are prime numbers they are coprime to each other. Therefore by using property P7 the above two equations can be combined as
\[
M^{ed} \equiv M \pmod{p*q}, \text{ by property P5}
\]
\[
M \equiv M^{ed} \pmod{p*q}
\]
Since \( M \) is chosen in the range 0 and \((p*q-1)\)
\[
M = M^{ed} \pmod{n} \quad [20]
\]

### 3.2.2.2.1.4. DSA – Digital Signature Algorithm

DSA is a public key algorithm that is used for Digital Signature. The DSA standard is specified FIPS 186-2, Digital Signature Standard An overview of the algorithm is given below.

**Parameter generation**

S1. Choose a 160-bit prime \( q \).
S2. For an integer \( z \), choose an L-bit prime \( p \), such that \( p = qz + 1, 512 \leq L \leq 1024 \), and \( L \) is divisible by 64.
S3. Choose \( h \), where \( 1 < h < p-1 \) such that \( g = h^Z \mod p > 1 \).
S4. Choose a random number \( x \), where \( 0 < x < q \).
S5. Calculate \( y = g^x \mod p \).
S6. Public key is \((p, q, g, y)\). Private Key is \( x \). [20]

**Signing**

Consider the device \( A \) that sign the data \( M \) that it sends to \( B \).
S7. Let \( x \) be \( A \)’s private key and \((p, q, g, y)\) be \( A \)’s public key.
S8. Generate a random per-message value \( k \), where \( 0 < k < q \).
S9. Calculate \( r = (g^k \mod p) \mod q \).
S10. Calculate \( s = (k^{-1}(M + x*r)) \mod q \), where \( M \) is the hash SHA1 of the message
S11. The signature is (r, s). [20]

**Verification**

S12. Let M be the message and (r, s) be the signature received from A.
S13. Let (p, q, g, y) be A's public key. Since (p, q, g, y) is public, B has access to it.
S14. Calculate \( w = s^{-1} \mod q \).
S15. Calculate \( u_1 = (M \cdot w) \mod q \), where M is the hash SHA1 of the message.
S16. Calculate \( u_2 = (r \cdot w) \mod q \).
S17. Calculate \( v = ((g^{u_1} \cdot y^{u_2}) \mod p) \mod q \).
S18. The signature is valid if \( v = r \), invalid otherwise. [20]

### 3.2.2.2.1.4.1. DSA - Mathematical Explanation

From S18, Signature is valid if \( v = r \), to prove the correctness of the algorithm it has to prove that \( v = r \) if signature is valid.

Form S17, \( v = ((g^{u_1} \cdot y^{u_2}) \mod p) \mod q \).

But from S5, \( y = g^x \mod p \), i.e \( y \equiv g^x \mod p \), i.e. using equation HX1, \( y^{u_2} \equiv g^{x \cdot u_2} \mod p \)
Therefore
\[
v = ((g^{u_1} \cdot g^{x \cdot u_2}) \mod p) \mod q. \quad \text{[DX1]}
\]

But from S16, \( g^{x \cdot u_2} \equiv g^{x \cdot (r \cdot w \mod q)} = g^{(x \mod q) \cdot (r \cdot w \mod q)} \)
From P2 \( (x \mod q) \cdot (r \cdot w \mod q) \equiv (x^* r^* w) \mod q \)
Using P1 \( (x \mod q) \cdot (r \cdot w \mod q) = (x^* r^* w) \mod q + k \cdot q \)
Where \( k \) is an integer, Therefore
\[
g^{x \cdot u_2} = g^{(x^* r^* w) \mod q + k \cdot q}, \text{ i.e.}
\]
\[
g^{x \cdot u_2} = (g^{(x^* r^* w) \mod q}) \cdot (g^{k \cdot q}) \quad \text{[DX2]}
\]
From S3, \( g = h^z \mod p \),
Using HX1 and S2, \( g^q \equiv h^{qz} \equiv h^{p-1} \mod p \)
By P6, Fermat's little theorem \( h^{p-1} \equiv 1 \mod p \), i.e. \( g^q \equiv 1 \mod p \)
Substituting the value of \( g^q \equiv 1 \mod p \) in DX2
\[
g^{x \cdot u_2} = (g^{(x^* r^* w) \mod q}) \quad \text{[DX3]}
\]
But from S15, \( g^{u_1} = g^{(M \cdot w \mod q)} \quad \text{[DX4]}
Substituting DX3 and DX4 in DX1
\[ v = ((g^{wM \mod q + xr \mod q} \mod p) \mod q) \mod q. \text{ i.e.} \]
\[ v = ((g^w (M+xr) \mod q) \mod p) \mod q. \text{ ---- [DX5]} \]

From S10, in signature algorithm
\[ s = (k^{-1}(M+x*r)) \mod q \text{ i.e.} \]
\[ k \equiv s^{-1} (M+x*w) \mod q \text{ ---- [DX6]} \]

But from S14, \( w = s^{-1} \mod q \text{ i.e.} \)
\[ s^{-1} \equiv w \mod q \]

Therefore equation DX6 can be written as
\[ k \equiv w(M+x*w) \mod q \]

Since \( k < q \), \( k = w(M+x*w) \mod q \text{ ---- [DX7]} \)

Combining DX5 and DX7 and using S9
\[ V = ((g^k \mod p) \mod q = r. \text{ i.e.} \ v = r \text{ [20]} \]

### 3.2.2.2.1.4.2. One-Way function in DSA

Consider the equation S5, \( y = g^x \mod p \), where \( x \) is the private key but \( y \), \( g \) and \( p \) are public.
Calculating \( y \) from \( g \), \( x \) and \( p \) is a forward operation but obtaining \( x \) from the given \( y \), \( g \) and \( p \) is the reverse operation and hence finding \( x \) is impossible for large numbers. This is known as discrete logarithm problem.

### 3.2.2.3. Hash Functions

A hash function is a form of encryption that takes some plaintext input and transforms it into a fixed-length encrypted output called the message digest. The digest is a fixed-size set of bits that serves as a unique "digital fingerprint" for the original message. If the original message is altered and hashed again, it will produce a different signature. Thus, hash functions can be used to detect altered and forged documents. They provide message integrity, assuring recipients that the contents of a message have not been altered or corrupted.

Hash functions are one-way, meaning that it is easy to compute the message digest but very difficult to revert the message digest back to the original plaintext (e.g., imagine trying to put a smashed pumpkin back to exactly the way it was) [21].
3.2.2.3.1. Hash Function features

Hash Functions features are listed here.

- A hash function should be impossible for two different messages to ever produce the same message digest. Changing a single digit in one message will produce an entirely different message digest.

- It should be impossible to produce a message that has some desired or predefined output (target message digest).

- It should be impossible to reverse the results of a hash function. This is possible because a message digest could have been produced by an almost infinite number of messages.

- The hash algorithm itself does not need to be kept secret. It is made available to the public. Its security comes from its ability to produce one-way hashes.

- The resulting message digest is a fixed size. A hash of a short message will produce the same size digest as a hash of a full set of encyclopaedias.

Hash functions may be used with or without a key. If a key is used, both symmetric (single secret key) and asymmetric keys (public/private key pairs) may be used. The two primary algorithms are listed next and the RFCs listed later provide more information on the protocols.

**MD-5**: A hash function designed by Ron Rivest, one of the inventors of the RSA public-key encryption scheme. The MD-5 algorithm produces a 128-bit output. Note that MD-5 is now known to have some weaknesses and should be avoided if possible. SHA-1 is generally recommended. This is discussed later.

**SHA-1**: (Secure Hash Algorithm-1) SHA-1 is an MD-5-like algorithm that was designed to be used with the Digital Signature Standard (DSS). The United States agencies NIST (National Institute of Standards and Technology) and NSA (National Security Agency) are
responsible for SHA-1. The SHA-1 algorithm produces a 160-bit MAC. This longer output is considered to be more secure than MD-5.

*Keyed MD5* is a technique for using MD-5. Basically, a sender appends a randomly generated key to the end of a message, and then hashes the message and key combination to create a message digest. Next, the key is removed from the message and encrypted with the sender's private key. The message, message digest, and encrypted key are sent to the recipient, who opens the key with the sender's public key (thus validating that the message is actually from the sender). The recipient then appends the key to the message and runs the same hash as the sender. The message digest should match the message digest sent with the message.

The result of a hash function that combines a message with a key is called a message authentication code, or MAC. A MAC is a "fingerprint" or "message digest" of the input in combination with a key available to parties in the message exchange.

Hash functions are used in authentication routines such as CHAP (Challenge Handshake Authentication Protocol). Both the client and server share a secret-the password used by the client, which has been previously exchanged but is never sent over the wire. When the client establishes a link to the server, the server sends a unique "challenge" value (sometimes called a nonce) to the client. The client combines his or her password with the challenge and then runs them through the hash function. The result is sent back to the server, which runs the same process and compares its results with those received from the client. If they compare, the client is considered authentic. Note that the actual password is never sent, only a hash of the challenge and password combination.

HMAC (Hashed Message Authentication Code) is a core protocol that is considered essential for security on the Internet along with IPSec, according to RFC (Request for Comments) 2316 (Report of the IAB, April 1998). It is not a hash function, but a mechanism for message authentication that uses either MD5 or SHA-1 hash functions in combination with a shared secret key (as opposed to a public/private key pair). Basically, a message is combined with a key and run through the hash function. The result is then combined with the key and run through the hash function again. This 128-bit result is truncated to 96 bits and becomes the MAC.
According to RFC (Request for Comments) 2104 (HMAC: Keyed-Hashing for Message Authentication, February 1997), HMAC should be used in preference to older techniques, notably keyed hash functions. Keyed hashes based on MD-5 are especially to be avoided, given the hints of weakness in MD-5. HMAC is the preferred shared-secret authentication technique, and it should be used with SHA-1. It can be used to authenticate any arbitrary message and is suitable for logins.

3.2.2.3.2. Hash Function Types

At the highest level, hash functions may be split into two classes: unkeyed hash functions, whose specification dictates a single input parameter (a message); and keyed hash functions, whose specification dictates two distinct inputs, a message and a secret key. To facilitate discussion, a hash function is informally defined as follows [22].

A hash function (in the unrestricted sense) is a function h which has, as a minimum, the following two properties:
1. Compression — h maps an input x of arbitrary finite bitlength, to an output h(x) of fixed bitlength n.
2. Ease of computation—given h and an input x, h(x) is easy to compute.

Based on functional classification hash functions are classified as

3.2.2.3.2.1. Modification detection codes (MDCs)

Also known as manipulation detection codes and less commonly as message integrity codes (MICs), the purpose of an MDC is (informally) to provide a representative image or hash of a message, satisfying additional properties as refined below. The end goal is to facilitate, in conjunction with additional mechanisms, data integrity assurances as required by specific applications. MDCs are a subclass of unkeyed hash functions, and themselves may be further classified.
(i) One-way hash functions (OWHFs): for these, finding an input which hashes to a pre-specified hash-value is difficult;
(ii) Collision resistant hash functions (CRHFs): for these, finding any two inputs having the same hash-value is difficult.
3.2.2.3.2.2. Message authentication codes (MACs)

A cryptographic message authentication code (MAC) is a short piece of information used to authenticate a message. A MAC algorithm accepts as input a secret key and an arbitrary-length message to be authenticated, and outputs a MAC. The MAC value protects both a message's data integrity as well as its authenticity, by allowing verifiers (who also possess the secret key) to detect any changes to the message content, and so should be called Message Authentication and Integrity Code (MAIC).

The purpose of a MAC is (informally) to facilitate, without the use of any additional mechanisms, assurances regarding both the source of a message and its integrity. MACs have two functionally distinct parameters, a message input and a secret key; they are a subclass of keyed hash functions. [21]

Figure illustrates this simplified classification.

![Figure 30 Simplified classification of cryptographic hash functions and applications.](image)
It is generally assumed that the algorithmic specification of a hash function is public knowledge. Thus in the case of MDCs, given a message as input, anyone may compute the hash-result; and in the case of MACs, given a message as input, anyone with knowledge of the key may compute the hash-result.

### 3.2.2.3.3. Basic properties of MDCs and MACs

**A Modification detection codes** (MDCs) is unkeyed hash function $h$ with inputs $x, x'$ and outputs $y, y'$, with the following properties.

**Preimage resistance**: For essentially all pre-specified outputs, it is computationally infeasible to find any input which hashes to that output, i.e., to find any preimage $x'$ such that $h(x') = y$ when given any $y$ for which a corresponding input is not known.

**2nd-preimage resistance**: It is computationally infeasible to find any second input which has the same output as any specified input, i.e., given $x$, to find a 2nd-preimage $x' \neq x$ such that $h(x) = h(x')$.

**Collision resistance**: It is computationally infeasible to find any two distinct inputs $x, x'$ which hash to the same output, i.e., such that $h(x) = h(x')$. (Note that here there is free choice of both inputs.)

Alternative names for preimage resistant is *one-way*, for 2nd-preimage resistance is *weak collision resistance*, and for collision resistance is *strong collision resistance*.

**A one-way hash function (OWHF)** is a hash function $h$ as per hash functions (i.e., offering ease of computation and compression) with the following additional properties, as defined above: preimage resistance, 2nd-preimage resistance.

**A collision resistant hash function (CRHF)** is a hash function $h$ as per hash functions (i.e., offering ease of computation and compression) with the following additional properties, as defined above: 2nd-preimage resistance, collision resistance.

An alternative name for OWHF is *weak one-way hash*, for CRHF is *strong one-way hash function*. 
A message authentication code (MAC) algorithm is a family of functions $h_k$ parameterized by a secret key $k$, with the following properties:

**Ease of computation:** for a known function $h_k$, given a value $k$ and an input $x$, $h_k(x)$ is easy to compute. This result is called the MAC-value or MAC.

**Compression:** $h_k$ maps an input $x$ of arbitrary finite bitlength to an output $h_k(x)$ of fixed bitlength $n$.

Furthermore, given a description of the function family $h$, for every fixed allowable value of $k$ (unknown to an adversary), the following property holds.

**Computation-resistance:** Given zero or more text-MAC pairs $(x_i; h_k(x_i))$, it is computationally infeasible to compute any text-MAC pair $(x; h_k(x))$ for any new input $x \neq x_i$ (including possibly for $h_k(x) = h_k(x_i)$ for some $i$).

If computation-resistance does not hold, a MAC algorithms subject to MAC forgery. While computation-resistance implies the property of key non-recovery (it must be computationally infeasible to recover $k$, given one or more text-MAC pairs $(x_i; h_k(x_i))$ for that $k$), key non-recovery does not imply computation-resistance (a key need not always actually be recovered to forge new MACs).

### 3.2.2.3.4. General model for iterated hash functions

Most unkeyed hash functions $h$ are designed as iterative processes which hash arbitrary length inputs by processing successive fixed-size blocks of the input, as illustrated in Figure.
A hash input $x$ of arbitrary finite length is divided into fixed-length $r$-bit blocks $x_i$. This pre-processing typically involves appending extra bits (padding) as necessary to attain an overall bitlength which is a multiple of the block length $r$, and often includes a block or partial block indicating the bitlength of the unpadded input. Each block $x_i$ then serves as input to an internal fixed-size hash function $f$, the compression function of $h$, which computes a new intermediate result of bitlength $n$ for some fixed $n$, as a function of the previous $n$-bit intermediate result and the next input block $x_i$. Letting $H_i$ denote the partial result after stage $i$, the general process for an iterated hash function with input $x = x_1, x_2, \ldots x_t$ can be modelled as follows:

$H_0 = IV; H_i = f(H_{i-1}; x_i), 1 \leq i \leq t; h(x) = g(H_t)$.

$H_{i-1}$ serves as the n-bit chaining variable between stage $i-1$ and stage $i$, and $H_0$ is a pre-defined starting value or initializing value (IV). An optional output transformation $g$ is used in a final step to map the n-bit chaining variable to an m-bit result $g(H_t)$; $g$ is often the identity mapping $g(H_t) = H_t$.

Particular hash functions are distinguished by the nature of the pre-processing, compression function, and output transformation.

Figure 31 General model for an iterated hash function. [22]
3.2.2.3.5. Hash Function Algorithms

The broadly used unkeyed hash functions are MD5 and SHA-1 and these structures are discussed below and TPM uses SHA-1 hash function.

3.2.2.3.5.1. MD5

In cryptography, MD5 (Message-Digest algorithm 5) is a widely used, partially insecure cryptographic hash function with a 128-bit hash value. As an Internet standard (RFC 1321), MD5 has been employed in a wide variety of security applications, and is also commonly used to check the integrity of files. An MD5 hash is typically expressed as a 32 digit hexadecimal number.

MD5 was designed by Ron Rivest in 1991 to replace an earlier hash function, MD4. In 1996, a flaw was found with the design of MD5; while it was not a clearly fatal weakness, cryptographers began recommending the use of other algorithms, such as SHA-1 (which has since been found vulnerable itself). In 2004, more serious flaws were discovered making further use of the algorithm for security purposes questionable. In 2007 a group of researchers including Arjen Lenstra described how to create a pair of files that share the same MD5 checksum.

3.2.2.3.5.1.1. Algorithms

The MD5 algorithm was derived from MD4 algorithm by making small changes to MD4. So before defining MD5 algorithm its better to know MD4 algorithm which was explained here. [22]

3.2.2.3.5.1.1.1. MD4 hash function Algorithm

INPUT: bitstring x of arbitrary bitlength b >= 0.
OUTPUT: 128-bit hash-code of x.

1. *Definition of constants*. Define four 32-bit initial chaining values (IVs): h1 = 0x67452301, h2 = 0xefcdab89, h3 = 0x98badcfe, h4 = 0x10325476.
Define additive 32-bit constants:
\[ y[j] = 0, \ 0 < j < 15; \]
\[ y[j] = 0x5a827999, \ 16 \leq j \leq 31; \text{(constant = square-root of 2)} \]
\[ y[j] = 0x6ed9eba1, \ 32 \leq j \leq 47; \text{(constant = square-root of 3)} \]

Define order for accessing source words (each list contains 0 through 15):
\[ z[0..15] = [0; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12; 13; 14; 15], \]
\[ z[16..31] = [0; 4; 8; 12; 1; 5; 9; 13; 2; 6; 10; 14; 3; 7; 11; 15], \]
\[ z[32..47] = [0; 8; 4; 12; 2; 10; 6; 14; 1; 9; 5; 13; 3; 11; 7; 15]. \]

Finally define the number of bit positions for left shifts (rotates):
\[ s[0..15] = [3; 7; 11; 19; 3; 7; 11; 19; 3; 7; 11; 19; 3; 7; 11; 19], \]
\[ s[16..31] = [3; 5; 9; 13; 3; 5; 9; 13; 3; 5; 9; 13; 3; 5; 9; 13], \]
\[ s[32..47] = [3; 9; 11; 15; 3; 9; 11; 15; 3; 9; 11; 15; 3; 9; 11; 15]. \]

2. Pre-processing. Pad x such that its bitlength is a multiple of 512, as follows. Append a single 1-bit, then append \( r^{-1} \) (\( \geq 0 \)) 0-bits for the smallest \( r \) resulting in a bitlength 64 less than a multiple of 512. Finally append the 64-bit representation of \( b \mod 264 \), as two 32-bit words with least significant word first. (Regarding converting between streams of bytes and 32-bitwords, the convention is little-endian; Let \( m \) be the number of 512-bit blocks in the resulting string \( b + r + 64 = 512m = 32 \times 16m \). The formatted input consists of \( 16m \) 32-bit words: \( x_0x_1 \ldots x_{16m-1} \).

Initialize: \((H_1, H_2, H_3, H_4) \leftarrow (h_1, h_2, h_3, h_4)\).

3. Processing. For each \( i \) from 0 to \( m - 1 \), copy the \( i \)th block of 16 32-bit words into temporary storage: \( X[j] \leftarrow x_{16i+j}; \ 0 \leq j < 15 \), then process these as below in three 16-step rounds before updating the chaining variables: (initialize working variables) \((A,B,C,D) \leftarrow (H_1, H_2, H_3, H_4)\).

(Round 1) For \( j \) from 0 to 15 do the following:
\[ t(A + f(B; C; D) + X[z[j]] + y[j]), (A; B; C; D) (D; t - s[j]; B; C). \]

(Round 2) For \( j \) from 16 to 31 do the following:
\[ t(A + g(B; C; D) + X[z[j]] + y[j]), (A; B; C; D) (D; t - s[j]; B; C). \]

(Round 3) For \( j \) from 32 to 47 do the following:
\[ t(A + h(B; C; D) + X[z[j]] + y[j]), (A; B; C; D) (D; t - s[j]; B; C). \]
(Update chaining values) \((H_1, H_2, H_3, H_4) \leftarrow (H_1+A; H_2+B; H_3+C; H_4+D)\).

4. **Completion.** The final hash-value is the concatenation: \(H_1||H_2||H_3||H_4\) (With first and last bytes the low- and high-order bytes of \(H_1, H_4\), respectively).

**Vulnerability:** MD4 Collisions have been found for MD4 in \(2^{20}\) compression function computations. For this reason, MD4 is no longer recommended for use as a collision-resistant hash function. While its utility as a one-way function has not been studied in light of this result, it is prudent to expect a preimage attack on MD4 requiring fewer than \(2^{128}\) operations will be found.

### 3.2.2.3.5.1.1.2. MD5 hash function Algorithm

MD5 was designed as a strengthened version of MD4, prior to actual MD4 collisions being found. It has enjoyed widespread use in practice. It has also now been found to have weaknesses [22]

The changes made to obtain MD5 from MD4 are as follows:

1. Addition of a fourth round of 16 steps, and a Round 4 function
2. Replacement of the Round 2 function by a new function
3. Modification of the access order for message words in Rounds 2 and 3
4. Modification of the shift amounts (such that shifts differ in distinct rounds)
5. Use of unique additive constants in each of the 4_16 steps, based on the integer part of \(2^{232} \cdot \sin(j)\) for step \(j\) (requiring overall, 256 bytes of storage)
6. Addition of output from the previous step into each of the 64 steps.

**Algorithm**

INPUT: bitstring \(x\) of arbitrary bitlength \(b \geq 0\).

OUTPUT: 128-bit hash-code of \(x\).

MD5 is obtained from MD4 by making the following changes.

1. **Notation.** Replace the Round 2 function by: \(g(u, v, w) \text{ def}= uw \lor vw\).
Define a Round 4 function: $k(u, v, w) \text{ def } = v \oplus (u \oplus w')$.

2. Definition of constants. Redefine unique additive constants:

$y[j] =$ first 32 bits of binary value $\text{abs} (\sin(j+1))$, $0 \leq j \leq 63$, where $j$ is in radians and “abs” denotes absolute value. Redefine access order for words in Rounds 2 and 3, and define for Round 4:

$z[16..31] = [1; 6; 11; 0; 5; 10; 15; 4; 9; 14; 3; 8; 13; 2; 7; 12]$,  
$z[32..47] = [5; 8; 11; 14; 1; 4; 7; 10; 13; 0; 3; 6; 9; 12; 15; 2]$,  
$z[48..63] = [0; 7; 14; 5; 12; 3; 10; 1; 8; 15; 6; 13; 4; 11; 2; 9]$.

Redefine number of bit positions for left shifts (rotates):

$s[0..15] = [7; 12; 17; 22; 7; 12; 17; 22; 7; 12; 17; 22; 7; 12; 17; 22]$,  
$s[16..31] = [5; 9; 14; 20; 5; 9; 14; 20; 5; 9; 14; 20; 5; 9; 14; 20]$,  
$s[32..47] = [4; 11; 16; 23; 4; 11; 16; 23; 4; 11; 16; 23; 4; 11; 16; 23]$,  
$s[48..63] = [6; 10; 15; 21; 6; 10; 15; 21; 6; 10; 15; 21; 6; 10; 15; 21]$.

3. Pre-processing. As in MD4.

4. Processing. In each of Rounds 1, 2, and 3, replace “$B \leftarrow (t - s[j])$” by “$B \leftarrow B + (t \leftarrow s[j])$”. Also, immediately following Round 3 add: (Round 4) For $j$ from 48 to 63 do the following:

$t \leftarrow (A+k(B,C,D)+X[z[j]]+y[j]). (A,B,C,D) \leftarrow (D,B+(t \leftarrow s[j]),B, C)$.

5. Completion. As in MD4.

Vulnerability

*MD5 compression function collisions* have been found for the MD5 compression function.

More specifically, these are called collisions for random IV.

**Example Hashes**

The 128-bit (16-byte) MD5 hashes (also termed message digests) are typically represented as a sequence of 32 hexadecimal digits. The following demonstrates a 43-byte ASCII input and the corresponding MD5 hash:
MD5 ("The quick brown fox jumps over the lazy dog")
= 9e107d9d372bb6826bd81d3542a419d6

Even a small change in the message will (with overwhelming probability) result in a completely different hash, due to the avalanche effect. For example, changing d to e:

MD5 ("The quick brown fox jumps over the lazy eog")
= ffd93f16876049265fbaef4da268dd0e

The hash of the zero-length string is:

MD5 ("")
= d41d8cd98f00b204e9800998ecf8427e

3.2.2.3.5.2. SHA-1 Algorithm

The Secure Hash Algorithm (SHA-1), based on MD4, was proposed by the U.S. National Institute for Standards and Technology (NIST) for certain U.S. federal government applications. [22]

The main differences of SHA-1 from MD4 are as follows:

1. The hash-value is 160 bits and five (vs. four) 32-bit chaining variables are used.
2. The compression function has four rounds instead of three, using the MD4 step functions f, g, and h as follows: f in the first, g in the third, and h in both the second and fourth rounds. Each round has 20 steps instead of 16.
3. Within the compression function, each 16-word message block is expanded to an 80-word block, by a process whereby each of the last 64 of the 80 words is the XOR of 4 words from earlier positions in the expanded block. These 80 words are then input one-word-per-step to the 80 steps.
4. The core step is modified as follows: the only rotate used is a constant 5-bit rotate; the fifth working variable is added into each step result; message words from the expanded message block are accessed sequentially; and C is updated as B rotated left 30 bits, rather than simply B.
5. SHA-1 uses four non-zero additive constants, whereas MD4 used three constants only two of which were non-zero.
The byte ordering used for converting between streams of bytes and 32-bitwords in the official SHA-1 specification is big-endian (see Note 9.48); this differs from MD4 which is little-endian.

Algorithm

INPUT: bitstring $x$ of bitlength $b_0$.
OUTPUT: 160-bit hash-code of $x$.
SHA-1 is defined (with reference to MD4) by making the following changes.

1. Notation. As in MD4.

2. Definition of constants. Define a fifth IV to match those in MD4: $h_5 = 0xc3d2e1f0$.
Define per-round integer additive constants: $y_1 = 0x5a827999$, $y_2 = 0x6ed9eba1$,
$Y_3 = 0x8f1bbcdc$, $y_4 = 0xca62c1d6$. (No order for accessing source words or specification of bit positions for left shifts is required.)

3. Overall pre-processing. Pad as in MD4, except the final two 32-bit words specifying the bitlength $b$ is appended with most significant word preceding least significant. As in MD4, the formatted input is $16m$ 32-bit words: $x_0x_1…x_{16m-1}$.
Initialize chaining variables: $(H_1, H_2, H_3, H_4) \leftarrow (h_1, h_2, h_3, h_4)$.

4. Processing. For each $i$ from $0$ to $m - 1$, copy the $i$th block of sixteen 32-bit words into temporary storage: $X[j] \leftarrow X_{16i + j}$, $0 <= j <= 15$, and process these as below in four 20-step rounds before updating the chaining variables: (expand 16-word block into 80-word block; let $X_j$ denote $X[j]$) for $j$ from $16$ to $79$, $X_j \leftarrow ((X_j-3 \leftarrow || X_j-8 \leftarrow || X_j-14 \leftarrow || X_j-16) \leftarrow 1)$

(Initialize working variables) $(A, B, C, D, E) \leftarrow (H_1, H_2, H_3, H_4, H_5)$.

(Round 1) For $j$ from $0$ to $19$ do the following:
$t \leftarrow ((A \leftarrow 5) + f(B, C, D) + E + X_j + y_1), (A, B, C, D, E) \leftarrow (t, A, B \leftarrow 30, C; D)$.

(Round 2) For $j$ from $20$ to $39$ do the following:
$t \leftarrow((A \leftarrow 5) + h(B, C, D) + E + X_j + y_2), (A, B, C, D, E) \leftarrow (t, A, B \leftarrow 30, C, D)$.

(Round 3) For $j$ from $40$ to $59$ do the following:
t \leftarrow ((A \leftarrow 5) + g(B, C, D) + E + X_j + y_3), (A, B, C, D, E) \leftarrow (t, A, B \leftarrow 30, C, D).

(Round 4) For j from 60 to 79 do the following:
\[ t \leftarrow ((A \leftarrow 5) + h(B, C, D) + E + X_j + y_4), (A, B, C, D, E) \leftarrow (t, A, B \leftarrow 30, C, D). \]

(Update chaining values)
\[ (H_1, H_2, H_3, H_4, H_5) \leftarrow (H_1 + A, H_2 + B, H_3 + C, H_4 + D, H_5 + E). \]

5. Completion. The hash-value is: \( H_1 || H_2 || H_3 || H_4 || H_5 \) (with first and last bytes the high- and low-order bytes of \( H_1, H_5 \), respectively).

Vulnerability

Compared to 128-bit hash functions, the 160-bit hash-value of SHA-1 provides increased security against brute-force attacks. SHA-1 is considered stronger than MD5. In SHA-1, a significant effect of the expansion of 16-word message blocks to 80 words in the compression function is that any two distinct 16-word blocks yield 80-word values which differ in a larger number of bit positions, significantly expanding the number of bit differences among message words input to the compression function. The redundancy added by this pre-processing evidently adds strength.

Example hashes

The following is an example of SHA-1 digests. ASCII encoding is assumed for all messages.

\[ \text{SHA1 ("The quick brown fox jumps over the lazy dog"}) \]
\[ = 2fd4e1c6 7a2d28fc ed849ee1 bb76c739 1b93eb12 \]

Even a small change in the message will, with overwhelming probability, result in a completely different hash due to the avalanche effect. For example, changing dog to cog:

\[ \text{SHA1 ("The quick brown fox jumps over the lazy cog"}) \]
\[ = \text{de9f2c7f d25e1b3a fad3e85a 0bd17d9b 100db4b3} \]
3.3. **Symmetric vs. asymmetric algorithms**

Unlike symmetric algorithms, asymmetric key algorithms use a different key for encryption than for decryption. I.e., a user knowing the encryption key of an asymmetric algorithm can encrypt messages, but cannot derive the decryption key and cannot decrypt messages encrypted with that key. A short comparison of these two types of algorithms is given below [23]:

**Speed**

Symmetric-key algorithms are generally much less computationally intensive than asymmetric key algorithms. In practice, asymmetric key algorithms are typically hundreds to thousands times slower than symmetric key algorithms.

**Key management**

One disadvantage of symmetric-key algorithms is the requirement of a shared secret key, with one copy at each end. In order to ensure secure communications between everyone in a population of n people a total of n (n – 1)/2 keys are needed, which is the total number of possible communication channels. To limit the impact of a potential discovery by a cryptographic adversary, they should be changed regularly and kept secure during distribution and in service. The process of selecting, distributing and storing keys is known as key management; it is difficult to achieve reliably and securely.

**Hybrid cryptosystem**

In modern cryptosystems designs, both asymmetric (public key) and symmetric algorithms are used to take advantage of the virtues of both. Asymmetric algorithms are used to distribute symmetric-keys at the start of a session. Once a symmetric key is known to all parties of the session, faster symmetric-key algorithms using that key can be used to encrypt the remainder of the session. This simplifies the key distribution problem, because asymmetric keys only have to be distributed authentically, whereas symmetric keys need to be distributed in an authentic and confidential manner.
In symmetric or asymmetric encryption the main problem is generation and storing of keys which will be generated using random number generators and to store the keys. To avoid above problems we will do the encryption which must be trusted and safer so that we can use the ideas of *Trusted Computing* which was implemented by TPM (Trusted Platform Module) a microcontroller chip where the key generation and storing takes place to make browser secure.

### 3.4. Structure of the Project

By using the above concepts the project which provides security for web browser cookies and passwords was developed and structure of the project was shown below and the explanation follows.

![Figure 32 Project Structure](image)

#### 3.4.1. Project Initialisation

The Project Initialisation checks the requirements need to run the project and checks whether the user was taken TPM ownership or not and then sets the environment to run the project and the requirements necessary to run the project are:
3.4.1.1. Requirements to run the project

The total project was developed under Linux Operating system, so it will work only under Linux operating system and the following products must have to be installed in your system before running my software.

1. **Java**: Download and install Java JRE >= 1.4 and JCE (Java Cryptography Extension) from http://java.sun.com/j2se/1.4.2/download.html

2. **TPM (Trusted Platform Module)**: Check whether there is a microcontroller chip is already located in your system motherboard or not, to know for this read the manual provided by your system manufacture. If not exists TPM, install TPM emulator, installation was explained after TPM section.

3. **Tpm4Java**: The total project developed by me was in Java platform, so I took the help of a library which was also developed in Java to access TPM (Trusted Platform Module) features. So download tpm4java from http://tpm4java.datenzone.de/trac/wiki and make it compile by using the `ant dist` command, if there is no ant in your system download and install from http://ant.apache.org/, this will generated a subdirectory dist with all important files in it. You will need at least the tpm4java.jar file. Get the tpm4java.jar file and add it to your class path.

4. **Browsers**: I developed the total project to work for only two browsers and it can be updated soon for remaining browsers also, and currently it will work only for Mozilla Firefox and Opera. So please check if exist any these browsers in your system and if not make install by downloading from corresponding websites.

5. **Operating System**: The last and but not least requirement to run the project is the Operating System it supports, I developed this project under Linux, so it will run only under Linux and for Windows it will be updated soon which works for all browsers it supports.
3.4.2. Authentication Process

Authentication process authenticates the user to run the project by checking the username and password of the user if he already created account and if not it gives error message to create account by mentioning password rules i.e. password must be equal to or more than 12 characters including small and capital letters, symbols and numbers.

3.4.3. Encryption Process

Encryption process triggers Sending commands to TPM for Encryption section if the user is new and cookies are existed for corresponding browsers in browser installation folder, if not the project runs until user opens and closes the browser. After closing the browser, Sending commands to TPM for Encryption was triggered to do encryption.

3.4.4. Sending commands to TPM for Encryption

This section sends commands to TPM to encrypt cookies and passwords for corresponding browsers by checking if there is any encrypted cookies were existed for particular browser in project installation folder and if it not then encrypts the cookies stored in browser installation folder and moves them to project installation folder.

3.4.5. Browser Checking Process

This section checks the browser status i.e. if the process detects the browser then Decryption Process section was initialised to do decryption by checking if there is any encrypted cookies were existed or not in project installation folder. And if the browser was not detected then it checks if there is any encrypted cookies were present in project installation folder, if cookies were not found, the Encryption Process will be initialised to do cookies encryption for particular browser. This process runs like an infinite loop until the user shuts down his computer.

3.4.6. Decryption Process

After getting the signal from Browser Checking Process, i.e. the particular browser was opened for browsing then this section was initialised to do decryption for particular browser cookies, which triggers Sending commands to TPM for Decryption section.

3.4.7. Sending commands to TPM for Decryption

When the user opens the browser for browsing, then the project sends commands to TPM for decryption by checking if there is any cookies existed in project installation folder. After decryption the cookies were transferred to particular browser installation folder which can be used by user can for browsing.
Chapter 4

4. Trusted Computing

Trusted Computing (TC) is a technology developed and promoted by the Trusted Computing Group. The term is taken from the field of trusted systems and has a specialized meaning. With Trusted Computing the computer will consistently behave in specific ways, and those behaviours will be enforced by hardware and software. Enforcing this Trusted behaviour is achieved by loading the hardware with a unique ID and unique master key and denying even the owner of a computer knowledge and control of their own master key. Trusted Computing is extremely controversial as the hardware is not merely secured for the owner; enforcing Trusted behaviour means it is secured against the owner as well [24].

4.1. Key concepts

Trusted computing encompasses five key technology concepts, of which all are required for a fully trusted system.

- Endorsement key
- Secure input and output
- Memory curtaining / protected execution
- Sealed storage
- Remote attestation

4.1.1. Endorsement key

The endorsement key is a 2,048-bit RSA public and private key pair, which is created randomly on the chip at manufacture time and cannot be changed. The private key never leaves the chip, while the public key is used for attestation and for encryption of sensitive data sent to the chip, as occurs during the TPM_TakeOwnership command.
This key is used to allow the executions of secure transactions: every Trusted Platform Module (TPM) is required to sign a random number, using a particular protocol created by the trusted computing group (the direct anonymous attestation protocol) in order to ensure its compliance of the TCG standard and to prove its identity; this makes it impossible for a software TPM emulator to start a secure transaction with a trusted entity. The TPM should be designed to make the extraction of this key by hardware analysis hard, but tamper-resistance is not a strong requirement.

4.1.2. Secure I/O

Secure input and output (I/O) refers to a protected path between the computer user and the software with which they believe they are interacting. On current computer systems there are many ways for malicious software to intercept data as it travels between a user and a software process — for example keyboard loggers and screen-scrappers. Secure I/O reflects a hardware and software protected and verified channel, using checksums to verify that the software used to do the I/O has not been tampered with. Malicious software injecting itself in this path could be identified. Secure I/O is traditionally known as a trusted path.

4.1.3. Memory curtaining

Memory curtaining extends common memory protection techniques to provide full isolation of sensitive areas of memory — for example, locations containing cryptographic keys. Even the operating system does not have full access to curtained memory, so the information would be secure from an intruder who took control of the OS.

4.1.4. Sealed storage

Sealed storage protects private information by binding it to platform configuration information including the software and hardware being used. This means the data can be read only by the same combination of software and hardware. For example, users who keep a song on their computer that has not been licensed to be listened will not be able to play it. Currently, a user can locate the song, listen to it, and send it to someone else, play it in the software of their choice, or back it up (and in some cases, use circumvention software to
decrypt it, such as hymn). Alternately the user may use software to modify the operating system's DRM routines to have it leak the song data once, say, a temporary license was acquired. Using sealed storage, the song is securely encrypted so that only the unmodified and untampered music player on his or her computer can play it.

4.1.5. Remote attestation

Remote attestation allows changes to the user's computer to be detected by authorized parties. That way, software companies can avoid users tampering with their software to circumvent technological protection measures. It works by having the hardware generate a certificate stating what software is currently running. The computer can then present this certificate to a remote party to show that its software has not been tampered with.

Remote attestation is usually combined with public-key encryption so that the information sent can only be read by the programs that presented and requested the attestation, and not by an eavesdropper, such as the computer owner.

4.2. Applications of Trusted Computing

4.2.1. Digital rights management

Trusted Computing would allow companies to create a Digital rights management system which would be very hard to circumvent, though not impossible. An example is downloading a music file. Remote attestation could be used so that the music file would refuse to play except on a specific music player that enforces the record company's rules. Sealed storage would prevent the user from opening the file with another player or another computer. The music would be played in curtained memory, which would prevent the user from making an unrestricted copy of the file while it is playing, and secure I/O would prevent capturing what is being sent to the sound system. Circumventing such a system would require either manipulation of the computer's hardware, capturing the analogue (and possibly degraded) signal using a recording device or a microphone, or breaking the encryption algorithm.
4.2.2. Identity theft protection

Trusted Computing could be used to prevent identity theft. Take for example, online banking. Remote attestation could be used when the user is connecting to the bank's server and would only serve the page if the server could produce the correct certificates. Then the user can send his encrypted account number and PIN, with some assurance that the information is private to him and the bank.

4.2.3. Preventing cheating in online games

Trusted computing could be used to combat cheating in online games. Some players modify their game copy in order to gain unfair advantages in the game; remote attestation, secure I/O and memory curtaining could be used to verify that all players connected to a server were running an unmodified copy of the software.

It should be noted that this only applies to games with a poor security design, in which the game client must be rendered as "trusted" in order to perform game arbitration which normally belongs to the server.

4.2.4. Protection from viruses and spyware

Digital signature of software will allow users to identify applications modified by third parties that could add spyware to the software. For example, a website offers a modified version of a popular instant messenger that contains spyware as a drive-by download. The operating system could notice the lack of a valid signature for these versions and inform the user that the program has been modified. Of course this leaves open the question of who determines if a signature is valid.

Trusted computing might allow increased protection from viruses. A possible improvement in virus protection would be to allow antivirus vendors to write software that could not be corrupted by virus attacks. However, as with most advanced uses of Trusted Computing technology, preventing software corruption necessitates a Trusted Operating System, such as
Trusted Gentoo In practice any operating system which aims to be backwards compatible with existing software will not be able to protect against viruses in this way.

4.2.5. Protection of biometric authentication data

Biometric devices used for authentication could use trusted computing technologies (memory curtaining, secure I/O) to assure the user that no spyware installed on his/her PC is able to steal sensitive biometric data. The theft of this data could be extremely harmful to the user because while a user can change a password if he or she knows that the password is no longer secure, a user cannot change the data generated by a biometric device.

4.2.6. Verification of remote computation for grid computing

Trusted computing could be used to guarantee participants in a grid computing system are returning the results of the computations they claim to be instead of forging them. This would allow large scale simulations to be run (say a climate simulation) without expensive redundant computations to guarantee malicious hosts are not undermining the results to achieve the conclusion they want.

The information about TPM and taking ownership of TPM was explained in following sections.
Chapter 5

5. Trusted Platform Module (TPM)

5.1. Introduction

Trusted Platform Module (TPM) is a microcontroller that stores keys, passwords and digital certificates. It's typically affixed to the motherboard of a PC. The nature of this silicon ensures that the information stored there is made more secure from external software attack and physical theft. Security processes, such as digital signature and key exchange, are protected through the secure TCG subsystem.

Access to data and secrets in a platform could be denied if the boot sequence is not as expected. Critical applications and capabilities such as secure email, secure web access and local protection of data are thereby made much more secure. TPM capabilities also can be integrated into other components in a system and it is often called "TPM chip", "Fritz chip" or "TPM Security Device".

A Trusted Platform Module offers facilities for the secure generation of cryptographic keys, and limitation of their use, in addition to a hardware pseudo-random number generator. It also includes capabilities such as remote attestation and sealed storage. Remote attestation creates a nearly unforgeable hash key summary of the hardware and software configuration. The extent of the summary of the software is decided by the program encrypting the data. This allows a third party to verify that the software has not been changed. Sealing encrypts data in such a way that it may be decrypted only if the TPM releases the associated decryption key, which it only does for software that can provide the same password that was supplied when software "ownership" of the TPM was initially configured. Binding encrypts data using the TPM endorsement key, a unique RSA key burned into the chip during its production, or another trusted key descended from it.
A Trusted Platform Module can be used to authenticate hardware devices. Since each TPM chip has a unique and secret RSA key burned in as it is produced, it is capable of performing platform authentication. For example, it can be used to verify that a system seeking access is the expected system [25].

5.2. **Trusted Platform Capabilities**

A generic TP must have the following [26].

*Shielded locations*: protected storage are designed for sensitive information (e.g. integrity metrics or cryptographic keys).

*Protected capabilities*: commands which exclusively manage data in shielded locations

- Integrity reporting,
- Key management,
- Random number generation and
- Sealing and binding of data

*Integrity measurement, logging and reporting*: include the calculation, storage and reporting of metrics over the state or (persistent) characteristics of a certain platform.

*Attestation mechanisms*: A TP has to provide different forms of attestation mechanisms that allow an external party to verify the accuracy of a certain piece of information known to the TP.

5.3. **Attestation mechanisms**

*Attestation by the TPM*: The TPM proves the possession of particular data by applying digital signatures with keys only known to the TPM

- Any payload can be signed and attested with AIKs

*Attestation to the platform*: This operation assures that a platform can be trusted for reporting integrity measurements (metrics describing the state or the characteristics of the platform) –

- Uses platform credentials, a concept similar to CA certificates

*Attestation of the platform*: The platform proves (possession and veracity) of a set of its current integrity measurements to an (external) verifier
- Signing of PCRs with AIKs

**Authentication of the platform:** The process of confirming TP’s identity.

- “Platform Authentication is performed using any non-migratable signing key. Certified keys (i.e. signed by an AIK) have the added semantic of being attestable. Since there are an unlimited number of non-migratable keys associated with the TPM, there are an unlimited number of identities that can be authenticated” *(TCG Arch Overview)*

### 5.4. **TPM Characteristics**

The TPM cannot be moved
- Attached to the platform

The TPM contains
- Cryptographic engine
- Protected storage

Functions and storage are isolated
- Provides a “Trust Boundary”

TCG defines TPM’s functionality
- Protected capabilities
- Shielded locations

Not the implementation
- Vendors are free to differentiate the TPM implementation
- Must still meet the protected capabilities and shielded locations requirements
5.5. **TPM component architecture**

![TPM Architecture Diagram]

**Shielded Locations**
- Non-Volatile Storage
- Platform Configuration Register (PCR)
- Attestation Identity Keys (AIK)
- Program Code
- Random Number Generator

**Packaging**
- Up to 2048 Bit, RSA asymmetric
- Operational state control, physical presence
- Protected capabilities

**Physical!**
- > 1280 Bytes
- ≥16 x 160 Bit

**Figure 33 TPM Architecture**

5.6. **Purpose of Design**

The main goal to design TPM is to stop the attacks from attackers on your system with various softwares, spywares and virus, you can say we can find so many softwares to prevent these things but only software can’t help to protect the system so the Trusted Computing Group recommended a hardware which can stop the attackers and the more purpose of designing this hardware was explained below [27].

Modern systems are incredibly complex. A typical UNIX or Windows system, including standard applications, represents something around 100 million lines of source code. Large applications often also have hundreds of millions lines of code. All together, there are billions of lines of code in use today. Several recent studies have shown that typical product-level software has roughly one security-related bug per thousand lines of source code across its
lifetime. Thus, a typical system will potentially have a hundred thousand security bugs. It is not surprising that we are finding six thousand of these bugs per year, and that the rate of finding them has increased so dramatically.

Second, compatibility requirements in the client space make a complete break from the commitment to the current system architecture unlikely. Even if we could build secure software systems, the amount of effort needed to replace the billions of lines of code in and for existing operating systems is simply prohibitive.

Third, without hardware support, it is likely impossible to detect the presence of malicious code in a system. This question is still an active area of research; but so far, all attempts to detect malicious changes in software (compromise) without hardware support have ultimately been circumvented. In contrast, with a little bit of hardware support, it is quite easy to detect compromise.

In Trusted Computing, the goals are to protect the most sensitive information, such as private and symmetric keys, from theft or use by malicious code. Trusted Computing assumes that client software is going to be compromised at some time during its life, and provides protection for its sensitive keys in case this should happen. The TPM has rather limited functionality, but that is an advantage when it comes to certifying that it works as designed. And the design has been made deliberately flexible so that it can be applied to almost any problem that occurs in the security field.

The TPM has been designed to protect security by ensuring the following:

• Private keys cannot be stolen or given away.
• The addition of malicious code is always detected.
• Malicious code is prevented from using the private keys.
• Encryption keys are not easily available to a physical thief.

The TCG chip accomplishes these goals with three main groups of functions, as follows:

• Public key authentication functions
• Integrity measurement functions
• Attestation functions
The public key authentication functions provide for on-chip key pair generation using a hardware random number generator, along with public key signature, verification, encryption, and decryption. By generating the private keys in the chip, and encrypting them anytime they are transferred outside the chip, the TPM guarantees that malicious software cannot access the keys at all. Even the owner of the keys cannot give the private keys away to phishing or pharming attacks, as the keys are never visible outside the chip unencrypted. Malicious code could use the private keys on the TPM, so some way needs to be provided to ensure that malicious code cannot use the keys either.

The integrity measurement functions provide the capability to protect private keys from access by malicious code. In a trusted boot, the chip stores in Platform Configuration Registers (PCRs) hashes of configuration information throughout the boot sequence. Once booted, data (such as private keys) can be “sealed” under a PCR. The sealed data can be unsealed only if the PCR has the same value as at the time of sealing. Thus, if an attempt is made to boot an alternative system, or a virus has “backdoored” the operating system, the PCR value will not match and the unseal will fail, thus protecting the data from access by the malicious code.

The attestation functions keep a list of all the software measurements committed to the PCRs, and can then sign them with a private key known only by the TPM. Thus, a trusted client can prove to a third party that its software has or has not been compromised.

Malicious programs, such as spyware and Trojans, will be detected by changes in the PCR measurement, which can then cause the TPM to refuse to unseal sensitive data, or to refuse to use private keys for signing or decryption. If vulnerable or misconfigured programs are exploited, any changes they make to files can similarly be detected, and sensitive data protected. Any attempts to gain authentication secrets, such as by phishing or pharming, will fail, as the owner of the authentication private keys cannot give the keys away. Data encrypted under keys sealed by the TPM will be much harder to access in the case of theft, as the attacker would need to open up the chip to get its storage root key in order to be able to unseal the protected keys. (While possible, this is really difficult and expensive.) Similarly, encrypted communications are much more immune to eavesdropping, if the encryption keys are exchanged or stored by a TPM.
In addition to privacy support, the Trusted Computing Group (TCG) technical committee had a number of design goals for the Trusted Platform Module (TPM). It was important that the design have the capability to do the following:

- Securely report the environment that booted
- Securely store data
- Securely identify the user and system (without encountering privacy concerns)
- Support standard security systems and protocols
- Support multiple users on the same system while preserving security among them
- Be produced inexpensively.

These potential other uses have given rise to privacy concerns. Consequently, to address these concerns, the TPM chip cannot be enabled via software alone - a "physical presence request" operation is required, whereby a human sitting at the computer must acknowledge the request to activate the device via a prompt at BIOS level. Furthermore, each application that uses the TPM must register a unique passphrase when it takes ownership of the TPM in order to prevent other applications from also making unauthorized use of the TPM while it's enabled. Future operating systems are expected to have increased TPM support for additional cryptographic features.

### 5.7. TPM basic Functions

- Generation of asymmetric and symmetric keys,
- Calculation of signatures and hash values,
- Asymmetric and symmetric encryption,
- Encryption of cryptographic keys (binding),
- Secure storage (shielded locations) and processing (protected capabilities) of small objects and hash values (of measurement values from the platform configuration),
- Creation of signed reports on the measurement values,
- Key management
- (Endorsement and attestation identity keys),
- Functions to let the owner of the platform take possession of the platform and (de) activate the TPM,
- A trustworthy timer.
- Minimum requirement: monotonic increment every 5 secs for at least 7 years. No reset–useful against roll-back attacks
5.8.  **Key types and classes**

- **EK**
- **Storage Keys**
  - Protects keys or external data
  - Storage Root Key SRK
  - Subordinate storage keys, generated by TPM and protected by SRK
- **Signing Keys**
  - Digital signatures
- **Attestation Identity Keys (AIKs)**
  - Special Signing keys
  - Provide pseudonymous attestation
  - Can be produced in unlimited number
  - Non-migratable
- **Non-Migratable Keys**
  - Permanently bound specific TPM, i.e., platform
- **Migratable Keys**
  - Can be migrated to other platforms
- **Certified Migration Keys**
  - Can be migrated only to “certified” authorities
- **Bind keys**
  - To encrypt small data packages (e.g. symmetric keys)
- **Legacy keys**
  - Created outside a TPM and imported
- **Authentication keys**
  - Symmetric keys to protect transport sessions

5.9.  **Root of Trust**

A **Root-of-trust** is a component that must behave as expected, because its misbehaviour cannot be detected.

- Roots of trust enable the gathering, storage and reporting of evidence/references about the trustworthiness of software environment running on the platform.
They represent the components of a TP which must be implicitly trusted if the evidence/references are to be trusted.

*A Root of Trust for Measurement (RTM)*: The component that can be trusted to reliably measure the software/firmware which executes after some sort of reset (e.g. BIOS ext)
- Provides cryptographic mechanism to digitally sign TPM state and information

*A Root of Trust for Reporting (RTR) and a Root of Trust for Storage (RTS)*: The components that can be trusted to store and report reliable information in and about the platform.
- The RTS Provides cryptographic mechanism to protect information held outside of the TPM, can be as simple as a key

The Core Root of Trust for Measurement (CRTM) and the Dynamic Root of Trust for Measurement (DRTM) are the roots of trust for measurement.
- For the foreseeable future, it is envisaged that the static-RTM will be integrated in the normal computing engine of the platform, where the provision of additional BIOS boot block or BIOS instructions (the CRTM) cause the main platform processor to function as the RTM.
- Static RTM is CPU after platform reset.

The TPM is the root of trust for reporting and the root of trust for storage.

**5.9.1. Key protection by the RTS**
- RTS manages a small amount of volatile memory where keys are held while performing signing and decryption operation
- SRK protects first level keys with in TPM
- Inactive keys may be encrypted and moved off-chip to make room for other more active keys.
- Management of the key slot cache is performed external to the TPM by a Key Cache Manager (KCM).
- RTS is optimized to store small objects roughly the size of an asymmetric key (e.g. ~210 byte payload). A variety of object types can be stored, such as symmetric and asymmetric keys, pass-phrases, cookies, authentication results and opaque data.
Figure 34 TPM/RTS

Figure 35 Key Hierarchy – upper level encrypt lower levels
5.10. **TPM operational states**

**Enabled/ disabled**
- TPM may be enabled/ disabled multiple times within a boot period. Disabled, the TPM restricts all operations except the ability to report TPM capabilities and to update PCRs. When enabled, all features of the TPM are available.
  - SHA still available
  - Ownership can be disabled
  - Persistence flag

**Activated/ deactivation**
- Deactivation is similar to disabled, but operational state changes are possible. (i.e. change owner or activation with physical presence). When activated all features of the TPM are available.
  - Persistent
  - Does not take effect until next re-initialisation

**Owned/ un-owned**
- A platform is owned when an EK exists and the true owner knows owner authorization data. The owner of a platform may perform all operations including operational state changes.

### 5.10.1. **Combinations of operational states**

<table>
<thead>
<tr>
<th>State #</th>
<th>E</th>
<th>A</th>
<th>O</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Fully operational state</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>X</td>
<td>X</td>
<td>Ownership is set and can be changed to new owner</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>X</td>
<td>nonsensical</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>Ownership is locked, cannot be changed</td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>X</td>
<td></td>
<td>Local and remote ownership possible</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>X</td>
<td>Ownership can be set</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>X</td>
<td></td>
<td>nonsensical</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td>TPM off, default shipping setting</td>
</tr>
</tbody>
</table>

Table 7 Truth table for TPM Operational States
A normal ‘TPM deployment procedure at the end-user as owner

- S8 – S6 – S5 – take ownership – S2 – enable – S1State

5.11. Ownership on TPM

TPM has no owner when shipped (to system integrator or user)

Taking ownership normally requires physical presence

- A user action that can’t be performed by software (yet possibly remote)
- Required in part. To enable TPM and establish initial ownership
- Implementation examples
  - Button directly wired to TPM (some IBM Laptops)
  - BIOS setup…
  - Future: Biometry?

TPM ownership can always be reset via physical presence–old secrets are discarded
TPM ownership can be asserted via physical presence–no secrets are exposed

Uses the Endorsement Key

- Nonmigratable2048 RSA pair (PUBEK, PRIVEK)
- Created by TPM manufacturer, unique per TPM
- PRIVEK never leaves TPM

Control state to be active, enabled
Optionally control physical presence

TPM_TakeOwnership (OwnerAuth …)

- Creates160-bit authentication value, encrypts with PUBEK and stores it in non-volatile memory
- Creates2048-bit RSA Storage Root Key (SRK) on TPM
- Cannot run TPM_TakeOwnership again: Ownership Enabled flag ← False
- Done once by IT department or laptop owner.
5.11.1. Taking TPM ownership on Windows

5.11.1.1. Initializing a TPM for First Use

Initializing a TPM configures it for use on a computer. The initialization process involves turning on the TPM and then setting ownership of the TPM. Although Windows Vista supports remote initialization of a TPM, you must have local access to the computer to turn on the TPM. On some new computers, the TPM is turned on by default. If this is the case with the computer you are working with, you can complete the initialization of the TPM remotely.

To initialize the TPM on your computer for first use, complete the following steps:

Log on locally to the computer with local administrator credentials.
Start the Trusted Platform Module Management console.
Under Actions, click Initialize TPM to start the TPM Initialization Wizard. On the Welcome page, click Next.

The next step depends on the state of the TPM:

If the TPM Initialization Wizard detects a BIOS that does not meet Windows Vista requirements, you will not be able to continue with the wizard. Instead, you will be alerted to consult the computer manufacturer’s documentation for instructions on turning on the TPM.
If the TPM is turned off, the TPM Initialization Wizard displays the Turn On The TPM Security Hardware page. Follow the instructions for turning on the TPM. Click Shutdown (or Restart), and then follow the BIOS screen prompts. After the computer restarts, confirm that you want to turn on the TPM when prompted.
If the TPM is already turned on, the first page you see is the Create The TPM Owner Password page. For details about setting the owner password, see the next procedure.

The second part of initializing the TPM for first use is setting ownership. By setting ownership of the TPM, you are assigning a password that helps ensure that only the authorized TPM owner can access and manage the TPM. The TPM password is required to turn off the TPM if you no longer want to use it and to clear the TPM if the computer is to be recycled.
To set the ownership of the TPM on your computer, complete the following steps:

- Log on locally to the computer with local administrator credentials.
- Start the Trusted Platform Module Management console.
- Under Actions, click Initialize TPM to start the TPM Initialization Wizard. On the Welcome page, click Next.
- On the Create The TPM Owner Password page, select Automatically Create The Password (Recommended), and then click Next.
- On the Save Your TPM Owner Password page, click Save, and then select a location to save the password. Ideally, you’ll save the TPM ownership password to removable media, such as a universal serial bus (USB) flash drive.
- Click Save again. The password file is saved as computer_name.tpm.
- Click Print if you want to print a hard copy of your password. Be sure to save the printout containing the password in a secure location.
- Click Initialize. The initialization process might take several minutes to complete.
- When initialization is complete, click Close. The status of the TPM is displayed under Status in the TPM Management console.

5.11.1.2. Turning off and Clearing the TPM

New computers that have a TPM might arrive with the TPM turned on by default. If you decide not to use the TPM, you should turn off and clear the TPM. If you want to reconfigure or recycle a computer, you should also turn off and clear the TPM. Windows Vista supports remotely turning off and clearing a TPM as well as using scripts to turn off and clear a TPM.

To turn off the TPM, complete the following steps:

- Log on locally to the computer with local administrator credentials.
- Start the Trusted Platform Module Management console.
- Under Actions, click Turn TPM Off.

In the Turn Off The TPM Security Hardware dialog box, select one of the following methods for entering your password and turning off the TPM:

- If you have the removable media on which you saved your TPM owner password, insert it, and then click I Have A Backup File With The TPM Owner Password. In the Select Backup File With The TPM Owner Password dialog box, click Browse, and
then use the Open dialog box to locate the .tpm file saved on your removable media. Click Open, and then click Turn TPM Off.

- If you do not have the removable media on which you saved your password, click I Want To Type The TPM Owner Password. In the Type Your TPM Owner Password dialog box, type your password (including dashes), and then click Turn TPM Off.
- If you do not know your TPM owner password, click I Don’t Have The TPM Owner Password, and then follow the instructions provided to turn off the TPM without entering the password. Because you are logged on locally to the computer, you will be able to turn off the TPM.

Clearing the TPM cancels the TPM ownership and finalizes the shutdown of the TPM. You should clear the TPM only when a TPM-equipped client computer is to be recycled or when the TPM owner has lost the TPM owner password and recovery information was not backed up.

**To clear the TPM, complete the following steps:**

- Log on locally to the computer with local administrator credentials.
- Start the Trusted Platform Module Management console.
- Under Actions, click Clear TPM.

**Note:**

Clearing the TPM resets it to factory defaults and finalizes its shutdown. As a result, you will lose all created keys and data protected by those keys.

In the Clear The TPM Security Hardware dialog box, select a method for entering your password and clearing the TPM:

- If you have the removable media on which you saved your TPM owner password, insert it, and then click I Have A Backup File With The TPM Owner Password. In the Select Backup File With The TPM Owner Password dialog box, click Browse, and then use the Open dialog box to locate the .tpm file saved on your removable media. Click Open, and then click Clear TPM.
If you do not have the removable media on which you saved your password, click I Want To Type The TPM Owner Password. In the Type Your TPM Owner Password dialog box, enter your password (including dashes) and then click Clear TPM.

If you do not know your TPM owner password, click I Don’t Have The TPM Owner Password, and then follow the instructions provided to clear the TPM without entering the password. Because you are logged on locally to the computer, you will be able to clear the TPM.

The status of the TPM is displayed under Status in the TPM Management console.

5.11.1.3. Changing the TPM Owner Password

If you suspect that the TPM owner password has been compromised, you can change the password by using the Trusted Platform Module Management console. To change the TPM owner password, complete the following steps:

- Log on locally to the computer with local administrator credentials.
- Start the Trusted Platform Module Management console.
- Under Actions, click Change Owner Password.
- Follow the prompts to provide the current password and change the password.

5.11.2. Taking TPM ownership on Linux

It was explained in the tpm4java which is another requirement to run my project.

5.11.3. Clearing the TPM ownership

If you do not know the password or if you are interested to remove the password, you should go to the BIOS and reset the TPM password. In order to reset the TPM password you must follow your manufactures instructions on how to access the computer BIOS. Please follow the instructions for your particular computer.
5.12. **TPM Emulator**

If there is no TPM in your system for example in older desktops and you are interested to get TPM features in your system, you have to install TPM Emulator, which can be downloaded from the following site: https://developer.berlios.de/project/showfiles.php?group_id=2491 and follow the installation instructions after file unzip.

5.13. **Uses**

5.13.1. **Disk encryption**

Full disk encryption applications, such as the BitLocker Drive Encryption feature of Microsoft's Windows Vista and Windows Server 2008 operating systems, can use this technology to protect the keys used to encrypt the computer's operating system volume and provide integrity authentication for a trusted boot pathway (i.e. BIOS, boot sector, etc.) A number of third party full disk encryption products also support the TPM chip [29].

Almost any encryption-enabled application can in theory make use of a TPM, including:
- Digital Rights Management
- Software license protection & enforcement
- Password protection

5.14. **Miscellaneous**

5.14.1. **Trusted Platform Module microcontrollers manufacturers**

Atmel
Broadcom
Infineon (Infineon TPM)
Intel (via AMT as iTPM)
Sinosun
STMicroelectronics
Winbond
Chapter 6

6. Implementation of the Project

6.1. Concept

The main theme of the project is to provide the security for browser cookies and passwords in Linux Operating System. Initially when the user installs this project, then it checks for is there any browser running, if not it encrypts the cookies stored in browser installation folder using the TPM key and copies from the browser installation folder to the project installation folder and protects them. After that whenever the user opens the browser, then the project detects the browser was running in system and then it decrypts the cookies stored in hidden folder where our project was installed using the key generated by TPM and then transfers the cookies to the browser installation folder so that the browser can use the cookies stored by user and user can continue his things using browser. When user stops browsing by closing his browser then our project does the reverse process, i.e. it encrypts the cookies stored in browser installation folder using TPM key for encryption and then copies to the project installation folder and saves them and protects them. If again user opens the browser then the process continues explained above and this total process continue until the user shut-down the system or he exits from the project.

6.2. Installation and Usage

There is no special procedure for installation, just download the file and extract it and make it run by login into the folder where you saved the download file and through terminal using the following command.

java -jar project.jar
6.3. **Conditions to make Project work**

Before running the project for the first time check the requirements mentioned above and after installation of the project i.e running the project, if you get any errors then make it clear have you taken the TPM ownership using your owner authorization password and SRK (Storage Root Key) password, if not try to take the ownership of your TPM if there exists TPM in your system by reading the manual from system provider, if not then take install the TPM Emulator by downloading from web address mentioned in Requirements section and then take the ownership of TPM Emulator which works like TPM by running the program TakeOwnership.java in project installation folder by entering your owner authorization password and SRK (Storage Root Key) password by copying the program to the security place where you feel or after running save it some where in your system by removing the owner_auth and srk_auth variables in program, after running this program and again if you get any errors, ensure that your TPM is running and if not make it run and if you have installed TPM emulator make it run by running the following command where you installed it

```bash
TPM_emulator_installed_folder:/ # modprobe tpmd_dev
TPM_emulator_installed_folder:/ # tpmd -f clear
```

If you are interested to change your TPM ownership password and SRK password then run the changeAuth.java program in project installation folder and then change the SRK Password in project also by clicking *changeSRK* button which appears on the window when you run the project.

**Caution:** Don't open the browser before running the project after first running because if you close your browser after first running, then the browser cookies are decrypted and protected by the project, so that you cannot able to access the browser cookies for your use, so open the browser after running the project only which is secure and the right way to secure your new cookies after your next browse and try to run the project every time instead of clicking *remember password* cookie which appears on window when you run the project.
The project implementation figure was shown below.

![Project Implementation Diagram](image)

**Figure 36 Project Implementation**

### 6.3.1. Initialisation of project

Initialise the project before running the project i.e. checking the requirements needed to run the project and then initialise TPM or TPM emulator by taking ownership and run them, and then run the project by the command given above in installation section.

### 6.3.2. Login to Project

Login into the project by entering the following fields:

*Username:* Enter the username of the user of the system which will be used to run your project under user and don't enter any another username instead of that it might raise errors while running the project.
Password: Enter the password of your project if you already created account to run the total project, if not enter password of characters equal and more than 12 characters including at least one upper case, lower case letters, numbers and symbols and then click create account if you filled all the necessary fields and if you have forgotten the password, re-install the program, means delete all the contents where you installed the project, so don't try to forgot the password and there will be no assistance to remember your password again because the password was stored in the installation folder is a hashed one for security purpose, so there will be no assistance.

If you are interested to change the password then click changePassword button on login screen and then enter your old password and the new password you are interested.

SRK Password: Enter the TPM SRK password which is used while taking the ownership of the TPM. SRK password is used to send commands to TPM to encrypt and decrypt data sent while running the project.

If you change your TPM ownership with different Owner and SRK passwords, then change the SRK password in the project also by clicking changeSRK button in login screen and then enter old and new SRK passwords.

The login screen to run the project, changePassword and change changeSRK password screens for changing corresponding passwords are shown below.
Figure 37 Login screen to run the Project.
Figure 38 Changing Password for Project
After entering the above information, the project checks whether the corresponding passwords are correct or not, if they are correct and then the project authenticates user and run the project with infinite loop.

6.3.3. Authentication process

When user enters the password to login to project, the program authentication reads the hashed password and salt from already saved password while create account and then converts the entered password into hash password called authenticate password using the salt read from saved password, and then it compares saved hash password and newly hash password, if they both are equal and the program authenticates the user to run the project, if passwords are not correct it cannot authenticate and gives error with Invalid Password dialogue.
6.3.4. Creating User Process

When user enters the password, then the project checks for all possibilities whether he entered right combination of password or not, means it checks if entered password is with combination of at least one upper case, lower case, numbers and symbols, and checks the length of the password if it is equal to 12 or more characters and then if all the conditions are correct then it creates the user account by converting the entered password into hashed password including salt which was generated on random process using SHA-1 algorithm. If any condition was wrong the project raises a dialogue box with the corresponding error to enter correct combination of password.

6.4. Project implementation

After authenticating user the program checks whether there is already encrypted cookies are there or not for Mozilla Firefox browser and Opera browser, if there is no encrypted cookies in the project installation folder, then it gets the directory path where the cookies are stored in browser installation folder and the directory path to store corresponding output encrypted cookies to the encrypted section.

6.4.1. Encryption

When the program sends the input file (browser cookie file) to encrypt and output file (encrypted file) where to save it, then the program reads SRK password saved in project installation folder and then passes these three variables to tpm4java TSSHighLevel to encrypt and saves the encrypted data in installation folder and hides it, after encryption the program deletes the cookie file in browser installation folder. In Encryption the project does encryption for cookies.txt and signons2.txt file for Mozilla Firefox browser and cookies4.dat file for Opera browser.

6.4.2. Browser Checking

Initially after encryption of cookies, the program goes into infinite mode where the program runs continuously and checks for the system process for every second and then if it finds any browser the program checks which browser it is and then switches to Decryption section to
decrypt the corresponding cookies and deletes hidden encrypt cookies in project installation folder. Then the browser checking process will continue until the user stops browsing and closes browser, then the program switches to Encryption section and do the corresponding process.

6.4.3. Decryption

When the browser checking process finds a browser it directs to decryption section where the following process occurs. In decryption process the program sends hidden cookie file for corresponding browser, for example if the detected browser is Mozilla Firefox browser then it sends path of encrypted cookies.txt and signons2.txt files and if browser is Opera then it send path of encrypted cookies4.dat to decrypt and path to save decrypted files i.e browser installation folder. Then the decryption section reads SRK password stored in project installation folder and sends these three variables to tpm4java TSSHighLevel to do decryption and after decryption the program deletes encrypted cookie files in project installation folder and saves the corresponding browser cookies in their installation folders.
Chapter 7

7. Conclusion

7.1. Final Thoughts: Better Browsing

Users will reap some benefits from upgrading to any of the browsers, and all are excellently engineered, well-working software with lots of convenience, capability, and security.

The tab interface in Opera—by far the most mature tab implementation of the bunch—is the easiest to use and understand: At the left of the tabs, there's a "New Tab" mini-tab, with an icon of a page and a plus sign. What could be clearer? In IE7, you don't see that you can add a tab unless you happen to hover the cursor over the small empty tab top to the right of the currently active one; in Firefox 2, you can get a new tab if you know to double-click the space to the right of the last tab, right click the tab and pick New tab from the choices, or actually go up to the File/New tab choice on the main menus. Opera, too, was the fastest to load in our testing and conformed to the standards tested by the Acid2 Browser test [11].

Opera alone currently lacks an anti-phishing feature, but I wonder how critical this lacuna is; how about people just learning to be a little circumspect about where they enter their passwords and personal info? If you enter your bank's URL, you're safe, if an email sends you to a site that looks like your banks, beware. More helpful is Microsoft's anti-malware tool, Defender, which scans your system for spyware you may catch while browsing.

If people switched to Firefox for tabbed browsing windows and extensions, they'll no longer have those reasons to shun Microsoft's slick new browser—as long as they can wait for its release. Microsoft seems to have enough confidence in the stability of Beta 3 to put its download button on the front page of its Internet Explorer site, rather than hidden in some developer area of the site. With IE7 so seemingly fully baked, we have to wonder why Microsoft is waiting for Vista to release it. Oh yeah, I guess they want another reason for you to buy Vista.
If the switch to Firefox was made for security—a major motivator, with all the holes found in IE and well publicized, the question is harder to answer. Microsoft has really taken security to heart, but only time will tell. With Firefox becoming such a market leader and boasting so much programmability, it's bound to become a prime target for hackers. For privacy, all have a "Delete Private Data" choice under their Tools menu—a nice way to keep people from seeing where you've been on the web.

IE7 now emulates the other two browsers by finally including a built-in search text box in the browser—yes you could always type your search into the address bar and get MSN search results, but the address bar is for URLs, not searches. RSS handling is another area where the other two have caught up to Opera, and all the browsers now do a decent job handling your subscriptions.

7.2. Summary

It's clear from these results that Opera was the best browser, followed by Mozilla Firefox and then Internet explorer 7. Then came the gecko-based browsers, and finally the other trident-based browsers. Which browser you choose depends on your taste, if you must have extreme customization, go with a gecko-based browser like k-meleon or Firefox. If you're looking for the absolute fastest, smallest browser, go with opera. IE 7 is a fine, slim browser, but it's speed doesn't stand out. Other than IE 7, you should avoid trident-based browsers. I currently use Firefox, but I'm not at all impressed with it's speed or resource usage. The only reason I keep using it is that I'm so used to my extensions; it would be a major change to live without them. If Firefox 3 isn't a major improvement, its nice to seriously consider switching to opera.

7.3. Final Word

By choosing browser on your own interest and after installation run the LIBROWS software before using which provides better security to your credential information stored in cookies and your passwords.
Appendix

8. Contents of the CD

1. Thesis

The electronic version of the thesis:
Path: /

2. Source Code

In the following folder, all the input and output data files and the source code of the applications are stored:
Path: /Sourcecode/

3. Documentation

The documentation of the source code is stored in the following folder:
Path: /Documentation/

4. Software

In the following folder, all the used programs and libraries are stored:
Path: /Execution File/
List of Acronyms

9. Acronyms

AIK  Attestation Identity Key
AMT  Active Management Technology
ATM  Automated teller machine

BIOS Basic Input/Output System

CRHF Collision resistant hash functions
CRTM Core Root of Trust for Measurement
CSS Cascading Style Sheets (CSS)

DAA Direct Anonymous Attestation
DH Diffie-Hellman Key Agreement Algorithm
DOM Document Object Model (DOM)
DRTM Dynamic Root of Trust Measurement
DSA Digital Signature Algorithm

EK Endorsement Key

FTP File Transfer Protocol

HMAC Hashed Message Authentication Code
HTML HyperText Markup Language
HTTP Hypertext Transfer Protocol (HTTP)
HTTPS Hypertext Transfer Protocol over Secure Socket Layer

IAIK Institute of Applied Information Processing and Communication
IMAP Internet message access protocol
IDM Identity Management

List of Acronyms

IMA Identity Management Architecture
IRC Internet Relay Chat (IRC)

KCM Key Cache Manager

MAC Message authentication code
MDC Modification detection codes
MD5 Message-Digest algorithm 5
MIME Multipurpose Internet Mail Extensions

NNTP Network News Transfer Protocol

OWHF One-way hash functions

PCR Platform Configuration Register
PDA personal digital assistant
PE Platform (endorsement)
PEP Policy Enforcement Point
PIN personal identification number
POP Post Office Protocol
PRIVEK Private EK
PUBEK Public EK

RFC Request for Comments
RM Reference Manifest
RTM Root of Trust for Measurement
RTR Root of Trust for Reporting
RTS Root of Trust for Storage

SHA Secure Hash Algorithm
List of Acronyms

**SMTP** Simple Mail Transfer Protocol
**SOAP** Simple Object Access Protocol
**SRK** Storage Root Key

**TBB** Trusted Building Block
**TCG** Trusted Computing Group
**TCS** TSS Core Services
**TDDL** TPM Device Driver Library
**TPM** Trusted Platform Module
**TSP** TCG Service Provider
**TSS** Trusted Software Stack

**URL** Uniform Resource Locator
**USB** Universal Serial Bus

**XSS** Cross-site scripting

**WYSIWYG** What You See Is What You Get
10. Bibliography


