Text Based Content Protection
And
CAPTCHA Solution
On The Basis of Font Parsing

A Dissertation

Submitted to:

Central Department of Computer Science and Information Technology

Tribhuvan University,

Kirtipur, Nepal

In partial fulfillment of the Requirements for the

Master's Degree of Computer Science and Information Technology

Submitted By:

Rabindra Maharjan

Roll Number: 16/2008-2010

[CDCSIT TU, NEPAL]

Under the Supervision of

Prof. Dr. Subarna Shakya (TU)

(Co-Supervisor)

Bikash Balami

Lecturer (TU)

August, 2013

Tribhuvan University

Central Department of Computer Science

And

Information Technology

Kirtipur, Nepal

Date:

LETTER OF RECOMMENDATION

Mr. Rabindra Maharjan has carried out this dissertation work entitled "**Text Base Content Protection and CAPTCHA Solution On The Basis of Font Parsing.**" under my supervision and guidance. In my best knowledge, this is an original work in computer science. I, therefore recommend for further evaluation.

……..

Prof. Dr. Subarna Shakya

Institute of Engineering (TU).

Pulchowk, Nepal

(Supervisor)

Tribhuvan University

Central Department of Computer Science

And

Information Technology

We certify that we have read this dissertation and in our opinion it is satisfactory in the scope and quality as a dissertation in the partial fulfillment for the requirement of master's degree in computer science and information technology.

Evaluation Committee

Prof. Dr. Subarna Shakya	**Mr. Nawaraj Paudel**
IOE, Pulchowk (TU)	HOD, CDCSIT-TU, Nepal
(Supervisor)	
(Internal Examiner)	**(External Examiner)**

Acknowledgment

I am thankful to the Central Department of Computer Science and Information Technology (CDCSIT), TU for providing an opportunity to write this dissertation, which could enhance the capabilities of students in the field of research work, I extend my profound indebtedness to **Prof. Dr. Subarna Shakya**, IOE (TU), Pulchowk as my dissertation supervisor whose invaluable assistance, guidance and suggestions continually inspired me to complete this work.

I extend my sincere gratitude to lecture, **Mr. Bikash Balami** for his constructive suggestions, inspirations and encouragement throughout the work as well as continuous support throughout two years of Master's degree.

I am thankful to HOD **Mr. Nawaraj Paudel** and all other teachers who have enriched my knowledge and encouraged me, by providing valuable suggestions, in all my academic endeavors.

I express deep gratitude to my wife Mrs. Basanti Maharjan who have always encouraged and supported for this achievement. This small piece of work is affectionately dedicated to her.

I express my deep appreciation to my friends Mr. Nripendra Nath Newa and Mr. Suresh Thapa for their great help in this work.

Finally, I would like to thank all the staff and colleague of Center Department of Computer Science and Information Technology for their cooperation and well wish.

Rabindra Maharjan

Abstract

The production of information and communication technologies in the last few decades has strongly influenced the way information is gathered and processed, and hence raised concerns over privacy. With the advent of digital technologies, the ease of unauthorized copying of and access to digital content leading to commercial field has motivated the need for developing content protection technologies.

This study is in area of protecting the web content from bots which are deployed to extract information from web, using the web-font technology. Web is an easiest medium to share information and thoughts, but it is not very safe to leave information on web. One of the important problems of leaving information on web is that various bots might sniff off the information and use the information in malicious manner. For example leaving email address on web might result in spam emails.

In this study, I have researched about how fonts are created and how they work from various sources. I have studied w3c's CSS specification for @font-face and its cross browser compatibility. I have studied about current technologies being used in the area. It is hoped that this study will change the way the content in web is protected from bots.

This research addresses three things: Font Parsing, Content Protection and CAPTCHA Solutions and implement them with the reference to an example using C# programming language and SQL database.

List of Abbreviations

AI : Artificial Intelligence

API : Application Program Interface

ASCII : American Standard Code for Information Interchange

CAPTCHA : Completely Automated Public Turing test for telling Computer and Human Apart.

CMAP : Character Map

CPU : Central Processing Unit

CSS : Cascading Style Sheet.

DoS : Denial of Service

DRM : Digital Rights Management

HTML : Hyper Text Markup Language

IIS : Internet Information Service

JSON : JavaScript Object Notion.

LINQ : Language Integrated Query

MVC : Model View Controller

OCR : Optical Character Recognition.

PHP : PHP Hypertext Preprocessor

RBAC : Role Based Access Control

REST : Representational State Transfer

Sfnt : Spline Font

SSL : Secured Socket Layer

SVG : Scalable Vector Graphic

URL : Universal Resource Locator

XML : eXtensible Markup Language

Table of Contents

CHAPTER 1

CHAPTER 2

CHAPTER 3

CHAPTER 4

CHAPTER 5

RESULT AND DISCUSSION

CHAPTER 6

List of Figures

Pages

List of Tables

Pages

CHAPTER 1

INTRODUCTION

1.1 General Background

Internet has become an integral part in everyone's day to day life and social system. Virtually everything, not only limited to management, education and governance, are dependent on internet in today's world. Internet has become the means through which people share the information with each other.

Due to the widespread of internet use, security has been always key concern and its issue has been growing since last few years. The information providers via internet sites are concerned with the access of their information. As some information in wrong hand could do an irreparable damage to human and society as a whole.

Search engines sites such as Google, Yahoo!, and Bing etc. are crawling the internet so as to make the content on internet more accessible to everyone. Without these bots the internet would have been very much unusable. Finding information from the vastness of internet would have been rather impossible for a human being. Even with the current cutting edge technologies being used by these crawlers, they are not able to scale through whole internet. What we can see from these types of search engines is just a tip of an iceberg. There are people and organization that are crawling the sites in internet on a 24/7 hour basis. The crawlers from these big search engines do follow certain rules and regulation laid by the internet community while crawling those sites. They won't scrap content that are not meant for them to crawl. Not all crawlers are there for the good of the internet and internet users. Some of the crawlers are malicious in nature. There are crawlers which are crawling over the internet for harmful purposes, such as:

- email harvesting,
- content surveillance,
- Spam form posting etc.

Such bots poses a very grave threat to privacy and security of the author of the site as well as other stakeholders. The email harvesting crawler could collect emails and

simply send spam mails with unwanted advertisements or pornographic materials in a huge number. Such spams are generally un-maintainable and can cause a lot of trouble to the email address holder. Some crawlers might simply copy content of one site to another hampering the search engine ranking of the original site.

Content surveillance is also an equal threat now a day. A very simple unintended wrong word used in a site in a different context could get people in the eyes of the government agencies and get them into trouble without any reason.

In general, the information on web is subject to many types of attacks such as plagiarism, web content harvesting, screen scrapping for data like emails etc. A web site which allows users to post forms is also vulnerable to spam form posting. There are malicious bots that dedicatedly fill up forms and post thousands and millions of forms in few seconds.

Spamming a web server with this type of fake form posts can have severe consequences on a web server. Some of impact that spamming can have on server are listed below:

- Spamming can lead to bulging of database entries costing lots of disk spaces
- Spamming can also lead to the server being irresponsive and become unable to give response to other valid users (the DoS attack).

With the growth of web the concern for its security has also grown. Today there are technologies such as SSL specifically invented to protect against tapping information on transits. But SSL is mainly focused towards protecting information being posted to server, rather than information being transmitted to client. Bots are clients so SSL does little to protect the content from bots.

From very beginning the web had been envisioned to be an open platform for sharing knowledge and information for scientific community. Hence, it is not surprising that web lacks the infrastructure for protecting the content. In today's world the use of web has grown and its position has changed from a humble document delivery system, to various other applications. Some of the major areas where web is being used today are as follows:

- Web is still serving as an information sharing platform as it was intended, in its inception period.
- Web serves as an online shopping store where users can visit view various items available and buy the ones they like.
- Web today serves as an application delivery system. Applications like project management system, Management information system etc. all are available on web today.

This study aims to protect the content in web page from unwanted bots, while remaining friendly to humans. The study also intends to protect web sites by preventing the bots from spamming the forms through use of CAPTCHA.

The core concept of this study is to use a ciphering algorithm to encrypt desired content and then using a font to decrypt the encrypted text. For that purpose there is a previously created special font that will have correct letter glyph which replaces the encrypted characters. This makes the technology a visual obfuscation that can be decoded visually, while using a real font means the users will be able to resize the font to their visual need. Since, its purpose is to protect web sites form bots.

1.2 Content Protection Overview

As on-demand transmission of web data and content which is rising at a desired time to a desired place has become possible demand. Users are generally satisfied with the convenience and flexibility of such content distribution, and enjoy being able to accesses to the desired content simply and efficiently. As content owners have moved to address users demand for such content distribution they are naturally concerned about the illegal use and distribution of their intellectual property. Therefore it is necessary to balance the needs of content owners with the demand of content users.

Numerous technologies are available for content protection, including data encryption and digital watermarks. Different systems may be used for different mechanism and protection technologies to distribute content security. As mentioned in [24], there is the problem with existing encryption systems is that an entire document, graphical image, and so forth, is encrypted in its entirety, such that if a hacker were to determine the key or keys necessary to unscramble the code, a serious security breach could result. IT and security professionals are in a perpetual arms race with attackers.

Packet sniffers, firewalls, virus scanners, and spam filters are doing a good job securing the borders, but what about insider threats? According to Forrester, "trusted" insiders and business partners, malicious or not, are responsible for 43% of security breaches [1].

Users are concerned about the importance of security in digital world. So, users want to be sure and will be able to enforce copyrights to protect their creations before they introduce their intellectual property into digital markets.

Content protection is a concept in which users can digitize their rights management process depends on different issues, for example:

- How far the user has evolved within the right management spectrum?
- How much security will be required?
- What impact that the security will have on the user experience?
- How much its content is valued in the desire place?
- In what way the user expect to access the content?

The terms "content protection," "encryption," and "digital rights management" (DRM) are often used interchangeably, but refer to different solutions. Content protection is a general term that refers to software or hardware systems that safeguard content. These systems can employ encryption, format manipulation, or other types of "obfuscation" mechanisms for tamper resistance to protect the content. Content encryption uses a cipher to encode the content so that it is rendered useless unless unencrypted using the same or a related cipher. Encryption protects digital content from being useful if it is intercepted, stolen, or illegally copied. DRM on other hand defines and enforces the rules by which the content may be used. DRM more over serves as a set of rules which must be followed before one gets an access to intended system or content. [5]

One definition of content protection is "a means for protecting copyrighted content from unauthorized access that protects the content and its associated usage rights so that only legitimate users and devices can unlock and consume the content." [27]
The above definition of content protection has two main implications:

- Content cannot be accessed until the party that wants to access content proves it is legitimate.

- The party that wants to consume content can be user or device or even a software system.

Content protection is technology that enables digital content providers, to safeguard their online content from unauthorized use, copying, and dispersal. Content protection technologies enable you to identify users and to specify and enforce rights and restrictions for digital content, which enables you to securely distribute digital content. Such systems enforce your rules about how users can access and copy digital content files and may protect the content from being used by unintended users. [20]

Site scraping, alternately called data harvesting, screen scraping or Web scraping, is a technique used to extract data from a web page. Site scraping can be performed by manually copying website content from web browsers. However, given the number of overwhelmingly large number of sites present in the internet, the majority of site scraping attacks are performed by software programs that automatically extract data from websites. Site scraping tools range from simple, home grown scripts to commercial scraping software with embedded Web browsers that can parse HTML and store scraped data in databases. [11]

1.3 CAPTCHA Overview

CAPTCHA *stands for* **C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part [15], **Reverse Turing Test**, Human Interaction Proofs (HIPs) - protocols used online for distinguishing between humans and malicious computer programs [8].

People are currently dealing with serious vulnerabilities in computer security because of malicious entities such as viruses, worms, etc. CAPTCHAs are now almost standard security mechanisms for defending against undesirable and malicious automated scripts or bot programs on the Internet. However, its design is far from trivial since challenges designed are to be easily solved by humans, but too hard for computers to solve. Therefore, the correct response to a CAPTCHA challenge is assumed to come from a human and the users are permitted into the website.

Nowadays, a common kind of CAPTCHA used on most of the website requires the users to enter the string of characters that appear in a distorted form on the screen is called text-based CAPTCHA. In which the challenge appears as an image of distorted

text that the user must decipher and retype. These schemes typically exploit the difficulty for programs to recognize distorted text. A good CAPTCHA must not only be human friendly but also robust enough to resist computer programs that attackers write to automatically pass CAPTCHA tests. Test based on the difficulty of recognizing characters was both straightforward to implement and enjoyed a fairly high success rate.

Spammers are constantly trying to build algorithms that read the distorted text correctly. So strong CAPTCHAs have to be designed and built so that the efforts of the spammers are thwarted. As [25] CAPTCHA scheme should be without considering the age, knowledge, sex and the educational information of people. All selected features should be considered completely in order to make a powerful and usable CAPTCHA scheme. The CAPTCHA is adequate until it is broken, and then there are no further mechanisms in place to prevent the service from being abused.
Human interactive proofs (HIPs) are schemes which require some kind of interaction from a human user that is tough for a program to simulate. “Completely automated public Turing test to tell computers and humans apart” (CAPTCHAs) are a class of HIPs which are tests that are so designed that humans can easily pass them while automated programs have a very tough time in passing them. Thus, such tests try to prevent malicious automated programs from accessing Web services which are meant to be used by human users only. Differences in the capabilities between humans and computer programs, which can be tested and evaluated over the Internet, are made use of to create a CAPTCHA. Generally, hard “artificial intelligence” (AI) problems are turned into CAPTCHAs. Usually such tests utilize schemes which exploit the differences in the cognitive capabilities between humans and computers, for instance, exploiting the difference between humans and computer programs in understanding distorted text.

1.4. CAPTCHAS and Turing Test

At the beginnings of modern computer science , Allan Turing presented his theories on the possibility of thinking machines in his famous article “Computing Machinery and Intelligence” [15]. [A. Turing. "Computing machinery and intelligence". *Mind*, 49:433–460, 1950.

CAPTCHA technology has its foundation in an experiment called the **Turing Test**. Alan Turing, sometimes called the father of modern computing, proposed the test as a way to examine whether or not machines can think or appear to think like humans.
When a user human or machine chooses to take the test (*e.g.* in order to enter a protected Web site), a program challenges the user with one synthetically generated image of text; the user must type back the text correctly in order to enter. This differs from Turing's proposal in at least four ways:

- The judge is a machine, rather than human;
- There is only one user, rather than two;
- The design goal is to distinguish, rather than to fail to distinguish, between human and machine; and
- The test poses only one challenge or very few rather than an indefinitely long sequence of challenges.

The challenges must be substantially different almost every time, else they might be recorded exhaustively, answered off-line by humans, and then used to answer future challenges. Thus this research proposes that they be generated pseudo randomly from a potentially very large space of distinct challenges.

The goal with CAPTCHA is to create a test that humans can pass easily but machines can't. It's also important that the CAPTCHA application is able to present different CAPTCHAs to different users. If a visual CAPTCHA presented a static image that was the same for every user, it wouldn't take long before a spammer spotted the form, deciphered the letters, and programmed an application to type in the correct answer automatically.

CHAPTER 2

PROBLEM DEFINITION

2.1 Problem Definition

Currently most of the web content protection technologies focus on protecting content while being transmitted through the wire, or uses image as a means to deter the bots from understanding the data being transmitted. It is desirable for the content to be protected at the destinations.

With a lot of privacy and security issues being raised, in the current web scenario, it has become very important for the web publishers to protect the data from malicious bots. These bots are deployed for collecting data from various sites. These data may pose serious threats to people's privacy, for example emails, people's address etc. Thereby the owner can define access policies to his/her resources based on their properties. The policies are defined using a natural language policy editor and stored on server. They are evaluated each time the resource is accessed. This mechanism provides a suitable easy-to-use solution for highly dynamic social platforms and in this way replaces a selection of a predefined security options like used in the most of the modern applications.

This proposal is basically concern with two purposes: Content protection for CAPTCHA solution and Content protection mechanisms conventionally target for copyright protection, copy protection, and conditional access for personal web content. These are the basic security requirements for an end-to-end content protection system.

2.2 Objective

The objective for this study is as follows:

- To provide an appropriate information infrastructure to protect the web content from crawling and spamming bots
- To provide a combination of encrypted fonts and web management process for human friendly access to protected information
- To develop and implement a new text base CAPTCHA for bots security.

2.3 Literature Review And Related Works

The terms “content protection,” “encryption,” and “digital rights management” (DRM) are often used interchangeably, but refer to different solutions. Content protection is a general term that refers to software or hardware systems that safeguard content. These systems can employ encryption, format manipulation, or other types of “obfuscation” mechanisms for tamper resistance to protect the content. Content encryption uses a cipher to encode the content so that it is rendered useless unless unencrypted using the same or a related cipher. Encryption protects digital content from being useful if it is intercepted, stolen, or illegally copied. DRM on other hand defines and enforces the rules by which the content may be used. DRM more over serves as a set of rules which must be followed before one gets an access to intended system or content. [5]

One definition of content protection is “a means for protecting copyrighted content from unauthorized access that protects the content and its associated usage rights so that only legitimate users and devices can unlock and consume the content.” [27]

The above definition of content protection has two main implications:

- Content cannot be accessed until the party that wants to access content proves it is legitimate.
- The party that wants to consume content can be user or device or even a software system.

Content protection is technology that enables digital content providers, to safeguard their online content from unauthorized use, copying, and dispersal. Content protection technologies enable you to identify users and to specify and enforce rights and restrictions for digital content, which enables you to securely distribute digital content. Such systems enforce your rules about how users can access and copy digital content files and may protect the content from being used by unintended users. [20]

Site scraping, alternately called data harvesting, screen scraping or Web scraping, is a technique used to extract data from a web page. Site scraping can be performed by manually copying website content from web browsers. However, given the number of overwhelmingly large number of sites present in the internet, the majority of site

scraping attacks are performed by software programs that automatically extract data from websites. Site scraping tools range from simple, home grown scripts to commercial scraping software with embedded Web browsers that can parse HTML and store scraped data in databases. [11]

Site scraping can be used for legitimate purposes, such as search engine indexing or mashups like comparative shopping tools. Site scraping can also offer a way for individual users to quickly and automatically gather data, such as stock price information or daily temperatures, which can be used for personal research. While not blatantly illegal, such actions can burden sites with traffic while bypassing online ads (which is the major source of income for many such sites). Site scraping quickly crosses over from being innocuous to malicious when individuals or businesses reuse scraped data for financial gain, rather than for social benefit. The following are just a few of the use cases for site scraping: [11]

- Collecting email addresses from websites for spam email campaigns
- Harvesting user information from social network sites or user forums
- Detecting changes on competitors' websites
- Gathering product and pricing data to undercut rivals' prices for goods and services
- Plagiarizing content such as news articles, blog posts, medical information, financial research
- Republishing website listings, such as job board postings, real estate listings, and phone directories
- Repurpose scraped content for applications such as comparison shopping sites or reverse phone lookup tools
- "Auction sniping" or placing bids on online auction sites within minutes or seconds of the auction ending.

The Spam Act 2003 Australia refers to spam as "unsolicited commercial electronic messaging" [2]. Address-harvesting software and harvested-address lists are often used for legitimate purposes such as collecting data for research, marketing or maintaining websites; but they are also often used to create distribution lists for sending spam. The legislation bans the use of address-harvesting software and

harvested-address lists, for the purpose of sending spam. You should ensure that the use of such software and lists are for purposes other than for sending unsolicited commercial electronic messages. Lists generated manually (for example by reviewing websites) are not prohibited. However, if addressees have included a statement adjacent to their electronic address indicating the addressee does not wish to receive commercial messages, this must be respected [2].

There are limited research have been performed on "Text Base New Content Protection and CAPTCHA Solution". [29] proposed a framework for text-based CAPTCHA that was based on Devanagari script. i.e. the written form for several languages including Hindi, Sanskrit, Kashmiri, Bihari, Bhojpuri, Marati and Nepali. Devanagari CAPTCHA has been used to enhance the security of Devanagari script-based Indian language content based web applications. Users are typically involved in content generation and access of applications in Devanagari – script based Indian languages. Users already have the necessary keyboard emulation or are using custom keyboards that facilitate keyboard input in that script so that aspect is taken care of.

In general context the present invention provides a methodology to protect content that is rendered and formatted using patchable system calls. The present invention is useful for protection of content within HTML pages and e-mail and more generally for protection of enterprise data. Altering text displayed in a formatted page, including locating a buffer of memory locations containing contents of a formatted page, locating a text string between two markers within buffer. Replacing the text string with an alternate text strings, inserting special fill characters in unfilled memory locations between the markers. The present invention preferably operates by encrypting protected content at its source, such as on a server computer and only decrypting the content when writing it into a display buffer for video display. Thus an application handling the content is in fact handling encrypted content and any attempt to copy content from the application, such as by copying a file or by attaching content to an email, can only expose encrypted content. Protected content is exposed when being written to the display buffer for display and while in the display buffer it is protected using applications invention.

2.4. Existing Technology

Images have been used to protect the content from bots while making image readable to humans. Some other technique used for the purpose is to write information such that bots would require human's reasoning to process information. For example a person could convey email address as "**firstname.lastname at gmail dot com**". Now, a human being who knows that person's first name and last name will be able to deduce the email address, while bots cannot. But problem is that using images as medium to transfer text is not very optimal. Not all information can be conveyed in ways where logical reasoning would be required.

Many people who are concerned about their email addressed being found by bots deploy the above mentioned techniques to hide the email address from bots, which specially relies on regular expression pattern matching for mining emails from a web page.

A real world example of obfuscating email address from bots while making it readable to human is shown in figure 1. The author has used phrase **"(first initial + last name) {at} cs.brown.edu**" to denote his emails address. A normal human being who knows the author's name won't find much difficulties in deciphering the given email address.

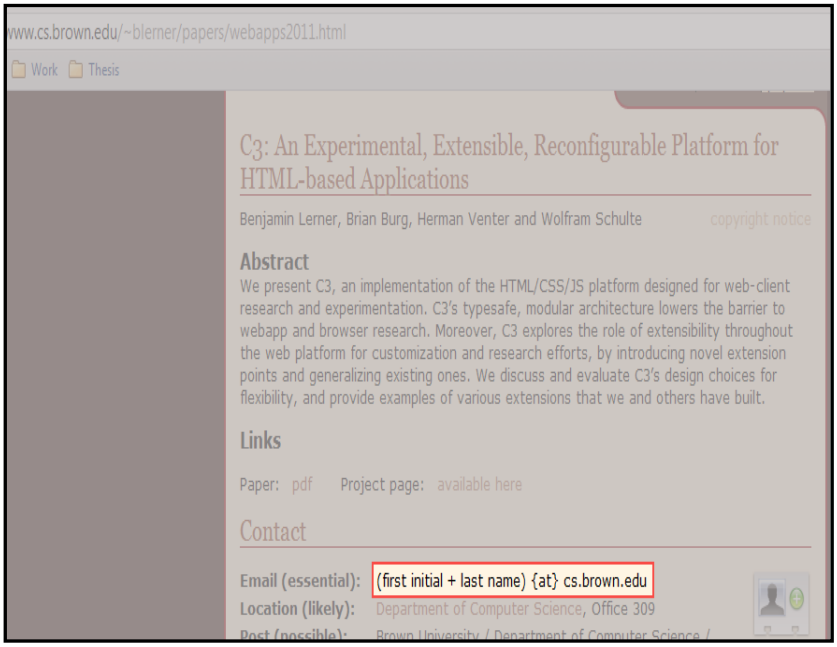


Figure 2.1: Email Address Format

Sources: http://www.cs.brow.edu/~blerner/papers/webapps2011.html

CHAPTER 3

METHODOLOGY

This dissertation aims to implement “content protection mechanisms target for copy protection” and “text base CAPTCHA solution”. The research methods are divided into two parts

3.1. Modeling Privacy-Friendly Content Protection Systems

The first part of the research focuses on providing a comprehensive framework to model content protection system and requirements. There are a number of fundamental questions to be explored, including conceptualizing privacy properties and threats, evaluating the relation between privacy and security properties, and designing a generic methodology to identify privacy threats and elicit privacy requirements in application-dependent systems.

The privacy properties and the derived threats are to cover a broader concept of privacy, such as data or content privacy. There is a need for privacy study to identify the privacy property or evaluate privacy threats in the targeted system. As a systematic approach to build in privacy is needed, it is worth the effort to investigate an exhaustive and comprehensive methodology to map privacy properties to privacy threats [19], to identify the threats, and to elicit privacy requirements.

The modeling of content protection systems is shown in fig 3.1. At the starting of system, first of all font parser section is created. It parses the font uploaded from font admin and extracts the information about cmap table. The extracted cmap table is used by character shuffling module to generate character map chart by shuffling cmap table. Furthermore description of cmap table is mentioned at the end of section 3.2.1

Admin section is responsible for role base management to control the user and font admin. Font Admin is use to upload the font which is suitable for CAPTCHA section and content protection section. User can only use font uploaded from font admin section. At last the end user application is created. This is the main output of the modeling also known as user portal. It contains CAPTCHA section and Content encryption section. CAPTCHA can be refreshing by CAPTCHA refresh module. Furthermore details of CAPTCHA and content protection is mention in section 3.2.5, 3.2.6 and 3.2.7

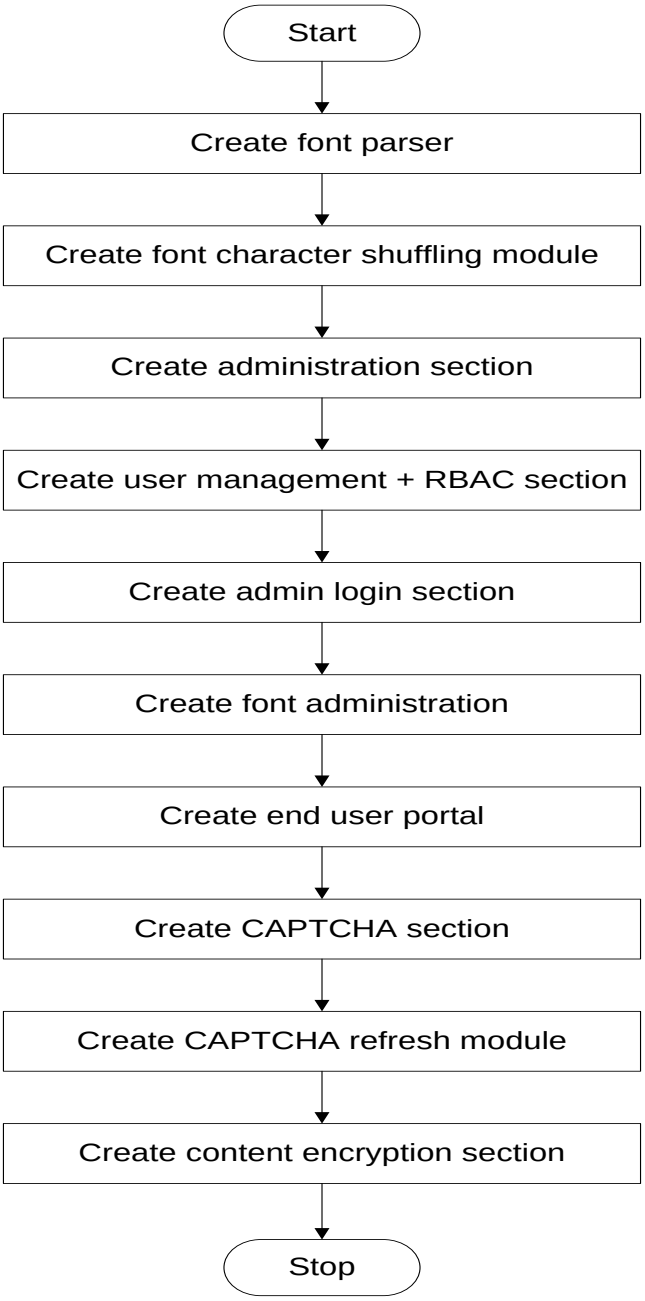


Figure 3.1: Modeling Privacy-Friendly Content Protection Systems

3.2. Designing Privacy-Friendly Content Protection Systems

The second part of the research focuses on exploring and designing individual privacy-friendly content protection systems with concepts: Privacy preserving systems to manage and protect personal data using content protection techniques.

3.2.1. Methods of content protection

There are following six possible methods to protect the content in this research. Among them all are used in designing content protection system except 'Displaying Images' and reason behind not using this is mention below.

3.2.1.1. End user authentication and authorization:

Content can be protected from unintended users and bots by hiding the content inside a protected area. Allowing user to login to know who is accessing the system and set the permission accordingly. Authentication is process of identifying the user's identity and authorization is the process of setting permissions to the content to which user gets access. [20]

3.2.1.2. Encryption:

Encryption can be used at various layers in the internet protocol suit to prevent the unwanted agents from gaining access to the content. [20] For example SSL is an encryption scheme used to protect the data that are being posted from client to server. SSL does rarely serve for protecting data from clients, but fulfills the objective of saving the posted data from being intercepted.

3.2.1.3. Obfuscation:

Obfuscation is lesser encryption. Obfuscation involves displaying the content in altered obvious form that is not understandable by unwanted parties, for example bots. [8]

3.2.1.4. Challenge Response:

An effective way to prevent bots from attacking a system is to present the user with challenge response. When any content is requested the server presents a challenge that would require human intelligence for example CAPTCHA, or some kind of logical or mathematical question that the user must answer. [4][26]

3.2.1.5. Displaying images:

The content is shown as an image. This method is very effective to protect against bots, it has some major disadvantages too. The band width usage compared to normal text is high for images, while the quality of image degrades when compressed to make the bandwidth acceptable. If the quality degrades up to the point where the text written in image is not clear for human then it is not very useful. [7]

3.2.1.6. Using JavaScript:

Using JavaScript to display content is yet another popular way of content protection. It is based on the assumption that most of the bots don't execute JavaScript before content harvesting. [7][28]

This research will present selected privacy preserving solutions in an attempt to resolve the contradicting forces between content protection (i.e., content providers trace and collect user information) and privacy protection (i.e., allow users to gain control over personal information), in order to enforce both the content provider's rights and the user's rights at the same time mapping of a number to character.

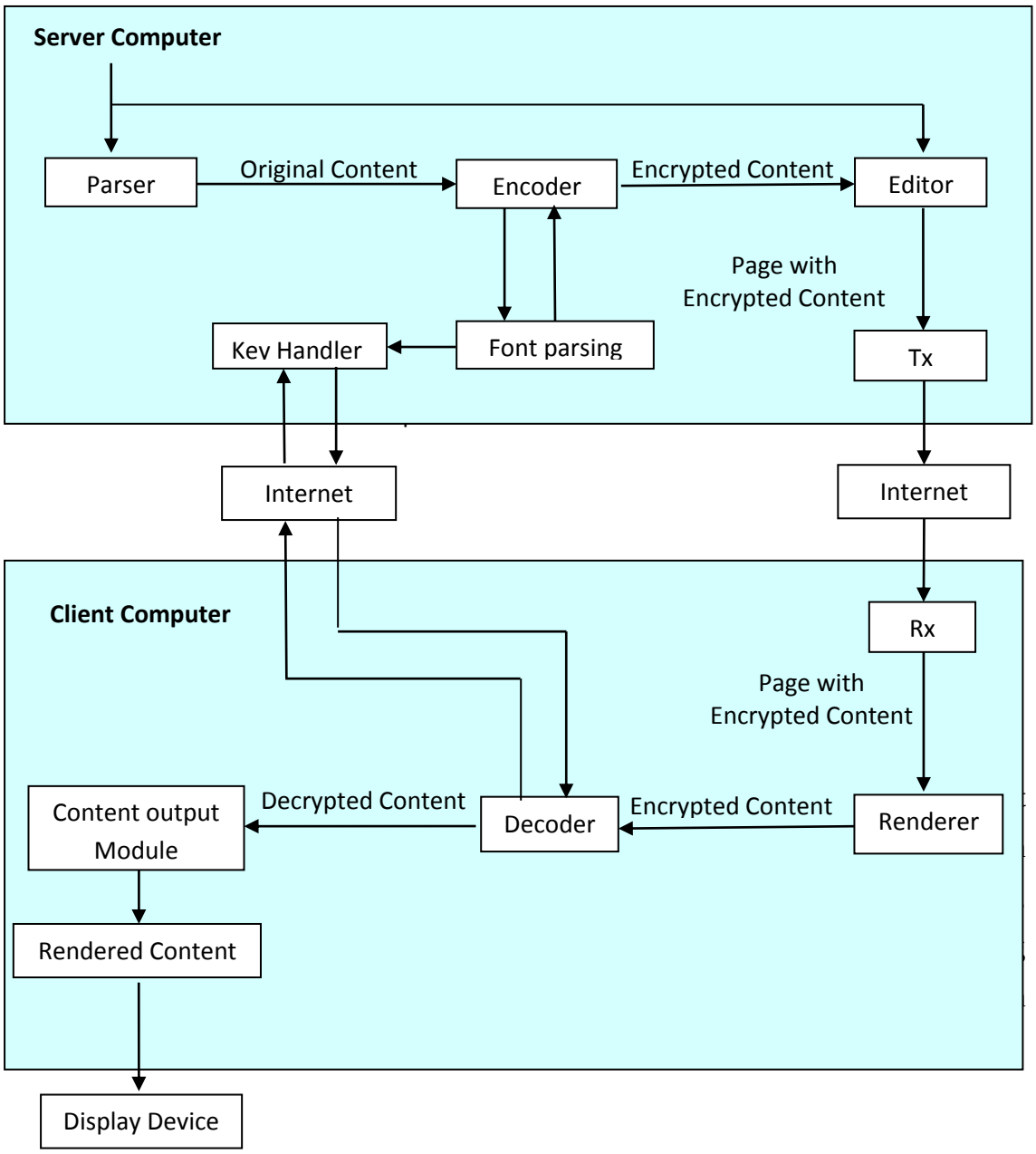


Figure 3.2: Framework Model of Encryption Technique.

This research uses two major existing technologies to achieve our purpose:

- Open-Type Fonts, as a medium to encrypt text for bots while making the text readable to human.
- Web-fonts, to transfer fonts on web without requiring the users to install the encrypted fonts locally.

This methodology parse open-type fonts from file into a well-defined data structure called ‘tables’, to be able to programmatically modify the fonts. The methodology modify the related tables such as ‘cmap’ table, the ‘head’ table etc. of Open-Type font to encrypt the font. The ‘cmap’ table consists of a mapping between a Unicode value and glyph. The Methodology shuffles the characters in ‘cmap’ table, so that a character points to different glyph. While shuffle the characters it also extract a mapping of original character to shuffled character in JSON format. The Methodology stores this JSON into database for future reference.

While encrypting content, it uses the JSON as a map to point out the encryption character for a plaintext character. The JSON that holds this map is generated during font encryption. It replaces original character of web content with the encrypted character pointed out by the JSON map.

When font is applied on that text encrypted using the JSON map, the font shows human readable characters visually, while the underlying text becomes a jumble of meaningless characters that does not make sense for a bot.
As shown in above figure 3.1, the methodology has followed following steps to achieve the desired results.

3.2.2. Parsing Algorithm:

A parser parses a page and identifies original content that is designated as protected. Such identified original content is transferred to an encoder that encrypts the original content into encrypted content. The encrypted content and the page are transferred to an editor that replaces the identified original content with the encrypted content, within the page. Transmitter then transmits the page with the encrypted content to client computer over internet.
Receiver within client computer receives the page with the encrypted content and transfers it to decoder and renderer for rendering the page into a graphics device. In the present invention, renderer identifies the encrypted content and transfers it to a

decoder that decodes the encrypted content prior to the content being passed to content output module. Content output module converts the decrypted content to graphics output, which is written into graphics device. Finally, apportion of data in graphics device, or all of the data in graphics device is displayed on display device connected to client computer.

3.2.3. Creating Font Parser:

A font file contains data, in table format. Rasterizers use combinations of data from the tables contained in the font to render the glyph outlines. In this research the methodology is dealing with Open type fonts with TrueType outline. An Open-Type font may also have postscript outline. The tables in the font files are supporting data for the rasterizers. Some of the supporting data is used no matter which outline format is used; some of the supporting data is specific to either TrueType or PostScript.

It is important to note that all Open-Type fonts big Endian, while most of the CPU use Intel-style byte ordering (Little Endian). So, while reading font data into memory the parser should reverse the order of bytes.

Data Type	*Description*
BYTE	8-bit unsigned integer.
CHAR	8-bit signed integer.
USHORT	16-bit unsigned integer.
SHORT	16-bit signed integer.
UINT24	24-bit unsigned integer.
ULONG	32-bit unsigned integer.
LONG	32-bit signed integer.
Fixed	32-bit signed fixed-point number (16.16)
FUNIT	Smallest measurable distance in the em space.
FWORD	16-bit signed integer (SHORT) that describes a quantity in FUnits.
UFWORD	16-bit unsigned integer (USHORT) that describes a quantity in FUnits.
F2DOT14	16-bit signed fixed number with the low 14 bits of fraction (2.14).
LONGDATETIME	Number of seconds since 12:00 midnight, Jan, 1904 signed 64-bit integer.
Tag	Array of four uint8s (length = 32 bits) used to identify a script, language system, feature, or baseline
GlyphID	Glyph index number, same as uint16(length = 16 bits)
Offset	Offset to a table, same as uint16 (length = 16 bits), NULL offset = 0x0000

Table 3.1: Data types used in a font file:

Tag	*Name*
cmap	Character to glyph mapping
head	Font header
hhea	Horizontal header
hmtx	Horizontal metrics
maxp	Maximum profile
name	Naming table
OS/2	OS/2 and Windows specific metrics
post	PostScript information

Table 3.2: Required Open-Type tables

A font file contains various tables. All Open-Type fonts start with an offset table, followed by table directory entries, again followed by the font data tables. There are many font data tables:

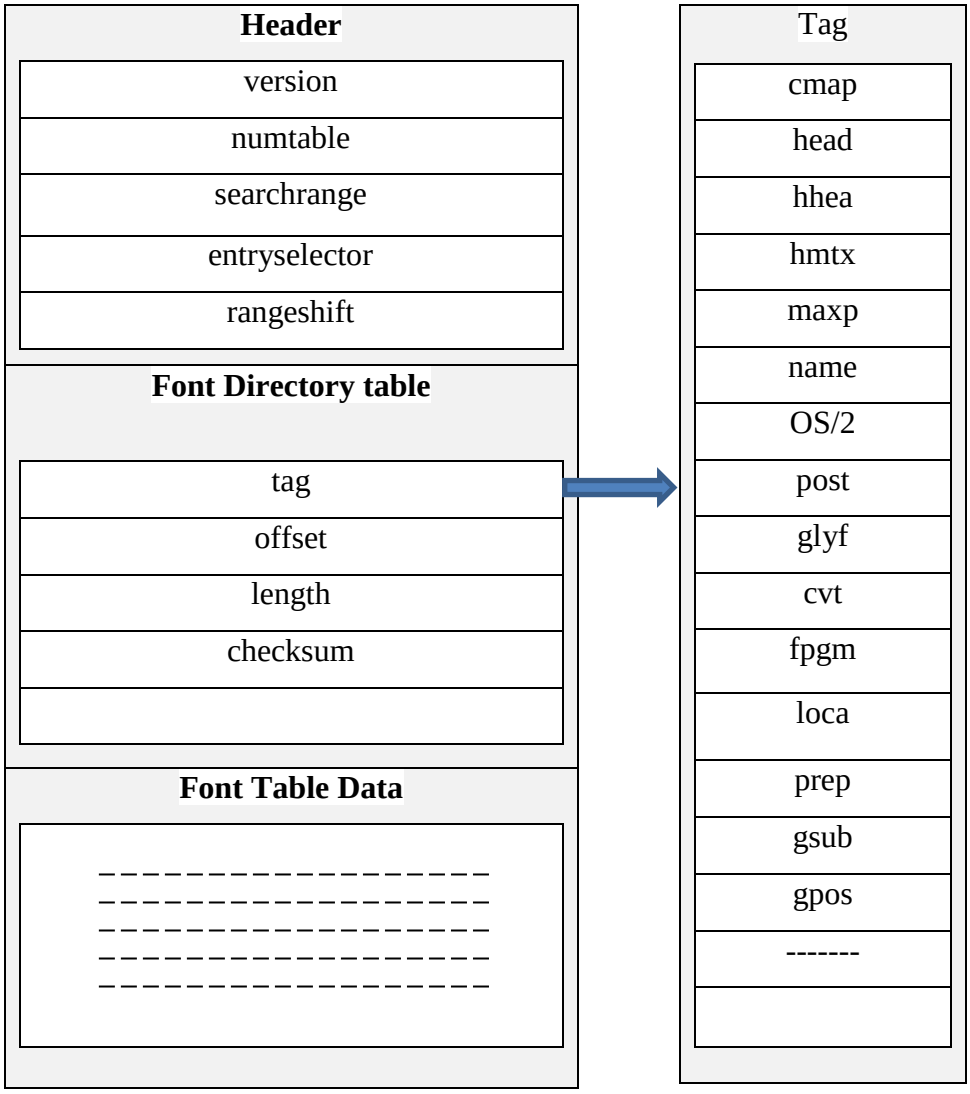


Table 3.3: Font File Structure.

Among all the above tables the tables grouped as ‘Required’ must always be present to form a valid Open-Type font. Without these tables the Open-Type fonts are not guaranteed to work properly.

For purpose of this research, methodology is always dealing with true-type fonts the tables related to post-script outlines can be safely left. For post- script outlines it needs more steps. All the true-type tables must be parsed, and all other remaining tables should be parsed if they are present in the font file.

Since the font data can be very huge in size (some might range up to 100 or even 500 MB in size), it use Memory mapped files to parse the fonts. The memory mapped files store data in virtual memory which is a section in hard disk that operating systems treat like the main memory. Using the memory mapped files help us to read only a small section of file into memory and convert that section into the table data structure. Thus allowing us to work with huge font file with very low memory foot print; which is very important as methodology must deal with very large collection of fonts and each font might range from few kb to few MB (typically 15 kb to 50 MB). The down side of memory mapped file is the fact that it is slower to access hard drive than the main memory.

3.2.4. Creating Font Character shuffling:

Character shuffling involves shuffling the characters in the ‘cmap’ table. Conceptually ‘cmap’ table might be thought of as a lookup table that maps each character to its corresponding glyph/glyphs.

Character	***Glyph index***
‘A’	1
‘B’	2
‘C’	3

Table 3.4: A Conceptual View Of 'Cmap' Table

The above table is just a conceptual view of a ‘cmap’ table. In reality the ‘cmap’ table is very complex data structure. The ‘cmap’ table can support multiple platforms, and multiple character encodings. The character encodings such as ASCII, Unicode etc

are the standard followed to represent the characters in computer understandable form. These encodings help in sharing data between various parties across the world.

3.2.5. Create CAPTCHA section:

The CAPTCHA module has five major components

3.2.5.1 CAPTCHA font collection:

The CAPTCHA font collection is a list of encrypted fonts generated from Normal font collection. One thousand different encryptions are generated from each font in the normal font collection. This is to avoid hackers from guessing the encryption schemes; if a particular font is always encrypted in same way there is a chance for hackers to guess a way to decrypt the CAPTCHA text.

3.2.5.2. CAPTCHA refresher:

The CAPTCHA refresher adds any new font added to normal font collection into the CAPTCHA font collection, this module also takes the oldest thousand fonts from CAPTCHA font collection and encrypts the font and refreshes Encryption chart. **Encryption chart** is a table which is generated at the time of shuffling the characters in font. It maps original character to shuffled characters hence helps encrypting a text.

3.2.5.3. CAPTCHA text generator:

This component is responsible for generation of a random text (also called CAPTCHA text.) and encrypting that text using a CAPTCHA font's **Encryption chart**. Along with CAPTCHA text the CAPTCHA text generator also generates a digital signature. The digital signature is consisting of CAPTCHA fontId, a message digest of the encrypted text + timestamp, and the timestamp itself. The message digest is generated using HMACSHA512 algorithm with a secret key. This module is a public API, and JavaScript client sends request to this module for showing CAPTCHA. Along with CAPTCHA text this module also sends back digital signature and CAPTCHA fontId to the client.

3.2.5.4. JavaScript client:

The JavaScript client is responsible for requesting the server for CAPTCHA text. The server responds this request with CAPTCHA text (the encrypted text), a digital signature and CAPTCHA fontId. The CAPTCHA fontId is used to create font's

URL for @font-face declaration. The encrypted CAPTCHA text is shown to the user and the digital signature is placed inside the form. The encrypted text becomes readable for user using the encrypted font. When user enters the CAPTCHA text the encrypted text, user input and the digital signature is passed to the server.

3.2.5.5. CAPTCHA validator:

This module is responsible for validating the user input. When a validation request is received this module looks for the CAPTCHA text, user input and the digital signature. If any one of them is missing then the CAPTCHA is not valid. If all are present then it proceeds to validate the digital signature. First the message digest and timestamp is extracted. The client supplied CAPTCHA text is combined with the timestamp and message digest is generated. The new message digest is compared with the user supplied message digest if two doesn't match then the CAPTCHA is invalid. If they match then proceed further. Using the timestamp check if the user input has been provided within a valid time, if not the CAPTCHA is invalid. If CAPTCHA was solved within the valid time then the user input is encrypted using the encryption chart. The user input is compared with the original CAPTCHA text; the user input is valid if two matches then.

3.2.6. Create CAPTCHA refresher module:

As discussed earlier the CAPTCHA refresher is a separate module scheduled in the operating systems task scheduler to run every day. This module is responsible to move any new unencrypted normal font into the CAPTCHA font collection. Each font is encrypted into 1000 different combinations. This module is also responsible for taking 1000 oldest CAPTCHA fonts and refreshing their encryptions.

3.2.7. Create content encryption section

The content encryption section consists of two different subsections:

3.2.7.1. Hosted encryption:

This module allows user to select a font and type text they want to encrypt. When the text is saved, this module creates a clone of the selected font shuffles the 'cmap' table of the cloned font and stores the font in database. After shuffling the font, this module encrypts the text typed by user and saves it into the database. Once all this is done, the module displays user with a JavaScript code and iframe

code. Any one of this code can be placed by user into their site so that a section of their page gets protected.

3.2.7.2. Self-hosted encryption:

This module allows user to select fonts, and a server side programming language. The user can download a kit after selecting the fonts. A kit is a folder containing related CSS, fonts and a programming language specific API. When user selects fonts and server side programming language, this module shuffles the font, writes the **Encryption chart** into a file, writes the 'CSS' with @font-face declaration into file. All the font, Encryption table, 'CSS' and the programming language specific API are combined into a zip file and downloaded into the user's machine. This kit must be placed into user's server, and user can call programming language specific API to encrypt their content.

CHAPTER 4

IMPLEMENTATION

4.1. Tools And Technologies

Above methodologies has been implemented using Asp.net MVC framework with c# as the programming language. The SQL-server 2008 is the database server. This implementation works with IIS7 server for hosting the web-site, and uses Window's task scheduler to execute repetitive tasks on defined schedules.

Technology:

1. Asp.net MVC v3.
    - **a.** C# v4
    - **b.** HTML
    - **c.** JavaScript
    - **d.** CSS 2
2. Asp.net Membership provider
3. Fluent Validation as the validation framework
4. NInject Dependency Injection framework
5. PetaPOCO ORM
6. SQL-server 2008
7. Internet Information Server (IIS) v7

Tools:

1. Visual studio 2010.
2. SQL Management Studio 2008.
3. IIS7 Manager (InetMgr).
4. Photoshop CS3

4.2. Web architecture:

- The MVC architecture

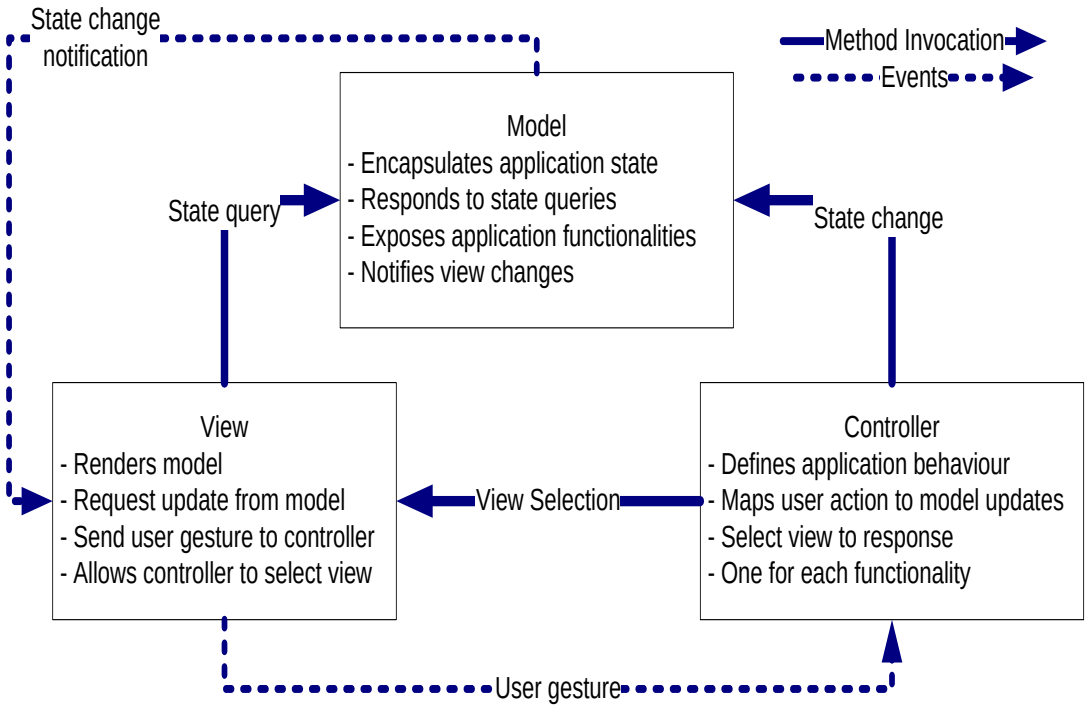


Figure 4.1: MVC architecture

The basic MVC architecture is divided into three major components:

1. Model:

The Model is responsible for persisting the application's state in a data store. The data store can be anything ranging from database, memory, file system etc. The model encapsulates the Data access layer and provides an interface for the business logic layer.

2. View:

The View is responsible for rendering the model in the format which the requesting client understands. For example, the view can render the Model as XML, HTML, and JSON etc.

3. Controller:

The controller is responsible for defining the application's behavior. It responds the client's request by interacting with the Model, and presenting the result to the client in form of view. It works like the glue between the Model and the View.

4.2.1. N-Tire architecture:

The Model-View-Controller architecture can be roughly mapped into 3 tire architecture.

1. Business logic layer:

The Model in MVC works as the business logic layer. This layer is responsible for implementing any business rules defined by the system's owner.

The architecture uses Active Record pattern for maintaining the business logic into the Model while encapsulating the Data access layer underneath the Model.

2. Data access layer:

The Model encapsulates the Data access layer. It is a layer which maps the model into the underlying data model. The business model may be fairly different from the data model. The Model in MVC removes the concern of transforming the business model (the 'M' in MVC) into a data model which can be persisted in the data store (database, file, memory etc).

The architecture uses PetaPOCO as the Data access layer. PetaPoco is responsible of transforming the Plain Old CLR Objects (POCO) into the database understandable Insert/Update queries for storing. While it is also responsible for transforming the result sets generated by SELECT queries into POCO objects understandable by the Model.

3. User Interface layer:

The Controller and View in MVC works together to form the User Interface layer. The Controller is responsible for retrieving the data from Model, while Model checks for various business constraints. The controller passes over the extracted model into the View. The controller decides what should the client get depending upon the client. For example the controller may decide to render the same Model as HTML, XML, JSON, text file, image etc depending upon what the client is requesting. This architecture uses Microsoft's Razor view engine, and Razor syntax for representing the Model as HTML. The XDocument (LINQ to XML) is used to render XML, and Microsoft's JSON serializer is used render the Model as the JSON object.

4.2.2. The client and the server

As with any web based application the browsers work as the client for this architecture. Since it is using Asp.net MVC framework, the server may range from IIS (Internet Information System) server, Cassini server that comes along with Visual studio. The business logic lies in the web server. With Mono framework the Apache web server also can be used to host this service. MSSQL server 2008 is the database server used in this architecture.

4.3. The application architecture

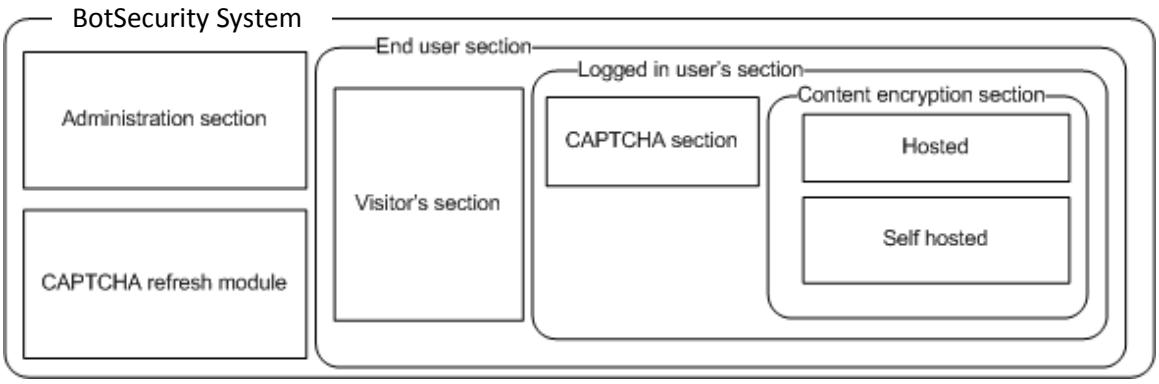


Figure 4.2: Application Architecture

The BotSecurity application consists of three major sections:

4.3.1. The administration section

The administration section is implemented as a different 'Area' of asp.net MVC3 framework. The application allows the administrators to manage fonts, users, CAPTCHA texts, Language, page contents etc.

4.3.2. The end user section

The end user section is the main asp.net MVC application. This is the section where normal user can visit the site and interact with the site. The end user section is again consisting of two different sections:

a. The visitor's section

A normal visitor as soon as visits the site, this is the section that the visitor reaches. This section is consisting of pages any normal website would have e.g. home page, about us, sign in and sign up pages etc.

b. The logged in user section

A visitor can register and then login to the BotSecurity system to use the service provided by BotSecurity application. The logged in user will again have two major sections:

b.1. CAPTCHA section

The CAPTCHA helps to protect a form from bots. The bot has to enter valid CAPTCHA text before being able to save the form in database, hence protecting the site's owner from bots spamming unnecessary content into form.

The CAPTCHA section allows the user to download the CAPTCHA related modules, and provides documentation on how CAPTCHA is to be integrated into his/her site.

Following server side technologies are supported:

- Asp.net web-form
- Asp.net MVC
- PHP

b.2. Content encryption section

Content encryption helps user to encrypt a section of webpage such that only human can read that section, while bots get obfuscated data.

This section allows the user to download the Content encryption related modules. These modules help the user to integrate the BotSecurity encryption service into their site.

Content encryption section module supports following methods for integration with site:

b.2.1. Hosted encryption

In this method the encrypted text is hosted by the BotSecurity system, and allows the user to integrate into their site through JavaScript or IFRAME.

b.2.2. Self hosted encryption

In this method, the user is provided with server side modules for the users to generate the encrypted content on the fly. The server side technologies supported are:

- ASP.net web-form
- Asp.net MVC
- PHP

4.3.3. CAPTCHA refresh module

The basic infrastructure for the CAPTCHA module is the collection of encrypted fonts. To protect bots from guessing the encryption scheme the BotSecurity application refreshes the CAPTCHA encryption fonts on a periodic basis. The CAPTCHA refresh module, is implemented as a windows application that will get initiated by windows task scheduler on a scheduled times.

CHAPTER 5

RESULT AND DISCUSSION

An important aspect of the present invention is that without the intervention of decoder, the page being rendered into graphics device contains encrypted content. Any other application that captures data from the page will only capture the encrypted content, which is typically, appears as garbage. Thus the original content designated as protected is not exposed to other applications.

In general terms the present invention preferably operates by encrypting protected content at its source, such as on a server computer, and only decrypting the content when writing it into a display buffer for video display. Thus an application handling the content is in fact handling encrypted content. And any attempt to copy content from the application, such as by copying a file or by attaching content to an email, can only expose encrypted content. So the output of this research is to implement text content protection on web and implement new CAPTCHA solution on the basis of text content protection using font parsing is as follows.

5.1. Result of Content Protection in web

As the internet grows into our daily lives and removes human to human interaction by considerable leaps and bounds, the necessity to identify whether the entity on the other side of internet is really a human being or an intelligent program has gained immense importance. Many ecommerce businesses which cater to such a growing population of human users on the web have business models in which the primary assumption is that humans are the users of the service. Automated programs are increasing able to perform many tasks on the web just like a human user. In many cases, these automated bots are to be denied access to the service. In all such scenarios content protections or CAPTCHA solution play the role of the guard which keeps the bots away from accessing the following services.

- Preventing spammers from online polls.
- Preventing spammers from getting free mail IDs
- Preventing chat bots from irritating people in chat rooms with advertisements
- Preventing automated dictionary attacks in password systems.

- Preventing unruly search engine bots from indexing sites.
- Preventing unethical pricing practices in ecommerce.
- Preventing inflating/deflating rankings in online recommender systems.
- Preventing spam in blog comments.
- Preventing game bots from playing online games.
- Preventing DDoS attacks.
- Preventing automated worm propagation

While these were some of the current reasons for the deployment of content protection and CAPTCHA solution. s ecommerce grows and as the Internet replaces human to human interaction, new scenarios requiring CAPTCHAs will emerge.

Here is an example for Content Protection output.

Plain Text: The quick brown fox jumps over the lazy dog.

Cipher Text: t« è·Ç\□ '□f úÂ□ 2dÒ'P ‹f·%° ;·r ¥d∑œþ ·T□ªÂ

5.2. Result of CAPTCHA Properties

The degree to which various CAPTCHA properties are presented and how they interact in a CAPTCHA scheme largely impact the scheme's overall usability and security. This section discusses several CAPTCHA properties and their known facts on both usability and security.

Here is an example for CAPTCHA Solution output.

Figure 5.1: CAPTCHA Output

5.2.1. Character Set

A larger character set leads to a more secure CAPTCHA scheme. A larger number of possibilities decrease the odds that any given guess from an automated process is correct.

However, this same concept negatively impacts usability. The increased security gained from using a large character set is negligible when compared to the decreased accuracy by human solvers. Studies have shown human solve rates as high as 98% in CAPTCHA images which use only numerical values (0-9) as the character set. This drops to 82% when using an alphanumeric and case sensitive character set (a-z, A-Z, 0-9) [29]. It can be assumed that this human accuracy would be further decreased by including non-alphanumeric symbols in the character set. A large character set also leads to more characters, such as lowercase i and j, looking similar after distortion and blurring are applied [6]. This is known to create several usability issues in various widely used CAPTCHA schemes.

5.2.2 String Text

A short string length combined with a small character set can lead to a situation where random guesses can solve a CAPTCHA with a success rate higher than 0.01% [8], the industry standard for determining whether or not a CAPTCHA is secure. For this reason, having a sufficiently long string length is important. String length has opposite implications in terms of usability depending on whether or not the string text contains a dictionary word or a sequence of random characters. In the case of a dictionary word, increasing the string length directly correlates to an increase in user accuracy. However, there is negative correlation between string length and user accuracy when applied to a CAPTCHA scheme using a string of randomly generated characters. This has been attributed to the fact that human solvers are able to use surrounding characters to infer indistinguishable characters when working with a dictionary word. This obviously is not possible when working with a string of random characters. While using a dictionary word clearly increases usability, it has a negative impact on security which is similar to using a smaller character set as there are fewer possible solutions compared to strings composed of random characters.

Another interesting aspect of the string text is the difference between a fixed-length and a variable-length scheme. Fixed-length string text has negative security

implications [15]. If the number of characters in a given CAPTCHA image is known, the segmentation process of an automated attack can make assumptions allowing it to locate individual characters with a higher success rate

5.2.3 Distortion

Distortion is often given credit for being the most effective CAPTCHA property to disrupt classification by the OCR component of an automated attack. However, it can also heavily decrease the accuracy for a human solver if implemented excessively. Given that distortion alone is not able to completely resist classification, the usability issues caused by distortion are often more significant than the increased security .

5.2.4 Font

A CAPTCHA scheme that makes use of multiple fonts is more resistant to automated attacks in that OCR software will have a decreased success rate. This is because varying fonts makes character size unpredictable. The use of multiple fonts has not been shown to negatively impact usability.

5.2.5 Complex Background

The goal of a complex background is to increase security by hiding the foreground text within the background making the CAPTCHA image more difficult for an automated process to segment. In recent years complex backgrounds have been popular despite the fact it has been shown that they generally add very little to the overall security of a CAPTCHA scheme. A common tactic is to use foreground and background colors that differ very little in their values on the RGB scale (a pixel represented in terms of its red, green, and blue values), yet are easily distinguishable by human vision. However, studies have shown this can be easily countered by converting color from the RGB scale to cylindrical coordinate color models that more accurately reflect human vision [29]. It has been determined that using very similar colors or an overly-complicated color scheme can create usability issues without necessarily increasing the overall security.

5.2.6 Stray Lines

There are three main types of stray lines which have varying impacts on segmentation. The first is small lines that do not cross characters. These have no proven security benefits as they can generally be removed in the preprocessing phase of an automated attack. The second type is small lines that connect distinct characters.

These provide more of a security increase than the first type, but are not entirely segmentation resistant. A histogram attack can yield high segmentation success rates against small connecting lines. The final type of stray lines is large lines that have the same thickness as characters in the CAPTCHA image. These have been proven to be an effective anti-segmentation measure in that it is very difficult for automated processes to differentiate them from characters. Large lines must be implemented appropriately, as in matching the foreground color and staying within the CAPTCHA foreground, or they will be more easily detected by a segmentation algorithm. Using the correct type of stray lines will significantly increase a CAPTCHA scheme's security through segmentation resistance without negatively impacting its usability.

5.2.7 Collapsing

Collapsing is often given credit for being the CAPTCHA property that most directly correlates to segmentation resistance. However, predictable collapsing, which uses a fixed string length and character width. The Blizzard CAPTCHA makes use of a complex background in an effort to hide the text from automated processes while keeping the text recognizable to a human solver. The Yahoo CAPTCHA makes use of thin connecting lines as the main resistance to automated segmentation attacks.

Google relies on collapsing as its main segmentation defense property in its CAPTCHA unacceptable success rates for automated segmentation processes. Predictable collapsing is a factor when the string length and approximate character width are known values. A segmentation algorithm can make high-probability guesses on where to segment, which leads to moderate success rates. This is why it is much more secure to use collapsing with variable-length string text and multiple fonts. While collapsing generally does not create usability issues, studies have proven that extreme collapsing (more than a 5-pixel character shift) can drastically reduce the accuracy of human solvers.

CHAPTER 6

CONCLUSION AND FURTHER STUDY

It is important to notice that to handle spam efficiently, multiple techniques are needed to be applied. In this research, the modular framework is capable to recognize between human and bots. i.e. Turing test have been made to detect bots in more effective way.

We have seen that by distinguishing between humans and computers, CAPTCHAs offer protection against automated attacks on systems and applications. The criterion for success of a text-based CAPTCHA is its robustness, usability, language independency and scalable for bot protection. We can use different fonts with different languages required and also can use custom CAPTCHA text, so that the length of CAPTCHA text is scalable and language independency

The implementation was envisioned to be an application to be implemented as SaaS (Software as a Service). After completion of above implementation, the application now can be hosted on any server supporting Asp.net, and can be used as a service. . The Content protection system has shown to be reliable when testing multiple times.

There are many possibilities for enhancement in future.

- Support for people with visual disabilities
- Support for other types for fonts that have wide use e.g. open type fonts with PostScript outline, SVG fonts etc.
- Support many other server side technologies, like python, java, Ruby on rails, Sinatra, OpenRasta, Fubu MVC etc.
- Since this service requires a lot of resource in terms of database space, hard disk space, memory and CPU cycles, moving this solution into cloud could yield a better result.

REFERENCE

[1]. **Andras Cser**, Forrester Research, Inc. "*Your Data Protection Strategy Will Fail Without Strong Identity Context*." June 27, 2011.

[2]. **Australian Government.** *Spam Act 2003: A practical guide for business.* 2004.

[3]. **BosaNova.** *Encryption methods for protecting data.* 2008.

[4]. **Chin, Jianying Zhou and Wee-Yung.** *An Effective Multi-Layered Defense Framework Against Spam.* 2008.

[5]. **Dell.** *CONTENT PROTECTION OF ONLINE MUSIC SERVICES.* 2003.

[6]. **G. Antoniou, M. Baldoni, P. A. Bonatti, W. Nejdl, and D. Olmedilla**, Rule-based policy specification. *In Secure Data Management in Decentralized Systems, volume 33 of Advances in Information Security. 2007.*

[7]. **Gils, Roel Van.** *Graceful E-Mail Obfuscation. www.alistapart.com. [Online] http://www.alistapart.com/articles/gracefulemailobfuscation/.* November 6, 2007.

[8]. **H. Baird and K. Popat**. Human interactive proofs and document image analysis. *Proc.IAPR 2002 Workshop on Document Analysis Systems*, August 2002.

[9]. **Hersch, Roger.** *Visual and technical aspects of type.*

[10]. How to protect web pages from email harvesting. *www.maxi-pedia.com.* [Online] November 20, 2010. http://www.maxi-pedia.com/how+to+protect+web+pages+from+email+harvesting.

[11]. **Imperva.** *Detecting and Blocking Site Scraping Attacks.* 2011.

[12]. **J. De Coi, P. Kaerger, D. Olmedilla, and S. Zerr**. Using natural language policies for privacy control in social platforms. *In Workshop on Trust and Privacy on the Social and Semantic Web (SPOT),* 2009.

[13]. **Jones, M.**, "JSON Web Key (JWK)," "JSON Web Algorithms (JWA)" December 2011

[14]. **Kaur, Clark Pope and Khushpreet.** *Is It Human or Computer? Defending E-Commerce with Captchas.* s.l. : IEEE Computer Society, 2005.

[15]. **L. Ahn, M. Blum and J. Langford**. Telling Humans and Computers Apart Automatically. *Communications of the ACM*, 47(2):57–60, 2004.

[16]. **Luisvon Ahn, Benjamin Maurer, Colin McMillen, David Abraham, Manuel Blum.** *reCAPTCHA: Human-Based Character Recognition via Web Security Measures.* s.l. : www.sciencemag.org, 2008.

[17]. **LuisvonAhn, ManuelBlum, NicholasJ.Hopper, and JohnLangford.** *CAPTCHA: Using Hard AI Problems For Security.*

[18]. **Luisvon Ahn, Manuel Blum, and John Langford.** *TELLING HUMANS AND COMPUTERS APART AUTOMATICALLY.* s.l. : COMMUNICATIONS OF THE ACM, 2004.

[19]. **M. Arnold, S. D. Wolthusen, and M. Schmucker**, Techniques and Applications of Digital Watermarking and Content Protection. Norwood, MA: Artech House Publishers, 2003.v

[20]. **Microsoft.** *Content protection in Silverlight.* s.l. : Microsoft, 2010.

[21]. **Microsoft.** The OpenType Font File. *www.microsoft.com.* [Online] http://www.microsoft.com/typography/otspec/otff.htm. May 5, 2008.

[22]. **Monotype Imaging.** A unicode font solution. *www.monotypefonts.com.* [Online] November 20, 2011. http://www.monotypefonts.com/Library/Unicode.asp?show=ucfontsol.

[23]. **Moshe Rubin, Daniel Schreiber**, Method and System for Copy Protection of Displayed Data Content, 2006.

[24]. **Patrick Zuili, Boca Raton**, Font Level Encryption Method and Apparatus, 2004.

[25]. **Rezvan Pakadel, Norafida Ithnin and Mohammad Hashemi.** "CAPTCHA, A survey of usability Features". Research General of Information Technology 3(4):215-228, 2011.

[26]. **Roman, Rodrigo.** *A Secure and Functional Anti-Spam Mechanism.* 2009.

[27]. **SEOW, Isa.** *Introduction to Content Protection and Potential Government Roles to Support its Deployment in Asia Pacific.* s.l. : Centre for Content Protection, 2011.

[28]. **Sethi, Amit.** *Digital Right managment and code Obfuscation.* 2003.

[29]. **Sushma Yalamanchiliand Kameswara Rao**, A Framework For Devanagari Script-Based CAPTCHA. 2010.

[30]. **Young, Anne L.** *Mathematical ciphers from caesar to rsa.* 2006

.

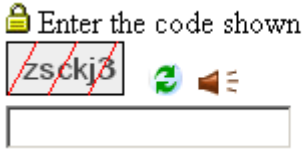
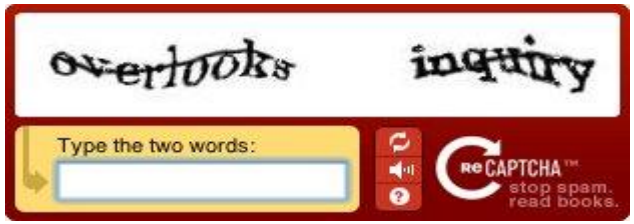
Appendix 1

Comparison of existing CAPTCHAs

Table A.1: Existing CAPTCHA Services Comparison Chart

Description	*Screenshot*	*Distinctive features*	*Audio*	*SSL*
AdCaptcher enables you to build and monetize custom captchas.		A new monetization method for your websites! Earn revenue by serving smart ads in your CAPTCHA.	**No**	
ADSCAPTCHA™ offers complete multi-language CAPTCHA solutions, from free security-only CAPTCHAs, to premium CAPTCHAs featuring images and video.		ADSCAPTCHA's™ innovative Pay Per Type™ platform creates profits from CAPTCHA technology. Instead of paying for CAPTCHAs, website owners and developers can make money from every ADSCAPTCHA™ that is typed in full.	**Yes**	

Adyoulike enables a new way of making money using Captchas. Their technology allows you to change the traditional CAPTCHA, difficult to read by an advertising CAPTCHA, nice, easy to read and remunerative.		The CAPTCHA is served as an Ad, so you can get money while securing your forms.	**No**
Are you a human is an alternative to CAPTCHA by using games : humans play but not bots...		Fun way to check that the visitor is an human : he plays a game	**No**
Captch Me			**No**

Captchas.net servers provide CAPTCHA images and audio files you can use in html-forms to prevent robots from filling it.			**Yes**	**Yes**
NuCaptcha provides the only intelligent CAPTCHA product on the market today		Behavioral Analysis System that adjusts security on the fly- brand able, SLA, advertising, fully customizable.	**Yes**	**Yes**
OpenCaptcha		Does not require any registration	**No**	**No**
ProtectWebForm			**Yes**	
reCAPTCHA is a free service that helps to digitize books, newspapers and old time radio shows while		The service helps to digitize books or audio shows	**Yes**	**Yes**

preventing automatic form filling by robots.				
Solve Media			**Yes**	**Yes**
WebSpamProtect.com allows you to add a verification image to your web site forms to protect them against spam robots. The service also provides full contact form management.		Online control panel to customize available CAPTCHA, or to create full contact forms.	**Yes**	**Yes** (premium accounts only)

Appendix 2

Test Cases

Home page:

- Given that user is not logged in. When loading home page user should see CAPTCHA sample and login box.
- Given that user is logged in. When loading homepage user should see CAPTCHA sample and no login box. The user should see his/her email address, “my account” link and “Log-off” link at top of the page.

Home page CAPTCHA:

- **CAPTCHA slider:**
  - When user drags the head of slider from left to right. Then the CAPTCHA text should become larger.
  - When user drags the head of slider from right to left. Then the CAPTCHA text should become smaller.
- **CAPTCHA refresh button:**
  - When user clicks on refresh button. Then the CAPTCHA text should change and the font of CAPTCHA text should also change randomly.
- **CAPTCHA verify button:**
  - Given that user has not entered the CAPTCHA text into the response text box. When user clicks the verify button. Then a notification should be shown in area between the CAPTCHA box and login box saying “Please try again”
  - Given that user has entered a wrong response in the response text box. When user clicks the verify button. Then a notification should be shown in area between the CAPTCHA box and login box saying “Please try again”.
  - Given that user had entered the CAPTCHA text into the response text box. When user clicks the verify button. Then a notification should be shown in area between CAPTCHA box and login box saying “You are Human” and Login box should get highlighted.

- **CAPTCHA Variations, Style dropdown:**
  - When user selects scroll left from style dropdown. Then the CAPTCHA text should move from right to left. After text disappears into the left boundary the process should repeat again.
  - When user selects scroll right from style dropdown. Then the CAPTCHA text should move from left to right. After text disappears into the right boundary the process should repeat again.
  - When user selects bounce horizontal from style dropdown. Then the CAPTCHA text should move from right to left. When the text reaches the left boundary of the box it should move from left to right and vice versa.
  - When user selects 3-D from the style dropdown. Then the CAPTCHA text should be shown in 3D.
  - When user selects "style" from the style dropdown. Then the CAPTCHA all the previously style must get reset.
- **CAPTCHA Variations, Font dropdown:**
  - When user selects any font name from the dropdown list. Then the CAPTCHA text should be applied the selected font.
  - When user selects "Fonts" from the dropdown list. Then the CAPTCHA text should get refreshed with a random font.
- **CAPTCHA Variation, Background color dropdown:**
  - When user selects any color name from the dropdown list. Then the background color of CAPTCHA text should change to that color.
  - When user selects "background" from the dropdown list. Then the background color of CAPTCHA text should get reverted to the original color.
- **CAPTCHA Variation combination:**
  - When user selects a style and then selects a font. Then the CAPTCHA text should apply that style be shown in selected font.
  - When user selects a style and then selects a background. Then the CAPTCHA should apply that style and shown with the selected background color.
  - When user selects a font and then selects a background color. Then the CAPTCHA text should be shown in that font and with selected background color.

- **CAPTCHA "hover to find what bots see" link:**
  - When user moves mouse pointer over the "hover to find what bots see". Then the link text should disappear and a random text should appear in its place.
  - Given that user is hovering mouse pointer over the "hover to find what bots see" link. When user moves mouse pointer out of the "hover to find what bots see". Then the link text should reappear and a random text should disappear.
  - Given that user has hovered over the "hover to find what bots see" and noted the text that is revealed. When user selects the CAPTCHA text, copies the text (Ctrl + c) and pastes the text in the text box. Then the user should see same random text that was seen when hovering over the link.

Home page Login Box:

- Given that a user has entered a text into "E-mail Address" textbox and the text is not a valid email address. When user clicks anywhere outside the "E-mail Address" textbox. Then the textbox should become red and user should see a message "E-mail Address is required" below the email address box.
- Given that a user has entered a valid email address but not a password. When user clicks Login button. Then the user should see message "Password is required" below the password box.
- Given that a user has entered a valid email address into "E-mail Address" textbox and a password into the password box and further given that the email address is not registered into the site. When user clicks login button. Then the user should be shown a message "Username or password is incorrect".
- Given that a user has entered a valid email address and a password and given that the email address is registered into site but password is not typed correctly. When user clicks login button. Then the user should be show a message "Username or password is incorrect".
- Given that a user has entered a valid email address and a password and given that the email address is registered into site and password is also correct. Then:
  - The user should be logged in into the site,
  - The user should be redirected into landing page.
  - The top of all pages should contain user's email address, "My account" link and "Log-off" link.

- Given that user has entered nothing into email and password in login form. When user clicks on Sign up button. Then user should see the sign up form with all the fields empty.
- Given that user has entered an email address and password in login form. When user clicks on sign up button. Then user should see the sign up form with email address and password fields already filled.

Home page Signup box:

- Given that user has entered a text in email textbox and the text is not a valid email. When user clicks anywhere outside the email textbox. Then the user should see red email address textbox and a message “Invalid Email Address” below the textbox.
- Given that user has entered a valid email text in email textbox and password in password box and nothing in Confirm password box. When user clicks Sign up button. Then the user should see red confirm password box and a message “Confirm Password is required”.
- Given that user has entered a valid email text in email textbox and password in password box and a text in Confirm password box that is different from what was entered in password. When user clicks Sign up button. Then user should see red confirm password box and a message “Confirm Password and Password should be match”.
- Given that user has entered a valid email text in email textbox and same text in password and confirm password field, and not checked “I agree terms and Condition” checkbox. Then the user should see message “Please check 'License Agreement'”.
- Given that user has entered a valid email text in email textbox and same text in password and confirm password field and checked “I agree terms and Condition” checkbox. Then the user should get registered and the user should automatically get signed in.
- Given that user is in sign up box. When user clicks “Back to login” link. Then the user should see login box again.

Login landing page:

- Given that user is logged in. When user clicks on “My account” link at the top. Then the user should be redirected into landing page.
- Given that user is logged in. When user is in landing page. Then user should see 3 tabs:
  - CAPTCHA solution.
  - Content protection and
  - Change password.
- Given that user is in landing page. The “CAPTCHA solution” tab should be the active tab.

CAPTCHA solution page:

- Given that user is not logged in. When user tries to enter the CAPTCHA solution page. Then the user should get redirected into home page with login box highlighted.
- Given that user is logged in. When the user is in landing page. Then user should see following links at the left:
  - Introduction
  - Language independent
  - Asp.net web from
  - Asp.net mvc
  - PHP
- Given that the user is in landing page. When user clicks any of the links listed in the left. Then user should see information about that link in the right.

Content protection page:

- Given that user is not logged in. When user tries to enter the content protection page. Then the user should get redirected into home page with login box highlighted.
- Given that user is in landing page. When user clicks “Content protection” tab. Then the user should see the content protection page.
- Given that user is logged in. When user is in content protection tab. Then user should see 3 sub tabs:

- Hosted encryption
- Self hosted encryption
- Fonts.
- Given that user is logged in. When user is in content protection tab. Then hosted encryption tab should be the active tab.

Hosted encryption page:

- Given that user has entered nothing into paragraph name textbox and content textarea. When user clicks “Save” button both the textbox should get red and show respective error messages.
- Given that user has entered text in both paragraph name textbox and content textarea. When user clicks “Save” button. Then:
  - The paragraph should get saved.
  - A new paragraph box should appear below the current paragraph box.
  - A cross (“x”) should appear at top right corner of the paragraph box.
- Given that user has entered text in the content textarea. When user selects a font from font’s dropdown list then the text in the content textarea should be shown in the selected font.
- Given that user has entered text in the content textarea. When user selects a font from size dropdown list then the text in the content textarea should be shown in the selected size.
- Given that user has saved a paragraph. When user clicks on the cross at top right corner of the paragraph box. Then the user should get a confirmation message asking if he/she wants to delete the message. If user clicks yes then message should get deleted
- Given that user has saved a paragraph. When user clicks on the cross at top right corner of the paragraph box. Then the user should get a confirmation message asking if he/she wants to delete the message. If user clicks no then message should not get deleted.
- Given that user has not saved a paragraph. When user clicks on the “code” button. Then the user should see a dropdown with options “Iframe” and “JavaScript” and blank text area.
- Given that user has saved a paragraph. When user clicks on the “Code” button. Then the user should see a dropdown and textarea with html code for Iframe.

- Given that user has saved a paragraph and clicked on “Code” button. When user changes the dropdown value to “JavaScript”. Then the user should see html code for JavaScript in the textarea.

Self hosted encryption page:

- Given that user is not logged in. When user tries to enter the Self hosted encryption page. Then the user should get redirected into home page with login box highlighted.
- Given that user has not selected a font from dropdown list. When user clicks “download kit” button. Then the font’s dropdown list should get highlighted in red color.
- Given that user has selected a font from dropdown list. When user clicks “download kit” button. Then a zip file should get downloaded.
- Given that user has selected

Font listing page:

- Given that user is not logged in. When user tries to enter the Font listing page. Then the user should get redirected into home page with login box highlighted.

Change password page:

- Given that user is not logged in. When user tries to enter the Change password page. Then the user should get redirected into home page with login box highlighted.
- Given that user has not entered Old password. When user clicks change password button. Then a message “'Old Password' should not be empty” should be shown.
- Given that user has not entered New password. When user clicks change password button. Then a message “New Password' should not be empty” should be shown.
- Given that user has not entered Confirm password. When user clicks change password button. Then a message “Confirm Password' should not be empty” should be shown.
- Given that user has entered different values in New password and Confirm password. When user clicks change password button. Then a message “'Confirm Password' should be equal to 'New Password'” should be shown.
- Given that user has entered wrong Old password, and same values into New password and confirm password. When user clicks change password button. Then

a message “The current password is incorrect or the new password is invalid.” Should be shown.

- Given that user has entered correct old password and same values into new password and confirm password. When user clicks change password button. Then the password should get changed.
- Given that user has changed password and logged off from the system. When user tires to login using old password. Then user should not be able to login.
- Given that user has changed password and logged off from the system. When user tires to login using new password. Then user should be able to login.

Logoff link:

- Given that user is logged in. When user clicks the Logoff link. Then the user should get logged out and redirected to homepage.
- Given that user is not logged in. When user is in any page. Then the user should not see the log-off link.

My account link:

- Given that user is logged in. When user clicks the “My account” link. Then the user should get redirected into the landing page “CAPTCHA solutions” tab.
- Given that user is not logged in. When user is in any page. Then the user should not see the “My account” link.

Database Reset Script

This test script should run before beginning testing. These scripts will cleanup data making the database ready for testing.

```
DELETE FROM [BotSecurity_test].[dbo].[aspnet_UsersInRoles]
DELETE FROM [BotSecurity_test].[dbo].[aspnet_Membership]
DELETE FROM [BotSecurity_test].[dbo].[HostedFontContent]
DELETE FROM [BotSecurity_test].[dbo].[HostedFont]
DELETE FROM [BotSecurity_test].[dbo].[HostedParagraph]
DELETE FROM [BotSecurity_test].[dbo].[aspnet_Users]
DBCC CHECKIDENT
('[BotSecurity_test].[dbo].[HostedFontContent]', reseed,0)
DBCC CHECKIDENT
('[BotSecurity_test].[dbo].[HostedFont]', reseed, 0)
DBCC CHECKIDENT
('[BotSecurity_test].[dbo].[HostedParagraph]', reseed, 0)
```

Appendix 3

System Diagrams

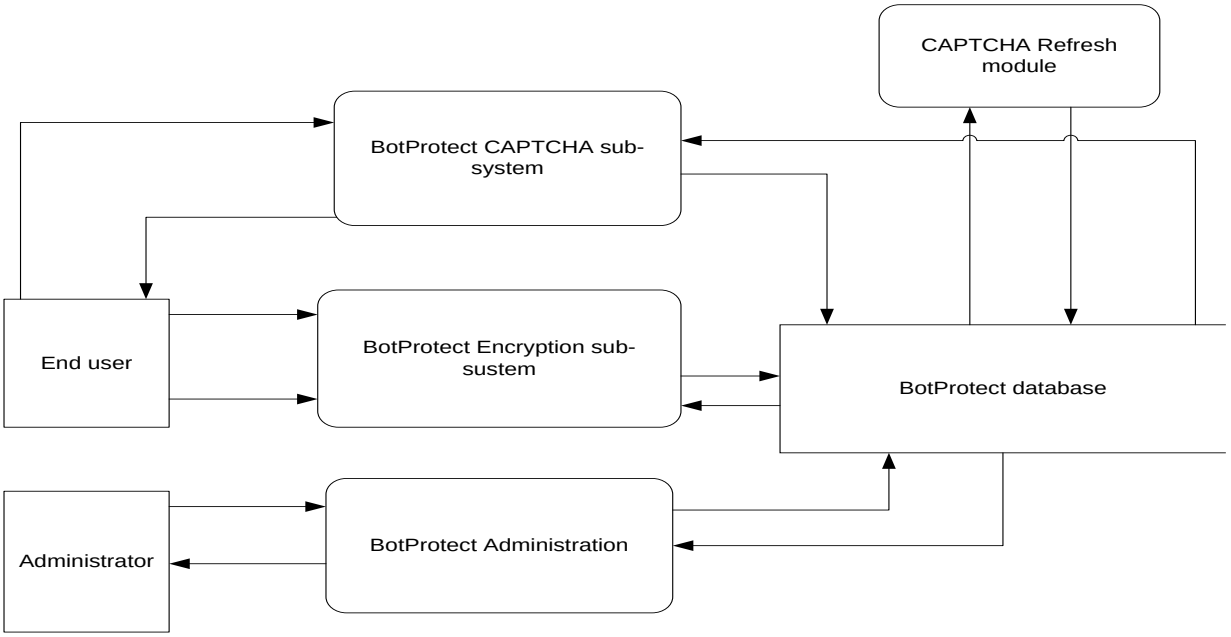


Figure A.2: System Diagram for BotSecurity system

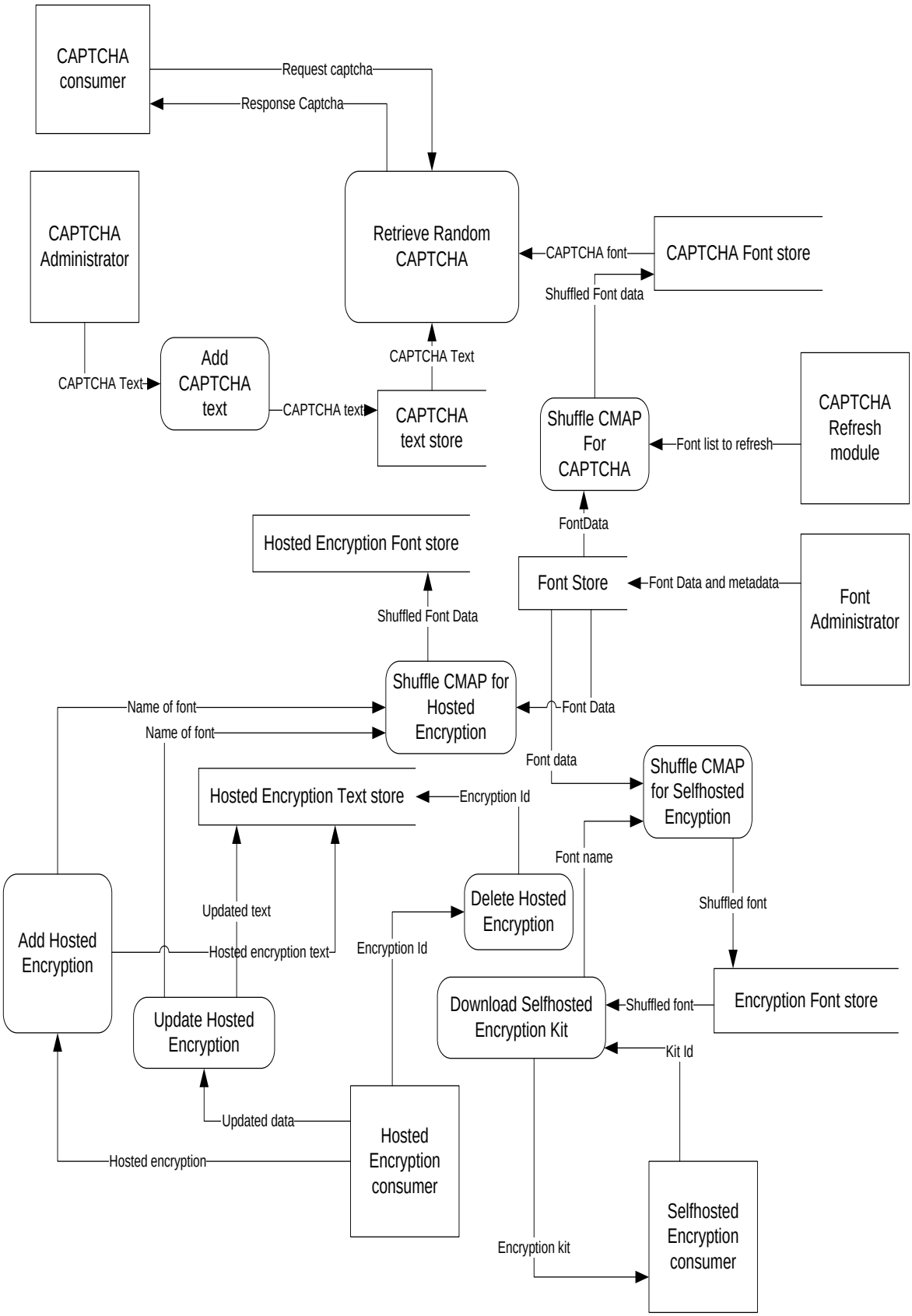


Figure A.3: Data Flow Diagram for BotSecurity system

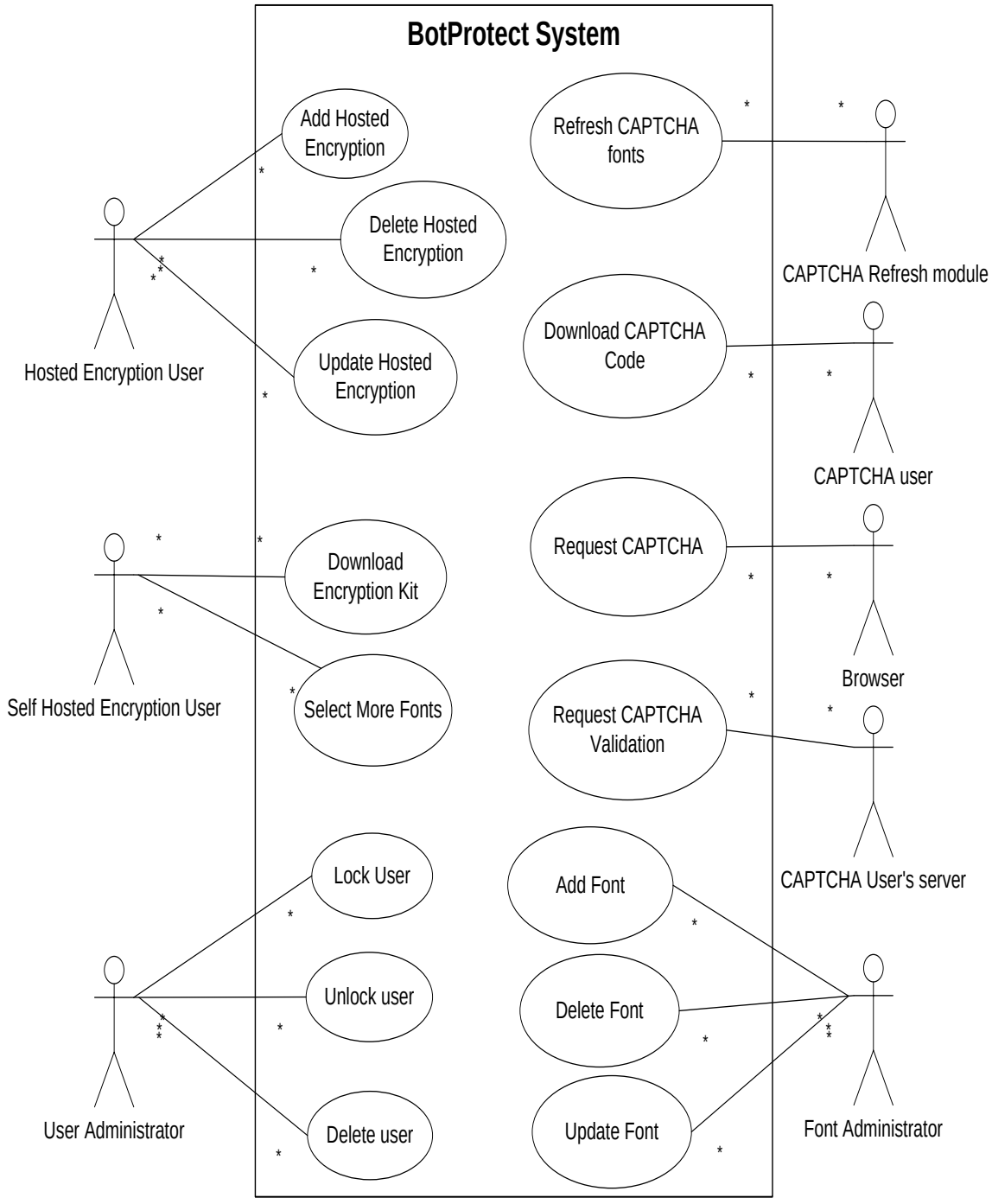


Figure A.4: Use Case Diagram for BotSecurity system

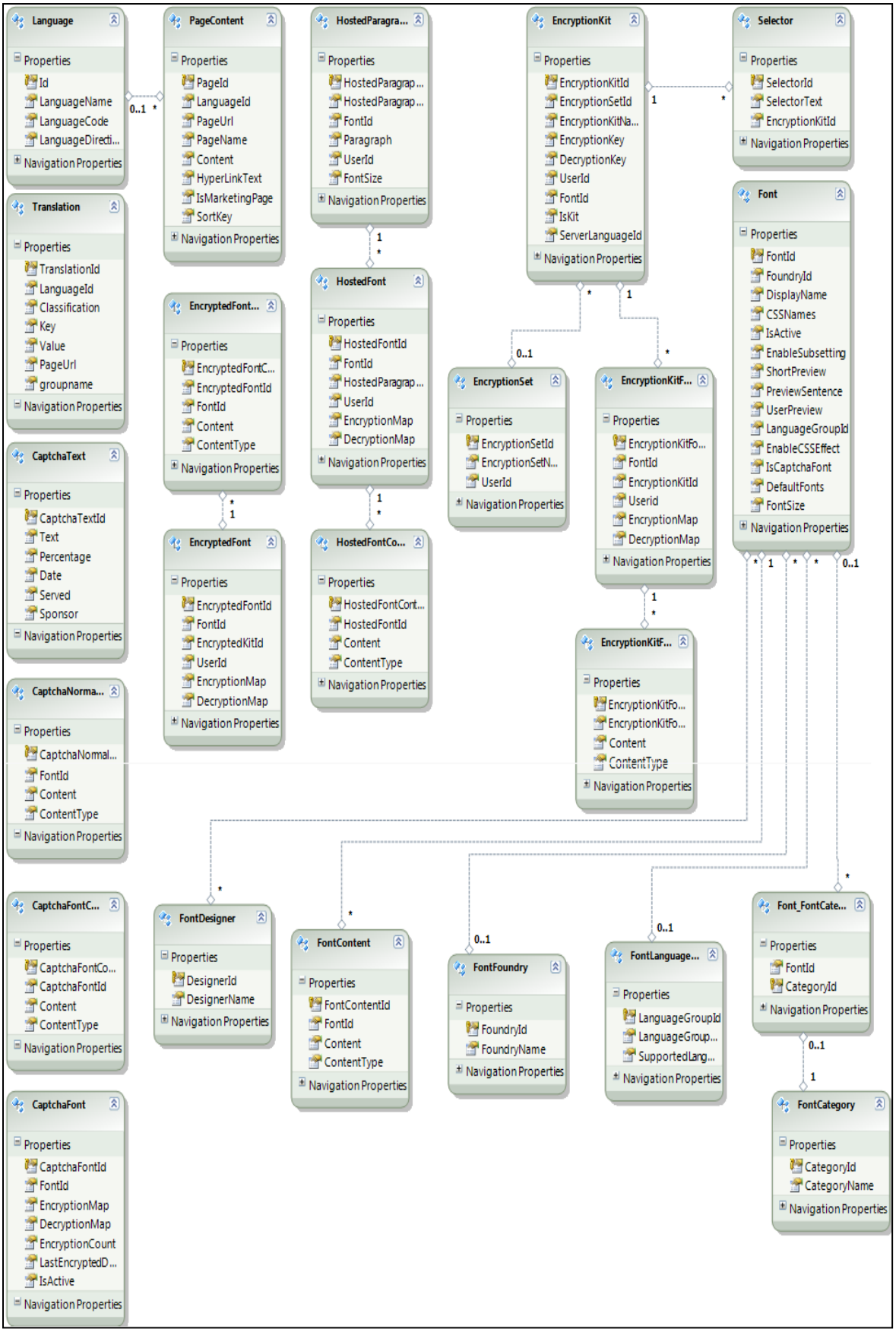


Figure A.5: Database Diagram for BotSecurity System