Evaluating the Security of CAPTCHAs utilized on Bangladeshi Websites

Md. Neyamul Islam Shibbir[a], Hasibur Rahman[a], Md Sadek Ferdous[b] and Farida Chowdhury[b,*]

[a]*Shahjalal University of Science and Technology , Sylhet, Bangladesh*
[b]*BRAC University, Dhaka, Bangladesh*

ARTICLE INFO

Keywords:
CAPTCHA
CAPTCHA Attack
Security Evaluation Framework
Automated Bot Attack
OCR
Bangladeshi Websites

ABSTRACT

A number of attacks such as spamming, email-address harvesting, dictionary attacks, brute-force attacks, credential surfing attacks, DOS attacks and others can be executed relatively easily if the target system cannot differentiate between requests coming from a real human user and those generated artificially by bots. In order to distinguish between human users and bots, the Completely Automatic Public Turing Test (CAPTCHA) has been used for almost 20 years. Nevertheless, CAPTCHA can be easily broken if it lacks security features. Consequently, the motivation for utilizing CAPTCHA is undermined. This article proposes a framework that can evaluate the security aspects of text CAPTCHAs. The effectiveness of the framework is validated by using a generalized attack method consisting of pre-processing and utilizing an Optical Character Recognition (OCR) library such as Tesseract OCR. We have carried out a systematic study of existing CAPTCHAs utilized on Bangladeshi websites. We have studied 238 Bangladeshi websites and found that only 44 websites used a CAPTCHA mechanism. Then, we have assessed the security levels of the selected text CAPTCHAs by the proposed framework. As per our analysis, we have found that there are 29 different CAPTCHA schemes being adopted on Bangladeshi websites and among them 23 are vulnerable to automated attacks. In order to address these issues, we also present a number of recommendations.

1. Introduction

The web has been playing a pivotal role in the world with a tremendous impact on different aspects of our lives all over the world. The web traffic is used to quantify the total usage of the web, an indicative parameter of the popularity of the web [1]. Surprisingly, according to the Imperva's Bad Bot Report [2], from 2015 to 2021, bot traffic comprises nearly half of all web traffic. Any non-human traffic to the web can be referred to as bot traffic. This means any legitimate collection of requests to the web made through an automated process rather than directly by a human user can be regarded as bot traffic. A bot represents a computer program acting as an agent on behalf of its owner or developer [3]. Bots on the web can sometimes be handy and sometimes very harmful. Bots used for malicious activities are known as malicious bots or bad bots. Table 1 presents the percentage of web traffic for humans, good bots and bad bots in between 2015 to 2021. As per the table, the percentage of malicious bot traffic is increasing yearly. For example from Table 1, in 2021, 14.6% of web traffic are from good bots. However, the malicious bots make up for almost a record-setting 27.7% of the web traffic. Malicious bots are being used for many malicious activities such as email-address harvesting, brute-force attacks, credential surfing attacks, DOS attacks, account creation, website scraping, spamming, denial of inventory, online poll attack and many more [4].

A preventive mechanism to mitigate these kinds of bad bot requests has been used for almost 20 years. This mechanism is known as the Reverse Turing Test or Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA). CAPTCHA is widespread, easy, and convenient to implement in any system. Nevertheless, it has many drawbacks, security flaws and usability issues and can also be broken by attacks [5, 6, 7, 8]. Many techniques are being developed to design a more secure CAPTCHA. However, CAPTCHA solver models are also being improved to attack them. Designing a secure CAPTCHA and attacking it to understand the security aspects of this mechanism to make it more secure is a recurring process.

However, both processes should be independent of each other, like a framework that can primarily evaluate how secure a CAPTCHA is without carrying any attacks on them. Generalizing attacks on all kinds of CAPTCHAs is also an open challenge, as almost all the other works on attacking methods can attack one particular type of CAPTCHA [9, 10]. This article proposes an Evaluation Framework that can primarily evaluate how secure a CAPTCHA is without running any attacks. To our knowledge, no work has been done on developing such a framework.

The framework could evaluate text CAPTCHAs in any website, however, in this article, we have utilized the framework to evaluate different security aspects of CAPTCHAs deployed on Bangladeshi websites. This is because security has been an important issue since the Bangladeshi Bank (the central Bank of Bangladesh) money heist in 2016 [11, 12, 13]. There are a few research articles which have found the lack of proper security mechanisms on Bangladeshi websites [14, 15]. In these articles, the authors proposed different methods consisting of various testing and analysis to detect

*Corresponding author
neyamul.sbr@gmail.com (Md.N.I. Shibbir); hamim4389@gmail.com (H. Rahman); sadek.ferdous@bracu.ac.bd (M.S. Ferdous); farida.chowdhury@bracu.ac.bd (F. Chowdhury)
ORCID(s): 0000-0002-8361-4870 (M.S. Ferdous); 0000-0001-9902-6291 (F. Chowdhury)

Table 1
Web Traffic Percentage By Year [2]

Year	2015	2016	2017	2018	2019	2020	2021
Human	54.4	61.3	57.8	62.1	62.8	59.2	57.7
Good bots	27	18.8	20.4	17.5	13.1	15.2	14.6
Bad bots	18.6	19.9	21.8	20.4	24.1	25.6	27.7

the vulnerabilities present on the selected websites. These articles draw attention to the flaws on Bangladeshi websites and the need for better web security measures. To stop data breaches and other cyberattacks, they underline the significance of identifying and mitigating vulnerabilities. Under the same line of motivation, we want to evaluate, using the proposed framework, how securely CAPTCHAs have been deployed on many Bangladeshi websites. Furthermore, we have found no systematic study regarding Bangladeshi websites utilizing CAPTCHAs, which raises the concern for various malicious attacks and leaves their security aspects unchecked. To address these identified issues, we have utilized the evaluation framework to examine 29 different CAPTCHA schemes (19 text CAPTCHA schemes and 10 HTML CAPTCHA schemes, explained later) adopted on different Bangladeshi websites. We have also evaluated the effectiveness of the proposed framework and we have found out that the evaluation framework has been successful to predict 9 out of 9 Vulnerable CAPTCHA schemes (an effectiveness of 100%), 4 out of 5 Moderately Secure CAPTCHA schemes (80% effective) and 5 out of 5 Secure CAPTCHA schemes (100% effective). All of the 10 HTML CAPTCHA schemes have been successfully broken using traditional web scraping techniques, detailed out in Section 5.2.1. We have conducted a comprehensive evaluation of the performance of our evaluation framework using F1 score and Accuracy, in predicting Vulnerable, Moderately Secure, and Secure CAPTCHAs. Our framework has achieved a micro-average accuracy of 96% and a micro-average F1 score of 94%. We have also calculated the p-value to underline the confidence on our evaluation framework.

A summary of major contributions of the article is presented next. This is the first article to:

- conduct a comparative analysis for CAPTCHAs deployed on Bangladeshi websites as to identify different implementation aspects and categorize the deployed CAPTCHAs using a few criteria.
- develop an evaluation framework for text CAPTCHAs that can evaluate the security level of CAPTCHAs without executing any attack.
- present a general attack method that is applicable to attack any textual CAPTCHA without requiring to manually annotate massive amounts of data of each type of CAPTCHA and train them individually.
- validate the effectiveness of the evaluation framework with the presented attack method.
- present a number of appropriate recommendations and suggestions to mitigate the security concerns found in scrapped CAPTCHAs.

2. Background

In this section, a brief background on CAPTCHA, its security, usability and other aspects are presented.

2.1. Reverse Turing Test

For many online services, it is imperative to distinguish between humans and bots automatically to countermeasure malicious bots. This is known as the *Reverse Turing Test* [16]. In this test, the investigator is the system/computer itself that identifies if the user is a human or a bot, hence the name Reverse Turing Test as in a Turing Test, a system and a human assume the opposite roles. A reverse Turing test employs tests that are easily solvable by humans but theoretically impossible to be solved by not-human entities namely automated bots. In reality, Reverse Turing Test is considered valid if the tests can be solved by humans with a success rate over 90% and bots success rate less than 0.1% [17].

2.2. CAPTCHA

CAPTCHA is one of the most important and widely used Reverse Turing Tests that is employed on the websites to prevent malicious bots.

CAPTCHA is one of the most important and most commonly used types of HIPs (Human Interactive Proofs) [18] system. CAPTCHA is a kind of HIPs problem or challenge used to differentiate between humans and computers. It is possible because CAPTCHA uses a problem easily solvable by humans but very hard to solve by computers. In light of this, CAPTCHA is a test software for completing a certain kind of work that is more appropriate for people than bots. If the CAPTCHA response is accurate, the computer recognizes the user as a person.

The most widely used CAPTCHAs are Text CAPTCHAs (also called static CAPTCHAs) [19], Image CAPTCHAs (also called dynamic CAPTCHAs) [20], Audio CAPTCHAs which are generally used for visually impaired visitors and Intellectual/Game based CAPTCHAs [21].

2.3. Security, Usability and Practicality

According to Brodes et al. [17] there are three important factors to consider while designing a CAPTCHA mechanism. They are:

- **Security**: The security element defines the process to protect the CAPTCHA from being attacked.
- **Usability**: The usability element defines the solvability of CAPTCHAs by the human users.
- **Practicality**: Finally, the practicality element defines the feasibility of the mechanism to be implemented on a website.

There is always a trade-off between CAPTCHA security and usability, as adding more security features make the CAPTCHA less readable and solvable for human users. That is what makes designing a CAPTCHA challenging, as they have to balance security and usability at the same time.

2.4. Security Features of CAPTCHAs

All the different types of existing CAPTCHAs have their strengths and weaknesses and different techniques and models to attack and break them. In order to increase the security of CAPTCHAs, one can include numerous security characteristics to make it difficult to crack, such as deformation, color variation, rotation, blurring, warping, multi-level structure, overlapping of the characters, noisy background and so on [17, 22, 5].

Text CAPTCHAs typically use distorted words to increase security. However, using certain letters are discouraged. These letters include 6/G, b/5, S/s, O/0, Z/z, nn/m, and others since they can be difficult for real humans to recognize after being distorted. Even so, typical text CAPTCHAs can easily be broken using OCR (Optical Character Recognition) techniques [23]. Some text CAPTCHAs are Gimpy [24], EZ-Gimpy [25], Baffle Text [26] and Pessimal Print [27].

Image CAPTCHAs are far more challenging to break than text CAPTCHAs. They must shield against image processing and pattern recognition attacks deployed using various machine learning and deep learning models. Image-based CAPTCHAs are non-OCR-based CAPTCHAs. Some of them are PIX CAPTCHAs [28] and Bongo CAPTCHAs [29]. Different sorts of distortions are crucial for image-based CAPTCHAs. The primary atomic distortions are luminance, color quantization, dithering, cutting and rescaling, and line and curve noise.

Audio CAPTCHA is another variety of CAPTCHA. The overlap of target sounds, the random number of target voices inside a cluster, and the addition of stationary noises are some defense mechanisms used by this version of CAPTCHA.

Then, there are math CAPTCHAs [30], which use math operation questions to challenge users.

It is to be noted that, in this article, we focus on text CAPTCHAs.

2.5. Evaluation Framework

A framework can be defined as a structure built to guide one through a solution set [31]. The evaluation framework on CAPTCHA security is the structure that will evaluate how secure a CAPTCHA is, considering the security and vulnerability features of that CAPTCHA. In our proposed framework, both security and vulnerability features for CAPTCHAs are defined and extracted from the CAPTCHAs accurately, and then an evaluation of how secure the CAPTCHA is made based on these features.

2.6. Generalized Attack Method

A generalized attack method on text CAPTCHAs is a single method of attack that applies to all the different kinds of text CAPTCHAs. Almost all the prior works on attacking text CAPTCHA are not generalized. All modern Machine Learning (ML) and Deep Learning (DL) based attacks need a huge amount of annotated data for each type of CAPTCHA [32]. This means a considerable level of exhausted human interactions and the individual training of a new model are needed for each type of text CAPTCHA, causing time and computational consumption. On the contrary, our proposed generalized attack pre-processes the CAPTCHA with some combinations of operations in such a way that it can be recognizable by any modern OCR.

2.7. Tesseract OCR Engine

Tesseract OCR (Optical Character Recognition) Engine is an open-source OCR engine [33]. Originally developed by Hewlett-Packard, however, it is currently an open-source software. The Tesseract OCR engine functions using the following steps: i. Adaptive threshold, ii. Connected component analysis, iii. Find lines and words and iv. Recognize words [34]. We have used the Tesseract OCR to recognize the noiseless text CAPTCHA characters because of its impressive accuracy.

3. Related Work

Studies on the security aspect of CAPTCHAs are not new. From the very first proposal of CAPTCHA [35], the researchers are proposing many different mechanisms of CAPTCHAs to make it more secure. Nevertheless, researchers are constantly discovering new ways to attack them. Earlier studies to break CAPTCHAs used shape context matching type techniques that could break earlier EZ-Gimpy CAPTCHAs at a decent success rate [36]. Some works also used ad-hoc techniques like distorted template image matching [37]. In 2005, Chellapila et al. [38] proposed a solving technique that introduced the first ML based recognition on single characters. However, they did not work on the segmentation and noise removal as they assumed segmentation had been solved on CAPTCHAs. The same authors proposed a method of automatic segmentation using machine learning. The method validated 1 false positive per 4 true positives. However, the results of the automatic segmentor they proposed were not provided.

In the early stage, CAPTCHAs had the least amount of security features. So back then, the attack methods presented in [39, 40, 40] worked on the CAPTCHAs with fewer security features such as no background interference or blurring. Then, CAPTCHA designers came up with CAPTCHAs such as Megaupload CAPTCHA, Reddit CAPTCHA, Wikipedia CAPTCHA, and Microsoft CAPTCHA [18] with more security features. The Megaupload CAPTCHAs robustness was cracked by El Ahmad et al. [41]. Also, the usability of the CAPTCHA used on megaupload.com has been criticized as poor [41]. Microsoft CAPTCHA was broken by Yan et al. [9]. They also broke Yahoo CAPTCHA that was used on the Yahoo website back then [42]. Then, the designers tried to design more secure text CAPTCHAs.

In 2011, Bursztein et al. [5] published some methods and ultimately developed an application named Decaptcha that successfully cracked 13 of the 15 popular and secure CAPTCHAs at that time. While it is clear that new CAPTCHAs are continually being developed with more security features to fight against automated attacks, new methods of attacking them are constantly being proposed and executed. Thus, the dynamic of one researcher creating new CAPTCHAs and another breaking them with new attack methods is a cyclic and continuous process.

Many ML based attacks have been proposed recently, like the CNN model with focal loss function [43], using GAN [44], PCA [45] and R-CNN [46]. All these ML methods have one main problem: human intervention. A massive set of manually annotated data is needed to train those ML models, which is exhausting and not ideal for attacks if it can be attacked in other less exhausted ways [44]. A semi-supervised approach [47] was proposed but resulted in low accuracy.

In 2014, Bursztein et al. [48] claimed to solve all the real-world text CAPTCHAs. They concluded that purely text CAPTCHAs might lose usefulness because of the advanced attack methods.

Nevertheless, in our findings, text CAPTCHAs are still the most commonly used CAPTCHAs by many websites [49]. It is because of its economical low cost and easy development aspects, low computational resource, and low hardware cost [50]. This is particularly true for underdeveloped and developing countries where the cost of the alternatives of text CAPTCHAs is not viable, text CAPTCHAs are used as a viable option. Also, the alternatives are few, and developing and deploying them is still an open challenge as academics and developers are still working on the alternatives of CAPTCHAs in the reverse Turing process [51, 52, 53]. Therefore, it is necessary to evaluate how secure text CAPTCHAs are without running any attacks.

The only work done regarding an evaluation framework for the systemic study of CAPTCHA is by Achint et al. [54]. However, the work lacks details as it only focuses on distortions as a security feature and has no information on evaluating new CAPTCHAs.

4. Methodologies

In this section, we describe our research methodologies. There are six major steps involved in our methodology which are as follows:

i) **Defining evaluation criteria of CAPTCHA**: Defining criteria for evaluating the strength and vulnerability of any text CAPTCHA.
ii) **CAPTCHA collection**: Scrapping CAPTCHAs from Bangladeshi websites.
iii) **Evaluation Framework**: Creating an evaluation framework that is based on defined criteria to predict the security level of any text CAPTCHAs.
iv) **CAPTCHA attack**: Attacking selected CAPTCHAs to break them.
v) **Evaluation**: Evaluating the success rate at breaking CAPTCHA based on the defined criteria.

Table 2
Security Features

Anti pre-process	Anti segmentation	Anti classification
Textured Background	Overlapping Characters	Rotation
Connected Characters	Deformation	Wrapping
Hollow Scheme	Blurring	Distortion
Noise Background		
Color Variation		
Multi-Layer Structure		
Rotation		
Warping		

Table 3
Vulnerability Features

Pre-process	Segmentation	Classification
Constant Background	Aligned Character	Only Letters and Numbers
Binary Color		Only Uppercase/ Lowercase
		Only Constant Font
		Dictionary Words

vi) **Recommendations**: Providing some recommendations for Bangladeshi websites on secure CAPTCHA deployment, based on our study, evaluation, and attack methods.

In this section, we present the steps i. to iii. The rest of the steps are presented in the subsequent sections.

4.1. Evaluation Framework and Security Level Assumptions

As discussed earlier, to the best of our knowledge, there is no empirical study conducted targeting the vulnerability of CAPTCHAs deployed on Bangladeshi websites. Towards this aim, we intend to develop a framework to evaluate the security aspects of any text CAPTCHAs.

4.1.1. Defining Evaluation Criteria of CAPTCHA

The first step to formulate an evaluation framework is to define the respective criteria for evaluating CAPTCHAs. We have fixed two sets of criteria: one defines how strong a CAPTCHA is, and the other defines how weak or vulnerable a CAPTCHA is. These two sets are discussed next:

i) **Security Features:** Security features represent those features that make a CAPTCHA secure, as they have Anti preprocess, Anti segmentation, and Anti classification properties [55] Our selected security features, belonging to different classification properties, for the framework are shown in Table 2.
ii) **Vulnerability Features:** Vulnerability features are the weaknesses of CAPTCHAs. These features represent those properties that make an automated attack on a CAPTCHA easier by utilizing pre-processing, Segmentation, or Classification techniques [17, 56]. Our selected vulnerability features for the framework are presented in Table 3 where they belong one the three techniques (pre-processing, segmentation and classification). The less these features are in a CAPTCHA, the more secure it becomes.

4.1.2. CAPTCHA Collection

After defining the evaluation criteria, we collected real world CAPTCHAs to evaluate them with the evaluation criteria. In this paper, we have focused only on Bangladeshi websites. To our knowledge, no research has been done on the security aspects of the CAPTCHAs used on Bangladeshi websites. In fact, no research has been done on Malicious bot countermeasure aspects on Bangladeshi websites. That is why it is not known till now about how secure Bangladeshi websites are against automated malicious bot attacks. Towards this aim, we have defined this collected CAPTCHA phase into the following sub-phases:

i) Curating a list of Bangladeshi websites
ii) Finding CAPTCHAs from the websites
iii) Identify any invisible reCAPTCHA [57] using Googlebot [58]
iv) Detecting any hidden CAPTCHA utilizing a semi-automated algorithm
v) Scrapping the CAPTCHAs from the websites

In the following, we have defined a CAPTCHA Scheme as follows: A CAPTCHA Scheme can be defined as a collection of CAPTCHAs that have some common features, e.g each CAPTCHA having the same font, random similar morphing, random but similar hollow scheme and among others. However, there will be variations in those features for each CAPTCHA. Each variation of the CAPTCHA will adhere to these criteria or any fixed, selected criteria, ensuring that they can be categorized under the same scheme. Next, we elaborate on each phase.

Phase 1: Creating a list of Bangladeshi websites: As our primary goal of this work is to measure the security of CAPTCHAs deployed on Bangladeshi websites, we have curated a list at first. The curated websites can be classified into two categories:

- Government (Govt.) owned websites
- Private websites

For the Govt. websites, there are two online portals named National Portal of Bangladesh [59] and Services Portal of Bangladesh [60]. From these two portals, we have collected 134 Govt. websites. On the other hand, 104 private websites have been collected, based on the traffic each generates, from multiple blogs and commercial and non-commercial websites. We have used Semrush [61] to get the list of top private websites in Bangladesh and then have used SimilarWeb [62] to filter the list based on the traffic it generates. In total, we have collected 238 Bangladeshi websites from these sources. We have mainly focused on collecting as many Govt. websites as possible, as they hold significant information about Bangladeshi citizens. The Bangladesh bank cyber heist [12] that occurred back in 2016 shows the lack of security measures employed in Govt. websites, the same is confirmed in [14] where the authors highlighted that between Govt. and private websites, Govt. websites were more vulnerable to various malicious attacks. Nevertheless, to get a general idea about CAPTCHA usage in Bangladeshi websites, we have curated a number of 104 private websites.

Phase 2: Finding CAPTCHAs from the websites: In order to prevent spamming, the creation of fictitious accounts, and account harvesting, CAPTCHAs are primarily used on the web forms like login and sign-up forms. Therefore, we mainly looked for them on login and registration pages of the selected Bangladeshi websites. For this, we created a bot to detect any hidden and invisible CAPTCHAs on the websites, which is discussed in detail in Phase 3. Then, we looked at the other online forms on the selected websites. Also, we implemented an algorithm to detect server-side hidden CAPTCHAs, if there are any, is described in Phase 4. Only 44 of the 238 Bangladeshi websites that we searched for had a CAPTCHA mechanism. Among these 44 websites, 19 websites used text CAPTCHAs, 15 used reCAPTCHAs and 10 used HTML text CAPTCHAs. We discovered only one math-based CAPTCHA among all the text CAPTCHAs. It is clear that text CAPTCHAs are used in the majority of them. Out of all 44 websites, all the 15 reCAPTCHA based websites are suitable for visually impaired people.

Phase 3: Detecting invisible CAPTCHAs: No user interaction is needed at all with the invisible reCAPTCHA. Google analyses user behavior, including typing habits, mouse movements, and browser history, in a manner equivalent to the "I'm not a robot" reCAPTCHA [63, 64].

Therefore, invisible CAPTCHAs remain hidden on the page and track the cursor's behaviour to detect humans or bots. In order to find this kind of invisible CAPTCHAs on websites, we needed to use Googlebot on the Chrome browser [65]. In the Chrome developer mode, we created a bot using Googlebot and we browsed the respective websites. If there were any invisible CAPTCHA mechanisms, they would appear. In this way we have managed to find 2 invisible reCAPTCHAs. Both of the websites are non Govt. websites.

Phase 4: Detecting hidden server-side CAPTCHAs: Hidden server-side CAPTCHAs are CAPTCHAs that are not initially presented on the client-side of the website. Rather these are deployed runtime from the server-side to the client-side due to some triggering mechanisms set by the website developer. These triggers could include delayed responses, bot-like activities, or frequent page refreshing. However, the specific thresholds for these activities, such as the duration of any inactivity or exact bot activities, or the number of page refreshes that would activate the CAPTCHA, are typically not disclosed. Therefore, without insight into the website development, it is challenging to predict exactly when and why a hidden CAPTCHA might be deployed.

Keeping this challenge in mind, we have devised a semi-automated approach to detect hidden CAPTCHAs on websites. The approach is termed as *semi-automated* as we manually visit a website, reload it several times (1000 to be specific) and examine the website using an algorithm

Algorithm 1 Hidden CAPTCHA Detection

```
Start
    Function isCAPTCHAPresent(driver):
        if The page contains a CAPTCHA form or an image with alt text containing 'captcha' or class name with text
          'recaptcha' or id with 'recaptcha' or class name with 'g-recaptcha'
            return TRUE
        else
            return FALSE
        end
    Function isProbableCAPTCHA(driver):
        if The refresh button, input text, and any image elements are in close proximity (within less than or equal to 100 pixels
          to each other).
            return TRUE
        else
            return FALSE
        end
    Function main():
        Initialize a list of urls
        Initialize num_reloads to 1000
        for each url in urls do
            Initialize an empty list messages
            for i from 1 to num_reloads do
                try
                    Initialize a Chrome WebDriver as driver with headless mode
                    Navigate to the given url
                    Wait for the page to load using WebDriverWait
                    Capture the HTML content before reloading the page
                    if isCAPTCHAPresent(driver)
                        Append "CAPTCHA detected on at reload no. i" to messages
                    Reload the page by navigating to the same URL
                    Wait for the page to load again
                    Capture the HTML content after reloading
                    if $HTML\_before \neq HTML\_after$
                        Append "New content detected on at reload no. i" to messages
                        if isCAPTCHAPresent(driver)
                            Append "Newly appeared content is a CAPTCHA on" to messages
                        else if isProbableCAPTCHA(driver)
                            Append "Newly appeared content is a probable CAPTCHA need manual check" to messages
                        else
                            Append "But It's not CAPTCHA" to messages
                        end
                    else
                        Append "No new content detected on at reload no. i" to messages
                    end
                catch(Exception e)
                    Append "Error e while loading" to messages
                end
                Close the WebDriver
            end
            Append messages to result_dict with url as the key
        end
End
```

(Algorithm 1), deployed as a script, to identify any hidden CAPTCHA. The algorithm compares the HTML content before and after each reload and searches for common CAPTCHA indicators and new HTML contents that could potentially be a CAPTCHA.

Another manual check is carried out when the algorithm identifies a probable CAPTCHA (when a TRUE value is returned by the *isProbableCAPTCHA* function in Algorithm 1). The *isProbableCAPTCHA* function is designed to detect CAPTCHAs on a web page by identifying common

CAPTCHA elements such as a refresh button, an input text box, and a base64-encoded PNG image. It then checks if these elements are in close proximity, located within 100 pixels of each other, as CAPTCHA elements are typically placed close together on a web page. If such a set of elements is found, the function returns a TRUE value, indicating a probable CAPTCHA. Otherwise, it returns a FALSE value. The manual check involves a thorough review process conducted by ourselves to identify any hidden CAPTCHA, ensuring the accuracy of our findings. After applying Algorithm 1 for all the websites, we have found no hidden CAPTCHAs in them.

While our algorithm is based on the assumption that 1000 refreshes should be sufficient to activate any hidden CAPTCHA, we acknowledge that this may not always be the case. However, considering the manual process of numerous refreshes for each website and the challenge to identify the exact trigger point to activate any hidden CAPTCHA on each website, we need to restrict ourselves to a reasonable number. For our experiment, we have fixed this number to be 1000.

Phase 5: Scrapping CAPTCHAs from the websites: After finding the CAPTCHAs, we scrapped them from the respective websites. For text CAPTCHAs, we downloaded the CAPTCHA images 55 times by reloading the respective web page to store different CAPTCHA samples for a specific CAPTCHA scheme. We have categorized all the collected text CAPTCHAs with an ID, namely *Scheme ID*, an unique ID assigned to a specific type of CAPTCHA so that it can be referenced in the article later. Many websites might utilize the same scheme of CAPTCHAs on them. Hence, one CAPTCHA scheme can be found on different websites. However, one CAPTCHA scheme will belong to one unique Scheme ID. However, during data collection in this phase, we observed a unique one-to-one association where each CAPTCHA scheme is linked to a single website. This mapping is presented in Table 4. Let us elaborate in details the underlying reason for choosing 55 samples for each scheme. Our model is designed for image processing. This implies that if it can identify and denoise a CAPTCHA sample from a specific CAPTCHA scheme, we would like to ensure that it can detect similar samples from the same scheme. All samples of each scheme share the same font and security features, albeit with minor variations. However, to account for all variations within a scheme, we needed some generic samples, these are samples from the same scheme ID, but they differ due to noise, confusing appearance, excessive deformation, or any other characteristics that might challenge our model. We can not arbitrarily select any number of samples, as our model would then be biased towards breaking almost identical samples from the same scheme ID. Generic samples are slightly different in this case, as they are not identical to other samples from the same scheme ID. This ensures a more robust and comprehensive evaluation of our model's performance. In cases where a CAPTCHA scheme presented unique challenges such as overlapping characters, text obscured by lines, or color similarities between the lines and the CAPTCHA text, we took these samples into account. These potentially problematic samples were used to test, making it more versatile and capable of handling a wide variety of CAPTCHAs. The number 55 was chosen as a practical upper limit for the number of CAPTCHA variations we considered for each scheme. This was not an arbitrary choice, but a reflection of the data; the highest number of variations we encountered for any scheme did not exceed 12, specifically for scheme IDs 6, 7, 10, 13, and 15. That is why 55 samples were downloaded from each scheme. If the CAPTCHAs were HTML CAPTCHAs, we did not scrap to take samples as it can be scrapped and solved with "Attack on HTML CAPTCHA" discussed in Section 5.2.1.

Table 4
Selected websites

Scheme ID	Scheme	Website(s)
1	E-Challan	echallan.gov.bd
2	E-Tender	eprocure.gov.bd
3	Bangladesh Aid info	aims.erd.gov.bd
4	Ministry Division BD	cabinet.gov.bd
5	Dept. of Environment	ecc.doe.gov.bd
6	Ibas++	ibas.finance.gov.bd
7	Passport	passport.gov.bd
8	Women and Children	vawcms.gov.bd
9	Visa Office	visa.gov.bd
10	Ministry of Planning	plancomm.gov.bd
11	Road Transport Authority	ipaybrta.cnsbd.com
12	CNG Retest Unit	cngrpgcl.com
13	Taletalk BD	Teletalk.com.bd
14	Bd Jobs	bdjobs.com
15	Ryans	ryanscomputers.com
16	BDStall	bdstall.com
17	Link3	selfcare.link3.net
18	Water Development Board	bwdb.gov.bd
19	Dhaka Power Distribution	dpdc.org.bd

4.1.3. *Evaluation Framework*

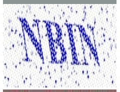
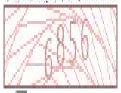
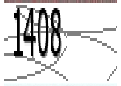
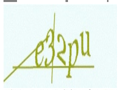
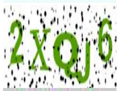
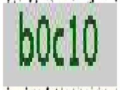
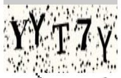
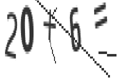
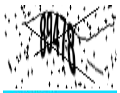
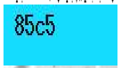
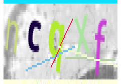
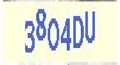
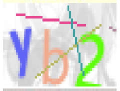
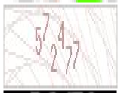
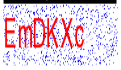
Our next step is to create an evaluation framework using the evaluation criteria discussed previously. Then, we can use the framework to evaluate and predict the perceived security of the collected CAPTCHAs. This is discussed next.

Features mapping: In this step, we utilized the evaluation criteria to map the security and vulnerability features of the collected CAPTCHAs. We manually mapped the text CAPTCHA schemes with evaluation criteria discussed in Section 4.1.1. First, the security features are presented in Table 5. Then, the vulnerability features are mapped in Table 6. These two tables represent the security and vulnerability features of the selected CAPTCHAs.

Some observations from the Table 5 and Table 6 are :

- There is a text-based math CAPTCHA which is different from other CAPTCHAs (Scheme ID 10). Though its current structure is very simple and can easily be broken, its security can be improved using complex math problems.
- Unlike all other CAPTCHAs, a CAPTCHA under Scheme ID 11 is not user-friendly. Though it has

Table 5
Security features of the selected CAPTCHAs

Scheme ID	Example	Anti pre-process	Anti-Segmentation	Anti-Recognition
1		-	-	-
2		-	Character Blurring	-
3		Textured Background,Rotation,Distortion, Noisy Background,Warping	-	Rotation, Distortion, Warping
4		Textured Background, Noisy Background, Same Character and Background, Rotation, Multi-Struct, Warping	Background Blurring	Rotation, Multi-Struct
5		Textured Background, Noisy Background, Same Character and Background	-	-
6		Textured Background , Noisy Background, Color Variation, Rotation,Distortion,Warping, Same Character and Background	Deformation	Rotation, Distortion, Warping
7		Textured Background , Noisy Background, Rotation,Warping	Overlapping Chars, Deformation	Rotation,Warping
8		Textured Background , Noisy Background	Character Blurring	-
9		Textured Background, Rotation, Noisy Background	Overlapping Chars, Deformation	Rotation
10		Textured Background , Noisy Background, Rotation,Same Character and Background	Overlapping Chars, Deformation	Rotation, Distortion, Warping
11		Textured Background , Noisy Background, Color Variation, Rotation, Distortion, Hollow Scheme	Overlapping Chars	Rotation, Distortion, Warping
12		-	Character Blurring, Background Blurring	-
13		Textured Background , Noisy Background, Color Variation, Rotation, Distortion	-	Rotation, Distortion
14		Color Variation	Character Blurring	-
15		Textured Background , Noisy Background, Color Variation, Rotation, Distortion	-	Rotation, Distortion
16		Textured Background , Noisy Background, Rotation,Multi-Struct, Same Character and Background	Background Blurring, Character Blurring	Rotation, Multi-Struct
17		Noisy Background	Deformation, Blurring	-
18		Textured Background , Noisy Background	-	-
19		Noisy Background	Deformation	-

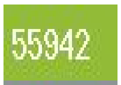
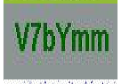
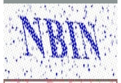
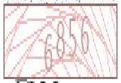
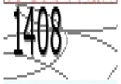
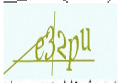
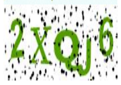
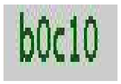
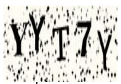
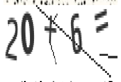
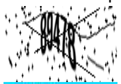
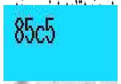
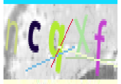
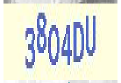
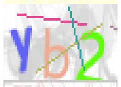
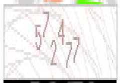
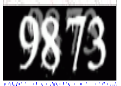
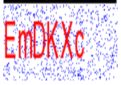
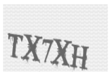
many security features, it is annoying to use and less readable and less usable.

- Even though Scheme ID 13 and 15 look like the same CAPTCHA, they have different numbers of characters in them, which makes them different, hence, needing different pre-processing and different domain attacks.

Security evaluations of CAPTCHAs: In this step, we have created an evaluation framework to assess the security of the selected CAPTCHAs without running any attack on them. For this we have used Table 5 and Table 6 as references.

The security level of any CAPTCHA will define how secure the CAPTCHA is. We have defined three security levels: i) *Vulnerable*, ii) *Moderately Secure* and iii) *Secure*.

Table 6
Vulnerability features of the selected CAPTCHAs

Scheme ID	Example	Pre-process	Segmentation	Recognition
1		Constant Background, Binary Color	Aligned Character	Only Numbers ,Fixed Font Size
2		Constant Background, Binary Color	Aligned Character	Fixed Font Size, Alphanumeric
3		-	-	Only Capital Letters
4		-	-	Only Numbers ,Fixed Font Size
5		-	Aligned Character	Only Numbers ,Fixed Font Size
6		-	-	Lower Case Alphanumeric, Fixed Font Size
7		-	-	Constant Font Size, Upper Case Alphanumeric
8		Constant Background	Aligned Character, Fixed Font Size, Constant Font	Lower Case Alphanumeric
9		-	Binary Color	Upper Case Alphanumeric
10		-	Binary Color	-
11		-	Binary Color	Only Numbers
12		Constant Background, Binary Color	-	Fixed Font Size , Constant Font, Lower Case Alphanumeric
13		-	-	Lower Case Alphanumeric
14		Constant Background, Binary Color	-	Fixed Font Size , Constant Font , Lower Case Alphanumeric
15		-	-	Lower Case Alphanumeric
16		-	-	Fixed Font Size , Constant Font
17		Constant Background, Binary Color	Aligned Character	Fixed Font Size , Constant Font, Only Numbers
18		-	Aligned Character	Fixed Font Size , Constant Font, Upper Case Alphanumeric
19		Constant Background	-	Alphanumeric

Following steps are involved in the evaluation framework:

i) For every individual CAPTCHA, count the total number of Vulnerable Features and Security Features.

ii) Calculate the difference between the total number of Security Features and the total number of Vulnerable Features on an individual CAPTCHA.

$$\Delta D_{C_i} = \sum SF_{C_i} - \sum VF_{C_i} \quad (1)$$

Table 7
Security Level of CAPTCHAs based on the Evaluation Criteria

Scheme ID	Security Features	Vulnerability Features	Differences	Security Level
1	0	5	-5	Vulnerable
2	1	5	-4	Vulnerable
3	7	1	6	Moderately Secure
4	9	2	7	Secure
5	3	3	0	Vulnerable
6	11	2	9	Secure
7	8	2	6	Moderately Secure
8	3	5	-2	Vulnerable
9	6	2	4	Moderately Secure
10	9	2	7	Secure
11	10	2	8	Secure
12	1	5	-4	Vulnerable
13	7	1	6	Moderately Secure
14	2	5	-3	Vulnerable
15	7	1	6	Moderately Secure
16	9	2	7	Secure
17	3	6	-3	Vulnerable
18	2	4	-2	Vulnerable
19	2	2	0	Vulnerable

In Equation 1, ΔD_{C_i} represents the difference between the total number of Security Features (denoted as SF_{C_i}) and Vulnerability Features (denoted as VF_{C_i}) for a particular CAPTCHA C belonging to a Scheme ID i. This implies that the higher the difference (ΔD_{C_i}) the more secure the respective CAPTCHA is.

iii) Set three different ranges for our three security levels.

The count for security features, vulnerability features and the corresponding difference are presented in Table 7. The higher the difference in value, the better the CAPTCHA scheme is.

In order to quantify the security levels, we have used Eq. 2.

$$C_i = \begin{cases} \text{Vulnerable;} & \text{if } \Delta D_{C_i} \leq 0 \\ \text{Moderately Secure;} & \text{if } 0 < \Delta D_{C_i} \leq 6 \\ \text{Secure;} & \text{if } \Delta D_{C_i} > 6 \end{cases} \tag{2}$$

We discuss this selection in the following.

i) **Vulnerable:** Vulnerable CAPTCHAs are defined as CAPTCHAs that can be easily bypassed with most of the automated attacks on the CAPTCHAs. In our framework, if the value of the difference is less than 0, then we have predicted the CAPTCHAs as Vulnerable (Eq. 2).

ii) **Moderately Secure:** If the value of the difference ranges from 0 to 6 (including 6), we have assumed that the CAPTCHA security level is moderate (Eq. 2).

iii) **Secure:** Secure CAPTCHAs are defined as those that are difficult to be compromised using automated bot attack. In this framework, we have considered CAPTCHAs to be secure if the difference is more than 6 (Eq. 2).

Table 8
Security Levels of CAPTCHAs

Security Level	No. of CAPTCHAs	Scheme ID(s)
Vulnerable	9	1, 2, 5, 8, 12, 14, 17, 18, 19
Moderately Secure	5	3, 7, 9, 13, 15
Secure	5	4, 6, 10, 11, 16

The motivation for selecting 3 levels of security with the specific range is as follows: We have selected these ranges for the 3 levels of predictions because of our selected total number of security features is 14 and total number of vulnerability feature is 7 as we can see in Table 2 and Table 3 respectively. So if we align all these features in a single line of distribution, where vulnerability features are -6 to 0 (total 7) and security features are in the positive range (1 to 14) and then calculate the *Equal Width Interval Discretization* [66] with k=3 (Secure, Vulnerable, Moderately Secure), we find the interval 6.66. That is why we have selected this range for the prediction described above.

After applying this security levels as per Eq. 2, we have found only 5 Secure CAPTCHAs, 5 Moderately Secure CAPTCHAs, and 9 Vulnerable CAPTCHAs as presented in Table 8.

5. Attack on the Bangladeshi CAPTCHAs

After scrapping CAPTCHAs from Bangladeshi websites, we have found three kinds of CAPTCHAs, and they are:

i) Text CAPTCHAs (19)

ii) HTML CAPTCHAs (10)

iii) reCAPTCHAs (15)

We have attacked the first two types of CAPTCHAs in this paper. We have not tried to attack any reCAPTCHA because its usage on Bangladeshi websites is comparatively very low.

5.1. Attacking Text CAPTCHAs

Text CAPTCHAs usually challenge users to identify a distorted sequence of English or Arabic characters. Designers have implemented diverse resistance strategies to enhance the security that might be categorized into anti-segmentation and anti-recognition processes. If these security features of CAPTCHAs are not properly utilized in a CAPTCHA, it can be easily broken or bypassed. However, there is a trade-off between usability and increased security while deploying a CAPTCHA. If we want to make a CAPTCHA more secure by adding more security features, the usability of the CAPTCHA decreases proportionally [67].

As per our investigation this is the most used type of CAPTCHAs among the three types of CAPTCHAs found on Bangladeshi websites. Examples of some of the text CAPTCHAs are presented in Figure 1. Next, we elaborate the steps for attacking text CAPTCHAs.

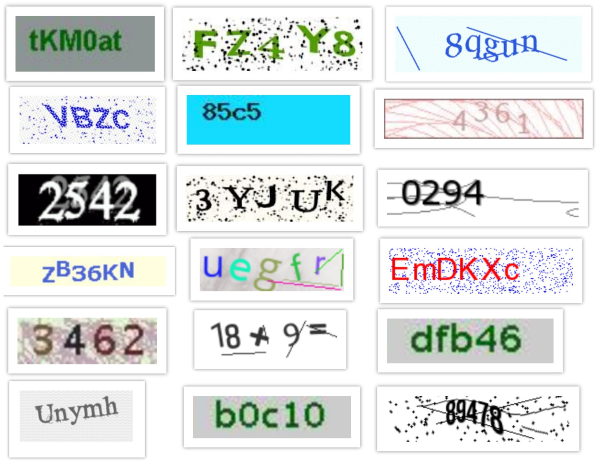


Figure 1: Some examples of text CAPTCHAs extracted from Bangladeshi websites

The method that we have utilized to attack a text CAPTCHA consists of the following three stages (Figure 2):

i) Pre-processing stage

i) OCR stage

i) Post-processing stage

In the first stage, we have extracted the targeted string from the security features like (Noise Background, Hollow Scheme, and others) to pre-process each CAPTCHA. In the second stage, without the security features, we have used a state-of-the-art OCR tool *Tesseract OCR* [68] to segment the CAPTCHA and then recognized the texts inside the CAPTCHA, thereby, breaking it. Finally, we have done some post-processing, if required.

In the following, we explain each of these stages in details.

5.1.1. Pre-processing Stage

In this stage, we have used OpenCV [69] and Pillow [70]. OpenCV is a popular open-source, pre-built library (package) for CPU-only computer vision. On the other hand, Pillow [70] provides an extensive selection of image processing options, including image modification, rotation, scaling, picture statistics and so on. It is made to quickly retrieve the data held in pixels and supports a wide-ranging file formats.

In this stage, we have tried to extract the CAPTCHA string in midst of all the security features embedded in the CAPTCHA. The primary goal of this stage is to eliminate all the security features of the CAPTCHAs. For this, we need to carry out different kinds of elimination operations for different types of CAPTCHAs. This is because different CAPTCHAs have different kinds of security features. For instance, we like to eliminate multi-color noise in CAPTCHAs or transform different colored letters into a single color to improve classification in the recognition step (discussed below). This step is not always necessary since some CAPTCHAs do not include noise components. Some CAPTCHAs are created in only black and white format, where image binarization [71] is not needed. That is why it is not needed for all CAPTCHA mechanisms as some are already in binary color. To summarize, each CAPTCHA needs different combinations of elimination techniques in the pre-processing stage to eliminate the security features from the CAPTCHA before being fed to the OCR stage for segmentation and recognition (explained in Section 5.1.2).

As mentioned earlier, there are many security feature elimination techniques available such as:

- Image Binarization
- Blurring
- Erosion
- Dilation
- Convert Grayscale

Next, we briefly discuss about different security feature elimination techniques that we have used.

Image Binarization: Image Binarization is a method for changing multi-color images into monochrome ones [71]. Monochrome means only two colors, basically black and white. The main idea is to go over each pixel in the picture and convert those whose RGB values are higher than a threshold value to black and those whose values are lower than the threshold to white. It is used as a text finding approach since it often separates text regions from background areas. Binarization is crucial to document processing since the effectiveness of character recognition in the OCR stage depends heavily on how well binarization performs.

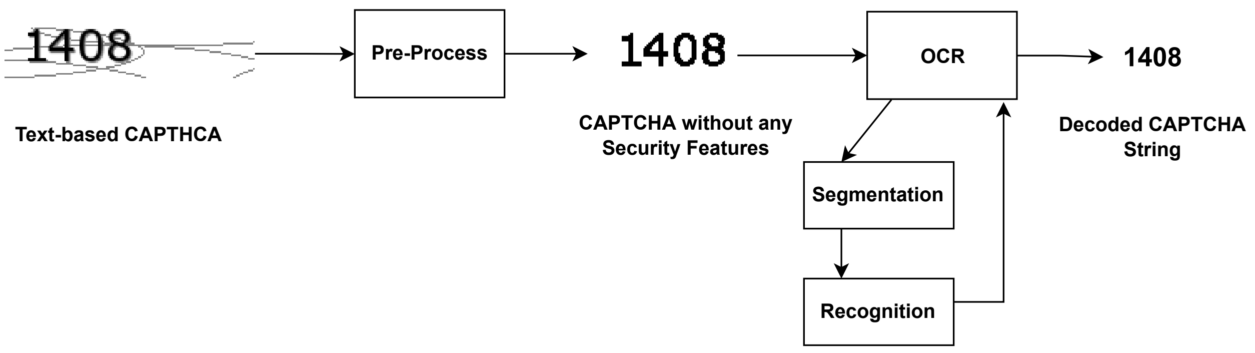


Figure 2: Text CAPTCHA attack mechanisms

In breaking text CAPTCHAs, image binarization is significant to CAPTCHAs with more than two colors in foreground and background. It can be beneficial for recognition because the noises and the target text can be distinguished with this binarization, an example is shown in Figure 3 for more clarity. With binarization, all the text in a CAPTCHA gets converted into single color where all the noises as well as the background get converted into another color.

In our work, we implemented image binarization with OpenCV-Python [72] with the following *cv2.threshold()* methods:

- **Simple Thresholding:** An uniform threshold value is used for every pixel. The pixels value is adjusted to 0 if it is less than the threshold or set to the maximum value if it is greater than the threshold. There are five kinds of simple threshold mechanisms available in OpenCV:
  (a) *THRESH_BINARY*
  (b) *THRESH_BINARY_INV*
  (c) *THRESH_TRUNC*
  (d) *THRESH_TOZERO*
  (e) *THRESH_TOZERO_INV*
- **Adaptive Thresholding:** The algorithm calculates the threshold for that particular pixel, based on a tiny area surrounding the pixel. As a consequence, we get various thresholds for various areas inside the same picture, which produces superior outcomes for photos with changing brightness. Two kinds of simple thresholding mechanisms are available in OpenCV:
  (a) *ADAPTIVE_THRESH_MEAN_C*
  (b) *ADAPTIVE_THRESH_GAUSSIAN_C:*
- **Otsu's Method (Otsu):** Otsu's approach [73], named after Nobuyuki Otsu, is used in computer vision and image processing to do automatic image thresholding. Otsu's method is an adaptive thresholding technique for image processing binarization. It determines the ideal threshold value for an image by running through all substantial threshold values [73, 74, 75]. There is only one OTSU method in OpenCV: *THRESH_OTSU* [76].

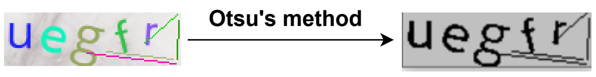


Figure 3: Binarization by Otsu's method

Blurring: Blurring can be defined as making the image less sharp or more smooth [77]. The image is blurred by convolving it with the kernel of a low-pass filter [78]. It is effective for noise reduction of an image. It eliminates high-frequency materials (such as noise and edges) from an image. Therefore, some blurring of edges occurs throughout this procedure. Blurring is a crucial pre-processing step in computer vision and image processing. Lowering the amount of information in a picture may locate items of interest more quickly. In the attack process, we can think the CAPTCHA string characters as the item of interest that we want to extract from the noises.

In the pre-processing of a CAPTCHA, blurring is one of the ways to eliminate one of the most widely used security features of a CAPTCHA: *noise*. If the noise objects inside a CAPTCHA are smaller than the string objects (texts), then blurring is most effective in eliminating the noises, as blurring makes the image sharper and the smaller noise objects disappear because of blurring. An example is shown in Figure 4. However, the main CAPTCHA string characters become a bit thinner and they need to be made thicker again by Dilation (discussed later).

In our work, we have implemented Blurring with OpenCV-Python [72] which has three types of blurring [79]:

(a) **Average Blurring**: Average blurring substitutes the central element by averaging all the pixels below the kernel region. This method is fast, however, it sometimes does not preserve the image edges. An example is shown in Figure 4.

(b) **Gaussian Blurring**: Gaussian blurring is similar to average blurring. Instead of employing a simple mean, we use a weighted mean, where pixels closer to the region of the centre pixel give more *weights* to the average. Noise that nearly follows a Gaussian distribution can be removed using Gaussian blurring. As a result, a Gaussian blurred picture is less blurred but more naturally blurred than the picture with an average blur. This blurring is slow, however, it gives good results than the average blurring.

(c) **Median Blurring**: When eliminating salt-and-pepper noise [80], the median blurring technique has shown to be the most successful. Imagine taking a picture, setting it on the table, and then adding salt and pepper to it to create random noises on that image. The salt and pepper in the photograph can be eliminated using the median blur technique. Therefore, a noise background security feature consisting of salt-and-pepper noise of a CAPTCHA can be effectively eliminated by Median Blurring.

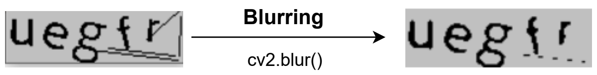


Figure 4: Noise reduction by Average Blurring

Erosion: After applying the binarization and blurring methods, most of the noises from a CAPTCHA is removed. However, there may be some little noises that can still be there after these operations. The erosion method [81] can be utlized to remove these noises by shrinking the foreground of the CAPTCHA. In other words, erosion is shrinking the foreground Pixels close to an item's border in an image, thereby "eroding" the noise away as illustrated in Figure 5.

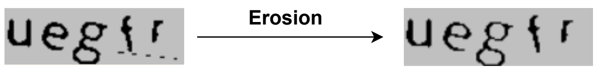


Figure 5: Noise reduction by Erosion

In order for erosion to function, a structuring element must be defined. This structuring element is then moved over the input picture from top to bottom and left to right. A foreground pixel in the input image will be kept if all pixels inside the structuring element are more significant than 0. Otherwise, 0 is assigned for the pixels.

In our work, the *OpenCV-Python* [72] library has been used to apply Blurring. Since the foreground is black and the background is whitish, we could not use the *cv2.erode()* function which works only for white foreground and black background. Therefore, in this scenario, to erode the foreground, we have to reverse it by calling the *cv2.dilate()* function, which ultimately does the erode operation in this situation.

Dilation: After applying the Erosion process, we have found that all noises have been eliminated, however, the real CAPTCHA string characters have shrunk down considerably and they are hardly recognizable by an OCR. Therefore, we need a way to thicken the characters again by applying the Dilation process. Dilation is the opposite of erosion. The foreground pixels will expand during a dilation, just as they do during an erosion (see Figure 6). Like previously, we have implemented dilation with OpenCV-Python [72].

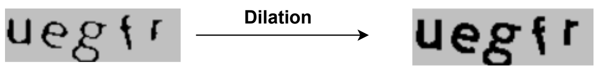


Figure 6: Character enhancement by Dilation

Convert Grayscale: Using grayscale operation to an image can simplify the image data as it will work with only one channel. This operation will also automatically reduce unnecessary data, reducing recognition time.

We can implement this operation with OpenCV-Python's [72] *cv2.cvtColor()* method [82].

The impact of different pre-processing steps is illustrated in Figure 7.

Sequence of pre-processing steps: All of our selected CAPTCHAs have been pre-processed using the concatenation of the operations described earlier. However, it is to be noted that different CAPTCHAs require different combinations and sequences of pre-processing operations, which are determined based on the specific security features of each CAPTCHA.

After applying these operations over each CAPTCHA, we have been able to make each CAPTCHA noiseless, segmentable and recognisable to the OCR, resulting in accurate determination of text CAPTCHAs.

Table 9 lists the output of applying the pre-processing steps for each type of CAPTCHA along with the combination of pre-processing operations on each CAPTCHA type, and the exact parameters utilized.

Next, we explain the different notations and parameters used in Table 9.

For Binarization:

- **T.B.(a)** denotes the **THRESH_BINARY** operation. The value of *a* is the threshold value. For instance, T.B.(180) means Thresh Binary threshold with a threshold of 180 out of 255.
- **T.B.INV(a)** denotes the **THRESH_BINARY_INVERSE** operation.
- **Otsu** : Otsu's method [76].

Similarly, for Blurring, the notations are:

- **G.B.(a,b)** denotes the **Gaussian Blur** operation where the values of *a* and *b* are the height and weight of the kernel respectively. For instance, G.B.(13,13) implies that Kernel Height is 13 and Weight is 13.
- **M.B.(a)** implies the **Median Blur** operation where *a* is the kernel size.
- **A.B.(a)** denotes the **Average Blur** operation where *a* is the same as G.B.

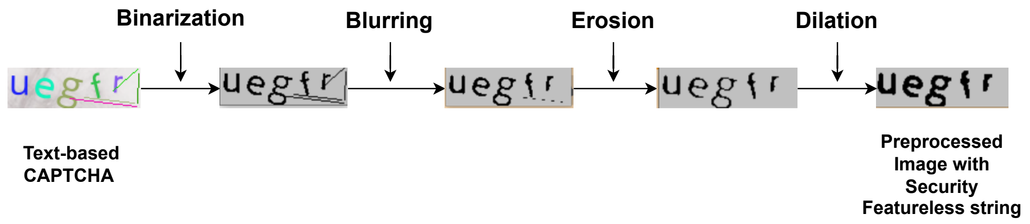
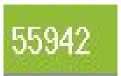
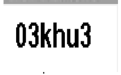
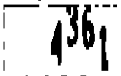
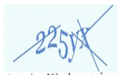
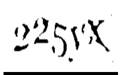
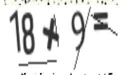
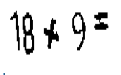
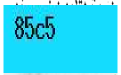
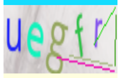
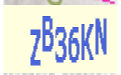
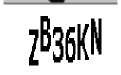
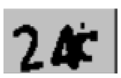
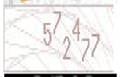
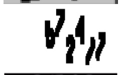
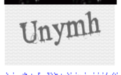
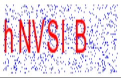


Figure 7: Pre-processing steps on a CAPTCHA

Table 9
Pre-processing of all of the selected CAPTCHAs

CAPTCHA	Binarization	Blurring	Erosion	Dilation	Gray-scale	P.P Output
	-	-	-	-	B2G	
	T.B.(180)	-	-	-	B2G	
	T.B.(195)	G.B.(13,13)	-	(2,2)	B2G	
	T.B.(205)	G.B.(3,3)	(1,1)	(3,3)	B2G	
	OTSU(128)	A.B. (1,1)	(2,1)	(3,2)	B2G	
	OTSU(128)	A.B. (1,1)	(2,1)	(3,2)	B2G	
	T.B.INV(100)	M.B.(5)	-	(2,2)	-	
	T.B.(140)	-	-	-	-	
	T.B.(100)	G.B.(13,13)	-	(2,2)	-	
	T.B.(135)	G.B.(3,3)	(5,5)	(1,1)	-	
	OTSU(128)	A.B. (1,1)	(2,1)	(3,2)	B2G	
	-	-	-	-	B2G	
	OTSU(128)	A.B. (1,1)	(2,1)	(3,2)	B2G	
	-	-	-	-	B2G	
	OTSU(128)	A.B. (1,1)	(2,1)	(3,2)	B2G	
	T.B (205)	G.B. (3,3)	(1,1)	(3,3)	B2G	
	-	-	-	-	B2G	
	OTSU(128)	A.B. (1,1)	(2,1)	(3,2)	B2G	
	OTSU(128)	A.B. (1,1)	(2,1)	(3,2)	B2G	

The notations for Erosion and Dilation are as follows:

- **(a,b)** means the kernel height is *a* and the weight is *b*.

For Gray-scale, the notations are:

- **B2G** denotes BGR2GRAY, converting the image to Gray-scale Image.

Finally, **P.P output** means Pre-Processed Output in Table 9.

5.1.2. OCR

In the OCR stage, the pre-processed CAPTCHAs go through two separate sub-stages: Segmentation and Recognition. The term "*Segmentation*" in text CAPTCHA refers to dividing a word or a sentence into a series of characters [83]. On the other hand, "*Recognition*" is used for recognizing those characters. These two are the last two traditional processes of breaking CAPTCHAs.

Generally, OCRs are not built for solving CAPTCHAs. It works accurately on noiseless standard texts on images. Therefore, in order to break a CAPTCHA with OCR, very sophisticated and rigorous pre-processing is needed for the CAPTCHAs as discussed previously. In this research, we have pre-processed the text CAPTCHAs so that any modern OCR model can accurately segment and recognize the text string. For our work, we have used Tesseract OCR [84]. Tesseract OCR is an open-source Optical character recognition model used to find and read text from images. In this stage, we fed our pre-processed CAPTCHAs to the Tessaract OCR, and it almost accurately segmented and recognized all the pre-processed CAPTCHAs.

5.1.3. Post-Processing

As we can see from Table 10, the majority of all the text CAPTCHA types have been cracked. However, the OCR could not predict the solution of some CAPTCHA types. Nevertheless, we can improve that also by finding a *mistake pattern*. We define a mistake pattern as a set of characters for which the OCR constantly gives the wrong prediction by predicting another set of characters. When we find those mistake patterns by checking with several CAPTCHAs within a CAPTCHA type, we can correct those by mapping those with the wrongly predicted characters to correct characters and replacing those wrong characters.

In this way, the attack will be more accurate. For instance, in Table 10 with Scheme ID 13, the predicted output is **ueg{r** while the actual ground truth is **uegfr**. There is a mismatch. However, we know from the CAPTCHA library that there are no characters other than lowercase letters and hence, there will never be anything like '{' on the CAPTCHA challenge. Therefore, we may map [**{ with f**]. Also, they share a similar shape. We can correct characters by finding mistake patterns after the OCR recognition stage. This will make CAPTCHA breaking more accurate.

5.1.4. Summary

A summarized illustration of all the steps used to attack the selected CAPTCHA types is presented in Figure 8.

5.2. Attack on HTML CAPTCHA

We have defined an HTML CAPTCHA [85] displaying some sort of challenges (e.g. solving a mathematical operation, writing a string and so on) which are embedded as a string inside the HTML code of the page where the CAPTCHA challenge is presented. These kinds of CAPTCHAs are implemented using CSS and JavaScript, where CSS is used for the front-end and JavaScript is used for programming of the CAPTCHAs. An example of an HTML CAPTCHA is presented in Figure 9 collected from the result publishing page of the Ministry of Education of Bangladesh [86]. The main difference between an HTML CAPTCHA and a text CAPTCHA is that an HTML CAPTCHA is just a text embedded on the website HTML code as a plain text without any security features, as shown in Figure 10, whereas text CAPTCHAs are those which are embedded on the website in the form of an image with texts and many security features as shown in Figure 1.

An HTML CAPTCHA is the weakest of all CAPTCHAs with respect to security. Such a CAPTCHA is embedded in the HTML <div> tag as a string. Therefore, an attacker can quickly scrap the web page's HTML and scrap the string from the specified div of HTML, then crack the CAPTCHA to bypass its solving mechanism. The underlying HTML code for the HTML CAPTCHA in Figure 9 is presented in Figure 10, where the red line highlights the exact string embedded in the HTML code.

5.2.1. Attack process on HTML CAPTCHA

To break an HTML CAPTCHA embedded as a string on an HTML page in an automated process, we have to scrap the web page and identify the specified portion where the CAPTCHA string is embedded. This process can be executed by **three** sub-processes:

i) **Crawl**: Crawl is the process of navigating to the target website, making an HTTP request, and then downloading the response.
ii) **Parse**: Parse is downloading the response and extracting the required data. In this article, when the CAPTCHA is in the HTML code position, Parse implies the extraction of a CAPTCHA from the respective web page.
iii) **Store**: Store is the process of storing the extracted data.

After extracting the HTML CAPTCHA by the above process, we can make an automated bot that can make hundreds of automated requests to the website. Once the CAPTCHA mechanism is compromised, there is no way for the server to detect if the requests are coming from an automated bot, thereby defeating the purpose of deploying a CAPTCHA.

6. Evaluation

We present the result of solving text CAPTCHAs using our proposed attack methods in Table 10. From that table, we can see that our proposed method can accurately solve 13 of the 19 selected text CAPTCHAs.

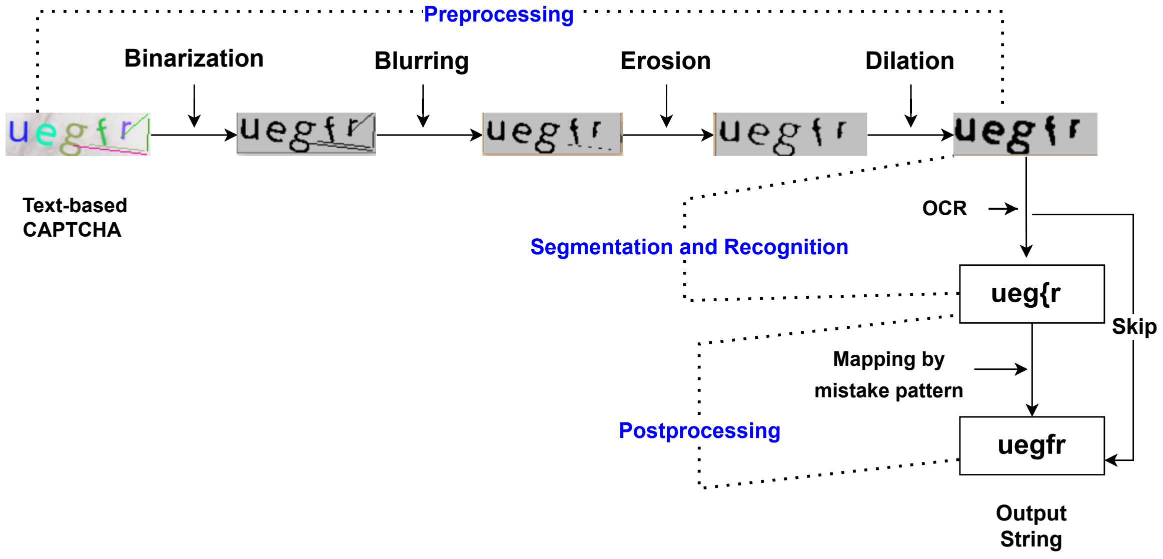
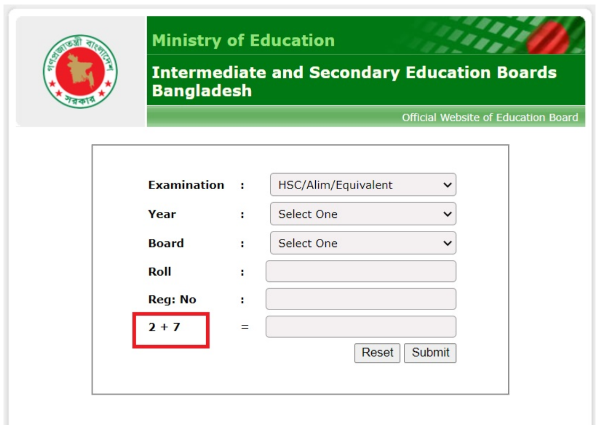
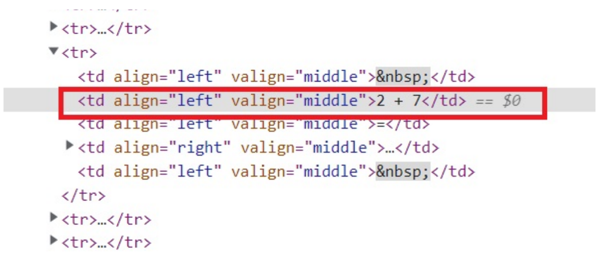


Figure 8: Steps of breaking a text CAPTCHA


Ministry of Education
Intermediate and Secondary Education Boards Bangladesh
Official Website of Education Board
Reset
Submit

Examination	:	HSC/Alim/Equivalent
Year	:	Select One
Board	:	Select One
Roll	:	
Reg: No	:	
2 + 7	=	

Figure 9: An HTML CAPTCHA example



```
▸<tr>…</tr>
▾<tr>
   <td align="left" valign="middle">&nbsp;</td>
   <td align="left" valign="middle">2 + 7</td> == $0
   <td align="left" valign="middle">=</td>
  ▸<td align="right" valign="middle">…</td>
   <td align="left" valign="middle">&nbsp;</td>
 </tr>
▸<tr>…</tr>
▸<tr>…</tr>
```

Figure 10: CAPTCHA string is embedded inside the pages HTML code

Also, we found 10 HTML CAPTCHAs, which can be accurately solved with our HTML CAPTCHA attack method described in Section 5.2.1. Therefore, as we can see by our attack methods, we could successfully solve 23 of the 29 CAPTCHA types found on Bangladeshi websites.

Here in Table 10, Scheme ID are the unique IDs of CAPTCHA types defined earlier in Table 4. Also, the P.P. CAPTCHA in the heading represents the pre-processed CAPTCHAs by Attack methods, the Predicted String is the output of our automated attack, and the ground truth is the real string value of the CAPTCHA.

Character Recognition Rate (CRR): In this research, we define Character Recognition Rate as the percentage of the Average Number of Characters detected from all the tested CAPTCHA sample ($N_s = 55$ for each CAPTCHA) sentences of a particular CAPTCHA scheme comprised of the total number of Characters, as shown in Equation 3.

$$CRR_{C_i} = \frac{\sum \frac{CDC_{C_i}}{TC_{C_i}} \times 100\%}{N_s} \tag{3}$$

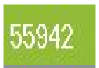
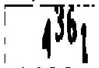
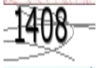
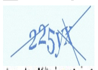
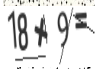
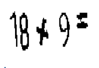
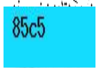
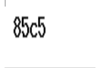
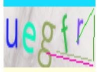
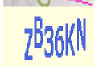
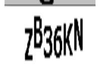
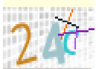
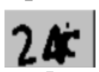
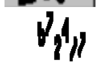
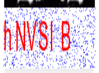
In Equation 3, C_i represents a particular CAPTCHA C belonging to a Scheme ID i, and the success rate of each CAPTCHA is denoted as CRR. On the other hand, CDC , TC, N_s stand for average of correctly detected characters, total characters of a single sample and total number of samples for each CAPTCHA respectively.

Success Rate (SR): The success rate in attacking a CAPTCHA scheme refers to the percentage of CAPTCHAs that can be correctly solved or bypassed by automated systems or algorithms. A text CAPTCHA sample is correctly solved if exactly all the characters in that CAPTCHA sample is correctly recognized. Equation 4 represents the success rate of attacking a CAPTCHA scheme.

$$SR_{C_i} = \frac{CSS}{N_s} \times 100\% \tag{4}$$

As per Equation 4, *SR* denotes the success rate, *CSS* is the number of correctly solved CAPTCHAs and N_s is the total number of CAPTCHA samples for each C_i. For our experiment we collected 55 samples of each of the CAPTCHA

Table 10
Attack Results on Bangladeshi Text CAPTCHAs

ID	Example	P.P. CAPTCHA	Predicted String	Ground Truth	Recognition Rate	Success Rate
1			55942	55942	100%	100%
2			03khu3	03khu3	100%	100%
3			SAVQ	SAVQ	100%	100%
4			36	4361	23%	0%
5			1408	1408	100%	100%
6			-	225yx	48%	0%
7			FZ4Y8	FZ4Y8	100%	100%
8			b0c10	b0c10	100%	100%
9			KUVRJ	KUVRJ	100%	100%
10			18	27	36%	0%
11			-	89478	0%	0%
12			85c5	85c5	100%	100%
13			uegfr	uegfr	100%	100%
14			ZB36KN	ZB36KN	100%	100%
15			2	24c	60%	20%
16			79447	572477	90%	13%
17			2542	2542	100%	100%
18			hNVSIB	hNVSIB	100%	100%
19			Unymh	Unymh	100%	100%

schemes and then tested the attack method on those samples to determine the success rate of each CAPTCHA scheme.

Analysis: Let us now compare our attack experiment's result with our evaluation criteria-based assumption defined in Table 7.

From Table 7 and Table 10, we can compile Table 11 (Validation of Security Level Assumption). Table 11 illustrates how accurate the assumptions have been as measured by the evaluation framework. We have compiled Table 11 by comparing the assumption level with the recognition and success rate that we have from Table 7 and Table 10 respectively. Some observations from Table 11 are discussed next:

- The CAPTCHAs predicted as Vulnerable with Scheme IDs 1, 2, 5, 8, 12, 14, 17, 18 and 19 can be completely broken/bypassed with the automated attack with a 100% recognition rate as well as with a 100% success rate.

- The CAPTCHA schemes with Scheme IDs 3, 7, 9, 13, 15 are assumed as Moderately Secure as we can see in Table 7. From Table 11 we can see that 1 of the 6 Moderately Secure CAPTCHAs with Scheme IDs 15 can be bypassed as the success rate of the automated attack is 20%. On the other hand, the rest four CAPTCHAs with Scheme IDs 3,7,9 13 are completely bypassed by the automated attack with 100% recognition and success rate.

Table 11
Validation of Security Level Assumption

Assumed Security Level	Scheme ID	Recognition Rate	Success Rate
Vulnerable	1	100%	100%
	2	100%	100%
	5	100%	100%
	8	100%	100%
	12	100%	100%
	14	100%	100%
	17	100%	100%
	18	100%	100%
	19	100%	100%
Moderately Secure	3	100%	100%
	7	100%	100%
	9	100%	100%
	13	100%	100%
	15	60%	20%
Secure	4	23%	0%
	6	48%	0%
	10	36%	0%
	11	0%	0%
	16	90%	13%

- Finally, as evident in Table 7, 5 CAPTCHAs are predicted as Secure by the evaluation framework whereas CAPTCHAs with Scheme IDs 4, 6, 11 and 16 cannot be bypassed by the automated attacks as we can see in Table 11.

From Table 11, we can see that all 9 predicted Vulnerable CAPTCHAs by the evaluation framework have been broken by the attacks with 100% recognition and success rates. Therefore, the effectiveness of predicting vulnerable CAPTCHAs is $(9/9) * 100\% = 100\%$. Also, 4 of 5 predicted Moderately Secure CAPTCHAs by the evaluation framework have been broken/bypassed. Therefore, the effectiveness of the prediction is $(4/5) * 100\% = 80\%$. Then, none (0 out of 5) of predicted Secured CAPTCHAs can be bypassed with 100% Recognition and Success Rate with automated attacks. Therefore, the effectiveness of the prediction is $(5/5) * 100\% = 100\%$

Also, using this framework we predicted that out of 1045 text CAPTCHAs (spanning 19 schemes), 495 (across 9 schemes) would be Vulnerable, 275 (across 5 schemes) would be Moderately Secure, and the remaining 275 (across 5 schemes) would be Secure. Upon executing the attack, our predictions were largely confirmed: all 495 CAPTCHAs identified as Vulnerable proved to be so, and 220 out of the 275 predicted as Moderately Secure were indeed Moderately Secure. The remaining 330 CAPTCHAs (across 6 schemes) were secure and resistant to the attacks, exceeding our initial prediction of 275 Secure CAPTCHAs (across 5 schemes). The result is elaborated in Table 12.

To underline the effectiveness of the evaluation framework, we can use metrics such as Precision, Recall, Accuracy and F1 score. These are commonly used in information retrieval and statistical classification tasks. In addition,

Table 12
Overall Comparison

CAPTCHAs	Predicted by Framework	Actual
Vulnerable	495 (9 Schemes)	495 (9 Schemes)
Moderately Secure	275 (5 Schemes)	220 (4 Schemes)
Secure	275 (5 Schemes)	330 (6 Schemes)
Total	1045 (19 Schemes)	1045 (19 Schemes)

Table 13
Effectiveness

Precision	Recall	Accuracy	F1 Score	P-Value
0.94	0.94	0.96	0.94	0.003

p-value is used for hypothesis testing. Using the results presented in Table 12, we can calculate the Micro-average Precision, Recall, Accuracy, F1 Score and p-value of the evaluation framework. For the p-value analysis, As per Chi-Squrae test of independence [87] we have proposed the following null hypothesis and an alternative hypothesis:

- **Null Hypothesis:** The actual frequencies for 'Vulnerable', 'Moderately Secure', and 'Secure' does not follow the predicted frequencies. In other words, there is a significant difference between our actual and predicted data.

- **Alternate Hypothesis:** The actual frequencies do follow the predicted frequencies. There is a significant association between our actual and predicted data.

The calculated Micro-average Precision, Recall, Accuracy, F1 Score and p-value are presented in Table 13 from where we can see that the Precision is 0.94, Recall is 0.94, Accuracy is 0.96 and the F1 Score is 0.94. In addition, the p-value is 0.003. If the p-value is less than the significance level of 5% or 0.05, we reject the null hypothesis and conclude that there is enough evidence to suggest an association between the predicted and actual data. Otherwise, we fail to reject the null hypothesis and conclude that there is enough evidence to suggest a difference between the predicted and actual data. In this case, as $p = 0.003 < 0.05$, it implies that the null hypothesis is rejected and there is an association between the predicted and actual data. The detailed calculations for the F1 score and p-value are presented in Appendix A and Appendix B.

Now, we can come to a conclusion that our defined evaluation criteria framework consisting of security and vulnerability features are accurate with a few exceptions. Also, we can see that 13 of 19 text CAPTCHAs can be successfully broken with our proposed Attack Method.

We had another 10 HTML CAPTCHAs to evaluate. As we discussed earlier, HTML CAPTCHAs are the weakest CAPTCHAs of all. Our proposed attack method solved all of them in an automated process described in Section 5.2.

To summarize, we have shown that 23 out of 29 selected CAPTCHA schemes utilized on Bangladeshi websites can be broken with our proposed automated attack method. Therefore, we can conclude that most Bangladeshi websites are vulnerable to automated bot attacks which could frequently bypass the vulnerable CAPTCHAs embedded on Bangladeshi websites.

7. Recommendation

Based on our research, we have some recommendations on the security aspect of CAPTCHAs for Bangladeshi websites. The recommendations are described below:

- It has been quite surprising to find out that 193 of 238 Bangladeshi websites do not use any CAPTCHA or automated bad bot prevention mechanism. This may lead to various bad bot traffic on those websites. Therefore, a suitable bot prevention mechanism is recommended for all Bangladeshi websites.

- Among 45 Bangladeshi websites that use some form of CAPTCHAs, 15 of them only use reCAPTCHA. reCAPTCHA is the most secure CAPTCHA developed by Google [88]. Although reCAPTCHA is not free, it is recommended to use reCAPTCHA on websites if it has enough revenue to pay for reCAPTCHA.

- An HTML CAPTCHA consisting of a string of characters without any security features inside the HTML code is the weakest CAPTCHA to defend against bad bots. Nevertheless, 10 Bangladeshi websites use this kind of CAPTCHA. Based on our analysis they are quite easy to break and hence, any HTML CAPTCHA is firmly not recommended.

- Text CAPTCHAs can be a free alternative to reCAPTCHA, which is less secure than reCAPTCHA. However, it can be made secure with a proper combination of security features and evaluation. In this work, we find some text CAPTCHAs that we cannot solve with our automated attack methods. Therefore, we are recommending a combination of security features present in those secure CAPTCHAs to make them secure from automated bot attacks. The recommended features are:
  - Overlapping characters
  - Textured background
  - Noisy background
  - Same colored foreground and noise
  - Rotation
  - Multi structure foreground
  - Hollow scheme

- In addition the above security features, we have some recommendations on building a secure CAPTCHA, such as:
  - It is recommended not to build any fixed length CAPTCHA. Instead, randomizing the length of the characters within the CAPTCHA is advisable [5].
  - Multiple fonts and font size usages are recommended.
  - Using small samples of CAPTCHAs with more security features is recommended in order to prevent Machine Learning based training on the CAPTCHA.
  - It might be beneficial to use random length lines and keep the lines on the CAPTCHA sentence to defend against any segmentation attack.
  - Overlapping the characters within a CAPTCHA as much as possible without hampering the human usability or human accuracy is also advisable.

- Lastly, even a secure CAPTCHA might be broken one day. It is always recommended to have a backup plan for that. If an automated bot somehow bypasses a CAPTCHA, a new backup CAPTCHA or other bad bot prevention mechanism should be implemented.

8. Conclusion

In this article, we have explored the security aspect of CAPTCHA, a widely-used bot management technique on the web. CAPTCHA is the most used countermeasure against malicious bot attacks for the web. However, if the CAPTCHA is not secure enough, attackers using automated bot might bypass or break the deployed CAPTCHA on the website, thereby nullifying the countermeasure with the website being exposed to any automated bot attack again.

As a result, evaluating the security aspects of CAPTCHAs is very crucial.

Towards this aim, we have explored the security aspects of CAPTCHAs deployed on Bangladeshi websites from various sources and also checked whether any alternative visible CAPTCHAs are available or not to counter the automated bot-based malicious activities.

To fulfill our objective, we have developed an evaluation framework in two steps: i) the creation of evaluation criteria and ii) security level assumption based on the created evaluation criteria. The evaluation criteria consist of security assessments and vulnerability assessments, acting as the base to evaluate the strengths and weaknesses for each individual CAPTCHA. We have also classified the security level of the selected Bangladeshi CAPTCHAs into three categories which are Vulnerable, Moderately Secure and Secure.

Following that, we have defined the generalized attack methods on the CAPTCHAs. We have experimented with the attack methods by trying to break the selected CAPTCHAs that have been evaluated by our evaluation framework. Then we have evaluated the results of the attacks to verify the evaluation framework.

Based on our analysis, we have found 23 out of 29 selected CAPTCHAs deployed on Bangladeshi websites to be vulnerable. Particularly, HTML CAPTCHAs are found to be more vulnerable than text CAPTCHAs. In order to mitigate this problem, we have also presented a number of recommendations which could help Bangladeshi websites to secure their deployed CAPTCHAs, thereby increasing the security against malicious automated bot attacks.

Limitations: It is to be noted that the 19 CAPTCHA schemes we identified do not necessarily represent all possible schemes on all Bangladeshi websites. Also, our observed one-to-one association, where a single scheme is linked to a specific website, does not imply exclusivity. The same scheme might indeed be utilized on other websites that we did not consider. We presented those websites that we found in our systematic study as presented in Section 4.1.2. Nevertheless, it must be highlighted that the presented evaluation framework (Section 4.1.3) represents a number of steps which could be utilized to attack CAPTCHAs belonging to a new CAPTCHA scheme as well. For example, if a new scheme is found, the process of vulnerability prediction (Section 4.1.3), testing based on our attack model (Section 5) and evaluation (Section 6) would remain the same.

References

[1] A. Petrosyan, Number of internet and social media users worldwide as of april 2023, [Online]. Available at: https://www.statista.com/statistics/617136/digital-population-worldwide/, 2023. (Accessed: 12 August 2023).

[2] Imperva, Credential stuffing, [Online]. Available at: https://www.imperva.com/learn/application-security/credential-stuffing/, 2020. (Accessed: 12 August 2023).

[3] H. Jonker, B. Krumnow, G. Vlot, Fingerprint surface-based detection of web bot detectors, in: Computer Security–ESORICS 2019: 24th European Symposium on Research in Computer Security, Luxembourg, September 23–27, 2019, Proceedings, Part II 24, Springer, 2019, pp. 586–605.

[4] C. Iliou, T. Kostoulas, T. Tsikrika, V. Katos, S. Vrochidis, Y. Kompatsiaris, Towards a framework for detecting advanced web bots, in: Proceedings of the 14th international conference on availability, reliability and security, 2019, pp. 1–10.

[5] E. Bursztein, M. Martin, J. Mitchell, Text-based captcha strengths and weaknesses, in: Proceedings of the 18th ACM conference on Computer and communications security, 2011, pp. 125–138.

[6] J. Yan, A. S. El Ahmad, Captcha security: a case study, IEEE Security & Privacy 7 (2009) 22–28.

[7] C. A. Fidas, A. G. Voyiatzis, N. M. Avouris, On the necessity of user-friendly captcha, in: Proceedings of the SIGCHI conference on human factors in computing systems, 2011, pp. 2623–2626.

[8] S. Lu, K. Huang, T. Meraj, H. T. Rauf, A novel captcha solver framework using deep skipping convolutional neural networks, PeerJ Computer Science 8 (2022) e879.

[9] J. Yan, A. S. El Ahmad, A low-cost attack on a microsoft captcha, in: Proceedings of the 15th ACM conference on Computer and communications security, 2008, pp. 543–554.

[10] P. Golle, Machine learning attacks against the asirra captcha, in: Proceedings of the 15th ACM conference on Computer and communications security, 2008, pp. 535–542.

[11] H. Zafarullah, H. Haque, Policies, instrumentalities, compliance and control: combatting money laundering in bangladesh, Journal of Money Laundering Control 26 (2023) 189–204.

[12] T. Bukth, S. S. Huda, The soft threat: The story of the bangladesh bank reserve heist, in: SAGE Business Cases, SAGE Publications: SAGE Business Cases Originals, 2017.

[13] J. A. Hill, SWIFT Bank Heists and Article 4A, Journal of Consumer and Commercial Law 22 (2018).

[14] M. Moniruzzaman, F. Chowdhury, M. S. Ferdous, Measuring vulnerabilities of bangladeshi websites, in: International Conference on Electrical, Computer and Communication Engineering (ECCE), 2019.

[15] A. A. Chowdhury, F. Chowdhury, M. S. Ferdous, A study of password security factors among bangladeshi government websites, in: 23rd International Conference on Computer and Information Technology (ICCIT), IEEE, 2020, p. 7.

[16] F. Alizadeh, A. Mniestri, G. Stevens, Does anyone dream of invisible AI? a critique of the making invisible of ai policing, in: Nordic Human-Computer Interaction Conference, 2022, pp. 1–6.

[17] D. Brodić, A. Amelio, The captcha: Perspectives and challenges: Perspectives and challenges in artificial intelligence (2019).

[18] K. Chellapilla, K. Larson, P. Simard, M. Czerwinski, Designing human friendly human interaction proofs (hips), in: Proceedings of the SIGCHI conference on Human factors in computing systems, 2005, pp. 711–720.

[19] J. Cui, L. Wang, J. Mei, D. Zhang, X. Wang, Y. Peng, W. Zhang, Captcha design based on moving object recognition problem, in: The 3rd International Conference on Information Sciences and Interaction Sciences, IEEE, 2010, pp. 158–162.

[20] M. Mohamed, S. Gao, N. Saxena, C. Zhang, Dynamic cognitive game captcha usability and detection of streaming-based farming, 2014.

[21] V. Fanelle, S. Karimi, A. Shah, B. Subramanian, S. Das, Blind and human: Exploring more usable audio captcha designs, in: Proceedings of the Sixteenth USENIX Conference on Usable Privacy and Security, 2020, pp. 111–125.

[22] G. Ye, Z. Tang, D. Fang, Z. Zhu, Y. Feng, P. Xu, X. Chen, Z. Wang, Yet another text captcha solver: A generative adversarial network based approach, in: Proceedings of the 2018 ACM SIGSAC conference on computer and communications security, 2018, pp. 332–348.

[23] L. Eikvil, Optical Character Recognition, Research Report, Norsk Regnesentral, Blindern. (1993).

[24] G. Mori, J. Malik, Recognizing objects in adversarial clutter: breaking a visual captcha, in: 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.,

volume 1, 2003, pp. I–I. doi:10.1109/CVPR.2003.1211347.

[25] K. Kaur, S. Behal, Captcha and its techniques: a review, International Journal of Computer Science and Information Technologies 5 (2014) 6341–6344.

[26] W. K. A. Hasan, A survey of current research on captcha, Int. J. Comput. Sci. Eng. Surv.(IJCSES) 7 (2016) 141–157.

[27] B. M. Powell, A. C. Day, R. Singh, M. Vatsa, A. Noore, Image-based face detection captcha for improved security, International Journal of Multimedia Intelligence and Security 1 (2010) 269–284.

[28] The CAPTCHA Project. Gimpy, [Online]. Available at: http://www.captcha.net/captchas/pix/, 2003. (Accessed: 9 November 2022).

[29] The CAPTCHA Project. Bongo, [Online]. Available at: http://www.captcha.net/captchas/bongo/, 2003. (Accessed: 9 November 2022).

[30] M. Guerar, L. Verderame, M. Migliardi, F. Palmieri, A. Merlo, Gotta captcha'em all: a survey of 20 years of the human-or-computer dilemma, ACM Computing Surveys (CSUR) 54 (2021) 1–33.

[31] P. Loshin, How to identify and evaluate cybersecurity frameworks, [Online]. Available at: https://www.techtarget.com/searchsecurity/feature/How-to-identify-and-evaluate-cybersecurity-frameworks, 2019. (Accessed: 22 November 2022).

[32] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, V. Shet, Multi-digit number recognition from street view imagery using deep convolutional neural networks, arXiv preprint arXiv:1312.6082 (2013).

[33] R. Smith, An overview of the tesseract ocr engine, in: Ninth international conference on document analysis and recognition (ICDAR 2007), volume 2, IEEE, 2007, pp. 629–633.

[34] C. Patel, A. Patel, D. Patel, Optical character recognition by open source ocr tool tesseract: A case study, International Journal of Computer Applications 55 (2012) 50–56.

[35] L. V. Ahn, M. Blum, N. J. Hopper, J. Langford, CAPTCHA: Using Hard AI Problems for Security, in: International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2003, pp. 294–311.

[36] G. Mori, J. Malik, Recognizing objects in adversarial clutter: breaking a visual captcha, in: 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings., volume 1, 2003, pp. I–I. doi:10.1109/CVPR.2003.1211347.

[37] G. Moy, N. Jones, C. Harkless, R. Potter, Distortion estimation techniques in solving visual captchas, in: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004., volume 2, IEEE, 2004, pp. II–II.

[38] K. Chellapilla, K. Larson, P. Y. Simard, M. Czerwinski, Computers beat humans at single character recognition in reading based human interaction proofs (hips)., in: CEAS, 2005.

[39] J. Yan, A. S. El Ahmad, Breaking visual captchas with naive pattern recognition algorithms, in: Twenty-Third annual computer security applications conference (ACSAC 2007), IEEE, 2007, pp. 279–291.

[40] C. Chesnut, Using ai to beat captcha and post comment spam, Online http://www. brains-nbrawn. com/aiCaptcha (2005).

[41] A. S. El Ahmad, J. Yan, L. Marshall, The robustness of a new captcha, in: Proceedings of the third european workshop on system security, 2010, pp. 36–41.

[42] J. Yan, E. Salah, A. Ahmad, Is cheap labour behind the scene?-low-cost automated attacks on yahoo captchas, School of Computing Science Technical Report Series (2008).

[43] Z. Wang, P. Shi, Captcha recognition method based on cnn with focal loss, Complexity 2021 (2021).

[44] G. Ye, Z. Tang, D. Fang, Z. Zhu, Y. Feng, P. Xu, X. Chen, J. Han, Z. Wang, Using generative adversarial networks to break and protect text captchas, ACM Transactions on Privacy and Security (TOPS) 23 (2020) 1–29.

[45] N. Rathoura, V. Bhatiab, Recognition method of text captcha using correlation and principle component analysis, International Journal of Control Theory and Applications 9 (2018) 46.

[46] F.-L. Du, J.-X. Li, Z. Yang, P. Chen, B. Wang, J. Zhang, Captcha recognition based on faster r-cnn, in: International Conference on Intelligent Computing, Springer, 2017, pp. 597–605.

[47] O. Bostik, K. Horak, L. Kratochvila, T. Zemcik, S. Bilik, Semi-supervised deep learning approach to break common captchas, Neural Computing and Applications 33 (2021) 13333–13343.

[48] E. Bursztein, J. Aigrain, A. Moscicki, J. C. Mitchell, The end is nigh: Generic solving of text-based {CAPTCHAs}, in: 8th USENIX Workshop on Offensive Technologies (WOOT 14), 2014.

[49] P. UmaMaheswari, S. Ezhilarasi, P. Harish, B. Gowrishankar, S. Sanjiv, Designing a text-based captcha breaker and solver by using deep learning techniques, in: 2020 IEEE International Conference on Advances and Developments in Electrical and Electronics Engineering (ICADEE), 2020, pp. 1–6. doi:10.1109/ICADEE51157.2020.9368949.

[50] Y. W. Chow, W. Susilo, Text-based CAPTCHAs over the years, in: IOP Conference Series: Materials Science and Engineering, volume 273, IOP Publishing, 2017, p. 012001.

[51] T. Whalen, T. Meunier, M. Kodali, A. Davidson, M. Fayed, A. Faz-Hernández, W. Ladd, D. Maram, N. Sullivan, B. C. Wolters, et al., Let the right one in: Attestation as a usable {CAPTCHA} alternative, in: Eighteenth Symposium on Usable Privacy and Security (SOUPS 2022), 2022, pp. 599–612.

[52] A. Foley, Biometric alternatives to captcha: Exploring accessible interface options (2012). Masters Dissertation. Technological University Dublin, 2012.

[53] Z. Al-Qudah, B. Al-Duwairi, O. Al-Khaleel, Ddos protection as a service: hiding behind the giants, International Journal of Computational Science and Engineering 9 (2014) 292–300.

[54] A. Thomas, K. Punera, L. Kennedy, B. Tseng, Y. Chang, Framework for evaluation of text captchas, in: Proceedings of the 22nd International Conference on World Wide Web, 2013, pp. 159–160.

[55] Y. Zhang, H. Gao, G. Pei, S. Luo, G. Chang, N. Cheng, A survey of research on captcha designing and breaking techniques, in: 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), IEEE, 2019, pp. 75–84.

[56] B. Boyter, All About CAPTCHA's: Decoding CAPTCHA's for Fun and Profit, Leanpub, 2015.

[57] Invisible recaptcha & google developers, [Online]. Available at: https://developers.google.com/recaptcha/docs/invisible, 2022. (Accessed: 12 August 2023).

[58] Googlebot, [Online]. Available at: https://developers.google.com/search/docs/advanced/crawling/googlebot, 1998. (Accessed: 12 August 2023).

[59] National portal bangladesh national information, People's republic of bangladesh, [Online]. Available at: https://bangladesh.gov.bd/index.php, 2022. (Accessed: 9 November 2022).

[60] Govt. services bangladesh (sebakunjo), [Online]. Available at: http://services.portal.gov.bd/, 2014. (Accessed: 15 September 2022).

[61] Open .trends. Semrush, Most visited retail websites in bangladesh 2022, 2008. URL: https://www.semrush.com/trending-websites/bd/retail, (Accessed: 22 November 2022).

[62] Website traffic - check and analyze any website | similarweb, 2007. URL: https://www.similarweb.com/, (Accessed: 22 November 2022).

[63] Invisible recaptcha, [Online]. Available at: https://developers.google.com/recaptcha/docs/invisible, 2017. (Accessed: 15 September 2022).

[64] M. Guerar, A. Merlo, M. Migliardi, F. Palmieri, Invisible cappcha: A usable mechanism to distinguish between malware and humans on the mobile iot, computers & security 78 (2018) 255–266.

[65] How to test invisible recaptcha, [Online]. Available at: https://www.tectite.com/fmhowto/test-invisible-recaptcha.php?WWWTECTITE=p32j2na5otc4rmtbmfsmf9rci6, 2017. (Accessed: 9 November 2022).

[66] J. Dougherty, R. Kohavi, M. Sahami, Supervised and unsupervised discretization of continuous features, in: Machine learning proceedings 1995, Elsevier, 1995, pp. 194–202.

[67] Y.-W. Chow, W. Susilo, P. Thorncharoensri, Captcha design and security issues, Advances in Cyber Security: Principles, Techniques, and Applications (2019) 69–92.

[68] Tesseract documentation, [Online]. Available at: https://tesseract-ocr.github.io/, 2005. (Accessed: 15 September 2022).

[69] G. Bradski, A. Kaehler, Opencv, Dr. Dobb's journal of software tools 3 (2000) 120.

[70] Pillow, [Online]. Available at: https://python-pillow.org/, 2011-2021. (Accessed: 12 August 2023).

[71] J. Sauvola, M. Pietikäinen, Adaptive document image binarization, Pattern recognition 33 (2000) 225–236.

[72] OpenCV-Python Tutorials, [Online]. Available at: https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html, 2022. (Accessed: 15 September 2022).

[73] N. Otsu, A threshold selection method from gray-level histograms, IEEE Transactions on Systems, Man, and Cybernetics 9 (1979) 62–66.

[74] X. C. Yuan, L. S. Wu, Q. Peng, An improved Otsu method using the weighted object variance for defect detection, Applied surface science 349 (2015) 472–484.

[75] J. Yousefi, Image binarization using otsu thresholding algorithm, Ontario, Canada: University of Guelph 10 (2011).

[76] Otsu's binarization, [Online]. Available at: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html, 2022. (Accessed: 15 January 2023).

[77] R. Szeliski, Computer vision: algorithms and applications, Springer Nature, 2022.

[78] Blurring images, Blurring Images – Image Processing with Python. Available from: https://datacarpentry.org/image-processing/06-blurring/, 2022. (Accessed: 15 September 2022).

[79] Smoothing images, OpenCV. Available from: https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html, 2008. (Accessed: 15 September 2022).

[80] R. H. Chan, C.-W. Ho, M. Nikolova, Salt-and-pepper noise removal by median-type noise detectors and detail-preserving regularization, IEEE Transactions on image processing 14 (2005) 1479–1485.

[81] A. Rosebrock, Opencv morphological operations, [Online]. Available at: https://pyimagesearch.com/2021/04/28/opencv-morphological-operations/, 2021. (Accessed: 15 September 2022).

[82] Python opencv: cv2.cvtcolor() method, [Online]. Available at: https://www.geeksforgeeks.org/python-opencv-cv2-cvtcolor-method/, 2019. (Accessed: 15 September 2022).

[83] S. Tingre, D. Mukhopadhyay, An approach for segmentation of characters in captcha, in: International Conference on Computing, Communication & Automation, 2015, pp. 1054–1059. doi:10.1109/CCAA.2015.7148562.

[84] R. Smith, An overview of the tesseract ocr engine, in: Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), volume 2, 2007, pp. 629–633. doi:10.1109/ICDAR.2007.4376991.

[85] Bangladesh high commission, london visa form, [Online]. Available at: https://bhclondon.org.uk/visa-appointment, 2023. (Accessed: 23 September 2023).

[86] Ministry of education, [Online]. Available at: http://www.educationboardresults.gov.bd/, 2008. (Accessed: 22 November 2022).

[87] W. Foundation, Chi-squared test, Avialable from: https://en.wikipedia.org/wiki/Chi-squared_test, 2023. (Accessed: 29 December 2023).

[88] N. Tanthavech, A. Nimkoompai, Captcha: Impact of website security on user experience, in: Proceedings of the 2019 4th International Conference on Intelligent Information Technology, 2019, pp. 37–41.

A. Calculation of F1 Score

At first, let us define the terms that will be used to calculate the F1 score:

- **True Positives (TP)**: CAPTCHAs correctly identified as entity X.
- **False Positives (FP)**: CAPTCHAs incorrectly identified as entity X.
- **True Negatives (TN)**: CAPTCHAs correctly identified as not entity X.
- **False Negatives (FN)**: CAPTCHAs incorrectly identified as not entity X.

Here, the entity X is Secure, Moderately Secure and Vulnerable CAPTCHAs for their respective category calculations.

Now, we can calculate the metrics as follows:

- **Precision** (also called Positive Predictive Value) is the fraction of relevant instances among the retrieved instances. It can be calculated as:

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

- **Recall** (also known as Sensitivity, Hit Rate, or True Positive Rate) is the fraction of the total number of relevant instances that were actually retrieved. It can be calculated as:

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

- **Accuracy** is the measure of all the correctly identified cases can be calculated as:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{7}$$

- **F1 Score** is the harmonic mean of Precision and recall and tries to find the balance between precision and recall. It can be calculated as:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{8}$$

We can calculate these values for each of the categories (Vulnerable, Moderately Secure, and Secure) to get a comprehensive understanding of our framework's performance. For a more overall performance measure, we have considered micro-averaging (aggregate the contributions of all classes to compute the average metric).

Now for Vulnerable CAPTCHAs from Table 14 and Equation 5, Equation 6, Equation 7 and Equation 8, we can calculate Precision as 1, Recall as 1, Accuracy is 1 and F1 Score as 1, respectively.

Then, for the Moderately Secure CAPTCHAs from Table 15 and Equation 5, Equation 6, Equation 7 and Equation 8, Precision is 0.8, Recall is 1, Accuracy is 0.98 and F1 Score is 0.89.

Finally, for the Secure CAPTCHAs from Table 16 and Equation 5, Equation 6, Equation 7 and Equation 8, Precision is 1, Recall is 0.83 Accuracy is 0.95, and F1 Score is 0.91.

To aggregate these for the whole framework, we can use micro-averaging, which aggregates the contributions of all

Table 14
Confusion Matrix for Vulnerable CAPTCHAs

	Positive	Negetive
Positive	TP1 = 495	FP1 = 0
Negative	FN1 = 0	TN1 = 550

Table 15
Confusion Matrix for Moderately Secure CAPTCHAs

	Positive	Negetive
Positive	TP2 = 220	FP2 = 55
Negative	FN2 = 0	TN2 = 770

Table 16
Confusion Matrix for Secure CAPTCHAs

	Positive	Negetive
Positive	TP3 = 275	FP3 = 0
Negative	FN3 = 55	TN3 = 715

classes to compute the average metric. This can be calculated as follows.

Micro-average Precision (MAP):

$$\begin{aligned} MAP &= \frac{TP1+TP2+TP3}{TP1+FP1+TP2+FP2+TP3+FP3} \\ &= \frac{495+220+275}{495+0+220+55+275+0} \\ &= 0.94 \end{aligned}$$

Micro-average Recall (MAR):

$$\begin{aligned} MAR &= \frac{TP1+TP2+TP3}{TP1+FN1+TP2+FN2+TP3+FN3} \\ &= \frac{495+220+275}{495+0+220+0+275+55} \\ &= 0.94 \end{aligned}$$

Micro-average Accuracy (MAA): At first we define the nominator and denominator as $MAA-NOM$ and $MAA-DENOM$ which are then used to calculate the MAA.

$$\begin{aligned} MAA-NOM =\, &TP1+TN1+TP2+TN2 \\ &+TP3+TN3 \end{aligned}$$

$$\begin{aligned} MAA-DENOM =\, &TP1+TN1+FN1+FP1 \\ &+TP2+TN2+FN2+FP2 \\ &+TP3+TN3+TP3+TN3 \end{aligned}$$

$$\begin{aligned} MAA &= \frac{MAA-NOM}{MAA-DENOM} \\ &= \frac{495+550+220+770+275+715}{3\times 1045} \\ &= 0.96 \end{aligned}$$

Micro-average F1 Score:

$$\begin{aligned} F1 &= \frac{2\times(MAP\times MAR)}{(MAP+MAR)} \\ &= \frac{2\times(0.94\times 0.94)}{0.94+0.94} \\ &= 0.94 \end{aligned}$$

Table 17
Calculation Result Table

	Precision	Recall	Accuracy	F1 Score
Vulnerable	1	1	1	1
Moderately Secure	0.8	1	0.98	0.89
Secure	1	0.83	0.95	0.91
Micro-average	0.94	0.94	0.94	0.96

From Table 17, we can see the calculated results: Precision, Recall, Accuracy and F1 Score.

B. Calculation of p-value

We have mainly used a Python code to calculate the p-value. The code can be found here. We have explained the process below.

The Chi-square test is used to determine whether there is a significant difference between the predicted frequencies and the actual frequencies in one or more categories of a contingency table. Table 18 represents the contingency table used in the p-value calculation.

Table 18
Contingency Table

	Actual	Predicted
Vulnerable	495	495
Moderately Secure	220	275
Secure	330	275

The frequency for predicted data and actual data for each cell in Table 18 is calculated using the formula:

$$Frequency = \frac{rowTotal \times columnTotal}{grandTotal} \tag{9}$$

We apply Equation 9 in Table 18 where *rowTotal* stands for the summation of all the current row values, *columnTotal* stands for all the current column values, and, *grandTotal* means the summation of all elements in the table. Next, we calculate the actual and predicted frequency values for Vulnerable, Moderately Secure and Secure in the following manner:

- Vulnerable, Actual: $(990\times 1045)/2090 = 495$
- Vulnerable, Predicted: $(990\times 1045)/2090 = 495$
- Moderately Secure, Actual: $(495\times 1045)/2090 = 247.5$

- Moderately Secure, Predicted: $(495 \times 1045)/2090 = 247.5$
- Secure, Actual: $(605 \times 1045)/2090 = 302.5$
- Secure, Predicted: $(605 \times 1045)/2090 = 302.5$

The Chi-square value (λ) is calculated using the formula:

$$\sum[(O - E)^2/E]$$

Here, O is the actual frequency and E is the predicted frequency. Next, we calculate it for each cell:

- Vulnerable, Actual: $((495 - 495)^2)/495 = 0$
- Vulnerable, Predicted: $((495 - 495)^2)/495 = 0$
- Moderately Secure, Actual: $((220 - 247.5)^2)/247.5 = 3.056$
- Moderately Secure, Predicted: $((275 - 247.5)^2)/247.5 = 3.056$
- Secure, Actual: $((330 - 302.5)^2)/302.5 = 2.5$
- Secure, Predicted: $((275 - 302.5)^2)/302.5 = 2.5$

Using these values we can calculate the Chi-square value (λ): $\lambda = 0 + 0 + 3.056 + 3.056 + 2.5 + 2.5 = 11.11$

The degree of freedom (k) is calculated using the formula:

$$k = (numberOfRows - 1) \times (numberOfColumns - 1) \quad (10)$$

In this case, it would be (3 - 1) * (2 - 1) = 2. Finally, the p-value is calculated as

$$1 - \gamma(k/2, \lambda/2)/\Gamma(k/2)$$

Here:

- γ denotes the lower incomplete gamma function
- Γ is the gamma function
- k is the degree of freedom
- λ is the Chi-square value

After calculating using this formula, our p-value is calculated as 0.003 that is less than 0.05. Therefore, it rejects the null hypothesis, which means there is a good association between actual and predicted data.