A Thesis/Project/Dissertation Report

on

URL SHORTENER

Submitted in partial fulfillment of the requirement for the award of the degree of

Bachelor of Technology

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
Name of Supervisor : Mr.Anupam Lakhanpal

Submitted By

Kashish Saxena	Shubham Satyam Dubey
19SCSE1180067	19SCSE1180059
19021180060	19021180053

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
DECEMBER, 2021
Contents

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled **"URL SHORTENER"** in partial fulfillment of the requirements for the award of the Bachelor of Technology submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of October, 2021 to November and 2021, under the supervision of Mr.Anupam Lakhanpal, Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Kashish Saxena 19SCSE1180067

Shubham Satyam Dubey 19SCSE1180059

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor Name
Mr.Anupam Lakhanpal
Designation

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of Kashish Saxena , 19SCSE1180067 ,Shubham Satyam Dubey ,19SCSE1180059 has been held on 09/12/2021 and his/her work is recommended for the award of Bachelor of Technology.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Abstract

Obviously, there is a cost to opening up a new browser window, plugging in the name of your favourite URL shortener and copy-pasting a new link in, just to share it once. It's a little annoying. But there are ways to speed this up, such as a Chrome plugin or iPhone app. Nonetheless, there are times when you should use a URL shortener – and times when you shouldn't.Use a URL shortener when you need to track the efforts of a social post or campaign. This is worthwhile when a link is sending traffic to your site, but especially when it's not. For example, maybe you send out 100 tweets linking to curated content each month. This is content which points followers to some of your favorite industry resources. Did anyone like that content? You can find out simply by checking the stats in your URL shortener. You'll be able to easily spot your top performers.

It's much simpler to share a shorter link than a long, complicated one. This especially matters in scenarios where your audience can't physically click your link or where you're promoting your website one-to-one. If you're giving a PowerPoint presentation or sharing a link directly with a prospective customer, a URL shortener can help you create a brief, memorable link that can be typed directly into a smartphone or web browser. One of the most important benefits of a URL shortener is the ability to track clicks on each link you share. Depending on the service you're using to compress web links, you can see a breakdown of visitors by demographics, such as country or gender. You can also tell which of your social channels or posts are driving the most clicks.

A basic understanding of the JavaScript programming language. A basic understanding of Node.js and Express framework. Have Postman HTTP API client installed on your system.A text editor, preferably VS Code. MongoDB database server installed on your system.

Social networking / "microblogging" sites like Twitter and Plurk only allow 140 characters per message, so a user could never include this map's URL, as it is 184 characters long. As I pseudo-mentioned above, people also use these sites to disguise affiliate links. Several times a year, but most notably on the first of April, hiding links to videos of Rick Astley also becomes popular. Just a note: the most popular Rick Roll video has quite a distinctive URL: it ends in "uuiU" so you may want to shorten that one next time you prank somebody because more and more people are starting to recognise it. Anyway. Once shortened by TinyURL, the

Introduction

URL shortening is a technique on the World Wide Web in which a Uniform Resource Locator (URL) may be made substantially shorter and still direct to the required page. This is achieved by using a redirect which links to the web page that has a long URL. For example, the URL "https://example.com/assets/category_B/subcategory_C/Foo/" can be shortened to "https://example.com/Foo", and the URL "https://en.wikipedia.org/wiki/URL_shortening" can be shortened to "https://w.wiki/U". Often the redirect domain name is shorter than the original one. A friendly URL may be desired for messaging technologies that limit the number of characters in a message (for example SMS), for reducing the amount of typing required if the reader is copying a URL from a print source, for making it easier for a person to remember, or for the intention of a permalink. In November 2009, the shortened links of the URL shortening service Bitly were accessed 2.1 billion times.[1]

Other uses of URL shortening are to "beautify" a link, track clicks, or disguise the underlying address. Although disguising of the underlying address may be desired for legitimate business or personal reasons, it is open to abuse.[2] Some URL shortening service providers have found themselves on spam blocklists, because of the use of their redirect services by sites trying to bypass those very same blocklists. Some websites prevent short, redirected URLs from being posted.

There are several reasons to use URL shortening. Often regular unshortened links may be aesthetically unpleasing. Many web developers pass descriptive attributes in the URL to represent data hierarchies, command structures, transaction paths or session information. This can result in URLs that are hundreds of characters long and that contain complex character patterns. Such URLs are difficult to memorize, type out or distribute. As a result, long URLs must be copied and pasted for reliability. Thus, short URLs may be more convenient for websites or hard copy publications (e.g. a printed magazine or a book), the latter often requiring that very long strings be broken into multiple lines (as is the case with some e-mail software or internet forums) or truncated.

On Twitter and some instant messaging services, there is a limit to the number of characters a message can carry – however, Twitter now shortens links automatically using its own URL shortening service, t.co, so there is no need to use a separate URL shortening service just to shorten URLs in a tweet.

Literature Survey

URL shortening services replace long URLs with shorter ones and subsequently redirect all requests for the shortened URL to the original long URL. In this paper we discuss and empirically analyze security and privacy risks caused by the use of URL shortening services. We empirically determine the most popular URL shortening services currently used on Twitter and analyze these with respect to malicious behavior, user tracking, ease of enumeration, and leakage of URLs to search engines. Also, we introduce a new attack scenario to enable SSL-only circumvention using SSLStrip and shortened URLs. Finally, we empirically analyze the use of URL shortening services in more than 7 million spam emails collected over the past seven years and determine the spam detection performance for the most popular services found.

Link-shortening services save space and make the manual entry of URLs less onerous. Short links are often included on printed materials so that people using mobile devices can quickly enter URLs. Although mobile transcription is a common use-case, link-shortening services generate output that is poorly suited to entry on mobile devices: links often contain numbers and capital letters that require time consuming mode switches on touch screen keyboards. With the aid of computational modeling, we identified problems with the output of a link-shortening service, *bit.ly*. Based on the results of this modeling, we hypothesized that longer links that are optimized for input on mobile keyboards would improve link entry speeds compared to shorter links that required keyboard mode switches. We conducted a human performance study that confirmed this hypothesis. Finally, we applied our method to a selection of different non-word mobile data-entry tasks. This work illustrates the need for service design to fit the constraints of the devices people use to consume services.

Link shortening services like *bit.ly*, *ow.ly* or *goo.gl* act as intermediaries between users and websites. Users provide a target link, for instance, https://www.elsevier.com/journals/international-journal-of-human-computer-studies/1071-5819/guide-for-authors. A shortening service generates a much shorter link, in this case, http://bit.ly/1OT4BPc. A mapping between these long and short links is stored by the shortening service. When a short URL is requested users are redirected to the original long URL.

Project Design

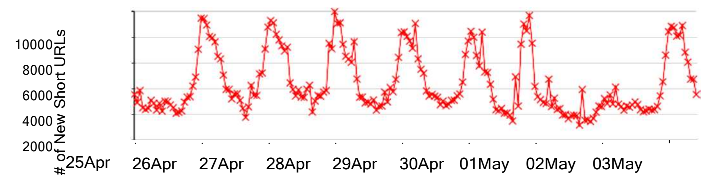
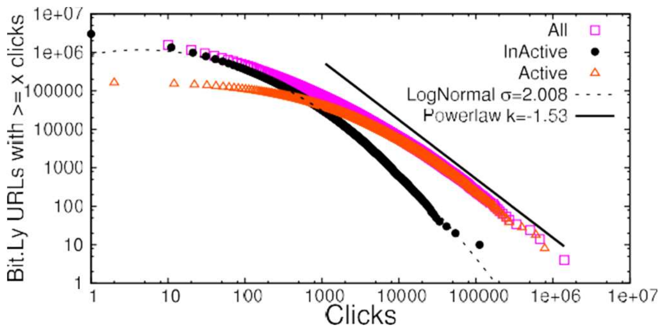
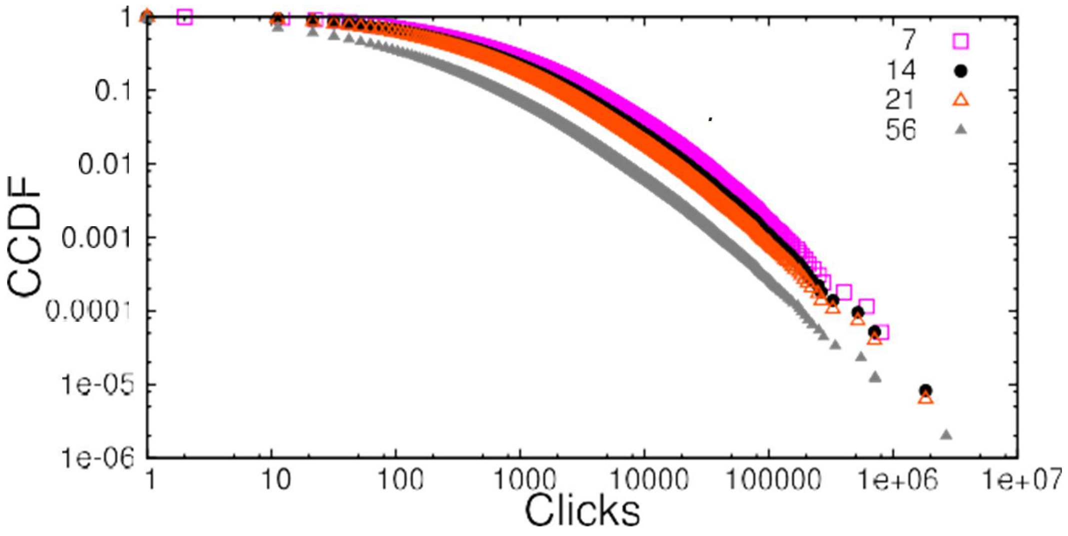


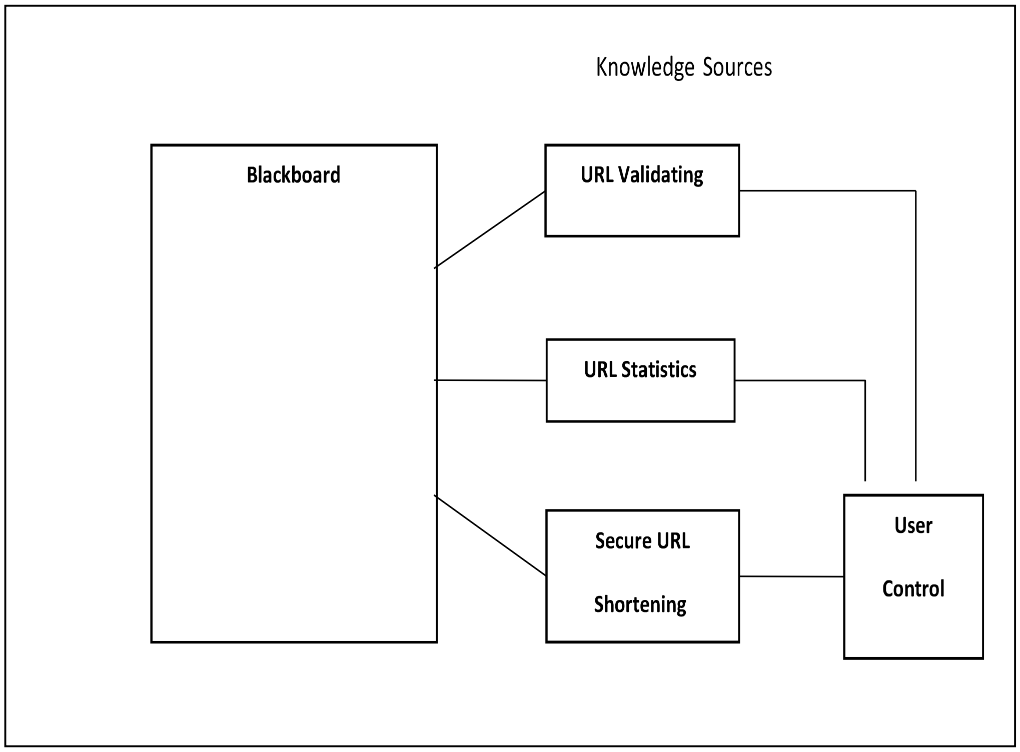
Figure 1: Number of ow.ly short URLs created as a function of time.



Popularity distributions for Active and In-Activebit.ly URLs from *twitter2* trace



Number of days a domain name is in TOP-100 during March and April 2010



The Proposed URL-Blackboard Framework

Modules Description

Each (KS) is represented as a procedure in the system and is called sequentially by the control, the output of the top KS isconsidered as an input to the next KS in the system, figure (2). EachKS gives a sub solution to the URL shortening problem to result thefinal solution; the following are the KS that are used in the proposed service:

1 URL Validation: This KS will check if the URL is long enough to be shortened or not; since not all URL need to be shortened, as described in algorithm (1). As in [2] short URLdoes not exceed 30 characters.

```
Algorithm (1): URL Validation
Input: URL
Output: Shortened URL or not
Begin
Step1: Enter URL
Step2: If length(URL) > 30 Then
Go to Algorithm (2); URL need to be
shortened
Else
Exit; Use the same URL
End If
    End
```

2. URL Statistics: This KS will count how many links are there in the URL depending on separators "/ ", as described in algorithm (2).

```
Algorithm (2): URL Statistics
Input: URL, output of Algorithm (1)
Output: Number of words (no_word) in URL, vector (index)
```

```
which has the start index for each word and vector (word) which
has all words in URL
Begin
Step1: no_word = 0
For i = 1 To length (URL)
ch = Mid(URL, i, 1)
If ch = "/" Then
no_word = no_word + 1 index(no_word) = i
End If
Next i
  Step2: print "Number of paths: " & no_wordStep3: For i = 1 To
  no_word
        words(i) = Mid(URL, index(i) , index(i + 1) - index(i))print "Word in URL
         " & word(i)
Next i
End
```

3.Secure URL Shortening: This KS will produce a secure structure of URL shortening service, as described in algorithm (3). The user enters a secret key by which the secret shortening is done. The secret key is converted to ASCII code to use it as index for URL as explained in step (2), then some characters of the original URL are selected depending on the secret key index as explained in step (3). This key will give the user evidence that the shortened URL is not attacked by hackers so he can use it safely.

```
Algorithm (3): Secure URL shortening
Input: URL, secret key (key), output of Algorithm (2)
Output: Secure URL
Begin
Step1: Enter secret key (key)
Step2: Convert secret key to ASCII code to use it as index
```

for URL

```
        For i = 1 To Len(key)
            ch = Asc((Mid(key, i, 1)))index(i) =
            ch Mod 26
        Next i
    Step 3: For i = 1 To Len(key)
            find = Mid(URL, index(i), 1)shorty =
            shorty & find
        Next i
        Print “secure URL is” & shortyStep 4:
    Produce final short URL:
            babyurl.com & word(1) & shorty
```

‘Note that word(1) will contain the name of the original website which is important in our proposed structure

Results

Property of service	**URL shortening services**	**Proposed URL shortening service**
URL shortening structure	**http://< USS_name >/<random_path>**	**http://<USS_name>/<original_host>/ <secure_path>**
String length of URL after shortening	Each USS has fixed length size of URL after shortening which is between 5 to 7 characters	Variable character length because it depends on the secret key length that is entered from the user. This variable length will decrease the possibility of guessing the short URL by the attacker
Relation between old long URL and new short URL	Produces short random string that differs completely from the original long URL	Produces short random string that isselected from the original long URL
Availability	If short URL is attacked and changed the access to the original web site will not allowed because the shortening string is random	If short URL is attacked and changed the access to the original web site will still be allowed because the proposed URL shortening structure has <**original_host**> in it
Security	Have no security	Have security by using secret keywhich is known by the user to used it as index to produce **<secure_path>**
Shortening method of URL	It is unknown for the user of the service	It is known for the user of the service so he/she can control his final short URLby using his/ her own secret key

Conclusions

The proposed secure URL shortening service will give the user more trust to use the service with security, availability, and confidentiality consideration because the proposed URL shortening structure will have the following features:

1. Putting the original host name in the shorten URL will give the user more trust to use the service.
2. The availability is considered; when the shorten URL is attacked the user can still access the original site and its original web page.
3. Using <secure_path> instead of <random_path> will give more security by using secret key.
4. By using secret key the user can check the confidently of the shortened URL so he can use it safely.
5. The user will know how the shortening is done, in previous services the shortening is unknown and is done randomly.
6. The user can choose his/her own secret key and change it when needed.
7. The secret key is converted as an index to select characters from the original URL, so the shortened URL is obtained.
8. The available URL shortening services replace the entire URL with a new short URL which has no relation between the two. In our propose service we replace the entire URL by selecting some characters from the original URL.

References

1. Gunter Ollmann; "***Security Best Practice: Host Naming & URL Conventions***", Security considerations for web-based applications, Next Generation Security Software Ltd., 2005.
2. Alexander N., Johannes B., Ulrike M.; "***Security and Privacy Implications of URL Shortening Services***", IT Security Group, RWTH Aachen University, 2011.
3. Alexander Neumann ; "***Analyzing Security Implications of URL Shortening Services***", Diploma Thesis, RWTH Aachen University - Research Group IT-Security, 2011.
4. Demetris Antoniades, Iasonas Polakis and Georgios Kontaxis;" ***we.b: The web of short URLs***", Hyderabad,India March 28 – April 1, 2011.