PROXIMAX: A Measurement Based System for Proxies Dissemination

HASEEBA HALEEM[1], DR. S. SATHISH KUMAR[2]
[1]PG Scholar, Dept of CSE, Shadan Women's College of Engineering and Technology, Hyderabad, TS, India,
[2]Associate Professor, Dept of CSE, Shadan Women's College of Engineering and Technology, Hyderabad, TS, India.

Abstract: Open interchanges over the Internet present genuine dangers to nations with severe administrations, driving them to create and send oversight instruments inside their systems. Lamentably, existing oversight circumvention frameworks don't give high accessibility assurances to their clients, as blue pencils can without much of a stretch distinguish, henceforth upset, the movement having a place with these frameworks utilizing the present propelled restriction advancements. In this paper, we propose serving the Web by Exploiting Email Tunnels (SWEET), an exceedingly accessible restriction safe framework. SWEET works by epitomizing an edited client's movement inside email messages that are extended open email administrations like Gmail and Yahoo Mail. As the activity of SWEET isn't bound to a particular email supplier, we contend that a control should square email correspondences all together keeping in mind the end goal to disturb SWEET, which is far-fetched as email comprises a critical piece of the present Internet. Through trials with a model of our framework, we locate that SWEET's execution is adequate for Web perusing. Specifically, consistent Websites are downloaded inside couple of seconds.

Keywords: Censorship Circumvention, Traffic Encapsulation, Email Communications.

I. INTRODUCTION

The Internet provides users from around the world with an environment to freely communicate, exchange ideas and information. However, free communication continues to threaten repressive regimes, as the open circulation of information and speech among their citizens can pose serious threats to their existence. Recent unrest in the middle east demonstrates that the Internet can be widely used by citizens under these regimes as a very powerful tool to spread censored news and information, inspire dissent, and organize events and protests. As a result, repressive regimes extensively monitor their citizens' access to the Internet and restrict open access to public networks [1] by using different technologies, ranging from simple IP address blocking and DNS hijacking to the more complicated and resource-intensive Deep Packet Inspection (DPI) [2], [3].With the use of censorship technologies, a number of different systems were developed to retain the openness of the Internet for the users living under repressive regimes [4]–[9]. The earliest circumvention tools are HTTP proxies [4], [9], [10] that simply intercept and manipulate client's HTTP requests, defeating IP address blocking and DNS hijacking techniques. The use of more advanced censorship technologies such as DPI [2], [11], rendered the use of HTTP proxies ineffective for circumvention. This led to the advent of more advanced tools such as Ultrasurf [5] and Psiphon [6], designed to evade content filtering. While the secircum vention tools have helped, they face several challenges. We believe that the biggest one is their lack of availability, meaning that a censor can disrupt their service frequently or even disable them completely [12]–[16].

The commoner as on is that the network traffic made by these systems can be distinguished from regular Internet traffic by censors, i.e., such systems are not unobservable. For example, the popular Tor [8] network works by having users connect to an ensemble of nodes with public IP addresses, which proxy users' traffic to the requested, censored destinations. This public knowledge about Tor's IP addresses, which is required to make Tor usable by users globally, can be and is being used by censors to block their citizens from accessing Tor[17],[18]. To improve availability, recent proposals for circumvention aim to make their traffic unobservable to the censors by pre-sharing secret with their clients [19]–[21]. Others [22]–[25] suggest to conceal circumvention by making infrastructure modification to the Internet. Nevertheless, deploying and scaling these systems is a challenging problem, as discussed in Section II.A more recent approach in designing unobservable circumvention systems is to imitate popular applications like Skype and HTTP, as suggested by Skype-Morph [26],Censor Spoofer [27], and StegoTorus [28]. However, it has recently been shown [29] that these systems' un observe ability is breakable; this is because a comprehensive imitation of today's complex protocols is sophisticated and infeasible in many cases.

A promising alternative suggested [29], [30] is to not mimic protocols, but run the actual protocols and find clever ways to tunnel the hidden content into their genuine traffic; this is the main motivation of the approach taken in this paper. In this paper, we design and implement SWEET, a censorship circumvention system that provides high availability by leveraging the openness of email communications. Fig. 1 shows the main architecture. A SWEET client, confined by a censoring ISP, tunnels its network traffic inside a series of email messages that are exchanged between herself and an email server operated by SWEET's server. The SWEET server acts as an Internet proxy [31] by proxying the encapsulated traffic to the requested blocked destinations. The SWEET client uses an oblivious, public mail provider (e.g., Gmail, Hotmail, etc.) to exchange the encapsulating emails, rendering standard email filtering mechanisms ineffective in identifying/ blocking SWEET-related emails. More specifically, to use SWEET for circumvention a client needs to create an email account with some public email provider; she also needs to obtain SWEET's client software from an out-of-bound

channel(similar to other circumvention systems). The user configures the installed SWEET software to use her public email account, which sends/receives encapsulating emails on behalf of the user to/from the email address of SWEET.

SWEET's un observ ability: We claim that a censor is note asily able to distinguish between SWEET's email messages and benign email messages. As described later in Section IV, a SWEET client has two options in choosing her email account: Alien-Mail a non-domestic email that encrypts emails (e.g., Gmail for users in China), and 2) Domestic-Mail a domestic email account with no need for encryption (e.g., 163.com for users in China). As described in Section IV, when Alien-Mails used by a client all of its SWEET emails are sent to a publicly known email address, e.g., tunnel @sweet.org, encrypted; however, a censor will not be able to identify these emails since they are proxied by the Alien-Mail server running outside the censoring area. In simpler words, the censor only observes that the client is exchanging encrypted messages with the Alien-Mail server (e.g., Gmail's mail server in U.S.), but he will not be able to observe neither the recipient's email address (tunnel@sweet.org), nor the IP address of the sweet.org mail server. As a result, existing approaches for spam filtering such as shooting the spamming SMTP servers and dropping spam emails are entirely infeasible. In the case of Domestic-Mail, the SWEET server uses a secondary secret email account, which is only shared with that particular client, for exchanging SWEET emails (i.e., my otheremail@163.cominstead oftunnel@sweet.org address). As a result, the censor will not be able to identify SWEET messages from their recipient fields (since the censor does not know the association of my other email @163.com with SWEET).Also the use of steganography/encryption to embed tunneled data renders DPI infeasible.

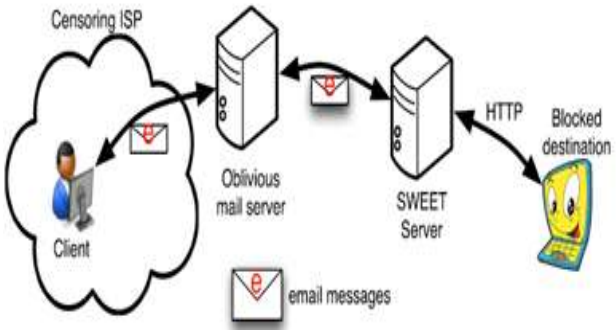

Fig.1. Overall architecture of SWEET.

SWEET's Availability: Given SWEET's unobserv ability discussed above, a censor cannot efficiently distinguish between SWEET emails and benign email messages. Hence, in order to block SWEET a censor needs to block all email messages to the outside world. However, email is an essential service in today's Internet and it is very unlikely that a censorship authority will block all email communications to the outside world, due to different financial and political reasons. This, along the fact that SWEET can be reached through a wide range of domestic/non-domestic email providers provides a high degree of availability for SWEET.

Prototype Implementation: We have built a prototype implementation for SWEET and evaluated its performance. We have also proposed and prototyped two different designs for SWEET client. The first client design uses email protocols, e.g., POP3 and SMTP, to communicate with the SWEET system, and our second design is based on using the web mail interface. Our measurements show that a SWEET client is able to browse regular-sized web destinations with download times in the order of couple of seconds. In fact, the high availability of SWEET comes for the price of higher, but bearable, communication latencies. Fig.2 compares SWEET with several popular circumvention systems regarding their availability and communication latency. As our measurements in Section VII show, SWEET provides communication latencies that are convenient for latency-sensitive activities like web browsing (i.e., few seconds). Such additional, tolerable latency of SWEET comes with the bonus of better availability, as discussed in Section V-B.

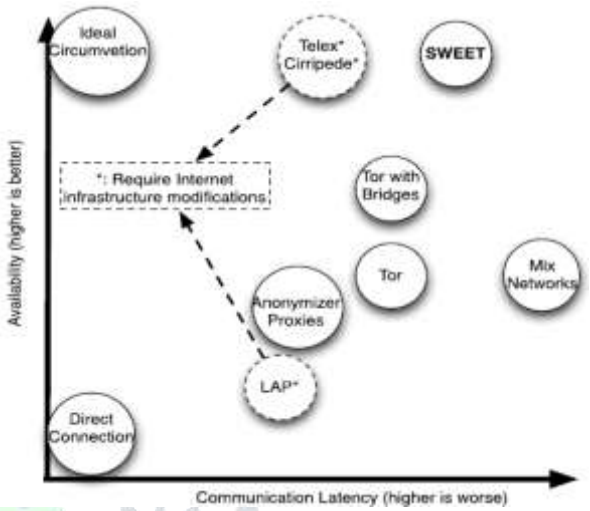

Fig.2. Availability and communication latency comparison of circumvention systems.

Our contributions: In summary, this paper makes the following main contributions: i) we propose a novel infrastructure for censorship circumvention, SWEET, which provides high availability, a feature missing in existing circumvention systems; ii) we develop two prototype implementations for SWEET(one using webmail and the other using email exchange protocols) that allow the use of nearly all email providers by SWEET clients; and, iii) we show the feasibility of SWEET for practical censorship circumvention by measuring the communication latency of SWEET for web browsing using our prototype implementation. Paper's organization: The rest of this paper is organized as follows; in Section II, we discuss the related work on unobservable censorship circumvention. In Section III, we reviews our threat model. We provide the detailed description of the proposed circumvention system, SWEET, in Section IV.

II. RELATED WORK

There has been much work on unobservable censorship circumvention systems [23], [24], [26]–[28], [30], [32]–[35].Similar to SWEET, Free-Wave [30], Cloud-Transport [32], and Covert-Cast [35] also work by tunneling circumvention traffic into the actual runs of popular network protocols. For instance, Free-Wave [30] tunnels Internet traffic inside VoIP communications. This tunneling approach provides much stronger unobservability against

the censors compared to imitation based circumvention systems [26]–[28], as demonstrated by Houmansadret al. [29].Several designs [19]–[21] seek unobservability by sharing secret information with their clients, which are not known to censors. For instance, the Tor network has recently adopted the use of Tor Bridges, a set of volunteer nodes connecting clients to the Tor network, whose IP addresses are selectively distributed among Tor users by Tor. As another example, In franet [19] shares a secret key and some secret URL addresses with a client, which is then used to establish an unobservable communication between the client and the system. Collage [20] works by having a client and the system secretly agree on some user-generated content sharing websites, e.g., flickr.com, and communicate using steganography. Unfortunately, sharing secrets with a wide range of clients is a serious challenge, as a censor can obtain the same secret information by pretending to be a client. Some recent research suggests circumvention being built into the Internet infrastructure to better provide unobservability [22]–[24]. These systems rely on collaboration from some Internet routers that intercept users' traffic to uncensored destinations to establish covert communication between the users and the censored destinations. Telex [23]and Cirripede [24] provide this unobservable communication without the need for some pre-shared secret information with the client, as the secret keys are also covertly communicated inside the network traffic. Cirripede [24] uses an additional client registration stage that provides some advantages and limitations as compared to Telex [23] and Decoy routing [22]systems. Recent studies investigate the real-world deployment of decoy routing systems by evaluating the placement of decoy routers on the Internet in adversarial settings [36]–[38].

III. DESIGN OF SWEET

In this section, we describe the detailed design of SWEET .Fig.1 shows the overall architecture. SWEET tunnels network connections between a client and a server, called SWEET server, inside email communications. Upon receiving the tunneled network packets, the SWEET server acts as a transparent proxy between the client and the network destinations requested by the client. A client's choices of email services: A SWEET client has two options for his email provider: Alien-Mail, and Domestic-Mail.

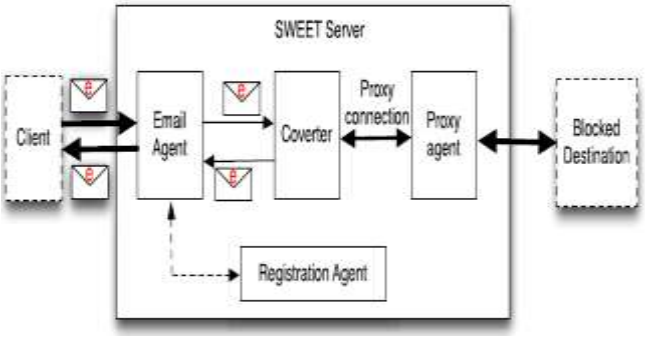

Fig. 3.The main architecture of SWEET server.

Alien-Mail: An Alien-Mail is a mail provider whose mail servers reside outside the censoring ISP, e.g., Gmail for the Chinese clients. We only consider Alien-Mails that provide email encryption, e.g., Gmail and Hush-mail. A SWEET client who uses an Alien-Mail does not need to apply any additional encryption/steganography to her encapsulated contents. Also, she simply sends her emails to the publicly advertised email address of SWEET server, e.g., tunnel@sweet.org, since the censors will not be able to observe (and block)the tunnel@sweet.org address inside SWEET messages, which are exchanged between the client and the Alien-Mail server in an encrypted format.

Domestic-Mail: A Domestic-Mail is an email provider hosted inside the censoring ISP and possibly collaborating with the censors, e.g., 163.com for the Chinese clients. Since the censors are able to observe the email contents, the SWEET client using a Domestic-Mail should hide the encapsulated contents through steganography (e.g., by doing image/text steganography inside email messages). Also, the client cannot send her SWEET emails to the public email address of SWEET server (tunnel@sweet.org) since the mail recipient field is observable to the Domestic-Mail provider and/or the censor. Instead, the client generates a secondary`email`address`myotheremail@somedomain.com(which could be either Domestic-Mail or Alien-Mail), and then provides the email credentials for this secondary account only to SWEET server through an out-of-band channel (e.g., through an online social network). The SWEET server uses this email address to exchange SWEET emails only with this particular client. In the following, we describe the details of SWEET's server and client architectures. To avoid confusion and without loss of generality, we only consider the case of Alien-Mail being used by the client. If Domestic-Mail is used, the client and server should also perform some steganography operations to hide the encapsulated traffic, as well as they should exchange a secondary email address, as described above.

A. SWEET Server

The SWEET server is the part of SWEET running outside the censoring region. It helps SWEET clients to evade censor ship by proxying their traffic to blocked destinations. More specifically, a SWEET server communicates with censored users by exchanging emails that carry tunneled network packets. Fig. 3 shows the main design of SWEET server, which is composed of the following elements:

- **Email Agent:** The email agent is an IMAP and SMTP server that receives emails that contain the tunneled Internet traffic, sent by SWEET clients to SWEET's email address. The email agent passes the received emails to another components of the SWEET server, the converter and the registration gent. The email agent also sends emails to SWEET clients, which are generated by other components of SWEET server and contain tunneled network packets or client registration information.
- **Converter:** The converter processes the emails passed by the email agent, and extracts the tunneled network packets. It then forwards the extracted data to another component, the proxy agent. Also, the converter receives network packets from the proxy agent and converts them into emails that are targeted to the email address of corresponding clients. The converter then passes these emails to the email agent for delivery to their intended recipients. As described later, the

converter encrypts/decrypts the email attachments of a user using a secret key shared with that user.

- **Proxy Agent:** The proxy agent proxies the network packets of clients that are extracted by the converter, and sends them to the Internet destination requested by the clients. It also sends packets from the destination back to the converter.
- **Registration Agent:** This component is in charge of registering the email addresses of the SWEET clients, prior to their use of SWEET. The information about the registered clients can be used to ensure quality of service and to prevent denial-of-service attacks on the server. Additionally, the registration agent shares a secret key with the client, which is used to encrypt the tunneled information between the client and the server.

B. SWEET Client

To use SWEET, a client needs to obtain a copy of SWEET's client software and install it on her machine. The client also needs to create one or two email account (depending on if she uses an Alien-Mail or a Domestic-Mail for her primary email).A client needs to configure the installed SWEET's software with information about her email account. Prior to the first use of SWEET by a client, the client software registers the email address of its user with the SWEET server and obtains a shared secret key kC, R, as described in Section IV-A. We propose two designs for SWEET client: a protocol-based design, which uses standard email protocols to exchange email with client's email provider, and a webmail-based design, which uses the webmail interface of the email provider. We describe these two designs in the following.1) Protocol-Based Design: Fig. 4(a) shows the three main elements.

Web Browser: The client can use any web browser that supports proxying of connections, e.g., Google Chrome, Internet Explorer, or Mozilla Firefox. The client needs to configure her browser to use a local proxy server, e.g., by settinglocalhost:4444 as the HTTP/SOCKS proxy. The client can use two different browsers for browsing with and without SWEET to avoid the need for frequent re-configurations of the browser. Alternatively, some browsers (e.g., Chrome, and Mozilla Firefox) allow a user to have multiple browsing profiles, hence, a user can setup two profiles for browsing with and without SWEET.

Email Agent: It sends and receives SWEET emails thorough the client's email account. The client needs to configure it with the settings of the SMTP and IMAP/POP3 servers of her email account. The client also needs to provide it with the account login information.

Converter: It sits between the web browser and the email agent, and converts SWEET emails into network packets and vice versa. It uses the keys shared with SWEET, kC, R, to encrypt/decrypt email content. Once the client enters a URL into the configured browser(1), the browser makes a proxy connection to the local port that the converter (3) is listening on. The converter accepts the proxy connection and keeps the state of the established TCP/IP connections. For packets that are received from the browser, the converter generates traffic emails, targeted to tunnel@sweet.org, having the received packets as encrypted email attachments (using the key kC,R). Such emails are passed to the email agent (2) that sends the emails to the SWEET server through the public email provider of the client. The email agent is also configured to receive emails from the client's email account through an email retrieval protocol, e.g., IMAP or POP3. This allows the email agent to continuously look for new emails from the server. Once new emails are received, the email agent passes them to the converter, who in turn extracts the packets from the emails, decrypts them, and sends them to the browser over the existing TCP/IP connection.

Webmail-Based Design: Alternatively, the SWEET client can use the webmail interface of the client's public email provider. as showed in Fig. 4(b). The main difference with the protocol-based design is that in this case the email agent (2)uses a web browser to exchange emails. More specifically, the email agent uses its web browser to open a webmail interface with the client's email account, using the user's authentication credentials for logging in. Through this HTTP/HTTPS connection, the email agent communicates with the SWEET server by sending and receiving emails.

C. The Choice of the Proxy Protocol

As mentioned before, the SWEET server uses a proxy agent to receive the tunneled traffic of clients and to establish connections to the requested destinations. We consider the useof both SOCKS [31] and HTTP [43] proxies in the design, as each provides unique advantages. Our server's proxy agent runs a SOCKS proxy and an HTTP proxy in parallel, each on a different port. A user can choose to use the type of proxy by configuring her client to connect to the corresponding port. The use of the SOCKS proxy allows the client to make any IP connection through the SWEET system, including dynamic web communications, such as Java script or AJAX, and instant messaging. In contrast, an HTTP proxy only allows access to HTTP destinations. However, an HTTP proxy may speedup connections by using HTTP-layer optimizations such as caching or pre-fetching of web objects.

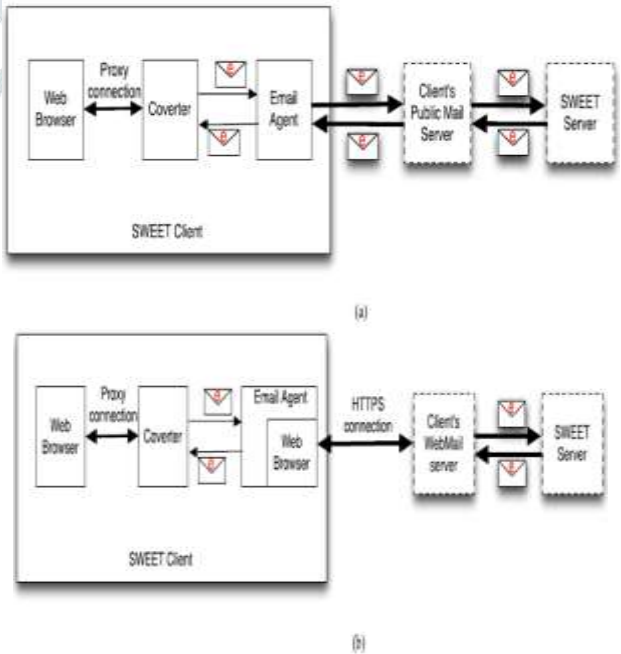

Fig.4. Design of SWEET client software. (a) The protocol-based design. (b) The webmail-based design.

D. An Alternative Approach: Web Download

A trivial approach in providing censorship circumvention using email is to download an entire webpage and attach it as an email attachment to emails that are targeted to the requesting users. In fact, this approach is under development by the open-source foe project [39], and the for-profit service of MailMy Web [40]. Unfortunately, this simple approach only provides a limited access to the Internet: a user can only access static websites. In particular, this approach cannot be used to access destinations that require end-to-end encryption, contain dynamic web applications like HTML5 and Java script sockets, or need user login information. Also, this approach does not support accessing web destinations that require a live Internet connection, e.g., video streaming websites, instant messaging, etc. In fact, the MailMy Web service uses some heuristics to tackle some of these shortcomings partially, which are privacy invasive and inefficient. For example, in order to access login based websites MailMyWeb requests a user to send her login credentials to MailMyWeb by email. Also, a user can request for videos hosted only on the YouTube video sharing website, which are then downloaded by MailMyWeb and sent as email attachments; this causes a large delay between the time a video is requested until it is has received by the user. SWEET, on the other hand, provides a comprehensive web browsing experience to its users since it can tunnel any kind of IP traffic.

IV. PROTOTYPE IMPLEMENTATION

A. Server Implementation

We implement the SWEET server on a Linux machine, which runs Ubuntu 10.04 LTS and has a 2 GHz quad-core CPU and 4 GB of memory. We run Postfix,1 a simple email server that supports basic functions. Postfix listens for new emails targeted to the register@sweet.org andtunnel@sweet.org email addresses. Postfix stores the received emails into designated file directories that are continuously watched by the converter and registration agent of SWEET server. Each stored email has a unique name, concatenating the email id of its corresponding client and an increasing counter.

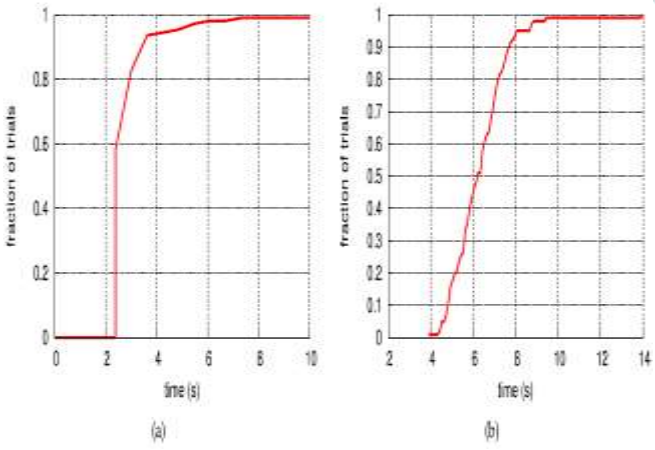

Fig.5. The CDF of (a) the time that a SWEET email takes to travel from the SWEET client to the SWEET server; (b) the registration time.

The converter agent is a simple Python based program that runs in the background and continuously checks the folder for new emails. The converter also converts proxied packets, passed by SWEET's proxy, into emails and sends them to their intended clients. For the proxy agent, we use Squid2 as our HTTP proxy and Suttree3 as our SOCKS proxy. Squid listens on a local port for connections from the converter.

B. Client Implementation

Protocol-Based Design: The client prototype is built on a desktop machine, running Linux Ubuntu 10.04 TLS. We setup a web browser to use the local port "local host: 9034" as the SOCKS/HTTP proxy. The converter is a simple python script that listens on port 9034 for connections, e.g., from our web browser. We implement the email agent of SWEET client using Fetchmail,4 a popular client software for sending and retrieval of emails through email protocols. We generate a free Gmail account and configure Fetch-mail to receive emails through IMAP5 and POP36 servers of Gmail, and to send emails through the SMTP server of Gmail.7 Note that our design does not rely on Gmail, and the client software can beset up with any email account. Webmail-based design: Our webmail-based implementation also runs on Linux Ubuntu 10.04 TLS, and uses the same converter as the one used in the protocol-based prototype. A Google Chrome browser is used for making connections through SWEET, configured to use "localhost:9034" as a proxy. We prototype the web-based email agent by running a UserScript8 inside the Mozilla Firefox9 browser. More specifically, we install a Firefox extension, Greasemonkey, 10to allow a user to run her own JavaScript, i.e., User script, while browsing certain destinations. We write a User Script that runs in Gmail's webmail interface and listens for the receipt of new emails. Our User Script saves new emails in a local directory, which is watched by the converter. Note that the Firefox browser is directly connected to the Internet and does not use any proxies (user needs to use the configured Chrome browser to surf the web through SWEET).

V. EVALUATION

We evaluate SWEET using our prototype implementation.

A. Performance

We use Gmail as the oblivious mail provider in our experiments. Our SWEET server is located in Urbana, IL, resulting in approximately 2000 miles of geographic distance between the SWEET server and Gmail's email server (we locate Gmail's location from its IP address). Fig. 5(a) shows the CDF of the time that a SWEET email (carrying the tunneled traffic) sent by a SWEET client takes to reach our SWEET server (the reverse path takes a similar time). As the figure shows, around 90% of emails take less than 3 seconds to reach the server, which is very promising considering the high data capacity of these emails. Note that based on our measurements, most of this delay comes from email handling(e.g, spam checks, making SMTP connections, etc.) performed by the oblivious mail provider (Gmail in our experiments),but not from the network latency (the network latency and client latency constitute only tens of milliseconds of the total latency). As a result,

the latency would be very similar for users with an even longer geographical distance from the oblivious mail server.

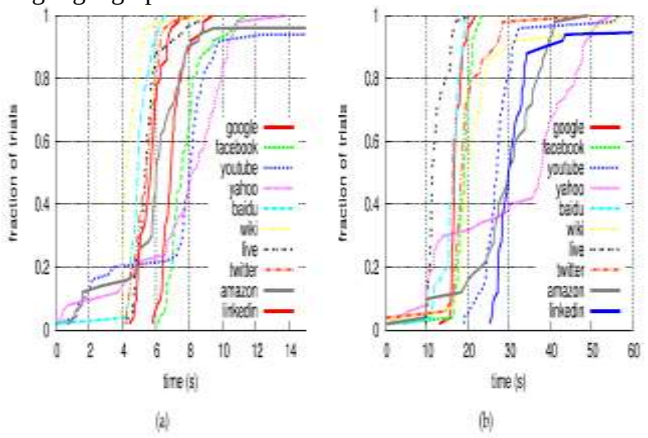

Fig.6. The CDF of (a) the time to the first appearance (TFA) and (b) the total browsing time (TBT) using SWEET.

Client Registration: Before being able to request data from Internet destinations, a user needs to be registered by the SWEET server. Fig. 5(b) shows the time taken to exchange registration messages between a client and the SWEET server. Note that the client registration needs to be performed only once for a long period of time. The figure shows that more than 90% of registrations establish in less than 8 seconds (with an average of 6.4 seconds).We use two metrics to evaluate the latency performance of SWEET in browsing websites: the time to the first appearance (TFA) and the total browsing time (TBT).The TFA is the time taken to receive the first response from a requested web destination. It is an important metric in measuring user convenience during web browsing. For instance, suppose that a client requests a URL, e.g., By the TFA time the client receives the first HTTP RESPONSE(s) from the destination, which include the URL's text parts (perhaps the new sarticle) along with the URLs of other objects on that page, e.g., images, ads hosted by other websites, etc. At this time the client can start reading the received portion of the website(e.g., the news article), while her browser sends requests for other objects on that webpage. On the other hand, the total browsing time (TBT) is the time after which the browser finishes fetching all of the objects in the requested URL. Using our prototype we measure the end-to-end web browsing latency for the client to reach different web destinations. Fig.6(a) shows the TFA for the top 10 web URLs from Alexa's most-visited sites ranking [46]. The median is about5 seconds across all experiments, which is very promising to user convenience. On the other hand, Fig. 6(b) shows the total browsing time (TBT) for the same set of destinations (50 runs for each web site). As can be seen, the destinations that contain more web objects (e.g., yahoo and linked in) take more time to get completely fetched (note that after the TFA time the user can start reading the webpage). We also run similar experiments through the popular Tor [45] anonymous network to compare its latency performance with SWEET. Fig. 7 compares the latency CDF for SWEET and Tor. As expected, our simple implementation of SWEET takes more time than Tor to browse web pages, however, it provides a sufficient performance for normal web browsing. This is in particular significant considering the strong availability of SWEET compared to other circumvention systems. Additionally, we believe that further optimizations on SWEET server's proxy (like those implemented by Tor exit nodes) will further improve the performance.

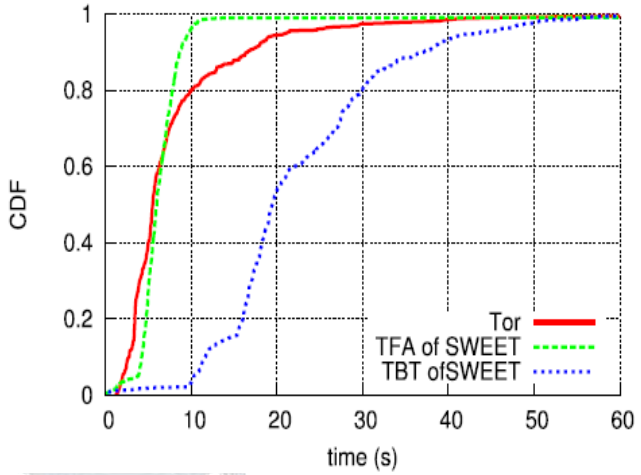

Fig.7. Comparing the average latency of SWEET and Tor.

Our techniques are also amenable to standard methods to improve web latency, such as plug in-based caching and compression, which can make web browsing tolerable in high delay environments [47].

B. Traffic Analysis

A powerful censor can perform traffic analysis to detect the use of SWEET, e.g., by comparing a user's email communications with that of a typical email user. As a result, a SWEET user who is concerned about unobservability needs to ensure that her SWEET email communications mimic that of a normal user (a user who does not fear reprisal from her government might opt to have lower unobservability in order to gain a higher communication bandwidth). It should be mentioned that such traffic analysis is expensive for censors considering the large volume of email communications; it is estimated11 that294 billion emails were sent per day in 2011.Fig. 8 shows the number of emails sent and received by a SWEET client to browse different websites. We observe that for any particular website the number of emails does not change at different runs. As can be seen, most of the web sites finish in less than three SWEET emails in each direction. The exception is the Yahoo web page as it contains many web objects, each hosted by different URLs (note that the number of emails affects the latency performance only sub-linearly, since some emails are sent and received simultaneously.). Also, the average number in each way of a connection is about 4 emails. A recent study [48] on email statistics predicts that an average user will send 35 emails and will receive75 emails per day in 2012 (the study predicts the numbers to increase annually).

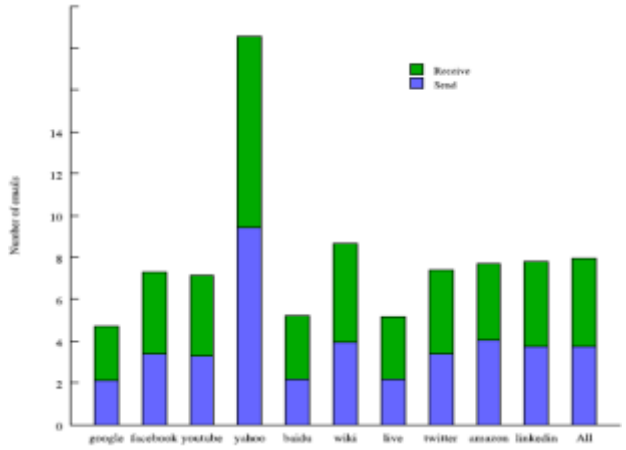

Fig.8. The number of emails sent and received by a SWEET client to get one of the websites from Alexa's top ten ranking.

In addition, membership in mailinglists12 and Internet groups13,14 is popular among Internet users, producing even more emails by normal email users. As an indication of the popularity of such services, Yahoo in 2010announced15 that 115 million unique users are collectively members of more than 10 million Yahoo Groups. Based on the mentioned statistics, we estimate that a conservative SWEET user can perform 35-70 web downloads per day, or make10-20 interactive web connections, while ensuring unobservability of SWEET usage. Note that the censored users use SWEET only to browse "censored" Internet web pages and they use regular web browsing for non-censored websites. Also, normal citizens who do not fear being caught by the censors may decide to ignore resistance against traffic analysis in order to achieve higher bandwidths. The censors may also try to detect SWEET users by analyzing the inter-arrival times of the email messages exchanged between SWEET users and their mail service providers. More specifically, a SWEET client may send and receive multiple emails in a shorter time interval compared to regular email clients. We, however, argue that this is nota serious vulnerability for SWEET. As discussed earlier, we require SWEET clients to use mail service providers that encrypt email exchanges (otherwise, the censors can simply filter SWEET emails by searching for the email addresses of SWEET servers). The use of encryption prevents the censors from identifying the number of emails exchanged between a SWEET user and her mail service provider.

VI. RESULTS

Results of this is paper is as shown in bellow Figs. 9 to 16.

Fig.9. Registration Form.

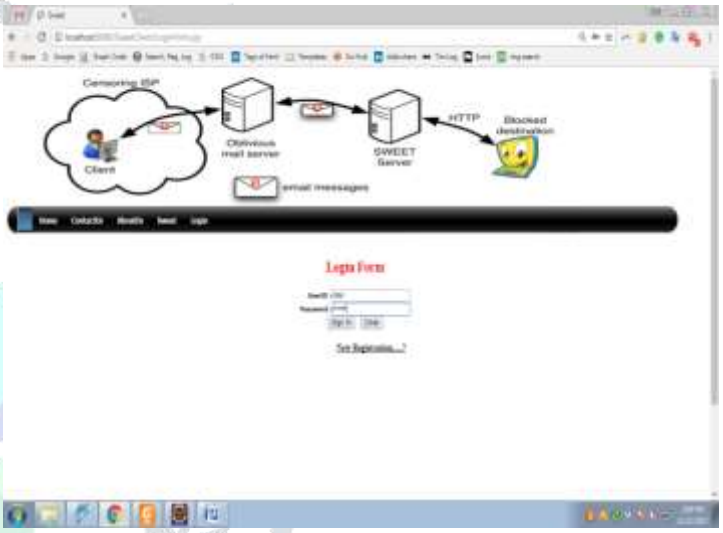

Fig.10. Login Form.

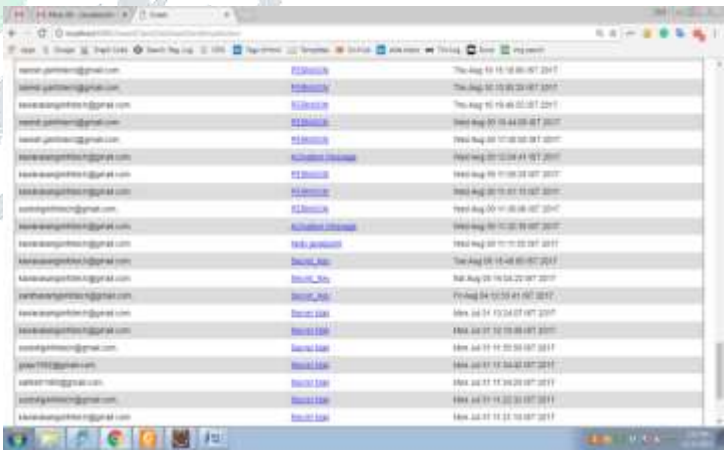

Fig.11. User Sent Mails On Sweet Server.

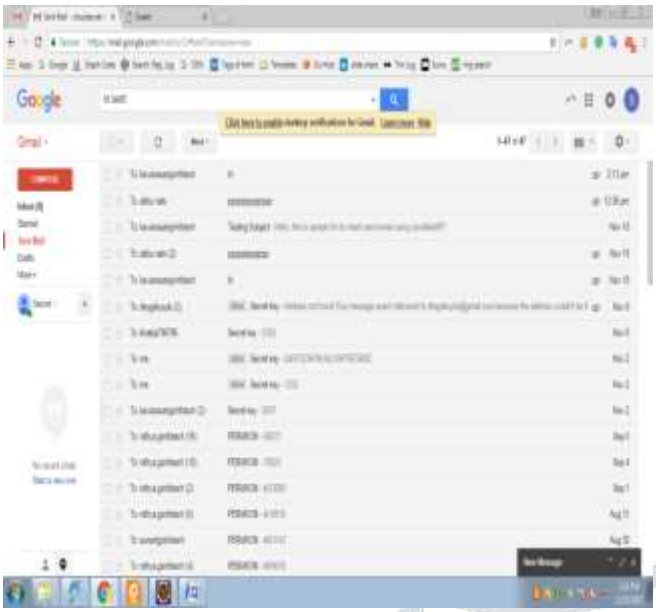

Fig.12. User Sent Mails On Gmail.

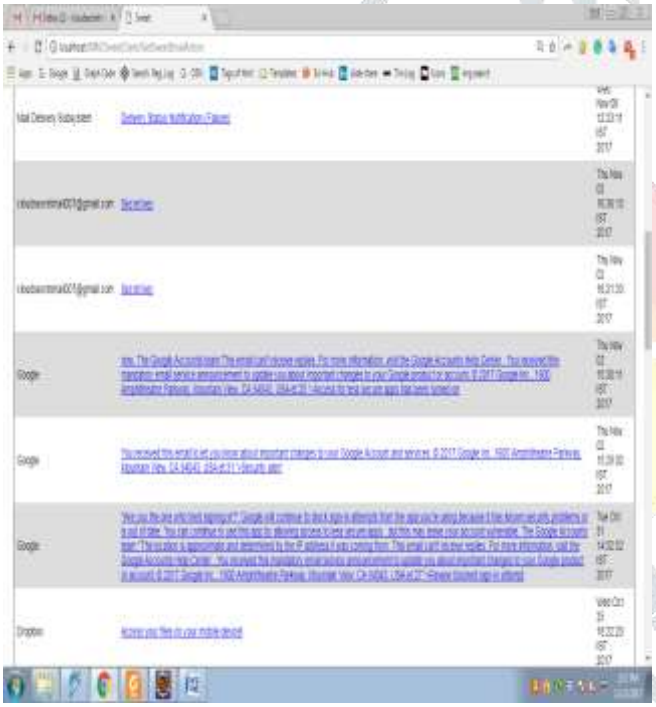

Fig.13. User Inbox On Sweet Server.

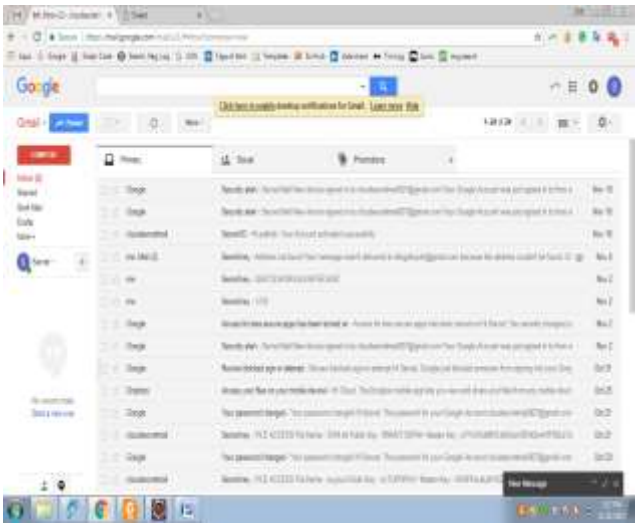

Fig.14. User Inbox On Gmail.

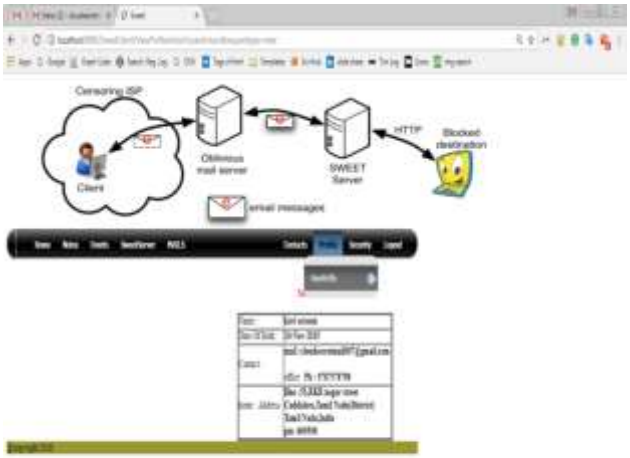

Fig.15. Profile Form.

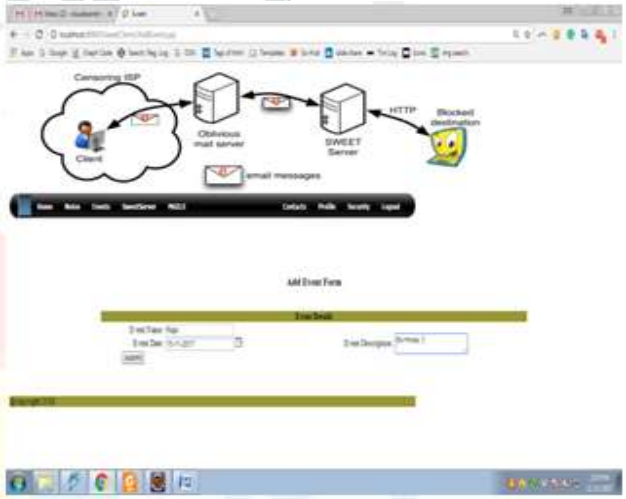

Fig.16. Add Event Form.

VII. CONCLUSION

A powerful censor can perform traffic analysis to detect the use of SWEET, e.g., by comparing a user's email communications with that of a typical email user. As a result, a SWEET user who is concerned about unobservability needs to ensure that her SWEET email communications mimic that of a normal user. It should be mentioned that such traffic analysis is expensive for censors considering the large volume of email communications; it is estimated13 that 294 billion email messages were sent per day in 2011. We observe that for any particular website the number of emails does not change at different runs. As can be seen, most of the web sites finish in less than three SWEET emails in each direction. The exception is the Yahoo web page as it contains many web objects, each hosted by different URLs SWEET, a deployable system for unobservable communication with Internet destinations. SWEET works by tunneling network traffic through widely used public email services such as Gmail, Yahoo Mail, and Hotmail. Unlike recently-proposed schemes that require a collection of ISPs to instrument router-level modifications in support of covert communications, our approach can be

deployed through a small applet running at the user's end host, and a remote email-based proxy, simplifying deployment. Through an implementation and evaluation in a wide-area deployment, we find that while SWEET incurs some additional latency in communications, these overheads are low enough to be used for interactive accesses to web services. We feel our work may serve to accelerate deployment of censorship-resistant services in the wide area, guaranteeing high availability.

VIII. REFERENCES

[1] J. Zittrain and B. Edelman, "Internet filtering in China," IEEE InternetCompute., vol. 7, no. 2, pp. 70–77, Mar. 2003.

[2] (Nov. 2007). Defeat Internet Censorship: Overview ofAdvanced Technologies and Products. [Online]. Available: http://www.internetfreedom.org/archive/DefeatInternet Censorship WhitePaper.pd

[3] C. S. Leberknight, M. Chiang, H. V. Poor, and F. Wong.(2010).A Taxonomy of Internet Censorship and Anti-Censorship. [Online].Available: http://www.princeton.edu/ chiangm/anticensorship.pdf

[4] I. Clarke, S. G. Miller, T. W. Hong, O. Sandberg, and B. Wiley,"Protecting free expression online with freenet," IEEE Internet Compute.,vol. 6, no. 1, pp. 40–49, Jan. 2002.

[5] Ultrasurf, accessed on Jan. 7, 2017. [Online]. Available:https://ultrasurf.us/

[6] J. Jia and P. Smith. (2004). Psiphon: Analysis and Estimation.[Online]. Available: http://www.cdf.toronto.edu/ csc494h/reports/2004-fall/psiphon_ae.html

[7] I. Cooper and J. Dilley, "Known HTTP proxy/caching problems," IETF,Fremont, CA, USA, Tech. Rep. Internet RFC 3143, Jun. 2001.

[8] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The secondgenerationonion router," in Proc. USENIX Secur. Symp., 2004,pp. 21–37.

[9] J. Boyan, "The anonymizer: Protecting user privacy on the Web,"Comput.-Mediated Commun. Mag., vol. 4, no. 9, pp. 1–6, Sep. 1997.

[10] DynaWeb, accessed on Jan. 7, 2017. [Online].Available:http://www.dongtaiwang.com/home_en. php

[11] R. Clayton, S. J. Murdoch, and R. N. M. Watson, "Ignoring the greatfirewall of China," in Proc. Int. Workshop Privacy Enhancing Technol.,2006, pp. 20–35.

[12] Y. Sovran, A. Libonati, and J. Li, "Pass it on: Social networks stymiecensors," in Proc. 7th Int. Conf. Peer-to-Peer Syst., Feb. 2008, p. 3.[Online]. Available: http://www.iptps.org/papers-2008/73.pdf

[13] D. McCoy, J. A. Morales, and K. Levchenko, "Proximax: A measurementbased system for proxies dissemination," Financial Cryptogr. DataSecur., vol. 5, no. 9, pp. 1–10, 2011.

[14] N. Feamster, M. Balazinska, W. Wang, H. Balakrishnan, and D. Karger,"Thwarting Web censorship with untrusted messenger discovery," in Int.Workshop Privacy Enhancing Technol., 2003, pp. 125–140.

[15] M. Mahdian, "Fighting censorship with algorithms," in Proc. Int.Conf. Fun Algorithms, 2010, pp. 296–306. [Online]. Available:http://dx.doi.org/10.1007/978-3-642-13122-6_29

[16] J. McLachlan and N. Hopper, "On the risks of serving whenever yousurf: Vulnerabilities in Tor's blocking resistance design," in Proc. 8thACM Workshop Privacy Electron. Soc., Nov. 2009, pp. 31–40. [Online].Available: http://portal.acm.org/citation.cfm?doid=1655188.1655193

[17] P. Winter and S. Lindskog. Apr. 2012. "How China is blocking Tor."[Online]. Available: https://arxiv.org/abs/ 1204.0447

[18] (Sep. 2007). Tor Partially Blocked in China. [Online]. Available:https://blog.torproject.org/blog/tor-partially-blocked-china[19] N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, andD. Karger, "Infranet: Circumventing Web censorship and surveillance,"in Proc. 11th USENIX Secur. Symp., Aug. 2002, pp. 247–262.[Online]. Available: http://www.usenix.org/events/sec02/feamster.html

[20] S. Burnett, N. Feamster, and S. Vempala, "Chipping awayat censorship firewalls with user-generated content," in Proc.USENIX Secur.Symp., 2010, pp. 463–468. [Online]. Available:http://www.usenix.org/events/sec10/tech/full_papers/Burnett.pdf

[21] R. Dingledine and N. Mathewson, "Design of a blocking-resistantanonymity system," Tor Project, Tech. Rep. 1, Nov. 2006.

[22] J. Karlinet al., "Decoy routing : Toward unblockable Internet communication,"inProc. USENIX (FOCI), 2011, pp. 1–6.

[23] E. Wustrow, S. Wolchok, I. Goldberg, and J. A. Halderman, "Telex:Anticensorship in the network infrastructure," in Proc. 20th USENIXSecur. Symp., Aug. 2011, pp. 1–15.

[24] A. Houmansadr, G. T. K. Nguyen, M. Caesar, and N. Borisov, "Cirripede:Circumvention infrastructure using router redirection with plausibledeniability categories and subject descriptors," in ACM Conf.Compute. Communication.Security.(CCS), Chicago, IL, USA, 2011, pp. 187–200.

[25] H.-C. Hsiao et al., "LAP: Lightweight anonymity and privacy," in Proc.IEEE Symp. Secur. Privacy, May 2012, pp. 506–520.

[26] H. M. Moghaddam, B. Li, M. Derakhshani, and I. Goldberg, "Skypemorph:
Protocol obfuscation for Tor bridges," in Proc. ACM Conf.Compute.Communication.Security.(CCS), 2012, pp. 97–108.

[27] Q. Wang, X. Gong, G. Nguyen, A. Houmansadr, and N. Borisov,"Censor Spoofer: Asymmetric communication using IP spoofingfor censorship-resistant Web browsing," in Proc. ACM Conf.Comput. Commun.Secur.(CCS), 2012, pp. 121–132.

[28] Z. Weinberg et al., "Stego Torus: A camouflage proxy for theTor anonymity system," in Proc. ACM Conf. Compute. Communication Security. (CCS), 2012, pp. 109–120.

[29] A. Houmansadr, C. Brubaker, and V. Shmatikov, "The parrot is dead:Observing unobservable network communications," in Proc. IEEE Symp.Secur. Privacy, May 2013, pp. 65–79.

Author's Profile:

Ms. HASEEBA HALEEM has completed her B.Tech (IT) from Shadan Women's College of Engineering and Technology, Khairtabad, JNTU University Hyderabad. Presently, she is pursuing her M.Tech in Computer Science from Shadan Women's College of Engineering and Technology, Hyderabad, TS. India.

Mr. Dr. S. SATHISH KUMAR has completed B.E. (CSE) from KSR College of Engineering, Anna University, Tiruchengodu, Chennai. M.E (CSE) from Sri Krishna Engineering College, Anna University, Chennai.PhD from Annamalai University Chidambaram. Currently he is working as an Associate Professor of CSE Department in Shadan Women's College of Engineering and Technology, Hyderabad, TS. India. He is having 5 years of experience in teaching and research.